В.Г.Хорошевский

Архитектура вычислительных систем

Издание второе, переработанное и дополненное

Допущено Министерством образования и науки
Российской Федерации
в качестве учебного пособия для студентов
высших учебных заведений,
обучающихся по направлению
«Информатика и вычислительная техника»

Москва
Издательство МГТУ имени Н .Э. Баумана
2008

УДК 681.32:519.68(075.8)
ББК 22.18
Х79

Рецензенты:
кафедра «Информатика и вычислительная техника»
Омского государственного технического университета
(д-р техн. наук, профессор *В.И. Потапов*);
д-р техн. наук, профессор *В.В. Сюзев*
(зав. кафедрой «Компьютерные системы и сети» Московского
государственного технического университета им. Н.Э. Баумана)

Хорошевский В.Г.

Х79 Архитектура вычислительных систем: Учеб. пособие. — 2-е изд., перераб. и доп. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2008. — 520 с.: ил. — (Информатика в техническом университете).

ISBN 978-5-7038-3175-5

Рассмотрены основные архитектурные концепции построения средств обработки информации, модели функциональной организации электронных вычислительных машин (ЭВМ Дж. фон Неймана, модель вычислителя) и параллельных вычислительных систем (модель коллектива вычислителей). Описаны канонические функциональные структуры и наиболее интересные промышленные реализации конвейерных, матричных, мультипроцессорных и распределенных вычислительных систем. Особое внимание уделено архитектурно наиболее совершенному классу ВС — системам с программируемой структурой.

Приведены инженерные методы комплексного анализа производительности, надежности, живучести и технико-экономической эффективности, а также нетрудоемкая технология экспресс-анализа качества функционирования ЭВМ и ВС и осуществимости параллельного решения сложных задач.

Второе издание (1-е — 2005 г.).

Для студентов вузов, а также для специалистов в области параллельных вычислительных технологий.

УДК 681.32:519.68(075.8)
ББК 22.18

ISBN 978-5-7038-3175-5

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Системы распределенной обработки информации и параллельные вычислительные технологии относятся к базовым средствам XXI столетия, обеспечивающим интенсификацию научно-технического прогресса.

Предлагаемое учебное пособие посвящено современным и перспективным архитектурным концепциям средств обработки информации. Из 12 глав учебника только две первые не связаны с параллельными технологиями вычислений. В главе 1 изучается предыстория вычислительной техники, а в главе 2 — архитектура электронных вычислительных машин (ЭВМ), или неймановская архитектура средств обработки информации. При описании архитектуры ЭВМ особое внимание уделено концептуальной модели вычислителя и семантике понятия «архитектура ЭВМ». Архитектура ЭВМ раскрывается через их функциональные структуры и такие характеристики эффективности, как производительность, надежность и технико-экономическая эффективность. При этом излагается математический инструментарий для анализа эффективности ЭВМ. Содержание первых двух глав позволяет также глубже понять архитектуру параллельных средств вычислительной техники. Глава 2 является введением для последующих глав, в которых изучаются методы анализа эффективности функционирования параллельных вычислительных систем.

Главы 3–8 посвящены средствам обработки информации, основанным на модели коллектива вычислителей. Такие средства называют вычислительными системами (ВС), и они характеризуются массовым параллелизмом при обработке информации и реконфигурируемостью (программируемостью) своих структур. В главе 3 изложены концептуальные основы предмета «Архитектура ВС», включающие принципы технической реализации модели коллектива вычислителей, элементарные понятия параллельного программирования и методику крупноблочного распараллеливания сложных задач.

В главах 4–8 дано достаточно полное описание таких классов вычислительных систем, как конвейерные, матричные, мультипроцессорные ВС, системы с программируемой структурой и транспьютерные ВС. Здесь представлены не только канонические функциональные структуры и промышленные реализации названных классов систем, но приведен качественный анализ их архитектурных возможностей и показаны пути их развития.

В последних главах рассмотрены инженерные методы анализа эффективности функционирования вычислительных систем. В главе 9 описаны ВС со структурной избыточностью, введен набор показателей надежности ВС, приведены методы расчета этих показателей и в переходном, и в стационарном режимах работы ВС. В главе 10 исследована живучесть ВС как ансамблей элементарных машин, т. е. способность ВС использовать все исправные ресурсы (машины) для реализации адаптирующихся параллельных программ. Просто и лаконично изложен континуальный подход к расчету показателей живучести большемасштабных ВС (с массовым параллелизмом). Глава 11 посвящена вероятностной теории осуществимости параллельного решения задач на неабсолютно надежных вычислительных системах. Наконец, глава 12 знакомит с математически простым аппаратом анализа технико-экономической эффективности функционирования вычислительных систем.

В Приложении 1 приведен математический аппарат расчета вероятностных характеристик надежности для переходного режима функционирования ВС. В Приложении 2 описан инструментарий экспресс-анализа эффективности функционирования большемасштабных вычислительных систем.

Структура пособия и изложение материала позволяют достаточно полно раскрыть данную тему, не отсылая читателя к другим источникам.

Каждая из глав автономна, и для понимания их содержания не требуется тщательное знание предшествующих глав. Если у читателя все же возникнут затруднения, он легко их преодолеет, воспользовавшись ссылкой на предшествующий материал.

Большой опыт работы профессором (более 35 лет), чтение соответствующих курсов лекций в Новосибирском государственном техническом университете (1969–1983), Сибирском государственном университете телекоммуникаций и информатики (с 1983 г.) и Новосибирском государственном университете (1995–2002), опыт написания ряда учебных пособий и книг, используемых в вузах Содружества независимых государств, — все это помогло автору издать настоящее пособие. В нем освещены новейшие достижения в области архитектур и организации функционирования параллельных ВС, включающие результаты научной школы автора.

Учебное пособие «Архитектура вычислительных систем» соответствует государственным образовательным стандартам высшего профессионального образования по направлениям подготовки: бакалавра (магистра) 552800 «Информатика и вычислительная техника»; дипломированного специалиста 654600 «Информатика и вычислительная техника».

Автор с благодарностью учтет замечания и примет советы по улучшению книги (105005, Москва, 2-я Бауманская ул., д. 5, Издательство МГТУ им. Н.Э. Баумана).

СПИСОК ОСНОВНЫХ СОКРАЩЕНИЙ

АЛУ — арифметико-логическое устройство
АУ — арифметическое устройство
БИС — большая интегральная схема
БОС — блок операций системы
ВЗУ — внешнее запоминающее устройство
ВМ — вычислительная машина
ВС — вычислительная система
ВТ — вычислительная техника
ЗУ — запоминающее устройство
КОП — код операции
ЛК — локальный коммутатор
МКМД — множественный поток команд и множественный поток данных
МКОД — множественный поток команд и одиночный поток данных
МП — микропроцессор
ОБП — обобщенный безусловный переход
ОЗУ — оперативное запоминающее устройство
ОКМД — одиночный поток команд и множественный поток данных
ОКОД — одиночный поток команд и одиночный поток данных
ОС — операционная система
ОУП — обобщенный условный переход
ПО — программное обеспечение
РН — регистр настройки
САПР — система автоматизированного проектирования
СУ — системное устройство
ЦП — центральный процессор
ЭВМ —электронная вычислительная машина
ЭМ — элементарная машина
ЭП — элементарный процессор

1. ПРЕДЫСТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Электронные вычислительные средства завоевали прочные позиции в жизненно важных сферах деятельности человека — в науке, технике, экономике и промышленности. Область применения средств обработки информации постоянно расширяется. Возможности вычислительной индустрии в существенной мере определяют научно-технический прогресс. Потребности общества в решении все более сложных задач постоянно растут, и они, в свою очередь, стимулируют развитие вычислительных средств.

Современная индустрия информатики располагает арсеналом средств от персональных электронных вычислительных машин (ЭВМ) до высокопроизводительных вычислительных систем (ВС) с массовым параллелизмом, обладающих быстродействием в пределах $10^{10}...10^{15}$ опер./с.

Вычислительная техника развивалась постепенно, в несколько этапов. В главе рассматриваются средства вычислительной техники начального этапа (простейшие механические вычислительные приборы, электромеханические машины и ЭВМ); описываются архитектурные особенности первых зарубежных и отечественных ЭВМ: ENIAC, EDVAC, МЭСМ, БЭСМ и др.

1.1. Эволюция вычислительной техники

Развитие человека и общества неразрывно связано с прогрессом в технике вообще и вычислительной технике в частности. Всегда существовала тенденция к постоянному усилению физических и вычислительных возможностей человека путем создания орудий, машин и систем машин. Установился своеобразный *дуализм в развитии техники*, который иллюстрируется двумя эволюционными «рядами»:

- *физический ряд*: рычаг и простейшие механические орудия → машины (подъемные краны и экскаваторы) → конвейеры (системы машин);
- *вычислительный ряд*: простейшие счетные орудия (или приборы, или инструменты) → механические и электромеханические вычислительные

машины → ЭВМ → параллельные вычислительные системы и сети ЭВМ (системы ЭВМ).

В последние десятилетия появилось и развивается физико-вычислительное направление в технике как композиция физического и вычислительного рядов:

станки с числовым программным управлением → роботы → гибкие автоматизированные производства → заводы-автоматы.

В истории вычислительной техники (ВТ) четко выделяются два периода:

1) простейшие механические и электромеханические приборы и машины для вычислений;

2) ЭВМ и параллельные вычислительные системы.

Первый период — это *предыстория современной вычислительной техники*, его называют также древней историей ВТ. Второй период — это *новая и новейшая история* ВТ.

Концептуальную основу ЭВМ составляют функциональная структура и принципы работы, предложенные в 1945 г. Дж. фон Нейманом. Первые ЭВМ были последовательными, машины второго и третьего поколений допускали совмещения во времени выполнения нескольких операций. Вычислительные системы базируются, в частности, на принципе параллелизма при обработке информации. И ЭВМ, и ВС постоянно совершенствуются. Современные ЭВМ основываются на архитектурных решениях, существенно отличающихся от машины Дж. фон Неймана, и значительно превосходят по своим возможностям высокопроизводительные конвейерные ВС 70-х и 80-х годов XX столетия. Развитие ВС осуществляется в направлении массового параллелизма и программируемости структуры (позволяющей адаптировать ВС под структуры решаемых задач).

Итак, для современного этапа развития ВТ характерен дуализм: ЭВМ и параллельная ВС. Параллельные ВС являются предметом особого рассмотрения в данной книге.

1.1.1. Простейшие вычислительные инструменты

Абак (греч. abax, abakion, лат. abacus — доска) — счетная доска для арифметических расчетов — открывает (754/753 г. до н. э.) первый период вычислительной техники. В качестве «носителя информации» использовались счетные марки (камешки, кусочки кости, монеты), распределенные по полосам доски. Счет осуществлялся передвижением марок в полосах. Абак применялся в Древней Греции и Риме, а затем в Западной Европе до XVIII в. Аналоги абака — китайский суан-пан и *счеты* были распространены в странах Дальнего Востока и в России соответственно.

Логарифмическая (счетная) линейка была изобретена в XVII в. В ее основе лежит учение о логарифмах, созданное шотландским математиком Д. Непером (J. Napier, 1550–1617) не позднее 1594 г. В данном счетном инструменте операции над числами заменяются действиями над логарифмами этих чисел; в качестве «носителя информации» служит отрезок прямой (отрезки с логарифмическими шкалами нанесены на корпус и передвигающийся в нем движок). Принцип работы логарифмической линейки очевиден:

$$\lg A \cdot B = \lg A + \lg B.$$

При помощи линейки реализуются следующие операции: умножение, деление, извлечение квадратного корня, возведение в степень и тригонометрические функции. Логарифмическая линейка на протяжении трех столетий была основным счетным инструментом инженеров и исследователей.

Таким образом, абак и логарифмическая линейка — первые простейшие счетные *инструменты* соответственно *дискретного и непрерывного действия*. Дуализм имел место на протяжении всей истории ВТ, однако дискретные (цифровые) средства обработки информации всегда занимали доминирующее положение (в сравнении с непрерывными или аналоговыми). Это объясняется тем, что дискретные средства ВТ обеспечивают бóльшую точность вычислений, чем непрерывные. Рассмотрим подробнее цифровые вычислительные машины (ВМ) и простейшие механические машины для арифметических расчетов.

1.1.2. Арифмометры

Вычислительный инструментарий постоянно совершенствовался; отметим лишь некоторые из разработок счетных машин. Описание первой цифровой механической счетной машины было сделано итальянским живописцем, скульптором, архитектором, ученым и инженером Леонардо да Винчи (Leonardo da Vinci, 1452–1519). Суммирующая машина (с переносом десятков) была изобретена в 1641 г. французским математиком, физиком, философом и писателем Б. Паскалем (Pascal, 1623–1662). Счетная машина, рассчитанная на четыре арифметических действия, была разработана в 1673 г. немецким математиком, физиком, философом и языковедом Г.В. Лейбницем (Leibniz, 1646–1716). Данные машины практического применения не нашли (хотя, например, Паскалем было изготовлено несколько машин). Механические машины Паскаля и Лейбница представляли собой первые арифмометры.

Арифмометр (от греч. arithmos — число и метр) — настольная механическая счетная машина с ручным управлением для выполнения четырех арифметических действий.

Широкое распространение имел арифмометр, сконструированный в 1874 г. петербургским механиком В.Т. Однером. Производство таких арифмометров было налажено и в России (1890 г.), и за рубежом. Арифмометр Однера послужил прототипом последующих моделей (в частности, для модели «Феликс», выпускавшейся в СССР до 1960-х годов).

Арифмометр представляет собой систему счетных колес. Счетные колеса использовались не только в качестве носителя информации, но и для ее преобразования. По окружности счетного колеса были нанесены однозначные числа от 0 до 9. При представлении многозначного числа для каждого разряда использовалось свое колесо. Система счетных колес имела устройство для передачи десятков, т. е. устройство, благодаря которому полный оборот колеса одного разряда влек за собой поворот на «единичный» угол (36°) колеса следующего старшего разряда. Такую систему колес называли *счетчиком*. Счетчик являлся одним из основных механизмов арифмометра. В состав арифмометра входили также регистры — узлы памяти для хранения многоразрядных чисел, механизм для установки чисел (для занесения чисел на регистры), устройство для гашения (сброса) результата и привод (ручной или электрический).

Оригинальная конструкция арифмометра принадлежит русскому математику и механику П.Л. Чебышеву (произносится Чебышёв, 1821–1894; академик Петербургской АН с 1856 г.). В данной конструкции была достигнута максимальная механизация выполнения всех арифметических действий. Арифмометр Чебышева состоял из двух основных частей: суммирующей машины (сконструированной в 1878 г.) и приставки для умножения (созданной в 1883 г.). В частности, в данном арифмометре после установки множимого и множителя подлежало только вращать рукоятку привода. Конструкция Чебышева была использована позднее в английских и американских арифмометрах.

Следует подчеркнуть, что любой *арифмометр обеспечивал не автоматизацию, а лишь механизацию вычислений* (благодаря таким средствам, как счетчик и регистры). Ввод данных, реализация алгоритма вычислений (как последовательности операций) и, как правило, даже одной арифметической операции, съем (вывод) результатов осуществлялись человеком (вычислителем).

Эволюционное развитие арифмометров привело к созданию клавишных вычислительных машин, а со сменой механической элементной базы на электронную — к производству электронных калькуляторов.

Счетно-аналитические машины появились в конце XIX – начале XX в. Были созданы вычислительные машины (ВМ) для выполнения бухгалтерских и финансово-банковских операций, статистические ВМ, машины для решения задач вычислительной математики. В таких машинах не только

был достигнут максимальный уровень механизации вычислений, но и была заложена возможность автоматизации при вводе чисел и при реализации целых серий операций. Характерным для счетно-аналитических машин было то, что в них перфокарты использовались как для ввода данных, так и для управления работой.

Счетно-аналитические машины — это комплекты, включавшие:

1) машины для выполнения арифметических действий над числами, нанесенными на перфокарты:

– суммирующие машины (табуляторы);

– множительные машины (умножающие перфораторы или мультиплееры);

2) машины (сортировальные и раскладочные или сортировально-раскладочные) для реализации информационно-логических операций: классификации, выборки карт с нужными числами и признаками, расположения карт в определенном порядке, сравнение чисел и т. п.;

3) перфораторы, т. е. машины, которые позволяли человеку наносить на карты отверстия (выполнять перфорирование карт);

4) вспомогательные машины; например, контрольные аппараты, репродукторы для переноса пробивок с одних карт на другие.

Первая ВМ для решения дифференциальных уравнений была создана в России в 1904 г. кораблестроителем, механиком и математиком А.Н. Крыловым (1863–1945; академик Петербургской АН с 1916 г.).

1.1.3. Вычислительная машина Ч. Беббеджа

Идея создания универсальной большой ВМ (Great Calculating Engine) принадлежит профессору математики Кембриджского университета (Великобритания), члену Лондонского Королевского Общества Чарльзу Беббеджу (Charles Babbage, 1792–1871; чл.-корр. Петербургской АН с 1832 г.). По сути, он хотел создать *автоматический* механический цифровой компьютер (или, говоря иначе, арифмометр с программным управлением). Проект ВМ был разработан в 1833 г.

Механическая машина Беббеджа по своей функциональной структуре была достаточно близка к первым электронным ВМ. В ней предусматривались арифметическое и запоминающие устройства, устройства управления и ввода-вывода информации. Автоматизация вычислений обеспечивалась устройством управления, которое работало в соответствии с программой — последовательностью закодированных действий на перфокартах. В машине Беббеджа была заложена возможность изменять ход программы в зависимости от полученного результата (на современном языке — команда условного перехода).

Машина должна была состоять из нескольких тысяч счетных колес, иметь запоминающее устройство емкостью 1000 50-разрядных чисел и встроенные таблицы логарифмов и других элементарных функций. Для размещения машины потребовалась бы площадь в несколько квадратных метров.

В 1835 г. была построена простейшая конфигурация ВМ Беббеджа, которая применялась для логарифмирования и решения алгебраических уравнений. Как писали современники, машина находила решения уравнений за минуты (в сравнении с опытным математиком, которому потребовались бы дни).

В машине Беббеджа арифметическое и запоминающее устройства планировалось реализовать на счетных колесах, а для хранения программы предусматривалось использование перфокарт.

Проект Беббеджа опережал запросы времени, технические и технологические возможности реализации, он был дорогостоящим. Именно поэтому Британский парламент в 1842 г. прекратил оплату проекта по гранту. Беббедж продолжал работу над проектом более 30 лет и разработал 239 детальных чертежей.

При жизни Беббеджа были изданы (1842) лишь лекции, описывающие проект ВМ. Полностью проект Беббеджа был опубликован в 1888 г. его сыном.

Спустя почти 100 лет вернулись к проблеме создания цифровых ВМ с программным управлением, по-видимому, не воспользовавшись работами Беббеджа.

1.1.4. Вычислительные машины К. Цузе

Первые идеи немецкого инженера К. Цузе (K. Zuse, 1910–1995) по конструированию «механического мозга» относятся к 1935 г., а основополагающие стенографические заметки по двоичному компьютеру — к 1937 г. Цузе построено семейство Z механических и электромеханических (релейных) вычислительных машин* Архитектурные возможности моделей семейства почти совпадают.

Модель Z1 создана в 1938 г.; это *первый в мире цифровой механический компьютер с программным управлением*. Архитектурными особенностями Z1 являлись также: двоичная кодировка и система представления чисел с плавающей запятой (или «полулогарифмическая» система, если использовать терминологию Цузе). При этом длина числа составляла 21 разряд, из которых 1 разряд отводился под знак числа, 7 разрядов предназначались для порядка и его знака, 13 разрядов — для мантиссы.

---

* *Raul Rojas.* Sixty Years of Computation — The Machines of Konrad Zuse. Preprint SC 96-9. — Berlin, Konrad-Zuse-Zentrum für Informationstechnik. Berlin, 1996. 26 p.

Выбор двоичной системы счисления позволил построить машину Z1 из элементов-переключателей, имеющих всего два состояния (а не из счетных колес, рассчитанных на 10 состояний). В качестве переключательных элементов использовались не реле, а вырезанные лобзиком тонкие металлические пластины (около 20 тысяч).

Тактовая частота вычислительной машины Z1 составляла 1 Гц, время выполнения операции умножения — 5 с; емкость памяти — 64 слова; ввод данных осуществлялся с клавиатуры и устройства считывания с перфоленты, а вывод — на панель из электрических ламп (в двоично-десятичном представлении); масса Z1 — около 500 кг.

Вычислительная машина Z1 по своей сути была опытной моделью, которая никогда не применялась для практических целей. В 1943 г. машина Z1 была уничтожена после авиабомбежки (вместе со всеми чертежами и схемами). В 1989 г. она была реконструирована в Берлине самим Цузе, и сейчас экспонируется в Берлинском музее техники.

Модель Z2 построена в 1939 г., в ней *впервые были применены электромеханические реле.* В машине Z2 арифметическое устройство и устройство управления были реализованы на 800 реле, а память оставалась механической (от модели Z1). Такая гибридная конфигурация ВМ была недостаточно надежной и практического применения не нашла.

Модель Z3 — *первая в мире двоичная электромеханическая ВМ с программным управлением.* Работы по созданию машины Z3 были начаты в 1939 г., а ее монтаж был полностью завершен 5 декабря 1941 г.

Рассмотрим архитектурные возможности ВМ Z3. При этом, следуя традиции анализа компьютеров, приведем технические характеристики и функциональную структуру машины Z3.

Машина Z3 предназначалась для выполнения операций сложения, вычитания, умножения, деления, извлечения квадратного корня и вспомогательных функций (в частности, двоично-десятичных преобразований чисел). Для представления чисел использовалась двоичная система с плавающей запятой. Длина числа — 22 двоичных разряда, из которых 1 разряд — знак числа, 7 разрядов — порядок или экспонента (в дополнительном коде), 14 разрядов — мантисса (в нормализованной форме). Тактовая частота Z3 — 5,33 Гц, а ее быстродействие при выполнении сложения составляло 3 или 4 операции в 1 с, а время умножения двух чисел — 4...5 с.

Функциональная структура машины Z3 представлена на рис. 1.1. И арифметическое устройство, и память в Z3 были рассчитаны на обработку чисел с плавающей запятой. Устройство управления предназначалось для выработки последовательностей микроинструкций (управляющих сигналов), причем каждая из последовательностей соответствовала своей команде.

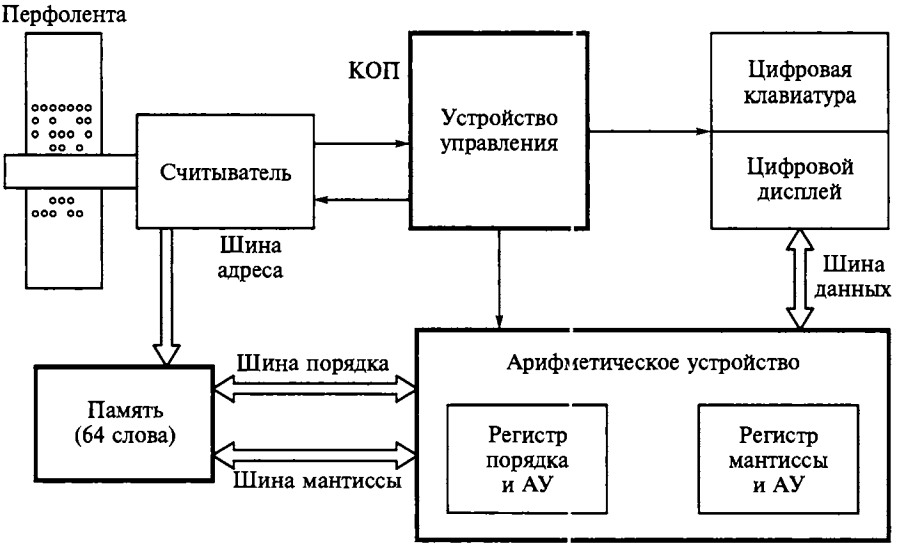


Рис. 1.1. Функциональная структура машины Z3:

→ — управляющие линии; КОП — код операции

Управление ВМ Z3 в целом осуществлялось от перфоленты. Итак, *перфолента была носителем программы вычислений.* Следовательно, в машине Z3 не могла быть реализована операция условного перехода.

Для ввода и вывода чисел использовались цифровая клавиатура и дисплей на электрических индикаторных лампах.

Для реализации машины в целом потребовалось 2600 реле (арифметическое устройство состояло из 600, память — из 1408 реле; остальные реле были применены в устройстве и цепях управления). Машина была смонтирована в трех стойках (высотой 6 футов*, шириной 3 фута), причем устройства арифметическое и управления размещались в одной стойке, а память — в двух. Масса Z3 — около 1 т.

Цена релейной машины Z3 составляла 25 000 немецких марок (что в то время соответствовало 6500 долл. США).

Вычислительная машина Z3 использовалась для решения задач Германского военно-исследовательского ракетного центра в Пенемюнде (Peenemünde, вблизи г. Росток). Здесь в 1942 г. была создана ракета V-2 («Фау-2») под руководством Вернера фон Брауна (Braun, 1912–1977).

Машина Z3 была разрушена в 1944 г. во время бомбардировки Берлина. Двадцать лет спустя ВМ была воссоздана и находится сейчас в Немецком музее в Мюнхене.

* 1 фут — 0,3048 м.

В 1942 г. было принято решение о создании более мощной ВМ Z4. Она должна была иметь память емкостью 1024 32-разрядных слова. Однако во время войны удалось создать лишь упрощенную модель такой ВМ. В 1950 г. машина Z4 была восстановлена и смонтирована в Федеральном политехническом институте в Цурихе (Швейцария); в 1955 г. ее перевезли во Французский научно-исследовательский аэродинамический институт (вблизи Базеля), где она эксплуатировалась до 1960 г.

В 1949 г. Цузе зарегистрировал компанию Zuse KG. Этой компанией была изготовлена 251 конфигурация ВМ Z4. В 1967 г. Zuse KG влилась в копанию Siemens AG.

1.1.5. Начальные понятия вычислительной техники

Функциональная структура вычислительного средства — блок-схема, отражающая состав основных функциональных устройств (или блоков), связи и взаимодействия между ними.

Позиционные системы счисления получили самое широкое распространение. В таких системах вес каждой цифры в любом числе определяется ее позицией (или разрядом, в котором она находится). Любое действительное число N, записанное в позиционной системе счисления как

$$N = \alpha_n\alpha_{n-1}\ldots\alpha_1\alpha_0,\ \ \alpha_{-1}\alpha_{-2}\ldots\alpha_{-(m-1)}\alpha_{-m},$$

допускает представление в виде следующей суммы степенного ряда:

$$N = \alpha_n A^n + \alpha_{n-1}A^{n-1} + \ldots + \alpha_1 A^1 + \alpha_0 A^0 +$$
$$+ \alpha_{-1}A^{-1} + \alpha_{-2}A^{-2} + \ldots + \alpha_{-(m-1)}A^{-(m-1)} + \alpha_{-m}A^{-m},$$

где A — основание, а каждый из элементов множества $\{\alpha_n, \alpha_{n-1}, \ldots, \alpha_1, \alpha_0, \alpha_{-1}, \alpha_{-2}, \ldots, \alpha_{-(m-1)}, \alpha_{-m}\}$ имеет значение одной из цифр из допустимого диапазона.

Если $A = 10$ и $A = 2$, то имеют место *десятичная* и *двоичная системы счисления* соответственно. В десятичной системе используют цифры 0, 1, 2, …, 9. Двоичная система применяется в вычислительной технике; цифрами здесь служат 0 и 1.

Например, десятичное число 3421,37 может быть представлено в виде суммы:

$$3\cdot10^3 + 4\cdot10^2 + 2\cdot10^1 + 1\cdot10^0 + 3\cdot10^{-1} + 7\cdot10^{-2} =$$
$$= 3000 + 400 + 20 + 1 + 0,3 + 0,07.$$

Теперь переведем двоичное число 1101, 101 в десятичное:

$$1101,\ 101_2 = 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 1\ 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3} =$$

$$= 1\cdot 8 + 1\cdot 4 + 0\cdot 2 + 1\cdot 1 + 1\cdot \frac{1}{2} + 0\cdot \frac{1}{4} + 1\cdot \frac{1}{8} = 13{,}625_{10}.$$

Формы представления чисел, используемые в компьютере, — с *фиксированной* и *плавающей запятыми* (точками). Первая форма отражает естественную запись чисел:

$$N = \pm\alpha_n\alpha_{n-1}\ldots\alpha_1\alpha_0,\ \ \alpha_{-1}\alpha_{-2}\ldots\alpha_{-(m-1)}\alpha_{-m};$$

машинное слово делится на три фиксированных поля. Первое поле — одноразрядное, используется для указания знака (±) числа; второе поле отводится для записи целой части

$$\alpha_n\alpha_{n-1}\ldots\alpha_1\alpha_0,$$

а третье — дробной части

$$\alpha_{-1}\alpha_{-2}\ldots\alpha_{-(m-1)}\alpha_{-m};$$

форма представления чисел с фиксированной запятой имеет следующий вид:

Знак	Целая часть	Дробная часть
Разряды машинного слова		

Недостатком формы с фиксированной запятой является малый диапазон представления чисел. Как правило, в этой форме записывают только целую часть числа и, следовательно, машинное слово трансформируется в слово из двух полей:

Знак	Число
	v разрядов

Максимальным по абсолютному значению целым числом будет число $2^v - 1$.

В машине знак «+» числа кодируется цифрой 0, а знак «–» кодируется как 1. Для представления положительных чисел используется прямой двоичный код, а отрицательных — обратный (инверсный) и дополнительный коды. Последнее упрощает реализацию арифметических действий над числами в двоичной системе счисления.

Обратный код отрицательного числа образуется поразрядной заменой в числе, представленном в прямом коде, 0 на 1, а 1 на 0. *Дополнительный код* отрицательного числа — это его обратный код, увеличенный на 1.

Число N, записанное в естественной форме, может быть представлено в следующем виде:

$$N = MA^p,$$

где M — мантисса, $M < 1$; A — основание системы счисления; $p = n + 1$ — порядок числа. Последний вид N называется нормальной формой записи числа.

Порядок записывается в машинном числе только степенью основания счисления. Он указывает местоположение в числе запятой, отделяющей целую часть числа от дробной. В зависимости от порядка запятая передвигается (плавает) по мантиссе.

Если в мантиссе справа от запятой стоит значащая цифра (не нуль), то число называется нормализованным. Условие нормализации

$$0{,}1 \leqslant |M| < 1.$$

Нормальная форма представления чисел с плавающей запятой имеет следующий вид:

Знак числа	Знак порядка	Порядок числа	Нормализованная мантисса

Диапазон представления чисел с плавающей запятой значительно больше, чем в случае с фиксированной запятой. Однако арифметические операции над числами с плавающей запятой проводить сложнее. Это объясняется тем, что при использовании плавающей запятой для каждой операции необходимо время для определения местоположения запятой. Точнее, в операциях над числами с плавающей запятой время расходуется на сравнение порядков и их выравнивание, на сдвиги мантисс, выполнение (арифметического или логического) действия над мантиссами, нормализацию мантиссы — результата, корректировку порядка результата.

Операнд — величина (или объект), над которой проводится операция в вычислительной машине. Например, при выполнении арифметических операций операндами являются числа.

Алфавит (в вычислительной технике) — это совокупность неразложимых и надежно отличаемых друг от друга символов (букв, цифр, знаков препинания и др.).

Слово — цепочка (последовательность) символов в некотором алфавите, имеющая определенное смысловое значение. *Машинное слово* — упо-

рядоченный набор букв, цифр, двоичных знаков и тому подобное, рассматриваемый в ВМ как единое целое. Число символов в машинном слове называют его длиной. Под словом понимается также одна из основных единиц количества информации (равная нескольким битам).

Данные — информация, представленная в виде, пригодном для обработки средствами вычислительной техники. *Цифровые (числовые) данные* — дискретная информация, представленная в некоторой системе счисления (например, в двоичной или десятичной). *Алфавитно-цифровые (текстовые) данные* составляются из любых знаков алфавита: букв, цифр и других символов.

Результат — совокупность данных, получаемых по завершении операции или программы.

Представление команд определяет функциональные возможности ВМ. Процесс машинной обработки информации (данных, чисел) задается программой. *Программа* — последовательность команд (инструкций). Под *командой* ВМ понимается информация, обеспечивающая выработку последовательностей управляющих сигналов, необходимых для выполнения машиной определенного действия.

Каждая команда состоит из полей, среди которых обязательно имеются операционные и адресные. В операционном поле находится код операции (КОП), который задает действие (арифметическое, или логическое, или другое) для машины. Адресные поля содержат адреса операндов (чисел, слов), используемых в операции. Под адресом понимается номер ячейки памяти, в которой хранится операнд. Количество указанных в команде адресов может быть от 1 до 3, поэтому различают одно-, двух- и трехадресные форматы команд.

Формат трехадресной команды имеет следующий вид:

КОП	A1	A2	A3

,

где A1, A2, A3 — адреса; например, A1 и A2 могут указывать на два операнда, а A3 — на номер ячейки, в которую следует записать результат.

1.1.6. Анализ механических и электромеханических вычислительных машин

Арифмометры и счетно-аналитические машины для выполнения статистических, бухгалтерских и финансово-банковских операций, а также для решения задач вычислительной математики развивались до 50-х годов XX столетия. Они полностью исчерпали возможности механики (*механической «элементной» базы*) и показали низкую эффективность ручного управления.

Следующим шагом в совершенствовании вычислительной техники явилось создание ВМ с программным управлением, использующих в качестве «носителя» программ перфокарты и перфоленты. В конце 1930-х годов начались работы по созданию ВМ на электромагнитных реле, т. е. внедрению *электромеханической элементной базы.*

Первой релейной ВМ была машина Z2 Цузе. В ней арифметическое устройство и устройство управления были построены на реле (см. разд. 1.1.4).

Многие годы считалось, что машина Mark 1 (H-Mark 1, Harvard Mark 1), созданная в США в 1944 г., является первой в мире релейной ВМ с программным управлением. Работы по созданию машины H-Mark 1 проходили в Гарвардском университете (Harvard University) под руководством Говарда Айкена (Howard Aiken, 1900–1973) с 1939 по 1944 г.

Машина Mark 1 была десятичной. Арифметическое устройство и устройство управления в ней были выполнены на десятипозиционных электромеханических элементах, при этом реле было основным логическим элементом. Программа работы ВМ размещалась на перфоленте и вводилась покомандно.

В машине Mark 1 использовалась форма представления чисел с фиксированной запятой. Машина обладала достаточно высоким для своего времени быстродействием: сложение двух 23-разрядных десятичных чисел выполнялось за 0,3 с, умножение — за 6 с, деление — за 11 с.

Стремление повысить надежность ВМ привело к созданию машины Mark 2 на шестиконтактных реле улучшенной конструкции, в которой было использовано 13 тыс. реле. Она могла выполнять операции сложения и умножения соответственно за 0,125 и 0,25 с.

Ради исторической справедливости следует подчеркнуть, что первым электромеханическим компьютером с программным управлением является машина Цузе Z3 (двоичная, с плавающей запятой), а не ВМ Айкена Mark 1 (десятичная, с фиксированной запятой).

1.2. Вычислительные машины на электронных лампах

Вычислительные машины на реле были ненадежны и не удовлетворяли требованиям по производительности и емкости памяти, которые предъявлялись к средствам обработки информации уже в 30-х и 40-х годах XX столетия. Указанные причины и достижения в электронике позволили начать разработку вычислительных машин на электронных лампах. В качестве основного элемента стали использовать ламповый триггер (переключательный элемент с двумя устойчивыми состояниями). Триггер был создан М.А. Бонч-Бруевичем еще в 1918 г. (аналогичную электронную схему предложили в США Икклз и Джордан в 1919 г.).

После налаживания производства достаточно надежных электронных элементов (1940-е годы) на их основе были созданы первые ВМ. Разработки таких ВМ выполнялись одновременно и за рубежом (в США, Англии), и в СССР.

Многие годы, вплоть до середины 80-х годов XX в., считалось, что американская машина ENIAC (1943–1945) является первым в мире компьютером на электронных лампах. Однако и здесь была своя предыстория.

1.2.1. Вычислительная машина Colossus

В 1936–1937 гг. английский математик Алан Тьюринг (Alan Turing, 1912–1954) теоретически доказал возможность создания *универсальной* цифровой вычислительной машины. Именно он показал, что любой алгоритм, в принципе, может быть реализован на дискретном автомате, и предложил такой автомат (абстрактный эквивалент алгоритма), получивший позднее название «машина Тьюринга».

В начале 40-х годов прошлого столетия в Великобритании А. Тьюрингом совместно с Х.А. Ньюменом была сконструирована специализированная электронная цифровая ВМ Colossus («Колосс» — название, подчеркивающее колоссальные возможности данного средства обработки информации). Данная ВМ предназначалась для разведывательных целей, для дешифрации германской секретной информации во время второй мировой войны.

Мощным стимулом для создания машины Colossus служила необходимость противодействия германскому подводному флоту и авиации, которые наносили большой ущерб Великобритании в начале войны. Для связи между германским командованием и войсками широко использовались радиопередачи информации. При кодировании этой информации применялись шифровальные машины Enigma («головоломка»). Дешифрация закодированной информации требовала огромного объема вычислений, который невозможно было выполнить даже большому коллективу профессионалов-вычислителей в необходимые сроки. Ясно, что указанную проблему можно было решить только при помощи высокопроизводительной ВМ.

Электронная машина Colossus запущена в эксплуатацию в 1943 г., однако публикаций о ней не было на протяжении последующих 30 лет.

1.2.2. Вычислительная машина ABC

Машина ABC является первым цифровым вычислительным средством, реализованным на электронных лампах, идея создания которой принадлежит болгарину Джону Винсенту Атанасову (John Vincent Atanasoff, 1903–1995).

В 1930-х годах испытывалась острая потребность в быстродействующих вычислительных средствах. В самом деле, в эти годы бурно развивалась квантовая физика, требовалось численными методами решать большие системы алгебраических уравнений (к которым сводились дифференциальные уравнения в частных производных). Существовавшие в то время электромеханические ВМ и аналоговые дифференциальные анализаторы не обладали необходимыми быстродействием и точностью.

Атанасов Дж.В., работая профессором теоретической физики в Колледже штата Айова США (Iowa State College), уже в 1933 г. стал задумываться над тем, как повысить скорость работы цифровых ВМ, и пришел к пониманию необходимости использования электронной (а не электромеханической) элементной базы. Однако он смог приступить к проектированию своей ВМ только в 1939 г., заручившись согласием частной корпорации Research Corp. и Агрономической станции штата Айова о финансовой поддержке. Атанасов со своим аспирантом Клиффордом Берри (Clifford Berry, 1918–1963) планировал построить специализированную цифровую ВМ на электронных лампах для решения систем линейных алгебраических уравнений (с 30-ю неизвестными).

Вначале были построены две малые машины, которые послужили прототипами большой электронной ВМ, которую позднее стали называть машина ABC (Atanasoff-Berry Computer). Машина ABC должна была войти в строй в 1943 г., но призыв Атанасова в армию не позволил завершить работу. Вместе с этим необходимо констатировать, что в 1942 г. машина ABC уже функционировала и осуществлялась отладка устройств ввода-вывода информации.

Остановимся на архитектурных особенностях машины ABC. Прежде всего функциональная структура ABC была рассчитана не на универсальные вычисления, а на реализацию операций сложения и вычитания векторов данных. В состав машины входили арифметическое устройство (АУ), запоминающие устройства: оперативное (ОЗУ) и внешние (ВЗУ). Для построения АУ были использованы радиолампы.

Машина ABC работала в двоичной системе счисления, слово состояло из 50 разрядов и представлялось в форме с фиксированной запятой.

Запоминающие устройства предназначались для хранения только данных. Память (или ОЗУ) имела емкость, рассчитанную на 32 слова. Она состояла из барабана, вращавшегося со скоростью 1 об./с. В барабане было смонтировано 1632 бумажных конденсатора, по 51 на каждую из 32 дорожек. Считывание и запись информации в память осуществлялись при помощи щеток (таких же, как в электродвигателях). Эта память была регенеративной, при каждом обороте барабана осуществлялось восстановление информации. Последнее было вызвано тем, что при считывании информации

происходил разряд конденсаторов и имели место естественные утечки заряда. Стоит отметить, что и в современной динамической памяти (например, на основе МОП-схем) также используется регенерация информации.

В качестве внешнего запоминающего устройства в АВС использовалось типовое перфокарточное оборудование. Данные представлялись в десятичной системе счисления и вводились в машину и выводились из нее с помощью перфокарт. Переводы чисел из десятичной системы в двоичную и обратно выполнялись аппаратурно. Программирование машины АВС осуществлялось путем ручных перекоммутаций, т. е. аппаратурно (hard, а не soft). Реализация операций условного перехода не предусматривалась.

Разработка АВС относится к числу пионерских, однако профессор Атанасов не удосужился получить патент на свою машину.

1.2.3. Вычислительная машина ENIAC

Вычислительная машина на электронной элементной базе ENIAC [1] была сконструирована и построена в Электротехнической школе Мура Пенсильванского университета США (Moore School of Electrical Engineering of the University of Pennsylvania) в 1943–1945 гг. Первая задача была решена на машине ENIAC в декабре 1945 г.; официальное ее представление состоялось в феврале 1946 г. Машина ENIAC подвергалась неоднократной модернизации и эксплуатировалась до 1955 г.

Идея создания машины ENIAC (Electronic Numerical Integrator And Computer — электронный цифровой интегратор и вычислитель) принадлежит Джону Мочли (John W. Mauchly, 1907–1980). Техническое руководство работами по конструированию ENIAC Мочли осуществлял совместно со своим молодым сотрудником Преспером Эккертом (John Presper Eckert, 1919–1995). В разработке ENIAC принимал участие и Джон фон Нейман (John von Neumann, 1903–1957).

Прежде чем обратиться к архитектуре ENIAC, заметим, что в начале 1940-х годов Атанасов поделился с Мочли информацией о принципах построения машины АВС. Более того, по приглашению Атанасова в 1941 г. Мочли посетил его лабораторию, где он детально изучил машину АВС. Позднее Мочли в своем письме обратился к Атанасову с просьбой разрешить ему продолжить разработки по электронной ВМ и получил от него положительный ответ. В 1973 г. в результате судебных разбирательств был доказан приоритет Атанасова в разработке принципов построения электронных ВМ, Мочли и Эккерт лишились возможности быть обладателями соответствующего патента.

Архитектурные возможности и состав машины ENIAC. Вычислительная машина ENIAC кардинально отличалась не только от предшест-

вующих ВМ, но и от машин, созданных позднее. В частности, от последних ВМ ее отличало использование множества полуавтоматических электронных вычислительных устройств, работающих параллельно и полунезависимо, и реализация памяти только на электронных лампах.

Вычислительная машина ENIAC имела следующие показатели: тактовая частота — 100 кГц; быстродействие — 5000 и 350 опер./с соответственно при сложении и умножении десятиразрядных десятичных чисел; количество электронных ламп и электромагнитных реле — 18 000 и 1500 соответственно; потребляемая мощность — 150 кВт; масса — 27 т; занимаемая площадь — 200 м^2.

Создание машины ENIAC оценивалось в 486 000 долл. и эта сумма превысила начальный бюджет на 225 %.

Машина ENIAC — это вручную перестраиваемая конфигурация, состоявшая из трех подсистем: управляющей, собственно вычислительной и ввода-вывода (рис. 1.2). Управляющая подсистема была представлена композицией из главного программного устройства (ГПУ) и двух дополнительных программных устройств (ДПУ). Вычислительная подсистема формировалась из 20 устройств накопления и суммирования (УНС), устройства умножения (УУМ), устройства деления и извлечения квадратного корня (УДК)

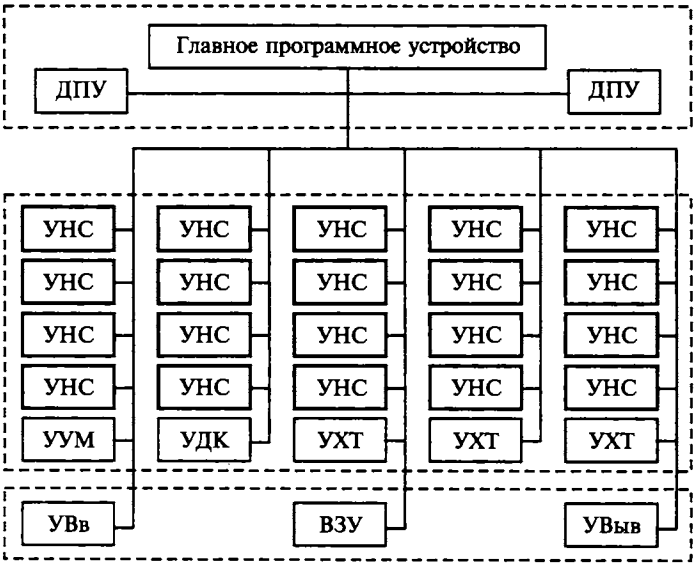


Рис. 1.2. Функциональная структура ENIAC:

ДПУ — дополнительное программное устройство; УНС — устройство накопления и суммирования; УУМ — устройство умножения; УДК — устройство деления и извлечения квадратного корня; УХТ — устройство хранения таблиц; УВв — устройство ввода; ВЗУ — внешнее запоминающее устройство; УВыв — устройство вывода

и трех устройств хранения таблиц (УХТ). Подсистема ввода-вывода состояла из устройств ввода (УВв) и вывода (УВыв) информации.

Устройства программного управления формировали последовательности управляющих сигналов для остальных устройств. Каждое УНС могло запоминать и суммировать 10-разрядные десятичные числа, оно, по сути, являлось регистром-аккумулятором информации. В ENIAC любое из вычислительных устройств (УНС, УУМ, УДК) выполняло функции как специализированного арифметического устройства, так и ячейки для хранения информации. Каждое УХТ предназначалось для хранения всевозможных констант. Оно было рассчитано на 104 десятичных числа.

В ENIAC использовалась любопытная система представления чисел с фиксированной запятой, а не двоичная (см. разд. 1.1.5) и не двоично-десятичная (при которой каждый десятичный разряд представляется тетрадой, т. е. четырьмя двоичными разрядами). В данной системе каждый разряд десятичного числа (каждая из цифр от 0 до 9) представлялся десятью двоичными разрядами, из которых только один был равен единице. При этом номер позиции, в которой стояла единица, определял значение всей цифры. Например, цифра 3 выглядела как 0000001000, а цифра 9 — как 1000000000. Физически это было реализовано следующим образом: десять триггеров на электронных лампах соединялись в кольцо, десять таких колец плюс два знаковых триггера образовывали одно УНС. Триггер был реализован на одной лампе — двойном триоде.

В машине параллелизм обеспечивался на макро- и микроуровнях. В самом деле, каждое вычислительное устройство (УНС, УУМ, УДК) имело свое локальное программное оборудование; следовательно, допускалась одновременная (параллельная) работа всех устройств. Кроме того, в каждом устройстве могла осуществляться и параллельная обработка всех 10 десятичных разрядов чисел.

Все схемы для логического управления и для выполнения арифметических операций, а также оперативная память ENIAC были реализованы на лампах. Для представления одного разряда десятичного числа в данной ВМ требовалось 10 электронных ламп. Машина содержала 18 000 электронных ламп. Частота срабатывания электронных цепей составляла 100 000 импульсов в секунду (тактовая частота — 100 кГц). Среднее время выполнения операций сложения, умножения и деления составляло 200, 2800 и 6000 мкс соответственно. Таким образом, машина ENIAC имела быстродействие на 3 порядка выше, чем у Z3, Н-Марк 1 и Н-Марк 2 (за счет применения электронных элементов вместо релейных).

В машине ENIAC не было отдельного устройства оперативной памяти. Как уже отмечалось, для запоминания информации могло использоваться 20 УНС, 3 УХТ, 1 УУМ и 1 УДК. Следовательно, можно считать, что

максимальное количество запоминаемых в ENIAC десятичных чисел было равно 334.

В качестве устройств для ввода исходных данных и вывода результатов использовалась распространенная в то время перфорационная аппаратура (для перфокарт и перфолент).

Функциональная структура ENIAC позволяла осуществлять подключение внешних запоминающих устройств. Для формирования ВЗУ могли быть применены элементы различных видов, в частности электромагнитные реле и управляемые вручную механические переключатели.

Структуру машины ENIAC можно было реконфигурировать; алгоритм ее функционирования проблемно не фиксирован (не специализирован). Адаптация структуры и алгоритма функционирования ВМ под конкретную задачу или программирование машины для решения выбранной задачи достигались вручную. При программировании ENIAC проводились соответствующие ручные перекоммутации (механическими переключателями) в блоках программных управлений каждого из вычислительных устройств, используемых при решении задачи, устанавливались соединения (при помощи гибких кабелей со штекерами) между блоками программных управлений, задавались таблицы функций (путем установки механических переключателей). Такая процедура программирования была длительной, трудоемкой, плохо контролируемой.

Области применения и эффективность машины ENIAC. Разработка машины ENIAC была поддержана Артиллерийским департаментом Армии США. С начала 1947 г. эта машина находилась на испытательном полигоне Баллистической исследовательской лаборатории в г. Абердине. Машина предназначалась прежде всего для расчетов по баллистике, в частности для составления таблиц стрельб. Она обеспечивала рекордную по тому времени скорость вычислений. Так, например, 60-минутная баллистическая траектория на машине ENIAC могла быть рассчитана всего лишь за 30 с, в то время как этот расчет квалифицированным человеком-вычислителем мог быть проведен на электромеханическом арифмометре за 20 ч. Данная ВМ работала в тысячу раз быстрее вычислительных машин Z3 или Н-Марк 2.

На ENIAC был сделан предварительный расчет первого американского термоядерного боеприпаса Mike («Майк»). Расчеты подобных устройств весьма трудоемки и громоздки, их практически нельзя выполнить без компьютера. Однако машина ENIAC все же мало устраивала специалистов-вычислителей из-за низкой производительности, надежности, малой емкости памяти.

Для удовлетворения потребностей Лос-Аламосского вычислительного центра в 1952 г. была создана новая генерация машины — MANIAC. Эта машина имела быстродействие 10 000 операций умножения в секунду, а ее память была реализована на магнитной ленте. Машина MANIAC по своей

архитектуре принципиально отличалась от ENIAC и прежде всего тем, что она обеспечивала *автоматизацию* (а не механизацию) *вычислений* (т. е. обладала способностью хранить программу расчетов в своей памяти).

Основные вычисления по термоядерному боеприпасу Mike были выполнены на MANIAC, он был взорван 1 ноября 1952 г. в Тихом океане.

Вычислительная машина MANIAC вошла в историю в качестве первой, на которой была реализована программа игры в шахматы.

Анализ машины ENIAC. Создание ENIAC расценивается как эпохальное событие в области механизации вычислений. Даже сейчас ни один экскурс в область истории информационных технологий не может игнорировать эту важнейшую разработку.

Машина ENIAC не имела жесткой структуры, более того, она не могла быть названа машиной (в современном понимании). Это был конструкторский набор из 30 основных вычислительных устройств, предоставляемый для программирования (по сути, конфигурирования) специализированной ВМ.

Программирование любой задачи осуществлялось аппаратурно и заключалось в ручных перекоммутациях внутри вычислительных устройств и соединений между ними.

Независимо от проблемной ориентации запрограммированной машины ENIAC ее архитектура относилась к классу SIMD, если придерживаться классификации* М. Флинна (M.J. Flynn), профессора Стенфордского университета (США). В средствах с такой архитектурой один поток команд управляет обработкой многих потоков данных (SIMD: Single Instruction stream / Multiple Data stream). Общее управление в ENIAC осуществлялось главным программным устройством (см. рис. ..2).

Отметим архитектурные достоинства машины ENIAC:

- SIMD-архитектура, распределенность и иерархия средств управления, смешанный синхронно-асинхронный способ управления вычислениями;
- параллелизм при обработке данных (допускалась одновременная работа нескольких вычислительных устройств и параллельная обработка десятичных разрядов чисел);
- ручная реконфигурируемость структуры (ручное программирование «неспециализированной» машины под структуру решаемой задачи);
- однородность, модульность и масштабируемость (варьируемость количества устройств).

Следовательно, машина ENIAC обладала совокупностью архитектурных свойств, которые присущи современным высокопроизводительным па-

* *Flynn M.* Some Computer Organizations and their Effectiveness//IEEE Trans. Computer. 1972. V. 21. № 9. P. 948–960.

раллельным вычислительным системам. Проект ENIAC опережал возможности элементной базы (ламповой электроники).

Если исходить из характеристик элементной базы 1940-х годов (а в то время ламповые элементы были самыми быстродействующими), то можно указать на следующие недостатки машины ENIAC:

- ручное (hard; механическое) трудоемкое программирование ВМ под структуру решаемой задачи (такое программирование длилось несколько часов или даже дней);
- низкая надежность, обусловленная применением большого числа ламп, электромагнитных реле, механических переключателей и кабелей, а также ручным программированием структуры машины;
- малая емкость оперативной памяти (334 10-разрядных десятичных чисел);
- громоздкость и дороговизна машины (18 000 электронных ламп, 486 000 долл.!);
- аппаратурная избыточность.

Поясним два последних пункта. Для представления 10-разрядного десятичного числа в ENIAC требовалось 100 триггеров (ламп), не считая знаковых. Однако если бы использовались двоичная или двоично-десятичная системы представления чисел, то в этой ситуации необходимо было бы 34 или 40 триггеров соответственно.

Машина ENIAC — это первая электронная ВМ, которая нашла практическое применение и была для своего времени инструментом решения сложных задач. Она завершает эпоху механизации вычислений, исчисляемую с появления первых арифмометров XVII в.

1.3. Первые электронные вычислительные машины

Прежде всего обратим внимание на предпосылки создания электронных вычислительных машин (ЭВМ) с хранимой программой.

В результате эксплуатации машины ENIAC стало ясно, что дальнейшее расширение возможностей ЭВМ по решению задач может быть достигнуто за счет автоматизации вычислений, повышения надежности, увеличения емкости памяти, совершенствования элементной базы. Для осуществления автоматизации вычислений ЭВМ должна обладать возможностью хранения программы в оперативной памяти. Следовательно, переход от механизации к автоматизации вычислений также связан с увеличением емкости оперативной памяти.

С целью создания оперативной памяти большей емкостью (по сравнению с памятью ENIAC) и более дешевой, чем ламповая, Эккерт предложил

использовать ртутные линии задержки. Опыт применения таких линий задержки уже был в военном радарном оборудовании. Идея Эккерта заключалась в том, чтобы создать циклическую (если говорить современным языком, динамическую) память на ртутной линии задержки путем подключения ее выхода к входу через ламповые усилитель и формирователь. В такой памяти можно было хранить порядка 1000 бит информации, а для ее реализации требовалась всего лишь одна линия задержки и несколько ламп (в то время как в ENIAC требовался ламповый двойной триод на каждый бит информации).

Необходимость автоматизации вычислений, возможности электронной элементной базы и физическая природа памяти на линиях задержки во многом предопределили архитектуру будущих ЭВМ, именно то, что они должны были быть полностью *синхронными* и *последовательными* и использовать *двоичную систему счисления*. К таким ЭВМ относится американская машина EDVAC [1], построенная в 1950 г.

Принято считать, что EDVAC положила начало первому поколению современных средств обработки информации. Данные средства обеспечивают автоматизацию вычислений, они способны не только хранить программу своей работы в собственной памяти, но и модифицировать ее. Последнее раскрывает суть предложений, сформулированных Дж. фон Нейманом в ходе работ над проектом EDVAC, или, как сейчас говорят, составляет сущность понятия «фоннеймановская архитектура».

Однако исторический анализ убеждает нас в том, что машина EDVAC была лишь третьей в ряду ЭВМ с хранимой программой. В самом деле, обратимся к работам 1940-х годов, которые велись по созданию автоматических ЭВМ и в Европе.

1.3.1. Вычислительная машина M-Mark 1

Машина M-Mark 1 является первой в истории реально работавшей ЭВМ с хранимой программой. Эта машина была построена в Манчестерском университете Великобритании (M-Mark 1 — Manchester Mark 1). Работы по ее созданию выполнялись в 1946–1948 гг. под руководством Ф. Уильямса и Т. Килберна (F.C. Williams and T. Kilburn).

В машине M-Mark 1 арифметическое устройство и память были построены как раздельные устройства. Причем арифметическое устройство реализовывалось на электронных лампах, а память — на электронно-лучевой трубке. В данной памяти двоичная информация представлялась в виде зарядов на поверхности экрана трубки. Память на электронно-лучевой трубке была предложена Ф. Уильямсом в 1948 г.

Числа в ЭВМ M-Mark 1 представлялись в двоичной форме с фиксированной запятой; способ обработки чисел — последовательный. Архи-

тектурным достоинством ЭВМ M-Mark 1 было и то, что в ней реализовывалась операция условного перехода (позволявшая изменять ход вычислительного процесса).

1.3.2. Вычислительная машина EDSAC

Машина EDSAC (Electronic Delay Storage Automatic Calculator — электронный автоматический вычислитель на линиях задержки) является второй ЭВМ с хранимой программой. Она была построена М. Уилксом (M. Wilkes) в 1949 г. в Кембриджском университете (Великобритания).

Вычислительную машину EDSAC характеризовали следующие показатели: тактовая частота — 500 КГц; быстродействие — 14 000 и 117 опер./с соответственно при сложении и умножении 17-разрядных двоичных чисел с фиксированной запятой; емкость памяти 18 тыс. бит.

Следует считать, что EDSAC является первой ЭВМ с фоннеймановской архитектурой. Синтез функциональной структуры EDSAC основывался на работах Дж. фон Неймана [1].

В машине EDSAC впервые была применена память на ртутных линиях задержки. Она состояла из 32 ртутных линий задержки, каждая из которых имела емкость 576 бит. Эта ЭВМ была 17-разрядной (однако в ней была реализована возможность обрабатывать слова и двойной длины). Сложение 17-разрядных чисел в EDSAC занимало 0,07 мс, а умножение — 8,5 мс.

Ввод данных в EDSAC производился с перфоленты, а вывод результатов — на пишущую машинку.

Работы по созданию первых ЭВМ в Советском Союзе велись независимо и параллельно с западными разработками. Первая отечественная машина МЭСМ была построена в период с 1948 г. по 1951 г., по своим архитектурным возможностям она не уступала EDVAC [2–4].

Компьютер EDVAC по существу стал одним из прототипов всех ЭВМ, разработанных в дальнейшем на Западе. Рассмотрим архитектурные возможности и функциональную структуру EDVAC.

1.3.3. Вычислительная машина EDVAC

Конструирование и изготовление машины EDVAC (Electronic Discrete Variable Automatic Computer — электронный автоматический вычислитель для дискретных величин) было осуществлено в 1944–1950 гг. в Электротехнической школе Мура Пенсильванского университета США (Moore School of Electrical Engineering of the University of Pennsylvania).

В основу ЭВМ были положены оригинальные принципы работы и решения по функциональной структуре и элементной базе, полученные

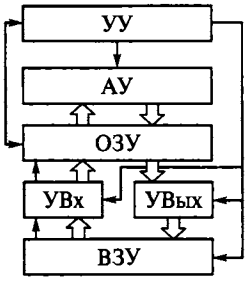


Рис. 1.3. Функциональная структура EDVAC:

УУ — центральное управляющее устройство; АУ — арифметическое устройство; ОЗУ — оперативное запоминающее устройство; УВх — входной узел; УВых — выходной узел; ВЗУ — внешнее запоминающее устройство; ⇒ — данные; → — команды и управляющие сигналы

Д. Мочли, П. Эккертом и Дж. фон Нейманом. Автоматизация вычислений была одной из основных парадигм при проектировании машины: EDVAC — это автоматический компьютер (Automatic Computer), т. е. ЭВМ, способная хранить в своей памяти программу вычислений.

Машина EDVAC вступила в строй в 1950 г. (хотя усовершенствования вносились до 1952 г.).

Отметим некоторые показатели EDVAC: тактовая частота — 1 МГц (на порядок выше, чем в ENIAC); быстродействие — 1000 опер./с над 32-разрядными двоичными числами; емкость оперативной памяти — 32 768 байт; количество электронных ламп — 3000.

Функциональная структура машины EDVAC. Вычислительная машина EDVAC (рис. 1.3) состояла из центрального арифметического устройства (АУ), оперативного запоминающего устройства (ОЗУ), внешних запоминающих устройств (ВЗУ), входного и выходного узлов (УВх, УВых) и центрального управляющего устройства (УУ). В отличие от ENIAC данная ЭВМ была последовательной машиной, она не могла выполнять две логические или арифметические операции одновременно. В то время это было технико-экономически обосновано.

Арифметическое устройство предназначалось для выполнения операций сложения, вычитания, умножения, деления, извлечения квадратного корня, для преобразования чисел из двоичной системы счисления в десятичную и обратно, для пересылок чисел из одних регистров АУ в другие, а также между ОЗУ и регистрами АУ и для осуществления выбора *одного из двух чисел в зависимости от знака третьего числа*. Последняя операция использовалась для передачи управления *(условного перехода)* от одной команды программы к другой. Числа в АУ обрабатывались последовательно, начиная с последнего значащего разряда, и в каждый момент времени выполнялась только одна операция. Регистры АУ представляют собой линии задержки на одно 32-разрядное двоичное слово.

Память (ОЗУ) содержала до 256 линий задержки, каждая из которых была способна хранить 32 слова, имеющих 32 двоичных разряда, а также переключательную схему, связывающую ячейки памяти с остальной частью машины. Память предназначалась для хранения начальных и граничных условий для дифференциальных уравнений в частных производных, произвольных числовых функций, промежуточных результатов вычислений, а

также *программы* (последовательности команд), управляющей ходом вычислений.

Внешние ЗУ были рассчитаны на следующие носители информации: перфокарты, бумажные перфоленты, магнитные ленты, фотопленку. Предполагалось использовать ВЗУ в качестве дополнительной медленнодействующей памяти, а также для ввода и вывода информации.

Следует подчеркнуть, что память EDVAC (как и в EDSAC) была последовательной, слова считывались из нее и записывались в нее последовательно бит за битом.

Входной узел предназначался для пересылки информации из ВЗУ в ОЗУ, выходной — из ОЗУ в ВЗУ. В оперативном запоминающем устройстве использовалась двоичная система счисления, а в ВЗУ — десятичная.

Устройство управления предназначалось для координации работы остальных устройств ЭВМ, в частности, оно формировало поток команд в АУ. Синхронизация работы всех устройств ЭВМ осуществлялась от единого источника импульсов, названного «часами» (сейчас это генератор тактовых или синхронизирующих импульсов).

В машине EDVAC первый двоичный разряд каждого слова служил для идентификации команд и чисел, причем единица соответствовала команде, а нуль — числу. В EDVAC использовались одноадресные команды, для задания кода операции и адреса операнда в ОЗУ отводилось соответственно 8 и 13 разрядов.

Рассмотрим типичный фрагмент программы обработки числовых данных и работу устройств ЭВМ. Пусть в АУ находится первое слагаемое, а в регистрах или ячейках ОЗУ α, $(\alpha+1)$ и $(\alpha+2)$ размещаются соответственно команда, задающая операцию сложения и адрес β, второе слагаемое и команда, которую предстоит выполнить вслед за сложением. Адекватной последовательностью действий ЭВМ будет следующее: пересылка команды из ячейки α в центральное УУ, передача слагаемого из $(\alpha+1)$ в АУ, выполнение операции сложения в АУ, запись суммы в ячейку β и, наконец, выполнение команды из ячейки $(\alpha+2)$.

Наряду с командой условного перехода (описанной выше) в машине EDVAC имелась команда безусловной передачи управления, именно команда с адресом γ, обеспечивавшая возможность для центрального УУ извлечения следующей команды из ячейки γ ОЗУ. Кроме того, в EDVAC была заложена возможность автоматической модификации адреса в команде. Последнее достигалось следующим образом:

- при пересылке некоторого числа из АУ в ячейку δ ОЗУ осуществлялся предварительный просмотр ее содержимого;

• если в δ была команда (т. е. слово с единичным первым разрядом), то вместо 13 адресных разрядов содержимого этой ячейки записывались 13 первых значащих двоичных разрядов результата.

Таким образом, машина *EDVAC была полностью автоматическим программируемым вычислительным средством*.

Анализ машины EDVAC. Машина имела жесткую функциональную структуру. По своей архитектуре EDVAC относится к классу SISD (Single Instruction stream / Single Data stream), если следовать классификации М. Флинна. В машине EDVAC одиночный поток команд обрабатывал одиночный поток данных (см. рис. 1.3).

Архитектурные особенности машины EDVAC:

• SISD-архитектура, синхронный метод управления устройствами;

• автоматизация вычислений (возможность хранения программы в памяти и ее автоматической модификации);

• последовательный способ обработки информации;

• фиксированность структуры (невозможность даже ручного реконфигурирования, за исключением ВЗУ);

• конструктивная неоднородность.

Архитектурные решения, положенные в основу EDVAC, привели к простоте ее реализации: потребовалось около 3000 электронных ламп (вместо 18 000 в ENIAC). Уровень сложности и достигнутые технические характеристики (показатели производительности, емкости памяти и надежности) ЭВМ вполне отвечали уровню техники и потребностям 50-х годов XX столетия. В самом деле, машина EDVAC характеризовалась следующими параметрами:

• количество двоичных разрядов для представления чисел — 32;

• тактовая частота — 1 МГц;

• емкость оперативной памяти — 2^{18} бит = 32 К байт.

Несмотря на последовательный характер работы, ВМ EDVAC не уступала по производительности ENIAC. Например, быстродействия ENIAC и EDVAC при выполнении операций умножения оценивались соответственно величинами: 357 опер./с (над 10-разрядными десятичными числами) и 1000 опер./с (над 32-разрядными двоичными числами).

Таким образом, *электронные вычислительные машины ENIAC и EDVAC отражают дуализм в развитии цифровых средств информатики, говоря иначе, констатируют неизбежность двух начал: параллельных и последовательных архитектур.*

1.3.4. Вычислительные машины MADAM и JOHNIAC

Первые три ЭВМ: M-Mark 1, EDSAC и EDVAC, безусловно, свидетельствовали о крупных достижениях электронной вычислительной техники.

Они стали прототипами для последующих разработок и показали технико-экономическую эффективность ОЗУ на электронно-лучевых трубках и линиях задержки. Однако наибольшую популярность получили ОЗУ на трубках; их использовали, пока не была внедрена память на ферритовых кольцах.

Машина MADAM. Вычислительная машина MADAM (MAnchester Digital Automatic Machine) является первой в мире коммерческой разработкой, использовавшей память на электронно-лучевой трубке. Руководителем проекта был А. Тьюринг, машина MADAM была построена в 1951 г. в Великобритании.

Машина JOHNIAC. Вычислительная машина JOHNIAC была создана в 1953 г. в Институте перспективных исследований США группой инженеров, возглавляемой Дж. Бигелоу. Название машине было дано в честь Дж. фон Неймана.

Во время работы над проектом EDVAC Дж. фон Нейман обратил внимание на перспективность применения памяти на электронно-лучевой трубке. Он предложил вариант такой памяти, использующий иконоскоп. (Иконоскоп — телевизионная передающая трубка с накоплением электрического заряда на мозаичной светочувствительной мишени.) Предполагалось хранить информацию на внутренней поверхности электронно-лучевой трубки, что стало основой разработки новой ЭВМ.

Дж. фон Нейман понимал, что машина с памятью на электронно-лучевой трубке будет намного превосходить по быстродействию все существовавшие тогда ЭВМ в основном по двум причинам. Во-первых, применение электростатической памяти обеспечивало непосредственный доступ к каждому разряду слова (тогда как в линии задержки разряд или все слово становились доступными лишь после прохождения их до конца этой линии). Во-вторых, стало возможным обрабатывать все разряды слова параллельно.

Машина JOHNIAC сыграла важную роль при создании в США термоядерной (водородной) авиационной бомбы. Бомба была испытана в США в 1954 г. (Термоядерная бомба впервые была испытана в СССР 12 августа 1953 г.)

1.4. Путь развития отечественной электронной вычислительной техники

Рассмотрим подробно отечественные электронные средства обработки информации. Путь развития электронных вычислительных средств в СССР был весьма тернист: он сопровождался невосприятием кибернетики в 1950-х годах, а также работ по параллельным вычислениям и по параллельным системам обработки информации в 1960-х годах. Тем не менее в Советском

Союзе были и крупные достижения: построены ЭВМ и вычислительные комплексы, не уступавшие зарубежным аналогам, созданы научные школы и вычислительная индустрия.

1.4.1. Малая электронная счетная машина

Первая ЭВМ в СССР и в континентальной Европе [2–4] — МЭСМ (Малая Электронная Счетная Машина) была сконструирована в Лаборатории моделирования и вычислительной техники Института электротехники Академии наук УССР (г. Киев). Работы по созданию МЭСМ были выполнены в 1948–1951 гг. и проводились под руководством основоположника отечественной электронной вычислительной техники С.А. Лебедева (1902–1974; академик АН СССР с 1953 г.; академик АН УССР с 1945 г.). Быстродействие МЭСМ составляло 50 опер./с; емкость оперативной памяти — 94 слова (63 — для программы и 31 — для чисел, двоичных 17-разрядных); количество электронных ламп — 6000; потребляемая мощность — 25 кВт, занимаемая площадь — 60 м^2.

Основные архитектурные принципы построения ЭВМ были разработаны С.А. Лебедевым (независимо от работ Дж. фон Неймана) в 1947 г. [3, с. 264]:

– в состав ЭВМ должны входить арифметическое устройство, память, устройство управления и устройство ввода-вывода;

– программа в машинных кодах должна храниться в той же памяти, что и числа;

– для представления чисел и команд должна применяться двоичная система счисления;

– вычисления должны выполняться автоматически в соответствии с программой, хранящейся в памяти;

– логические операции должны выполняться наряду с арифметическими операциями;

– память машины должна быть организована по иерархическому принципу.

Функциональная структура МЭСМ стала прототипом для последующих отечественных универсальных ЭВМ. В состав МЭСМ входили устройство управления (УУ), арифметическое устройство (АУ), запоминающее устройство (ЗУ), входное устройство и выводное устройство (для печати результатов). Остановимся на особенностях архитектуры МЭСМ.

МЭСМ была универсальной трехадресной синхронной ЭВМ. Машина имела всего 13 команд, которые позволяли реализовать, в частности, следующие арифметические и логические операции:

- сложение, вычитание, умножение и деление;
- сдвиг числа на заданное количество разрядов;

- сравнения двух чисел с учетом их знаков по их абсолютным величинам;
- сложение команд;
- передачу управления (из блока центрального управления в местное и обратно);
- останов машины.

Команды и числа представлялись словами из 17 двоичных разрядов. Форма представления чисел — с фиксированной запятой, один из 17 разрядов использовался для знака.

Команды условных переходов, изменение масштабов чисел, контроль исправности устройств в МЭСМ реализовывались программно. При переполнении разрядной сетки осуществлялся автоматический останов машины. Преобразование двоичных кодов в десятичные было реализовано схемно.

Устройство управления МЭСМ было представлено совокупностью блоков, среди которых, в частности:

- блок центрального управления (ЦУ), осуществлявший управление всеми операциями в машине и инициирование, а также завершение работы необходимых схем;
- блок управления командами, служивший для рассылки управляющих импульсов;
- блок управления операциями, определявший последовательность выполнения элементарных действий в арифметическом устройстве в соответствии с работой ЦУ;
- блок местного управления командами, обеспечивавший, в частности, работу магнитного запоминающего устройства.

Арифметическое устройство предназначалось для выполнения всех элементарных арифметических и логических операций. Оно включало накапливающий сумматор и два регистра на триггерах. В сумматоре была реализована цепочка сквозных переносов.

Запоминающее устройство включало оперативное ЗУ и внешнее ЗУ. Оперативная память состояла из электронных запоминающих устройств для чисел (ЭЗЧ) и для команд (ЭЗК). Емкость ЭЗЧ была рассчитана на 31 число, а ЭЗК — на 63 команды. Оперативная память была построена из ламповых триггеров и содержала 2500 триодов и 1500 диодов. Кроме того, имелось долговременное штекерное запоминающее устройство, которое позволяло вводить и хранить 31 число и 63 команды.

Внешнее ЗУ — магнитный барабан емкостью 5 тыс. слов. Была предусмотрена возможность подключения ЗУ на магнитной ленте (трехдорожечного магнитофона), которое могло использоваться для хранения и ввода подпрограмм.

Тактовая частота МЭСМ составляла 5 кГц. Полное время одного цикла, включавшего выборку двух чисел из ОЗУ, производство операции над ними,

передачу результатов в память и прием следующей команды, занимало 17,6 мс для всех операций, кроме деления, которое составляло от 17,6 до 20,8 мс.

Технические характеристики МЭСМ:

- система счисления — двоичная с фиксированной запятой;
- количество разрядов — 17 (из них один знаковый);
- вид запоминающего устройства — на триггерных ячейках (предусмотрена возможность использования магнитного барабана);
- емкость запоминающего устройства — 94 слова (31 — для чисел и 63 — для команд);
- емкость штекерного устройства — 94 слова (31 — для чисел и 63 — для команд);
- проводимые операции: сложение, вычитание, умножение, деление, сдвиг, сравнение с учетом знака, сравнение по абсолютной величине, передача управления, передача чисел с магнитного барабана, сложение команд, останов;
- система команд — трехадресная;
- арифметическое устройство — универсальное, параллельного действия, на триггерных ячейках;
- скорость работы — около 3000 опер./мин;
- система ввода чисел — последовательная;
- ввод исходных данных — с перфорационных карт или посредством набора кодов на штекерном коммутаторе;
- вывод результатов — фотографирование или посредством электромеханического печатающего устройства;
- контроль — программный;
- определение неисправностей — специальными тестами и переводом на ручную или полуавтоматическую работу;
- количество электронных ламп: триодов около 3500, диодов 2500;
- потребляемая мощность — 25 кВт;
- площадь помещения — 60 м^2 [3, с. 76 или 4, с. 349].

Машина МЭСМ явилась предвестницей ряда (семейства) быстродействующих ЭВМ, известных как БЭСМ. В процессе создания МЭСМ разрабатывались, монтировались и опробовались быстродействующие устройства и узлы для БЭСМ.

1.4.2. Быстродействующие электронные счетные машины БЭСМ-1 и БЭСМ-2

Главным конструктором машин семейства БЭСМ был академик С.А. Лебедев (БЭСМ — Быстродействующая Электронная Счетная Машина).

Машина БЭСМ АН СССР (БЭСМ-1). Электронная вычислительная машина общего назначения БЭСМ АН СССР (или БЭСМ-1), разработанная в Институте точной механики и вычислительной техники (ИТМиВТ) АН СССР (г. Москва), была самой быстродействующей машиной в Европе [3, 4]. Машина с оперативной памятью на ртутных линиях задержки (емкостью 1024 39-разрядных слов) была принята Государственной комиссией в апреле 1953 г.; с памятью на потенциалоскопах (запоминающих электронно-лучевых приборах, 1024 слова) — в начале 1955 г.; а с памятью на ферритовых сердечниках (2047 слов) — в 1957 г. Быстродействие БЭСМ-1 составляло 10 тыс. опер./с; среднее время полезной работы — 72 % общего времени функционирования ЭВМ; количество электронных ламп — 5000; потребляемая мощность ЭВМ (без системы охлаждения) — 30 кВт; занимаемая площадь — 100 м^2.

Архитектурные особенности машины. Система команд — трехадресная; число разрядов для кодов команд — 39; код операции — 6 разрядов; коды адресов — 3 адреса по 11 разрядов каждый. В систему операций машины входили: арифметические и логические операции, операции передач кодов и управления. Операции проводились с нормализованными и с ненормализованными числами.

Важной особенностью БЭСМ-1 стала реализация операций над числами с плавающей запятой, обеспечившая большой диапазон используемых чисел (от 10^{-9} до 10^{10}). На БЭСМ-1 достигалась высокая точность вычислений (около 10 десятичных знаков).

Система представления чисел — двоичная с плавающей запятой, число разрядов для кодов чисел — 39 (цифровая часть числа — 32 разряда; знак числа — 1 разряд; порядок числа — 5 разрядов; знак порядка — 1).

БЭСМ-1 была машиной параллельного действия: операции выполнялись одновременно над всеми двоичными разрядами чисел. Зарубежные ЭВМ того времени реализовывали последовательную или параллельно-последовательную системы обработки информации.

Кроме оперативной памяти в ней имелось долговременное запоминающее устройство (ДЗУ) на полупроводниковых диодах (емкостью до 1024 чисел), в котором постоянно хранились некоторые наиболее часто встречающиеся константы и подпрограммы. Содержимое ДЗУ не изменялось во время работы машины. Кроме того, ЭВМ имела внешний накопитель на магнитных лентах (НМЛ) — четыре блока по 30 тыс. чисел в каждом, а также промежуточный накопитель на магнитном барабане (НМБ) емкостью 5120 чисел (со скоростью выборки до 800 чисел в секунду).

Ввод информации в машину осуществлялся со считывающего устройства на перфоленте (1200 чисел в минуту), а вывод результатов — на электромеханическое печатающее устройство (1200 чисел в минуту) и фотопечатающее устройство (200 чисел в секунду).

Конструкция. ЭВМ БЭСМ-1 была собрана в одной основной стойке. В отдельной стойке размещалось ДЗУ и шкаф питания. Пульт управления служил для пуска и останова машины, отладки программ, а также для контроля за ее работой.

Элементно-конструкторская база. Она была представлена двух- и четырехламповыми блоками (ячейками), в которых были смонтированы триггеры, вентили, усилители и другие схемы, и соединительными платами без активных элементов. Один триггер (вместе с входами на диодах) занимал один четырехламповый блок. Вентили и усилители были двухламповыми. Усилители и некоторые вентили были выполнены на пентодах. В БЭСМ-1 было около 5 тыс. электронных ламп.

Программное обеспечение. Системное обеспечение в ЭВМ отсутствовало. Для машины БЭСМ-1 были разработаны система контрольных тестов, позволявшая быстро находить неисправности в машине, а также система профилактических испытаний для обнаружения мест возможных неисправностей.

Машина БЭСМ-2. Электронная машина БЭСМ-2 являлась серийным промышленным аналогом уникальной БЭСМ-1. ЭВМ разработана (1957) и внедрена (1958) в народное хозяйство коллективами ИТМиВТ АН СССР и завода им. Володарского (г. Ульяновск).

Основные технические характеристики БЭСМ-2 аналогичны характеристикам БЭСМ-1.

Системы команд машин отличались друг от друга лишь тем, что в БЭСМ-2 были исключены редко использовавшиеся команды (например, передача модуля числа) и добавлены некоторые новые команды.

Принципиальные особенности:

- оперативное запоминающее устройство было реализовано на ферритовых сердечниках. Емкость ОЗУ составляла 2048 39-разрядных чисел; время выборки — 10 мкс;
- внешние запоминающие устройства были выполнены на магнитных барабанах и сменных магнитных лентах. Емкость запоминающего устройства на одном барабане составляла не менее 5120 слов; скорость считывания или записи — 880 слов в секунду; частота импульсов — около 35 кГц; максимальное время ожидания первого числа — 80 мкс, среднее время ожидания — 40 мкс;
- количество устройств на магнитной ленте — четыре. Запись на ленту проводилась группами слов. Максимальное число слов в одной группе — 2047. Емкость каждой ленты составляла не менее 40 000 слов; скорость считывания или записи — 400 слов в секунду; частота следования импульсов — около 16 кГц;
- массовое применение полупроводниковых диодов. Количество полупроводниковых диодов — 5 тыс., электронных ламп — 4 тыс. Количество ферритовых сердечников — 200 тыс.;

• усовершенствованная (мелкоблочная) конструкция, значительно повысившая надежность и удобство эксплуатации. Применены разъемы с плавающими контактами.

Серийные машины БЭСМ-2 нашли широкое применение в вычислительных центрах и научно-исследовательских организациях как в СССР, так и за рубежом (КНР).

Были решены сотни тысяч задач чисто теоретических, прикладной математики, инженерных и пр. В частности, на БЭСМ-2 рассчитывались траектории полета космических аппаратов.

1.4.3. Электронные вычислительные машины М-20 и БЭСМ-4

Архитектуры машин М-20 и БЭСМ-4 предельно близки. Машины были созданы одним и тем же коллективом основных разработчиков. Главными конструкторами ЭВМ М-20 и БЭСМ-4 были академик С.А. Лебедев и канд. техн. наук О.П. Васильев.

Машина М-20. Создана в ИТМиВТ АН СССР; введена в действие в 1958 г.; выпускалась серийно.

Технические характеристики: быстродействие — 20 тыс. опер./с, оперативная память на ферритовых сердечниках емкостью 4096 слов, представление чисел — с плавающей запятой, разрядность — 45, система элементов — ламповые и полупроводниковые схемы, внешняя память — накопители на магнитных барабанах (НМБ) и лентах (НМЛ).

Принципиальные особенности:

• впервые в отечественной практике применена автоматическая модификация адреса;

• совмещение работы арифметического устройства (АУ) и выборки команд из памяти;

• введение буферной памяти для массивов, выдаваемых на печать; совмещение печати со счетом;

• использование НМЛ с быстрым пуском и остановом;

• для М-20 разработана одна из первых систем программного обеспечения ИС-2 (Институт прикладной математики АН СССР).

Машина БЭСМ-4. Введена в строй в 1962 г. Внедрение ее в народное хозяйство было осуществлено специальным конструкторским бюро ИТМиВТ АН СССР и заводом им. Володарского (г. Ульяновск).

БЭСМ-4 являлась фактически модернизированным вариантом ЭВМ М-20. В ней были использованы полупроводниковые элементы и расширенная система команд.

Технические характеристики: быстродействие — 20 тыс. опер./с, оперативное запоминающее устройство на ферритовых сердечниках емкостью

16 384 слова, представление чисел — с плавающей запятой, разрядность — 45, система элементов — полупроводниковые схемы, внешняя память — НМБ.

Принципиальные особенности:

- использование полупроводниковых элементов;
- программная совместимость с ЭВМ М-20;
- возможность подключения второго ОЗУ на ферритовых сердечниках емкостью 16 384 слова;
- работа с удаленными объектами по каналам связи (четыре входа с телефонных и 32 входа с телеграфных линий связи со скоростями передачи информации — 1200 и 50 бит/с соответственно).

1.4.4. Электронные вычислительные машины М-40, М-50 и 5Э92

Семейство ЭВМ: М-40, М-50, 5Э92 разработано под руководством главного конструктора академика С.А. Лебедева и заместителя главного конструктора В.С. Бурцева (1927; академик РАН с 1992 г.).

Машина М-40. Введена в действие в 1957 г., предназначалась для оснащения вычислительных комплексов системы противоракетной обороны (ПРО) Советского Союза. В ней нашли аппаратурное воплощение принципы распараллеливания вычислительного процесса. Все основные устройства ЭВМ (АУ, ОЗУ, внешние устройства) имели автономные системы управления и, следовательно, могли работать параллельно.

Технические характеристики: быстродействие — 40 тыс. опер./с, ОЗУ на ферритовых сердечниках емкостью 4096 40-разрядных слов, цикл памяти — 6 мкс, представление чисел с фиксированной запятой, разрядность — 36, система элементов — ламповая и феррит-транзисторная, внешняя память — НМБ емкостью 6 тыс. слов.

ЭВМ М-40 работала в комплексе с аппаратурой процессора обмена информацией с абонентами системы ПРО и аппаратурой счета и хранения времени.

Принципиальные особенности:

- плавающий цикл управления операциями, позволявший совместить во времени работу арифметического устройства, ОЗУ и процессора ввода-вывода;
- асинхронная работа с восемью дуплексными радиорелейными линиями связи с общей пропускной способностью 1 млн бит/с (без снижения производительности ЭВМ);
- наличие системы прерываний;
- впервые использовано совмещение выполнения операций с обменом;
- наличие мультиплексного канала обмена;

• работа в замкнутом контуре управления (в качестве управляющего звена);

• работа с удаленными объектами по радиорелейным дуплексным линиям связи;

• впервые введена аппаратура счета и хранения времени.

Область применения: комплексы управления радиолокационными станциями дальнего обнаружения и сопровождения цели и точного наведения противоракеты на баллистическую ракету противника. В марте 1961 г. под управлением комплекса системой ПРО впервые в мире была ликвидирована боевая часть баллистической ракеты осколочным зарядом противоракеты.

Машина М-50. Ввведена в строй в 1959 г. Она являлась модификацией ЭВМ М-40, обеспечившей выполнение операций с плавающей запятой.

На базе М-40 и М-50 был создан двухмашинный комплекс. Боевые пуски противоракет сопровождались записью информации на магнитные ленты контрольно-регистрирующей аппаратуры комплекса. Это давало возможность в реальном масштабе времени моделировать и анализировать каждый пуск (для чего М-40 и М-50 имели развитую систему прерываний).

Машина ЭВМ 5Э92. Данная ЭВМ представляла собой модификацию М-50, рассчитанную на применение в комплексе обработки данных. Особенности ЭВМ: широкое применение феррит-транзисторых элементов в низкочастотных устройствах.

1.4.5. Электронные вычислительные машины 5Э92б и 5Э51

Машина ЭВМ 5Э92б. Аванпроект ЭВМ был разработан в 1960 г.; год окончания разработки — 1961. Межведомственные испытания системы из восьми машин были осуществлены в 1967 г.

Архитектурные возможности. ЭВМ состояла из двух процессоров (большого и малого), работавших с общей оперативной памятью. Быстродействие большого процессора — 500 тыс. опер./с, а малого — 37 тыс. опер./с. Представление чисел — с фиксированной запятой, разрядность — 48, емкость оперативной памяти — 32 тыс. слов, основной цикл работы — 2 мкс.

Малый процессор осуществлял управление работой четырех НМБ (по 16 тыс. слов каждый) и 16 НМЛ. Он также обеспечивал работу ЭВМ с 28 телефонными и 24 телеграфными дуплексными каналами связи.

В этой ЭВМ впервые был реализован принцип многопроцессорности, внедрены новые методы управления внешними запоминающими устройствами, позволившие осуществлять одновременную работу нескольких машин на единую внешнюю память. Предусматривалась возможность объединения

ЭВМ в системы. В состав системы могло входить две, четыре или восемь ЭВМ.

Элементно-конструкторская база. ЭВМ была построена полностью на дискретной полупроводниковой элементной базе. Конструкция ЭВМ — ячеечная. Элемент замены — блок, содержащий 30 ячеек.

Принципиальные особенности:

- двухпроцессорная архитектура с общим полем оперативной памяти;
- одна из первых полностью полупроводниковых ЭВМ;
- полный аппаратный контроль;
- возможность создания многомашинных систем с общим полем внешних запоминающих устройств;
- возможность автоматического скользящего резервирования машин в системе;
- развитая система прерываний с аппаратным и программным приоритетами;
- работа с удаленными объектами по дуплексным и телеграфным линиям.

Программное обеспечение:

- специальное обеспечение реального времени;
- развитая система контрольных и диагностических тестов, использовавшая аппаратный контроль и позволившая определять неисправные блоки.

Машина 5Э92б применялась в вычислительных и управляющих информационных комплексах системы ПРО, комплексах управления космическими объектами, центрах контроля космического пространства и др.

Машина 5Э51. Это модифицированный вариант ЭВМ 5Э92б. Серийный выпуск и работа ЭВМ в системе ПРО осуществлялись с 1965 г.

Отличительные особенности ЭВМ 5Э51:

- представление чисел с плавающей запятой;
- виртуальная память;
- многозадачный режим работы с аппаратной поддержкой защиты информации в каналах обмена с оперативной и внешней памятью.

1.4.6. Электронная вычислительная машина БЭСМ-6

Главный конструктор ЭВМ — академик С.А. Лебедев; заместители главного конструктора: В.А. Мельников (1928–1993, академик РАН с 1981 г.), Л.Н. Королев (1926; чл.-корр. РАН с 1981 г.), Л.А. Теплицкий.

Машина БЭСМ-6 была самой распространенной моделью семейства БЭСМ, разработанного в ИТМиВТ АН СССР. Разработка машины БЭСМ-6 была закончена в конце 1966 г., ввод в эксплуатацию осуществлен в 1967 г.

Технические характеристики. ЭВМ обладала быстродействием около 1 млн одноадресных операций в секунду. Длина слова — 50 разрядов (из которых два контрольных); система счисления — двоичная; форма представления чисел — с плавающей запятой; среднее время выполнения операций: сложения — 1,2 мкс, умножения — 2,1 мкс, деления — 5,4 мкс, поразрядных логических — 0,55 мкс; система команд — одноадресная; длина команды — 24 разряда (две команды в слове); количество команд — 76.

Особенности функциональной структуры. Центральный процессор БЭСМ-6 составляли устройства управления и арифметико-логическое, а также буферная память из 16 50-разрядных регистров (с циклом обращения 0,33 мкс).

Оперативная память была реализована на магнитных сердечниках и состояла из блоков («магнитных кубов»), ее емкость составляла 32...128 К 50-разрядных слов (К = 1024). Архитектура ЭВМ допускала варьирование числа блоков памяти в диапазоне от 8 до 32. Время выборки слова из памяти равнялось 0,8 мкс.

Внешняя память формировалась из накопителей на магнитных барабанах и на магнитных лентах. Предусматривалось подключение к ЭВМ до 16 магнитных барабанов (каждый из которых обладал емкостью 32 К слов) и до 16 накопителей на лентах, имеющих по два лентопротяжных механизма (с емкостью бабины 1 млн слов). С 1972 г. в комплектацию серийных ЭВМ включались диски, а с 1978 г. — внешние устройства ЭВМ третьего поколения (семейства ЕС ЭВМ).

Электронная часть машины (устройство управления, арифметико-логическое устройство, устройство управления и коммутаторы для внешних устройств) были выполнены на полупроводниках (основная тактовая частота — 10 МГц).

Архитектурные достоинства:

- локальный параллелизм (на основе асинхронной конвейерной структуры);
- совмещение выполнения операций обращения к оперативной памяти с работой устройства управления и арифметико-логического устройства;
- совмещенный со счетом параллельный обмен массивами данных по шести каналам с магнитными дисками, барабанами и лентами;
- использование виртуальной памяти;
- возможность организации магазинного (стекового) способа обращения к памяти;
- наличие «сверхбыстродействующего» ассоциативного буферного запоминающего устройства;
- широкие возможности переадресации, включая косвенную адресацию;
- реализация режимов мультипрограммирования и разделения времени.

Последняя архитектурная особенность машины БЭСМ-6 поддержана схемами прерывания ЭВМ и защиты ее памяти, средствами преобразования математических адресов в физические оперативной памяти.

Программное обеспечение. В БЭСМ-6 было развитое программное обеспечение, в состав которого входили: операционная система (для управления мультипрограммным режимом обработки информации) и системы программирования, рассчитанные на символический машинно-ориентированный язык и на языки высокого уровня: ФОРТРАН и АЛГОЛ (для вычислительных задач) и ЛИСП (для обработки списков). Кроме того, для ЭВМ БЭСМ-6 были разработаны библиотека программ, средства отладки программ и тест-программы.

Машина БЭСМ-6 выпускалась серийно вплоть до 1983 г. и была, пожалуй, самой популярной машиной исследователей и математиков-вычислителей Советского Союза.

1.4.7. Другие направления отечественной вычислительной техники

В Советском Союзе выпускались и другие оригинальные ЭВМ, например «Стрела» (1953), семейство машин «Минск», «Урал», «Мир» и др. В 1970-х годах стали производиться ЭВМ третьего поколения. Уместно отметить, что эти машины основывались на архитектурах известных западных семейств ЭВМ. Так, например, прототипом семейств ЕС ЭВМ (единой системы электронных вычислительных машин) и АСВТ-Д (агрегатных средств вычислительной техники на дискретных компонентах) послужило семейство IBM 360 (а для последующих моделей ЕС ЭВМ — семейство IBM 370). Для семейств АСВТ-М и СМ-ЭВМ (системы малых ЭВМ) были взяты архитектуры семейств машин фирм HP (Hewlett-Packard) и DEC (Digital Equipment Corporation). Ориентация на «переповторение» западных архитектур привела примерно к 10–15-летнему отставанию отечественной вычислительной техники по сравнению с американской.

В 1962–1965 гг. в Сибирском отделении РАН были разработаны концептуальные основы построения вычислительных средств, основанных на новых принципах обработки информации (или, как сейчас говорят, с нефоннеймановской архитектурой). Эти средства стали называть вычислительными системами (ВС) или параллельными ВС. Публикация в 1962 г. об американском проекте системы SOLOMON появилась примерно через 6 месяцев после выхода в свет первой отечественной печатной работы по параллельным ВС. Первой параллельной ВС (причем с программируемой структурой) была система «Минск-222» [5]. Она была разработана и построена Институтом математики Сибирского отделения АН СССР (г. Новосибирск) совместно с Конструкторским бюро завода им. Г.К. Орджоникидзе

(г. Минск) в 1965–1966 гг. Для сравнения — американская система ILLIAC-IV была построена в 1972 г., т. е. через 6 лет после ввода в эксплуатацию ВС «Минск-222». Работы по созданию ВС ILLIAC-IV были начаты в 1966 г. в Иллинойском университете, они потребовали больших материальных затрат (не менее 40 млн долл.) Реализованная единственная конфигурация ILLIAC-IV многие годы оставалась самой мощной ВС ($2 \cdot 10^8$ опер./с) — это факт, однако она по архитектурным свойствам и функциональной гибкости уступала конфигурациям системы «Минск-222».

В России есть несколько первоклассных научных школ и промышленных организаций, способных создать оригинальные высокопроизводительные (10^{12}...10^{15} опер./с) средства для индустрии обработки информации текущего столетия.

1.5. Современный уровень вычислительной техники

Развитие средств ВТ идет по двум направлениям [5, 6].

1. *Электронные вычислительные машины и простейшие вычислительные системы.* Эти вычислительные средства основываются на эволюционных модификациях концептуальной последовательной машины Дж. фон Неймана (1945). Их процесс развития отражен в трех поколениях. Функциональные структуры ЭВМ первого поколения (1949) полностью основаны на машине Дж. фон Неймана и на ламповой элементной базе. Создание ЭВМ второго (1955) и третьего (1963) поколений сопровождалось не только отходом от принципа последовательной обработки информации, но и сменой элементной базы: переходом на транзисторы и интегральные схемы соответственно. Пределом в эволюционной модификации концептуальной ЭВМ Дж. фон Неймана является конвейерный способ обработки информации в сочетании с векторизацией данных. Последний нашел воплощение уже в архитектурно развитых ЭВМ третьего поколения (допускающих одновременное или параллельное выполнение небольшого числа операций). А такие ЭВМ, по сути, представляют собой простейшие вычислительные системы.

Вычислительные средства данного направления постоянно совершенствуются. Однако расширение функциональных возможностей, повышение быстродействия и надежности, уменьшение стоимости и сокращение габаритных размеров ЭВМ и простейших ВС достигаются главным образом за счет улучшения физико-технических характеристик элементов и внутренних информационных каналов.

Технический прогресс в этом направлении был настолько бурным, что уже после третьего поколения ЭВМ трудно выделить периоды для указания каких-либо новых поколений.

Для любого из трех поколений ЭВМ, для каждого из последующих этапов технического и технологического развития ВТ можно указать суперЭВМ или суперВС, т. е. вычислительные средства, обладающие предельными характеристиками по эффективности. Характерным для современного этапа является то, что архитектурные решения, которые были прерогативой суперЭВМ 1970-х и 1980-х годов, переместились с макроуровня на микроуровень, т. е. применялись в современных микропроцессорах (или в больших интегральных схемах — БИС).

Например, в суперВС Cray-1 1976 г. выпуска производительность 160 млн операций в секунду достигалась за счет параллельной работы 12 функционально ориентированных конвейеров. Каждый конвейер представляет собой цепочку из сегментов, количество которых соответствует числу стадий обработки операндов в конвейере и, следовательно, объему элементарных операций, выполняемых в нем за один такт. Максимальное число сегментов — 14 — было в конвейере для вычисления обратной величины. Современные микропроцессоры содержат конвейеры с числом сегментов, близким к 20. В частности, в микропроцессорах восьмого поколения фирмы AMD (Advanced Micro Devices) целочисленный конвейер насчитывает 12 сегментов, а конвейер для операций с плавающей запятой — 17. При этом производительность такого AMD-микропроцессора оценивается величиной в несколько миллиардов операций в секунду.

2. *Вычислительные системы.* Эти средства базируются на принципе массового параллелизма при обработке информации. Вычислительные системы (в концептуальном плане) являются диалектической противоположностью ЭВМ, их функционирование основано на имитации работы не отдельных людей, занятых расчетами, а коллективов людей-вычислителей. Это позволяет преодолеть барьер производительности, существующий для ЭВМ, достичь высокой надежности и живучести, осуществимости решения задач, значительно улучшить технико-экономические показатели. Данное направление адекватно учитывает текущие достижения в технологии БИС и ориентировано на применение полупроводниковых пластин с большим количеством элементов обработки информации. Вычислительные системы относятся к четвертому и последующим поколениям средств обработки информации.

Современная ВТ представлена широким спектром средств обработки информации от персональных компьютеров до ВС с массовым параллелизмом. Уровень быстродействия ЭВМ составляет миллиарды операций с плавающей запятой в секунду (GigaFLOPS). Вычислительные системы могут иметь в своем составе сотни, тысячи и даже миллионы процессоров (арифметико-логических устройств). Так, например, в Connection Machine, реализованной еще в 1980-х годах американской фирмой Thinking Machines Corp., количество процессоров достигало 65 536. В зависимости от

конфигурации быстродействие Connection Machine достигало значения до 10^{12} FLOPS.

Анализ самых мощных компьютеров мира, созданных в первом десятилетии XXI в., показывает, что число процессоров в них достигает 10^2–10^5. При этом производительность компьютеров составляет 10^{12}...10^{15} операций с плавающей запятой в секунду.

В декабре 1999 г. фирма IBM (International Business Machines Corporation) анонсировала проект вычислительной суперсистемы Blue Gene, обеспечивающей производительность 1 квадриллион (10^{15}) операций с плавающей запятой в секунду (1 PetaFLOPS). Предварительно объявленная стоимость реализации проекта — 100 млн долл. — уже в 2006 г. была значительно превышена. Конфигурация ВС Blue Gene/L (2008 г.) обладает пиковой производительностью 596,378 TeraFLOPS и включает в себя 212 992 процессора (с производительностью 2,8 GigaFLOPS). Системы Blue Gene/P и Blue Gene/Q будут обладать пиковой производительностью 1 и 3 PetaFLOPS соответственно; число процессоров в их конфигурациях составляет порядка 10^5–10^6.

Архитектура ВС постоянно совершенствуется, существует четко выраженная тенденция к построению распределенных систем с программируемой структурой. В таких ВС нет единого общего ресурса, отказ которого приводил бы к отказу системы в целом, средства управления и обработки информации, а также память распределены «в пространстве». Они обладают способностью автоматически реконфигурироваться, т. е. программно настраиваться под структуру и параметры решаемой задачи, под сферу применения.

Прогресс в индустрии обработки информации обусловлен достижениями в архитектуре и теории функционирования «большемасштабных» ВС, в параллельной вычислительной математике, в программном обеспечении систем, но и также успехами интегральной технологии.

Первый полупроводниковый триод или транзистор был изобретен в 1947 г. американскими учеными У. Шокли (W. Shockley), У. Браттейном (W. Brattain) и Дж. Бардином (J. Bardeen) в фирме Bell Labs (США).

Первая интегральная схема (или чип — chip), т. е. электронная схема (из транзисторов, диодов, резисторов и других элементов и соединений между ними), размещенная на одной пластине, была изобретена в 1959 г. Робертом Нойсом (будущим основателем фирмы Intel). В 1961 г. была уже создана первая четырехтранзисторная схема для выполнения арифметических операций (фирма Intel). В дальнейшем количество транзисторов, размещаемых на единице площади интегральной схемы, увеличивалось приблизительно вдвое через каждые год или полтора года. Затем в 1970 г. был построен первый 4-разрядный процессор в интегральном исполнении Intel-4004. Позднее эти

большие интегральные схемы стали называть микропроцессорами. Современная технология БИС позволяет создавать микропроцессоры, которые по своим архитектурным возможностям и по техническим характеристикам превосходят суперкомпьютеры XX в.

Отметим достигнутый уровень в микропроцессорной технике:

- тактовая частота — до 10^{10} Гц (10 GHz);
- количество транзисторов на кристалле — до 10^9;
- число выводов с кристалла — до 500;
- технологические нормы — 45 нм;
- диаметр кремниевой пластины — 300 мм;
- площадь кристалла — порядка 100 мм^2;
- потребляемая мощность — десятки ватт.

Предпринимаются попытки создания микропроцессора с тактовой частотой 10^{11}...10^{12} Гц (до 1 THz). Безусловно, такие микропроцессоры должны создаваться на архитектурных решениях современных параллельных ВС.

Очевидно, что будущие БИС — это ансамбли взаимосвязанных процессоров (System-on-chip), размещенных на пластине большого размера (с диаметром 200...500 мм). Такие интегральные схемы могут быть названы *системными БИС*, *или «системами—на—кристалле»*, так как они по сути будут параллельными микроВС с массовым параллелизмом. Ожидается, что системные БИС в 2010 г. будут состоять из 128 элементарных процессоров.

Считается, что дальнейшая миниатюризация кремниевых чипов станет экономически невыгодной после достижения технологического предела в 20 нм (физический предел — около 10 нм).

На смену процессу производства БИС на основе кремния придут нанотехнологии. Многие компании уже инвестируют в «посткремниевые» технологии. Так, фирма IBM делает ставку на углеродную элементную базу. В ее лабораториях уже в 2001 г. велись работы с образцами логических элементов на базе этой технологии.

В 2006 г. построен одномолекулярный транзистор с временем переключения, не превышающем 640 мкс. Размер органической синтезированной молекулы составляет около 1,5 нм.

В Bell Labs «транзистор в одну молекулу» — органический транзистор (основанный на углероде) был создан еще в 2001 г. Он выращен методом химической самосборки молекул. Длина канала органического транзистора Bell Labs (т. е. расстояние между электродами) составляет всего лишь 1...2 нм (1 нм = 10^{-3} мкм = 10^{-9} м), т. е. примерно в 50 раз меньше, чем в последних достижениях кремниевой технологии (65 и 45 нм).

Дальнейший прогресс в индустрии обработки информации связывают с созданием квантовых компьютеров. Такие компьютеры основаны на зако-

нах квантовой механики и используют спин электрона или частиц атомного ядра и при представлении информации, и при ее обработке.

В конце 2001 г. в корпорации IBM построен квантовый компьютер из семи атомов, которые благодаря своим физическим свойствам могут выполнять одновременно функции процессора и памяти.

Квантовый компьютер — это кардинальное решение и по повышению эффективности средств обработки информации, и по дальнейшей микроминиатюризации интегральных схем.

Прежде чем будет построен промышленный квантовый компьютер, предстоит решить ряд сложных физико-технологических проблем и создать «квантовую» вычислительную математику.

Примечание. Спин элементарной частицы — собственный момент количества движения, имеющий квантовую природу и не связанный с перемещением частицы как целого. При введении данного понятия предполагалось, что элементарная частица может рассматриваться как вращающийся волчок, а ее спин — как характеристика такого вращения; отсюда следует название «спин» (Spin — вращение).

Для момента количества движения (или кинетического момента, или момента импульса) материальной точки относительно центра O справедлива формула $[\vec{r} \cdot m\vec{v}]$, где $\vec{r}$ — радиус-вектор движущейся точки, проведенной из центра O, $m\vec{v}$ — вектор количества движения (или импульс), m — масса, $\vec{v}$ — скорость точки.

Спин — величина векторная, измеряется в единицах постоянной Планка $\hbar \approx 1{,}0546 \cdot 10^{-34}$ Дж · с и равен $J\hbar$, где J — характерное для каждого сорта частиц положительное целое (0, 1, 2, ...) или полуцелое (1/2, 3/2, ...) число, называемое спиновым квантовым числом (обычно его называют просто спин).

Проекция спина на любое фиксированное направление в пространстве может принимать значения $J, J-1, \ldots, 1, 0, -1, \ldots, -J$. Следовательно, частица со спином J может находиться в $2J+1$ состояниях. Например, спин электрона (или протона, или нейтрона) равен 1/2, значит, эта частица может находиться в двух спиновых состояниях.

2. АРХИТЕКТУРА ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

Электронные вычислительные машины (ЭВМ) получили широкое применение в науке, технике, экономике и промышленности [3], их интенсивное развитие диктуется постоянно растущими потребностями общества в решении все более сложных задач. Процесс совершенствования ЭВМ характеризуется расширением функциональных возможностей, сокращением стоимости и уменьшением габаритных размеров. Как на начальном, так и на современном этапах развития ВТ улучшения характеристик ЭВМ обеспечиваются главным образом благодаря достижениям в элементной базе. В настоящее время они в значительной мере определяются возможностями технологии больших интегральных схем (БИС), возможностями микропроцессоров.
В этой главе даются понятия об архитектуре ЭВМ, о модели вычислителя и семействе ЭВМ; описаны поколения ЭВМ. Особое внимание уделено производительности и надежности ЭВМ.

2.1. Каноническая функциональная структура ЭВМ Дж. фон Неймана

Электронная вычислительная машина (Computer) — средство, предназначенное для автоматической обработки информации — данных (прежде всего в процессе решения вычислительных и информационно-логических задач). Путем итерации перейдем от этого интуитивного представления к более строгому понятию ЭВМ.

Функционирование ЭВМ определяется не только ее технической конструкцией (Hardware — аппаратурной частью), но и безусловно алгоритмом обработки информации (решения задачи), который обязательно представляется (записывается) на языке, доступном машине. Такое представление алгоритма обработки информации называют *программой*. Любая программа в конечном счете интерпретируется на машинном языке (как правило, в двоичных кодах). Говоря иначе, окончательное представление программы обработки информации есть последовательность команд, каждая из которых

задает операцию (действие) и указывает на данные, над которыми следует ее провести. Следовательно, в ЭВМ (в ее аппаратной части) должны быть устройства как для реализации операций, так и для хранения программ и данных.

Работа ЭВМ не обходится без так называемых системных программ (Software — программных средств), обеспечивающих, в частности, функциональную целостность машины, управление ее процессами и ресурсами и выполнение функций сервиса для пользователей.

Таким образом, *ЭВМ — это аппаратурно-программный комплекс (Hardware & Software), предназначенный для автоматического выполнения логико-вычислительной работы: для ввода (сбора), обработки, хранения и вывода (передачи вне) информации.*

Конструкция ЭВМ основывается на предложениях, выдвинутых Дж. фон Нейманом. Как уже отмечалось в § 1.3, с вычислительной техникой он впервые столкнулся летом 1944 г. К этому времени электронный компьютер ENIAC [1] был почти построен. Во время разработки машины EDVAC, в середине 1945 г., Дж. фон Нейман написал 100-страничный отчет, суммирующий результаты работ над ЭВМ. Этот отчет стал известен как первый «набросок» («First Draft of a Report on the EDVAC»). Отчет был недописан, в нем недостает многих ссылок. Однако в своем отчете Дж. фон Нейман достаточно ясно изложил принципы работы и функциональную структуру ЭВМ («the working principles and functional structure of modern computers»). Главное то, что он предложил отказаться от ручных переключателей, используемых при программировании ENIAC, и хранить программу работы ЭВМ в ее оперативном запоминающем устройстве (памяти) и модифицировать программу с помощью самой же машины.

Рассмотрим архитектурные принципы построения ЭВМ [7].

1. *Программное управление работой ЭВМ.* Программы состоят из отдельных шагов — команд; команда осуществляет единичный акт преобразования информации. Все разнообразие команд, использующихся в конкретной ЭВМ, составляет язык машины или ее систему команд. Таким образом, программа — это последовательность команд, необходимая для реализации алгоритма.

2. *Условный переход.* Условный переход — это возможность перехода в процессе вычислений на тот или иной участок программы в зависимости от промежуточных, получаемых в ходе вычислений результатов (обычно в зависимости от знака результата после завершения арифметической операции или от результата выполнения логической операции). Условный переход позволяет легко осуществлять в программе циклы (с автоматическим выходом из них), итерационные процессы и т. п. Благодаря этому число команд в программе получается во много раз меньше, чем число выполненных

машиной команд при исполнении данной программы (за счет многократного вхождения в работу участков программы).

3. *Принцип хранимой программы.* Этот принцип предопределяет запоминание программы вместе с исходными данными в одной и той же оперативной памяти. При функционировании ЭВМ команды выбираются из памяти в устройство управления, а операнды — в арифметико-логическое устройство. В машине и команда, и число считаются словами. Если команду направить в АЛУ в качестве операнда, то над ней можно выполнять арифметические операции. Это открывает возможность преобразования программ в ходе их выполнения. Кроме того, принцип хранимой программы обеспечивает одинаковое время выборки команд и операндов из памяти, позволяет быстро менять программы или части их, вводить непрямые системы адресации (которые позволяют работать с памятью произвольно большой емкости), видоизменять программы по определенным правилам.

4. *Использование двоичной системы счислений для представления информации в ЭВМ.* Этот принцип существенно расширил номенклатуру физических приборов и явлений для применения в ЭВМ. Действительно, в двоичной системе имеются только две цифры: 0 и 1, поэтому для их представления может быть использована любая система с двумя стабильными состояниями. Например, триод (открытое или закрытое состояния), триггер (элемент с двумя устойчивыми состояниями), участок ферромагнитной поверхности (намагниченный или ненамагниченный), импульсная схема (наличие или отсутствие электрического импульса) и т. п. К логическим схемам (построенным по двоичной системе счисления) можно применять математический аппарат булевой алгебры. Итак, двоичная система счисления существенно упрощает техническую конструкцию ЭВМ.

5. *Иерархичность запоминающих устройств (ЗУ).* С самого начала развития ЭВМ существовало несоответствие между быстродействиями АЛУ и оперативной памяти. Путем построения памяти на тех же элементах, что и АЛУ, удавалось частично разрешить это несоответствие, но такая память получалась слишком дорогой и требовала значительного количества электронных компонентов (что снижало надежность ЭВМ). Иерархическое построение ЗУ позволяет иметь быстродействующую оперативную память сравнительно небольшой емкости (только для операндов и команд, участвующих в счете в данный момент и в ближайшее время). При этом следующий более низкий уровень представляют внешние ЗУ на магнитных лентах, барабанах и дисках. Внешние ЗУ имеют относительно малую цену, обладают большой емкостью, но меньшим быстродействием, чем оперативная память. Иерархичность ЗУ в ЭВМ является важным компромиссом между емкостью, быстродействием, относительной дешевизной и надежностью.

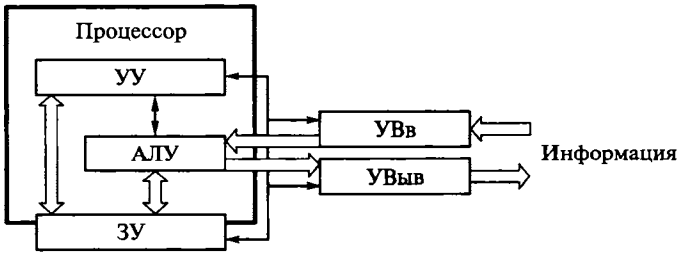


Рис. 2.1. Функциональная структура ЭВМ Дж. фон Неймана:

АЛУ — арифметико-логическое устройство; ЗУ — запоминающее устройство; УВв — устройство ввода; УВыв — устройство вывода; УУ — устройство управления; ⇒ — операнды и команды; → — управляющие сигналы

Эти принципы Дж. фон Неймана, несмотря на свою простоту и очевидность, являются фундаментальными положениями, определившими на многие годы бурное развитие вычислительной техники и кибернетики.

Каноническую функциональную структуру ЭВМ, представленную на рис. 2.1, связывают теперь с именем Дж. фон Неймана. Функциональное назначение устройств ЭВМ: АЛУ служит для выполнения арифметических и логических операций над данными (операндами: числами или словами, в частности, буквенными последовательностями), а также операций условного и безусловного переходов; ЗУ используется для хранения программ и данных; УВв — для ввода программ и данных, а УВыв — для вывода из ЭВМ любой информации (в частности, результатов); УУ координирует работу всех остальных устройств при выполнении программ.

Композицию из АЛУ, УУ и части ЗУ сейчас называют процессором. Если процессор имеет интегральное исполнение (т. е. представлен одной или несколькими БИС), то его называют *микропроцессором.*

Уместно отметить, что во время разработки EDVAC Дж. фон Нейман предпринимал попытки создать (в Институте перспективных исследований США) свою собственную версию машины, известную под именем IAS. Она имела каноническую функциональную структуру (см. рис. 2.1). Машина IAS рассчитывалась на двоичную арифметику с фиксированной запятой. Емкость памяти IAS составляла 4096 40-разрядных слов. Внутри АЛУ находился специальный регистр — *аккумулятор.* Он использовался для хранения одного из операндов или результата операции. При этом типичными командами стало прибавление слова из памяти к содержимому аккумулятора или занесение содержимого аккумулятора в ячейку памяти. Ясно, что аккумулятор позволял ограничиться одноадресными командами (см. разд. 1.1.5).

2.2. Модель вычислителя

Какова семантика слова «вычислитель» (Computer или Calculator — калькулятор)? (По «Толковому словарю живого великорусского языка» В.И. Даля «*Вычисл(я)итель* — кто вычисляетъ что-либо. *Вычислитель* также механическій снарядъ для вычисленья».) Приведенное определение свидетельствует о дуализме понятия «вычислитель». Рассмотрим это понятие с позиций ВТ.

В технике «вычислитель» — это средство обработки информации (например, арифмометр, электронный калькулятор, процессор, ЭВМ), работа которого основывается на примитивной имитации деятельности человека, занятого расчетами. Процесс обработки информации на вычислителе требует участия человека — оператора. Чем шире функциональные возможности вычислителя, чем выше уровень или механизации, или автоматизации вычислений, тем ниже частота взаимодействий между ним и оператором. Среди вычислителей как технических средств наивысшей степенью автоматизации счета обладает ЭВМ.

Итак, состав устройств и структура ЭВМ есть результат технической интерпретации функциональной организации человека — вычислителя. Процесс обработки информации (решения задач) на ЭВМ, по сути, сводится к имитации вычислительной деятельности человека. Следовательно, концептуальную основу конструкции ЭВМ и ее функционирования (или же основу автоматизации вычислений на ЭВМ) должна составить модель вычислителя.

Модель вычислителя есть пара:

$$c = \langle h, a \rangle, \tag{2.1}$$

где h и a — описания конструкции вычислителя и алгоритма его функционирования при обработке информации (или коротко: h — конструкция вычислителя, a — алгоритм его работы).

Конструкция вычислителя допускает следующее представление:

$$h = \langle U, g \rangle, \tag{2.2}$$

где $U = \{u_i\}$ – множество устройств $u_i, i \in \{1, 2, ..., k\}$ ($k = 5$ для концептуальной машины Дж. фон Неймана); g – описание структуры (или просто структура) сети связей между устройствами u_i. Иначе говоря, структура вычислителя — это граф, вершинам которого сопоставлены устройства u_i, а ребрам — линии связей между ними. В основе конструкции вычислителя лежат следующие три принципа:

1) последовательная обработка информации, т. е. последовательное выполнение:

– операций на множестве U устройств u_i, взаимодействующих через связи структуры g;

– микроопераций в пределах устройств u_i, $i = \overline{1, k}$;

2) фиксированность (автоматическая неизменяемость) структуры (и g, и микроструктуры устройств $u_i \in U$);

3) неоднородность составляющих устройств ($u_i \in U$, $i = \overline{1, k}$) и связей между ними (структуры g).

Алгоритм функционирования вычислителя обеспечивает согласованную работу всех устройств (множества U) и связей между ними (структуры g) в процессе обработки информации или, говоря иначе, при решении задач. Для решения любой задачи вычислитель должен иметь исходные данные D и программу p или запись алгоритма вычислений (на одном из возможных языков). Поэтому алгоритм a допускает представление в виде суперпозиции:

$$a(p(D)). \tag{2.3}$$

Для заданных D и p алгоритм (2.3) должен приводить к однозначному результату. Степень универсальности алгоритма работы вычислителя определяется разнообразием классов решаемых задач.

Итак, на основании (2.1) и (2.2) модель вычислителя

$$c = \langle U, g, a(p(D)) \rangle, \tag{2.4}$$

где U — множество устройств, обеспечивающих ввод, обработку, хранение и вывод информации; g — структура связей между устройствами; a — алгоритм работы вычислителя или алгоритм управления вычислительными процессами при реализации программы p обработки данных D. В модель вычислителя вкладывается каноническая ЭВМ Дж. фон Неймана.

Следует отметить, что описанные три принципа конструирования ЭВМ соответствовали уровню развития ВТ лишь в 50-х годах XX столетия, они позволили создать первые технико-экономически эффективные электронные машины. В последующих ЭВМ, основанных на новой элементной базе, технико-экономическая эффективность машин была достигнута уже за счет совмещения операций во времени их выполнения, ручной реконфигурируемости структур, возможности изменения (upgrade) составов машин. Каждый новый проект ЭВМ характеризовался очередной модификацией принципов построения, смена поколений ЭВМ сопровождалась все большим отходом от трех первоначальных принципов. В конце концов создатели средств обработки информации

пришли к необходимости применения диалектических противоположностей названных здесь принципов.

Таким образом, мы можем дать еще одно определение: *средство обработки информации, основанное на модели вычислителя, называется ЭВМ.* Процесс проектирования ЭВМ включает в себя выбор системы счисления и формы представления данных D; определение средств для написания программ p вычислений; подбор состава U вычислительных устройств и системы операций, реализуемых ими; формирование структуры g и разработку микроструктуры («логический» синтез) устройств $u_i \in U$; выбор элементной базы и конструирование устройств $u_i \in U$; построение такого алгоритма a функционирования вычислителя c, который обеспечивал бы реализацию и программ p, и, в частности, операций как последовательности микроопераций.

Допустимы аппаратурные, аппаратурно-программные и программные реализации модели вычислителя (2.4). Аппаратурное исполнение вычислителя c предопределяет каноническая ЭВМ Дж. фон Неймана; такое исполнение соотносится с первыми ЭВМ (первым поколением ЭВМ). Однако здесь уместно заметить, что даже в этих машинах имели место эволюционные модификации. Так, в машине JONIAC в отличие от EDVAC уже осуществлялась параллельная обработка всех разрядов слова (что может рассматриваться как «параллельное выполнение микроопераций»). В последующих разработках ЭВМ закладывалась и возможность совмещения операций.

Аппаратурно-программная реализация c, включая конструкцию h и алгоритм a (если учесть микропрограммное управление), сопоставляется с современными ЭВМ. Имеет место тенденция к вложению функций системного программного обеспечения в аппаратуру. Последнее поддерживается непрерывным совершенствованием технологии БИС, удешевлением элементной базы (в современных условиях — микропроцессоров).

Программное исполнение c следует воспринимать как машинный имитатор средства обработки информации, основанного на модели вычислителя (2.4). Говоря иначе, при программном исполнении модели вычислителя порождается виртуальная ЭВМ (или машинная модель ЭВМ).

Развитие ВТ по пути создания ЭВМ (как аппаратурных или аппаратурно-программных реализаций модели вычислителя или функциональной структуры машины Дж. фон Неймана) может осуществляться в ограниченных пределах, обусловленных, в частности, конечностью скорости распространения сигналов в физических средах (конечностью скорости света, которая в вакууме равна (299 792 ± 0,4) км/с).

2.3. Понятие об архитектуре ЭВМ

Широкое применение средств ВТ сильно укоренило понятие ЭВМ и породило, в свою очередь, понятие «архитектура ЭВМ». Последнее понятие, несмотря на свою распространенность, воспринимается, как правило, интуитивно и употребляется чаще всего при сравнении ЭВМ. В чем же заключается суть этого понятия?

2.3.1. Определения понятия «архитектура ЭВМ»

Понятие «*архитектура ЭВМ*» (Computer Architecture), по-видимому, впервые введено в 60-х годах XX столетия при создании машин IBM 360 фирмы International Business Machines. Это понятие было определено как «полная и детальная спецификация интерфейса «пользователь—ЭВМ». В качестве пользователя понимается все то, что имеет доступ к аппаратурно-программным средствам ЭВМ с целью обеспечить переработку на них информации. Например, это могут быть программисты, занятые отладкой и производством прикладных или системных программ на ЭВМ, или специалисты, подключающие технические комплексы к машине, или технические средства — «интеллектуальные» терминалы и т. п. Под интерфейсом следует понимать ту часть аппаратурно-программных средств машины, которая обеспечивает (пусть даже через устройства ввода-вывода информации) общение пользователя с ЭВМ. Говоря иначе, *интерфейс — это аппаратурно-программный посредник между пользователем и ЭВМ.* Ясно, что эффективность взаимодействия с ЭВМ, т. е. эффективность использования аппаратурно-программных средств ЭВМ, определяется возможностями или специфическими особенностями интерфейса. Итак, при такой трактовке понятия архитектуры ЭВМ главным становится то, что предлагается пользователю, и как воспользоваться сервисом, предоставляемым со стороны машины.

Под архитектурой ЭВМ, как и вообще любых других средств обработки информации, в узком смысле понимают совокупность их свойств и характеристик, призванных удовлетворить потребности пользователей. Пользователей в первую очередь интересуют такие свойства, которые раскрывают функциональные особенности вычислительного средства, а именно те, которые определяют: классы и сложность задач, доступных для решения; возможности автоматизированного обучения программированию и работе на данном вычислительном средстве; языки программирования; возможности отладчиков и редакторов, используемых при производстве программ; возможности операционной системы и организации различных режимов функционирования (моно- и мультипрограммных, разделения времени, реального масштаба времени и др.); реализуемость диалогового режима; орга-

низация работы с файлами; способы обработки информации (последовательный, конвейерный, матричный, распределенный и др.); возможность реализации надежных (отказоустойчивых) вычислений; совместимость с другими аппаратурно-программными средствами и т. п.

Большой интерес для пользователей средств ВТ представляют такие технические характеристики, как быстродействие, объем памяти, разрядность для слов, форма представления чисел (с фиксированной и/или плавающей запятой), показатели эффективности внешних устройств, количественные характеристики надежности и живучести, технико-экономические показатели (цена вычислительного средства или стоимость машинного времени), показатели безопасности работы и т. п. В меньшей степени пользователей волнует организация технического обслуживания и почти не интересуют такие архитектурные проявления особенностей средств переработки информации, как организация его функциональной структуры или отдельных его устройств, аппаратурно-программные решения, являвшиеся как следствие возможностей элементной и технологической баз, конструктивное оформление и т. п.

Понятия «архитектура ЭВМ» и «архитектура вычислительного средства» применяют не только пользователи, но и специалисты — создатели аппаратурно-программных комплексов для переработки информации. Поэтому узкое толкование понятия об архитектуре вычислительных средств для них является явно недостаточным. Учитывая ориентацию данной книги и на разработчиков средств ВТ, и на специалистов по формированию конфигураций и по эксплуатации ЭВМ и систем на их основе, дадим общее определение архитектуры средств обработки информации.

Архитектура вычислительного средства — совокупность его свойств и характеристик. То, что для пользователей средств ВТ было мало значимым, становится для создателей этих средств весьма существенным; например функциональная структура и организация вычислительных процессов.

Очевидно, что для разных специалистов одно и то же вычислительное средство выглядит в архитектурном плане различным образом. Каждого узкого специалиста занимают в большей степени свои архитектурные аспекты. Однако для всех профессионалов в области переработки информации представляют бесспорно большой интерес и методология разработки вычислительных средств, их аппаратурных и программных частей; и концептуальные основы и принципы построения; и схемотехнические решения, основанные на новых способах обработки информации и (или) имеющие лучшие параметры по сравнению с уже известными схемами; и алгоритмы и приемы организации функционирования; и методика экспресс-анализа эффективности (производительности, надежности, живучести, осуществимости решения задач и технико-экономической эффективности); и методика

обеспечения контроля и диагностики; и достигнутые технические характеристики в классе тех или иных вычислительных средств.

Учитывая вышесказанное и вспомнив каноническую функциональную структуру ЭВМ Дж. фон Неймана и модель вычислителя (формулы (2.1) и (2.2)), дадим еще одно определение: *архитектура вычислительного средства — концепция взаимосвязи и функционирования его аппаратурных (Hardware) и программных (Software) компонентов*. Очевидно, что данная концепция раскрывается через совокупность свойств и характеристик, присущих вычислительному средству.

2.3.2. SISD-архитектура ЭВМ

Во всех средствах ВТ, безусловно, имеют место информационные потоки — потоки команд и данных. Под потоком команд понимается их последовательность, выполняемая вычислительным средством, а под потоком данных — последовательность данных (включающая исходные данные, промежуточные и окончательные результаты), порождаемая, управляемая и обрабатываемая потоком команд.

Классификация архитектур на основе потоков команд и данных была предложена в 1966 г. профессором Стенфордского университета (Stenford University, США) М.Д. Флинном (M.J. Flynn). В соответствии с этой классификацией архитектура ЭВМ относится к классу SISD или ОКОД (SISD — Single Instruction stream / Single Data stream; ОКОД — один поток команд / один поток данных). Она предопределяет такое функционирование ЭВМ, когда одиночный поток команд управляет обработкой одиночного потока данных.

Вычислительное средство с архитектурой SISD представлено на рис. 2.2; поток команд (*1*) из памяти в процессор (точнее, в АЛУ, см. рис. 2.1) порождает потоки данных (*2*) в процессор (для их обработки) и результатов (*3*) из процессора (т. е. данных после обработки).

Очевидно (см. рис. 2.1 и 2.2), что известная нам машина Дж. фон Неймана имеет SISD-архитектуру. Архитектуру SISD и близкие к ней стали называть *фоннеймановскими* (von Neumann Architectures). Итак, фоннеймановская архитектура предопределяет такую функциональную организацию ВМ, при которой она состоит из двух основных частей: памяти, содержащей команды программы и данные, и процессора, выбирающего из памяти команды и их операнды и записывающего в нее результаты; каждая команда явно или неявно указывает адреса операндов, результата и следующей команды. Практически архитектура всех ЭВМ первого поколения — фоннейма-

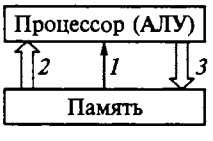


Рис. 2.2. SISD-архитектура ЭВМ:
1 — поток команд; *2* — потоки данных; *3* — потоки результатов

новская, а архитектуры ЭВМ второго и третьего поколений — модифицированные фоннеймановские.

Современные вычислительные средства высокой производительности (10^9...10^{15} опер./с) имеют нефоннеймановские архитектуры (non von Neumann Architectures). В таких средствах модель организации вычислений принципиально отличается от классической (т. е. от машины Дж. фон Неймана и от модели вычислителя). Нефоннеймановские архитектуры обязательно предопределяют непоследовательные организации выполнения команд. Средства обработки информации с такими архитектурами называют вычислительными системами (также суперЭВМ, так как они потенциально обеспечивают рекордную производительность).

2.4. Понятие о семействе ЭВМ

Современные ЭВМ выпускаются семействами или, как еще говорят, рядами. Что представляет собой семейство ЭВМ (Computer Family, Computer Series)?

Если рассматривать этот вопрос с позиций различных специалистов в области ВТ или же рассматривать конкретную ЭВМ под различными углами, то в поле зрения окажутся в общем случае различные совокупности архитектурных свойств и характеристик, т. е. одна и та же ЭВМ воспринимается различным образом. С другой стороны, если смотреть на каждую ЭВМ некоторого множества под одним и тем же углом, то можно увидеть подмножество архитектурно родственных машин. *Совокупность архитектурно близких ЭВМ, выделенную для фиксированного уровня развития ВТ и электронной технологии, называют семейством, или рядом, ЭВМ.* Пионером методологии семейств ЭВМ является фирма IBM.

Границы семейства ЭВМ устанавливаются чисто условно, строго указать набор архитектурных свойств машин не представляется возможным: можно говорить об архитектурной близости ЭВМ и понимать последнее на интуитивном уровне. Машины одного семейства могут различаться по техническим характеристикам (например, по производительности) и по конструктивному исполнению. На практике констатация любого семейства достигается путем перечисления марок машин (или, как говорят, моделей), в него входящих. Как правило, название семейства ЭВМ связывают с наименованием фирмы, которая разработала и производит машины.

Понятие «семейство ЭВМ» всегда ассоциируется с *совместимостью машин*. Последнее означает, что любая пользовательская программа, подготовленная для любой ЭВМ (модели) семейства, дает одни и те же результа-

ты на каждой из остальных машин (с учетом ограничений на направление совместимости; обычно ЭВМ совместимы «снизу вверх» — от «младших» моделей к «старшим», точнее: от моделей с меньшей к моделям с большей производительностью). Совместимость позволяет легко наращивать и сокращать функциональные возможности ЭВМ (путем изменения состава устройств) и просто заменять младшую модель на старшую, сохраняет задел в области программного обеспечения, упрощает процесс обучения обслуживающего персонала.

Различают аппаратурную, программную и информационную совместимость ЭВМ в границах семейства. *Аппаратурная совместимость* обеспечивается единством конструкторских решений, модульностью построения ЭВМ и стандартизацией связей и процедур управления как на уровне центрального процессора и оперативной памяти, так и на уровне внешних устройств машины. *Программная совместимость* достигается единством в функциональных (логических) структурах ЭВМ, единством команд, используемых в наборах команд машин, единством систем адресации. Набор команд любой модели семейства является подмножеством единой системы команд. Это гарантирует совместимость программ как снизу вверх (программы младших моделей могут исполняться на старших), так и сверху вниз (на старших моделях могут генерироваться программы для младших моделей). *Информационная совместимость* обеспечивается использованием единых форматов для представления данных, единых способов построения файлов (директорий массивов) данных и применением одинаковых носителей данных.

Следует подчеркнуть, что когда говорят о семействе ЭВМ, то прежде всего подразумевают архитектурное родство его представителей — моделей, но не одинаковость их по техническим характеристикам. Все модели, входящие в состав семейства, разнятся между собой векторами значений технических характеристик, среди которых главными выступают быстродействие, емкость памяти, разрядность слов, показатели надежности и стоимости, количественные характеристики функционирования средств взаимодействия ЭВМ с пользователями (внешней средой).

Приведем примеры семейств ЭВМ. Самыми распространенными семействами «больших» машин третьего поколения в мире были IBM S/360 и IBM S/370, а в Советском Союзе — ЕС ЭВМ и АСВТ-Д. Семейство ЕС ЭВМ включало в свой состав два подсемейства: «Ряд 1» и «Ряд 2». Машины «Ряд 1» были близки по архитектуре к моделям семейства IBM S/360, а машины «Ряд 2» — к моделям IBM S/370. Архитектура ЭВМ «Ряд 2» была совершеннее, чем у моделей «Ряд 1»:

1) были созданы аппаратурные и аппаратурно-программные средства для формирования в пределах «Ряд 2» многопроцессорных и многомашинных систем;

2) эффективность ЭВМ «Ряд 2» в 4–5 раз была выше, чем для ЭВМ «Ряд 1» (цена операции — в 2–3 раза меньше; быстродействие, емкость памяти, надежность — выше).

Примерами семейств мини-ЭВМ и микроЭВМ могут служить семейства HP и PDP фирм Hewlett-Packard и Digital Equipment Corp. соответственно. В СССР аналогами указанных семейств были АСВТ-М и СМ ЭВМ, а также «Электроника».

Впечатляющи успехи в области персональных компьютеров (Personal Computers), появившихся как результат эволюционного развития мини- и микроЭВМ. Персональные компьютеры уже сейчас превосходят суперЭВМ 70-х и 80-х годов XX столетия не только по функциональным возможностям, но и по производительности и емкости памяти; кроме того, они выигрывают у суперЭВМ и по цене за 1 опер./с, по габаритным размерам, удобству эксплуатации.

Сейчас самым распространенным является семейство персональных компьютеров IBM PC. Следует отметить, что в семейство IBM PC включаются как машины, произведенные фирмой IBM, так и совместимые с ними ЭВМ других фирм.

2.5. Поколения ЭВМ

Развитие современной ВТ характеризуется процессом смены одного поколения промышленных средств обработки информации другим, архитектурно более совершенным. Первые три поколения — это поколения ЭВМ (Computer Generations), основанных на модели вычислителя, их архитектура относилась к классу SISD.

Третье поколение ЭВМ берет начало в 1963–1965 гг. Бурное развитие ЭВМ (точнее, совершенствование их прежде всего по техническим параметрам) в последующие годы осуществлялось на фоне достижений в элементной базе. После появления первых интегральных схем (1959 г.) их возможности каждые 18 месяцев удваивались. В 1970 г. фирмой Intel был создан первый четырехразрядный микропроцессор Intel 4004. Архитектурные возможности современных микропроцессоров (см. § 1.5) представляются фантастическими, если их соотнести с параметрами ЭВМ середины 1960-х годов. Именно архитектура микропроцессора определяет и архитектуру современной ЭВМ.

Интенсивный прогресс в разработке и производстве все более совершенных ЭВМ обусловливает достаточно короткие периоды времени, которые характеризуются постоянством архитектурных свойств и параметров ЭВМ. Следовательно, после третьего поколения ЭВМ не представляется

возможным выделить очередное их поколение. Сейчас не говорят о поколениях ЭВМ вообще, а говорят лишь о генерациях машин в пределах семейств (например, семейства IBM PC).

Следует отметить и то, что на макроуровне все современные ЭВМ по архитектурным свойствам близки к машинам третьего поколения. Однако на микроуровне, т. е. на уровне функциональной структуры микропроцессора, отмечаются революционные решения: современный микропроцессор уже использует параллелизм при обработке информации, конвейеризацию вычислений и т. п.

Вычислительные средства четвертого и последующих поколений базируются на архитектурных решениях, диалектически отрицающих принципы конструирования ЭВМ. Такие средства, в частности, используют массовый параллелизм при обработке информации. Эти средства называются вычислительными системами (ВС), они будут рассмотрены в гл. 3–8.

Поколения вычислительных средств будем характеризовать совокупностью показателей эффективности и архитектурных свойств (табл. 2.1). Для представления эффективности ЭВМ каждого поколения используем вектор

$$\mathbf{E} = \{\omega, \nu, \vartheta, \sigma\},$$

где ω — показатель производительности (опер./с) или среднее число операций, выполняемых в секунду ЭВМ (процессором при работе с оперативной памятью); ν — емкость оперативной памяти (бит); ϑ — среднее время безотказной работы ЭВМ (или средняя наработка до отказа, ч); σ — «цена операции», определяемая как отношение цены ЭВМ к ее показателю производительности (измеряется в долларах, отнесенных к опер./с). Ниже будем указывать достигнутый порядок значений компонентов вектора **E**.

При описании архитектурных свойств ЭВМ будут рассмотрены: способы и режимы обработки информации; конструктивные особенности (составы устройств и структуры); алгоритмы управления вычислительными процессами или алгоритмы функционирования машин, т. е. $a(p(D))$ (см. § 2.2); возможности программного обеспечения (языки, операционные системы и т. п.); свойства элементной базы, характер проектирования и производства ЭВМ.

Первое поколение ЭВМ. Первое поколение ЭВМ появилось в 1949–1951 гг. ($\omega = 10^5$ опер./с, $\nu = 10^6$ бит, ϑ = 1...10 ч, $\sigma = 10$ долл./(опер. $\cdot$ с$^{-1}$)). Машины предназначались для последовательной обработки информации в монопрограммном режиме (были рассчитаны на то, что в любой момент времени в ЭВМ могла находиться только одна задача, представленная в виде последовательной программы).

Состав вычислительных устройств и структура ЭВМ первого поколения — канонические (см. рис. 2.1); разнообразие ЭВМ — варианты техниче-

Таблица 2.1

Поколение ЭВМ	Годы появления	Возможности пользования	Алгоритм управления $a(p(D))$, структура	Элементная и логико-конструктивная базы	Производство	Программное обеспечение	Средства обмена	Показатели			
								Производительность ω, опер./с	Объем памяти ν, бит	Безотказность ϑ, ч	Цена 1 опер./с σ, долл.
1	1949–1951	Одна задача, пассивный режим	Последовательный алгоритм, фиксированная структура	Лампы, компоненты	Индивидуальное	Машинные языки	Устройства ввода-вывода (УВВ)	10^5	10^6	1...10	10
2	1955–1960	Набор задач, пассивный режим	Последовательно-параллельный алгоритм, фиксированная структура	Полупроводники, вентили	Мелкосерийное	Алгоритмические языки, трансляторы, диспетчеры	УВВ, каналы связи	10^6	10^7	10^2	10^0
3	1963–1965	Мультипрограммирование, активный режим	Последовательно-параллельный алгоритм, ручное изменение структуры	Интегральные схемы, группы вентилей	Серийное	Система языков, операционные системы	УВВ, каналы связи, оптические устройства	10^7	10^8	10^3	10^{-1}

ской реализации концептуальной ЭВМ Дж. фон Неймана. Алгоритм управления вычислительными процессами $a(p(D))$ был универсальным и последовательным, и был адаптирован под фиксированную структуру ЭВМ. Этот алгоритм закладывался в аппаратуру машины при ее конструировании и оставался неизменным в течение всего периода существования ЭВМ. Следовательно, изменения алгоритма работы ЭВМ ни в процессе решения задачи, ни перед ее решением были не допустимы.

Возможности программного обеспечения ЭВМ первого поколения примитивны: использовались, как правило, машинные языки (двоичные коды) для записи алгоритмов обработки информации и стандартные подпрограммы (например, для вычисления элементарных функций).

Элементную базу ЭВМ первого поколения составляли электронные лампы. Проектировали машины вручную, производство было индивидуальным.

Второе поколение ЭВМ. Годы формирования: 1955–1960 ($\omega = 10^6$ опер./с, $\nu = 10^7$ бит, $\vartheta = 10^2$ ч, $\sigma = 10^0$ долл./(опер. $\cdot$ с$^{-1}$)). Способ обработки информации в ЭВМ оставался последовательным; однако допускалось и мультипрограммирование. *Мультипрограммирование* — такой режим обработки данных, при котором ресурсы ЭВМ одновременно используются более чем одной программой. Особенностью мультипрограммирования для второго поколения ЭВМ являлось то, что реализация размещенного в машине набора (пакета) последовательных программ решения задач оставалась последовательной. Это было следствием того, что процессор (вычислитель) мог в любой момент времени выполнять только одну команду. Вместе с этим допускалась параллельная работа внешних устройств, они могли одновременно с процессором выполнять команды, принадлежавшие нескольким программам набора. Такой мультипрограммный режим обработки информации позволял повысить производительность ЭВМ (точнее, снизить простои процессора) за счет устранения несоответствия между быстродействием процессора и скоростью работы внешних устройств (прежде всего устройств ввода и вывода информации).

Состав устройств и структура ЭВМ второго поколения не претерпели существенных модификаций. Однако по сравнению с первым поколением в процессор были введены структурные решения, ускорившие процесс реализации арифметических операций, а также схемы прерывания, обеспечившие работу ЭВМ в реальном масштабе времени (при управлении объектами, технологическими процессами, научными экспериментами и т. п.). Кроме того, в структуру ЭВМ закладывалась возможность подключения каналов связи для обеспечения пользователям теледоступа. Структура ЭВМ оставалась фиксированной, но при этом универсальный алгоритм управления вычислительными

процессами $a(p(D))$ стал последовательно-параллельным (т. е. допускал некоторые совмещения операций, например ввода и вывода с другими).

Программное обеспечение ЭВМ второго поколения приобрело многие функции, связанные с рациональным использованием машинных ресурсов и сервисом для пользователя. Появились диспетчеры — простейшие версии операционных систем; средства автоматизации программирования: языки для записи алгоритмов обработки информации (в частности, алгоритмические языки, например ALGOL 60, FORTRAN) и соответствующие трансляторы — программы для автоматического перевода с языка высокого уровня на машинный, сервисные программы, облегчающие редактирование и отладку пользовательских программ, и т. п.

Основы элементной и логико-конструктивной баз ЭВМ второго поколения составляли соответственно полупроводниковые приборы и вентили. При разработке ЭВМ использовалось моделирование и внедрялись элементы автоматизированного проектирования (например, при создании печатных плат). Производство машин было мелкосерийным, частично механизированным.

Способ обработки информации в ЭВМ и первого, и второго поколений был последовательным и процедурным. *Процедурная обработка* основывается на фиксированной системе команд обладающей полнотой (для обеспечения универсальности ЭВМ), и заключается в представлении любого алгоритма преобразования информации в виде программы — последовательности команд ЭВМ. Следовательно, процедурный способ обработки данных характеризуется тем, что время выполнения программы существенно зависит от адекватности системы команд ЭВМ решаемой задаче, а каждый акт преобразования данных сопровождается накладными расходами на выборку команды из памяти и ее расшифровку.

Третье поколение ЭВМ. Годы зарождения: 1963–1965; показатели эффективности: $\omega = 10^7$ опер./с, $\nu = 10^8$ бит, $\vartheta = 10^3$ ч, $\sigma = 10^{-1}$ долл./ (опер. $\cdot$ с$^{-1}$). При сохранении в основном последовательного способа обработки информации в архитектуру ЭВМ стали внедрять мультипрограммные режимы: пакетная обработка и разделение времени. *Пакетная обработка* (как и во втором поколении ЭВМ) заключалась в такой реализации набора последовательных программ, когда пользователь оказывался пассивным, лишенным возможности активно вмешиваться в вычислительные процессы. Режим разделения времени давал возможность нескольким пользователям осуществлять в интерактивном (или оперативном, on-line) режиме реализацию своих последовательных программ.

Режим разделения времени предоставлял каждому пользователю вполне определенный квант процессорного времени в соответствии с принятыми (и обычно детерминированными) правилами. Машина одновременно эксплуатировалась несколькими пользователями (хотя и с распределени-

ем времени процессора между ними). При этом создавалось представление, что каждый пользователь постоянно имел в своем распоряжении ЭВМ с вполне определенной архитектурой и техническими характеристиками (конечно, не превосходившими того, что было в реальной машине). Последнее обосновывало целесообразность употребления понятия «виртуальная машина» применительно к вычислительным ресурсам ЭВМ, выделенным пользователю. Режим разделения времени позволял повысить производительность ЭВМ (в частности, уменьшить простои процессора) путем устранения несоответствия между быстродействием процессора и скоростью работы пользователей ЭВМ.

Состав вычислительных устройств в машинах третьего поколения был дополнен спецпроцессорами, оптическими устройствами ввода-вывода информации, накопителями (на магнитных лентах и дисках) большой емкости и другими устройствами. Конструктивное оформление устройств выполнялось в виде модулей. Модули, одинаковые по функциональному назначению, могли отличаться друг от друга по техническим характеристикам.

Структурной особенностью ЭВМ третьего поколения являлось то, что они имели единый ресурс, через который осуществлялись взаимодействия между (центральным) процессором и остальными устройствами — модулями: спецпроцессорами, внешней памятью, устройствами ввода-вывода информации и др. В качестве такого ресурса выступали селекторный и мультиплексный каналы, общая шина и т. п.

В пределах любого семейства допускалось ручное формирование таких конфигураций ЭВМ, которые по своей архитектуре, структуре и составу были наиболее адекватны области применения (структурам и характеристикам алгоритмов решаемых задач).

В ЭВМ третьего поколения наряду с процедурным способом вычислений начали внедряться *элементы структурного способа*. Суть этого способа в общем случае заключается в возможности автоматической настройки таких структурных схем из устройств ЭВМ, которые были бы адекватны реализуемым алгоритмам обработки информации. В третьем поколении допускались модификации системы команд ЭВМ путем замены одной модели центрального процессора на другую модель, кроме того, и за счет микропрограммирования. Последнее основывается на представлении любой операции ЭВМ в виде микропрограммы и на введении в структуру ЭВМ микропрограммного управляющего запоминающего устройства. Модификация системы команд требует введения новых интерпретирующих микропрограмм и сводится к перепрограммированию управляющего ЗУ.

Для третьего поколения был характерен последовательно-параллельный алгоритм $a(p(D))$ управления вычислительными процессами, он обладал возможностью адаптации под конфигурации ЭВМ, порождавшиеся вручную.

Программное обеспечение машин этого поколения было представлено спектром операционных систем и систем автоматизации программирования. Операционные системы обеспечивали функционирование ЭВМ в основных режимах обработки информации (среди которых: пакетная обработка, разделение времени, работа в реальном масштабе времени). Системы программирования включали универсальные и проблемно-ориентированные языки и соответствующие трансляторы (компиляторы, интерпретаторы), средства отладки и редактирования программ и другие программные средства сервиса. В состав программного обеспечения включался и комплекс средств технического обслуживания ЭВМ (наладочные, контрольные и диагностические тест-программы).

Элементная база ЭВМ третьего поколения опиралась на интегральную технологию. Комплекты интегральных схем (включавшие микропроцессорные БИС) позволили существенно упростить проектирование ЭВМ; широкое применение получили системы автоматизированного проектирования (САПР). Производство ЭВМ стало серийным и автоматизированным. Машины третьего поколения выпускаются в виде семейств (например, IBM, ЕС ЭВМ, HP, DEC, СМ ЭВМ, «Электроника»).

Таким образом, любая ЭВМ с первого по третье поколение представлялась как комплекс аппаратурно-программных средств. Распределение стоимости между компонентами ЭВМ отражено на рис. 2.3.

Наблюдались относительный рост стоимости системного (программного) обеспечения и, следовательно, относительное уменьшение стоимости оборудования в пределах стоимости ЭВМ в целом. При этом имело место и абсолютное увеличение объемов ПО ЭВМ от поколения к поколению; для первого, второго, третьего поколений ЭВМ объемы соответственно были равны $4 \cdot 10^4$, $6 \cdot 10^5$, $6 \cdot 10^6$ К байт. Для ЭВМ третьего поколения затраты труда на

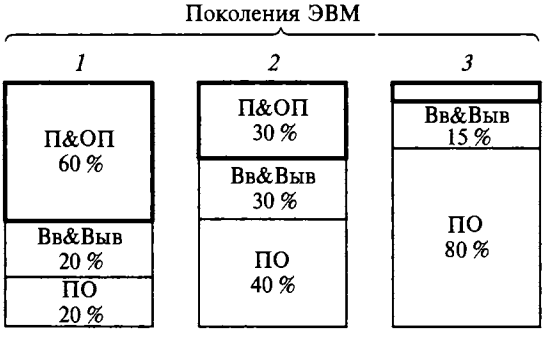


Рис. 2.3. Распределение стоимости между компонентами ЭВМ:

П&ОП — процессор и оперативная память; Вв&Выв — устройство ввода-вывода информации; ПО — программное обеспечение

разработку системного программного обеспечения составили 5000 человеколет, а стоимость программного обеспечения оценивалась в 50 млн долл.

Что же сейчас происходит в сфере ЭВМ? ЭВМ остались, их не вытеснили вычислительные средства с не-фоннеймановской архитектурой. Об успехах в области ЭВМ можно судить исходя из возможностей семейства IBM PC. Однако следует все же подчеркнуть, что архитектура современной ЭВМ имеет заметные отличия от канонической фоннеймановской. Даже в микропроцессор внедрены новые архитектурные решения, например конвейеризация архитектуры.

ЭВМ производились и будут производиться, впрочем, то же самое можно сказать даже и о калькуляторах. Но архитектура таких средств станет более совершенной, а технические характеристики будут значительно улучшены. Безусловно, микрокалькулятор XXI столетия не будет уступать по своим возможностям современной ЭВМ семейства IBM PC.

2.6. Производительность ЭВМ

Рассмотрим понятие о производительности ЭВМ и введем показатели и единицы измерения производительности машин.

2.6.1. Понятие о производительности ЭВМ

Электронные вычислительные машины являются наиболее распространенным, достаточно производительным средством индустрии обработки информации. Неослабевающий интерес к вопросам анализа эффективности функционирования ЭВМ объясняется следующими тремя причинами:

1) ЭВМ нашли применение во многих областях человеческой деятельности; число задач, допускающих решение на ЭВМ, постоянно растет; правильный выбор той или иной ЭВМ для конкретной области применения может быть осуществлен на основе анализа их возможностей, на основе численного анализа их показателей эффективности (в частности, производительности);

2) ЭВМ в расширенных или минимальных конфигурациях (процессоры с памятью) выступают в качестве массовых функциональных элементов в параллельных вычислительных системах;

3) ЭВМ и параллельные ВС составляют основу сложных систем различного назначения и, в частности, более мощных средств обработки информации, таких, например, как вычислительные сети и распределенные ВС.

Ясно, что эффективность функционирования ЭВМ не может быть исчерпывающим образом охарактеризована с помощью одного, пусть даже со-

ставного (комплексного, универсального) показателя. В самом деле, часто в качестве составного показателя эффективности ЭВМ используют выражение в виде дроби, в числителе которой стоят те количественные характеристики, которые желательно максимизировать, а в знаменателе — которые необходимо минимизировать. Общим недостатком составных показателей является то, что малая эффективность по одному показателю всегда может быть скомпенсирована за счет другого (например, малая производительность — за счет низкой стоимости машины). Именно поэтому должен рассматриваться комплекс показателей, которые в совокупности позволят оценить эффективность ЭВМ: производительность и надежность ЭВМ, осуществимость решения задач на машине и технико-экономическую эффективность ее функционирования. При комплексном оценивании поведения ЭВМ могут использоваться и составные показатели, но вместе с другими и обязательно с теми, которые входят в составные показатели.

Некоторые из показателей эффективности ЭВМ мы уже использовали в гл. 1 и в § 2.5, например быстродействие и среднее время безотказной работы (среднюю наработку до отказа). Это позволило достичь завершенности в анализе архитектурных решений, примененных в ЭВМ трех поколений. Однако строгие определения показателей не приводились; расчет был сделан на интуитивное восприятие смысла того или иного показателя эффективности функционирования ЭВМ.

В данном параграфе и ниже математически строго будут определены показатели эффективности работы ЭВМ и выведены расчетные формулы.

Общеизвестно, что производительность физического труда и оборудования измеряется объемом работы и продукции, производимыми в единицу времени. Измерение производительности умственного труда и производительности такого оборудования, как ЭВМ, является специфической и сложной проблемой. Ясно, что количественные характеристики для последних измерений связаны с информацией. Здесь предметом рассмотрения будет производительность ЭВМ (Computer Performance).

Под производительностью ЭВМ понимается ее способность обрабатывать информацию. Как правило, когда говорят о производительности, то понимают под этим потенциальную возможность ЭВМ по обработке информации (а не реальную, учитывающую аномальности в работе ЭВМ, например простои из-за отказов, из-за профилактического обслуживания и т. п.). В процессе обработки информации в ЭВМ реализуются те или иные операции из ее набора (или системы) операций. Состав набора операций, безусловно, характеризует архитектуру ЭВМ и, следовательно, определяет ее производительность.

Для оценки способности ЭВМ производить обработку информации используют *количественные характеристики или показатели производительности.* Эти показатели, безусловно, связаны с количеством информа-

ции, которое способна обработать ЭВМ в единицу времени, и, следовательно, с временем выполнения операций. Время выполнения операции в общем случае складывается из времени выборки (из памяти ЭВМ в процессор) команды (кода операции и адресов операндов), времени чтения операндов, времени реализации собственно операции и времени занесения результатов в память. При выполнении последовательности команд возможны совмещения во времени выполнения нескольких операций. При этом всегда имеет место зависимость времени выполнения операции от времени обращения к памяти.

2.6.2. Показатели производительности ЭВМ

Распространенным и простейшим показателем производительности ЭВМ является *тактовая частота*. Она указывает, сколько элементарных операций (тактов) может осуществить в единицу времени (секунду) ЭВМ (точнее, ее процессор). Или, говоря иначе, время такта — время выполнения элементарной операции процессором ЭВМ. Например, время такта в первых ЭВМ измерялось в миллисекундах (в EDSAC оно было равно 4 мс), а в современных машинах оно выражается в наносекундах (1 нс = 10^{-9} с) или даже величинами порядка 10^{-10} с.

Процессоры с одинаковой архитектурой (системой команд и функциональной структурой) могут иметь разную тактовую частоту. Процессор с более высокой тактовой частотой обладает большей производительностью (и, следовательно, более дорогой). С другой стороны, в различных моделях семейства процессоров одни и те же операции (например, сложения или умножения) выполняются за разное число тактов (за счет совмещения во времени выполнения нескольких элементарных операций). Чем «старше» модель, тем меньше (как правило) требуется тактов для выполнения одних и тех же операций, следовательно, тем выше ее производительность и цена (даже при одинаковых тактовых частотах).

В качестве простейших показателей производительности ЭВМ используются также числа однотипных операций, выполняемых в единицу времени (над операндами с одинаковой разрядностью). Для оценки производительности ЭВМ получили распространение числа операций, например сложения с фиксированной запятой, сложения с плавающей запятой, умножения и деления, выполняемых в секунду. Поскольку в общем случае длительности операций даже при одних и тех же операндах зависят от их типов, то для характеристики производительности ЭВМ целесообразно использовать средние числа операций, выполняемых в единицу времени.

Номинальным (или пиковым, или техническим) быстродействием (Nominal Speed или Peak Speed) ω' ЭВМ назовем среднее число операций,

выполняемых процессором (при равновероятном их выборе) в единицу времени (секунду) при работе только с оперативной памятью. (В состав такой памяти включаются собственно оперативная память и сверхоперативная — регистры общего назначения или кэш-память процессора.)

Пусть $\{\kappa_1, \kappa_2, \ldots, \kappa_j, \ldots, \kappa_k\}$ — часть операций из их полного набора (системы), которые не требуют обращений процессора к внешней памяти и устройствам ввода-вывода информации; τ_j — время выполнения операции κ_j, с; вероятность выбора любой операции κ_j ($j \in \{1, 2, \ldots, k\}$) есть постоянная вида $1/k$. Тогда по определению математическое ожидание времени выполнения операции в ЭВМ будет равно $k^{-1}\sum_{j=1}^{k}\tau_j$; следовательно, номинальное быстродействие

$$\omega' = k\left(\sum_{j=1}^{k}\tau_j\right)^{-1} \tag{2.5}$$

Очевидно, что при реализации на ЭВМ реальных программ решения задач имеет место неравновероятное использование тех или иных операций. Пусть ρ_j ($j \in \{1, 2, \ldots, k\}$) — вероятность выбора операции κ_j (вероятность спроса на κ_j, или вес операции κ_j). Тогда средним временем выполнения операции ЭВМ и *быстродействием ЭВМ по Гибсону* будут:

$$\sum_{j=1}^{k}\rho_j\tau_j; \qquad \omega^{o} = \left(\sum_{j=1}^{k}\rho_j\tau_j\right)^{-1} \tag{2.6}$$

Распределение вероятностей $\{\rho_j\}$, $j = \overline{1, k}$, $\sum_{j=1}^{k}\rho_j = 1$ или набор весов ρ_j зависят от характера задач. Существует несколько наборов весовых коэффициентов или, как говорят, несколько «смесей Гибсона», которые отражают статистику задач, решаемых на ЭВМ.

На практике достаточно часто используют модификации показателей (2.5) и (2.6), в которых в подмножество $\{\kappa_j\}$, $j = \overline{1, k}$, включаются только операции с фиксированной запятой (см. разд. 1.1.5).

Из (2.6) следует, что даже при работе с оперативной памятью процессор ЭВМ будет выполнять в единицу времени различное число операций при решении задач различных типов. Кроме того, при решении задачи

на ЭВМ требуются в общем случае затраты машинного времени на ввод программы и данных, обращение к внешней памяти, работу операционной системы, вывод результатов и т. п. Из сказанного видно, что в общем случае среднее число операций, связанных с решением задач и выполняемых процессором в единицу времени, будет отличаться и от номинального быстродействия машины, и от быстродействия ЭВМ по Гибсону. Поэтому для характеристики производительности ЭВМ при решении задач целесообразно ввести дополнительные показатели.

Пусть $\{I_1, I_2, \ldots, I_i, \ldots, I_L\}$ — набор типовых задач (эталонных тестов или программ оценки производительности, Benchmarks); ν_i — число операций, непосредственно входящих в программу решения задачи I_i; t_i — время решения задачи I_i (в t_i входят время, расходуемое ЭВМ собственно на счет, и дополнительные затраты машинного времени при решении I_i), с; быстродействием ЭВМ при решении типовой задачи I_i, $i \in \{1, 2, \ldots, L\}$, называют величину $\omega_i = \nu_i / t_i$. Отношение $1/\omega_i$ является средним временем выполнения одной операции при решении задачи типа i, $i \in \{1, 2, \ldots, L\}$. Пусть $\{\pi_1, \pi_2, \ldots, \pi_i, \ldots, \pi_L\}$ — распределение вероятностей спроса на типовые задачи I_i, $i = \overline{1, L}$, $\sum_{i=1}^{L} \pi = 1$, тогда $\sum_{i=1}^{L} \pi_i / \omega_i$ будет средним временем выполнения одной операции при решении набора типовых задач. *Средним быстродействием ЭВМ* называют величину

$$\omega = \left(\sum_{i=1}^{L} \pi_i / \omega_i \right)^{-1} \qquad (2.7)$$

Технология применения показателя ω для оценки производительности совместимых машин или ЭВМ в пределах одного семейства очевидна. Но формулу (2.7) можно использовать и для анализа производительности ЭВМ различных семейств. В самом деле, если машины имеют совместимость по языкам, то все тесты I_i, $i = \overline{1, L}$, можно представить программами на одном и том же языке (например, на языке высокого уровня FORTRAN). Тогда, используя для различных ЭВМ свои значения t_i и ν_i и ω_i, можно легко рассчитать величину ω (см. формулу (2.7)).

Существует множество тестовых наборов $\{I_i\}$, $i = \overline{1, L}$, среди которых достаточно популярен LINPACK. Он состоит из программ на языке FORTRAN для решения задач линейной алгебры и позволяет оценить производительность ЭВМ на вычислениях с плавающей запятой.

Как уже отмечалось выше, при сравнении производительности (2.7) несовместимых ЭВМ можно «пропускать» наборы эталонных тестов и оценивать время их решения. Однако в этом случае следует учитывать то, что для каждого теста должны быть произведены свои программы на языках команд анализируемых ЭВМ. Ясно, что эти программы будут состоять из различных команд (так как их наборы в общем случае зависят от архитектуры ЭВМ).

Можно поступить иначе: ввести унификацию операций и форматов операндов. В качестве унифицированной может быть взята операция, через которую могут быть выражены все остальные операции в каждой из сравниваемых несовместимых машин. Унифицированным форматом операндов может служить одна из структурных единиц информации, например байт. В этих условиях в качестве показателя производительности ЭВМ целесообразно использовать среднее эффективное быстродействие. Эффективным быстродействием ЭВМ при решении типовой задачи I_i, $i \in \{1, 2, \ldots, L\}$, назовем величину $\omega_i^* = \nu_i^* / t_i$, где ν_i^* — число унифицированных операций, с помощью которых можно интерпретировать программу решения задачи I_i; t_i — время решения задачи I_i на ЭВМ (включающее время реализации собственно программы и накладные расходы времени).

Среднее эффективное быстродействие ЭВМ

$$\omega^* = \left(\sum_{i=1}^{L} \pi_i / \omega_i^* \right)^{-1} \qquad (2.8)$$

Введенные показатели производительности выражают значение потенциально возможной производительности ЭВМ, причем номинальное быстродействие (2.5) и быстродействие по Гибсону (2.6) показывают потенциально возможную производительность при условии, что ЭВМ исправна, а среднее (2.7) и среднее эффективное (2.8) быстродействия — при условии, что машина исправна и занята решением задач («полезной работой»). Ясно, что показатели (2.7) и (2.8) зависят от набора эталонных тестов.

При техническом описании ЭВМ нередко ограничиваются оценкой простейших показателей производительности, таких как тактовая частота, время выполнения операций, число операций сложения с фиксированной (или плавающей) запятой, выполняемых в секунду, и т. п. Показатель производительности по Гибсону был популярен при анализе возможностей ЭВМ третьего поколения (например, семейств IBM и ЕС ЭВМ). Для оценки производительности микропроцессоров при целочисленных и вещественных вычислениях (с фиксированной и плавающей запятыми) используют тесты SPECint95 и SPECfp95 соответственно.

Показатель типа (2.7) применяется и при анализе производительности параллельных ВС. В этом случае берутся наборы параллельных эталонных тестов, например такие, как LINPACK-Parallel и NAS Parallel Benchmark. Первый набор оценивает производительность ВС при решении систем линейных алгебраических уравнений, а второй — состоит из восьми программ, взятых из реальных аэрокосмических расчетных пакетов. При составлении списка 500 самых мощных компьютеров мира (Top500 supercomputer list) используется тестовый набор LINPACK-Parallel.

2.6.3. Единицы измерения производительности ЭВМ

Для измерения производительности современных ЭВМ применяют крупные единицы, которые на несколько порядков отличаются от базовых: герц и операция в секунду.

Для измерения тактовой частоты ЭВМ используют мегагерцы — МГц, а также гигагерцы (ГГц или GHz).

Для оценки номинального быстродействия (2.5) и быстродействия ЭВМ по Гибсону (2.6) в случае, когда учитываются только операции с фиксированной запятой, применяются следующие единицы измерения:

- MIPS (Million of Instructions Per Second), 1 MIPS = 10^6 опер./с;
- GIPS, 1 GIPS = 10^9 опер./с.

Измерение производительности на тестовых наборах задач осуществляется в следующих единицах:

- 1 FLOPS (FLoating-point Operations Per Second), 1 операция с плавающей запятой в секунду;
- 1 MFLOPS = 10^6 опер./с = 1 млн операций с плавающей запятой в секунду;
- 1 GFLOPS = 10^9 опер./с = 1 млрд опер./с;
- 1 TFLOPS = 10^{12} опер./с = 1 трлн опер./с;
- 1 PFLOPS = 10^{15} опер./с = 1 квадриллион опер./с.

2.7. Количественные характеристики памяти ЭВМ

Запоминающее устройство, или *память* (Memory, Storage) ЭВМ, — функциональное устройство, предназначенное для приема, хранения и выдачи информации. Любой физический эффект, который может привести к созданию элемента с несколькими (минимум с двумя) устойчивыми состояниями, может быть положен в основу памяти. Память ЭВМ любого поколения имеет иерархическую структуру и строится на основе разнообразных физических эффектов.

2.7.1. Количество информации

Для оценки возможностей памяти применяются показатели. Все они связаны с понятием *«количество информации»*, введенном в 1948 г. американским инженером и математиком К.Э. Шенноном (C.E. Shannon, 1916–2002).

Для оценки количества информации будем использовать формулу

$$H = -\sum_{i=1}^{n} P_i \log_2 P_i,$$

где i — одно из альтернативных устойчивых состояний памяти; P_i — вероятность нахождения памяти в состоянии $i \in \{1, 2, \ldots, n\}$, $\sum_{i=1}^{n} P_i = 1$; при этом считается, что $0 \log_2 0 = 0$.

Если память может находиться в любом состоянии с равной вероятностью, т. е. если $P_i = 1/n$, $i = \overline{1, n}$, то количество информации определяется формулой

$$H = \log_2 n.$$

Единицей количества информации называется *бит* (англ. bit от binary — двоичный и digit — знак). Бит — количество информации ($H = 1$), посредством которого выделяется одно из двух альтернативных и равновероятных состояний ($n = 2$) памяти. Слово bit используется также и для обозначения двоичной цифры и двоичного разряда.

Запоминающее устройство, способное хранить 1 бит информации, называется *элементом* (или *ячейкой*) *памяти*. Самым распространенным элементом памяти является триггер (trigger — электронная схема с двумя устойчивыми состояниями). Считается, что в одном состоянии триггер хранит число 1, а в другом — 0.

Единицы количества информации представлены в табл. 2.2.

Таблица 2.2

Количественная характеристика	Единица количества информации				
	Бит (Bit)	Тетрада (Tetrad)	Байт (Byte)	Слово (Word)	Массив (Array)
H	1 бит	4 бита	8 бит	l бит	L слов
$n = 2^H$	2	16	256	2^l	2^{lL}

2.7.2. Показатели памяти

Емкость памяти (Memory Capacity) — максимальное количество информации, которое может в ней храниться.

В качестве простейших единиц измерения емкости памяти применяют бит и байт. Существуют и укрупненные единицы емкости памяти ЭВМ:

1 К бит (1 K bit) = 1024 бит = 2^{10} бит;

1 Мбит (1 M bit) = 1024 К бит = 2^{20} бит;

1 Гбит (1 G bit) = 1024 Мбит = 2^{30} бит;

1 Тбит (1 T bit) = 1024 Тбит = 2^{40} бит;

1 Пбит (1 P bit) = 1024 Пбит = 2^{50} бит.

Ширина выборки определяется количеством информации, записываемой в память или считываемой из нее за одно обращение.

Время выборки — промежуток времени с момента подачи сигналов чтения или записи до завершения соответствующей операции.

Время обращения складывается из времени выборки и времени, которое расходуется на то, чтобы память была готова к реализации следующей операции обращения. Это время называют также *длительностью цикла* обращения к памяти. В течение цикла можно выбирать (считывать или записывать) информацию, обновлять или модернизировать состояние некоторых элементов памяти.

Следует отметить, что длительность цикла обращения к динамической памяти всегда превышает время выборки, так как такая память требует регенерации информации через определенные промежутки времени. В самом деле, динамическая память, как правило, строится из элементов, реализованных по МОП-технологии, которая ориентирована на изготовление интегральных схем на основе униполярных полевых транзисторов и структур типа «металл–оксид–полупроводник». В статической памяти длительность цикла почти равна времени выборки, поскольку она не требует регенерации. Существует статическая память на бинополярных и униполярных полупроводниковых схемах. Статическая МОП-память имеет существенно меньшее быстродействие по сравнению с динамической МОП-памятью.

Быстродействие памяти характеризуется также *пропускной способностью* или *скоростью обмена информацией* между ней и другими устройствами. Эта скорость определяется количеством информации, которое можно записать в память или считать из нее в единицу времени. В качестве основной единицы измерения скорости обмена используют 1 бод = 1 бит/с (или 1 boud = 1 bit per second). К укрупненным единицам, характеризующим быстродействие памяти, относят:

1 Kboud = 1 Килобод = 1 Кбод = 10^3 бод;

1 Mboud = 1 Мегабод = 1 Мбод = 10^6 бод;

1 Gboud = 1 Гигабод = 1 Гбод = 10^9 бод.

Иногда пропускную способность памяти измеряют в следующих единицах:

1 byte/s = 1 байт/с, 1 K byte/s = 1 K байт/с, 1 Mbyte/s = 1 Мбайт/с и т. п.

Для оценки технико-экономических возможностей памяти используют показатель *удельной стоимости (цены)*. Этот показатель равен отношению стоимости (цены) памяти к ее емкости. Единицей для численного выражения удельной стоимости памяти служит *стоимость бита хранимой информации* (например, 1 руб./бит или 1 dollar/bit).

Для характеристики памяти ЭВМ применяют также показатели надежности, масс-габаритные параметры, величину потребляемой электроэнергии. Важной характеристикой памяти является также способность сохранения информации при отключении источника питания.

2.8. Надежность ЭВМ

Современная ЭВМ — достаточно сложный объект с позиции теории надежности. ЭВМ — это ансамбль, включающий технические (Hardware) и программные средства (Software). Даже техническая часть ЭВМ далеко не проста при изучении ее надежности в зависимости от надежности входящих в нее устройств. Работа устройств и узлов в общем случае основывается на различных физических принципах. При изучении функционирования ЭВМ как системы могут быть использованы известные методы и теории надежности, и теории массового обслуживания. Рассмотрим ЭВМ в целом, не вдаваясь в детальную ее структуру и архитектуру. Изучим способности ЭВМ функционировать в условиях отказов. Приведенный ниже материал будет полезен при исследовании потенциальной надежности параллельных ВС.

2.8.1. Основные понятия надежности ЭВМ

Основополагающими понятиями теории надежности ЭВМ являются отказ и восстановление.

Отказом называется событие, при котором ЭВМ теряет способность выполнять заданные функции по переработке информации (включая функции по вводу и выводу информации, хранению и собственно преобразованию информации). Это событие может произойти вследствие выхода из установленных пределов (допусков) значений одного или нескольких физических параметров, нарушения контактов, возникновения электрических пробоев, порчи программного обеспечения и т. п.

Различают полный и частичный отказы ЭВМ. *Полный отказ* приводит к абсолютному нарушению работоспособности ЭВМ, или, говоря иначе, к потере ее способности выполнять любые из заданных функций по переработке информации, что может произойти, например, вследствие аварийного отключения электропитания ЭВМ. *Частичный отказ* ЭВМ вызывает ухудшение качества ее функционирования или сокращение количества выполняемых функций. В данном параграфе полный и частичный отказы ЭВМ различать не будем.

Отказы приводят к таким изменениям в функционировании ЭВМ, которые носят постоянный характер. Для подчеркивания этого постоянства часто вместо термина «отказ» используют термин «устойчивый отказ». *Устойчивые отказы* могут быть устранены только в результате ремонта (или восстановления) машины. Наряду с отказами, как показывает опыт, в ЭВМ происходят неожиданные изменения физических параметров, когда они выходят за допустимые пределы. Такие изменения носят временный характер, они самоустраняются и называются *перемежающимися (или неустойчивыми) отказами или сбоями*. Ниже будем рассматривать только устойчивые отказы и для краткости использовать термин «отказ».

В случае, когда в машине произошел отказ, и он не устранен, то говорят, что ЭВМ находится в неработоспособном состоянии (в состоянии отказа) или отказавшая ЭВМ. При эволюционном развитии ЭВМ уменьшается вероятность состояния отказа, увеличивается период работоспособности ЭВМ. Так, работоспособность машин первого поколения ограничивалась, как правило, десятками минут, а в ЭВМ третьего поколения она достигала тысячи и более часов, в современных машинах составляет годы и десятки лет.

Несмотря на достаточную редкость отказов современных ЭВМ, требуются средства для поддержания работоспособности машины, т. е. средства технического обслуживания, контроля, диагностики и устранения неисправностей (или самоконтроля, самодиагностики и самоустранения неисправностей). Среди этих средств имеются микропрограммные, аппаратурные, аппаратурно-программные и программные компоненты. Более того, в ЭВМ общего назначения контроль, диагностика и устранение неисправностей осуществляются бригадами технического обслуживания.

Восстановлением называется событие, заключающееся в том, что отказавшая ЭВМ полностью приобретает способность выполнять заданные функции по обработке информации. Восстановление отказавшей ЭВМ может быть осуществлено автоматически (в общем случае с помощью аппаратурно-программных средств) или полуавтоматически (с участием бригады технического обслуживания). Ниже способ восстановления отказавшей машины различать не будем, а будем считать, что оно производится некоторым *восстанавливающим устройством* (ВУ). Следовательно, переход отка-

завшей ЭВМ в работоспособное состояние может произойти лишь в результате работы ВУ или, говоря точнее, после ремонта машины восстанавливающим устройством.

Для характеристики качества работы ЭВМ используют систему показателей надежности, каждый из которых определяется через понятия отказа и (или) восстановления. Прежде чем дать определения показателей надежности ЭВМ, введем случайные функцию $\omega(\tau)$ и величину ξ. Пусть

$$\omega(\tau) = \begin{cases} 1, \text{ если в момент времени } \tau \geqslant 0 \\ \quad \text{ЭВМ находится в работоспособном состоянии;} \\ 0, \text{ если в момент времени } \tau \geqslant 0 \\ \quad \text{ЭВМ находится в неработоспособном состоянии.} \end{cases} \tag{2.9}$$

Назовем $\omega(\tau)$ производительностью ЭВМ в момент времени $\tau \geqslant 0$ и пусть ξ является моментом, когда возникает первый отказ при работе ЭВМ в заданных условиях эксплуатации.

2.8.2. Функция надежности ЭВМ

Функция надежности (или вероятность безотказной работы) ЭВМ относится к основным показателям и характеризует производительность ЭВМ на заданном промежутке времени или, говоря иначе, характеризует способность ЭВМ обеспечить на промежутке времени потенциально возможную производительность.

Функцией надежности ЭВМ называется функция

$$r(t) = P\{\forall \tau \in [0, t) \rightarrow \omega(\tau) = 1\}.$$

Здесь $P\{\forall \tau \in [0, t) \rightarrow \omega(\tau) = 1\}$ — вероятность того, что для всякого τ, принадлежащего промежутку времени $[0, t)$, производительность $\omega(\tau)$ ЭВМ равна единице (2.9), т. е. равна потенциально возможной (см. § 2.6).

Для доказательства свойств функции $r(t)$ и определения показателей надежности, производных от $r(t)$, удобнее дать следующее определение:

$$r(t) = P\{\xi > t\}, \tag{2.10}$$

где $P\{\xi > t\}$ — вероятность события $\{\xi > t\}$, состоящего в том, что момент ξ возникновения первого отказа при работе ЭВМ в заданных условиях эксплуатации наступит после времени $t \geqslant 0$.

Функция $r(t)$ обладает следующими свойствами:

1) $r(0) = 1$; действительно, так как событие $\xi > 0$ (т. е. событие, заключающееся в том, что в момент начала функционирования ЭВМ работоспособна) считается достоверным, то $P\{\xi > 0\} = 1$;

2) $r(+\infty) = 0$; поскольку событие $\xi > (+\infty)$ считается невозможным (или, говоря другими словами, событие, заключающееся в том, что ЭВМ работоспособна на конечном промежутке времени, является достоверным), то, следовательно, $P\{\xi > (+\infty)\} = 0$;

3) $r(t_1) \geqslant r(t_2)$ для $t_1 \leqslant t_2$; в самом деле, события $\xi > t_2$ и $t_2 \geqslant \xi > t_1$ являются не совместными, поэтому по теореме сложения вероятностей имеем

$$r(t_1) = P\{\xi > t_1\} = P\{(\xi > t_2) \cup (t_2 \geqslant \xi > t_1)\} =$$
$$= P\{\xi > t_2\} + P\{t_2 \geqslant \xi > t_1\} \geqslant P\{\xi > t_2\} = r(t_2).$$

Функцией ненадежности (или вероятностью отказа) ЭВМ называется

$$q(t) = 1 - r(t).$$

Функцию $q(t)$ можно рассматривать как интегральную функцию распределения случайной величины ξ. Для оценки $q(t)$ на практике пользуются формулой

$$q(t) \approx \tilde{q}(t) = n(t)/N,$$

где N — число работоспособных ЭВМ в начале испытаний; $n(t)$ — число отказавших машин в промежутке времени $[0, t)$.

Функция $q(t)$ позволяет определить такие производные показатели надежности ЭВМ, как среднее время безотказной работы (средняя наработка до отказа) и интенсивность отказов машины. По определению *среднее время* ϑ *безотказной работы* ЭВМ и оценка $\tilde{\vartheta}$ соответственно равны:

$$\vartheta = \int_0^\infty t dq(t) = -\int_0^\infty t dr(t) = -tr(t)\Big|_0^\infty + \int_0^\infty r(t)dt = \int_0^\infty r(t)dt; \quad \tilde{\vartheta} = \frac{1}{N}\sum_{i=1}^{N} t_i, \quad (2.11)$$

где t_i — время безотказной работы i-й машины, $i \in \{1, 2, \ldots, N\}$.

Интенсивностью отказов (лямбда-характеристикой) ЭВМ называется функция

$$\lambda(t) = \frac{1}{1-q(t)}\frac{dq(t)}{dt} = -\frac{1}{r(t)}\frac{dr(t)}{dt}. \quad (2.12)$$

На практике для определения $\lambda(t)$ используют кусочно-линейную аппроксимацию

$$\lambda(t) \approx \tilde{\lambda}(t) = n(\Delta t)/[N(t)\Delta t]. \qquad (2.13)$$

Здесь $n(\Delta t)$ — число отказавших ЭВМ в промежутке времени $[t, t+\Delta t)$; $N(t)$ — число безотказно работающих ЭВМ в момент t. В самом деле, подставив в (2.13) оценки

$$n(\Delta t) = n(t+\Delta t) - n(t) \approx N[q(t+\Delta t) - q(t)];$$
$$N(t) = N - n(t) \approx N[1 - q(t)]$$

и осуществив соответствующий предельный переход при $\Delta t \to 0$, получим формулу (2.12).

Интегрируя от 0 до t выражение (2.12), получаем

$$\int_0^t \lambda(\tau)d\tau = -\ln r(t); \qquad r(t) = \exp\left[-\int_0^t \lambda(\tau)d\tau\right].$$

Практически установлено, что зависимость интенсивности отказов от времени имеет место на периоде приработки ЭВМ. После приработки ЭВМ интенсивность отказов остается постоянной (до вхождения в предельное состояние или, по крайней мере, в течение промежутка времени, перекрывающего время морального старения). Следовательно, в нормальных условиях эксплуатации ЭВМ $\lambda = \text{const}$, а функция надежности (2.10) и математическое ожидание времени безотказной работы (2.11) соответственно равны:

$$r(t) = \exp(-\lambda t); \qquad \vartheta = \int_0^\infty e^{-\lambda t}\,dt = -\frac{1}{\lambda}e^{-\lambda t}\Big|_0^\infty = \frac{1}{\lambda}. \qquad (2.14)$$

Таким образом, время безотказной работы ЭВМ подчиняется экспоненциальному закону. Следовательно, величина λ — *среднее число отказов, появляющихся в машине в единицу времени.*

Функция $r(t)$ есть вероятность того, что в ЭВМ произойдет ноль отказов за время t. Тогда вероятность появления в ЭВМ k отказов за время t будет равна

$$r_k(t) = \frac{(\lambda t)^k}{k!}e^{-\lambda t}; \qquad \sum_{k=0}^{\infty} r_k(t) = 1, \qquad (2.15)$$

где $r_0(t) = r(t)$. В самом деле, среднее число отказов, появляющихся на промежутке времени $[0, t)$, равно

$$\sum_{k=1}^{\infty} k r_k(t) = e^{-\lambda t}\lambda t \sum_{k=1}^{\infty} \frac{(\lambda t)^{k-1}}{(k-1)!} = \lambda t, \qquad (2.16)$$

так как последняя сумма в (2.16), очевидно, равна $\exp \lambda t$. *Таким образом, поток отказов в ЭВМ является пуассоновским, или простейшим* (2.15).

2.8.3. Функция восстановимости ЭВМ

Функция восстановимости (или вероятность восстановления работоспособного состояния) ЭВМ — основной показатель, характеризующий надежностные способности и ЭВМ, и восстанавливающего устройства одновременно. Или, говоря иначе, эта количественная характеристика дает информацию о том, как приспособлена машина к восстановлению своей производительности (после отказа) с помощью ВУ. *Функцией восстановимости* ЭВМ назовем функцию

$$u(t) = 1 - P\{\forall \tau \in [0, t) \to \omega(\tau) = 0\}, \qquad (2.17)$$

где $P\{\forall \tau \in [0, t) \to \omega(\tau) = 0\}$ — вероятность того, что (при выполнении восстановительных работ в машине) для всякого τ, принадлежащего промежутку времени $[0, t)$, производительность $\omega(\tau)$ ЭВМ остается равной нулю (2.9) (или есть вероятность того, что отказавшая ЭВМ при работе восстанавливающего устройства не будет восстановлена за время t).

Для функции $u(t)$ справедливо: 1) $u(0) = 0$; 2) $u(+\infty) = 1$; 3) $u(t_1) \leqslant u(t_2)$ для $t_1 \leqslant t_2$. Следовательно, $u(t)$ является интегральной функцией распределения времени восстановления отказавшей ЭВМ.

Для практической оценки вероятности восстановления ЭВМ на промежутке времени $[0, t)$ используют формулу

$$u(t) \approx \tilde{u}(t) = m(t)/M.$$

Здесь M — число отказавших машин в начале восстановления; $m(t)$ — число восстановленных машин за время t при условии, что ремонт каждой ЭВМ осуществляется своим ВУ.

Основываясь на практическом материале по эксплуатации ЭВМ и применяя статистические критерии о достоверности гипотез относительно распределения случайных величин, можно доказать справедливость формул

$$u(t) = 1 - \exp(-\mu t); \quad \tau = \int_0^{\infty} t\,du(t) = 1/\mu, \qquad (2.18)$$

где τ — *среднее время восстановления работоспособного состояния ЭВМ*; μ — *интенсивность восстановления ЭВМ*, или среднее число восстановлений ЭВМ, которое может произвести ВУ в единицу времени.

Примечания. 1. Проведение статистических экспериментов, которые бы позволили определить $\tilde{q}(t)$, $\tilde{\vartheta}$ и $\tilde{\lambda}(t)$, для машин первого и второго поколений и для мощных ЭВМ третьего поколения было невозможным (из-за их большой стоимости и мелкосерийности производства). Только после освоения массового производства мини-ЭВМ и особенно микроЭВМ появилась принципиальная возможность проведения таких экспериментов (по крайней мере, опытов на статистически достаточном количестве ЭВМ простейших конфигураций, т. е. машин, каждая из которых состояла из микропроцессора и памяти).

2. При определении оценок $\tilde{q}(t)$ и $\tilde{\vartheta}$ для функции ненадежности $q(t)$ и среднего времени ϑ безотказной работы ЭВМ целесообразно применять *эргодическую гипотезу*. Эта гипотеза позволяет вместо статистических результатов наблюдения за большим числом N машин воспользоваться результатами наблюдения за одной машиной в течение длительного времени.

3. Справедливость экспоненциального закона (2.14) распределения времени безотказной работы ЭВМ подтверждена обработкой статистических данных по эксплуатации ЭВМ первого-третьего поколений. При проверке этой гипотезы был применен критерий согласия χ^2 Пирсона.

4. Среднее время безотказной работы ϑ (2.11) современных микропроцессорных ЭВМ оценивается многими годами, например, для электронной части ЭВМ оно составляет $10^5 \ldots 10^8$ ч. Поэтому для определения оценок показателей надежности ЭВМ разработаны методики ускоренных экспериментов (например, использующие нагревание интегральных схем).

2.8.4. Функция готовности ЭВМ

Функции надежности (2.10) и восстановимости (2.17) характеризуют на промежутке времени $[0, t)$ возможности ЭВМ соответственно по обеспечению потенциальной производительности и по достижению этой производительности после отказа. Первый показатель связан с понятием отказа, а второй — с понятием восстановления ЭВМ. Следовательно, функция надежности позволяет пользователю оценить возможность решения той или иной задачи на ЭВМ (оценить, с какой вероятностью при наличии отлаженной программы и при известной оценке времени решения задача может быть пропущена через машину). Функция восстановимости информирует пользователя о том, с какой вероятностью отказавшая ЭВМ к заданному времени будет восстановлена (и, в частности, позволяет пользователю определить ожидаемое время простоя ЭВМ после отказа).

Функции надежности и восстановимости ЭВМ характеризуют поведение машины только на начальном этапе ее работы (на конечном промежутке времени $[0, t)$) или, как говорят, *в переходном режиме функционирования*. Эти показатели не информативны при оценке работы ЭВМ в течение длительного времени или в стационарном режиме функционирования. Действительно, для стационарного режима, т. е. для режима, который устанавливается при $t \to \infty$, имеем

$$\lim_{t\to\infty} r(t) = 0; \quad \lim_{t\to\infty} u(t) = 1.$$

Из сказанного следует, что требуется комплексный показатель надежности ЭВМ, который бы характеризовал производительность ЭВМ и в переходном, и в стационарном режимах и который был бы связан одновременно с понятиями отказа и восстановления. Прежде чем определить такой показатель, введем обозначения: $E_0^1 = \{0, 1\}$ — множество состояний ЭВМ, причем $i = 0$ соответствует состоянию отказа, а $i = 1$ — работоспособному состоянию машины; $P_j(i, t)$– вероятность нахождения ЭВМ в момент $t \geqslant 0$ в состоянии $j \in E_0^1$ при условии, что ее начальным было состояние $i \in E_0^1$.

В качестве показателя, позволяющего достичь вышепоставленной цели, можно использовать *функцию готовности ЭВМ*

$$s(i, t) = P_1(i, t) = P\{i; \omega(t) = 1\}, \qquad (2.19)$$

где $P\{i; \omega(t) = 1\}$ — вероятность того, что (в условиях потока отказов и восстановлений) машина, начавшая функционировать в состоянии $i \in E_0^1$, будет иметь в момент времени $t \geqslant 0$ производительность, равную единице (т. е. равную потенциально возможной, см. (2.9)).

Функция готовности ЭВМ обладает следующими свойствами:

1) $s(0, 0) = 0, \ s(1, 0) = 1$;

2) $s(i, +\infty) = s = \text{const}, \ 0 < s < 1, \ i \in E_0^1$;

3) $s(0, t_1) \leqslant s(0, t_2), \ s(1, t_1) \geqslant s(1, t_2)$ для $t_1 \leqslant t_2$.

Смысл этих свойств интуитивно ясен. Дадим физические пояснения к этим свойствам.

Функция готовности (2.19) одновременно учитывает и отказы, и восстановления и характеризует производительность ЭВМ не на промежутке времени $[0, t)$, а в *момент* $t \geqslant 0$, следовательно, в качестве ее начального значения (начального состояния ЭВМ) может быть взято одно из возможных значений 0 или 1 (одно из состояний ЭВМ: «ЭВМ отказала», $i = 0$, или «ЭВМ работоспособна», $i = 1$). Из сказанного и из определения

(2.19) вытекает справедливость первого свойства функции готовности ЭВМ. Далее, относительно второго свойства функции готовности ЭВМ: $s(i, t)$ характеризует поведение ЭВМ в любой момент $t \geqslant 0$, т. е. не только в переходном, но и в стационарном режимах работы. Ясно, что в режиме длительной эксплуатации ЭВМ ($t \to \infty$) при наличии ЗУ (процедуры восстановления) вероятность отказа не равна единице, следовательно, можно записать

$$s(i, +\infty) = \lim_{t \to \infty} s(i, t) = s = \text{const},$$

где s не зависит от начального состояния $i \in E_0^1$ машины, $0 < s < 1$. Равенство $s = \text{const}$ объясняется следующим: поскольку за время $t = +\infty$ происходят и отказы, и восстановления ЭВМ, то «забывается» предыстория, т. е. «теряется» зависимость от $i \in E_0^1$. Величина s называется *коэффициентом готовности ЭВМ*. Наконец, справедливость третьего свойства видна из первых двух свойств функции готовности ЭВМ.

Говоря иначе, функция готовности есть вероятность того, что ЭВМ в момент времени $t \geqslant 0$ работоспособна (т. е. вероятность того, что ЭВМ способна выполнять возлагаемые на нее функции). Следовательно, функция готовности несет информацию о том, может ли пользователь начать работу на ЭВМ в данный момент времени (и в переходном, и в стационарном режимах функционирования). Если же ЭВМ общего назначения и находится в постоянной эксплуатации, то пользователь может оценить возможность решения задач на ней только по коэффициенту готовности.

Выведем дифференциальное уравнение для расчета функции готовности ЭВМ. Пусть Δt — промежуток времени бесконечно малой длины. Оценим вероятность того, что ЭВМ в момент $(t + \Delta t)$ находится в работоспособном состоянии. Последнее событие сложное и может наступить, если произойдет, например, одно из несовместных событий:

1) ЭВМ при $t \geqslant 0$ неработоспособна, а на промежутке времени $[t, t + \Delta t)$ она будет восстановлена;

2) ЭВМ при $t \geqslant 0$ работоспособна, и на промежутке времени $[t, t + \Delta t)$ она не откажет.

Очевидно, что события, составляющие любое из этих несовместных событий, являются независимыми.

Оценим вероятности изучаемых событий Вероятности того, что ЭВМ при $t \geqslant 0$ неработоспособна или работоспособна, соответственно равны $P_0(i, t) = 1 - s(i, t)$ или $P_1(i, t) = s(i, t)$, $i \in E_0^1$ (см. (2.19)). Вероятность того, что отказавшая ЭВМ за время Δt будет восстановлена (2.18), равна $u(\Delta t)$ или, применяя разложение функции восстановимости в ряд Маклорена, имеем

$$u(\Delta t) = u(0) + \frac{u'(0)}{1!}\Delta t + \frac{u''(0)}{2!}(\Delta t)^2 + \ldots = 1 - e^{-\mu\Delta t} =$$

$$= 0 + \frac{1}{1!}\mu\Delta t + o(\Delta t) = \mu\Delta t + o(\Delta t), \qquad (2.20)$$

где $o(\Delta t)$ — величина более высокого порядка малости, чем Δt. Аналогично действуя, находим вероятность того, что ЭВМ за время Δt не откажет:

$$r(\Delta t) = r(0) + \frac{r'(0)}{1!}\Delta t + \frac{r''(0)}{2!}(\Delta t)^2 + \ldots = e^{-\lambda\cdot\Delta t} = 1 - \lambda\Delta t + o(\Delta t). \qquad (2.21)$$

Заметим, что ЭВМ может оказаться в момент $t + \Delta t$ в работоспособном состоянии и не только в результате наступления описанных выше событий. Например, при условии, что ЭВМ при $t \geqslant 0$ работоспособна, наступление сложного события, заключающегося в том, что на промежутке времени $[t, t + \Delta t)$ происходят отказ и восстановление ЭВМ, приводит к интересующему нас состоянию. Однако вероятность того, что данное сложное событие произойдет, равна

$$r_1(\Delta t)u(\Delta t) = \lambda\Delta t[1 - \lambda\Delta t + o(\Delta t)][\mu\Delta t + o(\Delta t)] = o(\Delta t),$$

что следует из (2.15), (2.20) и (2.21). Легко убедиться в том, что если ЭВМ в момент $(t + \Delta t)$ оказывается в работоспособном состоянии в результате наступления на промежутке времени $[t, t + \Delta t)$ более одного из событий «отказ» и «восстановление», то изучаемые вероятности также имеют вид $o(\Delta t)$.

Таким образом, вероятность нахождения ЭВМ в момент времени $(t + \Delta t)$ в работоспособном состоянии будет равна

$$s(i, t + \Delta t) = [1 - s(i, t)]u(\Delta t) + s(i, t)r(\Delta t) + o(\Delta t) =$$
$$= [1 - s(i, t)]\mu\Delta t + s(i, t)(1 - \lambda\Delta t) + o(\Delta t) \qquad (2.22)$$

(в силу независимости и несовместности рассматриваемых событий). Перенеся $s(i, t)$ в левую часть в (2.22), разделив на Δt и перейдя к пределу при $\Delta t \to 0$ в обеих частях рассматриваемого равенства, получим следующее дифференциальное уравнение:

$$\frac{ds(i, t)}{dt} = \mu - (\lambda + \mu)s(i, t), \quad i \in \{0, 1\}. \qquad (2.23)$$

Решениями уравнения (2.23), как легко убедиться путем подстановок, при начальных состояниях ЭВМ $i = 0$, $i = 1$ соответственно будут функции:

$$s(0,t)=\frac{\mu}{\lambda+\mu}-\frac{\mu}{\lambda+\mu}e^{-(\lambda+\mu)t}; \tag{2.24}$$

$$s(1,t)=\frac{\mu}{\lambda+\mu}+\frac{\lambda}{\lambda+\mu}e^{-(\lambda+\mu)t} \tag{2.25}$$

Полученные формулы (2.24), (2.25) позволяют оценить готовность ЭВМ в переходном режиме функционирования. Если достаточно ограничиться анализом стационарного режима работы ЭВМ, то вместо (2.24) и (2.25) можно использовать предельно простую формулу для коэффициента готовности ЭВМ:

$$s=\lim_{t\to\infty}s(i,t)=\mu/(\lambda+\mu), \tag{2.26}$$

в которой нет зависимости от начального состояния ЭВМ $i\in E_0^1$. Уместно заметить, что ЭВМ «быстро» входят в стационарный режим работы, поэтому на практике, как правило, следует использовать не функцию готовности, а коэффициент s (2.26).

Замечание. На практике и в технической литературе довольно часто используется выражение типа «Надежность ЭВМ равна 0,999», однако это жаргонное выражение, которое не имеет однозначного толкования. Под «надежностью» можно понимать значения или функции надежности, или функции готовности ЭВМ для некоторого времени, но чаще всего под «надежностью» понимают коэффициент готовности машины.

2.8.5. Функция осуществимости решения задач на ЭВМ

Цель функционирования ЭВМ — решение поступивших задач (выполнение программ решения задач). Однако введенные показатели надежности ЭВМ (2.10), (2.17), (2.19) устанавливают взаимосвязь лишь между потенциально возможной производительностью и надежностью машины (безотказностью, ремонтопригодностью, готовностью), т. е. характеризуют качество функционирования ЭВМ безотносительно к процессу решения задач. Этот пробел можно устранить, если использовать для характеристики работы ЭВМ *функцию осуществимости решения задач*

$$f(t)=r(t)\varphi(t),$$

где $r(t)$ — вероятность безотказной работы ЭВМ (2.10); $\varphi(t)=P\{0\leqslant\eta<t\}$, т. е. $\varphi(t)$ — вероятность события $\{0\leqslant\eta<t\}$; τ — случайная величина, являющаяся моментом решения задачи на работоспособной (абсолютно надежной) ЭВМ. В качестве закона распределения времени решения задач на ЭВМ может быть взят экспоненциальный

$$\varphi(t) = 1 - \exp(-\beta t).$$

Здесь β — интенсивность решения задач на машине.

Говорят, что решение задачи на ЭВМ осуществимо, если для некоторого t одновременно выполняются $f(t) \geqslant f^{\circ}$ и $t \leqslant t^{\circ}$, где f° и t° — «пороги осуществимости»; они выбираются из практических соображений. Интерес представляет также величина $f(t_m) = \max\limits_{t} f(t)$, которая определяется численными методами.

2.9. Технико-экономический анализ функционирования ЭВМ

К технико-экономическим относятся такие показатели ЭВМ, которые характеризуют экономическую сторону разработки, производства, ввода в действие и эксплуатации машины как единого комплекса аппаратурно-программных средств. Ниже будут введены стоимостные параметры, связанные с эксплуатацией ЭВМ, и цена операции; детально рассмотрен аналитический подход к технико-экономическому анализу функционирования машины; изучена взаимосвязь между надежностью и стоимостью ЭВМ.

2.9.1. Стоимость эксплуатации ЭВМ

Рассмотрим суммарные расходы V_1, связанные с эксплуатацией ЭВМ в течение достаточно длительного времени $\mathfrak{T}$:

$$V_1 = \sum_{i=1}^{6} v_i.$$

Здесь компоненты v_i определяются за время $\mathfrak{T}$, причем:

v_1 — стоимость амортизации ЭВМ и вспомогательного оборудования для нее; $v_1 = kv$, v — *цена машины* и вспомогательного оборудования для нее; k — коэффициент амортизации;

v_2 — стоимость содержания восстанавливающего устройства (бригады обслуживания);

v_3 — стоимость запасных технических средств (материалов, деталей, приборов, интегральных схем и типовых элементов замены и т. п.), расходуемых при устранении отказов в ЭВМ;

v_4 — стоимость вспомогательных средств (бумаги, картриджей, дискет, компакт-дисков и т. д.), необходимых для нормального функционирования ЭВМ;

v_5 — стоимость электроэнергии, потребляемой при эксплуатации ЭВМ и вспомогательного оборудования;

v_6 — накладные расходы (амортизация помещения и т. д.).

Величину T — часть времени $\Im$, использованную на решение задач, будем считать полезным временем эксплуатации ЭВМ.

Отношение $c_1' = V_1 / T$ будет себестоимостью единицы полезного времени при эксплуатации машины.

Стоимостью эксплуатации ЭВМ назовем

$$c_1 = (V_1 + V_2)/T \tag{2.27}$$

где $V_2 = v_7 + v_8$, v_7 — плановая прибыль от эксплуатации машины; v_8 — отчисления в фонд развития (т. е. на расширение аппаратурно-программного сервиса: приобретение и подключение новых технических средств, составление стандартных программ, разработку пакетов прикладных программ и языков программирования и т. д.) за время $\Im$.

Себестоимость и стоимость содержания восстанавливающего устройства в единицу времени определяются выражениями:

$$c_2' = v_2 / \Im; \quad c_2 = v_2 / T. \tag{2.28}$$

Стоимость запасных технических средств, расходуемых при однократном восстановлении отказавшей ЭВМ

$$c_3 = v_3(\lambda + \mu)(\Im\lambda\mu)^{-1} \tag{2.29}$$

Здесь λ — интенсивность отказов ЭВМ; μ — интенсивность восстановления ЭВМ восстанавливающим устройством. В самом деле, среднее время между двумя восстановлениями ЭВМ равно $\lambda^{-1} + \mu^{-1} = (\lambda + \mu)(\lambda\mu)^{-1}$ и, следовательно, число восстановлений за время $\Im$ составляет $\Im\lambda\mu(\lambda + \mu)^{-1} = l$. Тогда стоимость $c_3 = v_3 / l$, что и доказывает справедливость (2.29).

2.9.2. Цена быстродействия ЭВМ

В отечественных и зарубежных исследованиях выявлен эмпирический закон, устанавливающий взаимосвязь между производительностью и стоимостью ЭВМ. Этот закон называют законом Гроша (Grosch's Law) и записывают в следующем виде:

$$\omega = hv^a,$$

где ω — показатель производительности машины (как правило, ω — номинальное быстродействие ЭВМ, см. (2.5)); v — цена машины; константа $a \geqslant 2$ и коэффициент h, имеющий размерность, зависят от технологии производства. Закон Гроша используют при оценке качества проектирования средств вычислительной техники. Следует заметить, что он имеет силу в ограниченном диапазоне производительности (до 10^8 опер./с), точнее, для вычислительных средств, структура которых близка к ЭВМ Дж. фон Неймана. При переходе к параллельным структурам вычислительных средств закон Гроша теряет силу.

Широкое распространение получила на практике количественная характеристика ЭВМ $\sigma = v/\omega$, которую называют *ценой* (одной) *операции* (в секунду). Если в качестве ω рассматривается не номинальное быстродействие (2.5), а среднее быстродействие (2.4), то величина σ называется *ценой быстродействия ЭВМ.*

2.9.3. Математическое ожидание бесполезных расходов при эксплуатации ЭВМ

В процессе эксплуатации ЭВМ ее состояния чередуются. Очевидно, что если машина работоспособна, то простаивает ВУ; появление отказа в ЭВМ прекращает простой ВУ, но приводит к простою самой машины. Эксплуатационные расходы (потери), вызванные этими простоями, будем считать бесполезными [6].

Пусть $P_{lk}(\Delta t)$ — условная вероятность того, что ЭВМ, находящаяся в момент времени $t \geqslant 0$ в состоянии l, перейдет через время Δt в состояние k; $l, k \in E_0^1$. Пусть также $\Gamma(i,t)$ — *математическое ожидание бесполезных эксплуатационных расходов к моменту* $t > 0$ *при условии, что в начальный момент* (при $t = 0$) *машина находилась в состоянии* $i \in E_0^1$. Выведем дифференциальное уравнение для функции $\Gamma(i,t)$, $i \in E_0^1$. Средние расходы $\Gamma(i, t+\Delta t)$, $i \in E_0^1$, при эксплуатации ЭВМ в течение времени $t+\Delta t$ (где Δt — бесконечно малый промежуток времени) складываются из затрат за время t и издержек, которые имеют место на промежутке времени $[t, t+\Delta t)$.

Вероятности $P_0(i,t)$ или $P_1(i,t)$ того, что ЭВМ при $t \geqslant 0$ неработоспособна или работоспособна, соответственно равны $[1-s(i,t)]$ или $s(i,t)$, $i \in E_0^1$ (см. формулу (2.19)); вероятность $P_{00}(\Delta t)$ того, что отказавшая ЭВМ на промежутке времени $[t, t+\Delta t)$ не будет восстановлена, равна $(1-u(\Delta t))$, (см. формулу (2.17)), а вероятность $P_{11}(\Delta t)$ того, что работоспособная ЭВМ на этом же промежутке не откажет, равна $r(\Delta t)$ (см. формулу (2.10)).

Если ЭВМ в момент времени $t \geqslant 0$ неработоспособна и на промежутке времени $[t, t+\Delta t)$ не будет восстановлена, то потери вследствие отказа машины составят

$$P_0(i,t)P_{00}(\Delta t)c_1\Delta t + o(\Delta t) = [1 - s(i,t)][1 - u(\Delta t)]c_1\Delta t + o(\Delta t),\ i \in E_0^1. \quad (2.30)$$

Если же машина в момент времени $t \geqslant 0$ работоспособна и сохранит свое состояние и в последующие Δt единиц времени, то бесполезные эксплуатационные расходы определяются простоем ВУ:

$$P_1(i,t)P_{11}(\Delta t)c_2\Delta t + o(\Delta t) = s(i,t)r(\Delta t)c_2\Delta t + o(\Delta t),\ i \in E_0^1. \quad (2.31)$$

Учитывая выражения (2.30) и (2.31), можно записать

$$\Gamma(i, t+\Delta t) = \Gamma(i,t) + [1 - s(i,t)][1 - u(\Delta t)]c_1\Delta t + s(i,t)r(\Delta t)c_2\Delta t + o(\Delta t),\ i \in E_0^1. \quad (2.32)$$

Подставив в (2.32) оценки (2.20) и (2.21) для $u(\Delta t)$ и $r(\Delta t)$, получим

$$\Gamma(i, t+\Delta t) = \Gamma(i,t) + [1 - s(i,t)][1 - \mu\Delta t + o(\Delta t)]c_1\Delta t + \\ + s(i,t)[1 - \lambda\Delta t + o(\Delta t)]c_2\Delta t + o(\Delta t),\ i \in E_0^1. \quad (2.33)$$

Далее, если в (2.33) $\Gamma(i,t)$ перенести в левую часть, затем поделить на Δt обе части равенства и перейти к пределу при $\Delta t \to 0$, то получим дифференциальное уравнение

$$\frac{d}{dt}\Gamma(i,t) = [1 - s(i,t)]c_1 + s(i,t)c_2,\ i \in E_0^1; \quad (2.34)$$

в качестве начальных условий естественно взять $\Gamma(i,0) = 0$.

Решение дифференциального уравнения (2.34) при заданных начальных условиях имеет вид

$$\Gamma(i,t) = -\varepsilon_i + \gamma t + \varepsilon_i\delta(t), \quad (2.35)$$

где

$$\gamma = \frac{\lambda}{\lambda+\mu}c_1 + \frac{\mu}{\lambda+\mu}c_2; \quad (2.36)$$

$$\delta(t) = e^{-(\lambda+\mu)t}; \quad (2.37)$$

$$\varepsilon_0 = -\frac{\mu}{(\lambda+\mu)^2}(c_1 - c_2);\ \ \varepsilon_1 = \frac{\lambda}{(\lambda+\mu)^2}(c_1 - c_2). \quad (2.38)$$

В справедливости формул (2.35)–(2.38) легко убедиться, подставив $s(i,t)$ из выражений (2.24), (2.25) в формулу (2.34) и проинтегрировав.

Для невосстанавливаемой ЭВМ, т. е. для случая, когда отсутствует ВУ, $\mu = 0$, $c_2 = 0$ и формулы (2.35)–(2.38) дают следующий результат:

$$\Gamma(0, t) = c_1 t;\ \Gamma(1, t) = (-1/\lambda)c_1 + c_1 t + (1/\lambda)c_1 e^{-\lambda t} \qquad (2.39)$$

Заметим, что первое слагаемое в правой части $\Gamma(1, t)$ исключает из бесполезных потерь «доход», который можно ожидать от эксплуатации ЭВМ. Действительно, работоспособная ЭВМ функционирует до отказа в среднем $1/\lambda$ единиц времени, следовательно, до отказа ЭВМ может принести доход в размере c_1/λ.

Формулы (2.35)–(2.39) характеризуют поведение ЭВМ и в переходном, и в стационарном режимах функционирования. Однако в стационарном режиме, т. е. при длительной эксплуатации ЭВМ (или математически при $t \to \infty$), функция $\delta(t)$ (2.37) становится пренебрежимо малой, т. е. $\delta(t) \to 0$. Кроме того, первое слагаемое (2.38) в правой части формулы (2.35) становится пренебрежимо малым по сравнению со вторым, т. е. с γt. Таким образом, математическое ожидание бесполезных эксплуатационных расходов за время t для стационарного режима работы ЭВМ не зависит от начального состояния машины и имеет вид

$$\Gamma(t) = \gamma t,$$

где γ вычисляется по (2.36), а в случае отсутствия восстанавливающего устройства $\gamma = c_1$ (см. формулу (2.39)).

Величину γ назовем *коэффициентом эксплуатационных потерь ЭВМ.* Очевидно, что γ есть математическое ожидание потерь в единицу времени в стационарном режиме из-за отказов машины и простоя восстанавливающего устройства. Следовательно, коэффициент γ можно рассчитать без вывода дифференциального уравнения (2.34), если известны вероятности состояний ЭВМ.

В самом деле, пусть P_0 — вероятность того, что ЭВМ (в стационарном режиме) находится в состоянии отказа или, говоря иначе, вероятность того, что ВУ занято; P_1 — вероятность того, что ЭВМ (в стационарном режиме) работоспособна или вероятность того, что ВУ простаивает. Тогда

$$\gamma = \sum_{i=0}^{1} P_i c_{i+1},$$

где c_1 и c_2 определяются по формулам (2.27) и (2.28). Подставив в последнюю формулу $P_0 = 1 - s$, $P_1 = s$ и (2.26), получим известное выражение (2.36) для коэффициента эксплуатационных потерь ЭВМ.

Задача минимизации функции $\Gamma(i, t)$, $i \in E_0^1$ (2.35) и, в частности, коэффициента γ (2.36) решается численными методами и с использованием экспериментальных результатов. Так, при $c_1 = \text{const}$, $\lambda = \text{const}$ экспериментально должны быть подобраны такие c_2 и μ, чтобы коэффициент γ принимал минимальное значение. Последнее может быть достигнуто, например, подбором состава средств обслуживания ЭВМ, т. е. средств контроля, диагностики и восстановления (или реконфигурирования).

2.9.4. Математическое ожидание дохода ЭВМ

Введем функцию $D(i, t)$, $i \in E_0^1$, значения которой $D(0, t)$ и $D(1, t)$ являются *математическими ожиданиями дохода, приносимого при эксплуатации ЭВМ в течение времени* $t \geqslant 0$, при условиях, что машина в момент начала функционирования находилась в неработоспособном и работоспособном состояниях соответственно. Пусть d_{ii} — доход, приносимый ЭВМ за единицу времени при пребывании машины в состоянии $i \in E_0^1$, d_{ij} — доход, приносимый ЭВМ при переходе из состояния i в состояние j, $i \neq j$; $i, j \in E_0^1$.

Математическое ожидание дохода $D(i, t)$, $i \in E_0^1$, может быть рассчитано классически, с помощью аппарата марковских процессов с доходами [8], и упрощенно [6], аналогично тому, как это сделано в разд. 2.9.3.

Способ 1. При использовании марковских процессов с доходами отсчет времени будем вести в обратном направлении, т. е. будем считать, что процесс начинается в момент $(t + \Delta t)$, а оканчивается в момент $t = 0$. Тогда ожидаемый доход ЭВМ за время $(t + \Delta t)$ равен

$$D(i, t + \Delta t) = P_{ii}(\Delta t)[d_{ii}\Delta t + D(i, t)] + P_{ij}(\Delta t)[d_{ij} + D(j, t)], \quad i \neq j; \quad i, j \in E_0^1. \tag{2.40}$$

В самом деле, в течение времени Δt, т. е. на промежутке времени $(t + \Delta t, t]$, ЭВМ может либо остаться в состоянии i, либо перейти в состояние $j = (1 - i)$, $i \in E_0^1$. Если ЭВМ остается в состоянии $i \in E_0^1$ в течение времени Δt, то доход составит величину $d_{ii}\Delta t$ плюс ожидаемый доход $D(i, t)$, который она принесет за оставшиеся t единиц времени. Если машина перейдет из состояния $i \in E_0^1$ в состояние $j = 1 - i$, то доход составит величину d_{ij} плюс ожидаемый доход $D(j, t)$ за оставшееся время t, если бы начальным было состояние j.

ЭВМ общего назначения — восстанавливаемые, следовательно, возможны переходы из состояния отказа в исправное состояние, т. е. из состояния 0 в состояние 1. Тогда из (2.14) и (2.18) видны оценки:

$$\left.\begin{aligned} P_{00}(\Delta t) &= 1 - u(\Delta t) = e^{-\mu\Delta t} \approx 1 - \mu\Delta t; \\ P_{01}(\Delta t) &= u(\Delta t) = 1 - e^{-\mu\Delta t} \approx \mu\Delta t; \\ P_{10}(\Delta t) &= 1 - r(\Delta t) = 1 - e^{-\lambda\Delta t} \approx \lambda\Delta t; \\ P_{11}(\Delta t) &= r(\Delta t) = e^{-\lambda\Delta t} \approx 1 - \lambda\Delta t. \end{aligned}\right\} \qquad (2.41)$$

Положим, что

$$d_{00} = -c_2', \quad d_{01} = -c_3, \quad d_{11} = c_1 - c_2', \quad d_{10} = 0. \qquad (2.42)$$

Заметим, что в величине c_1 учитываются расходы на содержание ВУ, поэтому можно было бы взять $d_{11} = c_1$ и $d_{00} = 0$. Однако в условиях коммерческой эксплуатации при исправном состоянии машины ВУ (бригада обслуживания) простаивает, поэтому можно положить $d_{11} = c_1 - c_2'$. Кроме того, можно считать также, что если ЭВМ пребывает в состоянии отказа, а ВУ не способно изменить ее состояние (на работоспособное) в течение некоторого времени, то вычислительный центр бесполезно теряет в единицу времени сумму c_2'; следовательно, $d_{00} = -c_2'$.

Подставляя (2.41), (2.42) в (2.40) и пренебрегая членами более высокого порядка малости по сравнению с Δt, получаем

$$\left.\begin{aligned} \frac{D(0, t+\Delta t) - D(0, t)}{\Delta t} &= -(c_2' + \mu c_3) - \mu D(0, t) + \mu D(1, t); \\ \frac{D(1, t+\Delta t) - D(1, t)}{\Delta t} &= (c_1 - c_2') + \lambda D(0, t) - \lambda D(1, t). \end{aligned}\right\} \qquad (2.43)$$

Далее, переходя к пределу при $\Delta t \to 0$ в обеих частях равенств (2.43) и используя обозначения

$$d_0 = -c_2' - \mu c_3, \quad d_1 = c_1 - c_2',$$

получаем систему дифференциальных уравнений:

$$\left.\begin{aligned} \frac{d}{dt} D(0, t) &= d_0 - \mu D(0, t) + \mu D(1, t); \\ \frac{d}{dt} D(1, t) &= d_1 + \lambda D(0, t) - \lambda D(1, t). \end{aligned}\right\} \qquad (2.44)$$

Естественно задать следующие начальные условия: $D(0, 0) = 0$, $D(1, 0) = 0$.

Легко убедиться (путем подстановки), что решением системы уравнений (2.44) при заданных начальных условиях будет система

$$\left.\begin{aligned} D(0,t) &= -\frac{\mu(d_1-d_0)}{(\lambda+\mu)^2}+\frac{\lambda d_0+\mu d_1}{\lambda+\mu}t-\frac{\mu(d_1-d_0)}{(\lambda+\mu)^2}e^{-(\lambda+\mu)t};\\ D(1,t) &= \frac{\lambda(d_1-d_0)}{(\lambda+\mu)^2}+\frac{\lambda d_0+\mu d_1}{\lambda+\mu}t-\frac{\lambda(d_1-d_0)}{(\lambda+\mu)^2}e^{-(\lambda+\mu)t} \end{aligned}\right\} \qquad (2.45)$$

Формулы (2.45) в частном случае, когда отсутствует ВУ ($\mu = 0$, $c_2' = 0$), принимают следующий вид:

$$D(0,t) = 0,\quad D(1,t) = c_1\lambda^{-1} - c_1\lambda^{-1}e^{-\lambda t} \qquad (2.46)$$

Формулы (2.45) и (2.46) характеризуют функционирование восстанавливаемых и невосстанавливаемых ЭВМ в переходном режиме.

Исследуем поведение ЭВМ в стационарном режиме. Легко заметить, что при $t \to \infty$ значение $e^{-(\lambda+\mu)t} \to 0$, а первые слагаемые и для $D(0, t)$, и для $D(1, t)$ в (2.45) становятся пренебрежимо малыми по сравнению со вторыми. Таким образом, математическое ожидание дохода, приносимого ЭВМ за время t в стационарном режиме работы, не зависит от начального состояния машины и выражается функцией

$$D(t) = gt,$$

где

$$g = \frac{\lambda d_0+\mu d_1}{\lambda+\mu} = \frac{\mu}{\lambda+\mu}(c_1-\lambda c_3) - c_2'. \qquad (2.47)$$

Величину g — средний доход, приносимый в единицу времени при длительной работе ЭВМ, — называют *прибылью*.

Если же ЭВМ невосстанавливаемая, то с вероятностью, равной единице, машина при длительной эксплуатации окажется в состоянии отказа. Действительно, вероятность отказа ЭВМ за время t равна $q(t) = 1 - r(t)$ (см. (2.10)), а $\lim\limits_{t\to\infty} q(t) = 1$. Значит, невосстанавливаемая ЭВМ в стационарном режиме может приносить нулевую прибыль, $g = 0$ (см. (2.47)). Математическое ожидание дохода за длительное время эксплуатации невосстанавливаемой ЭВМ равно

$$D = \lim_{t\to\infty} D(1,t) = c_1/\lambda = \text{const},$$

что следует из (2.46). Или, рассуждая иначе: c_1 — стоимость одного часа полезной работы ЭВМ, а λ^{-1} — среднее время безотказной работы ЭВМ, значит, если

невосстанавливаемая машина начинает функционировать в работоспособном состоянии, то ожидаемый доход до момента отказа ЭВМ составит величину $c_1\lambda^{-1}$

Решение системы дифференциальных уравнений (2.44) можно определить с помощью операционного исчисления, например с помощью преобразования Лапласа—Карсона [9]. Пусть $\overline{D}(i, p)$ — изображение для $D(i, t)$, p — комплексный параметр, $i \in E_0^1$. Используя известные формулы

$$\frac{d}{dt}D(i, t) \sim p[\overline{D}(i, p) - D(i, 0)], \quad i \in \{0, 1\}; \tag{2.48}$$

$$\frac{p+\alpha}{p(p+a)} \sim \frac{a-\alpha}{a^2} + \frac{\alpha}{a}t - \frac{a-\alpha}{a^2}e^{-at}, \tag{2.49}$$

можно вместо (2.44) записать следующую систему алгебраических уравнений:

$$\left.\begin{aligned} p\overline{D}(0, p) &= d_0 - \mu\overline{D}(0, p) + \mu\overline{D}(1, p); \quad (2.50)\\ p\overline{D}(1, p) &= d_1 + \lambda\overline{D}(0, p) - \lambda\overline{D}(1, p). \quad (2.51) \end{aligned}\right\}$$

Из (2.50) видно, что $\overline{D}(0, p) = [d_0 + \mu\overline{D}(1, p)](p+\mu)^{-1}$. Подставим в (2.51) значение $\overline{D}(0, p)$ и получим

$$\overline{D}(1, p) = d_1 \frac{p + (\mu + \lambda d_0 d_1^{-1})}{p[p + (\lambda + \mu)]}. \tag{2.52}$$

Если же выразить $\overline{D}(1, p)$ через $\overline{D}(0, p)$ при помощи (2.51) и подставить $\overline{D}(1, p)$ в (2.50), то получим

$$\overline{D}(0, p) = d_0 \frac{p + (\lambda + \mu d_1 d_0^{-1})}{p[p + (\lambda + \mu)]}. \tag{2.53}$$

Из (2.52), (2.53) и (2.49) следует решение (2.45) системы дифференциальных уравнений (2.44).

Таким образом, аппарат марковских процессов с доходами позволяет вывести расчетные формулы для математического ожидания дохода, приносимого при эксплуатации ЭВМ. Вычислительные трудности при таком аналитическом расчете незначительны; численный расчет значений $D(i, t)$, $i \in E_0^1$, также не выдвигает серьезных трудностей. Однако здесь мы имели дело с наиболее простым объектом — вычислительной машиной, имеющей два состояния. Если при функционировании ЭВМ необходимо (с точки зрения надежности) различать много состояний или же если исследуется ВС

параллельного действия, то надеяться на простоту технико-экономического анализа не приходится. Трудности анализа ВС с помощью марковских процессов с доходами становятся настолько сложными, что сам этот подход отвергается инженерной практикой. Далее рассмотрим подход, который не сложнее только что рассмотренного и приводит к тем же расчетным формулам. Достоинство предлагаемого подхода заключается в возможности его трансформации для параллельных ВС, причем такой трансформации, которая не приводит к заметному увеличению расчетных трудностей.

Способ 2. Найдем оценку $D(i, t+\Delta t)$, $i \in \{0, 1\}$, математического ожидания дохода ЭВМ к моменту времени $(t+\Delta t)$ с учетом малых величин порядка не выше Δt. Эта оценка определяется как сумма ожидаемых доходов на промежутках времени $[0, t)$ и $[t, t+\Delta t)$ за вычетом расходов на восстановление ЭВМ и издержек, связанных с содержанием восстанавливающего устройства в течение времени Δt.

Если ЭВМ при $t \geqslant 0$ неработоспособна и на промежутке времени $[t, t+\Delta t)$ будет восстановлена, то из-за издержек на восстановление имеет место отрицательный доход

$$-P_0(i, t)P_{01}(\Delta t)c_3 + o(\Delta t) = -[1 - s(i, t)]u(\Delta t)c_3 + o(\Delta t).$$

В случае если за Δt восстановление отказавшей машины не осуществляется, доход равен нулю.

Если ЭВМ в момент времени $t \geqslant 0$ работоспособна и сохранит свое состояние неизменным в последующее время Δt, то ожидаемый доход за Δt составит величину

$$P_1(i, t)P_{11}(\Delta t)c_1\Delta t + o(\Delta t) = s(i, t)r(\Delta t)c_1\Delta t + o(\Delta t).$$

В случае перехода ЭВМ из работоспособного состояния в неработоспособное на промежутке времени $[t, t+\Delta t)$ ожидаемый доход равен нулю.

Независимо от состояния ЭВМ в момент времени $t \geqslant 0$ в условиях ее коммерческой эксплуатации ожидаемый доход на промежутке времени $[t, t+\Delta t)$ сокращается на величину $c_2'\Delta t$, равную себестоимости содержания ВУ (2.28).

Учет найденных оценок позволяет записать

$$D(i, t+\Delta t) = D(i, t) - [1 - s(i, t)]u(\Delta t)c_3 + s(i, t)r(\Delta t)c_1\Delta t - c_2'\Delta t + o(\Delta t). \quad (2.54)$$

Подставив в (2.54) оценки (2.41) для $r(\Delta t)$ и $u(\Delta t)$, получим

$$D(i, t+\Delta t) = D(i, t) - [1 - s(i, t)][\mu\Delta t + o(\Delta t)]c_3 + \\ + s(i, t)[1 - \lambda\Delta t + o(\Delta t)]c_1\Delta t - c_2'\Delta t + o(\Delta t). \quad (2.55)$$

От выражения (2.55) очевиден переход к дифференциальному уравнению для функции $D(i, t)$:

$$\frac{d}{dt} D(i, t) = s(i, t)(c_1 + \mu c_3) - (c_2' + \mu c_3). \qquad (2.56)$$

Учитывая вид функции $s(i, t)$ [см. (2.24), (2.25)], можно убедиться, что решением уравнения (2.56) при начальных условиях $D(i, 0) = 0$, $i \in E_0^1$, будет

$$D(i, t) = D_i + gt - D_i\delta(t), \qquad (2.57)$$

где прибыль g и функция $\delta(t)$ рассчитываются соответственно по формулам (2.47) и (2.37):

$$D_0 = -\frac{\mu(c_1 + \mu c_3)}{(\lambda + \mu)^2}; \quad D_1 = \frac{\lambda(c_1 + \mu c_3)}{(\lambda + \mu)^2}. \qquad (2.58)$$

Результат (2.57) произведенного расчета полностью совпадает с известным результатом (2.45), полученным ранее с помощью марковских процессов с доходами.

2.9.5. Технология экспресс-анализа функционирования ЭВМ

Основываясь на результатах, полученных в § 2.8 и 2.9, можно предложить следующую технологию оперативного анализа функционирования ЭВМ.

Заданные количественные характеристики ЭВМ:

λ^{-1} — среднее время безотказной работы машины, ч;

μ^{-1} — среднее время восстановления отказавшей ЭВМ, ч;

c_1 — стоимость часа полезного времени работы ЭВМ (или арендная плата за один час эксплуатации машины), руб./ч;

c_2' и c^2 — себестоимость и стоимость содержания восстанавливающего устройства (или бригады обслуживания) в течение часа, руб./ч;

c_3 — средняя стоимость технических средств (БИС, плат и т. п.), расходуемых при однократном восстановлении отказавшей ЭВМ, руб.;

$\{0, 1\}$ — множество состояний ЭВМ, $i = 0$ — ЭВМ неработоспособна; $i = 1$ — машина работоспособна; i — начальное состояние ЭВМ.

Рассчитываемые показатели эффективности ЭВМ.

1. *Коэффициент и функция готовности ЭВМ*: s и $s(i, t)$.

1.1. Стационарный режим; s — вероятность того, что ЭВМ находится в работоспособном состоянии при длительной эксплуатации (практически при эксплуатации более 10 ч).

Расчет проводить по формуле

$$s = \mu/(\lambda + \mu).$$

1.2. Переходный режим; $s(i, t)$ — вероятность того, что ЭВМ, начавшая функционировать в состоянии $i \in \{0, 1\}$, будет находиться в момент времени $t > 0$ в работоспособном состоянии.

Расчет выполнять по формуле

$$s(i, t) = s + \begin{cases} -\mu(\lambda + \mu)^{-1}\delta(t), & i = 0; \\ \lambda(\lambda + \mu)^{-1}\delta(t), & i = 1, \end{cases}$$

где $\delta(t) = \exp[-(\lambda + \mu)t]$.

2. *Средние бесполезные эксплуатационные расходы ЭВМ*: γ и $\Gamma(i, t)$.

2.1. Стационарный режим; γ — средние бесполезные расходы за 1 ч при длительной эксплуатации ЭВМ (практически более чем 10-часовой эксплуатации).

Расчет осуществлять по формуле

$$\gamma = (\lambda c_1 + \mu c_2)(\lambda + \mu)^{-1}.$$

2.2. Переходный режим; $\Gamma(i, t)$ — средние бесполезные эксплуатационные расходы к моменту времени $t \geqslant 0$ при условии, что в начальный момент времени ЭВМ находилась в состоянии $i \in \{0, 1\}$.

Расчет выполнять по формулам

$$\Gamma(i, t) = -\varepsilon_i + \gamma t + \varepsilon_i\delta(t);$$

$$\varepsilon_i = \frac{c_1 - c_2}{(\lambda + \mu)^2} \begin{cases} -\mu, & i = 0; \\ \lambda, & i = 1. \end{cases}$$

3. *Средний доход ЭВМ*: g и $D(i, t)$.

3.1. Стационарный режим; g — прибыль ЭВМ, или средний доход, приносимый за 1 ч при длительной (более 10-часовой) эксплуатации машины.

Расчет проводить по формуле

$$g = \mu(c_1 - \lambda c_3)(\lambda + \mu)^{-1} - c_2'.$$

3.2. Переходный режим; $D(i, t)$ — средний доход, который приносит ЭВМ за время $t \geqslant 0$, если она начинает функционировать в состоянии $i \in \{0, 1\}$.

Расчет проводить по формулам

$$D(i,t) = D_i + gt - D_i\delta(t);$$

$$D_i = (c_1 + \mu c_3)(\lambda + \mu)^{-2} \begin{cases} -\mu, & i = 0; \\ \lambda, & i = 1. \end{cases}$$

4. *Информация эксплуатационникам и пользователям ЭВМ.*

Расчет функций $s(i,t)$, $\Gamma(i,t)$, $D(i,t)$ выполняют только в случае необходимости более точного анализа функционирования ЭВМ, для оценки времени вхождения машины в стационарный режим работы. В условиях коммерческой эксплуатации режим работы ЭВМ стационарен, поэтому достаточно рассчитать только s, γ, g.

2.10. Предпосылки совершенствования архитектуры ЭВМ. Представление о вычислительных системах

Если исходить из глобального трактования архитектуры ЭВМ (см. § 2.3), то легко заметить, что первые три поколения ЭВМ полностью основываются на модели вычислителя (см. § 2.2), на принципах, положенных в ее основу. Архитектуры ЭВМ, принадлежащих второму и даже третьему поколениям, являются модификациями архитектуры Дж. фон Неймана.

Развитие средств обработки информации, направленное на достижение высокой производительности, надежности и живучести, натолкнулось в рамках модели вычислителя на серьезные препятствия.

2.10.1. Анализ возможностей совершенствования ЭВМ

Последовательная обработка информации на ЭВМ. Эволюционное совершенствование средств ВТ связано с постоянной борьбой за увеличение производительности, следовательно, и за наращивание возможностей памяти, и повышение надежности, и за улучшение технико-экономических показателей. Существуют следующие способы повышения производительности ЭВМ при обработке информации:

1) совершенствование и разработка алгоритмов решения задач;

2) создание эффективного ПО и оптимизация программ;

3) повышение быстродействия и улучшение физико-технических свойств элементов и внутримашинных информационных каналов;

4) улучшение алгоритмов выполнения машинных операций и соответствующая модификация структуры процессора;

5) модернизация алгоритма управления вычислительными процессами и канонической структуры ЭВМ.

Первый и второй способы основываются на фундаментальных достижениях математики; на тщательном анализе исходной задачи; на выборе методов и алгоритмов ее решения, наиболее адекватных структуре и количественным характеристикам как самой задачи, так и ЭВМ; на скрупулезном программировании, в максимальной степени учитывающем архитектурные возможности машины. Их целесообразно применять для уникальных машин, ориентированных на решение специальных классов задач. Они не дают ощутимого эффекта для ЭВМ широкого назначения.

Третий способ опирается на возможности использования новых физических явлений, материалов и совершенствования технологии производства БИС. В качестве базы для создания перспективных элементов можно указать на квантовую электронику (см. § 1.4).

Для современной микроэлектроники известны оценки, полученные на основании принципа неопределенности Гейзенберга и конечности скорости света ((299 792 ± 0,4) км/с, в вакууме): время обращения к памяти емкостью 2 бит равно 10^{-21} с для предельно допустимой плотности вещества. Это значение времени увеличивается с ростом емкости памяти. Для современной кремниевой технологии изготовления БИС достигнуто значение произведения мощности рассеивания на время переключения в элементе (полупроводниковом переходе), не превышающее 10^{-10} Дж. Теоретический предел рассеивания тепловой энергии равен 10^{-12} Дж. Это значение ограничивает плотность упаковки и быстродействие БИС. Проблема теплоотвода лимитирует время переключения элемента (при кремниевой технологии) значением 10^{-11} с. Время переключения современных элементов уже оценивается величинами от 10^{-10} до 10^{-11} с.

Следовательно, указанные выше обстоятельства не позволяют в условиях современной технологии БИС (да и перспективных некремниевых технологий, например, на основе углеродных транзисторов) существенно повысить производительность (до 10^{15} опер./с) последовательных ЭВМ за счет увеличения частотных возможностей и улучшения физико-технических свойств элементной базы. Кроме того, непрерывные успехи в микроминиатюризации позволяют (и будут позволять, видимо, до 2040 г.) увеличивать число транзисторов на чипе в 2 раза каждые 18 месяцев. Количество транзисторов на кристалле уже сейчас достигает 10^8 Однако эффективно распорядиться таким большим количеством транзисторов невозможно, если использовать последовательную обработку информации. Именно поэтому в современных микропроцессорных БИС уже вложены нефоннеймановские, или параллельные архитектуры средств обработки информации.

Четвертый способ повышения производительности ЭВМ связан с поиском форм представления чисел и алгоритмов, убыстряющих реализацию машинных операций. При этом следует учитывать, что основные логиче-

ские операции (типа сравнения) дальнейшему ускорению не поддаются. Алгоритмы, которые дают заметное ускорение, для разных арифметических операций существенно различаются между собой, что приводит к техническим сложностям в реализации процессора. Переход на эффективные алгоритмы реализации операций, конечно, может обеспечить в современных условиях рост быстродействия ЭВМ. Однако при этом следует заметить, что предельные варианты модификации алгоритмов выполнения арифметических операций достигаются при конвейерных вычислениях. Но если использовать последнее, то будет создана не ЭВМ, а вычислительная система с нефоннеймановской архитектурой.

Пятый способ повышения производительности средств обработки информации основывается на заметной модернизации алгоритма управления вычислительными процессами и, следовательно, канонической функциональной структуры ЭВМ. Наиболее яркой новацией в этом способе является доведение последовательно-параллельного алгоритма управления и структуры вычислительного средства до возможности выполнения конвейерных вычислений. При конвейеризации процесс обработки данных состоит из нескольких этапов, причем над различными частями данных допускается одновременная реализация этих этапов. Последнее достигается предельной трансформацией канонической последовательной структуры ЭВМ в конвейер, состоящий из специализированных вычислителей. Следовательно, конвейеризация вычислений основывается на значительном отходе от модели одиночного вычислителя и от архитектуры ЭВМ Дж. фон Неймана и закладывает фундамент для новой (параллельной) модели.

Фиксированность структуры ЭВМ. Отсутствие возможности автоматического изменения структуры не позволяет в полной мере адаптировать ЭВМ к области применения (подобрать адекватную структуру и режим обработки), учесть особенности и характеристики задач при их программировании и т. п. Жесткий алгоритм управления вычислительными процессами вынуждает создавать программы решения задач, в которых фиксированы последовательность выполнения операций и порядок использования данных. Такая фиксированность сохраняется при повторных решениях задач даже с другими массивами данных (независимо от их объемов и структур, связанных с физической сущностью задач). Жесткость структуры ЭВМ в ряде случаев приводит к значительным трудностям программирования задач и не позволяет использовать эффективные методы их решения. Говоря иначе, фиксированность структуры ЭВМ однозначно приводит к процедурному способу обработки информации (см. § 2.5), вариации в ходе реализации программ ограничены возможностями команд условных переходов.

Неоднородность ЭВМ. Существовавшее в 40-х годах XX в. представление о методах обработки информации и об организации вычислительных

процессов, острая потребность в средствах автоматизации вычислителей, уровни развития электроники и техники для вычислений, а также экономические ограничения однозначно определили структуру и состав приемлемого средства автоматической обработки данных — ЭВМ Дж. фон Неймана. Конструктивный принцип неоднородности в машине Дж. фон Неймана реализован на нескольких уровнях: структура ЭВМ в целом нерегулярна, а состав гетерогенный: каждое из пяти устройств (см. рис. 2.1) имеет свое функциональное назначение и свою логическую организацию, основывается на специфических принципах, обладает своими особенностями технической реализации. В пределах всей конфигурации ЭВМ однородность достигается лишь фрагментарно (например, оперативная память допускает однородную реализацию). Построение неоднородных ЭВМ находится в резком противоречии с тенденцией развития микро- и наноэлектроники.

Таким образом, путь эволюционного совершенствования средств ВТ на основе модели вычислителя (принципов последовательной обработки информации, фиксированности структуры и конструктивной неоднородности) не даст кардинального улучшения их технических характеристик.

Замечание. Уместно обратить внимание на следующий факт. Современная ЭВМ — это аппаратурно-программный комплекс (Computer = Hardware & Software). Необходимость решения на ЭВМ все более сложных задач требовала от математиков-вычислителей и программистов создания изощренных вычислительных методов и изящного (и вместе с тем трудоемкого) программирования. В качестве примеров сложных задач, решенных на ЭВМ (например, БЭСМ-6), могут служить задачи ядерной энергетики и космоса. Математики и программисты при решении сложных задач на ЭВМ совершали «интеллектуальные подвиги». С другой стороны, конструкторы также были вынуждены искать утонченные решения с целью повышения производительности ЭВМ. Научно-технический прогресс требовал от математиков, программистов и инженеров, использующих модель вычислителя и функциональную структуру ЭВМ Дж. фон Неймана, напряженной и все возрастающей умственной нагрузки. Из сказанного следует, что успех решения сложных задач определяется возможностями триады: мозг специалиста, аппаратура и ПО ЭВМ (Greyware & Hardware & Software).

Первый компонент триады (Greyware) «породил» новые архитектурные решения для средств обработки информации. Был найден кардинальный путь преодоления ограничений, присущих ЭВМ. Этот путь связан с диалектическим отрицанием принципов модели вычислителя.

Примечание. *Принцип неопределенности* предложен в 1927 г. В. Гейзенбергом (Heisenberg, p. 1901). Данный принцип (или соотношение неопределенностей) является фундаментальным положением квантовой теории. Он утверждает, что любая физическая система (микрочастица) не может находиться в состояниях, в которых координаты ее центра инерции и импульс одновременно принимают вполне определенные, точные значения. Говоря иначе, имеют место неопределенности значений координат (Δx, Δy, Δz) и неопределенности проекций p_x, p_y, p_z импульса на оси x, y, z такие,

что каждое из произведений $\Delta x p_x$, $\Delta y p_y$, $\Delta z p_z$ по порядку величины не меньше постоянной Планка $h \approx 10^{34}$ Дж · с (Plank, 1858–1947). Принцип неопределенности обусловлен корпускулярно-волновой природой микроскопических объектов.

Импульс **p**, или количество движения — мера механического движения; является векторной величиной, равной для материальной точки произведению ее массы m на скорость **u** и направленной так же, как вектор скорости: $\mathbf{p} = m\mathbf{u}$.

2.10.2. Архитектурные особенности параллельных вычислительных систем

На современном этапе развития микроэлектроники (элементной базы для ВТ) и в перспективе технико-экономически оправдано создание средств обработки информации, функционирование которых должно быть основано на имитации работы не одиночных вычислителей, а коллективов вычислителей. Такие средства обработки информации получили название *вычислительных систем* (ВС). Средства, использующие конвейерный способ обработки информации, являются не только пределом в эволюционном развитии ЭВМ Дж. фон Неймана, но и простейшим классом ВС.

С предельной ясностью можно заключить, что первыми двумя членами архитектурного ряда автоматических средств обработки информации являются ЭВМ (во всех своих модификациях трех поколений) и конвейерные вычислительные системы; виден и третий член — это параллельные ВС.

Параллельные ВС относят к четвертому, пятому и последующим поколениям средств обработки информации. Вообще говоря, параллельные ВС различаются по своей архитектуре и функциональной структуре, по способам обработки информации и т. п. Существуют классификации параллельных ВС, и в каждом классе выделяются свои поколения ВС; более того, говорят даже о поколениях ВС, выпускаемых той или иной фирмой. Ниже будут рассмотрены самые общие архитектурные возможности ВС, относящихся к средствам обработки информации четвертого и пятого поколений.

Для количественной характеристики поколений вычислительных средств будем использовать вектор $\{\Omega, V, \Theta, \Sigma\}$, где Ω — показатель производительности, или среднее число операций, выполняемых в секунду всеми процессорами ВС, опер./с; V — емкость оперативной памяти ВС, бит; Θ — среднее время безотказной работы ВС в целом, ч; Σ — «цена операции», рассчитываемая как отношение цены ВС к показателю производительности (долл./(опер. · $с^{-1}$)).

Четвертое поколение вычислительных средств (первое поколение ВС). Годы появления: 1964–1972; количественные характеристики: $\Omega \geqslant 10^8$ опер./с, $V \geqslant 10^9$ бит, $\Theta \geqslant 10^4$ ч, $\Sigma \leqslant 10^{-2}$ долл./(опер. · $с^{-1}$).

Появление четвертого поколения средств обработки информации (Fourth-generation Computers) связано с качественно новыми требованиями к реализации вычислительного процесса при переходе от решения одной задачи (набора задач) к решению сложной задачи. Задача-система представляет собой совокупность подзадач, связанных друг с другом. Она не допускает представления в виде набора простых задач и может быть решена лишь целиком и при условии применения средств обработки информации с нефоннеймановской структурой (построенных на основе модели коллектива вычислителей). Алгоритм управления вычислительными процессами в средствах четвертого поколения — это универсальный параллельно-последовательный алгоритм с автоматическим изменением своей структуры. Структура вычислительных средств может также автоматически изменяться (программироваться) в зависимости от структуры и параметров решаемой задачи.

Возможности ПО вычислительных средств четвертого поколения достаточно обширны: это совокупности операционных систем, системы (расширяемых) языков параллельного программирования, пакеты прикладных параллельных программ и сервисные программные комплексы. Характерной особенностью средств четвертого поколения стало то, что многие функции программного обеспечения ЭВМ третьего поколения получили аппаратурную реализацию. Элементную базу ВС составили БИС (микропроцессоры и кристаллы памяти).

Пятое поколение вычислительных средств (второе поколение ВС). Годы возникновения — 80-е годы XX столетия; количественные характеристики: $\Omega \geqslant 10^{11}$ опер./с, $V \geqslant 10^{12}$ бит, $\Theta \geqslant 10^{5}$ ч, $\Sigma \leqslant 10^{-3}$ долл./(опер. $\cdot$ с$^{-1}$).

Это поколение вычислительных средств (Fifth-generation Computers) связано с решением еще более сложных (суперсложных) системных задач, известных под общим названием «проблемы искусственного интеллекта». Для решения задач такой сложности требуются самоорганизующиеся вычислительные средства, архитектура и функциональная структура которых должны допускать автоматические изменения универсального алгоритма управления процессом вычислений в течение всего времени решения задачи.

В последующих главах будут рассмотрены архитектуры и функциональные структуры ВС.

3. АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Концептуальное представление о средствах обработки информации, получивших название ВС, базируется на модели коллектива вычислителей, на понятиях параллельного алгоритма, архитектуры и макроструктуры ВС.
Логика развития ВТ и дуализм понятия «вычислитель» (см. § 2.2) порождают понятие «коллектив вычислителей», допускающее двойное толкование: и как ансамбль людей, занятых расчетами, и как система аппаратурно-программных средств для обработки информации. Коллектив аппаратурно-программных вычислителей является ВС. Архитектура ВС основывается на структурной и функциональной имитации коллектива (ансамбля) людей-вычислителей. Степень адекватности такой имитации определяет потенциальные архитектурные возможности вычислительной системы.
Модель, приведенная в данной главе, может рассматриваться как модель функциональной организации ВС или просто как модель ВС.

3.1. Модель коллектива вычислителей

3.1.1. Принципы построения вычислительных систем

Каноническую основу конструкции ВС и ее функционирования составляет модель коллектива вычислителей [5], которая представляется парой:

$$S = \langle H, A \rangle, \tag{3.1}$$

где H и A — описание конструкции (или конструкция) и алгоритм работы коллектива вычислителей.

Конструкция коллектива вычислителей описывается в виде

$$H = \langle C, G \rangle, \tag{3.2}$$

где $C = \{c_i\}$ — множество вычислителей c_i, $i = \overline{0, N-1}$; N — мощность множества C; G — описание макроструктуры коллектива вычислителей, т. е. структуры сети связей между вычислителями $c_i \in C$ (или структура коллектива).

Конструкция коллектива вычислителей есть отражение следующих основополагающих архитектурных принципов:

1) *параллелизма (Parallelism, Concurrency)* при обработке информации (параллельного выполнения операций на множестве C вычислителей, взаимодействующих через связи структуры G);

2) *программируемости (Programmability, Adaptability) структуры* (настраиваемости структуры G сети связей между вычислителями, достигаемой программными средствами);

3) *однородности (Homogeneity) конструкции* H (однородности вычислителей $c_i \in C$ и структуры G).

Суть принципов становится ясной, если учесть, что они противоположны принципам, лежащим в основе конструкции вычислителя. Целесообразно подчеркнуть лишь то, что принцип программируемости структуры является столь же фундаментальным в области архитектуры средств обработки информации, сколь основательны предложения Дж. фон Неймана. (Напомним, что он предложил хранить программу работы ЭВМ в ее памяти и модифицировать программу с помощью самой же машины, см. § 2.1.) Требования принципа программируемости структуры сводятся к тому, чтобы в коллективе вычислителей была заложена возможность хранения описания его изначальной физической структуры, априорной автоматической (программной) настройки проблемно-ориентированных (виртуальных) конфигураций и их перенастройки в процессе функционирования с целью обеспечения адекватности структурам и параметрам решаемых задач и достижения эффективности при заданных условиях эксплуатации.

Уровень развития вычислительной математики и техники, а также технологии микроминиатюризации (микроэлектроники и наноэлектроники) уже сейчас позволяет в некоторых областях вместо принципа однородности (состава C и структуры G) использовать принцип квазиоднородности (или виртуальной однородности) конструкции H. Более того, можно ограничиться лишь требованием совместимости вычислителей в коллективе и использовать неоднородные структуры. Однако вместе с этим следует отметить, что принцип однородности при создании высокопроизводительных ВС имеет не меньшую значимость, чем принцип совместимости ЭВМ третьего поколения.

3.1.2. Структура вычислительных систем

Структура (Structure, Topology) коллектива вычислителей представляется графом G, вершинам (узлам) которого сопоставлены вычислители $c_i \in C$, а ребрам — линии связи между ними. Проблема выбора (синтеза) структур ВС не является тривиальной. В самом деле, универсальное решение — структура в виде полного графа (Complete graph), однако такая

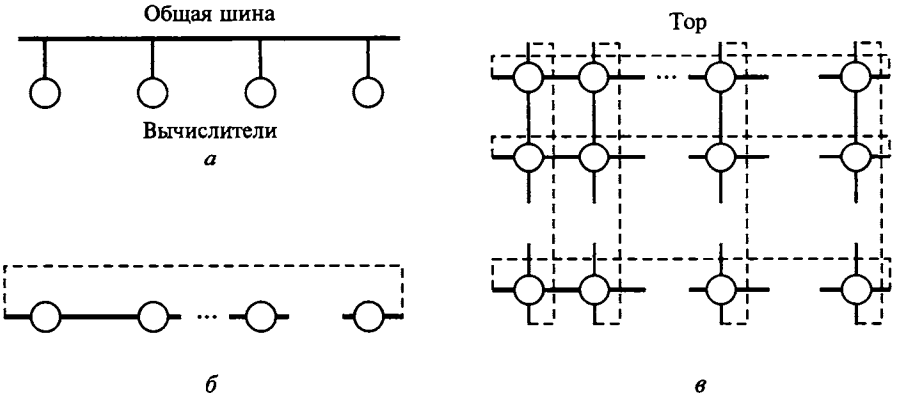


Рис. 3.1. Фрагменты простейших структур ВС:
а — нульмерная; *б* — одномерная; *в* — двумерная

структура практически реализуема при небольшом числе вычислителей. Для достижения производительности ВС в диапазоне $10^{10}...10^{15}$ опер./с при существующих интегральных технологиях необходимо около $10–10^5$ вычислителей. Следовательно, структуры таких систем не могут быть организованы по типу полного графа хотя бы из-за ограничений на число выводов с корпусов интегральных схем. Рассмотрим структуры сети связей между вычислителями, которые используются при формировании ВС.

Простейшие структуры ВС. Различают нульмерные, одномерные и двумерные простейшие структуры ВС (рис. 3.1). В первом случае структура сети межвычислительных связей «вырождена», взаимодействие между вычислителями ВС осуществляется через общую шину (Common bus, Uni bus). В случае одномерных структур («линейки» — Linear graph или «кольца» — Ring) обеспечивается связь каждого вычислителя с двумя другими (соседними) вычислителями (рис. 3.1, *а*, *б*). В нульмерных структурах имеется общий ресурс — шина, в одномерных же структурах этот ресурс трансформируется в распределенный, т. е. в локальные связи между вычислителями. Следовательно, архитектурные возможности (в частности, надежность) последних структур существенно выше, чем у нульмерных.

Увеличение размерности структуры повышает структурную надежность ВС. В самом деле, двумерные структуры предоставляют каждому вычислителю непосредственную связь с четырьмя соседними. В качестве примеров двумерных структур (рис. 3.1, *в*) может служить «решетка» (точнее, 2D-решетка — Two-dimensional grid) и тор (2D-тор — Two-dimensional torus). Следовательно, в системах с двумерной структурой при отказах некоторых вычислителей и (или) связей между ними сохраняется возможность организации связных подмножеств исправных вычислителей.

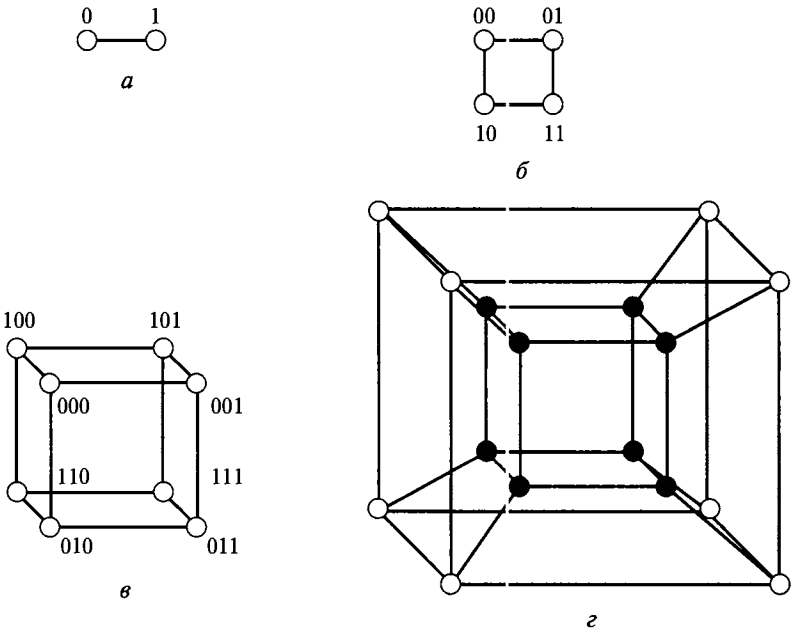


Рис. 3.2. Гиперкубические структуры ВС:

а — 1D-куб ($N = 2$, $n = 1$); *б* — 2D-куб ($N = 4$, $n = 2$); *в* — 3D-куб ($N = 8$, $n = 3$); *г* — 4D-куб ($N = 16$, $n = 4$)

В n-мерных структурах каждый вычислитель связан с $2n$ соседними вычислителями. Существуют технико-экономические и технологические ограничения в наращивании размерности структуры ВС.

Гиперкубические структуры ВС. Гиперкубы, или структуры в виде булевых n-мерных кубов, нашли широкое применение при построении современных высокопроизводительных ВС с массовым параллелизмом. Гиперкуб (Hypercube) по определению — это однородный граф, для которого справедливо

$$n = \log_2 N,$$

где N — количество вершин; n — число ребер, выходящих из каждой вершины; n — называют также *размерностью* гиперкуба. Каждый вычислитель в гиперкубической ВС имеет связь ровно с n другими вычислителями. На рис. 3.2 представлены гиперкубические структуры ВС. Гиперкуб размерности n называют также nD-кубом (D означает размерность (Dimension)).

Если вершины гиперкуба пронумеровать от 0 до $(N-1)$ в двоичной системе счисления так, что каждый разряд соответствует одному из n направлений, то получим булев n-мерный куб (см. рис. 3.2).

Вид структуры ВС с массовым параллелизмом существенно влияет на производительность системы в целом. В самом деле, даже при одном и том же количестве вершин существует разнообразие структур. Для создателей ВС интерес представляют структуры, обеспечивающие наименьшие временные задержки при обменах информацией между вычислителями. Следовательно, нужны структуры с минимальным диаметром. Под диаметром структуры понимается максимальное расстояние (число ребер) между двумя вершинами, определяемое на множестве кратчайших путей. Возможности распространенных структур ВС отражены в табл. 3.1.

Таблица 3.1

Характеристика	Тип структуры ВС					
	Полный граф	Линейка	Кольцо	2D-решетка	2D-тор	Гиперкуб
Диаметр	1	$N-1$	[N/2]*	$2(\sqrt{N}-1)$	$2\left[\sqrt{N}/2\right]$	$\log_2 N$
Количество ребер	$N(N-1)/2$	$N-1$	N	$2(N-\sqrt{N})$	$2N$	$(N\log_2 N)/2$

* $[N]$ — целая часть числа N.

Из табл. 3.1 следует, что структуры с меньшим диаметром имеют большее количество ребер; удвоение числа вершин в гиперкубе увеличивает его диаметр только на единицу.

Примечания: 1. Тор (от лат. Torus — выпуклость) — геометрическое тело, образуемое вращением круга вокруг непересекающей его и лежащей в одной с ним плоскости прямой. Приблизительно форму тора имеет баранка и спасательный круг. 2. В теории структур ВС тор — это решетка, в которой имеют место отождествления граничных связей в каждой «строке» и в каждом «столбце» (см. рис. 3.1). Легко представить размещение такой структуры–графа на поверхности геометрического тора. 3. Гиперкуб ($N = 16$, $n = 4$), представленный на рис. 3.2, *г*, является двумерным тором.

3.1.3. Алгоритм функционирования вычислительных систем

Алгоритм A работы коллектива S обеспечивает согласованную работу всех вычислителей $c_i \in C$ и сети связей между ними (структуры G) в процессе решения общей задачи. Данный алгоритм может быть представлен в виде суперпозиции

$$A(P(D)), \tag{3.3}$$

где D — исходный массив данных, подлежащих обработке в процессе решения задачи,

$$D = \bigcup_{i=0}^{N-1} D_i, \tag{3.4}$$

D_i — индивидуальный массив данных для вычислителя $c_i \in C$, причем в общем случае

$$\bigcap_{i=0}^{N-1} D_i \neq \varnothing; \tag{3.5}$$

P — *параллельная программа* решения общей задачи,

$$P = \bigcup_{i=0}^{N-1} P_i, \quad \bigcap_{i=0}^{N-1} P_i = \varnothing, \tag{3.6}$$

P_i — ветвь i программы P.

Следует отметить, что формула (3.5) представляет собой *условие информационной избыточности*, оно позволяет организовать отказоустойчивые параллельные вычисления. В самом деле, за счет локальной информационной избыточности в каждом вычислителе имеется потенциальная возможность к восстановлению всей исходной информации (или, по крайней мере, необработанных данных) при отказе отдельных вычислителей.

Ясно, что решение общей задачи осуществляется по некоторому параллельному алгоритму, а именно он является основой параллельной программы (3.6). Распределенные по вычислителям данные (3.4) и параллельная программа инициируют работу алгоритма A (3.3) и определяют функционирование конструкции H коллектива вычислителей до конца решения задачи.

Эквивалентным представлением алгоритма (3.3) работы коллектива вычислителей является композиция

$$(A_0 * A_0') * \ldots * (A_i * A_i') * \ldots * (A_{N-1} * A_{N-1}').$$

Здесь $(A_i * A_i')$ осуществляет функционирование вычислителя c_i среди других вычислителей множества C; A_i и A_i' — соответственно алгоритм автономной работы c_i и алгоритм реализации взаимодействий (в частности, обменов данными) с вычислителями $c_j \in C \setminus c_i$. Последний алгоритм является суперпозицией

$$A_i'(P_i'(G)),$$

где G — описание структуры коллектива C вычислителей; P_i' — программа для установления связей и выполнения взаимодействий между вычислите-

лем c_i и другими вычислителями подмножества $C \setminus c_i$. Программа P_i' является, как правило, частью ветви P_i параллельной программы (3.6).

Многообразие архитектурных реализаций модели коллектива вычислителей (многообразие типов ВС) является следствием разницы в способах воплощения совокупности алгоритмов $\{A_i'\}$, $i = \overline{0, N-1}$, задания $\{P_i'\}$ и выбора структуры G, а также разнообразия в правилах композиции алгоритмов.

Аппаратурные средства, с помощью которых реализуется совокупность алгоритмов $\{A_0', A_1', \ldots, A_i', \ldots, A_{N-1}'\}$ и которые вместе с сетью связей составляют среду для осуществления взаимодействий между вычислителями коллектива, называются *коммутатором*. От способа реализации коммутатора во многом зависит эффективность ВС параллельного действия. Коммутатор настолько же необходим при создании ВС, насколько нужны процессор и оперативная память для построения ЭВМ.

При полном воплощении принципа однородности имеют место отношения эквивалентности: $A_i \equiv A_j$, $A_i' \equiv A_j'$, $i \neq j$, $i; j = \overline{0, N-1}$, которые, в частности, обеспечивают высокую технологичность в проектировании и в производстве технических реализаций вычислителей, приводят к распределенному коммутатору, состоящему из N локальных идентичных коммутаторов (находящихся во взаимно однозначном соответствии с вычислителями), не вызывают никаких сложностей в формировании сети связей между вычислителями.

Принцип однородности приводит к простоте при разработке параллельного программного обеспечения. Он обеспечивает высокую технико-экономическую эффективность коллектива вычислителей как единого аппаратурно-программного комплекса.

3.1.4. Модель вычислительной системы

Формулы (3.1)–(3.3) позволяют записать для модели коллектива вычислителей или модели ВС

$$S = \langle C, G, A(P(D)) \rangle, \tag{3.7}$$

где C — множество вычислителей; G — структура сети связей между вычислителями; A — алгоритм работы множества C как коллектива вычислителей (взаимосвязанных через сеть G) при реализации параллельной программы P обработки данных D.

Модель ВС (3.7) является формализованным описанием коллектива вычислителей, ориентированного («запрограммированного») на решение

сложной задачи и реализующего процесс параллельных вычислений. Здесь уместно отметить опыт по организации параллельных вычислений, которые были осуществлены в 1950-х годах математиком А.А. Самарским (1919–2008; академик АН СССР с 1976 г.). Требовалось выполнить сложные расчеты, связанные с созданием ядерного оружия, а в распоряжении были только арифмометры. Для расчетов был привлечен большой коллектив сотрудниц-вычислителей, которым были распределены исходные данные. Каждая сотрудница реализовывала свою ветвь вычислений (рассчитывала свои узлы сетки). Обмен данными между сотрудницами (параллельными ветвями вычислений) осуществлялся путем передачи листов бумаги с полученными числами. Такой параллельный процесс вычислений позволил решить задачу в срок!

Модель коллектива вычислителей (3.7) допускает создание средств обработки информации разнообразных конфигураций. Представление коллектива вычислителей в качестве макровычислителя делает возможным формирование сверхколлективов или систем коллективов, или *макровычислительных систем*. Модель коллектива вычислителей (3.7) применима и на микроуровне; с ее помощью можно строить вычислитель как коллектив, составленный из микровычислителей.

Средства, основанные на модели коллектива вычислителей, называются вычислительными системами. Они заняли прочные позиции в современной индустрии обработки информации (точнее, в индустрии параллельных вычислений). Наиболее полно принципы модели коллектива вычислителей воплощены в *ВС с программируемой структурой* (см. гл. 7 или [5, 6]). Концепция таких ВС была предложена в Сибирском отделении АН СССР. Разработка теоретических основ и принципов технической реализации, а также создание первых ВС с программируемой структурой были осуществлены к началу 70-х годов XX в.

3.2. Техническая реализация модели коллектива вычислителей. Архитектурные свойства вычислительных систем

3.2.1. Принципы технической реализации модели коллектива вычислителей

Каноническое описание модели коллектива вычислителей (см. § 3.1), основополагающие принципы ее конструкции, а также состояние микроэлектроники и уровень развития (параллельной) вычислительной математики определяют принципы технической реализации модели или принципы построения ВС. Выделим модульность и близкодействие как главные принципы технической реализации модели коллектива вычислителей.

Модульность (Modularity) — принцип, предопределяющий формирование ВС из унифицированных элементов (называемых модулями), которые функционально и конструктивно закончены, имеют средства сопряжения с другими элементами, разнообразие которых составляет полный набор. Функциональные и конструктивные возможности модулей, разнообразие их типов определяются исходя из требований, предъявляемых к ВС, и, безусловно, из возможностей микроэлектронной базы.

Модульность вычислительной системы обеспечивает:

1) возможность использования любого модуля заданного типа для выполнения любого соответствующего ему задания пользователя;

2) простоту замены одного модуля на другой однотипный;

3) масштабируемость, т. е. возможность увеличения или уменьшения количества модулей без коренной реконфигурации связей между остальными модулями;

4) открытость системы для модернизации, исключающую ее моральное старение.

Следует заметить, что принцип модульности распространим и на средства программного обеспечения ВС.

При конструировании ВС с массовым параллелизмом достаточно ограничиться единственным модулем-вычислителем, который бы обладал вычислительной и соединительной полнотой. Следовательно, модуль должен иметь средства автономного управления, располагать арифметико-логическим устройством и памятью и содержать локальный коммутатор — схему для связи с другими модулями. На практике принято такой модуль-вычислитель называть либо *элементарным процессором* (ЭП), либо *элементарной машиной* (ЭМ). При этом считается, что ЭП — это композиция из процессора и локального коммутатора. Разрядность таких ЭП в различных ВС колеблется от 1 до 64. Под элементарной машиной понимается архитектурно более развитая композиция из ЭВМ и локального коммутатора.

Близкодействие (Short-range interaction) — принцип построения ВС, обусловливающий такую организацию информационных взаимодействий между модулями-вычислителями, при которой каждый из них может непосредственно (без «посредников») обмениваться информацией с весьма ограниченной частью модулей-вычислителей. Следовательно, структура ВС позволяет осуществлять информационные взаимодействия между удаленными вершинами-вычислителями лишь с помощью промежуточных вершин-вычислителей, передающих информацию от точки к точке (point-to-point). Удаленными считаются те вершины в структуре ВС, расстояние между которыми более 1 (число ребер между которыми более 1).

Принцип близкодействия допускает реализацию механизма управления ВС (организации функционирования коллектива вычислителей как еди-

ного целого), не зависящий от количества составляющих ее вычислителей. Данный принцип, в частности, выражается в том, что поведение каждого вычислителя $c_i \in C$ зависит от поведения только ограниченного подмножества $C^* \subset C$ других вычислителей системы.

Во взаимосвязи с принципом близкодействия говорят также о *локальности (locality) связей и взаимодействий между вычислителями*. Последнее означает, что состояние $E_i(t+1)$ вычислителя c_i, $i \in \{0, 2, \ldots, N-1\}$, на очередном временно́м шаге $(t+1)$ зависит от состояний (на предшествующем шаге t) непосредственно с ним связанных вычислителей $c_j \in C^*$ т. е. состояние является функцией

$$E_i(t+1) = f(E_i(t), E_{i_1}(t), E_{i_2}(t), \ldots, E_{i_M}(t)),$$

где i_l, $l = \overline{1, M}$ — номера вычислителей, составляющих C^*, $M < N$. При этом вычислители подмножества

$$C^* = \{c_{i_1}, c_{i_2}, \ldots, c_{i_M}\}$$

называются соседними по отношению к $c_i \in C$. Для достижения однородности структуры сети связей необходимо, чтобы каждый вычислитель был соединен с M другими вычислителями.

Вычислительные системы, основанные на принципах модульности и близкодействия, удовлетворяют также требованиям асинхронности, децентрализованности и распределенности.

Асинхронность функционирования (Asynchronous functioning) ВС обеспечивается, если порядок срабатывания ее модулей определяется не с помощью вырабатываемых тем или иным образом отметок времени, а достижением заданных значений определенных (как правило, логических) функций. Использование асинхронных схем позволяет достичь в системе алгоритмически предельного быстродействия: модули ВС срабатывают немедленно после выполнения соответствующего условия. Применение асинхронных схем обмена информацией между вычислителями позволяет не учитывать разброс в их тактовых частотах и колебания времени задержки сигналов в линиях связи.

Децентрализованность управления (Decentralized control) ВС достигается, если в системе нет выделенного модуля, который функционирует как единый для всей системы центр управления. Децентрализованное управление системой основано на совместной работе всех исправных модулей системы, направленной на принятие решений, доставляющих оптимум выбранной целевой функции. Выполняя свою часть работы по выработке согласо-

ванного решения об управлении системой, каждый модуль пользуется только локальной информацией о системе.

Децентрализованное управление системой (в отличие от централизованного) позволяет:

1) достичь живучести ВС, т. е. ее способности продолжать работу при отказах модулей (в том числе и тех, которые предназначены для принятия решений);

2) избежать очередей при обслуживании заявок на управление.

Распределенность ресурсов (State of distribution) ВС позволяет создавать такую систему, в которой нет единого ресурса, используемого другими в режиме разделения времени. Под ресурсами ВС понимаются все объекты, которые запрашиваются, используются и освобождаются в ходе выполнения вычислений. В качестве ресурсов ВС выступают процессоры или даже модули, входящие в их состав, модули оперативной памяти, внешние устройства, линии межмодульных связей, шины, файлы данных, компоненты ПО. Вместе с этим каждый ресурс распределенной ВС рассматривается как общий, доступный любому потребителю.

3.2.2. Архитектурные свойства ВС

Основополагающие принципы (параллелизма, программируемости, однородности) и принципы модульности и близкодействия (см. разд. 3.1.1 и 3.2.1) позволяют достичь полноты архитектурных свойств в ВС. Отметим важнейшие свойства архитектуры ВС. При этом заметим, что не все свойства и не в полной мере могут проявляться в той или иной реализации ВС.

Масштабируемость (Scalability) ВС. Под масштабируемостью ВС понимается их способность к наращиванию и сокращению ресурсов, возможность варьирования производительности. Сложность (трудоемкость) задач, решаемых на вычислительных средствах, постоянно растет. Для сохранения в течение длительного времени способности ВС адекватно решать сложные задачи необходимо, чтобы она обладала архитектурным свойством масштабируемости. Это означает, в частности, что производительность, достигнутую ВС на заданном количестве вычислителей, можно увеличить, добавив еще один или несколько вычислителей. Выполнение этого свойства ВС гарантируется принципами модульности, локальности, децентрализованности и распределенности.

Свойство наращиваемости производительности предоставляет потенциальную возможность решать задачи любой априори заданной сложности. Однако для практической реализации этой возможности требуется, чтобы алгоритм решения сложной задачи удовлетворял условию локальности, а межмодульные пересылки информации слабо влияли на время решения за-

дачи. Это можно достичь за счет крупноблочного распараллеливания сложных задач (см. § 3.3) и (или) аппаратурных средств, позволяющих совместить межмодульные обмены информацией с вычислениями.

Универсальность (Genericity, Generality, Versatility) ВС. Вычислительные системы алгоритмически и структурно универсальны. Принято считать, что ЭВМ (основанные на модели вычислителя) являются алгоритмически универсальными, если они обладают способностью (без изменения своих структур) реализовать алгоритм решения любой задачи. С другой стороны, ВС — это коллектив вычислителей, каждый из которых обладает алгоритмической универсальностью, следовательно, и система универсальна (в общепринятом смысле).

В вычислительных системах могут быть реализованы не только любые алгоритмы, доступные ЭВМ, но и параллельные алгоритмы решения сложных задач. Последнее следует из определений модели коллектива вычислителей (3.7) и, в частности, алгоритма функционирования ВС (3.3).

Структурная универсальность ВС является следствием воплощения архитектурных принципов коллектива вычислителей (см. разд. 3.1.1), в частности принципа программируемости структуры. Суть этого принципа — возможность автоматически (программно) порождать специализированные (проблемно-ориентированные) виртуальные конфигурации, которые адекватны структурам и параметрам решаемых задач.

Таким образом, ВС сочетают в себе достоинства цифровой техники, где процесс вычислений в основном задается алгоритмически (точнее, программно), и аналоговой техники, где процесс вычислений предопределяется структурными схемами.

Структурная универсальность позволяет говорить и о специализированности ВС: для каждой задачи допустима автоматическая настройка такой конфигурации из ресурсов ВС, которая наиболее адекватна алгоритму решения задачи. Таким образом, *ВС — это средство, в котором диалектически сочетаются противоположные свойства универсальности и специализированности.*

Алгоритмическая и структурная универсальность ВС проявляются также в возможности организации «виртуальных» конфигураций с произвольной архитектурой (на уровне потоков команд и данных) и реализации в ней известных режимов обработки информации.

Производительность (Performance, Throughput, Processing power) ВС. В отличие от ЭВМ, построенных на основе модели вычислителя, ВС не имеют принципиальных ограничений в повышении производительности. Увеличение производительности в них достигается за счет не только повышения физического быстродействия микроэлектронных элементов, а главным образом увеличения числа вычислителей. Следует подчеркнуть, что благодаря

свойству однородности наращиваемость ВС осуществляется простым подключением дополнительных вычислений без конструктивных изменений первоначального состава системы. При этом достигается простота настройки ПО на заданное число вычислителей в системе. На основании последнего обеспечивается *совместимость* ВС различной производительности.

Полнота воплощения принципа параллелизма при выполнении операций позволяет достичь априори заданной производительности ВС как в монопрограммном режиме (при решении одной сложной задачи, т. е. задачи с большим числом операций), так и в мультипрограммных режимах (при обработке наборов и обслуживании потоков задач произвольной сложности). Задачи представляются параллельными программами, число ветвей в каждой из которых является, в частности, функцией от сложности задачи. Значения производительности, емкости памяти, скорости ввода-вывода информации для системы определяются числом вычислителей и их составом.

Реконфигурируемость (Reconfigurability) ВС. Структурная и функциональная гибкости ВС обусловлены широкими возможностями систем по статической и динамической реконфигурации. *Статическая реконфигурация ВС* обеспечивается: варьированием числа вычислителей, их структуры и состава; выбором для вычислителей числа полюсов для связи с другими вычислителями; возможностью построения структур в виде графов, относящихся к различным классам; допустимостью применения в качестве связей каналов различных типов, различной физической природы и различной протяженности и т. п. Благодаря приспособленности ВС к статической реконфигурации достигается адаптация системы под область применения на этапе ее формирования.

Динамическая реконфигурация ВС поддерживается возможностью образования в системах таких (виртуальных) подсистем, структуры и функциональные организации которых адекватны входной мультипрограммной ситуации и структурам решаемых задач. Следовательно, способность ВС к динамической реконфигурации приводит к ее высокой *универсальности*, при которой достигается заданный уровень производительности при решении широкого класса задач; реализуются известные в вычислительной технике режимы функционирования (коллективное пользование, пакетная обработка и др.), способы управления вычислительным процессом (централизованный, децентрализованный и др.), структурные схемы (изолированные вычислительные машины, системы из нескольких процессоров и одной ЭВМ, системы из одной ЭВМ и нескольких устройств памяти и т. п.) и способы обработки информации (конвейерный, матричный, распределенный и др.).

Способность ВС к динамической реконфигурации является следствием полноты воплощения принципов коллектива вычислителей и прежде всего принципа программируемости структуры. Эта способность ВС позволяет

ей в процессе функционирования проводить автоматическую перенастройку своей структуры для реализации обменов информацией между вычислителями, осуществлять «подстройку» состояний функциональных устройств и узлов в вычислителях с целью достижения адекватности между ВС и совокупностью совместно протекающих в ней процессов.

Надежность и живучесть (Reliability and Robustness) ВС. Данные два понятия семантически близки, оба призваны характеризовать архитектурные способности ВС по выполнению возлагаемых на них функций. Однако каждое из них отражает специфические особенности ВС по использованию исправных ресурсов при переработке информации.

Под надежностью ВС понимается ее способность к автоматической (программной) настройке и организации функционирования таких структурных схем, которые при отказах и восстановлении вычислителей обеспечивают заданный уровень производительности или, говоря иначе, возможность использовать фиксированное число исправных вычислителей (при реализации параллельных программ решения сложных задач). Это понятие характеризует возможности ВС по переработке информации при наличии фиксированной структурной избыточности (представленной частью вычислителей) и при использовании параллельных программ с заданным числом ветвей.

При изучении надежности ВС под *отказом* понимают событие, при котором система теряет способность выполнять функции, связанные с реализацией параллельной программы с заданным числом ветвей. Если ВС находится в состоянии отказа, то число неисправных вычислителей превосходит число вычислителей, составляющих структурную избыточность. Понятие надежности ВС вкладывается в общепринятое понятие надежности систем. При этом структурные схемы, порождаемые в пределах ВС для надежной реализации параллельных программ с фиксированным числом ветвей, выступают как виртуальные системы (достаточно близкие к системам с резервом).

Под живучестью ВС понимают свойство программной настройки и организации функционирования таких структурных схем, которые в условиях отказов и восстановления вычислителей гарантируют при выполнении параллельной программы производительность в заданных пределах или возможность использования всех исправных вычислителей. Понятие «живучесть» ВС характеризует их способности по организации отказоустойчивых вычислений или, говоря иначе, по реализации параллельных программ, допускающих варьирование числа ветвей в известных пределах.

При рассмотрении живучести ВС выделяют *полный и частичный отказы*. Под полным отказом ВС понимают событие, при котором система теряет способность выполнять параллельную программу с переменным чис-

лом ветвей. Частичным отказом называют событие, при котором имеют место отказы вычислителей, но сохраняется возможность реализации на ВС параллельной программы с переменным числом ветвей. При полном отказе производительность системы становится равной нулю. Частичный отказ приводит лишь к некоторому снижению производительности, т. е. к увеличению времени реализации параллельной программы с переменным числом ветвей. Понятия полного и частичного восстановления ВС очевидны.

В живучих ВС допустимо использование аппаратурной избыточности на уровне отдельных функциональных устройств и узлов вычислителей, однако эта избыточность играет лишь вспомогательную роль.

Следует подчеркнуть, что *в живучей ВС в любой момент функционирования используется суммарная производительность всех исправных вычислителей.* Из этого следует, что программы решения задач должны обладать свойством адаптируемости (под число исправных вычислителей) и иметь информационную избыточность.

Ясно, что описанные выше принципы технической реализации ВС как коллективов вычислителей (см. разд. 3.2.1) являются необходимыми условиями достижения ими свойства живучести.

Самоконтроль и самодиагностика (Self-testing and Self-diagnostics) ВС. Организация надежного и живучего функционирования ВС связана с контролем правильности их работы и с локализацией неисправностей в них. В системах-коллективах вычислителей может быть применен нетрадиционный подход к контролю и диагностике:

1) в качестве контрольно-диагностического ядра ВС могут быть использованы любые исправные вычислители и в пределе ядро любого произвольно выбранного вычислителя;

2) выбор ядра системы и определение ее исправности могут быть выполнены автоматически (с помощью средств ВС).

Предлагаемый подход позволяет говорить о самоконтроле и самодиагностике ВС. Заключение об исправности или неисправности отдельных вычислителей системы принимается коллективно всеми вычислителями на основе сопоставления их индивидуальных заключений об исправности соседних с ними вычислителей.

Сказанное выше относительно надежности, живучести, самоконтроля и самодиагностики ВС в равной степени относится и к отдельным частям систем, к их подсистемам. Следовательно, надежность и живучесть ВС могут быть достигнуты и в случае мультипрограммной работы.

Технико-экономическая эффективность (Technical-economical Efficiency) ВС. Конструктивная однородность позволяет резко сократить сроки разработки и изготовления систем, обеспечивает высокую технологичность производства, упрощает и статическую, и динамическую реконфигурации

ВС, облегчает их техническую эксплуатацию. Она существенно упрощает процесс организации взаимодействий между вычислителями ВС и облегчает создание ПО. Полнота воплощения трех основных принципов модели коллектива вычислителей позволяет заметно ослабить зависимость между повышением производительности ВС и увеличением трудоемкости их проектирования и изготовления, а также созданием системного ПО. Тем самым открывается возможность построения высокопроизводительных экономически приемлемых ВС при существующей физико-технологической базе. Более того, возможность неограниченно наращивать производительность позволяет применить для построения ВС микроэлектронные элементы с быстродействием, далеким от предельного, и следовательно, обладающие более высокой надежностью и меньшим энергопотреблением.

3.3. Параллельные алгоритмы

Рассмотрим основные понятия и приемы параллельного программирования и изучим методику крупноблочного распараллеливания сложных задач.

3.3.1. Элементарные понятия параллельного программирования

Понятие параллельного алгоритма (Parallel Algorithm) относится к фундаментальным в теории ВС. Это понятие прежде всего ассоциируется с ВС с массовым параллелизмом. *Параллельный алгоритм — описание процесса обработки информации, ориентированное на реализацию в коллективе вычислителей.* Такой алгоритм в отличие от последовательного предусматривает одновременное выполнение множества операций в пределах одного шага вычислений и как последовательный алгоритм сохраняет зависимость последующих этапов от результатов предыдущих.

Параллельный алгоритм решения задачи составляет основу параллельной программы *P*, которая, в свою очередь, влияет на алгоритм функционирования коллектива вычислителей (см. разд. 3.1.3). *Запись параллельного алгоритма на языке программирования,* доступном коллективу вычислителей, *называют параллельной программой (Parallel Program)*, а сам язык — параллельным (Parallel Language). Параллельные алгоритмы и программы следует разрабатывать для задач, которые недоступны для решения на средствах, основанных на модели вычислителя (сложные или трудоемкие задачи).

Методы и алгоритмы обработки информации, решения задач, как правило, являются последовательными. Процесс «приспособления» методов к реализации на коллективе вычислителей или процесс «расщепления» после-

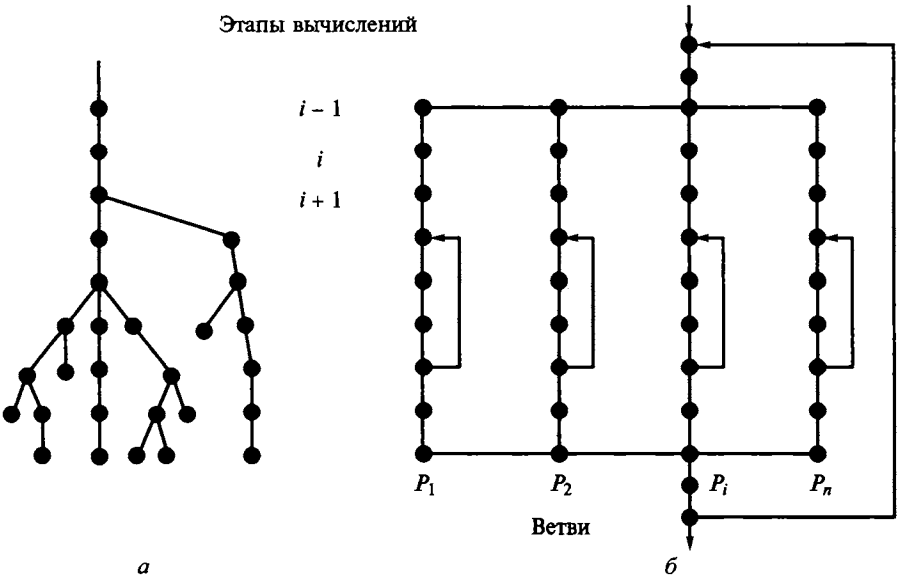


Рис. 3.3. Фрагменты схемы параллельных алгоритмов:
а — локальное распараллеливание; *б* — глобальное распараллеливание; ● — операторы

довательных алгоритмов решения сложных задач называется *распараллеливанием (Paralleling or Multisequencing).*

Теоретическая и практическая деятельность по созданию параллельных алгоритмов и программ обработки информации называется *параллельным программированием* (*Parallel or Concurrent Programming*).

Качество параллельного алгоритма (или его эффективность) определяется методикой распараллеливания сложных задач. При распараллеливании задач используют два подхода (рис. 3.3) — *локальное* и *глобальное (крупноблочное) распараллеливание.* Первый подход ориентирован на разбиение алгоритма решения сложной задачи на предельно простые блоки (операции или операторы) и требует выделения для каждого этапа вычислений максимально возможного количества одновременно выполняемых блоков. Он не приводит к параллельным алгоритмам, эффективно реализуемым коллективом вычислителей. В самом деле, процесс такого распараллеливания весьма трудоемок, а получаемые параллельные алгоритмы характеризуются не только структурной неоднородностью, но и существенно разными объемами операций на различных этапах вычислений. Последнее является серьезным препятствием на пути (автоматизации) распараллеливания и обеспечения эффективной эксплуатации ресурсов коллектива вычислителей. Локальное распараллеливание позволяет оценить предельные возможности коллектива вычислителей при решении сложных задач, получить предельные оценки по распараллеливанию сложных задач.

Второй подход ориентирован на разбиение сложной задачи на крупные блоки-подзадачи, между которыми существует слабая связность. Тогда в алгоритмах, построенных на основе крупноблочного распараллеливания, операции обмена между подзадачами будут составлять незначительную часть по сравнению с общим числом операций в каждой подзадаче. Такие подзадачи называют *ветвями параллельного алгоритма*, а соответствующие им программы — *ветвями параллельной программы*.

Пусть для выполнения параллельной программы выделяется (под)множество $C = \{c_i\}$, $i \in \{1, 2, \ldots, n\}$, вычислителей c_i, тогда

$$P = \bigcup_{i=1}^{n} P_i$$

и P_i — ветвь, реализуемая вычислителем c_i. Принцип однородности коллектива вычислителей позволяет создавать параллельные алгоритмы и программы с идентичными ветвями.

Одним из конструктивных приемов крупноблочного распараллеливания сложных задач является *распараллеливание по циклам*. Это позволяет представить процесс решения задачи в виде параллельных ветвей, полученных расщеплением цикла на части, число которых в пределе равно числу повторений цикла. На входе и выходе из цикла процесс вычислений — последовательный (см. рис. 3.3); доля последовательных участков в общем времени решения задачи незначительна. Говоря на языке физики, процесс вычислений можно представить в виде «пучностей» — параллельных ветвей, периодически перемежающихся «узлами» — последовательными процессами. На переходах между пучностями и узлами выполняются функции управления параллельным вычислительным процессом и обмены информацией между ветвями.

Как правило, последовательные участки имеют место в начале и конце параллельной программы. На начальном участке осуществляется инициирование программы и ввод исходных данных, а на конечном — вывод результатов (запись в архивные файлы). При достаточно полном воплощении принципов модели коллектива вычислителей допустимы реализации в системе параллельных ввода и вывода информации.

Последовательные участки в параллельной программе также могут иметь место, если используются негрупповые обмены информацией между ветвями (вычислителями).

Параллельная программа решения сложной задачи допускает свое представление в виде композиции

$$P_1 * P_2 * \ldots * P_i * \ldots * P_n,$$

в которой P_i — i-я ветвь программы, а n — допустимое число ветвей. Необходимо отметить, что каждая из ветвей P_i реализуется только на одном вычислителе коллектива, а информационные связи между ее операторами и операторами других ветвей $\{P_j\}$, $j \neq i$ осуществляются при помощи специальных операторов обмена информацией.

В дальнейшем для краткости наряду с терминами «параллельный алгоритм» и «параллельная программа» будем использовать соответственно «*P-алгоритм*» и «*P-программа*».

3.3.2. Параллельный алгоритм умножения матриц

Структуры параллельных алгоритмов (и, следовательно, *P*-программ) определяются графами информационных и управляющих связей, вершинам которых сопоставлены операторы ветвей, а ребрам — информационные и управляющие связи между операторами. Анализ прямых и итерационных методов вычислительной математики показывает, что в их основе лежат, как правило, операции над матрицами и векторами данных.

Проиллюстрируем методику крупноблочного распараллеливания на примере умножения матриц больших размеров. Пусть требуется построить параллельный алгоритм, вычисляющий произведение двух прямоугольных матриц:

$$\mathbf{A}[1:N,1:K]\times\mathbf{B}[1:K,1:M]=\mathbf{C}[1:N,1:M]$$

или, что то же самое,

$$\begin{Vmatrix} a_{11} & a_{12} & a_{1h} & a_{1K} \\ a_{21} & a_{22} & a_{2h} & a_{2K} \\ & & & \\ a_{i1} & a_{i2} & a_{ih} & a_{iK} \\ \cdots & & & \\ a_{N1} & a_{N2} & a_{Nh} & a_{NK} \end{Vmatrix} \times \begin{Vmatrix} b_{11} & b_{12} & b_{1j} & b_{1M} \\ b_{21} & b_{22} & b_{2j} & b_{2M} \\ \cdots & & & \\ b_{h1} & b_{h2} & b_{hj} & b_{hM} \\ \cdots & & & \\ b_{K1} & b_{K2} & b_{Kj} & b_{KM} \end{Vmatrix} =$$

$$= \begin{Vmatrix} c_{11} & c_{12} & c_{1j} & c_{1M} \\ c_{21} & c_{22} & c_{2j} & c_{2M} \\ \cdots & & & \\ c_{i1} & c_{i2} & c_{ij} & c_{iM} \\ \cdots & & & \\ c_{N1} & c_{N2} & c_{Nj} & c_{NM} \end{Vmatrix},$$

где элементы матрицы-произведения $\mathbf{C}[1:N, 1:M]$ вычисляются по формуле

$$c_{ij} = \sum_{h=1}^{K} a_{ih} b_{hj}, \quad i = \overline{1, N}, \quad j = \overline{1, M}. \tag{3.8}$$

Допустим также, что параллельный алгоритм ориентирован на реализацию в ВС, состоящей из n вычислителей. Пусть размеры $N \times K$ и $K \times M$ матриц **A** и **B** достаточно большие и таковы, что имеют место неравенства $N \gg n$, $K \gg n$, $M \gg n$. При параллельной обработке необходимо, чтобы каждый вычислитель производил расчет своих элементов матрицы **C**. При этом легко заметить, что размещение матриц **A** и **B** целиком в каждом вычислителе требует большой суммарной емкости памяти. Минимум емкости памяти будет достигнут, если каждая из исходных матриц будет разбита на n равных частей, и в каждый вычислитель будет размещено по одной такой части матриц **A** и **B**. Например, каждую из матриц **A** и **B** можно разрезать на n равных соответственно горизонтальных и вертикальных полос. Причем в первом вычислителе можно разместить строки 1, 2, ..., $]N/n[$ и столбцы 1, 2, ..., $]M/n[$; в l-м вычислителе — строки $(l-1)\cdot]N/n[+1$, $(l-1)\cdot]N/n[+2$, ..., $l]N/n[$ и столбцы $(l-1)\cdot]M/n[+1$, $(l-1)\cdot]M/n[+2$, ..., $l]M/n[$; в n-м вычислителе — строки $(n-1)\cdot]N/n[+1$, $(n-1)\cdot]N/n[+2, \ldots, n]N/n[$ и столбцы $(n-1)\cdot]M/n[+1$, $(n-1)\cdot]M/n[+2, \ldots, n]M/n[$ матриц **A** и **B** соответственно. Через $]x[$ обозначено такое ближайшее к x целое число, для которого справедливо неравенство $]x[\geqslant x$. При N и (или) M, некратном n, $n]N/n[-N$ и (или) $n]M/n[-M$ последних строк и (или) столбцов соответствующих полос для n-го вычислителя заполняются нулями. В результате будет получено однородное распределение данных по вычислителям коллектива (рис. 3.4).

Параллельный вычислительный процесс можно организовать следующим способом. Сначала первый вычислитель передает остальным вычислителям первую строку из своей полосы матрицы **A**. После этого каждый из вычислителей по формуле (3.8) рассчитывает $]M/n[$ элементов первой строки своей полосы для результирующей матрицы **C**. Затем первый вычислитель рассылает во все остальные вычислители вторую строку своей полосы матрицы **A** и производятся вычисления элементов второй строки матрицы **C** и так до тех пор, пока первый вычислитель не перешлет все строки своей части матрицы **A**. После этого пересылками будут заниматься последовательно второй вычислитель, третий вычислитель и далее до n-го вычислителя. Матрица **C** получается распределенной по вычислителям, причем в каждом будет своя вертикальная полоса (см. рис. 3.4). При этом следует учи-

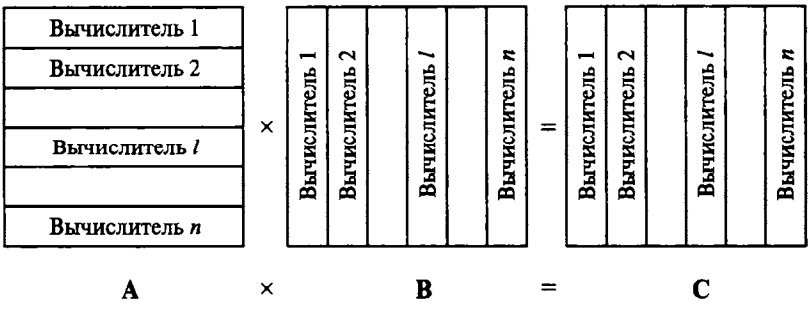


Рис. 3.4. Распределение данных по вычислителям ВС

тывать, что в результирующую матрицу **C** не должны включаться $n]N/n[- N$ последних строк из полученных вертикальных полос любого из вычислителей, а также $n]M/n[- M$ последних столбцов из полосы n-го вычислителя.

Итак, вследствие однородного распределения данных получены одинаковые ветви параллельного алгоритма, однако при этом ветви используют различные части данных. Поскольку для каждой ветви своих данных недостаточно, то ветви (точнее, реализующие их вычислители) вступают во взаимодействия (между ними осуществляются обмены информацией). Операторная схема l-й ветви P-алгоритма, реализуемая на вычислителе с номером l, $1 \leqslant l \leqslant n$, представлена на рис. 3.5.

Операции над матрицами и векторами (матрицами-строками или матрицами-столбцами) данных широко используются не только в прямых, но и в итерационных методах вычислительной математики. Так, алгоритм решения системы линейных алгебраических уравнений итерационным методом сводится к вычислению

$$x_i^{(k+1)} = b_i + \sum_{j=1}^{L} a_{ij} x_j^{(k)}, \quad i = 1, 2, \ldots, L,$$

где $x_i^{(k+1)}$ — значение i-й переменной, рассчитываемое в $(k + 1)$-й итерации; b_i и a_{ij} — коэффициенты, составляющие матрицу-столбец (вектор-столбец) и матрицу соответственно. Условием окончания итераций является

$$\max\left\{\left|x_i^{(k+1)} - x_i^{(k)}\right|\right\} < \delta.$$

Применяя методику крупноблочного распараллеливания и учитывая опыт построения P-алгоритма умножения матриц (см. рис. 3.5), легко построить параллельный алгоритм для последней задачи, в котором, например, l-я ветвь будет выполнять вычисления x_i для $i \in \{(l-1)]L/n[+ 1,$

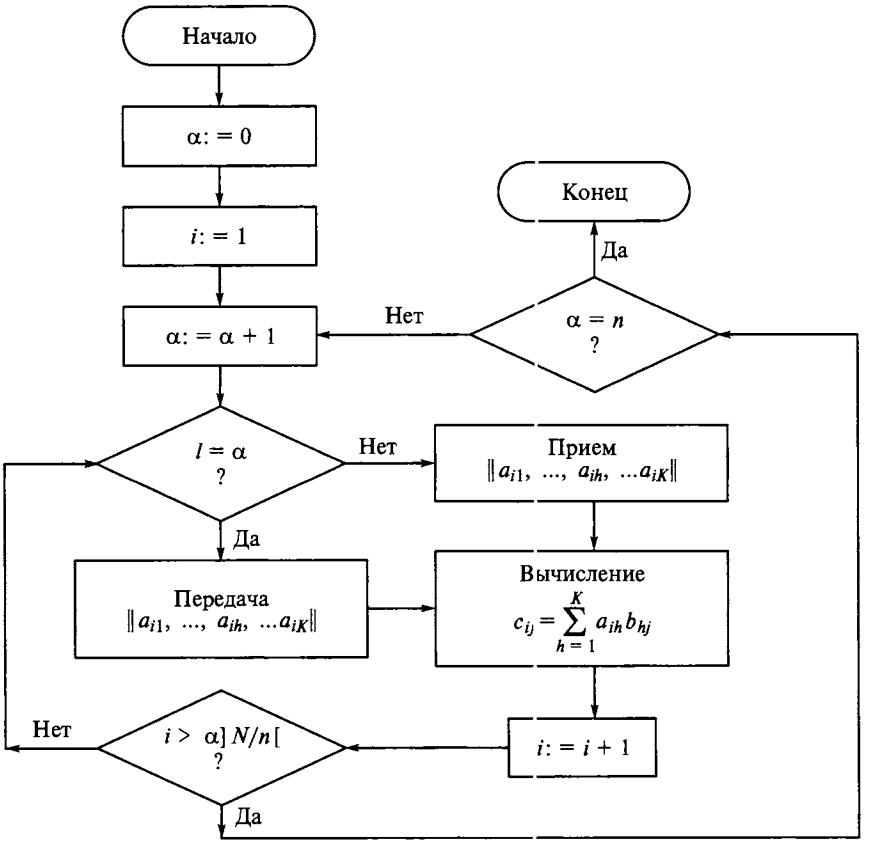


Рис. 3.5. Схема ветви параллельного алгоритма умножения матриц:

α — номер передающего вычислителя; {1, 2, ..., α − 1, α + 1, ..., n} — номера принимающих вычислителей; $]M/n[(l-1) < j \leqslant]M/n[l$

$(l-1)]L/n[+2, \ldots, l]L/n[\}$, где L — число переменных x_i или, по сути, предельно допустимое число ветвей P-алгоритма; n — число вычислителей в системе; $L \geqslant n$.

3.3.3. Показатели эффективности параллельных алгоритмов

Параллельный алгоритм (или P-программа) представляет собой композицию связных ветвей. Каждая ветвь — последовательность операторов как обычных (например, вычислительных и логических), так и реализующих взаимодействия (обмены информацией) между данной ветвью и другими.

Аксиоматически ясно, что все взаимодействия между ветвями — это накладные расходы, следовательно, чем меньше взаимодействий, тем выше эффективность параллельного алгоритма.

Введем показатели эффективности параллельных алгоритмов. *Коэффициентом накладных расходов* [5] называется

$$\varepsilon = t/T, \tag{3.9}$$

где t — время, расходуемое ВС на организацию и реализацию всех обменов информацией между ветвями; T — время, необходимое на выполнение арифметических и логических операций при реализации P-алгоритма.

Рассчитаем показатель (3.9) для алгоритма умножения матриц большого размера (см. рис. 3.5). Ясно, что после приема строки из K элементов матрицы **A** в каждом вычислителе выполняется $K \cdot]M/n[$ операций умножения и $(K-1)\cdot]M/n[$ операций сложения. При достаточно большом значении K можно считать, что на каждый принятый элемент матрицы **A** приходится $\rho =]M/n[$ операций умножения и сложения.

Пусть $t_{\text{п}}$ — время пересылки одного слова (элемента матрицы); $t_{\text{у}}$ и $t_{\text{с}}$ — время выполнения операций соответственно умножения и сложения. Тогда эффективность параллельного алгоритма (следовательно, и P-программы) умножения матриц большого размера можно характеризовать показателями:

$$\varepsilon = \frac{t_{\text{п}}}{\rho(t_{\text{у}} + t_{\text{с}})} = \frac{e}{\rho}, \quad e = \frac{t_{\text{п}}}{t_{\text{у}} + t_{\text{с}}}. \tag{3.10}$$

Очевидно, что максимум накладных расходов будет при $\rho = 1$, или, что то же самое, равенство $\varepsilon = e$ достигается при $n = M$. Величина $\rho = 1$ информирует о минимально допустимом размере матриц, при котором еще целесообразно решение задачи на n вычислителях. С другой стороны, при фиксированном n увеличение размера матриц приводит к росту ρ и, следовательно, к уменьшению накладных расходов. Ясно, что чем больше размеры матриц (чем больше объем вычислений), тем выше эффективность параллельного алгоритма. Сказанное относится и к задаче решения системы линейных алгебраических уравнений итерационным методом.

Имеющая место зависимость коэффициента накладных расходов (3.10) от количества элементов в матрицах и от числа вычислителей системы *в виде*

$$\varepsilon \to 0 \text{ при } \rho \to \infty$$

раскрывает суть методики крупноблочного распараллеливания сложных задач.

Коэффициент ускорения, как общепринято [10], для параллельного алгоритма вычисляется по формуле

$$\chi = \tau_1 / \tau_n, \tag{3.11}$$

где τ_1 — время решения задачи на одном вычислителе; τ_n — время выполнения соответствующего *P*-алгоритма в системе из *n* вычислений.

Коэффициентом ускорения параллельного алгоритма по сравнению с наилучшим последовательным алгоритмом называется отношение

$$\chi' = \tau_1' / \tau_n. \tag{3.12}$$

Здесь τ' — время реализации самого быстрого последовательного алгоритма на одном вычислителе.

Закон Амдала (Amdahl):

$$\chi^* \leqslant \frac{\kappa}{\delta + (1-\delta)/n},$$

где *n* — количество вычислителей, образующих ВС; δ — относительная доля операций *P*-программы, выполняемых последовательно, $0 \leqslant \delta \leqslant 1$, причем значения $\delta = 1$ или $\delta = 0$ соответствуют программам полностью последовательным или параллельным; κ — корректирующий коэффициент, $0 \leqslant \kappa \leqslant 1$.

Корректирующий коэффициент позволяет учесть специфику ВС, именно временные издержки, связанные с настройкой структуры системы, синхронизацией ветвей и обменом информацией между ними при реализации *P*-программы.

Закон Амдала позволяет оценить качество параллельной программы: насколько метод решения сложной задачи приспособлен к параллельному представлению. Этот закон применяют в специальном классе ВС — в конвейерных системах (см. разд. 3.4.2 и гл. 4). Показатель χ^* используют также для оценки ускорения, которое может быть получено при реализации *P*-программы на ВС с массовым параллелизмом.

Коэффициент ускорения (3.11) информирует о том, во сколько раз сокращается время решения на ВС задачи по выбранному алгоритму по сравнению с соответствующим временем для одного вычислителя. Опыт распараллеливания и решения сложных задач показывает, что далеко не всегда алгоритм, удобный для распараллеливания и дающий эффективную параллельную программу, будет самым быстрым при реализации на ЭВМ (или однопроцессорной ВС). Поэтому в последней ситуации целесообразно в качестве показателя эффективности параллельного алгоритма использовать коэффициент ускорения χ' [см. формулу (3.12)].

Наряду с коэффициентами ускорения (3.11) и (3.12) в параллельном программировании используют *коэффициент эффективности P-алгоритма*

$$E = \chi / n \tag{3.13}$$

и коэффициент эффективности P-алгоритма по отношению к наилучшему последовательному алгоритму

$$E' = \chi' / n. \tag{3.14}$$

Как правило, $\chi \leqslant n$, $\chi' \leqslant n$, следовательно, $E' \leqslant E \leqslant 1$. Если P-алгоритм обеспечивает максимальное ускорение вычислительного процесса, т. е. если $\chi = n$, то коэффициент эффективности параллельного алгоритма $E = 1$.

Одной из целей распараллеливания сложных задач является достижение максимума коэффициента ускорения: $\max \chi = n$ (3.11). Однако на практике этот максимум достигается не всегда. Главными факторами, ограничивающими ускорение, могут быть следующие:

- время, расходуемое на синхронизацию параллельных ветвей (процессов) и на обмены информацией между ними, а также на «конфликты» памяти (обусловленные, например, из-за ее общедоступности либо, наоборот, из-за ее распределенности);
- несбалансированность (неоднородность) нагрузки вычислителей и (или) невозможность построения P-алгоритма с числом ветвей, равным n (числу вычислителей в ВС).

Методика крупноблочного распараллеливания сложных задач позволяет создавать P-алгоритмы, эффективно реализуемые на параллельных ВС.

Следует отметить, что модели вычислителя и коллектива вычислителей являются границами для варьирования архитектурных возможностей средств обработки информации. Одна граница — это ЭВМ, другая — ВС с массовым параллелизмом. Существуют средства, которые одновременно являются предельным вариантом модификации последовательной ЭВМ и простейшей конфигурацией параллельной ВС. Эти средства называют конвейерными ВС, либо векторными компьютерами, в них аппаратурно реализованы команды для оперирования не только со скалярными величинами, но и с векторами данных.

Для векторных компьютеров могут быть введены показатели эффективности, аналогичные (3.11) и (3.12) — *коэффициенты ускорения алгоритма при использовании векторных операций:*

$$\Psi = \tau / \mathcal{T}; \quad \Psi' = \tau^* / \mathcal{T}, \tag{3.15}$$

где τ — время выполнения алгоритма на векторном компьютере при использовании только скалярной арифметики; $\mathcal{T}$ — время реализации алго-

ритма при применении векторной арифметики; τ^* — время выполнения быстрейшего последовательного алгоритма на компьютере.

«Парадокс» параллелизма:

$$\chi > n, \quad E > 1 \tag{3.16}$$

состоит в достижении ускорения (3.11) и эффективности (3.13) параллельного алгоритма, превышающих значения n и 1 соответственно. Говоря другими словами, *«парадокс» параллелизма выражается в более чем линейном росте производительности параллельной ВС с увеличением количества n ее вычислителей.*

«Парадокс» параллелизма (3.16) был впервые обнаружен и исследован на многих классах вычислительных, стохастических и информационно-логических задач специалистами Отдела вычислительных систем СО АН СССР при работе на советских параллельных ВС в конце 60-х — начале 70-х годов XX в. Он постоянно шокирует специалистов и исследователей, осваивающих параллельные вычислительные технологии.

«Парадокс» параллелизма по сути не является таковым, а неравенства (3.16) объясняются следующими факторами:

- эффектом памяти;
- возможностью применения (априори параллельных) методов решения задач, реализация которых затруднительна на последовательной ЭВМ (например, снова из-за оперативной памяти, точнее, ее ограниченного объема).

Поясним первый фактор. Вычислительная система — это коллектив вычислителей, каждый из которых имеет свою локальную оперативную память. Сеть связей между вычислителями делает память всех вычислителей общедоступной. Если суммарная локальная память имеет достаточно большую емкость и допускает «вложение» параллельного алгоритма и исходных данных и, следовательно, нет необходимости использовать внешнюю (более медленную) память, то будет достигнут минимум времени τ_n (3.11) реализации алгоритма на ВС при заданном числе n вычислителей. С другой стороны, если сложность задачи достаточно высока (т. е. ее последовательный алгоритм и исходные данные не размещаются в локальной памяти отдельного вычислителя), и если она допускает решение на одном вычислителе только при условии использования внешней памяти, то будет достигнуто время решения τ_1, отличающееся от минимального значения. Наконец, если время передачи слов между вычислителями ВС сравнимо с временем обращения к локальной оперативной памяти в отдельном вычислителе, то, очевидно,

$$\tau_1 = A_n n \tau_n,$$

где коэффициент $A_n > 1$ и, следовательно,

$$\chi > n, \quad E > 1.$$

Таким образом, показатели (3.9)–(3.14) составляют достаточно полный набор, позволяющий оценить эффективность реализации параллельных алгоритмов на ВС.

3.3.4. Понятие о сложных задачах

Эффективность функционирования ВС, как следует из разд. 3.3.2 и 3.3.3, зависит от количества операций, которые требуется выполнить при решении задач и от числа вычислителей, на котором реализуются *P*-алгоритмы (точнее, *P*-программы), от степени адекватности вложения структурных схем алгоритмов решения в структуры ВС. Среди показателей качества *P*-алгоритмов используют, как уже отмечалось, коэффициент накладных расходов (3.9), который можно представить в следующем виде:

$$\varepsilon(V, n) = t(V, n) / \mathrm{T}(V, n), \tag{3.17}$$

где V — количество операций, которые необходимо выполнить при решении задачи на ВС; n — число параллельных ветвей или число вычислителей, на которых решается задача, $n \geqslant 2$; $t(V, n)$ — время, затрачиваемое на синхронизацию параллельных ветвей алгоритма, на настройку (программирование структуры) системы, на реализацию обменов информацией между ветвями (вычислителями); $\mathrm{T}(V, n)$ — время, расходуемое системой собственно на вычисления.

На основе анализа задач и опыта их решения (с использованием методики крупноблочного распараллеливания) на ВС установлено, что при $n = \mathrm{const}$ показатель (3.17) асимптотически стремится к нулю с ростом объема операций в задаче, т. е. имеет место $\varepsilon(V, n) \to 0$ при $V \to \infty$. Значения $\varepsilon(V, n)$ будут практически удовлетворительными при выполнении неравенства

$$V \geqslant n \cdot 10^{l}, \tag{3.18}$$

где l — эмпирический коэффициент, $l \geqslant 1$.

Очевидно, что имеет место зависимость l от быстродействия ν каналов связи между вычислителями: $l \to 1$ при $\nu \to \nu^{*}$, где $1/\nu^{*}$ — время обращения к локальной памяти в вычислителе. При удовлетворении неравенства (3.18) достигается адекватное размещение параллельной программы на системе из n вычислителей (при произвольной структуре сети связей) и обеспечивается эффективное использование этих вычислителей.

Таким образом, если объем операций V, связанных с решением задачи, на несколько порядков превышает количество вычислителей n, на которых

должно осуществляться ее решение, то достигается эффективное функционирование системы.

Задачу, для которой выполняется неравенство (3.18), будем называть сложной, или системной, или трудоемкой, или с большим объемом вычислений. Сложность задачи будем характеризовать количеством операций, которые необходимо выполнить при ее решении. Задача тем сложнее, чем больше V. Задачу, которая имеет небольшой объем вычислений и, следовательно, не допускает эффективного распараллеливания, будем называть *простой*. Простая задача требует для своего решения одного вычислителя.

3.3.5. Схемы обмена информацией между ветвями параллельных алгоритмов

Пусть P-алгоритм (и, следовательно, P-программа) состоит из n ветвей. Аксиоматически ясно, что произвольные схемы обмена информацией в P-алгоритмах могут быть реализованы с помощью одного *дифференцированного обмена*. При этом обмене осуществляется передача информации из одной ветви в любую другую ветвь или из одного вычислителя (передатчика) к другому вычислителю (приемнику) (рис. 3.6, *а*). Видно, что при дифференцированном обмене в работе находятся только два вычислителя, а остальные простаивают. Ясно, что такой обмен не может быть отнесен к числу эффективных, т. е. таких, которые сводят к минимуму простои ресурсов ВС.

Существуют ли «коллективные» (или «групповые») обмены информацией, при которых в работе находятся все вычислители? Анализ P-алгоритмов и P-программ показал, что многообразие встречающихся в них схем обмена информацией между ветвями сводится к пяти видам: дифференцированному (ДО), трансляционному (ТО), трансляционно-циклическому (ТЦО), конвейерно-параллельному (КПО) и коллекторному обменам (КО).

При *трансляционном обмене (One-to-all Broadcast)* осуществляется передача одной и той же информации из одной (любой) ветви одновременно во все остальные ветви параллельного алгоритма (рис. 3.6, *б*).

Трансляционно-циклический обмен (All-to-all Broadcast) реализует трансляцию информации из каждой ветви во все остальные. Следовательно, если трансляционный обмен выполняется за 1 такт, то трансляционно-циклический — за n тактов.

Конвейерно-параллельный обмен обеспечивает передачу информации между соседними ветвями; он выполняется за два такта (рис. 3.6, *в*). Так, например, при четном n в первом такте осуществляется передача информации из ветвей $P_1, P_3, \ldots, P_{i-1}, \ldots, P_{n-3}, P_{n-1}$ соответственно в ветви $P_2, P_4, \ldots, P_i, \ldots, P_{n-2}, P_n$; во втором такте информация из последней по-

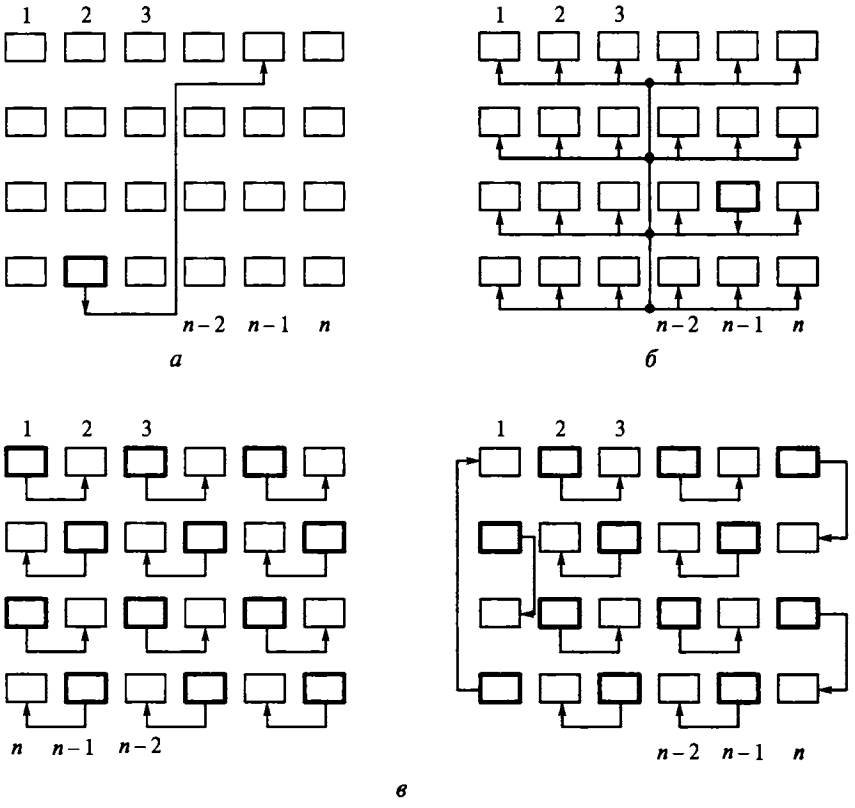


Рис. 3.6. Схемы обмена информацией между ветвями параллельного алгоритма:
а — дифференцируемый обмен; *б* — трансляционный обмен; *в* — конвейерно-параллельный обмен; □ — приемник; ■ — передатчик

следовательности ветвей поступает соответственно в ветви $P_3, P_5, \ldots$ $\ldots, P_{i+1}, \ldots, P_{n-1}, P_n$.

Коллекторный обмен представляет собой инвертированный трансляционный обмен (см. рис. 3.6, *б*), т. е. вычислители, которые были приемниками, превращаются в передатчики, и наоборот, вычислитель-передатчик становится приемником. При коллекторном обмене в одну ветвь последовательно собирается информация из $l \leqslant n$ ветвей. Такой обмен требует l тактов и реализуется как последовательность из l дифференцированных обменов.

В параллельных алгоритмах умножения матриц и решения системы линейных алгебраических уравнений итерационным методом используются трансляционно-циклические схемы обмена информацией (см. рис. 3.5).

Статистическая информация, отражающая частоту использования схем обмена (ДО, ТО, ТЦО, КПО, КО) при реализации крупноблочных па-

раллельных алгоритмов или программ, представлена ниже:

Схема обмена	ДО	ТО	ТЦО	КПО	КО
Частота использования, %	2	17	40	34	7

Таким образом, групповые схемы обмена информацией между ветвями параллельных алгоритмов или программ (именно ТО, ТЦО, КПО) составляют более 90 % от общего количества обменов. Это позволяет создавать высокоэффективные параллельные алгоритмы и обеспечивает одновременную работу всех вычислителей системы. Следовательно, методика крупноблочного распараллеливания сложных задач сводит к минимуму простои вычислителей системы при обменах информацией.

3.3.6. Опыт применения методики крупноблочного распараллеливания сложных задач

Многолетняя эксплуатация (с 1965 г.) советских параллельных ВС и опыт конструирования в Отделе вычислительных систем СО РАН параллельных алгоритмов и программ на основе методики крупноблочного распараллеливания сложных задач позволяет сделать следующие выводы.

1. Сложные задачи допускают построение параллельных алгоритмов, состоящих из идентичных ветвей. Разработка таких алгоритмов характеризуется простотой, а процесс написания любой параллельной программы сводится к написанию последовательной для одной ветви соответствующего параллельного алгоритма.

2. При распределении исходных данных между ветвями параллельного алгоритма эффективным и достаточным является принцип однородного расчленения массивов информации, при котором обеспечивается равенство объемов составных частей, единообразие при введении избыточности в каждой из них и взаимно однозначное соответствие между их номерами и номерами ветвей.

3. Для построения параллельных алгоритмов достаточно использовать пять схем обмена информацией между ветвями дифференцированную, трансляционную, трансляционно-циклическую, конвейерно-параллельную и коллекторную схемы (см. разд. 3.3.5). Осуществление обменов по данным схемам в общем случае связано с транзитным прохождением информации через вычислители, лежащие на пути между взаимодействующими вычислителями системы. Однако если эти схемы воспринимать как виртуальные, то реализацию обменов можно проводить посредством локальных взаимодействий между вычислителями (пересылок информации между соседними вычислителями).

4. Простота организации обменов по рассмотренным схемам позволяет использовать в основном регулярные (однородные) структуры ВС и кон-

фигурации-подсистемы (программно-настраиваемые в пределах ВС) в виде «линейки», «кольца», «решетки», гиперкуба и тора.

5. Для записи параллельных алгоритмов решения сложных задач эффективны версии распространенных языков высокого уровня FORTRAN, ALGOL, C и другие, которые являются расширениями соответствующих последовательных языков средствами организации взаимодействий между вычислителями ВС. Объем системного расширения составляет несколько процентов от общего объема трансляторов. Экспертно установлено, что сложность программирования на параллельных языках имеет тот же порядок, что и на последовательных языках; увеличение трудоемкости не превышает 10 % от трудоемкости последовательного программирования.

6. Простота схем обмена и распределения данных по ветвям ведет к простоте записи и реализации параллельных программ. Затраты на организацию взаимодействий между ветвями составляют менее 10 % общего объема программы.

7. Трансляционная, трансляционно-циклическая и конвейерно-параллельные схемы обмена информацией обеспечивают одновременную работу вычислителей системы и составляют не менее 90 % всех схем, реализуемых в процессе выполнения параллельных программ сложных задач. Время обмена информацией между ветвями параллельной программы, отнесенное к общему времени ее выполнения, асимптотически стремится к нулю с ростом объема исходных данных.

Приведенные статистические характеристики по реализации параллельных программ получены в результате эксплуатации, в частности, систем «Минск-222» (1965 г.), МИНИМАКС (1975 г.), СУММА (1976 г.) и трех моделей семейства МИКРОС (1986 г., 1992 г., 1996 г.), разработанных Отделом вычислительных систем СО АН СССР (СО РАН) совместно с промышленными организациями. Первая система имела одномерную (кольцевую) структуру и строилась из вычислителей-двухполюсников (степень вершины $\upsilon = 2$). Вторая ВС МИНИМАКС имела две сети связи — одномерную управляющую (для передачи настроечной информации, $\upsilon = 2$) и двумерную информационную (для обмена операндами, $\upsilon = 4$). На производительность ВС МИНИМАКС практическое влияние оказывала двумерная сеть. Следовательно, можно считать, что каждая вершина в системе МИНИМАКС имела степень $\upsilon = 4$. Вычислительная система СУММА формировалась из вычислителей-трехполюсников ($\upsilon = 3$). Вычислительные системы МИКРОС компоновались из вычислителей-многополюсников, причем допускалось варьирование степени вершины в пределах от 2 до 6 ($\upsilon = \overline{2, 6}$).

Накопленный опыт параллельного программирования и сделанные выше выводы являются основой для развития параллельных (отказоустойчивых) вычислительных технологий XXI столетия.

Замечание. Следует особо подчеркнуть теоретическую и практическую значимость сформулированных выводов и, в частности, статистики использования схем обмена информацией между вычислителями ВС. Фактически закономерности параллельного крупноблочного программирования были установлены в Отделе вычислительных систем Института математики СО АН СССР еще в середине 1960-х годов, при работе на первой в мире ВС с программируемой структурой «Минск-222» (см. § 7.3 или [5]).

Начиная с 1990-х годов широко используется при производстве параллельных программ стандарт MPI (Message Passing Interface — интерфейс передачи сообщений). Если изучить процедуры MPI [11, 12], то легко обнаружить их совместимость не только с принципами коллектива вычислителей, но и со схемами обмена информацией между ветвями параллельных алгоритмов. Так, например, дифференцированный обмен в стандарте MPI нашел воплощение в виде функций двухточечной передачи и приема данных (Point-to-Point Communications): функций MPI_Send и MPI_Recv соответственно. Такие обмены данными, как трансляционный, трансляционно-циклический и коллекторный реализуются в MPI при помощи функций коллективных взаимодействий процессов (Collective Communications), например MPI_Bcast, MPI_Alltoall и MPI_Gather соответственно.

Схемы обмена информацией между ветвями *P*-алгоритмов и другие системные взаимодействия даже в первой ВС «Минск-222» поддерживались соответствующей библиотекой программ. Это позволило использовать широко распространенные языки для написания *P*-программ. Реализация стандарта MPI — это тоже библиотека функций (программ), предназначенная для поддержки работы параллельных процессов (в терминах передачи сообщений). Существуют реализации MPI для языков FORTRAN, C и C++ для различных архитектур ВС, использующих в своих узлах-вычислителях операционные системы Unix, Linux и Windows.

Таким образом, *отечественные пионерские работы в области параллельных вычислений и результаты эксплуатации методики крупноблочного распараллеливания сложных задач предвосхитили западную MPI-технологию.*

3.4. Концептуальное понятие и классификация архитектур вычислительных систем

Вычислительные системы — современное достижение индустрии обработки информации. Рассмотрим подробнее понятие о ВС, их архитектурную классификацию и основные тенденции развития архитектуры средств вычислительной техники.

3.4.1. Понятие о ВС

Дать достаточно общее и четкое определение системы вообще не представляется возможным. Для того чтобы несколько облегчить интуитивное понимание систем, приведем для них неформальные определения.

В энциклопедическое понятие системы (от греч. systema — целое, составленное из частей; соединение) вкладывается множество элементов, находящихся в отношениях и связях друг с другом, которое образует определенную целостность, единство. При определении понятия системы необходимо учитывать теснейшую взаимосвязь его с понятиями целостности, структуры, связи, элемента, подсистемы и др. Авторы книги [5] дают следующее определение: «Системой называется совокупность взаимосвязанных элементов, реализация функций которыми и функциональное взаимодействие между которыми позволяют оптимальным образом достичь общей цели». Понятие системы имеет чрезвычайно широкую область применения: практически каждый объект может рассматриваться как система, исключением не являются и средства вычислительной техники, в частности ЭВМ и микропроцессорные БИС. Конкретизируем понятие вычислительной системы, считая его частным по отношению к приведенным.

Предварительно заметим, что коллектив вычислителей обладает свойствами системы и является средством, которое наиболее удобно и перспективно для решения сложных (системных) задач. В дальнейшем *под вычислительной системой будем понимать средство обработки информации, базирующееся на модели коллектива вычислителей.* Следовательно, ВС — это композиция аппаратурно-программных средств, предназначенная для параллельной обработки данных.

При достаточно общей трактовке под ВС понимается совокупность взаимосвязанных и одновременно функционирующих аппаратурно-программных вычислителей, которая способна не только реализовать (параллельный) процесс решения сложной задачи, но и в процессе работы автоматически настраиваться и перестраиваться с целью достижения адекватности между своей структурно-функциональной организацией и структурой и характеристиками решаемой задачи.

3.4.2. Классификация архитектур ВС

Модель коллектива вычислителей (3.1)–(3.7) есть диалектическое обобщение модели вычислителя (2.1)–(2.4), следовательно, ВС в сравнении с ЭВМ Дж. фон Неймана являются принципиально новыми средствами техники обработки информации, средствами с качественно новыми архитектурными возможностями. В самом деле, в архитектурном плане выделяют четыре класса архитектур вычислительных средств: SISD, MISD, SIMD, MIMD (ОКОД, МКОД, ОКМД, МКМД). Только первый класс, а именно SISD (Single Instruction stream / Single Data stream) или ОКОД (Одиночный поток Команд и Одиночный поток Данных) относится к ЭВМ. Под потоком команд понимается любая их последовательность, поступающая для испол-

нения вычислительным средством (ЭВМ или процессором, в случае SISD-архитектуры). При выполнении команд потока требуются операнды (данные), следовательно, поток команд «порождает» поток данных. Итак, SISD-архитектура предопределяет такое функционирование ЭВМ, при котором один поток команд управляет обработкой одного потока данных (см. разд. 2.3.2 и рис. 2.2).

Архитектуры MISD, SIMD, MIMD относятся к ВС. В этих архитектурах имеет место множественность потоков или (и) команд, или (и) данных. Множественность характеризуется количеством одновременно реализуемых потоков команд или (и) данных. Архитектура MISD (Multiple Instruction stream / Single Data stream) или МКОД (Множественный поток Команд и Одиночный поток Данных) позволяет нескольким потокам команд обрабатывать один поток данных. Архитектура SIMD (Single Instruction stream / Multiple Data stream) или ОКМД (Одиночный поток Команд и Множественный поток Данных) предоставляет возможность одному потоку команд обрабатывать несколько потоков данных. Архитектура MIMD (Multiple Instruction stream / Multiple Data stream) или МКМД (Множественный поток Команд и Множественный поток Данных) допускает обработку несколькими потоками команд нескольких потоков данных.

Приведенная классификация архитектур средств обработки информации была предложена профессором Стенфордского университета США М. Дж. Флинном (M.J. Flynn) в 1966 г. и получила широкое распространение.

В архитектурах классов MISD, SIMD, MIMD (рис. 3.7) допустимо построение нескольких типов ВС, среди которых для целей данной книги наибольший интерес представляют: конвейерные ВС, матричные ВС, мультипроцессорные ВС, распределенные ВС и ВС с программируемой структурой.

Конвейерные ВС — системы, архитектура которых является предельным вариантом эволюционного развития последовательной ЭВМ и простейшей версией модели коллектива вычислителей. В основе таких систем лежит конвейерный (или цепочечный) способ обработки информации, а их функциональная структура представляется в виде последовательности связанных элементарных блоков обработки (ЭБО) информации. Все блоки работают параллельно, но каждый из них реализует лишь свою операцию над данными одного и того же потока. Сказанное позволяет относить конвейерные ВС к MISD-системам (см. рис. 3.7). Реальные промышленные высокопроизводительные ВС являются, как правило, мультиконвейерными. В них единое управляющее устройство (управляющая подсистема или ЭВМ — Host computer, контроллер и т. п.) формирует один поток команд и несколько параллельных потоков данных на подсистемы-конвейеры. Последнее обстоятельство позволяет относить такие мультиконвейерные ВС к системам с архитектурой SIMD.

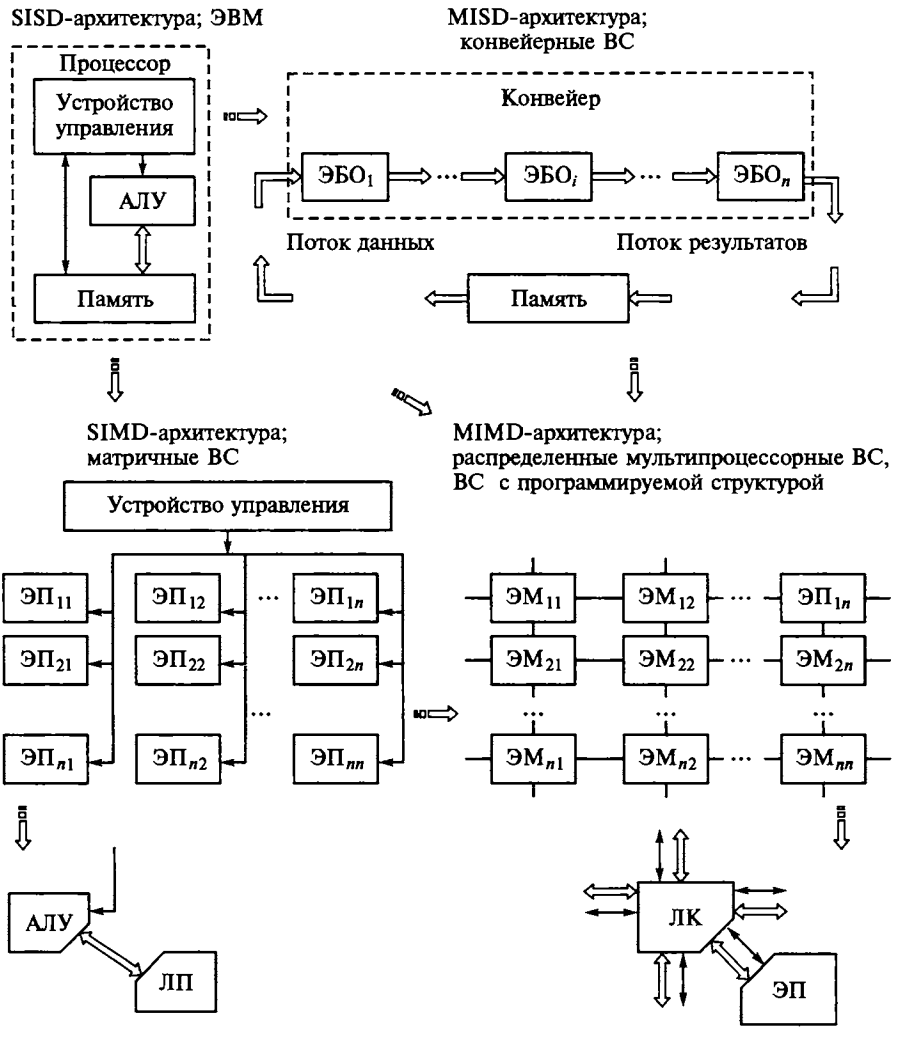


Рис. 3.7. Развитие архитектуры вычислительных средств:

АЛУ — арифметико-логическое устройство; ЭБО — элементарный блок обработки; ЭП — элементарный процессор; ЭМ — элементарная машина; ЛП — локальная память; ЛК — локальный коммутатор; → — поток команд; ⇨ — поток данных; ▫⇨ — направление трансформации архитектуры

Матричные ВС основываются на принципе массового параллелизма, в них обеспечивается возможность одновременной реализации большого числа операций на элементарных процессорах (ЭП), объединенных в матрицу. Каждый ЭП представляет собой композицию из арифметико-логического устройства (АЛУ) и локальной памяти (ЛП); последняя предназначается для

хранения части данных (но не части программы или параллельной ветви!). Поток команд на матрицу ЭП формируется устройством управления (следовательно, оно имеет в своем составе память для хранения программ обработки данных). Такие ВС рассчитаны, в частности, на решение задач матричной алгебры. Они имеют SIMD-архитектуру в классическом виде.

Современные высокопроизводительные матричные ВС — масштабируемые, в них, как правило, можно варьировать количество матриц ЭП и устройств управления. Такие ВС уже следует относить к системам с архитектурой MIMD.

Мультипроцессорные ВС — обширная группа систем, в которую, в частности, могут быть включены конвейерные и матричные ВС (а также многомашинные ВС). Однако принято к мультипроцессорным ВС относить системы с MIMD-архитектурой, которые состоят из множества (не связанных друг с другом) процессоров и общей (возможно и секционированной, модульной) памяти; взаимодействие между процессорами и памятью осуществляется через коммутатор (общую шину и т. п.), а между процессорами — через память.

Распределенные ВС — мультипроцессорные ВС с MIMD-архитектурой, в которых нет единого ресурса (общей памяти). Распределенная ВС основывается на принципах модульности и близкодействия (см. разд. 3.2.1). Основные компоненты распределенной ВС (такие, как коммутатор, устройство управления, арифметико-логическое устройство или процессор, память) допускают представление в виде композиции из одинаковых элементов (локальных коммутаторов и устройств управления, локальных процессоров и модулей памяти).

Примером промышленной реализации распределенных ВС являются транспьютерные системы. *Транспьютерная ВС* — композиция из одинаковых взаимосвязанных микропроцессоров, называемых транспьютерами. В состав транспьютера входят локальные процессор и память, а также локальные средства коммутации и линки (Link — связь), позволяющие организовать взаимодействия с другими транспьютерами.

Вычислительные системы с программируемой структурой полностью основываются на модели коллектива вычислителей и являются композицией взаимосвязанных элементарных машин (ЭМ). Каждая ЭМ в своем составе обязательно имеет локальный коммутатор (ЛК), процессор и память; может иметь также внешние устройства. Локальная память ЭМ предназначается для хранения и части данных, и, главное, ветви параллельной программы. Архитектура ВС с программируемой структурой относится к типу MIMD. Такие ВС по своим потенциальным архитектурным возможностям не уступают ни одному из перечисленных выше классов систем. Они прежде всего ориентированы на распределенную обработку информации; эффективны и при конвейерной, и при матричной обработке. При распределен-

ном способе обработки данных на ВС полностью используются возможности MIMD-архитектуры. При конвейерном и матричном способах обработки данных архитектура MIMD виртуально трансформируется соответственно в архитектуру MISD и SIMD. Системы с программируемой структурой рассчитываются на работу во всех основных режимах: решения сложной задачи, обработки наборов задач, обслуживания потоков задач, реализации функций вычислительной сети.

Концепция ВС с программируемой структурой была сформулирована в Сибирском отделении АН СССР, первая система («Минск-222») была построена в 1965–1966 гг. (см. § 7.3 или [5]).

Кластерные ВС — разновидность мультипроцессорных систем, интуитивная оценка архитектурных возможностей которых вытекает из семантики слова кластер (Cluster — группа). Такие системы получили широкое распространение уже в 90-х годах XX в. В списке Top500, т. е. 500 наиболее мощных суперкомпьютеров мира, кластерные системы доминируют; например в 28-й и 29-й редакциях списка их количество составляет соответственно 361 и 373.

Термин «вычислительный кластер», по-видимому, был впервые введен DEC (Digital Equipment Corporation). По определению DEC, кластер — это группа компьютеров, которые связаны между собой и функционируют как единое средство обработки информации. Из приведенного определения видно, что корпорация DEC, по сути, ввела синоним термину «вычислительная система», а не особый тип средств обработки информации. Для создания кластерных ВС используются и MISD-, и SIMD-, и MIMD-архитектуры, различные функциональные структуры и конструктивные решения.

В наиболее общей трактовке *кластерная ВС, или кластер*, — это композиция множества вычислителей, сети связей между ними и программного обеспечения, предназначенная для параллельной обработки информации (в частности, реализации параллельных алгоритмов решения сложных задач). При формировании кластерной ВС могут быть использованы как стандартные промышленные компоненты, так и специально созданные средства. Однако в кластерных ВС, как правило, превалируют массовые аппаратурно-программные средства. Последнее, по существу, является принципом конструирования кластерных ВС, обеспечивающим их высокую технико-экономическую эффективность.

Начало XXI в. ознаменовалось созданием пространственно-распределенных мультикластерных ВС как макроколлективов (см. разд. 3.1.4) рассредоточенных кластеров, взаимодействующих между собой через локальные и глобальные сети (включая всемирную сеть Internet).

Суперкомпьютеры — вычислительные средства, характеризующиеся рекордной эффективностью (производительностью, надежностью, живуче-

стью и технико-экономической эффективностью) для фиксированного этапа развития индустрии обработки информации. Они могут быть основаны на любой из архитектурных парадигм. Современные суперкомпьютеры (10^{12}...10^{15} опер./с) являются вычислительными системами с массовым параллелизмом и, если придерживаться терминологической строгости, их следует называть *суперВС*.

Достигнутый уровень развития ВТ и интегральной технологии позволяет производить персональные суперВС. Осенью 2006 г. компания Intel продемонстрировала кремниевую пластину из 80 чипов, каждый из которых содержал по 80 ядер (80 процессорных элементов). В начале 2007 г. компания выпустила первый рабочий образец 80-ядерного микропроцессора. Данный микропроцессор обладает производительностью 10^{12} операций с плавающей запятой в секунду (1 TeraFLOPS) и скоростью обмена данными между ядрами порядка нескольких терабит в секунду и характеризуется низким электроэнергопотреблением — 62 Вт. Таким образом, микропроцессор сам по себе уже является суперВС.

Для оценки значимости достижений компании Intel отметим, что терафлопсный суперкомпьютер конца 90-х годов XX в. состоял примерно из 10 000 процессоров Intel Pentium Pro, характеризовался потребляемой мощностью 500 кВт и занимал площадь 185 м^2

В последующих главах систематически будут рассмотрены архитектуры ВС (см. рис. 3.7). Предварительно заметим, что архитектура современных высокопроизводительных ВС, как правило, отличается от своих изначальных канонов. Архитектура одних и тех же систем в зависимости от уровня рассмотрения их функциональных структур может выглядеть и как MISD, и как SIMD, и как MIMD. Таким образом, можно констатировать, что мультиархитектура стала парадигмой при конструировании высокопроизводительных ВС.

Обобщая опыт развития индустрии обработки информации, можно заключить, что независимо от изначальной архитектурной парадигмы фирмы-создатели суперкомпьютеров к началу XXI столетия перешли на платформу ВС с программируемой структурой.

4. КОНВЕЙЕРНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Конвейерные ВС относились к числу самых популярных высокопроизводительных средств обработки информации в 70-х и 80-х годах XX в. Они обеспечивали быстродействие порядка $10^8 ... 10^9$ опер./с, которое в то время воспринималось как рекордно высокое. Последнее позволяло называть конвейерные ВС как суперЭВМ (Supercomputers). Конвейерные ВС имели аппаратурно реализованные команды, позволявшие выполнять операции над векторами данных. Поэтому такие ВС называли также векторными компьютерами (Vector Computers).

В главе рассмотрена каноническая структура конвейерного процессора и промышленные ВС на ее основе. Описаны параллельно-векторные ВС (PVP-системы), которые представляются связным множеством одновременно функционирующих конвейерных (векторных) процессоров, а также конвейерные ВС с массовым параллелизмом (MPP-системы).

Наконец, читателю будет предоставлена возможность изучить архитектуру сверхвысокопроизводительных ВС первого десятилетия XXI в., которые основываются на достижениях и PVP-, и MPP-систем.

4.1. Каноническая функциональная структура конвейерного процессора

В конвейерных ВС основной объем операций по обработке данных выполняется одним или несколькими конвейерными процессорами (или кратко: конвейерами). Конвейеры оперируют с векторами данных, которые являются *одномерными массивами или одномерными упорядоченными совокупностями элементов данных одного типа.* Если воспользоваться терминами алгебры матриц, то вектор данных — это или столбец, или строка, или диагональ двумерной матрицы, либо матрица-столбец или матрица-строка вида:

$$\mathbf{A} = \|A_1, A_2, \ldots, A_i, \ldots, A_n\| = (A_1, A_2, \ldots, A_i, \ldots, A_n),$$

где A_i — i-й компонент (или элемент, или элемент-операнд, или скалярная величина, или просто «скаляр», или число), $i = \overline{1, n}$.

В конвейере векторные операции реализуются аппаратурно, поэтому его называют также *векторным процессором*. При этом всегда предусматриваются операции покомпонентного сложения и покомпонентного умножения двух векторов, а также либо покомпонентное деление векторов, либо формирование вектора из чисел, обратных компонентам данного вектора. Для более сложных операций (например, покомпонентного извлечения квадратного корня) могут быть введены свои векторные команды. В конвейеры может быть заложена возможность реализации триад (Linked Triad — сцепленных триад), т. е. операций вида

$$\mathbf{A} + \alpha\mathbf{B},$$

где **A** и **B** — векторы данных; α — скаляр; $\alpha\mathbf{B}$ — вектор, компоненты которого равны соответствующим компонентам **B**, умноженным на α. Возможны и другие разновидности триады, например:

$$(\mathbf{A} + \alpha)\mathbf{B},$$

где $\mathbf{A} + \alpha$ — вектор, получаемый из **A** путем прибавления числа α к каждому компоненту.

В основу функциональной организации конвейера положен принцип сегментирования арифметико-логического устройства на «специализированные» части. Каждая из таких частей-сегментов должна быть ориентирована на реализацию вполне определенной операции (макро- или микрооперации, в частности) над парой скаляров-операндов (каждый из которых является элементом своего вектора).

Конвейер (Pipeline) организуется, в общем случае, как цепочка из элементарных блоков обработки информации (ЭБО$_i$) и памяти (ЭБП$_i$), $i = \overline{1, n}$ (рис. 4.1). Каждый из блоков ЭБО$_i$, $i = \overline{1, n}$, осуществляет частичное преобразование $\varphi_i(\mathbf{A}, \mathbf{B})$ компонентов векторов-операндов:

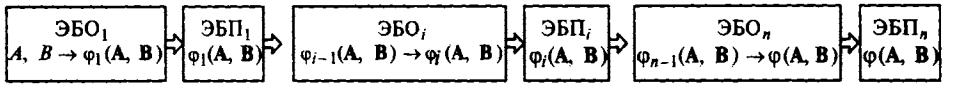
$$\mathbf{A} = \|A_1, A_2, \ldots, A_i, \ldots, A_n\|, \quad \mathbf{B} = \|B_1, B_2, \ldots, B_i, \ldots, B_n\|.$$

Конвейер в целом обеспечивает реализацию достаточно сложного преобразования $\varphi(\mathbf{A}, \mathbf{B})$, являющегося результатом цепочки преобразований:

$$\mathbf{A}, \mathbf{B} \to \varphi_1(\mathbf{A}, \mathbf{B}) \to \cdots \to \varphi_i(\mathbf{A}, \mathbf{B}) \to \cdots \to \varphi_n(\mathbf{A}, \mathbf{B}) = \varphi(\mathbf{A}, \mathbf{B}).$$

Блоки ЭБП$_i$ ($i = \overline{1, n-1}$) и блок ЭБП$_n$ используются для хранения промежуточных результатов $\varphi_i(\mathbf{A}, \mathbf{B})$ и искомого результата $\varphi(\mathbf{A}, \mathbf{B})$. Конструктивно блоки ЭБП$_i$, $i = \overline{1, n}$, могут быть объединены в единое целое: в оперативную память либо в векторные регистры.

Каноническая структура



Функционирование

$A_1, B_1 \to \varphi_1(\mathbf{A}_1, \mathbf{B}_1)$				
$A_2, B_2 \to \varphi_1(\mathbf{A}_2, \mathbf{B}_2)$	...			
	...			
$A_i, B_i \to \varphi_1(\mathbf{A}_i, \mathbf{B}_i)$	...	$\varphi_{i-1}(\mathbf{A}_1, \mathbf{B}_1) \to \varphi_i(\mathbf{A}_1, \mathbf{B}_1)$		
$A_{i+1}, B_{i+1} \to \varphi_1(\mathbf{A}_{i+1}, \mathbf{B}_{i+1})$	...	$\varphi_{i-1}(\mathbf{A}_2, \mathbf{B}_2) \to \varphi_i(\mathbf{A}_2, \mathbf{B}_2)$	...	
	...		...	
$A_n, B_n \to \varphi_1(\mathbf{A}_n, \mathbf{B}_n)$	...	$\varphi_{i-1}(\mathbf{A}_{n-i+1}, \mathbf{B}_{n-i+1}) \to \to \varphi_i(\mathbf{A}_{n-i+1}, \mathbf{B}_{n-i+1})$	...	$\varphi_{n-1}(\mathbf{A}_1, \mathbf{B}_1) \to \varphi(\mathbf{A}_1, \mathbf{B}_1)$
$A_{n+1}, B_{n+1} \to \varphi_1(A_{n+1}, B_{n+1})$	...	$\varphi_{i-1}(A_{n-i+2}, B_{n-i+2}) \to \to \varphi_i(A_{n-i+2}, B_{n-i+2})$	...	$\varphi_{n-1}(\mathbf{A}_2, \mathbf{B}_2) \to \varphi(\mathbf{A}_2, \mathbf{B}_2)$
	...		...	

Рис. 4.1. Конвейерный процессор:

ЭБО — элементарный блок обработки информации; ЭБП — элементарный блок памяти; **A, B** — векторы-операнды; $\varphi_i(\mathbf{A}, \mathbf{B})$ — частичное преобразование векторов **A, B**

В простейшем случае элементарные блоки обработки конвейера могут реализовывать отдельные фазы операций (например, арифметических или вычисления элементарных функций), т. е. выполнять микрооперации. Например, при сложении двух вещественных (рациональных) чисел, представленных в форме с плавающей запятой, выполняются следующие микрооперации: сравнение порядков, выравнивание порядков, сложение мантисс, нормализация и т. п. В более общем случае блоки ЭБО$_i$ могут вычислять промежуточные результаты $\varphi_i(\mathbf{A}, \mathbf{B})$ $(i = \overline{1, n})$, являющиеся, например, или суммой, или разностью, или произведением, или частным для компонентов векторов **A, B**. Как правило, преобразование $\varphi(\mathbf{A}, \mathbf{B})$ осуществляет одну из арифметических операций над элементами векторов **A, B**.

Элементы векторов подаются в конвейер в дискретные моменты времени и в соответствии с их расположением в векторах. На каждом временном шаге в ЭБО$_1$ конвейера заносится новая пара элементов-операндов текущих векторов **A** и **B**, а в ЭБО$_i$ $(i = \overline{2, n})$ — информация из ЭБП$_{i-1}$ и в общем случае извне.

Процесс вычисления $\varphi(\mathbf{A}, \mathbf{B})$ для пары элементов векторов **A** и **B** разделен на n этапов. Все блоки конвейера работают параллельно, но каждый из них реализует свой этап вычислений и обрабатывает свои элементы-

операнды в фиксированный момент времени. Очевидно, что время обработки на конвейере конкретных элементов векторов равно суммарному времени их пребывания во всех ЭБО$_i$ ($i = \overline{1, n}$). Выдача результатов из «наполненного» конвейера осуществляется через промежутки времени, равные времени выполнения самого медленного этапа. Таким образом, параллелизм в работе блоков конвейера в принципе позволяет достичь производительности, недоступной ЭВМ, базирующимся на модели вычислителя.

Необходимость использования конвейеризации была осознана разработчиками ЭВМ к концу 1950-х годов. Так, например, в советской ЭВМ М-20 (см. разд. 1.4.3), введенной в эксплуатацию в 1958 г., было реализовано совмещение работы арифметического устройства с выборкой очередной команды. Далее, в машине второго поколения ATLAS, разработанной в 1963 г. в Манчестерском университете США, выполнение команды было разбито на четыре этапа: выборку команды, вычисление адреса операнда, выборку операнда и выполнение операции. Конвейеризация позволила достичь в ЭВМ ATLAS времени выполнения операции, равного 1,6 мкс (в то время как для последовательной ЭВМ оно было бы равно 6 мкс). Машина БЭСМ-6 (см. разд. 1.4.6), разработанная в 1966 г., характеризуется параллелизмом в работе устройств и конвейерной структурой процессора.

4.2. Конвейерные системы типа «память-память»

Фирма CDC (Control Data Corporation основана в 1957 г. Сеймором Креем и Уильямсом Норрисом) начиная с 1973 г. выпустила ряд конвейерных ВС, архитектура которых относится к типу «память-память» (рис. 4.2). В таких ВС элементы-операнды векторов **A** и **B**, необходимые для выполнения векторных команд, выбираются непосредственно из оперативной памяти и результат $\varphi(\mathbf{A}, \mathbf{B})$ записывается в ту же память. В качестве преобразования $\varphi(\mathbf{A}, \mathbf{B})$ может быть результат одной из арифметических операций над элементами векторов **A** и **B**.

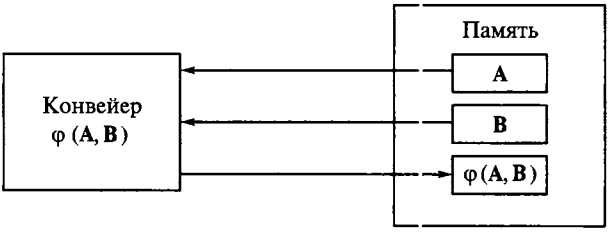


Рис. 4.2. Архитектура ВС типа «память-память»

Ясно, что для хранения векторов **A**, **B**, φ(**A**, **B**) требуются области памяти одинаковой емкости. Если векторы *n*-элементные, то для хранения любого из векторов используется *n* ячеек памяти.

4.2.1. Система STAR-100

Разработка конвейерной ВС STAR-100 (STAR — STring ARray computer — векторный компьютер) осуществлялась фирмой CDC с 1965 по 1973 г. Система была анонсирована в 1970 г., а первая ее поставка была проведена в августе 1973 г. Быстродействие ВС — 10^8 опер./с, стоимость — 15 млн долл.

Система STAR-100 создавалась с учетом языка программирования APL (A Programming Language). Язык APL (или АПЛ) — диалоговый язык программирования, характеризуется развитыми средствами работы с регулярными структурами данных (векторами, матрицами, массивами) и богатым набором базовых операций и компактностью записи.

Вычислительная система STAR-100 состояла из двух подсистем [5, 13]: первая осуществляла переработку данных, вторая — функции операционной системы. Ядром первой подсистемы являлся *процессор*, образуемый из нескольких конвейеров. В типовых конфигурациях системы STAR-100 процессоры состояли из трех конвейеров: K_1, K_2, K_3 (рис. 4.3). Конвейеры были специализированными: два из них (K_1, K_2) служили для выполнения векторных операций, а третий (K_3) — для реализации операций над скалярными операндами, т. е. K_1 и K_2 — конвейеры (Floating-Point Pair Pipelines), каждый из которых служил для выполнения операций с плавающей запятой над парами векторов данных, K_3 — конвейер (string data pipeline), предназначавшийся для обработки обычных операндов, не организованных в векторы. Конвейеры K_1 и K_2 выполняли основной объем вычислений, следовательно, они определяли уровень быстродействия системы STAR-100 в целом.

Конвейеры STAR-100 имели программируемую структуру (т. е. были с изменяемой конфигурацией), следовательно, в них можно было (на одном и том же множестве элементарных блоков обработки) выполнять различные арифметические операции. Однако до начала новой операции конвейер следовало перенастроить (запрограммировать на выполнение очередной операции).

В конвейерах K_1 и K_2 путем введения служебного булевского вектора была обеспечена избирательная обработка компонентов векторов-операндов. Единица в *i*-м разряде булевского вектора означала, что опера-

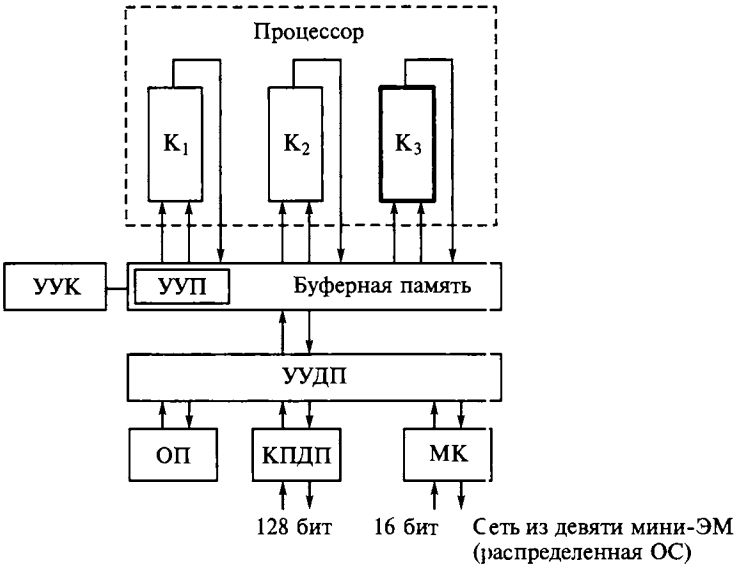


Рис. 4.3. Функциональная структура системы STAR-100:

K_1, K_2, K_3 — конвейеры; УУК — устройство управления командами; УУП — устройство управления потоками; УУДП — устройство управления доступом к памяти; ОП — оперативная память; КПДП — канал прямого доступа в память; МК — мультиплексный канал

ция над i-ми компонентами соответствующей пары векторов производиться не будет.

В каждом конвейере была заложена возможность реализации операции сложения, а в двух из них — K_1 и K_2 — операций умножения и деления. Состав элементарных блоков обработки информации конвейеров был выбран с учетом распределения вероятностей использования микроопераций различных типов.

Каждый конвейер K_i ($i = 1, 2, 3$) мог включать в себя приблизительно 30 блоков обработки информации. Все блоки работали параллельно, но каждый из них оперировал с вполне определенными элементами векторов данных либо со своими скалярными операндами.

Любой конвейер воспринимал 64-разрядный код либо как один 64-разрядный операнд, либо как два 32-разрядных операнда. Время выполнения операции над парой операндов в любом из блоков конвейеров не превышало 40 нс. Следовательно, данные могли поступать в процессор (точнее, только в конвейеры K_1 и K_2) со скоростью 100 млн опер./с.

Система STAR-100 имела набор из 230 команд, из которых 65 команд предназначалось для работы с векторами данных и 130 команд — для работы со скалярами.

Средства управления подсистемой переработки данных были представлены композицией из устройства управления командами (УУК), устройства управления потоками (УУП) и устройства управления доступом к памяти (УУДП). Первое устройство (УУК) имело буфер опережающего просмотра команд (емкостью в четыре 512-разрядных суперслова) со стековым механизмом работы. Второе устройство (УУП) использовалось для управления потоками операндов и команд между УУДП, конвейерами и УУК.

Оперативная память предназначалась для хранения программ и данных. Она была реализована на магнитных сердечниках и имела емкость 512–1024 К 64-разрядных слов, т. е. до 8 Мбайт. Память могла включать в себя до 32 модулей и относилась к классу памятей с перемежающимися адресами. Время цикла памяти было равно 1,28 мкс, однако допускались одновременные обращения к составляющим модулям.

Имелись четыре виртуальных канала обращения к памяти, которые реализовывались устройством управления доступом к памяти. Два канала использовались для чтения операндов (для каждого из конвейеров K_1 и K_2, работавших параллельно, из памяти выбиралось по два 64-разрядных операнда); один — для записи результатов (64-разрядный результат от каждого из конвейеров K_1 и K_2); один — для обмена информацией с устройствами ввода-вывода (либо с одним быстродействующим устройством с полосой пропускания 128 бит, либо с восемью медленными устройствами в режиме разделения времени).

Буферная память — введена вследствие того, что быстродействие оперативной памяти было существенно ниже быстродействия процессора. Буферная память представляла собой совокупность регистров с временем цикла 40 нс. Назначение канала прямого доступа в память (КПДП) и мультиплексного канала (МК) следует из их названий и структуры связей между устройствами STAR-100 (см. рис. 4.3).

Операционная система (ОС) STAR-100 относилась к классу распределенных. Ее функции, включая управление внешними запоминающими устройствами и устройствами ввода-вывода информации, реализовывались специальной вычислительной сетью из девяти мини-машин. Система программирования STAR-100 включала компиляторы с языков APL-STAR, COBOL и FORTRAN.

Первый образец системы STAR-100 был установлен в Ливерморской радиационной лаборатории им. Лоуренса (Lawrence Livermore Laboratory). Были осуществлены поставки системы в правительственные организации и в армию США. Система STAR-100 использовалась для управления запуском антиракет в системе противоракетной обороны США; она широко применялась при решении сложных проблем науки, техники и экономики.

Следует заметить, что вычислительная система STAR-100 допускала модификации: в ней можно было изменять число конвейеров, число и состав внешних устройств, емкость оперативной и внешней памяти и т. д. Создавались и усеченные варианты STAR-IB, а также система SUPERSTAR (или CDC 8600), которая в 1,5–4 раза превосходила по производительности STAR-100 и была более компактной (реализована на более совершенных интегральных схемах).

4.2.2. Семейство систем CYBER

Эволюция архитектуры STAR-100 привела к созданию семейства конвейерных систем CYBER-203 (1979) и CYBER-205 (1981).

Остановимся на архитектурных особенностях систем семейства CYBER. Архитектура этих ВС не изменялась в процессе развития, т. е. была типа «память-память». Производительность системы CYBER-203 (или STAR-100A, как она первоначально называлась) также оставалась 100 млн опер./с. Эту систему можно было рассматривать как модернизированный вариант STAR-100, она была конвейерной, имела ту же систему команд и полностью совместимое программное обеспечение. Однако в отличие от STAR-100 система CYBER-203 содержала обычный скалярный процессор (вместо конвейера $К_3$), который обеспечивал шестикратное увеличение быстродействия при скалярной обработке информации. Емкость оперативной памяти CYBER-203 была увеличена до 16 Мбайт, скорость выборки из памяти — до 100 млрд бод (разрядность слов — 64). Элементную базу системы CYBER-203 составляли БИС.

Система CYBER-205 обладала более совершенной архитектурой в сравнении с CYBER-203. Так, в ней допускалось варьирование числа конвейеров (с изменяемой конфигурацией) от одного до четырех. Пиковая производительность ВС CYBER-205 достигала 200 млн опер./с, емкость оперативной памяти — 32 Мбайт.

Однако все конвейеры CYBER-205 могли работать только в унисон, т. е. все они могли выполнять одновременно только одну и ту же векторную операцию (а не несколько различных). Следовательно, архитектура CYBER-205 в целом представляла собой архитектуру SIMD. В составе аппаратурно реализованных векторных операций CYBER-205 имелись также триады

$$\mathbf{A} + \alpha\mathbf{B},$$

где **A** и **B** — векторы; α — скаляр; $\alpha\mathbf{B}$ — вектор, получаемый из **A** путем умножения его компонентов на число α.

Система CYBER-205 могла выполнять триады почти с такой же скоростью, как отыскание суммы или произведения векторов.

4.3. Конвейерные системы типа «регистр-регистр»

Наряду с фирмой CDC разработкой и производством конвейерных ВС занималась фирма Cray Research Inc., которая была основана в 1972 г. главным конструктором систем CDC 6600 и 7600 Сеймором Р. Креем (Seymour R. Cray, 1925–1996). Однако конвейерные ВС фирмы Cray Research существенно отличались по архитектуре от систем STAR-100, CYBER-203 и CYBER-205.

Архитектура первых систем Cray относилась к типу «регистр-регистр» (рис. 4.4). Архитектура такого типа предопределяет в составе ВС векторные регистры, каждый из которых способен хранить вектор-операнд. Кроме того, при реализации векторных команд векторы-операнды извлекаются покомпонентно из векторных регистров, а вектор-результаты запоминаются также в одном из векторных регистров. До начала реализации векторной команды вектор-операнды должны быть загружены в векторные регистры из оперативной памяти. Предусматривается возможность переноса вектор-результатов из векторных регистров в память.

Векторные регистры играют почти такую же роль, как сверхоперативная память — КЭШ-память (Cache Memory) в обычных ЭВМ. Для повышения эффективности эксплуатации конвейерных ВС этого типа требуется как можно более интенсивно использовать операнды, пока они находятся в векторных регистрах.

В рамках архитектурной концепции конвейерных ВС типа «регистр-регистр» фирма Cray Research выпустила ряд совместимых моделей: Cray-1, Cray X-MP, Cray Y-MP, Cray C90, Cray T90. В этом ряду только первая модель, т. е. Cray-1, была однопроцессорной, а остальные члены ряда — мультипроцессорные ВС. Процессор в любой из этих систем ориентирован на реализацию векторных операций. Он по сути являлся мультиконвейером, т. е. представлялся программируемой композицией из специализированных конвейеров.

Мультипроцессорные модели Cray X-MP, Cray Y-MP, Cray C90 и Cray T90 называют также параллельно-векторными ВС или PVP-системами (PVP — Parallel Vector Processors).

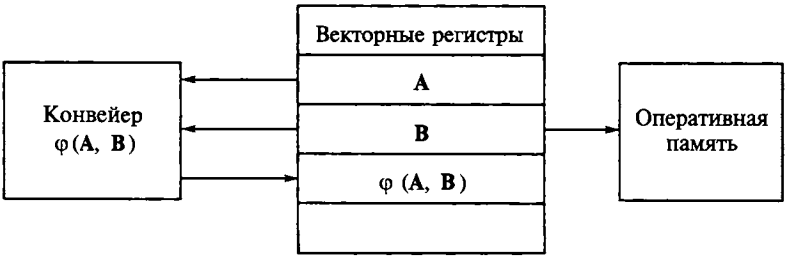


Рис. 4.4. Архитектура ВС типа «регистр-регистр»

Фирма Cray Research выпускала также и модификации отмеченных выше моделей и создала еще один ряд мультипроцессорных ВС: Cray-2, Cray-3, Cray-4, которые были архитектурно несовместимы с рядом, эволюционировавшим от Cray-1.

Следует особо отметить, что в процессе разработок мультипроцессорных ВС фирма Cray Research сильно отошла от изначального архитектурного канона (см. рис. 4.4). Каждая очередная разработка была заметным развитием архитектуры предшествующей системы, и в конце концов фирма Cray Research с диалектической неизбежностью встала на платформу распределенных ВС. Подтверждением сказанному служит семейство ВС с массовым параллелизмом или MPP-систем: Cray T3D, Cray T3E, Cray T3E-900, Cray T3E-1200, Cray T3E-1350 (MPP-systems — Massively Parallel Processing Systems, массово-параллельные ВС).

С момента своего создания фирма Cray Research претерпела ряд преобразований. Так, в 1989 г. из данной фирмы под проект Cray-3 выделилась Cray Computer Corp., которую возглавил С. Крей. В феврале 1996 г. произошло поглощение Cray Research Inc. фирмой SGI (Silicon Graphics Inc.). В августе 1999 г. SGI создает подразделение Cray Research исключительно для бизнеса в области суперкомпьютеров. В марте 2000 г. активы этого подразделения были проданы Tera Computer Company, и с этого времени начинает функционировать Cray Incorporation. Ныне Cray Inc. — мировой лидер по суперкомпьютерам.

Далее будут рассмотрены архитектурные возможности Cray-систем, относящихся к видам PVP и MPP.

4.3.1. Система Cray-1

Создание ВС Cray-1 было завершено фирмой Cray Research в 1976 г. Быстродействие системы составляло 160 MFLOPS ($16 \cdot 10^7$ опер./с при выполнении операций с плавающей запятой над векторами данных и 37 млн опер./с — над скалярами), емкость оперативной памяти 8...64 Мбайт, длина слова данных — 64 двоичных разряда, цена — 5...9 млн долл. Система Cray-1 успешно конкурировала на рынке супер ЭВМ вплоть до 1982 г.

Вычислительная система Cray-1 предназначалась для векторной и скалярной обработки данных. Эта система состояла из четырех функциональных подсистем: управления программой, конвейеров, регистров, памяти и ввода-вывода (рис. 4.5).

Подсистема управления программой ВС Cray-1 наряду со стандартными устройствами и узлами (счетчик команд, средства организации ветвлений, устройство прерывания и т. п.) имела и буферную память для команд.

Рис. 4.5. Функциональная структура системы Cray-1

Подсистема конвейеров — это и есть процессор ВС Cray-1. Он состоял из 12 функционально ориентированных конвейеров, которые подразделялись на четыре группы: для операций над адресами, скалярных операций, операций над числами с плавающей запятой и векторных операций. Конвейеры состояли из сегментов — ЭБО. Каждый ЭБО был ориентирован на выполнение своей микрооперации, длительность цикла любого ЭБО составляла 12,5 нс ($12{,}5 \cdot 10^{-9}$ с). Каждый конвейер мог выдавать результаты на каждом цикле работы, следовательно, цикл системы — 12,5 нс.

Группа конвейеров для операций над адресами состояла из конвейеров для сложения и умножения целых чисел, причем первый из них имел в своем составе два ЭБО, а второй — шесть ЭБО. Группа конвейеров для скалярных операций была представлена счетчиком (3 ЭБО) и тремя конвейерами: для сложения целых чисел (3 ЭБО), логических операций (1 ЭБО) и сдвига (3 ЭБО). Группа конвейеров для операций с плавающей запятой состояла из конвейеров для сложения, умножения и вычисления обратной величины (6, 7 и 14 ЭБО соответственно). В группу конвейеров для векторных операций входили конвейеры для сложения целых чисел (3 ЭБО), логических операций (2 ЭБО) и сдвига (4 ЭБО). Все конвейеры могли работать одновременно (параллельно). Деление в системе Cray-1 осуществлялось с помощью конвейера вычисления обратной величины.

Подсистема регистров ВС Cray-1 включала следующие основные регистры с программным доступом:

1) 8 24-разрядных адресных *A*-регистров;

2) 64 24-разрядных промежуточных адресных *B*-регистров;

3) 8 64-разрядных скалярных *S*-регистров;

4) 64 64-разрядных промежуточных скалярных *T*-регистров;

5) 8 векторных *V*-регистров.

Каждый из *V*-регистров был способен хранить вектор из 64-х 64-разрядных компонентов.

Кроме этих пяти групп регистров имелись также: программно доступный регистр, устанавливавший необходимую длину векторов; 64-разрядный регистр маскирования векторов, разряды которого соответствовали элементам векторных регистров; 64-разрядный регистр часов реального времени.

Подсистема регистров ВС Cray-1 — это сверхоперативная память (с циклом 6 нс), обладающая емкостью 4888 байт.

Конвейеры имели доступ (для получения операндов и для записи результатов) только к *A*-, *S*- и *V*-регистрам; *B*- и *T*-регистры позволяли повысить скорость скалярной обработки. Скорость передачи для *B*-, *T*- и *V*-регистров равнялась одному слову за цикл системы, а для *A*- и *S*-регистров — одному слову за два цикла. Следовательно, совокупность *B*-, *T*- и *V*-регистров (как память для кратковременного хранения данных) благоприятно повлияла на производительность системы.

Подсистема памяти и ввода-вывода ВС Cray-1 имела в своем составе оригинально организованную оперативную память. Последняя обладала емкостью 1 М слов и состояла из 16 независимых банков емкостью 64 К слов каждый. В свою очередь, любой банк включал в себя 72 модуля памяти, причем каждый из них предназначался для хранения одного разряда всех слов данного банка. Из 72 разрядов слова 64 служили в качестве рабочего слова (команды или операнда), а остальные 8 разрядов предназначались для исправления одиночных и обнаружения двойных ошибок в рабочем слове. Время цикла одного банка было равно четырем циклам системы, т. е. составляло 50 нс. Однако наличие 16 независимых банков позволило организовать 16-кратное чередование адресов.

Ввод-вывод информации в ВС Cray-1 осуществлялся через 12 входных и 12 выходных каналов, которые обеспечивали суммарную скорость 500 тыс. 64-разрядных слов в секунду.

Система команд Cray-1 содержала 128 основных команд. Команды могли иметь одну или две 16-разрядных части. При формате команды (например, арифметической или логической) в виде одной части 7 разрядов отводились под код операции, по 3 разряда — для адресов двух регистров, в

которых хранились два операнда, 3 разряда — для адреса регистра, в который заносился результат. Разряды команды, которые использовались для двух адресов операндов, могли применяться как единое поле команды для адресации *B*- или *T*-регистров. Это же поле при формате команд в виде двух частей в совокупности с 16 разрядами второй части использовалось и для адресации основной оперативной памяти.

В целях ускорения выполнения команд была предусмотрена их буферизация при помощи четырех специальных буферов, в каждом из которых могло быть размещено до 64 16-разрядных частей команды (см. рис. 4.5). Скорость передачи информации для буферов команд составляла 16 команд за цикл.

Арифметические операции с фиксированной запятой выполнялись над числами форматов 24 или 64 разряда. Для выполнения арифметических операций с плавающей запятой под мантиссу отводились 49 разрядов, а под порядок — 15 разрядов, что обеспечивало представление чисел в диапазоне от 10^{-2500} до 10^{+2500}

Операционная система COS (Cray Operating System) обеспечивала режим пакетной обработки (до 63 задач). Оптимизирующий компилятор CFC (Cray Fortran Compiler) для языка высокого уровня ANSI 66 FORTRAN IV учитывал особенности векторной обработки в системе Cray-1. В программное обеспечение входили также макроассемблер CAL (Cray Assembler Language), библиотека стандартных программ, загрузчик и другие сервисные средства.

Особенность архитектуры ВС Cray-1 состояла в том, что она обладала способностью адаптации к структуре решаемой задачи. Последнее достигалось настройкой (программным формированием) цепочек (макроконвейеров) из произвольного числа конвейеров и с произвольной их последовательностью. В таких макроконвейерах передача информации между соседними конвейерами осуществлялась непосредственно (через регистры), т. е. без пересылок в оперативную память. Следует также подчеркнуть, что в Cray-1 допускалась параллельная работа как конвейеров, так и элементарных блоков обработки в пределах любого конвейера. Система была способна выполнять как скалярные, так и векторные операции, причем одновременно могло выполняться несколько как скалярных, так и векторных операций.

Конструкция ВС Cray-1 уникальна. Система была выполнена в виде 12 клинообразных стоек, имеющих высоту 1,96 м и расположенных по дуге в 270° внутри окружности с диаметром 2,63 м (причем на высоте 0,48 м диаметр окружности уменьшался до 1,44 м). Такой «кольцевой» принцип компоновки конструкции ВС Cray-1 позволил достичь незначительных длин электрических соединений между устройствами и узлами (не более 4 фу-

тов*), следовательно, уменьшить задержки при прохождении сигналов. В системе было применено фреоновое охлаждение.

Комплекс архитектурных, структурных и конструктивных решений позволил в условиях микроэлектронной базы начала 1970-х годов достичь в ВС Cray-1 длительности цикла в 12,5 нс и высокого уровня надежности.

Первая поставка системы Cray-1 была осуществлена в Лос-Аламосскую национальную лабораторию (Los Alamos National Laboratory) в 1976 г. Поставленная конфигурация ВС имела память емкостью 1 млн слов, цена поставки — 8,8 млн долл. Фирмой Cray Research было произведено всего 16 систем Cray-1.

4.3.2. Параллельно-векторные системы Cray

Системы Cray вида PVP являются коллективами (или кластерами), образованными из конвейерных процессоров. Ниже будут рассмотрены два семейства PVP-систем, первое из которых включает следующие модели: Cray X-MP, Cray Y-MP, Cray C90 и Cray T90, а второе — Cray-2, Cray-3 и Cray-4. Эти PVP-системы перекрывают диапазон производительности от сотен до десятков тысяч MFLOPS, обладают емкостью оперативной памяти от десятков миллионов до десятков миллиардов байт, способны работать с 64-разрядными операндами. Параллельно-векторные системы Cray являются суперкомпьютерами конца XX в.

Вычислительные системы Cray X-MP и Cray Y-MP. *Система Cray X-MP* — первая суперВС — кластер из конвейерных процессоров. Архитектура ВС Cray X-MP относится к классу MIMD, однако эту систему следует воспринимать как «старшую» модель, совместимую с Cray-1. Образец двухпроцессорной ВС был создан в 1982 г., а четырехпроцессорной — в 1984 г. В системе Cray X-MP может быть два или четыре процессора, максимальное быстродействие составляет 940 MFLOPS, быстродействие одного процессора — 235 MFLOPS, емкость оперативной МОП-памяти произвольной выборки — 64...128 Мбайт, цена четырехпроцессорной конфигурации ВС — 14,6 млн долл.

Компания Cray Research производила также мини-суперкомпьютер (Minisupercomputer) Cray XMS, совместимый с Cray X-MP с воздушным охлаждением.

Эффективной областью применения ВС Cray X-MP являлось, например, моделирование авиакосмических объектов. Задачи этой области допускают расщепления вычислительного процесса на два и четыре самостоятельных процесса. Например, на двух процессорах можно рассчитывать

* 1 фут = 0,3048 м.

воздушные потоки для каждого из двух крыльев самолета или при составлении прогноза погоды можно обрабатывать данные для каждого из двух полушарий Земли.

Вычислительная система Cray Y-MP создана в 1988 г., ее архитектурные характеристики заметно превосходят Cray X-MP. Количество процессоров в системе Cray Y-MP составляло от одного до восьми, максимальное быстродействие ВС — 2,65 GFLOPS, быстродействие одного процессора — 333 MFLOPS, стандартная и максимальная емкости оперативной памяти — 256 Мбайт и 32 Гбайт.

Максимальная производительность 1 GFLOPS (т. е. 10^9 опер./с над 64-разрядными данными с плавающей запятой) была достигнута в ВС Cray Y-MP в 1989 г.

С 1994 г. фирма Cray Research начала производить системы Cray J90, совместимые с Cray Y-MP, но обладавшие бóльшими возможностями по «масштабированию» (от 4 до 32 процессоров), более компактные и дешевые (с воздушным охлаждением). Система Cray J90 была наиболее популярной в мире (было выпущено свыше 400 шт.).

Вычислительные системы Cray C90 и Cray T90.

Система Cray C90 («Cray for the 90s» — ВС для 90-х годов XX в.) была построена в 1991 г., ее максимальное быстродействие достигало 16 GFLOPS. Для формирования данной системы были впервые применены процессоры с быстродействием 1 GFLOPS. Допустимое количество процессоров в конфигурациях ВС — 2, 4, 8 и 16; емкость оперативной памяти — 512 Мбайт...8 Гбайт.

Архитектура ВС Cray C90 в целом относится к классу MIMD; это мультипроцессорная система с общей памятью. В ее состав входят: подсистема процессоров, подсистема межпроцессорных взаимодействий, общая память и подсистема ввода-вывода информации.

Функциональная структура процессора ВС Cray C90 близка к структуре Cray-1 (композиция секции управления, конвейеров, регистров и сети связей). Конвейеры и регистры предназначаются для обработки и хранения данных трех типов: адресов (*A*- и *B*-регистры), скалярных операндов (*S*- и *T*-регистры) и векторных операндов (*V*-регистры). *Конвейеры* подразделяются на четыре группы: адресные, скалярные, векторные и для операций с плавающей запятой. Последняя группа конвейеров предназначается для выполнения как скалярных, так и векторных команд. Общее число конвейеров составляет 14–16. *Регистры* трех основных наборов (*A*, *S*, *V*) имеют связи как с конвейерами, так и с оперативной памятью. Регистры *B* и *T* играют роль буферных для основных *A*- и *S*-регистров (см. разд. 4.3.1).

Восемь адресных 32-разрядных *A*-регистров предназначаются для хранения и вычисления адресов, индексации, указания величины сдвигов и числа

итераций циклов и т. д. В Cray C90 64 32-разрядных *B*-регистров. Восемь скалярных 64-разрядных *S*-регистров применяются для хранения данных и результатов операций скалярной арифметики; их также можно использовать для хранения элементов векторов данных при векторных вычислениях. В системе 64 64-разрядных *T*-регистров. Восемь векторных регистров (*V*-регистры) рассчитаны для хранения 128-компонентных векторов данных, причем каждый компонент представляет собой 64-разрядное слово. Эти регистры используются только для выполнения векторных команд. Наряду с названными имеются также регистр длины вектора (8 разрядов) и регистр маски вектора (128 разрядов). При функционировании процессор способен в каждом такте (каждые 4,1 нс) выдавать результаты двух операций. Если выполняется операция «зацепления» сложения и умножения, то процессор фактически за такт реализует четыре арифметических операции. Следовательно, пиковая производительность процессора достигает почти 1 GFLOPS (10^9 опер./с).

Подсистема межпроцессорных взаимодействий предназначена для организации и реализации передач данных и управляющей информации между процессорами. Это, по сути, композиция общедоступных регистров, в которой выделены одинаковые кластеры (группы). Каждый кластер содержит 8 32-разрядных общедоступных адресных регистров (*SB*), 8 64-разрядных общедоступных скалярных регистров (*ST*) и 32 1-разрядных регистра для однобитовых семафоров.

Оперативная память Cray C90 является общедоступной для всех процессоров и подсистемы ввода-вывода информации. Каждый процессор имеет доступ к памяти через четыре порта, пропускная способность любого порта составляет два слова за один такт (за 4,1 нс). При этом один из портов всегда связан с подсистемой ввода-вывода и, по крайней мере, еще один из портов всегда выделен под операцию записи. Ячейки памяти способны хранить 80-разрядные слова (64 разряда — для хранения операнда и 16 разрядов — для коррекции ошибок).

В максимальной конфигурации память разделена на восемь секций, каждая секция — на восемь подсекций и, наконец, каждая подсекция — на 16 банков. Ячейкам памяти присвоены адреса таким образом, что имеет место их чередование по секциям, подсекциям и банкам. При этом возможны конфликты при одновременном обращении к какой-либо части памяти из разных портов. Так, при одновременном обращении к одной и той же секции возникает задержка на 1 такт, и при обращениях к одной и той же подсекции в пределах одной секции задержка варьируется от 1 до 6 тактов. При выборке последовательно расположенных данных или при выборке с любым нечетным шагом конфликтов не возникает.

Подсистема ввода-вывода информации ВС Cray C90 представлена тремя типами каналов, которые различаются по скорости передачи:

- низкоскоростные каналы (Low-speed channels) — 6 Мбайт/с;
- высокоскоростные каналы (High-speed channels) — 200 Мбайт/с;
- сверхскоростные каналы (Vety high-speed channels) — 1800 Мбайт/с.

Архитектурные особенности системы Cray C90 связаны прежде всего с тем, что эта ВС по своей производительности относится к диапазону GigaFLOPS (2...16 GFLOPS) и в ней достаточно полно внедрены принципы параллелизма и конвейеризации.

В системе реализованы следующие режимы *многопроцессорной обработки*: выполнения несколько независимых программ на различных процессорах (Multiprogramming) и одной (параллельной) программы на нескольких процессорах (Multitasking).

Вычислительная система Cray C90 может выполнять *векторную обработку* информации. При этом скорость обработки возрастает за счет того, что некоторая (арифметическая) операция * над двумя векторами **A** и **B** данных (т. е. над двумя одномерными массивами данных) выполняется одной командой. В скалярном режиме для выполнения этой же операции * над массивами **A** и **B** потребуется выполнить n-краткий цикл команд: выбрать элемент $A_i \in \mathbf{A}$ $(i = \overline{1, n})$, выбрать элемент $B_i \in \mathbf{B}$, выполнить операцию «*», записать результат $C_i = A_i * B_i$, увеличить параметр i цикла, проверить условие $i < n$ цикла. Скорость выполнения операций в векторном режиме приблизительно в 10 раз выше скорости скалярной обработки.

Архитектура ВС Cray C90 допускает реализацию режима *зацепления конвейеров* (или векторных операций). Это означает, что результаты, вычисляемые одним конвейером, могут передаваться на вход другого. Точнее, в данной ВС допустимо использовать регистр, в который занесен результат векторной операции, в качестве входного регистра для конвейера, в котором будет выполняться последующая векторная операция. Вообще говоря, глубина зацепления может быть достаточно большой, например, может быть осуществлено зацепление операций в такой последовательности: чтение, сложение, умножение и запись векторов данных.

Векторные операции, реализуемые на различных конвейерах и использующие различные регистры, могут выполняться параллельно. Допустимо также параллельное выполнение скалярных операций на независимых конвейерах.

Для архитектуры ВС Cray C90 характерна *конвейеризация* всех *основных команд, выполняемых процессором, обращения к памяти, обработки команд и выполнения операций.*

Вычислительная система Cray T90 была создана в 1995 г., максимальное быстродействие достигало 64 GFLOPS. Промышленные модели данной ВС: Cray T94, Cray T916 и Cray T932, состояли соответственно из 4, 16 и 32 процессоров. Емкость оперативной памяти ВС Cray T932 со-

ставляла 512 Мбайт...8 Гбайт, скорость обмена информацией с памятью — 800 Мбод.

Время процессорного цикла ВС Cray T90 составляло 2,2 нс, однако за счет конвейеризации процессор имел быстродействие 2 GFLOPS (вычислительные средства с такой производительностью даже в начале 90-х годов XX в. относились к суперкомпьютерам).

Система Cray T90 по своей архитектуре относилась к классу MIMD. Кроме того, в ней была предусмотрена возможность построения макросистем, как объединений нескольких ВС Cray T90.

Вычислительные системы Cray-2 и Cray-3. Параллельно-векторная *вычислительная система Cray-2* была разработана под руководством С. Крея и построена в 1985 г. в корпорации Cray Research. Эта ВС имела четырехпроцессорную конфигурацию и самые лучшие технические характеристики для 80-х годов XX столетия. Так, пиковое быстродействие ВС составляло 1,95 GFLOPS (т. е. $1{,}95 \cdot 10^9$ опер./с над операндами с плавающей запятой, составлявшими векторы) и 250 MIPS (т. е. $2{,}5 \cdot 10^8$ опер./с над скалярными величинами с фиксированной запятой). Емкость оперативной памяти ВС Cray-2 достигала 512 Мбайт...32 Гбайт. При этом цена ВС Cray-2 превышала в 2 раза цену ВС Cray-1.

По архитектуре ВС Cray-2 резко отличалась от ВС Cray-1, она имела новый набор команд и новую операционную систему. В ней каждому процессору помимо векторных регистров была придана локальная оперативная память емкостью не менее 16 К 64-разрядных слов.

Вычислительная система Cray-2 была построена на быстродействующей элементной базе, что позволило достичь длительности цикла в 4,1 нс (вместо 12,5 нс, как это имело место в Cray-1). Быстродействие одного процессора Cray-2 составляло 488 MFLOPS.

Конструкция ВС Cray-2 оригинальна и достаточно компактна, ее можно было разместить в цилиндре с основанием 1,35 м и высотой 1,15 м. Компактность конструкции позволила применить для охлаждения Cray-2 метод полного погружения в инертную жидкость.

Система Cray-3 разрабатывалась также под руководством С. Крея, но уже в фирме Cray Computer. В Cray-3 входило до 16 процессоров, время цикла процессора составляло 2,11 нс, ожидаемое быстродействие 16-процессорной конфигурации ВС — 7,3 GFLOPS. В 1989 г. была построена конфигурация Cray-3 с быстродействием 5 GFLOPS и емкостью оперативной памяти до 33 Гбайт.

По своей архитектуре ВС Cray-3 была достаточно близка к ВС Cray-2. В качестве элементной базы для системы Cray-3 были использованы арсинид-галиевые интегральные схемы. Эта элементная база была более быстродействующей, но и более дорогой, чем кремниевая.

В фирме Cray Computer Corp. велись работы не только по системе Cray-3, но и по Cray-4, однако эти системы не имели коммерческого успеха.

4.4. Массово-параллельные вычислительные системы Cray

В конце 80-х годов XX в. ряд компаний (Thinking Machines, Kendal Square, NCube, MasPar и Mieko) успешно проводили исследования и разработки новых архитектур суперВС (Massively Parallel Processing Systems), в которых высокая эффективность достигалась за счет применения большого количества элементарных (простых) процессоров. Системы с массовым параллелизмом (MPP-системы) стали альтернативой для векторно-параллельных ВС (PVP-систем). Было разработано семейство массово-параллельных ВС, включающее возможные конфигурации моделей: Cray T3D, Cray T3E, Cray XT3 и Cray XT4. Эти ВС обеспечивают обработку информации с производительностью от десятков GigaFLOPS до сотен TeraFLOPS и предоставляют память емкостью от Гигабайт до сотен Терабайт.

4.4.1. Вычислительная система Cray T3D

Вычислительная система Cray T3D — первая MPP-система корпорации Cray Research, ее разработка была завершена в 1993 г. Это позволило фирме Cray Research Inc. быстро захватить лидерство на рынке MPP-систем. Количество элементарных процессоров в конфигурациях системы Cray T3D достигало 32...2048, а диапазоны производительности и емкости памяти были соответственно равны 5...300 GFLOPS и 512 Мбайт...128 Гбайт. Система в максимальной конфигурации никогда не выпускалась; обычная конфигурация Cray T3D — 64-процессорная, она обеспечивала быстродействие, равное 10 GFLOPS.

Архитектура системы Cray T3D — MIMD, а сама ВС принадлежит к виду распределенных. В системе достаточно полно воплощены принципы модели коллектива вычислителей (см. разд. 3.1.1). Последнее позволило, в частности, достичь в ВС Cray T3D высокой надежности и живучести, а также масштабируемости (варьируемости числа процессоров в пределах от 32 до 2048 с шагом 32). Следовательно, архитектура ВС Cray T3D приспособлена к формированию конфигураций с заданной производительностью и/или стоимостью.

Система Cray T3D работает под управлением хост-системы (Host System — управляющая ВС). Одной из функций хост-системы является производительная подготовка программ (включающая компиляцию) и ввод-вывод данных для Cray T3D. В качестве хост-системы могут быть использо-

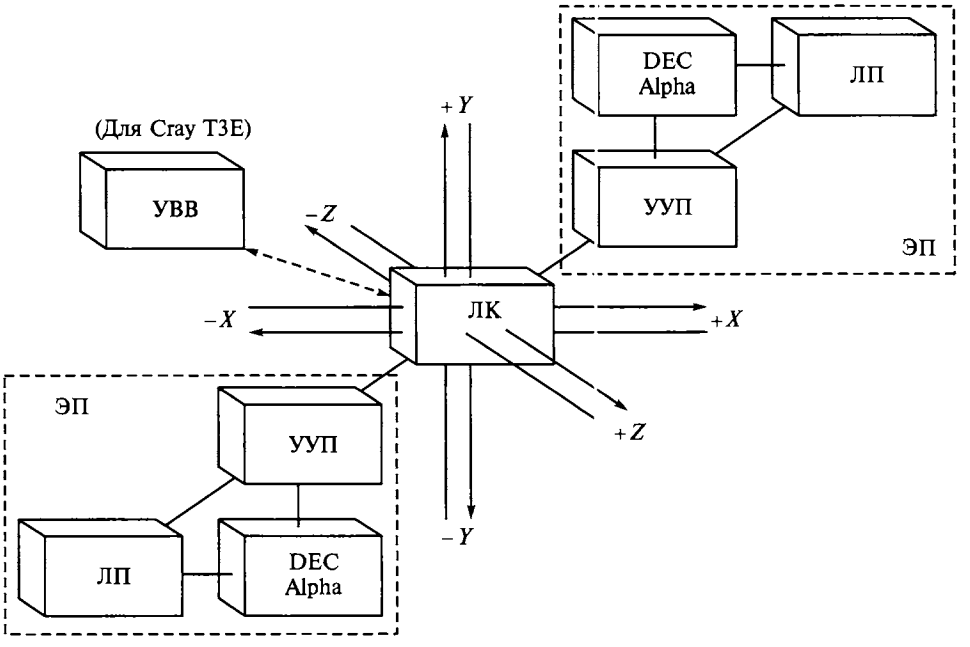


Рис. 4.6. Вычислительный узел системы Cray T3D (Cray T3E):

УВВ — устройство ввода-вывода; ЛП — локальная память; УУП — устройство управления памятью; ЭП — элементарный процессор; ЛК — локальный коммутатор

ваны, в частности, конфигурации ВС Cray Y-MP и Cray C90. Между хост-ВС и системой Cray T3D предусмотрен высокоскоростной канал связи (200 Мбайт/с).

Вычислительная система Cray T3D представляет собой композицию множества вычислительных узлов, коммуникационной сети (или сети межузловых связей), каналов ввода-вывода информации и средств синхронизации.

Вычислительный узел Cray T3D. Все вычислительные узлы (ВУ), составляющие ВС Cray T3D, однородные. Каждый узел (Processing Element Node) системы включает в себя (рис. 4.6) два одинаковых ЭП и ЛК.

Элементарный процессор (Processing Element — процессорный элемент) представляется композицией из микропроцессора, локальной памяти (ЛП) и устройства управления памятью (УУП). Микропроцессор — это DEC 21064 Alpha chip (или просто DEC Alpha), т. е. RISC-процессор типа Alpha фирмы DEC (Reduced Instruction Set Computer — компьютер с сокращенным набором команд). Микропроцессор имеет кэш-память для команд и кэш-память для данных. Набор команд предусматривает и логические, и арифметические операции целочисленной и вещественной арифметики. Характеристики архитектуры DEC Alpha:

Разрядность	64
Тактовая частота, МГц	150
Производительность (3 инструкции за цикл)	150 MFLOPS, 300 MIPS
Емкость кэш-памяти для команд и данных, К байт	8 и 8
Технология	КМОП (комплементарная «металл–окисел–проводник»)
Технологические нормы, мкм	0,75

Локальная память ЭП представляет собой DRAM-память емкостью 16...64 Мбайт (Dynamic Random Access Memory — динамическая память с произвольной выборкой). Каналы связи микропроцессора с локальной памятью ЭП характеризуются малой задержкой и высокой пропускной способностью. Устройство управления памятью ЭП (Support Circuitry) осуществляет поддержку обмена данными между элементарными процессорами.

Локальный коммутатор обеспечивает непосредственную связь ВУ с соседними узлами и представляет собой шестиполюсник. В состав ЛК входят: сетевой маршрутизатор, сетевой интерфейс и контроллер для пересылки блоков данных.

Сетевой маршрутизатор (Network Router) ВУ — основной элемент управления коммуникационной сетью Cray T3D. Он способен работать с тремя парами двунаправленных межузловых связей (Communication links), что позволяет создавать трехмерные структуры ВС. Маршрутизатор каждого ВУ определяет путь перемещения каждого пакета данных и может осуществлять параллельный транзит данных по всем трем межузловым связям.

Сетевой интерфейс (Network Interface) вычислительного узла специальным образом кодирует информацию перед ее пересылкой по коммуникационной сети другому ВУ или в канал ввода-вывода. Сетевой интерфейс служит также для приема данных от других ВУ или из канала ввода-вывода и распределяет их между элементарными процессорами данного ВУ.

Контроллер для пересылки блоков данных (Block Transfer Engine) осуществляет асинхронное перераспределение данных в пределах всей распределенной памяти ВС Cray T3D, т. е. перераспределение информации, находящейся в локальной памяти разных ЭП системы, без прерывания работы самих элементарных процессоров.

Следует подчеркнуть, что в системе Cray T3D память распределенная, точнее, она физически распределенная, но логически общая. Каждый ЭП имеет непосредственный доступ к своей локальной памяти, но он может обратиться и к локальной памяти другого ЭП, не прерывая его работы. Такие возможности поддерживаются аппаратурой ВУ и коммуникационной сетью Cray T3D. Очевидно, что обращение элементарного процессора к памяти другого ЭП осуществляется медленнее, чем обращение к собственной ло-

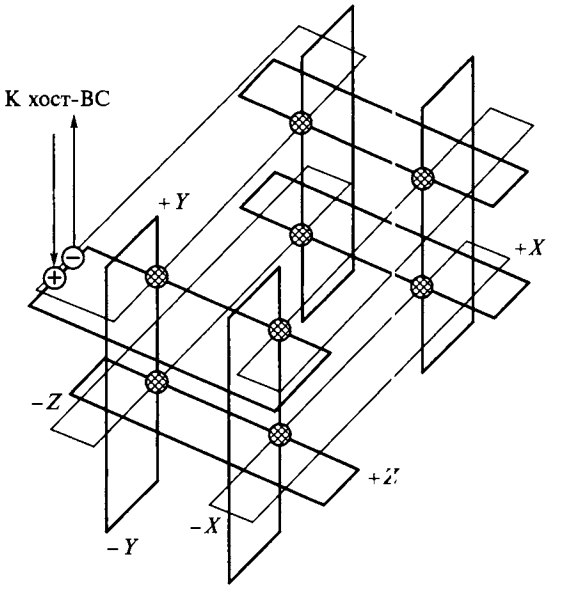


Рис. 4.7. Фрагмент структуры коммуникационной сети Cray T3D (Cray T3E):
⊗ — вычислительный узел; ⊕ — узел ввода; ⊖ — узел вывода

кальной памяти. Величина задержки при таком обращении определяется количеством линков, соединяющих взаимодействующие процессоры.

Коммуникационная сеть Cray T3D. Коммуникационная сеть (Interconnect Network) системы Cray T3D предназначается для реализации обменов информацией между ВУ, а также между ВУ и каналами ввода-вывода. Она образуется из связей (Communication Links) и сетевых маршрутизаторов (Network Routers) как ВУ, так и каналов ввода-вывода информации (рис. 4.7).

Ориентация ВС на решение трехмерных сложных задач предопределила ее структуру, именно трехмерную структуру коммуникационной сети. В системе Cray T3D каждый ВУ связан с соседними по трем направлениям *X*, *Y* и *Z*, причем по каждому направлению вершины образуют замкнутое кольцо. Говоря точнее, структура коммуникационной сети Cray T3D является циркулянтным графом с тремя образующими (см. разд. 3.1.2 и 7.2.1) или *трехмерным (3D) тором* (см. рис. 4.7). Каждая связь между двумя соседними узлами представляется двумя однонаправленными каналами передачи данных, что допускает одновременный обмен информацией в противоположных направлениях.

Быстродействие коммуникационной сети Cray T3D по каждому из двух направлений передачи информации составляет 140 Мбайт/с.

Двунаправленный трехмерный тор имеет преимущества перед «незамкнутыми» трехмерными топологиями:

- повышенная живучесть структуры — возможность выбора маршрутов для обхода поврежденных узлов и связей;
- возможность быстрой связи граничных узлов и небольшая латентность (задержка) при передаче информации между вершинами (диаметр — максимальное расстояние из кратчайших между любыми двумя вершинами — для конфигурации из 128 ЭП он равен 6, а для 2048 ЭП — 12).

Следует подчеркнуть, что в 3D-торе каждый ВУ непосредственно связан с шестью соседними узлами, и он входит в три кольца ВУ, соответствующих направлениям *X*, *Y*, *Z*. Трехмерный тор может быть представлен в виде $N(x \times y \times z)$, где N — общее число узлов, а x, y, z — количество узлов в кольцах по направлениям соответственно *X*, *Y*, *Z*. Например, в 3D-торе 8 ($2 \times 2 \times 2$) имеется 8 вычислительных узлов, причем каждый из них входит в три кольца из двух ВУ (см. рис. 4.7), а в 3D-торе 64 ($4 \times 4 \times 4$) — 64 узла, каждый из которых входит в три кольца из четырех ВУ.

Адресация (нумерация) вычислительных узлов Cray T3D разделена на физическую, логическую и виртуальную. Каждому ВУ присвоен свой *физический адрес*, определяющий его абсолютное расположение в системе; этот адрес используется непосредственно аппаратурой. Вычислительному узлу может быть присвоен также *логический адрес*, определяющий его расположение в логической конфигурации системы, которая уже и будет представлять собой трехмерный тор. Например, 512-процессорная конфигурация системы Cray T3D реально содержит 260 физических ВУ, четыре из которых составляют резерв. Следовательно, логические конфигурации ВС, по сути, являются физическими системами с повышенной надежностью. Виртуальная адресация ВУ введена для того, чтобы пользователю предоставлять дополнительный сервис: он не должен учитывать при программировании физические и логические адреса ВУ, а может вводить свои (виртуальные) адреса ВУ. При этом каждой программе пользователя из трехмерного тора будет выделен вполне определенный прямоугольный параллелепипед, на котором и будет исполняться данная программа (не учитывая средств операционной системы).

Любой из адресов ВУ представляется трехкомпонентным вектором: (x, y, z), который однозначно определяет расположение узла в трех измерениях коммуникационной сети Cray T3D. Физический адрес ВУ — это абсолютный неизменяемый номер узла в сети, а логический и виртуальный адреса ВУ являются относительными адресами (относительно абсолютного).

Относительные адреса получают операцией смещения исходного адреса. Поясним смысл относительной адресации на примере 3D-тора вида 64 ($4 \times 4 \times 4$), вершины в каждом из колец которого пронумерованы 0, 1, 2, 3 (по модулю 4). Смещение в направлении увеличения номера имеет знак «+»,

а в обратном — знак «–». Пусть заданы следующие исходный и относительные адреса:

x	y	z	
2	0	3	— исходный адрес
3	2	0	— относительный адрес
+1	+2	–3	— смещения

Легко заметить, что требуемый относительный адрес в рассматриваемом торе может быть получен и при помощи смещения (–3, – 2, + 1). Приведенный пример демонстрирует простоту механизма преобразования физического (или логического) адреса ВУ в его логический (или виртуальный) адрес.

В системе Cray T3D отображение логических адресов на физические адреса ВУ обеспечивается таблицей маршрутизации, загружаемой в сетевые маршрутизаторы. Гибкость средств отображения адресов и маршрутизации позволяет логически изолировать неисправные ВУ в Cray T3D.

Каналы ввода-вывода Cray T3D. Каналы ввода-вывода (Input/Output Gateways — шлюзы ввода-вывода) предназначены для обмена информацией между Cray T3D и управляющей системой (Host System) или кластером ввода-вывода (Input/Output Cluster).

Канал ввода-вывода Cray T3D представляется композицией из узлов ввода и вывода и низкоскоростного устройства передачи запросов и ответов.

Функциональные структуры узлов ввода и вывода предельно близки к структуре ВУ, в состав каждого из первых двух узлов входят элементарный процессор, локальный коммутатор и схемы ввода или вывода соответственно. Локальный коммутатор любого из узлов ввода или вывода включает в свой состав сетевой маршрутизатор (Network Router), сетевой интерфейс (Network Interface) и контроллер для асинхронной передачи данных (Block Transfer Engine). Однако, в отличие от ВУ, здесь сетевой маршрутизатор рассчитан на работу со связями только по направлениям X и Z.

Узлы ввода и вывода взаимодействуют друг с другом не только через маршрутизаторы, но и через устройство передачи запросов и ответов.

Каналы ввода-вывода включаются в коммуникационную сеть Cray T3D только в «кольца» направлений X и Z (см. рис. 4.7). Хост ВС подсоединяется к каналам через высокоскоростные (200 Мбайт/с) схемы ввода и вывода, а также через низкоскоростные устройства передачи запросов и ответов. В комплексе «хост-ВС—Cray T3D» запросы и ответы используются для управления потоком передачи данных через высокоскоростные схемы ввода и вывода.

Средства синхронизации Cray T3D. В вычислительной системе Cray T3D весь коллектив ВУ и каналы ввода-вывода работают синхронно. Это достигается при помощи генератора тактовых импульсов (Clock), кото-

рый посылает импульсы одновременно и в ВУ, и в узлы ввода и вывода. Генератор работает на частоте 150 МГц.

Для синхронизации параллельных вычислительных процессов (реализуемых в различных ЭП) в ВС Cray T3D имеются специальные аппаратурные средства. Эти средства распределенные, т. е. они организуются из специальных локальных схем поддержки синхронизации, расположенных в элементарных процессорах.

В системе Cray T3D осуществлена аппаратурная реализация механизмов синхронизации «барьер» и «эврика» (Barrier/Eureka). Для реализации механизма «барьер» в каждой ветви параллельной программы задается точка синхронизации, при достижении которой каждый элементарный процессор должен ждать до тех пор, пока остальные ЭП не дойдут до своих точек, и лишь после этого все процессоры могут продолжать работу дальше. Ясно, что такая синхронизация требуется перед осуществлением коллективных обменов информацией между ветвями параллельной программы (см. разд. 3.3.5). Механизм синхронизации «эврика» реализует, по сути, операцию перехода в параллельных процессах, если из них даже только один достиг точки синхронизации. Механизмы синхронизации необходимы для реализации программирования, характерного и для SIMD-, и для MIMD-архитектур.

Эффективность системы Cray T3D. Не претендуя на полноту исследования, оценим эффективность ВС Cray T3D по Амдалу. Закон Амдала обычно применяют при оценке ускорения векторных ВС (с конвейерной организацией вычислений). Система Cray T3D относится к ВС с массовым параллелизмом, вместе с тем она принадлежит к числу изделий фирмы Cray Research Inc., поэтому по Амдалу рассчитывают и ее эффективность. В табл. 4.1 приведены максимальные значения для коэффициента ускорения по Амдалу:

$$\chi^* \leqslant \frac{1}{\delta + (1-\delta)/n},$$

где δ — доля последовательных вычислений при реализации программы; n — число элементарных процессоров.

Таблица 4.1

Число ЭП	Доля последовательных вычислений, %				
	50	25	10	5	2
32	1,94	3,66	7,80	12,55	19,75
512	1,99	3,97	9,83	19,28	45,63
2048	2,00	3,99	9,96	19,82	48,83

Из табл. 4.1 видно, что ускорение тем выше, чем меньше доля последовательных вычислений. Избежать последовательных участков в параллельной программе нельзя; в самом деле, в программе всегда присутствуют сугубо последовательные действия, например инициализация и операции ввода-вывода. Однако результаты, приведенные в табл. 4.1, заставляют задуматься и над основным: как достичь при работе на ВС с массовым параллелизмом линейной зависимости ускорения от числа элементарных процессоров? Резюмируя опыт работы пользователей Cray T3D, а главное, опираясь на наши отечественные результаты по параллельному программированию, можно заключить, что кардинальный путь повышения эффективности ВС с массовым параллелизмом связан с методикой крупноблочного распараллеливания сложных задач (см. разд. 3.3.6).

4.4.2. Вычислительная система Cray T3E

Система с массовым параллелизмом Cray T3E создана с учетом опыта создания и эксплуатации ВС Cray T3D. Она была построена в 1995 г. Количество элементарных процессоров в ВС достигает 2048, производительность — 2,76 TFLOPS, емкость памяти — 1 Тбайт. Цена 128-процессорной конфигурации Cray T3E составила 3...4 млн долл.

Производительность ВС определяется как количеством процессоров, так и возможностями базового микропроцессора. Выделяют несколько модификаций ВС: Cray T3E, Cray T3E-900, Cray T3E-1200, Cray T3E-1200E, Cray T3E-1350, с тактовыми частотами от 300 до 675 МГц. Барьер производительности 1 TeraFLOPS, т. е. 10^{12} операций с плавающей запятой в секунду над 64-разрядными данными, был впервые преодолен на системе Cray T3E-1200 в 1998 г.

Архитектура ВС Cray T3E относится к классу MIMD, но она более развитая по сравнению с совместимой архитектурой Cray T3D. Отметим две архитектурные особенности Cray T3E:

- мультипрограммирование — возможность одновременной реализации нескольких параллельных программ на различных подсистемах;
- масштабируемость — варьируемость количества элементарных процессоров с квантом 4 или 8 (производятся модулями с 4 или 8 ЭП в зависимости от вида охлаждения ВС, воздушного или жидкостного).

Следует отметить, что в ВС Cray T3D был реализован только монопрограммный режим. Следовательно, если для решения какой-либо задачи не требовались все ресурсы ВС, то имели место простои неиспользованных ЭП. В Cray T3E мультипрограммирование позволяет избежать простоев элементарных процессоров. Далее, в системе Cray T3D допускались конфигурации только с числом ЭП, кратным 32. В системе Cray T3E существенно шире

возможности по формированию конфигураций, адекватных сферам применения.

Функциональные структуры ВС Cray T3D и Cray T3E на макроуровне полностью идентичны. Каждая из них представляется композицией множества ВУ, коммуникационной сетью в виде трехмерного тора, каналов ввода-вывода и средств синхронизации. Однако при технической реализации Cray T3E нашли место новшества. Рассмотрим архитектурные возможности ВС Cray T3E подробнее.

Вычислительный узел Cray T3E. Ядром элементарного процессора ВУ (см. рис. 4.6) в любой модификации Cray T3E служит микропроцессор семейства DEC 21164 Alpha. Остановимся на спецификации микропроцессора для модификации ВС Cray T3E-1350, т. е. DEC 21164 Alpha (EV5.6):

Разрядность операндов	32 или 64
Тактовая частота, МГц	675
Производительность:	
2 операции с плавающей запятой за такт, MFLOPS	1350
4 инструкции за такт, MIPS	2700
Быстродействие канала к памяти, Мбайт/с	1200

Элементарный процессор располагает своей локальной памятью, емкость которой варьируется от 64 до 512 Мбайт (в зависимости от модификации ВС, в частности). В системе Cray T3E-1350 локальная память ЭП составляет 250...512 Мбайт и формируется из 64-Мбайтных DRAM-схем.

В вычислительном узле ВС Cray T3E в отличие от ВУ системы Cray T3D предусмотрена специальная связь (Link) для непосредственного подключения устройств ввода-вывода (УВВ) информации (см. рис. 4.6). Эта связь предоставляет потенциальную возможность подключения к любому ВУ внешних средств. Однако далеко не все ВУ должны оснащаться устройствами ввода-вывода. Если же такое подключение имело место, то устройство ввода-вывода становилось общим ресурсом для четырех вычислительных узлов.

Коммуникационная сеть Cray T3E. Сеть межузловых связей Cray T3E представляет собой трехмерный тор с двунаправленными каналами (см. рис. 4.7). Она имеет малое время задержки при пересылке сообщений (обладает низкой латентностью, Latency) и характеризуется значительной шириной полосы пропускания. Так, например, модификация Cray T3E-1350 имеет быстродействие 650 Мбайт/с в каждом из двух направлений передачи информации. Данная сеть в 3–4 раза превосходит по быстродействию аналогичную сеть Cray T3D.

Каналы ввода-вывода Cray T3E. В системе Cray T3E реализована возможность осуществлять обмен информацией с внешней средой через множество каналов ввода-вывода (портов).

Каналы ввода-вывода ВС Cray T3E интегрированы в трехмерную коммуникационную сеть так, что их количество всегда пропорционально числу элементарных процессоров в любой конфигурации системы. Таким образом, при масштабировании ВС происходит и адекватное масштабирование пропускной способности каналов ввода-вывода.

Ясно, что все каналы ввода-вывода (все их узлы ввода и вывода) Cray T3E закоммутированы в два гигакольца (GigaRings), данные по которым перемещаются в противоположных направлениях. Суммарная пропускная способность этих гигаколец равна 1 Гбайт/с; максимальная полоса пропускания любого интерфейса гигакольца составляет 500 Мбайт/с.

Конструктивные особенности системы Cray T3E. Вычислительная система Cray T3E изготовляется в двух вариантах корпусов: с воздушным и жидкостным охлаждением. В первом варианте конструктивный модуль для компоновки (масштабирования) ВС представляется платой из четырех элементарных процессоров, а при применении жидкостного охлаждения подобный модуль имеет две платы (8 ЭП). Следовательно, в системах с воздушным или жидкостным охлаждением масштабирование осуществляется на величину, кратную 4 или 8 ЭП соответственно. При этом каждая 4-процессорная плата имеет только один вывод на разъем корпуса для ее включения в гигакольцо ввода-вывода.

В корпусе с жидкостным охлаждением (например, Cray T3E-1350) размещается 272 ЭП, из которых 256 ЭП являются основными, а остальные 16 ЭП составляют избыточность (резерв). Следовательно, на каждые 16 основных ЭП предусматривается один избыточный процессор. Максимальная конфигурация ВС Cray T3E размещается в восьми корпусах и насчитывает 2176 элементарных процессоров, из которых число основных ЭП равно 2048.

Ясно, что в любой конфигурации Cray T3E избыточность оценивается 6,25 %, и она используется компонентами операционной системы и обеспечивает высокий уровень надежности ВС в целом.

Программное обеспечение Cray T3E. Архитектурные особенности MPP-систем потребовали от Cray Research Inc. разработки нового ПО, учитывающего мировой опыт и традиции в параллельном программировании, а также в программировании PVP-систем Cray.

Операционная система UNICOS/mk, разработанная для ВС Cray T3E, является распределенной и масштабируемой версией UNICOS (последняя использовалась в PVP-системах: Cray-1, Cray X-MP, Cray-2; UNICOS — в свою очередь, производная от системы UNIX).

Масштабируемая ОС (Scalable Operating System) UNICOS/mk разделена на программы — серверы, распределенные по элементарным процессорам ВС Cray T3E. Локальные серверы ОС обрабатывают запросы, специфичные для каждого ЭП ВС. Глобальные серверы обеспечивают общесис-

темные возможности, такие как управление процессами и файловые операции. Последние серверы размещаются в специальных системных ЭП и не дублируются в пределах ВС.

Система UNICOS/mk поддерживает масштабируемую архитектуру ввода-вывода Cray T3E. Она использует стандартные утилиты и команды ОС UNIX, следовательно, она обеспечивает знакомую операционную среду для пользователей и администраторов.

Средства программирования Cray T3E:

- языки программирования и компиляторы: FORTRAN 90, C и C++, они используются для написания программ и их преобразования в эквивалентные объектные программы (на машинном языке);
- пакет поддержки параллельного программирования MPT (Message Passing Toolkit) реализует взаимодействия между ветвями параллельной программы. Пакет включает широко применяемые интерфейсы передачи сообщений: MPI, MPI-2 и PVM;
- отладчик (Cray Total View Debugger) используется для отладки прикладных параллельных программ на уровне исходного текста. Он позволяет пользователям отображать и анализировать информацию о параллельных процессах;
- интерактивная среда (Cray Program Browser) применяется для отображения и редактирования файлов и прикладных программ;
- обучающая система (MPP Apprentice) дает рекомендации по повышению производительности MPP-системы, отображает данные по производительности и интерпретирует их;
- библиотеки оптимизированных параллельных прикладных программ.

В результате многолетней работы фирмы Cray Research Inc. по развитию архитектуры средств обработки информации пройден путь от канонической конвейерной ВС до ВС с массовым параллелизмом. Последние системы с достаточной полнотой основываются на модели коллектива вычислителей (см. § 3.1). По архитектуре они вплотную подошли к распределенным ВС с программируемой структурой, архитектурно гибкие образцы которых были разработаны и построены еще в середине 1970-х годов. Отделом вычислительных систем Сибирского отделения АН СССР совместно с промышленными организациями (см. гл. 7).

4.4.3. Вычислительная система Cray XT3

Массово-параллельную вычислительную систему Cray XT3 производит один из лидеров в области суперкомпьютеров — Cray Inc. (основана в 2000 г.). В конфигурациях системы Cray XT3 число элементарных процес-

соров может достигать 30 720, а производительность — 318 TFLOPS, емкость памяти — 239 Тбайт.

Вычислительная система Cray XT3 представляется композицией множества элементарных процессоров трехмерной тороидальной структуры и системы ввода-вывода информации.

Элементарный процессор Cray XT3. Элементами структуры ВС являются вычислительные и сервисные элементарные процессоры (Processing Elements). Независимо от функционального назначения в состав каждого ЭП входит локальный коммутатор (называемый Cray SeaStar), микропроцессор AMD Opteron и локальная память. Помимо названных компонентов в сервисном ЭП имеются две 64-разрядные шины (Dual PCI-X) для взаимодействия с системой ввода-вывода информации.

Локальный коммутатор (ЛК) SeaStar — сложное функциональное устройство, реализованное на одной БИС. Он предназначен для выполнения высокоскоростной маршрутизации и обменов информацией (как межпроцессорных, так и внутри ЭП). В состав ЛК входят: коммуникационный процессор (Communications and Management Processor), высокоскоростной сетевой маршрутизатор (Interconnect Router), канал HyperTransport, контроллер прямого доступа к памяти DMA (Direct Memory Access), а также сервисный порт.

Сетевой маршрутизатор работает с шестью межпроцессорными каналами связи и тем самым обеспечивает непосредственную связь данного ЭП с шестью соседними ЭП по трем направлениям X, Y, Z в 3D-торе. Пиковая пропускная способность каждого межпроцессорного канала в двунаправленном режиме составляет 7,6 Гбайт/с. Маршрутизатор реализует также протокол коррекции ошибок и повторной передачи информации по межпроцессорным каналам.

Канал HyperTransport служит для связи микропроцессора данного ЭП с коммуникационной сетью Cray XT3 (следовательно, со всеми остальными ЭП). Пропускная способность данного канала — 6,4 Гбайт/с.

Контроллер DMA обеспечивает (без каких-либо прерываний микропроцессора) доступ к локальной памяти данного ЭП другим элементарным процессорам через коммуникационную сеть Cray XT3. Контроллер DMA и операционная система функционируют совместно, что дает возможность минимизировать латентность коммуникационной сети.

Сервисный порт кристалла SeaStar позволяет системе обеспечения надежности и управления получить доступ к его регистрам через специальную супервизорную сеть Cray XT3. Данные средства необходимы для осуществления начальной загрузки, технического обслуживания, обеспечения надежности и мониторинга суперВС Cray XT3.

Для компоновки элементарных процессоров могут быть применены любые 64-разрядные микропроцессоры семейства AMD Opteron (включая одно- и двухъядерные — Single or Dual Core).

Локальная память каждого ЭП Cray XT3 составляет 1...8 Гбайт, скорость обращения микропроцессора к локальной памяти — 6,4 Гбайт/с.

Коммуникационная сеть Cray XT3. Структура сети межмашинных связей Cray XT3 является традиционной для MPP-систем фирмы Cray — трехмерный тор (3D-тор). В вершинах такой тороидальной структуры находятся локальные коммутаторы SeaStar.

Коммуникационная сеть Cray XT3 характеризуется высокой пропускной способностью (7,6 Гбайт/с), низкой латентностью (средним временем задержки при межпроцессорных обменах — около 3 нс).

Конструктивные особенности системы Cray XT3. ВС Cray XT3 компонуется из стоек, в каждой из которых может размещаться до 96 элементарных процессоров. Возможные конфигурации ВС и их характеристики представлены в табл. 4.2.

Таблица 4.2

Число стоек	Структура ВС — фрагмент 3D-тора: $N(x \times y \times z)$	Число процессоров		Пиковая производительность, TFLOPS		Емкость памяти, Тбайт
		вычислительных	сервисных	A*	B**	
6	576 (6×12×8)	548	14	2,6	5,6	4,3
24	2304 (12×12×16)	2260	22	10,8	23,4	17,7
96	9216 (24×16×24)	9108	54	43,7	94,6	71,2
320	30720 (40×32×24)	30508	106	147	318	239

* При использовании одноядерного процессора, 2,4 ГГц.

** При применении двухъядерного процессора, 2,6 ГГц.

При формировании ВС Cray XT3 используют конструктивные модули двух типов — вычислительные и сервисные (Compute and Service Blades). Первые модули включают в себя четыре вычислительных ЭП, а вторые — два сервисных ЭП. Количество ЭП, находящихся в модуле, и определяет минимальный фрагмент для масштабирования структуры ВС.

Размер стойки — 204,5×57,2×144,1 см^3, ее масса — 694 кг, потребляемая мощность — 14,5 кВт.

Программное обеспечение Cray XT3. В состав программного обеспечения Cray XT3 входят масштабируемые операционная система UNICOS/lc и среда программирования. Операционная система UNICOS/lc (модифицированная UNICOS/mk, позволяющая эффективно функционировать конфигурациям до 120 000 процессорных ядер).

Основу системы UNICOS/lc составляют два компонента — микроядро для вычислительных элементарных процессоров и полнофункциональная ОС для сервисных ЭП.

Среда программирования для суперВС Cray XT3 представлена компиляторами FORTRAN 77, 90, 95 и C, C++, коммуникационными библиотеками (в частности, MPI 2.0), математическими библиотеками программ и др.

Система обеспечения надежности и управления Cray XT3. СуперВС Cray XT3 оснащена аппаратурно-программной системой CRMS (Cray RAS and Management System; RAS — Reliability, Availability, Serviceability — надежность, готовность, обслуживаемость), которая обеспечивает мониторинг, идентификацию неисправностей и восстановление. В состав CRMS входят специальные управляющие процессоры и гигабитная супервизорная сеть, а также программный комплекс, размещенной в рабочей станции. В каждой стойке установлено 24 процессора для управления вычислительными и сервисными модулями и один процессор для связи с рабочей станцией.

Система CRMS осуществляет мониторинг всех основных и аппаратурных, и программных компонентов Cray XT3. Кроме того, CRMS контролирует включение и выключение электропитания и последовательность начальной загрузки, управляет коммуникационной сетью и выводит на дисплей информацию о состоянии ВС для системного администратора.

Система CRMS — независимая, ее работа не отвлекает ресурсы ВС, занятые при выполнении пользовательских программ.

Система ввода-вывода Cray XT3. Система ввода-вывода суперВС является высоко масштабируемой параллельной файловой системой. Она реализована как массив дисков, подключенных к сервисным элементарным процессорам. Высоконадежные диски со световодными каналами и контроллеры обеспечивают скорость ввода-вывода информации до 100 Гбайт/с.

Вычислительная система Cray XT4. СуперВС Cray XT4 является генерацией MPP-систем 2007 г., созданных Cray Inc. Данная ВС полностью совместима с Cray XT3, но формируется из многоядерных микропроцессоров (в частности, из четырехъядерных AMD Opteron).

Конфигурация системы Cray XT4, состоящая из 23016 процессорных ядер, характеризуется пиковой производительностью 119,35 TeraFLOPS (101,7 TFLOPS на тестовом наборе LINPACK). Она занимает вторую позицию в 29-й редакции списка top500 и эксплуатируется в США (Oak Ridge National Laboratory) с 2007 г.

Cray Inc. планирует создать конфигурацию Cray XT4 из 120000 процессорных ядер, обладающую пиковой производительностью свыше 1 PetaFLOPS.

4.5. Сверхвысокопроизводительные вычислительные системы семейства Cray X

Семейство сверхвысокопроизводительных ВС Cray X разрабатывается Cray Inc. В конце 2002 г. анонсированы две модели: Cray X1 и

Cray X2 соответственно с производительностью порядка 5 10^{13} опер./с и 8 10^{14} опер./с.

Система Cray X1 относится к числу самых высокопроизводительных средств обработки информации первого десятилетия XXI в. Ее рассматривают как промежуточный этап в решении стратегической проблемы США (и, в частности, корпорации Cray) — достичь к 2010 г. скорости вычислений 1 PFLOPS, т. е. одного квадриллиона или 10^{15} операций с плавающей запятой в секунду. Данная проблема была поставлена в 1999 г. в докладе Президентского консультационного комитета по информационным технологиям (President's Information Technology Advisory Committee). В США считается, что создание высокопроизводительных ВС осуществляется исключительно в интересах национальной безопасности (!).

Вычислительная система Cray X1 предназначается как для академических, так и прикладных исследований, для решения сверхтрудоемких (high-end) задач науки, техники, экономики и военной сферы. Разработка ВС Cray X1 получила поддержку от нескольких организаций правительства США, включая Агентство национальной безопасности (NSA — National Security Agency).

4.5.1. Особенности архитектуры Cray X1

Максимальная конфигурация ВС Cray X1 состоит из 4096 элементарных процессоров (или 49 152 вычислителей, среди которых 32 768 векторных конвейеров и 16 384 скалярных блоков). Она имеет производительность 52,4 TFLOPS и память емкостью 16...64 Тбайт. Вес такой конфигурации ВС составляет примерно 230 т (при воздушном охлаждении) или 170 т (при жидком хладагенте). Цена 16-процессорной ВС (204,8 GFLOPS) составляет 16,4 млн долл.

Система Cray X1 была официально анонсирована в ноябре 2002 г. Первые поставки Cray X1 (в упрощенных конфигурациях, но допускающих модернизацию) произведены в конце 2002 — начале 2003 г. К числу первых организаций, которые приобрели конфигурации Cray X1, относятся: Научно-исследовательский центр высокопроизводительных вычислений армии США (AHPCRC — U.S. Army High Performance Computing Research Center), Испанский национальный институт метеорологии (Spain's National Institute of Meteorology), Оук-Риджская национальная лаборатория (ORNL — Oak Ridge National Laboratory) Отдела энергетики США (U.S. Department of Energy).

Вычислительная система Cray X1 — это MIMD-система с общей распределенной памятью (Distributed Shared Memory). В системе Cray X1 просматривается иерархия уровней ее функциональной структуры, организо-

ванной по принципу матрешки. Действительно, в вычислительный элемент фиксированного уровня вкладывается композиция элементов очередного нижнего уровня. Модель коллектива вычислителей реализована на всех иерархических уровнях функциональной структуры Cray X1.

В архитектуре Cray X1 нашли отражение множество достижений из различных классов ВС (см. разд. 3.4.2), включая как PVP-, так и MPP-системы. Данная ВС основывается на тороидальной топологии и имеет широкую полосу пропускания и низкую латентность (малые задержки при передаче информации между ресурсами). Cray X1 характеризуется высокой надежностью и живучестью, а также масштабируемостью. Диапазоны возможных конфигураций, производительности и емкости памяти Cray X1 соответственно равны: 8...4096 процессоров, 102,4 GFLOPS...52,4 TFLOPS и 32 Гбайт...64 Тбайт.

В систему Cray X1 вложен новейший набор команд, активные исследования по которому велись в корпорации Cray в течение 10 лет. Считается, что архитектура ВС с этим набором команд будет отвечать достижениям в интегральной технологии, по крайней мере, в течение десятилетия. Набор команд Cray X1 весьма прост, в нем нет сложных и избыточных инструкций. Он рассчитан на использование очень больших регистровых файлов, поддерживает 64- и 32-разрядные вычисления, реализует новый механизм синхронизации, обеспечивающий масштабируемость ВС и др. В результате Cray X1 обладает рядом преимуществ по сравнению с другими архитектурами суперкомпьютеров:

- высоким вычислительным параллелизмом (при низкой пропускной способности инструкций);
- незначительной сложностью управления;
- небольшим энергопотреблением, соотнесенным к одной операции в секунду;
- низкой латентностью.

Таким образом, архитектура ВС Cray X1 позволяет формировать конфигурации, адекватные областям применения, параметрам решаемых суперсложных задач.

Система Cray X1 — это композиция множества мультипроцессорных вычислительных узлов, коммуникационной сети между узлами и средств ввода-вывода данных. Среда программирования Cray X1 поддерживается специальным сервером.

4.5.2. Вычислительный узел Cray X1

Вычислительная система Cray X1 может иметь в своем составе от 2 до 1024 однородных вычислительных узлов (ВУ). В каждом ВУ имеется четыре ЭП и распределенная общедоступная оперативная память (рис. 4.8).

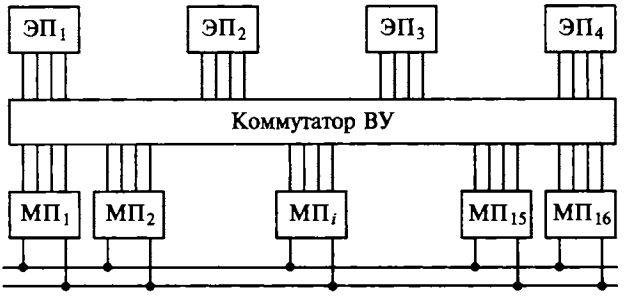


Рис. 4.8. Вычислительный узел Cray X1:
ЭП — элементарный процессор; МП — модуль памяти

Взаимодействие между процессорами и оперативной памятью в узле осуществляется при помощи коммутатора ВУ (Crossbar). Следовательно, вычислительный узел по своей функциональной структуре является мультипроцессорной ВС (см. разд. 3.4.2).

Каждый ЭП представляет собой специально спроектированный конвейерный (или векторный) процессор, обладающий производительностью 12,8 GFLOPS (при обработке 64-разрядных операндов). Процессор поддерживает также арифметику над 32-разрядными данными.

Элементарный процессор относится к типу мультипотоковых процессоров (MSP — Multi-Streaming Processors), если придерживаться терминологии Cray Inc. Вообще такой процессор, по сути, является конвейерным (или векторным), но с той особенностью, что он состоит из множества небольших конвейеров (Pipes), работающих параллельно. В вычислительной системе Cray X1 функциональная структура MSP усовершенствована, в процессоре дополнительно имеются схемы синхронизации и кэш-память.

В системе Cray X1 элементарный процессор (рис. 4.9) состоит из четырех секций обработки информации (СОИ), четырех блоков кэш-памяти и коммутатора ЭП. Каждая из секций обработки включает в себя скалярный блок с кэш-памятью для данных (СБ & КЭШ) и пару векторных конвейеров (ВК). Скалярный блок имеет тактовую частоту 400 МГц и может выполнять две операции за такт. Быстродействие четырех скалярных блоков ЭП составляет 3,2 GIPS (3,2 10^9 операций с фиксированной запятой в секунду).

Векторные конвейеры ЭП работают параллельно, синхронно и с тактовой частотой 800 МГц. Их суммарная производительность составляет 12,8 GFLOPS или 25,6 GFLOPS при обработке 64- или 32-разрядных данных. (В самом деле, на каждый из конвейеров поступает два вектора-операнда, следовательно, за один такт восемь конвейеров способны обработать 16 элементов векторов.)

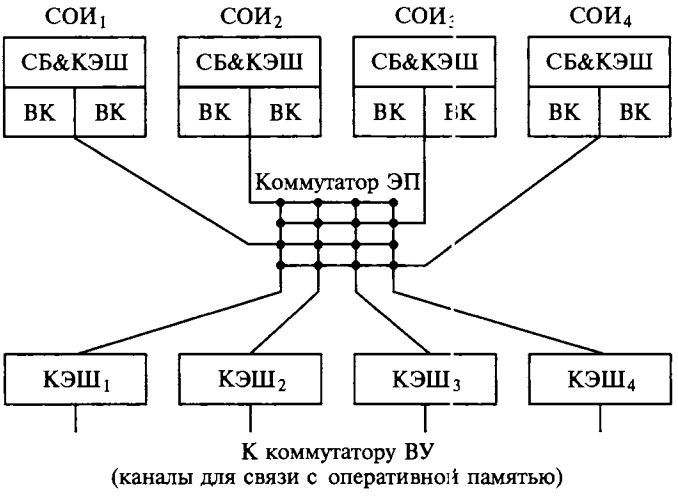


Рис. 4.9. Элементарный процессор Cray X1:

СОИ — система обработки информации; СБ — скалярный блок; КЭШ — кэш-память; ВК — векторный конвейер

Следует отметить, что читатель без труда обнаружит сходство функциональных структур системы STAR-100 (см. разд. 4.2.1 и рис. 4.3) и секции обработки информации Cray X1 (рис. 4.9). В той и другой структурах имеется по два векторных конвейера и один скалярный вычислитель со своей сверхоперативной памятью (называемой буферной памятью и КЭШ соответственно в STAR-100 и Cray X1).

Кэш-память ВУ обеспечивает когерентность между быстродействием при обработке информации и скоростью ввода данных из оперативной памяти, т. е. она играет роль сверхоперативной буферной памяти между секциями обработки информации и оперативной памятью (см. разд. 4.2.1). Кэш-память состоит из четырех блоков, ее суммарная емкость достигает 2 Мбайт.

Коммутатор ЭП (Crossbar) обеспечивает доступ каждой секции обработки информации к любому блоку кэш-памяти.

Полоса пропускания в направлении от кэш-памяти к секциям обработки информации равна 102,4 Гбайт/с, а наоборот — 51,2 Гбайт/с. Четыре канала (см. рис. 4.9) между кэш-блоками и оперативной памятью обеспечивают обмен информацией со скоростью 76,8 Гбайт/с.

В составе ВУ имеется оперативная память, доступная каждому ЭП. Память ВУ формируется из Rambus DRAM-микросхем, производимых Samsung Electronics Co. Ltd. Rambus-чипы характеризуются значительными емкостью и пропускной способностью.

Память любого узла (см. рис. 4.8) представляется множеством из 16 4-канальных МП; для максимизации ее пропускной способности используется 16 контроллеров.

Каждый элементарный процессор (при помощи своих четырех кэш-блоков) имеет доступ (через коммутатор ВУ) к каждому модулю ОП узла. При этом любой кэш-блок ЭП связан только со своей группой из четырех модулей памяти. Поскольку любая из четырех секций обработки информации (см. рис. 4.9) связана через коммутатор со всеми блоками $\{КЭШ_i\}$, $i = \overline{1, 4}$, то в пределах узла любая $СОИ_i$ имеет доступ к любому модулю памяти $МП_j$, $j = \overline{1, 16}$. Пропускная способность «канала» между ЭП и оперативной памятью в вычислительном узле составляет 34,1 Гбайт/с.

Оперативная память ВУ доступна для других ВУ системы (см. рис. 4.8); этот доступ реализуется при помощи специальных маршрутизаторов.

Итак, все модули памяти в системе Cray X1 физически распределены по ВУ (и, следовательно, по элементарным процессорам), но логически они доступны каждому ЭП, т. е. оперативная память ВС Cray X1 является и распределенной, и общей.

Следует отметить, что элементарный процессор является основным функциональным элементом Cray X1. Он конструктивно выполнен в виде многокристального модуля. Конструкция вычислительного узла Cray X1 оформлена в виде платы, содержащей четыре конструктивных модуля — процессора, схемы памяти и коммутатора.

4.5.3. Коммуникационная сеть Cray X1

Взаимодействие между вычислительными ресурсами (узлами и, следовательно, элементарными процессорами и памятью) в системе Cray X1 осуществляется через коммуникационную сеть. Архитектурные решения, заложенные в коммуникационную сеть, позволили достичь в сверхвысокопроизводительной системе Cray X1 высокой надежности и живучести, масштабируемости, большой пропускной способности и незначительной латентности (задержки) при передаче информации между ресурсами. Так, например, пропускная способность сети в 64-процессорной конфигурации Cray X1 (819,2 GFLOPS, 256 Гбайт) с жидкостным охлаждением составляет 400 Гбайт/с.

В системе Cray X1 для реализации коммуникационной сети применен модифицированный двумерный тор (Modified 2D Torus). В чем состоит суть модификации 2D-тора и что является его вершиной?

Выше (см. разд. 4.5.2) было отмечено, что ВУ (см. рис. 4.8) обладает двумя маршрутами для связи с другими ВУ в пределах ВС Cray X1. (Если быть более точным, то в узле имеется 16 пар отдельных маршрутов, по од-

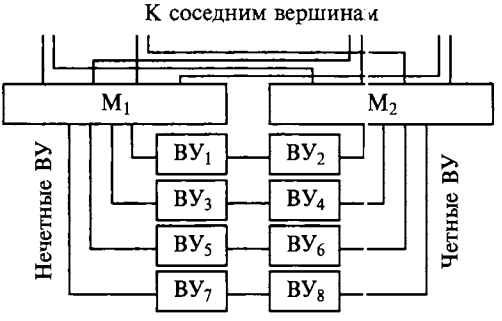


Рис. 4.10. Вычислительная вершина двумерной тороидальной системы Cray X1:
М — маршрутизатор; ВУ — вычислительный узел

ной на каждый из 16 модулей памяти, что гарантирует живучесть и необходимую полосу пропускания связей между ВУ.) Один из этих маршрутов используется для того, чтобы организовать в системе Cray X1 множество связных пар: «ВУ с нечетным номером — смежный ВУ с четным номером». Другой маршрут служит для подключения ВУ к маршрутизатору (Router).

В качестве вершины (Vertex) двумерного тора используется композиция из четырех пар вычислительных узлов и двух маршрутизаторов (M_1, M_2), работающих на четыре внешних связи (рис. 4.10). Индекс в обозначении $ВУ_k$, $k = \overline{1,8}$, не является физическим номером ВУ в пределах ВС, он информирует лишь о четности или нечетности номера. Маршрутизаторы M_1 и M_2 обеспечивают два параллельных канала связи данной вершины с соседними вершинами в 2D-торе.

Очевидно, что структура (граф) вершины в системе Cray X1 обладает диаметром, равным двум. (Диаметр графа — максимальное расстояние, определяемое на множестве кратчайших путей между вершинами всевозможных пар.) Это следует из того, что маршрутизатор не имеет задержки, сравнимой со временем обмена информацией между памятями различных ВУ. Значит, в вершине обмен информацией между любыми ВУ осуществляется с использованием максимум одного транзитного узла или, говоря иначе, он производится посредством двух пересылок (Hops): из данного ВУ — в транзитный, а затем в ВУ — приемник.

Двумерный тор Cray X1 представляет собой «бублик», на поверхности которого размещена двумерная структура, вершины (см. рис. 4.10) которой связаны в двух направлениях: по окружности «бублика» и по окружности его сечения. Примерами таких структур могут служить трех- и четырехмерные гиперкубы (с числами вершин и ребер, равными 8 и 3 или 16 и 4, см. рис. 3.2). Не требуется особого воображения увидеть в этих гиперкубах двумерные торы.

Четырехмерный гиперкуб использован в конфигурации Cray X1, состоящей из 128 ВУ (512 ЭП). Ясно, что в этой конфигурации ВС сама вершина имеет, в свою очередь, свою структуру из восьми ВУ и двух маршрутизаторов, а ребра в гиперкубе отражают двойные каналы межвершинных связей (см. рис. 4.10).

Четырехмерный гиперкуб (см. рис. 3.2) — это структура из трехмерного куба внутри такого же куба и с ребрами между соответствующими вершинами этих кубов. Четырехмерный куб Cray X1 характеризуется тем, что вершины (точнее, их маршрутизаторы) каждого трехмерного куба входят в двойные циклы. Следовательно, в четырехмерном кубе один из концентрических циклов представляет вычислительные узлы (см. рис. 4.10) с нечетными номерами, а другой — ВУ с четными номерами. В четырехмерном гиперкубе Cray X1 имеют место также связи между соответствующими циклами с нечетными ВУ двух трехмерных кубов, а также между циклами с четными узлами этих же кубов. Заметим, что связи между циклами с нечетными ВУ и с четными узлами организуются в пределах вершин и так, как это показано на рис. 4.10.

Оценим задержки, которые существуют при передаче информации между вычислительными узлами в четырехмерном гиперкубе Cray X1. Ясно, что максимальное расстояние (из кратчайших) между любыми двумя нечетными ВУ (или четными узлами) равно четырем, т. е. для обмена информацией между этими узлами потребуется максимум четыре пересылки. Максимальное расстояние (из кратчайших) между любыми нечетным и четным ВУ увеличивается на единицу. Здесь используется не три, а четыре транзитных узла (необходима пересылка между узлами внутри вершины).

Рассмотрим, как решается проблема масштабирования структуры ВС Cray X1. Корпорация Cray в конфигурациях системы Cray X1 не использует структуры в виде гиперкубов при числе ВУ, превышающем 128. Гиперкубы бóльших размерностей, чем четыре, потребовали бы включения в состав вершин дополнительных маршрутизаторов (см. рис. 4.10), что породило бы набор неоднородных вершин. Вместо этого Cray Inc. «растягивает» четырехмерный куб, превращая его в 2D-тор с бóльшей «окружностью», и тем самым увеличивает число вершин (рис. 4.11). Очевидно, что при таком способе масштабирования ВС величина «приращения» (или «сокращения») для количества вершин равна четырем. При этом следует заметить, что минимальное число вычислительных узлов в вершине равно двум. Следовательно, минимальная величина аппаратурного приращения ВС равна восьми ВУ (или 32 ЭМ).

Система Cray X1 имеет иерархическую структуру сети связей между вычислительными ресурсами. На каждом структурном уровне используется

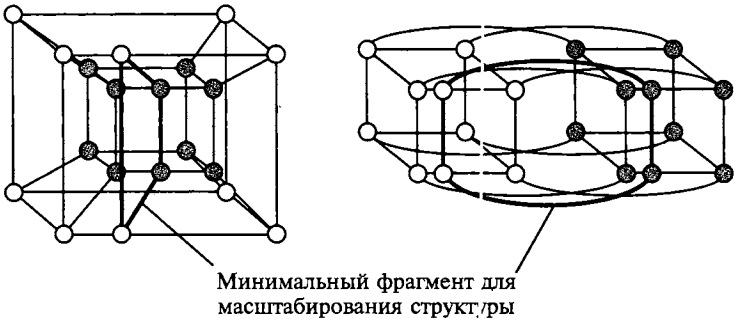


Рис. 4.11. Растянутый 4D-куб (2D-тор с 20 вершинами)

свой тип графа межресурсных связей и свой тип вычислительных ресурсов — элементов обработки информации (табл. 4.3). Такое структурное решение в системе Cray X1 позволило достичь оптимума по эффективности в условиях технических и технологических ограничений на рубеже между XX и XXI столетиями.

Таблица 4.3

Структурный уровень Cray X1	Вычислительный элемент структуры
Макроуровень — двумерный тор (см. рис. 3.2)	Вершина — четырехполюсник, отражающий композицию из восьми вычислительных узлов (ВУ) и двух маршрутизаторов (M_1, M_2). Элементы M_1 и M_2 формируют два двумерных канала
Структура вершины — граф, состоящий из четырех пар связных ВУ с четными и нечетными номерами и двух маршрутизаторов, каждый из которых соединен ребрами либо с четными, либо с нечетными узлами (см. рис. 4.10). Диаметр сети межузловых связей равен двум	Вычислительный узел — двухполюсник, представляющий композицию из четырех элементарных процессоров (ЭП), 16 модулей памяти (МП) и коммутатора ВУ. Маршрутизатор — четырехполюсник по внешним связям, позволяет формировать двумерные структуры
Структура вычислительного узла — граф, дающий связность каждого из четырех ЭП с каждым из 16 модулей памяти (см. рис. 4.8)	Элементарный процессор — четырехполюсник, соответствующий композиции из четырех секций обработки информации (СОИ), четырех блоков кэш-памяти и коммутатора ЭП. Коммутатор ВУ обеспечивает связность между процессорами $ЭП_1$–$ЭП_4$ и модулями памяти ($МП_1$–$МП_{16}$)

Окончание табл. 4.3

Структурный уровень Cray X1	Вычислительный элемент структуры
Структура элементарного процессора — граф, создающий связность каждой из четырех СОИ с каждым из четырех блоков кэш-памяти (см. рис. 4.9)	Секция обработки информации — композиция из скалярного блока с кэш-памятью (СБ & КЭШ) и двух векторных конвейеров (ВК). Коммутатор ЭП дает связность между $СОИ_1$–$СОИ_4$ и блоками $КЭШ_1$–$КЭШ_4$

4.5.4. Средства ввода-вывода Cray X1

Средства ввода-вывода информации ВС Cray X1 распределены по ее ВУ. Каждый ВУ располагает четырьмя каналами ввода-вывода (I/O System Port Channels). Пиковая пропускная способность одного канала ввода-вывода составляет 1,2 Гбайт/с.

Каналы ввода-вывода ВС Cray X1 служат для подключения дисков и других периферийных устройств. Предусмотрена возможность использования волоконно-оптических линий связи.

Поддержка различных сетевых протоколов (в частности, для гигабитной Ethernet) осуществляется специальным сервером CNS (Cray Network Server).

4.5.5. Конструкция системы Cray X1

Для формирования ВС Cray X1 используются корпуса двух вариантов, с воздушным и водяным охлаждением. В корпусе первого варианта размещается четыре вычислительных узла (16 элементарных процессоров), а второго варианта — 16 узлов (64 ЭП).

В табл. 4.4 приведены физические характеристики конструктивов для системы Cray X1.

Таблица 4.4

Тип корпуса ВС	Размер площадки, $м^2$	Вес, кг
Основной с воздушным охлаждением	0,9 × 1,5	895
Основной с жидкостным охлаждением	1,3 × 2,6	2610
Для средств ввода-вывода	0,75 × 1,1	512

4.5.6. Программное обеспечение Cray X1

Архитектура сверхвысокопроизводительной ВС Cray X1 является объединением архитектур PVP- и MPP-систем. Поэтому в ОС UNICOS/mp собрано все лучшее из PVP UNICOS и MPP UNICOS/mk.

Среди средств программирования Cray X1 имеются языки высокого уровня (параллельные FORTRAN и C), интерфейсы передачи сообщений (MPI), интерактивный отладчик, средства для анализа производительности ВС и др.

В системе Cray X1 среда программирования поддерживается специальным сервером — CPES (Cray Programming Environment Server). В частности, компиляторы работают не на самой системе Cray X1, а на CPES.

4.5.7. Области применения системы Cray X1

Cray X1 — универсальная сверхвысокопроизводительная масштабируемая вычислительная система. Архитектура Cray X1 позволяет формировать конфигурации ВС, в которых достигается оптимум между быстродействием, емкостью памяти, надежностью и ценой и которые адекватны областям применения. Все разнообразие видов деятельности человека, связанных с трудоемкими вычислениями, и составляет прикладные области для Cray X1. Но главными областями для данной ВС все же являются наука, техника и экономика (как гражданской, так и военных сфер).

Области применения Cray X1:

- фундаментальные научные исследования, вычислительная математика, физика, химия, астрономия (включая астрофизику), биология, науки о Земле;
- биотехнические исследования (изучение геномов организмов и строения белка), биоинформатика и моделирование биологических процессов;
- медицина и фармакология; виртуальная хирургия; создание лекарств; предсказание естественной пандемии и локальных эпидемий;
- экология, окружающая среда, климат, погода; моделирование процессов ионосферной плазменной физики; моделирование климата и атмосферы, долгосрочное, краткосрочное и очень краткосрочное прогнозирование погоды; изучение взаимодействия между океанической, воздушной и земной средами; моделирование цунами и воздушных течений, предсказание сильных штормов (бурь);
- аэрокосмические исследования и индустрия; вычислительная механика; моделирование летательных аппаратов; механика сплошных сред; аэро- и гидродинамика; проектирование в авиации и космонавтике, анализ эффективности горючего;
- экономика, моделирование национальной экономики и инвестиций в отрасли народного хозяйства (энергетику, транспорт и др.);

• оборона и военные приложения; моделирование и планирование обороны, наступления и боя; управление сложными техническими системами; предсказание климата после военных действий и погоды после боя; борьба с биотерроризмом и эпидемиями;

• промышленность, машиностроение, энергетика, металлургия; создание новых материалов; нанотехнологии и наноматериалы; анализ, проектирование и обеспечение безопасности в автомобилестроении; нефтехимический сейсмический анализ и др.

4.6. Анализ конвейерных вычислительных систем

Значительный интерес (на протяжении более 30 последних лет XX в.) к конвейерным ВС объяснялся тем, что они позволили преодолеть барьер производительности машин третьего поколения и давали рекордную производительность.

Конвейерные процессоры являются пределом модификации архитектуры последовательной ЭВМ. Следовательно, возможности повышения быстродействия средств ВТ при помощи конвейерной обработки информации принципиально ограничены: последовательность операций при передаче результатов по конвейеру не может быть произвольно длинной хотя бы из-за ограниченной надежности элементарных блоков обработки.

Конвейерные процессоры в архитектурном плане занимают промежуточное место среди средств обработки информации, базирующихся на модели вычислителя, и средств, основанных на модели коллектива вычислителей. Существует тенденция к неуклонному совершенствованию архитектуры конвейерных ВС в направлении к коллективу вычислителей. Высокий уровень быстродействия был достигнут в конвейерных ВС за счет мультиконвейерности (параллельной работы множества конвейеров) и конвейеризации на микроуровне (на уровне фаз выполнения арифметических операций).

Параллельно-векторные системы (PVP-systems) отражают предельный вариант в совершенствовании архитектуры конвейерных ВС. В самом деле, PVP-система — это коллектив, образованный из процессоров, а последние, в свою очередь, есть ни что иное, как композиция конвейеров. Следовательно, PVP-системы основываются на самой совершенной платформе — на MIMD-архитектуре и распределенности ресурсов.

Разнообразие конвейерных ВС — следствие возможностей в технической реализации модели коллектива вычислителей. Три принципа (см. разд. 3.1.1), положенные в основу коллектива вычислителей, достаточно ярко проявляются во всех конвейерных системах.

1. *Параллельность выполнения операций* в конвейерных системах обеспечивается на различных уровнях. На макроуровне параллельность выражается в возможности параллельной работы произвольного числа процессоров (Cray-2 — до 4, Cray Y-MP — до 8, Cray C90 — до 16, Cray T90 — до 32). На уровне процессоров предусматривается возможность параллельной работы порядка 10 конвейеров (в STAR-100 — трех, в Cray-1 — 12). На микроуровне закладывается возможность организации конвейеров из десятков элементарных блоков обработки информации (Cray-1 — от 1 до 14, STAR-100 — до 30).

2. *Программируемость структуры* — принцип, который с развитием архитектуры конвейерных ВС находит все более полное воплощение. Так, например, в STAR-100 обеспечивалась избирательная обработка компонентов векторов данных путем введения специального булевского вектора. В Cray-1 стало возможным программировать последовательность использования 12 конвейеров процессора. В последующих разработках систем Cray реализована возможность программирования межпроцессорных взаимодействий.

3. *Конструктивная однородность* — принцип, который в конвейерных ВС проявляется достаточно четко. Даже для простых конвейерных систем характерно наличие нескольких идентичных (или почти идентичных) конвейеров, небольшое разнообразие элементарных блоков обработки для формирования конвейеров, регулярность связей между блоками в конвейере, наличие нескольких одинаковых блоков в конвейере. Мультиконвейерные ВС (PVP-системы) полностью основаны на принципе однородности: это композиция множества однородных процессоров. В мультиконвейерных ВС каждый процессор — это композиция конвейеров с канонической структурой.

Мультиконвейерные ВС (PVP-systems) позволили достичь быстродействия, измеряемого в десятках GigaFLOPS.

Технико-экономические соображения и необходимость достижения высокой производительности и архитектурного совершенства обусловили переход Cray Research Inc. на платформу распределенных средств обработки информации, полностью основанных на модели коллектива вычислителей. Это дало возможность фирме создать ВС, обладающие быстродействием в диапазоне от GigaFLOPS до TeraFLOPS (см. MPP-системы: Cray T3D и Cray T3E).

Архитектурным совершенством начала XXI в. обладают сверхвысокопроизводительные ВС семейства Cray X. Данное семейство полностью основывается на модели коллектива вычислителей, оно в высшей степени впитало в себя архитектурные, функциональные и структурные достижения PVP- и MPP-систем.

Конвейерные ВС стартовали в 1970-х годах с производительностью порядка 100 MFLOPS (STAR-100, 1973 г.; Cray-1, 1976 г.); к 1990 г. они достигли быстродействия 1 GFLOPS (Cray-2, 1986 г. и Cray Y-MP, 1989 г.).

В 2007 г. производительность вычислительных систем, выпускаемых Cray Inc., оценивается в несколько десятков TFLOPS; в 2010 г. системы будут обладать быстродействием один квадриллион операций в секунду — 1 PFLOPS.

Диалектическое развитие архитектуры конвейерных ВС привело к следующим результатам:

1) конвейерные ВС «переродились» в распределенные мультипроцессорные системы с MIMD-архитектурой и массовым параллелизмом;

2) каноническая структура конвейера, которая была основой архитектуры суперЭВМ (конвейерных ВС 1970–1980-х годов), получила внедрение в микропроцессорных больших интегральных схемах (например, в микропроцессорах Intel Pentium, IBM Power PC, DEC Alpha, AMD Opteron, Intel Xeon);

3) современные «конвейерные» микропроцессоры по своей производительности превосходят векторные суперЭВМ 1970–1980-х годов.

Таким образом, ход развития архитектуры конвейерных вычислительных систем (канонический конвейер, первые ВС, PVP-системы, MPP-системы, сверхвысокопроизводительные ВС) показывает, что фирмы-разработчики за тридцатилетний период прошли путь от простейших конвейерных ВС (10^8 опер./с) до PFLOPS-систем, полностью основанных на модели коллектива вычислителей.

Мировой опыт работы в области конвейерных систем привел к необходимости создания распределенных ВС с программируемой структурой, концепция которых была сформулирована, теоретически и практически обоснована еще в 60-х и 70-х годах XX столетия в Сибирском отделении АН СССР.

5. МАТРИЧНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Матричные ВС обладают более широкими архитектурными возможностями, чем конвейерные ВС: их каноническая архитектура относится к классу SIMD. Матричные ВС — системы с массовым параллелизмом (Massively Parallel Processing Systems), следовательно, они не имеют принципиальных ограничений в наращивании своей производительности.

Матричные ВС предназначены для решения сложных задач, связанных с выполнением операций над векторами, матрицами и массивами данных (Data Arrays).

Работы по созданию матричных систем были начаты в 60-х годах XX столетия, первые высокопроизводительные (10^8 опер./с) реализации ВС появились в 1970-х годах. Современные ВС, которые завершают архитектурный ряд матричных систем, обладают быстродействием порядка 10^{12} опер./с.

В главе 5 рассмотрены каноническая структура матричного процессора и первые промышленные матричные ВС, освещено текущее состояние в указанной области и проанализированы архитектуры данного класса ВС.

5.1. Каноническая функциональная структура матричного процессора

Матричный, или векторный *процессор* (Array Processor) представляет собой «матрицу» связанных идентичных элементарных процессоров (ЭП), управляемых одним потоком команд (рис. 5.1). Каждый ЭП включает в себя АЛУ, память и локальный коммутатор. Сеть связей между ЭП (точнее, локальными коммутаторами) позволяет осуществлять обмен данными между любыми процессорами. Поток команд поступает на матрицу ЭП от единого устройства управления (SIMD-архитектура, в каноническом виде).

Архитектура матричного процессора была выбрана в начале 60-х годов XX в. и обоснована существовавшими экономическими ограничениями и необходимостью обеспечения высокой производительности при решении сложных задач. В самом деле, в то время основная доля стоимости ЭВМ приходилась на схемы устройства управления, а не на схемы АЛУ или

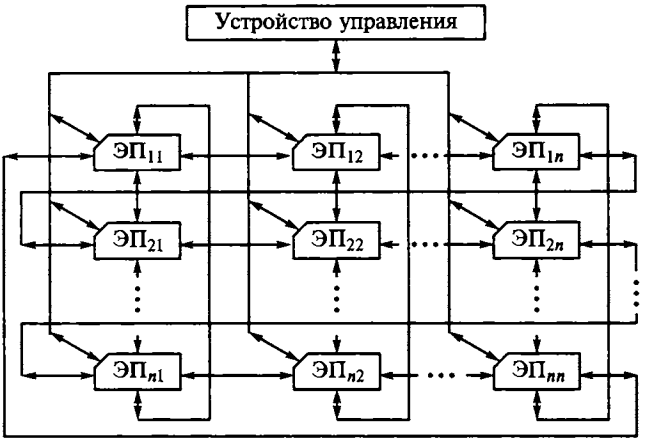


Рис. 5.1. Матричный процессор:
ЭП — элементарный процессор

памяти. Поэтому централизация устройства управления и параллелизм устройств обработки и хранения информации обеспечивали компромисс между стоимостью и производительностью ВС. С развитием интегральной технологии для производства электронных схем эти экономические соображения стали несущественными при выборе архитектуры вычислительного средства. Однако в конце XX в. при производстве БИС, включающих множества процессоров, матричная архитектура вновь стала технико-экономически привлекательной.

Существует широкий спектр сложных научных, технических и экономических задач, которые эффективно решаются на матричных процессорах. При решении сложных задач фактически один и тот же алгоритм параллельно (одновременно) реализуется над многими частями исходного массива данных. Ясно, что перед решением сложной задачи на матричном процессоре требуется предварительная организация данных в векторы или упорядоченные множества. Следовательно, в каждом ЭП размещаются и обрабатываются свои компоненты векторов или свои локальные массивы данных.

Итак, параллелизм в работе элементарных процессоров принципиально позволяет достичь любого уровня быстродействия матричного процессора.

Безусловно, матричные процессоры ориентированы на работу в монопрограммном режиме (когда решается только одна задача, представленная в параллельной форме). Однако такие процессоры можно использовать и в мультипрограммных режимах (когда решается несколько задач или выполняется несколько параллельных программ с различным числом ветвей в общем случае). Реализация мультипрограммных режимов в матричном процессоре осуществляется за счет разделения и «времени», и «пространства».

В самом деле, в матричном процессоре имеется единственное устройство управления и множество ЭП, следовательно, в мультипрограммной ситуации должны «делиться» время первого и элементарные процессоры («пространство») между программами.

Первая матричная ВС SOLOMON (Simultaneous Operation Linked Ordinal MOdular Network — вычислительная сеть синхронно функционирующих упорядоченных модулей) была разработана в Иллинойском университете (University of Illinois) США под руководством Даниэль Слотника (Daniel L. Slotnick). Планировалось, что она [13] будет иметь матрицу из 32×32 элементарных процессоров, способную выполнять операции над словами с переменной разрядностью от 1 до 128 разрядов. Каждый ЭП должен был иметь в своем составе АЛУ с последовательной поразрядной обработкой и память емкостью 16 К бит. Все ЭП в любой момент времени могли выполнять только одну и ту же операцию над числами, хранящимися в их ячейках памяти (с одними и теми же адресами). При этом каждый ЭП мог находиться либо в активном состоянии и выполнять команды, поступающие из устройства управления, либо в пассивном состоянии и не реагировать на эти команды.

Устройством управления в системе SOLOMON могла служить серийно выпускаемая ЭВМ. Эта машина должна была иметь память для хранения программ и осуществлять связь с внешними устройствами.

Работы по проекту SOLOMON велись с 1962 г., однако этот проект промышленного воплощения не нашел; в 1963 г. был создан лишь макет ВС размером 3×3 элементарных процессора. Позднее была построена конфигурация ВС размером 10×10 ЭП в фирме Westinghouse Electric Corp.

5.2. Вычислительная система ILLIAC IV

Матричная ВС ILLIAC IV создана Иллинойским университетом и корпорацией Бэрроуз (Burroughs Corporation). Работы по созданию ILLIAC IV были начаты в 1966 г. под руководством Д.Л. Слотника. Монтаж системы был закончен в мае 1972 г. в лаборатории фирмы Burroughs (Паоли, штат Панама), а установка для эксплуатации осуществлена в октябре 1972 г. в Научно-исследовательском центре НАСА им. Эймса в штате Калифорния (NASA's Ames Research Center; NASA — National Aeronautics and Space Administration — Национальное управление аэронавтики и космоса). Количество процессоров в системе — 64; быстродействие — $2 \cdot 10^8$ опер./с; емкость оперативной памяти — 1 Мбайт; полезное время составляло 80...85 % общего времени работы ILLIAC IV, стоимость $4 \cdot 10^7$ долл., вес — 75 т, занимаемая площадь — 930 м^2

Система ILLIAC IV была включена в вычислительную сеть ARPA (Advanced Research Projects Agency — Управление перспективных исследований и разработок Министерства обороны США) и успешно эксплуатировалась до 1981 г.

Публикация о проекте SOLOMON* появилась примерно через шесть месяцев после первой печатной работы [14] по однородным ВС, изданной в Сибирском отделении АН СССР; вариант ILLIAC IV** реализован через шесть лет после создания советской системы «Минск-222» [5].

5.2.1. Функциональная структура системы ILLIAC IV

Матричная ВС ILLIAC IV (рис. 5.2) должна была состоять из четырех квадрантов (K_1–K_4), подсистемы ввода-вывода информации, ведущей ВС В 6700 (или В 6500), дисковой памяти (ДП) и архивной памяти (АП). Планировалось, что ВС обеспечит быстродействие 10^9 опер./с. В реализованном варианте ILLIAC IV содержался только один квадрант, что обеспечило быстродействие 2 10^8 опер./с. При этом ILLIAC IV оставалась самой быстродействующей ВС вплоть до 80-х годов XX в.

Квадрант — матричный процессор, включавший в себя устройство управления и 64 ЭП. Устройство управления представляло собой специализированную ЭВМ, которая использовалась для выполнения операций над скалярами и формировала поток команд на матрицу ЭП. Элементарные про-

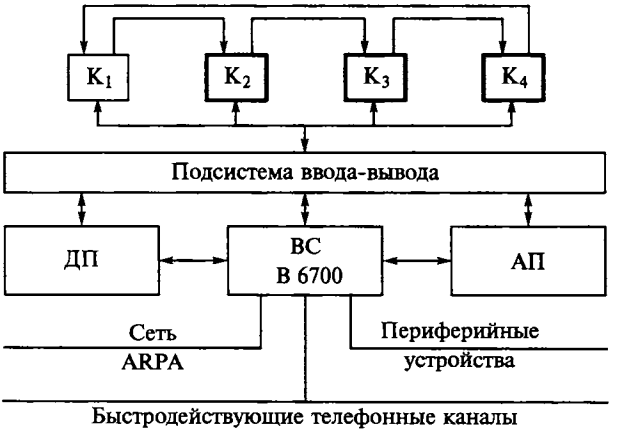


Рис. 5.2. Функциональная структура системы ILLIAC IV:
К — квадрант; ДП — дисковая память; АП — архивная память

* *Slotnick D.L. e.a.* The SOLOMON Computer//Proc. EJCC. 1962. P. 97–107.
** *Слотник Д.* и др. Система ИЛЛИАК IV//ТИИЭР, 1972. Т. 60. № 4. С. 36–62.

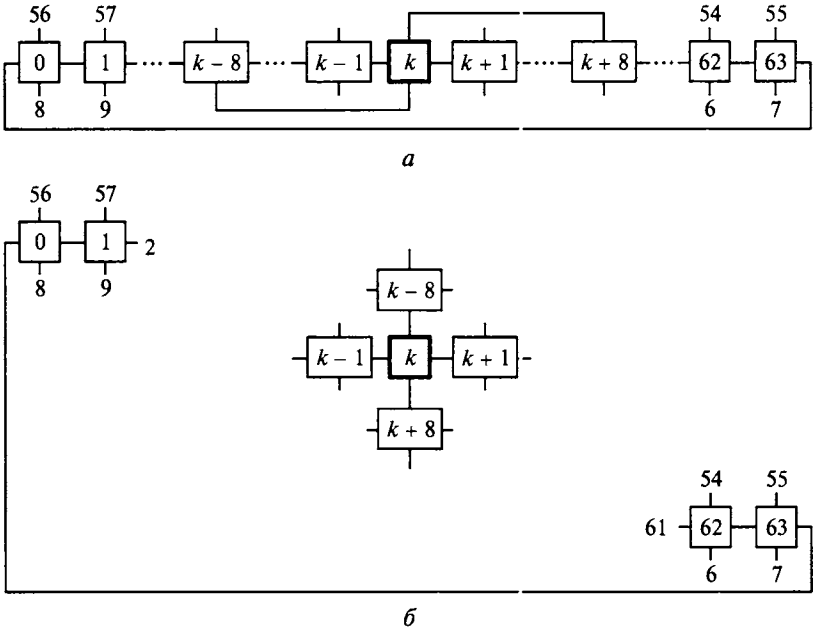


Рис. 5.3. Варианты изображения структуры квадранта ILLIAC IV

цессоры матрицы регулярным образом были связаны друг с другом. Структура квадранта системы ILLIAC IV представлялась двумерной решеткой, в которой граничные ЭП были связаны по канонической схеме (см. рис. 5.1). Позднее подобные структуры стали называться D_n-графами (термин предложен в Отделе вычислительных систем СО АН СССР [5]), и еще позже, в 90-х годах XX в., — циркулянтными графами. Структура квадранта ВС ILLIAC IV представляла собой D_2-граф вида {64; 1, 8} (см. разд. 3.1.2). Следовательно, в этой структуре, например, ЭП с номером 0 был связан с ЭП, имевшими номера 1, 8, 56, 63 (рис. 5.3).

Матрица из 64 ЭП предназначалась для выполнения операций над векторами. Все 64 ЭП работали синхронно и единообразно. Допускалось одновременное выполнение скалярных и векторных операций. Таким образом, в ILLIAC IV была заложена возможность параллельной работы не только ЭП в матрице, но и устройства управления и матрицы в целом.

В системе ILLIAC IV использовалось слово длиной 64 двоичных разряда. Числа могли представляться в следующих форматах: 64 или 32 разряда с плавающей запятой, или 48, или 24, или 8 разрядов с фиксированной запятой. При использовании 64-, 32- и 8-разрядных форматов матрица из 64 ЭП могла обрабатывать векторы операндов, состоявшие из 64, 128 и 512 компонентов. Система ILLIAC IV при суммировании 512- и 8-разрядных чисел

имела быстродействие почти 10^{10} опер./с, а при сложении 64-разрядных чисел с плавающей запятой — $1{,}5 \cdot 10^8$ опер./с.

Каждый ЭП имел накапливающий сумматор, регистр второго операнда, регистр передаваемой информации (из данного ЭП в соседний ЭП), регистр, использовавшийся как временная память, регистр модификации адресного поля команды, регистр состояния данного ЭП. Элементарный процессор мог находиться в одном из двух состояний — активном или пассивном. В первом состоянии ему разрешалось, а во втором запрещалось выполнять команды, поступавшие из устройства управления. Состояние ЭП задавалось при помощи специальных команд. Среди них, например, была команда, устанавливавшая в пассивное состояние все ЭП, у которых передаваемая информация была больше содержимого накапливающего сумматора. Накапливающий сумматор и все регистры ЭП были программно адресуемы. Они имели 64 разряда, кроме регистров модификации адреса и состояния, которые состояли из 16 и 8 разрядов соответственно.

Память каждого ЭП обладала емкостью 2048 64-разрядных слов, была реализована на биполярных интегральных схемах и время цикла составляло 300 нс. К каждой памяти непосредственный доступ имел собственный ЭП. Обмен информацией между памятями различных ЭП осуществляется по сети связи при помощи специальных команд пересылок.

Подсистема ввода-вывода состояла из устройства управления, буферного запоминающего устройства и коммутатора. Комплекс этих устройств обеспечивал обмен информацией между квадрантами ILLIAC IV и средствами ввода-вывода: ЭВМ B 6700, дисковой и архивной памятью, периферийными устройствами, сетью ARPA.

Ведущая ВС B 6700 — мультипроцессорная система корпорации Burroughs; могла иметь в своем составе от 1 до 3 центральных процессоров и от 1 до 3 процессоров ввода-вывода информации и обладала быстродействием $(1\ldots3) \cdot 10^6$ опер./с. Она использовалась для реализации функций операционной системы (включая ввод-вывод информации, операции по компиляции и компоновке программ, распределение аппаратных ресурсов, исполнение служебных программ, например таких, как программы перезаписи «перфокарты — диски», «перфокарты — ленты» и т. п.). По сути, система B 6700 являлась быстродействующим периферийным устройством ILLIAC IV.

Дисковая память (*ДП*) состояла из двух дисков и обрамляющих электронных схем, она имела емкость порядка 10^9 бит и была снабжена двумя каналами, по каждому из которых можно было параллельно передавать и принимать информацию со скоростью $0{,}5 \cdot 10^9$ бит/с. Среднее время обращения к диску составляло 20 мс.

Архивная память (*АП*) — постоянная лазерная память с однократной записью, разработанная фирмой Precision Instrument Company емкостью

10^{12} бит. Запись двоичных данных осуществлялась аргоновым лазером путем прожигания микроскопических отверстий в металлической пленке, нанесенной на полоски полиэфирной подложки. На каждой полоске можно было записать около 2,9 млрд бит. Имелось 400 информационных полосок, которые размещались на вращающемся барабане. Время поиска данных на любой из 400 полосок достигало 5 с; время поиска в пределах полоски — 200 нс. Существовало два канала обращения к архивной памяти, скорость считывания и записи данных по каждому из которых была равна $4 \cdot 10^6$ бит/с.

В системе ILLIAC IV насчитывалось более $6 \cdot 10^6$ электронных компонентов. Отказы компонентов или соединений могли происходить через несколько часов. По этой причине в систему была включена обширная библиотека контрольных и диагностических тестов, а также предусмотрена возможность оперативной (но не автоматической) замены отказавших ЭП исправными ЭП резерва. Отключенный неисправный ЭП восстанавливался при помощи диагностической ЭВМ.

5.2.2. Программное обеспечение ВС ILLIAC IV

Главная цель разработки ILLIAC IV — создание мощной ВС для решения задач с большим числом операций (см. разд. 3.34). Программы таких задач были структурно единообразны, они содержали три части.

1. «Предпроцессорная» часть обеспечивала инициирование задачи и десятично-двоичные преобразования. Эта часть обычно имела последовательную форму представления.

2. «Ядро» осуществляло собственно решение задачи и представлялось в параллельной форме. Размер ядра составлял 5...10 % полного объема программы, но его исполнение на последовательной машине требовало 80...95 % рабочего времени.

3. «Постпроцессорная» часть осуществляла запись результатов в архивные файлы, двоично-десятичные преобразования, вычерчивание графиков, вывод результатов на печать и т. п. Эта часть, как правило, имела последовательную форму представления. Из «постпроцессорной» части управление могло быть передано в «ядро» для выполнения последующих итераций.

Главная цель создания ILLIAC IV и предопределила возможности программного обеспечения: операционной системы и средств программирования.

Операционная система ILLIAC IV состояла из набора асинхронных программ, выполнявшихся под управлением главной управляющей программы B 6700. Она работала в двух режимах: в первом режиме выполнялся контроль и диагностика неисправностей в квадранте и в подсистеме ввода-

вывода информации; во втором — осуществлялось управление работой ILLIAC IV при поступлении на B 6700 заданий от пользователей. Задание для ILLIAC IV обычно состояло из перечисленных ниже составных частей.

1. Программы B 6700, написанные, как правило, на версиях языков ALGOL или FORTRAN и осуществлявшие подготовку (и преобразование) входных двоичных файлов. Эти программы представляли собой «предпроцессорную» часть программы решаемой задачи.

2. Программы ILLIAC IV, обычно написанные на языках Glynpir или FORTRAN, которые использовались ILLIAC IV (составляли «ядро») для обработки файлов, подготовленных программами B 6700, а также для формирования двоичных выходных файлов.

3. Программы B 6700 (на версиях языков ALGOL или FORTRAN), которые преобразовывали двоичные файлы ILLIAC IV в требуемый выходной формат. Они составляли «постпроцессорную» часть программы решаемой задачи.

4. Программа на управляющем языке Illiac, определявшая задание. Эта программа ориентировала операционную систему на работу, предусмотренную заданием. (Допускалось использование операционной системы для одновременного выполнения нескольких программ на управляющем языке.)

Средства программирования ILLIAC IV включали язык ассемблера (Assembler Language) и три языка высокого уровня: Tranquil, Glynpir, FORTRAN.

Язык ассемблера ILLIAC IV — традиционный язык программирования, адаптированный под архитектуру ВС. В частности, он имел сложные макроопределения, которые можно было применять для включения стандартных операций ввода-вывода и других операций связи между программами B 6700 и ILLIAC IV. Кроме того, в языке ассемблера были предусмотрены псевдооперации, использовавшиеся для распределения памяти и элементарных процессоров.

Языки высокого уровня благодаря архитектурным особенностям ILLIAC IV отличались от соответствующих языков ЭВМ.

1. Распределение двумерной памяти. Была разрешена адресация отдельных слов в памяти ЭП и строк (из 64 слов) в пределах запоминающих устройств матрицы ЭП. Адресация по «столбцу» группы слов в памяти одного ЭП была недопустима.

2. Параллелизм и управление режимом обработки. Параллелизм ВС предопределяет работу с векторами данных. Следовательно, языки ILLIAC IV должны были допускать операции над векторами данных или строками матриц. Размерность вектора и количество подлежащих обработке элементов вектора определялись словами режима. Языки ILLIAC IV обеспечива-

ли эффективную реализацию широкого круга вычислений и обработку слов режима.

3. Вид команд пересылок и индексации. Каждый из языков ILLIAC IV должен был содержать команды пересылок и индексации с различными приращениями в каждом элементарном процессоре.

Tranquil подобен языку ALGOL и полностью не зависел от архитектуры ILLIAC IV. Он был разработан для обеспечения параллельной обработки массивов информации. Компилятор языка Tranquil потребовал больших системных затрат, связанных с маскированием архитектуры ILLIAC IV. Поэтому работы по созданию компилятора были приостановлены и предприняты шаги по модификации языка Tranquil.

Glynpir являлся языком также алгольного типа с блочной структурой. Он позволял опытному программисту использовать значительные возможности архитектуры ВС ILLIAC IV.

Язык FORTRAN был разработан для рядовых пользователей ILLIAC IV. Он в отличие от Glynpir освобождал пользователя от детального распределения памяти и предоставлял ему возможность мыслить в терминах строк любой длины. Он позволял путем применения модифицированных операторов ввода-вывода воспользоваться параллельной программой как последовательной и выполнить ее на одном ЭП. Иными словами, FORTRAN давал возможность отлаживать параллельные программы на одном элементарном процессоре. В языке FORTRAN нашли отражение многие операционные свойства языков Tranquil и Glynpir. В результате длительной эксплуатации системы ILLIAC IV установлена высокая эффективность транслятора с языка FORTRAN на Glynpir.

5.2.3. Применение ВС ILLIAC IV

Вычислительная система ILLIAC IV входила в состав сети ARPA. Она являлась средством решения сложных задач. В самом деле, бóльшую часть численных алгоритмов, разработанных для ЭВМ, нельзя было с помощью незначительных изменений превратить в эффективные схемы параллельных вычислений. При построении параллельных алгоритмов необходимо было учитывать, что ВС ILLIAC IV управлялась одним потоком команд, т. е. структура системы позволяла одновременно выполнять одинаковые операции над 64 множествами данных, записанных в памяти различных ЭП. Следовательно, эффективные параллельные алгоритмы могли быть построены только для тех сложных задач, решение которых было связано с многократными вычислениями одной и той же функции для различных значений аргументов.

В качестве примера рассмотрим умножение матриц **A** и **X** большого размера. Элементы z_{ik}, $i, k = \overline{1, n}$, матрицы $\mathbf{Z} = \mathbf{A} \cdot \mathbf{X}$ определяются выражением

$$z_{ik} = \sum_{j=1}^{n} a_{ij} x_{jk},$$

где a_{ij}, x_{jk} — элементы строки i матрицы **A** и столбца k матрицы **X** соответственно. Ясно, что это требует вычисления функций

$$f_k(y, x) = \sum_{j=1}^{n} y_j x_{jk}, \quad k = \overline{1, n},$$

в которых аргумент $(y_1, y_2, \ldots, y_n)$ последовательно заменяется на $(a_{i1}, a_{i2}, \ldots, a_{in})$ при $i = 1, 2, \ldots, n$.

Практически установлено, что ILLIAC IV была эффективна при решении широкого спектра сложных задач. Назовем классы задач, которые допускают построение параллельных алгоритмов с высокой степенью использования ЭП: матричная арифметика, системы линейных алгебраических уравнений, линейное программирование, исчисление конечных разностей в одномерных, двумерных и трехмерных случаях, квадратуры (включая быстрое преобразование Фурье), обработка сигналов. Отметим классы задач, обеспечивающих неполное использование ЭП: движение частиц (метод Монте-Карло и т. д.), несимметричные задачи на собственные значения, нелинейные уравнения, отыскание корней полиномов (в некоторых случаях эти задачи попадают в первую группу). Эффективность системы ILLIAC IV, например, характеризует время решения типичной задачи линейного программирования, имеющей 4000 ограничений и 10 000 переменных. Это время для ILLIAC IV составляло менее 2 мин, а для большой ЭВМ третьего поколения — 6...8 ч. Названные задачи говорят о большой сфере применения матричных ВС (наука, экономика, проектирование, атомная энергетика, предсказание погоды, радиолокация, оборона, космические исследования и т. д.).

Технические и программные средства ILLIAC IV многократно подвергались модификациям, что улучшало технико-экономические показатели, увеличивало вычислительную мощность системы, расширяло сферу применения системы ILLIAC IV.

5.3. Вычислительная система DAP

Разработку матричной ВС DAP (Distributed Array Processor — распределенный матричный процессор) осуществляла английская фирма ICL (International Computers Ltd.). Работы были начаты в 1972 г., опытные образцы системы DAP (на дискретных элементах) из 1024 и 4096 элементарных процессоров были построены в 1976 г. и 1977 г. Планировалось к 1980 г. организовать серийное производство ВС DAP на основе больших интегральных схем, однако из-за финансовых трудностей работы были прекращены.

Система DAP по своей архитектуре относилась к SIMD-типу, это была ВС с массовым параллелизмом. Планировалось, что ВС будет состоять из 50 000 параллельно работающих ЭП, управляемых одним потоком команд. Каждый ЭП будет представлять собой монолитную большую интегральную схему, подсистема ввода-вывода информации будет выполнена также на БИС. Предполагалось аппаратно реализовать многие функции программного обеспечения.

Функциональная структура ВС DAP [13] — это композиция ведущей ВС и собственно DAP (рис. 5.4). Ведущая ВС (Host Computer) предназначалась для реализации функций операционной системы (включая подготовку данных и команд для DAP, распределение данных по ЭП). В опытных образцах ВС DAP в качестве ведущей была использована серийная ВС Control

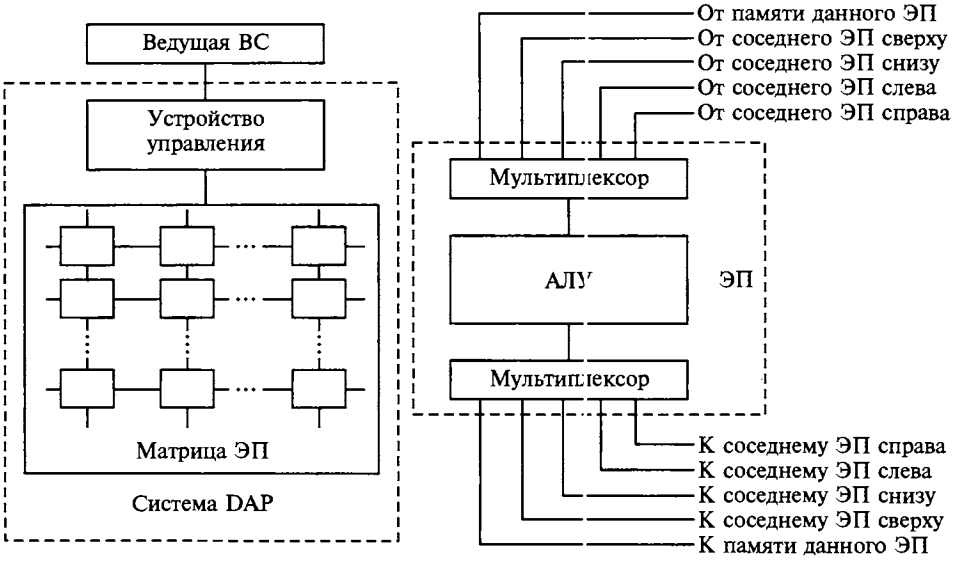


Рис. 5.4. Функциональная структура ВС DAP:

ЭП — элементарный процессор; АЛУ — арифметико-логическое устройство

Data Corporation: конвейерная система CDC 7600 (восемь конвейеров, длительность такта — 27,5 нс, быстродействие 10...15 млн опер./с). Собственно матричная система DAP применялась для массовых параллельных вычислений.

Устройство управления формировало поток команд на матрицу ЭП, в частности оно направляло команды, адреса и другую информацию, необходимую элементарным процессорам для выполнения «матричных» операций.

Архитектуры систем ILLIAC IV и DAP на макроуровне близки, однако DAP имела свои особенности. Каждый элементарный процессор DAP представлял собой одноразрядный микропроцессор, связанный с локальной памятью емкостью в 4096 бит. Матрица ЭП и их память для удобства представлялись в виде «прямоугольного вычислительного параллелепипеда» из 4097 горизонтальных слоев. Верхний слой параллелепипеда — это матрица ЭП, в которой строка имела длину Ld элементов (d — длина слова ведущей ВС, L — число слов ведущей ВС в строке), а столбец состоял из 2^m элементов (m — некоторое выбранное целое положительное число). Таким образом, матрица ЭП имела размеры $Ld \cdot 2^m$, следовательно, она была способна параллельно обрабатывать $L \cdot 2^m$ d-разрядных слов. Под каждым ЭП в параллелепипеде располагалась его локальная память — вертикальная колонка из 4096 разрядов. Локальная память всех ЭП составляла распределенную память системы DAP в целом. Каждый слой разрядов этой распределенной памяти был пронумерован от 0 до 4095 и был рассчитан на $L \cdot 2^m$ d-разрядных слов. Функции по распределению информации из ведущей ВС (по вложению слов ведущей ВС в 4096-разрядную память ЭП и наоборот) выполнялись устройством управления DAP.

Между ЭП существовала сеть связей. Она обеспечивала через мультиплексоры связь каждого ЭП с регистрами АЛУ четырех ближайших соседей, расположенных сверху, снизу, слева и справа от него. Следовательно, сеть связей обеспечивала архитектурную гибкость ВС DAP. Она, в частности, превращала распределенную память в общедоступную для каждого ЭП, позволяла работать с произвольными L-элементными векторами и матрицами размером $L \cdot 2^m$ элементов, допускала варьирование разрядности d элементов.

Каждый ЭП имел в своем составе предельно простое одноразрядное АЛУ, которое обеспечивало последовательную поразрядную обработку информации (элементов векторов и матриц). Предусматривалась реализация арифметических и логических операций, при этом результат операции мог быть отослан из соответствующего регистра АЛУ в память данного ЭП или в регистр одного из четырех соседних ЭП (через «нижний» мультиплексор, см. рис 5.4).

В каждом ЭП был заложен механизм, позволявший программировать состояние процессора. Этот механизм был предельно прост и представлял собой одноразрядный регистр активности, одно из состояний которого (пассивное состояние) запрещало ЭП выполнять поступившую команду. Следовательно, регистры активности использовались для маскирования частей матриц ЭП или отдельных ЭП в зависимости от условий выполнения вычислений.

Команды для матрицы ЭП DAP поступали из устройства управления. Они содержали всю информацию, которая требовалась для выполнения операции (адреса для операндов и результата и др.).

Найдем «оценку сверху» быстродействия ВС DAP. При выполнении арифметических операций разряды операндов должны были обрабатываться отдельно. Поэтому, например, для выполнения сложения d-разрядных чисел с фиксированной запятой требовалось d раз повторить цикл из трех команд DAP. Время цикла работы DAP составляло 200 нс. Следовательно, на выполнение указанной операции на одном ЭП затрачивается $3d \cdot 200$ нс; при $d = 20$ эта величина равна 12 мкс. Ясно, что матрица из N ЭП была способна выполнить в 1 с до $N/3d \cdot 200$ таких операций сложения над d-разрядными операндами; при $d = 20$ максимальное быстродействие матрицы из 4096 ЭП составляло около 340 млн опер./с.

Следует подчеркнуть, что в ВС DAP (в отличие от ILLIAC IV) в каждом ЭП использовалось не параллельное, а очень простое и дешевое одноразрядное АЛУ. Следовательно, любые операции над многоразрядными операндами должны были выполняться в ЭП при помощи программ. (Это имело место и при организации DAP в виде прямоугольного вычислительного параллелепипеда, который был описан выше.) Последовательные ЭП в отдельности имели низкое быстродействие, однако матрица из большого числа параллельно работавших ЭП с программно управляемыми связями между ними обеспечивала высокую производительность. Такой подход был очень гибким и с точки зрения вычислительных технологий, в частности, он позволял пользователю выбирать точность вычислений. При этом подход обеспечил: во-первых, экономное расходование памяти (операнды занимали только столько разрядов, сколько их требовалось для достижения заданной точности представления чисел); во-вторых, минимум времени выполнения любой арифметической операции (так как соответствующая ей последовательность команд выполнялась только над таким количеством разрядов операндов, которое соответствовало требуемой точности).

Архитектурной особенностью ВС DAP являлось и то, что ее распределенная память могла быть использована ведущей ВС CDC 7600 как обычная память.

В 1976 г. создан опытный образец DAP, который имел матрицу из 32×32 ЭП, каждый из которых был оснащен локальной памятью 1 К бит.

Позднее эта конфигурация была расширена до 64 × 64 ЭП с емкостью локальной памяти каждого в размере 4 К бит. Последняя конфигурация ВС DAP превосходила по производительности в 5–10 раз старшие модели ЭВМ третьего поколения (ЭВМ IBM 360/195 обладала быстродействием $(7...10) \cdot 10^6$ опер./с и емкостью оперативной памяти 1024...4096 К байт). Интегральные (на основе БИС) реализации ВС DAP не были созданы из-за финансовых трудностей.

Вычислительная система DAP была ориентирована на решение задач, связанных с предсказанием погоды. Планировалось установить системы в двух организациях: European Centre for Medium Term Weather Forecasting; Meteorological Office in Bracknell. Очевидно, что данная ВС могла найти ряд других областей применения.

5.4. Семейство вычислительных систем Connection Machine

Эволюция архитектуры матричных ВС и достижения в технологии БИС привели к созданию ВС с массовым параллелизмом, архитектура которых не может быть вписана в какой-либо один из канонов. Архитектура данных систем в зависимости от «глубины» просмотра может быть отнесена к классам MIMD и SIMD одновременно. Например, система в целом может иметь архитектуру MIMD, а ее основные процессорные компоненты — SIMD; сети связей между элементами обработки информации на различных иерархических уровнях могут быть также различными. К таким ВС относятся модели семейства Connection Machine: CM-1, CM-2 и CM-5.

Замысел создания ВС Connection Machine* возник еще в конце 1970-х годов в Лаборатории искусственного интеллекта Массачусетсского технологического института США (Massachusetts Institute of Technology's Artificial Intelligence Laboratory, MIT AI Laboratory) и принадлежит У.Д. Хиллису (W.D. Hillis). В начале 1980-х годов были разработаны архитектура и схемы для создания прототипа этой системы.

Актуальность построения системы привела к основанию в 1983 г. Thinking Machines Corp. (США). Эта корпорация выполнила разработку всех моделей семейства Connection Machine: CM-1 (1986), CM-2 (1987), CM-5 (1991).

Отметим архитектурные особенности систем семейства CM:

- превалирующий класс архитектуры — SIMD (MIMD, любой из моделей в целом);

---

* *Hillis W.D.* The connection machine // Scient. American, 1987. V. 156. N 6. P. 108–115.

- массовый параллелизм (MPP — Massively Parallel Processing);
- максимальное число ЭП — 65 536 (или 2^{16});
- быстродействие — до 1 TFLOPS;
- однородность и программируемость структуры сети межпроцессорных связей;
- масштабируемость ВС (возможность создания конфигураций в СМ-1 из 16 К, 32 К, 48 К и 64 К элементарных процессоров, в СМ-2 — из 32, 64, 128, ..., 32 К, 64 К процессоров, а в СМ-5 — из произвольного числа ЭП, не превышающего 16 К).

Thinking Machines Corp. была одной из лидирующих компаний XX в. по разработке MPP-суперкомпьютеров. Начиная с 1996 г. она сконцентрировала свои усилия только на разработке ПО для ВС с массовым параллелизмом (конструируемых другими компаниями). Выпуск моделей семейства Connection Machine прекращен. Однако архитектурные решения, заложенные в это семейство, представляют и сейчас большой интерес для создателей высокопроизводительных средств ВТ. Эти решения могут составить основу при конструировании СБИС, содержащих коллективы связных элементарных процессоров.

5.4.1. Вычислительная система СМ-1

Вычислительная система СМ-1 — первая модель семейства Connection Machine — была спроектирована в Thinking Machines Corp. в течение 1983 г. и первой половины 1984 г. Прототип модели СМ-1 из 16 К процессоров был построен к концу 1984 г. при финансовой поддержке Агентства по перспективному планированию исследований в области обороны США (Defense Advanced Research Projects Agency). Демонстрация возможностей прототипа СМ-1 была осуществлена в мае 1985 г. Полная 65 536-процессорная конфигурация СМ-1 была собрана и успешно продемонстрирована в ноябре 1985 г.

Максимальная конфигурация модели СМ-1 (из 64 К процессоров) имела быстродействие 2000 MIPS ($2 \cdot 10^9$ опер./с над 32-разрядными целыми числами) и обладала распределенной оперативной памятью емкостью 32 Мбайт.

Эти данные говорят о простоте проектирования и изготовления высокопроизводительных MPP-систем.

Функциональная структура системы СМ-1. Модель СМ-1 семейства Connection Machine имеет достаточно развитую функциональную структуру, характеризуется иерархией средств управления процессами обработки информации (рис. 5.5). В состав ВС СМ-1 входят: параллельное процессорное устройство (Parallel Processor Unit); четыре сервисных процессора ($СП_0$–$СП_3$, Front-ends) с интерфейсом шин (ИШ); коммутатор (Nexus).

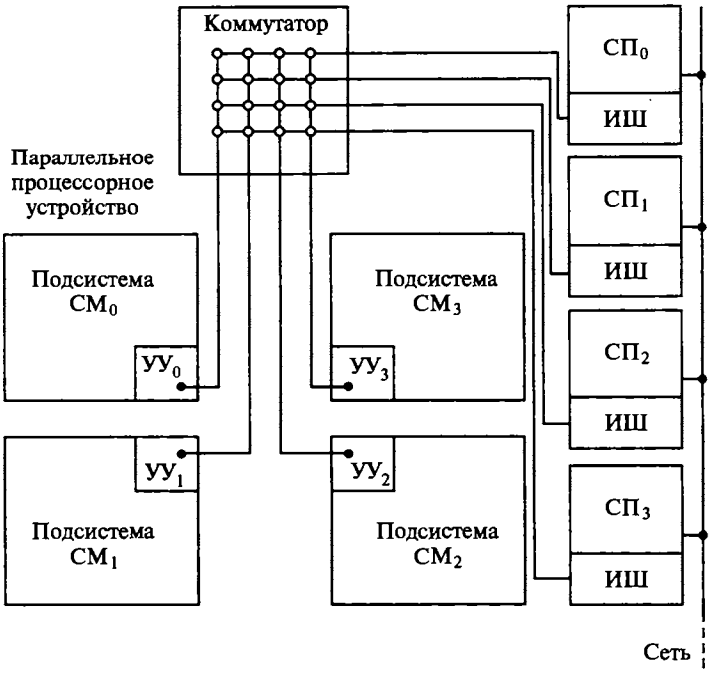


Рис. 5.5. Функциональная структура ВС СМ-1:
СП — сервисный процессор; ИШ — интерфейс шин; УУ — устройство управления

Основу любой конфигурации СМ-1 в целом составляет *параллельное процессорное устройство* с архитектурой MIMD, которое может иметь в своем составе от одной до четырех подсистем: CM_0–CM_3. Архитектура подсистем CM_0–CM_3 относится к классу SIMD. Следовательно, в пределах каждой из подсистем данные распределяются по процессорам и одна и та же программа управляет работой множества процессоров (но каждого над своим подмножеством данных). Любая из подсистем CM_0–CM_3 является композицией из устройства управления (УУ) и 16 384 ЭП. Таким образом, максимальная конфигурация модели СМ-1 имеет 65 536 ЭП.

Устройство управления (Sequencer) ВС СМ-1 — специально спроектированный микрокомпьютер для реализации функций виртуальной машины (архитектура которой существенно удобнее для пользователя, чем у реальной физической ВС). Это устройство содержит память нанокоманд емкостью 16 К 96-разрядных слов. На входы четырех устройств управления поступает поток информации «высокого уровня», а именно — операций виртуальной машины и аргументов. Этот поток поступает из коммутатора по синхронному параллельному (32-разрядному) каналу данных. На выходе УУ имеет место поток нанокоманд, которые и управляют работой элементарных процессоров и памяти.

Все элементарные процессоры ВС СМ сгруппированы в *вычислительные узлы* (или *вершины*) по 16 ЭП. Каждый узел конструктивно оформлен как объединение процессорного кристалла и кристалла памяти. Такой узел в целом называют просто *процессорным кристаллом*. В каждой из подсистем $СМ_0$–$СМ_3$ имеется 1024 узла. Взаимодействие между узлами осуществляется через сеть связей, структура которой представляет собой 10-мерный гиперкуб.

Сервисные процессоры (Front-ends), по сути, составляют аппаратурно-программную среду для разработки системного ПО. Они выполняют также функции ведущих (Host) процессоров и обеспечивают взаимодействие с сетью ЭВМ (Network). В качестве такого процессора могут быть использованы серийно производимые средства (например, система DEC VAX). *Интерфейс шин* (Bus Interfase) поддерживает 32-разрядный параллельный асинхронный канал между сервисными процессорами и коммутатором.

Коммутатор (Nexus) предназначается для организации взаимодействий между сервисными процессорами и устройствами управления, он имеет размер 4×4 (4×4 Cross-point Switch). Коммутатор реализует механизм разделения, который позволяет в пределах ВС СМ-1 конфигурировать до четырех подсистем, работающих под управлением своего сервисного процессора. Это дает возможность применять различные подсистемы для решения задач разных пользователей или, например, выделять одну из них для диагностирования и восстановления работоспособности, а остальные использовать для решения задач. В случае, если к одному сервисному процессору подсоединяется более одного устройства управления, осуществляется их синхронизация (при помощи генератора синхроимпульсов коммутатора).

Элементарные процессоры и вычислительные узлы ВС СМ-1. Элементарный процессор — основной функциональный элемент системы СМ-1 (см. рис. 5.5). Он имеет архитектуру SISD и является одноразрядным последовательным средством обработки информации. В состав каждого ЭП (рис. 5.6) входят:

- одноразрядное АЛУ;
- битно-адресуемая локальная память (ЛП) емкостью 4 К бит;
- восемь одноразрядных регистров признаков (РП) или флагов;
- интерфейс маршрутизатора (ИМ);
- двумерный интерфейс сети межпроцессорных связей (ИСМС).

Элементарный процессор ВС СМ-1 не является конструктивно оформленным элементом. В качестве конструктивной (да и функциональной) единицы выступает вершина или вычислительный узел (см. рис. 5.6). При реализации узла используются два типа кристаллов. Первый — это

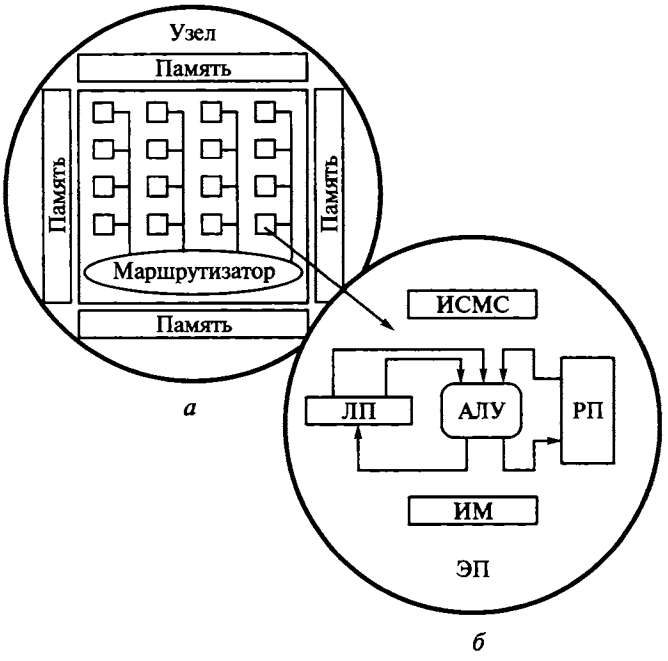


Рис. 5.6. Функциональные структуры ЭП и узла ВС СМ-1:

а — ВУ; *б* — ЭП; ИСМС — интерфейс сети межпроцессорных связей; ЛП — локальная память; АЛУ — арифметико-логическое устройство; РП — регистр признаков; ИМ — интерфейс маршрутизатора; ЭП — элементарный процессор

оригинальный, специально спроектированный (заказной) кристалл (Proprietary Custom Chip). Этот кристалл называют процессорным, он содержит АЛУ, регистры признаков и коммуникационный интерфейс для 16 ЭП (маршрутизатор и средства межпроцессорной сети связей). Второй кристалл — коммерческая статическая память с произвольным доступом и с защитой по четности. Таким образом, в полной конфигурации ВС СМ-1 (из 64 К ЭП) содержится 4096 процессорных кристаллов (узлов) и имеется память с произвольной выборкой емкостью 65 536 v бит (v — емкость локальной памяти одного ЭП).

Арифметико-логическое устройство ЭП имеет три входа и два выхода и включает в себя логические элементы, одноразрядные регистры-защелки (Latches) и интерфейс памяти (см. рис. 5.6). Опишем цикл функционирования АЛУ. Вначале осуществляется чтение двух одноразрядных операндов из памяти и одного бита данных из регистра признака. Затем логические элементы выполняют операцию над этими тремя входными битами и выдают два одноразрядных результата. Наконец, один из двух результатов запоминается в памяти, а другой — в регистре признака. Логические элементы АЛУ способны

вычислять значения любых булевых функций от трех переменных. Арифметические операции выполняются последовательно, побитно.

Элементарный процессор системы СМ-1 относится к классу RISC (RISC — Reduced Instruction Set Computer — компьютер с упрощенным набором команд). Длительность цикла ЭП (АЛУ) составляет 0,75 мкс. Цикл ЭП подразделяется на подциклы. В каждом подцикле ЭП выполняет команду низкого уровня, называемую нанокомандой (Nanoinstruction). Нанокоманды поступают в ЭП из устройства управления. Одновременно с реализацией нанокоманды в памяти ЭП может выполняться одна из операций: чтение или запись. Выполнение целочисленной операции сложения двух операндов требует трех подциклов, соответствующих следующим нанокомандам: LoadA — чтение из памяти операнда A, LoadB — чтение из памяти операнда B и Store — выполнение операции и запоминание результата. Ясно, что время сложения L-разрядных операндов в ЭП потребует $3L\tau$ единиц времени (τ — длительность подцикла, τ = 0,375 мкс). Тогда быстродействие ВС СМ-1 из 64 К ЭП составит $65\,536/3L\tau$ целочисленных операций в единицу времени. Например, при сложении 32-разрядных чисел быстродействие СМ-1 оценивается величиной $1{,}82 \cdot 10^9$ опер./с.

Имеются также нанокоманды управления маршрутизатором и межпроцессорной сетью связей (NEWS Grid), а также нанокоманды, реализующие функции диагностики (в модели СМ-1 используется двумерная сеть, это и предопределило аббревиатуру NEWS: North — East — West — South).

Каждый ЭП может находиться либо в активном состоянии (и воспринимать нанокоманды), либо быть замаскированным.

Гиперкубическая межузловая сеть СМ-1. В составе любой из подсистем CM_0–CM_3 содержится 16 К ЭП, сгруппированных в 1024 ВУ. Взаимодействие между ВУ осуществляется через сеть связей, которая представляется в виде гиперкуба (Hypercube Data Network). Что собой представляет такой куб?

Понятие о гиперкубе дано в разд. 3.1.2, а на рис. 3.2 представлены гиперкубы размерностей 1, 2, 3, 4. Стандартное представление гиперкуба размерности 4, или 4D-куба — это конструкция «куб в кубе», в которой каждая вершина одного куба соединена ребром с симметричной вершиной другого куба (см. рис. 3.2). Очевидно, что такую конструкцию четырехмерного гиперкуба легко трансформировать к видам, представленным на рис. 5.7.

В представлении 4D-куба в виде одномерного куба (см. рис. 5.7) считается, что вершины — трехмерные кубы, а гиперребро — совокупность из восьми ребер, каждое из которых соединяет соответствующие вершины 3D-кубов.

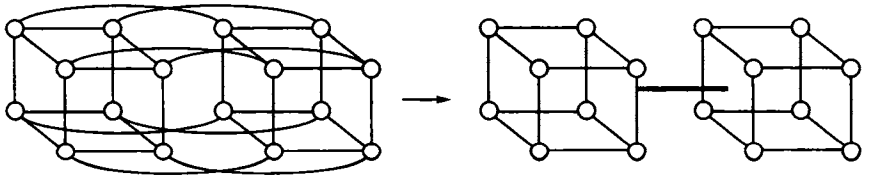


Рис. 5.7. 4D-куб:
——— — гиперребро

Очевидны и преобразования 4D-куба в гиперкубы большей размерности. Так, 5D-куб — это 2D-куб (квадрат), образованный из вершин, являющихся 3D-кубами (обычными кубами), и имеющий гиперребра размерности 8 (рис. 5.8). Далее, 6D-куб представляется 3D-кубом (обычным кубом) с вершинами, являющимися также 3D-кубами, и гиперребрами размерности 8 (рис. 5.9).

Гиперкуб размерности 9 (рис. 5.10) изображается 3D-кубом, в вершинах которого находятся 6D-кубы, а гиперребра имеют размерность 64.

Структура любой из процессорных подсистем в модели СМ-1 представляется гиперкубом размерности 10, т. е. 10D-кубом. Следовательно, она может быть изображена в виде двух 9D-кубов, соединенных между собой гиперребром размерности 512.

В гиперкубе диаметр, т. е. максимальное расстояние (из кратчайших) между любыми двумя узлами, равно его размерности. Наращивание (или сокращение) числа узлов в гиперкубе в два раза приводит к увеличению (или уменьшению) длины пути между двумя максимально удаленными узлами только на единицу. Следовательно, ВС с гиперкубической структурой характеризуются низкой латентностью, т. е. малым средним временем задержки при межузловых обменах информацией. В самом деле, при использовании в ВС n-мерного гиперкуба обмен информацией между двумя узлами требует ее прохождения не более чем через $(n - 1)$ транзитных узла.

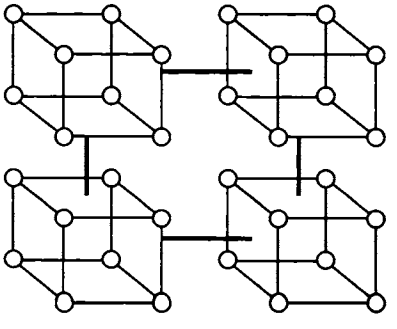


Рис. 5.8. 5D-куб

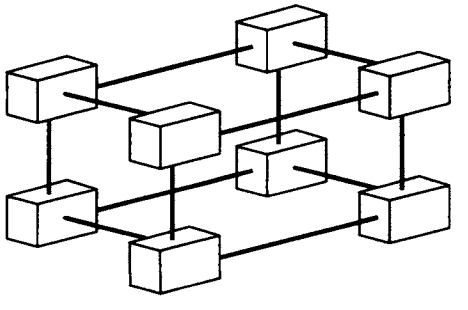


Рис. 5.9. 6D-куб

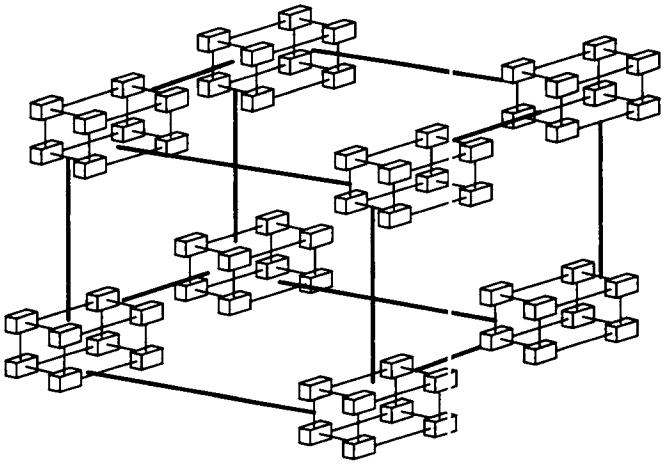


Рис. 5.10. 9D-куб

Коммуникационная среда ВС СМ-1. Вычислительные системы семейства СМ имеют развитую программно-настраиваемую коммуникационную среду, обеспечивающую межпроцессорные взаимодействия (именно это обосновывает название «Connection Machine»). В зависимости от типа и структуры решаемой задачи программируются межпроцессорные взаимодействия в системах СМ. Чтобы выразить отношения между элементарными процессорами, разработчики алгоритмов обычно используют технику структурирования данных. Например, в системах распознавания образов для представления пикселов обычно используются двумерные решетки из элементарных процессоров (рис. 5.11). Однако на поздней стадии обработки для представления более абстрактных отношений между объектами и их частями могут применяться структуры в виде дерева или реляционные графы (графы отношений).

Следует заметить, что в последовательной ЭВМ, имеющей память с произвольной выборкой, для создания сложных структур данных используется указатель элементов памяти. В архитектурах ВС с массовым параллелизмом элементы данных назначаются на индивидуальные ЭП, а отношения между элементами очень больших структур данных реализуются через межпроцессорные коммуникации.

В коммуникационной среде ВС СМ-1 поддерживаются следующие механизмы.

Широковещательная управляющая сеть (Broadcast Control Network) обеспечивает в ВС безотлагательный прием одновременно всеми ЭП управляющей информации, поступающей от устройства управления или коммуникационного процессора.

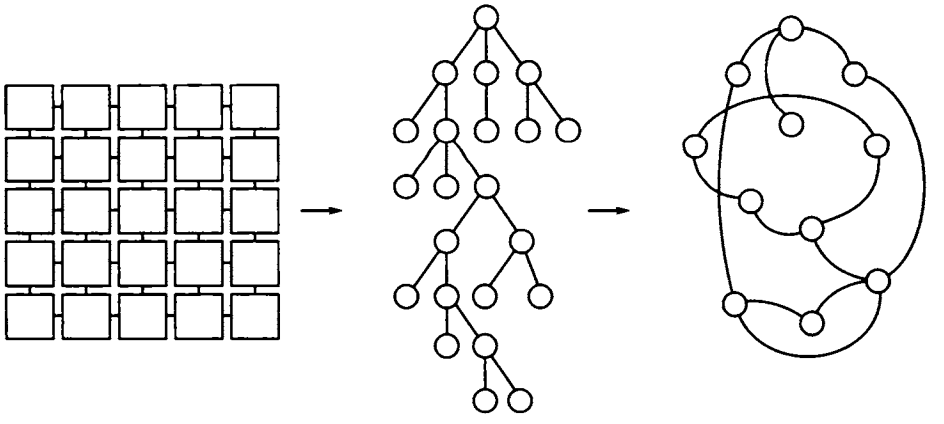


Рис. 5.11. Программирование межпроцессорных связей в ВС семейства СМ

Связь «обобщенное ИЛИ» (Global OR) осуществляет реализацию логического ИЛИ над значениями переносов в АЛУ всех элементарных процессоров; это дает возможность организовать управление параллельными процессами, произвести их синхронизацию либо выработать условие окончания вычислений (команда обобщенного условного перехода в ВС «Минск-222», разд. 7.3.2).

Гиперкубическая сеть (Hypercube Data Network) является средой для маршрутизаторов и многочисленных параллельных примитивов, поддерживаемых моделью виртуальной машины. Топология сети узловых связей ВС СМ-1 представляется 10D-кубом.

Маршрутизатор (Router) 16-процессорного кристалла (см. рис. 5.6) непосредственно осуществляет пересылку пакетов сообщений в ЭП. Адреса процессоров содержатся в указателе (Pointer), указатель следует вместе с пакетом сообщений. Контроллер маршрутизатора (реализованный в процессорном кристалле) использует гиперкуб при передаче данных между узлами. Эти аппаратурные средства обеспечивают частичное совмещение коммутации сообщений с решениями о маршрутизации, с буферизацией и объединением сообщений, направляемых по некоторому адресу.

Межпроцессорная сеть (NEWS Grid) предоставляет прямой доступ из данного ЭП к ближайшим соседям в пределах узла (процессорного кристалла). Структура узла представляет собой декартову решетку или 2D-решетку (тор) из 16 ЭП, или гиперкуб размерности 4 (см. разд. 3.1.2, рис. 3.1 и 3.2). Поскольку при использовании этой сети связей все процессоры передают информацию в одном и том же направлении (на север — N, восток — E, запад — W или юг — S), а адреса задаются неявно и никаких коллизий не происходит, то она заметно быстрее (примерно в 6 раз), чем сеть для маршрутизаторов (межузловая гиперкубическая сеть).

Виртуальная машина ВС СМ-1. Пользователю ВС СМ-1 предоставляется удобный сервис — виртуальная машина. Архитектура этой машины весьма близка к архитектуре физической системы Connection Machine и имеет два существенных расширения: ее набор параллельных команд (названный Paris — Парис) существенно расширен и в ней имеется абстракция виртуального процессора.

Границами набора команд Paris являются простые логические и арифметические операции и высокоуровневые операции, такие как APL-операции (APL — A Programming Language), сортировка и коммуникационные операции. Функции интерфейса Paris (между коммуникационным процессором и остальной частью системы СМ-1) сводятся к формированию потока кодов операций и аргументов. Аргументы — это, как правило, начальный адрес и количество бит (разрядность операнда). В качестве аргумента могут служить непосредственно данные или информация о широковещательном приеме. Большая часть набора Paris реализована в аппаратуре УУ, где осуществляется синтаксический анализ потока кодов операций и аргументов и его преобразование в соответствующую последовательность нанокоманд для ЭП. Поскольку Paris является набором команд виртуальной машины, то вполне допустимо использовать те же самые имена и для языка ассемблера системы СМ-1.

Виртуальный процессор необходим во многих областях параллельной обработки данных (так как часто требуются специфические процессоры, которые заметно отличаются от физических ЭП данной системы). Программное обеспечение ВС СМ-1 предоставляет механизм виртуального процессора, он поддерживается Paris и легко понятен пользователю. При инициализировании ВС СМ-1 указывается количество виртуальных процессоров, которое необходимо в конкретном применении ВС. В случае, если это количество превышает количество физических ЭП, то локальная память каждого ЭП расщепляется на области, а время процессоров автоматически делится между этими областями.

Например, если необходимо обработать 1 М порций данных, то, естественно, требуется $V = 2^{20}$ виртуальных процессоров. Пусть реальное число физических ЭП равно $P = 2^{16}$ и каждый процессор имеет локальную память емкостью 2^m бит (для модели СМ-1 $m = 12$, а для СМ-2 $m = 16$). Тогда каждый физический ЭП будет поддерживать $V / P = 16$ виртуальных процессоров.

Отношение $V / P = k$ называют коэффициентом виртуального процессора. В нашем примере каждый виртуальный процессор имел бы память емкостью $2^m / k = 2^{m-4}$ бит и обрабатывал бы коды с быстродействием, которое составляло бы всего $1/k = 1/16$ от быстродействия физического ЭП. Фактически виртуальные процессоры часто превышают эту скорость вы-

числений; расходы времени на декодирование команд в УУ могут быть уменьшены за счет множества виртуальных процессоров, поддерживаемых одним физическим ЭП.

Программное обеспечение ВС СМ-1. Основу системного ПО СМ-1 составляет операционная система (Operating System), являющаяся штатной операционной средой (либо UNIX, либо LISP) сервисных процессоров с небольшим расширением. Пользователям предоставляется возможность применять языки и конструкции всего программного инструментария сервисных процессоров. Кроме того, пользователи могут без особого труда разработать программы, рассчитанные на эксплуатацию всей вычислительной мощности аппаратуры системы Connection Machine.

Тем не менее следует заметить, что стандартные языки программирования все же имеют некоторые расширения, поддерживающие параллельные конструкции данных. Однако эти расширения не требуют изучения какого-либо нового стиля программирования.

Язык CM-FORTRAN системы Connection Machine использует расширения (Array Extensions) для работы с векторами, матрицами и массивами данных в стандарте FORTRAN 8x (предложенном American National Standards Institute Technical Committee). Эти расширения, естественно, отображаются в основном параллельном оборудовании системы СМ-1.

Языки *LISP (читается Star LISP) и CM-LISP являются параллельными диалектами обычного языка LISP (LISP — LISt Processing language — язык обработки списков). Язык *LISP позволяет программистам высококачественно управлять аппаратурой системы СМ-1. Диалект CM-LISP является языком высокого уровня. Он в отличие от обычного LISP имеет небольшие синтаксические дополнения в своем интерфейсе, которые и превращают его в язык параллельного программирования.

Язык C* является параллельной версией С. Программы на C* могут быть прочитаны и записаны так же, как последовательные С-программы. Расширения незначительные и легко воспринимаются.

Язык ассемблера (точнее, Paris) системы СМ-1 является объектным (выходным) языком компиляторов языков высокого уровня. Набор команд Paris логически расширяет набор команд сервисного процессора и маскирует физическую реализацию системы СМ-1.

5.4.2. Вычислительная система СМ-2

Модель СМ-2 — развитая версия СМ-1. При создании СМ-2 преследовали следующие цели:

- обеспечение совместимости с моделью СМ-1;

- увеличение производительности и емкости памяти;
- повышение общей надежности;
- упрощение производства;
- подключение высокоскоростной системы ввода-вывода (для внешней памяти и дисплеев).

Процессорный кристалл модели СМ-2 был спроектирован в начале 1986 г., а первый прототип СМ-2 был построен уже в конце 1986 г. Первые коммерческие поставки ВС были осуществлены осенью 1987 г.

Максимальная конфигурация СМ-2 состояла из 64 К ЭП (4 К вычислительных узлов) и 2 К акселераторов, реализующих операции с плавающей запятой. Такая конфигурация СМ-2 обладала быстродействием 2500 MIPS и 32 GFLOPS (над 32-разрядными числами) и оперативной памятью емкостью 512 Мбайт или 2 Гбайт (с пропускной способностью 300 Гбайт/с).

Начиная с 1989 г. Thinking Machines Corp. выпускала модификации СМ-2А Model 4 и СМ-2А Model 8, конфигурируемые из 4096 и 8192 ЭП соответственно. Эти модификации обладали рекордным соотношением производительность/стоимость. Если для высокопроизводительных компьютеров имело место среднее соотношение 1 млн опер./с за 100 тыс. долл., то в ВС с массовым параллелизмом СМ-2А оно достигало 1 млн опер./с за 500 долл. (см. разд. 2.10.2, $\sum = 5 \cdot 10^{-4}$ долл./(опер./с)).

Особенности функциональной структуры системы СМ-2. В состав ВС СМ-2 входят параллельное процессорное устройство (собственно СМ-2), коммуникационные процессоры, система ввода-вывода (I/0 system) и внешняя массовая память данных (Data Vault).

Функциональная структура СМ-2 масштабируемая. Минимальная конфигурация СМ-2 состоит из 2-х ВУ (из 32-х элементарных процессоров), а максимальная — из 4-х К узлов (из 65 536 ЭП). Структура любой конфигурации СМ-2 обязательно представляется гиперкубом. Следовательно, в СМ-2 могут быть использованы гиперкубы любой размерности от 1 до 12. Двенадцатимерный гиперкуб (12D-куб) представляется 3D-кубом, в котором каждая вершина является 9D-кубом (см. рис. 5.10), а гиперребра имеют размерность 512.

Модель СМ-2 — это система с программируемой структурой, следовательно, в «физическом» 12D-кубе могут быть построены произвольные «виртуальные» конфигурации.

Система ввода-вывода модели СМ-2 содержит восемь контроллеров, которые обеспечивают скорость обмена информацией между параллельным процессорным устройством и внешними устройствами, равную 320 Мбайт/с. Полоса пропускания шины ввода-вывода — 80 бит (64 бит — для данных, 8 бит — для контроля по четкости, 8 бит — для управления).

Внешняя массовая память состоит из восьми устройств и обеспечивает суммарную емкость 5 или 10 Гбайт и скорость обмена информацией 320 Мбайт/с.

Элементарные процессоры и вычислительные узлы СМ-2. Архитектуры элементарных процессоров моделей СМ-1 и СМ-2 совместимы. Основные особенности ЭП модели СМ-2:

- емкость битадресуемой памяти — 64 К бит или 256 К бит (вместо 4 К бит);
- четыре одноразрядных регистра признаков (вместо восьми);
- акселератор для операций с плавающей запятой, необязательный (Accelerator — ускоритель операций);
- универсальный интерфейс для поддержки *n*-мерных решеток (вместо двумерных, или NEWS-решеток);
- интерфейс ввода-вывода;
- развитые схемы обнаружения ошибок.

В модели СМ-2 используется четыре (а не два, как в СМ-1) типа кристаллов. Один из кристаллов — это специально спроектированный (заказной) процессорный кристалл, содержащий АЛУ, регистры признаков, маршрутизатор, интерфейс *n*-мерных решеток и интерфейс ввода-вывода для 16 элементарных процессоров и контроллер для организации гиперкубической сети. Второй кристалл — это коммерческая динамическая память с произвольной выборкой, в которой реализуются коррекция однобитных ошибок и обнаружение двухбитных ошибок. Еще два заказных кристалла используются для формирования акселератора, причем один из них реализует собственно операции с плавающей запятой, а другой — функции интерфейса. Композиция из последних двух кристаллов рассчитана на подключение к двум процессорным кристаллам (к кластеру из 32 элементарных процессоров). Таким образом, в полной 65536-процессорной конфигурации ВС СМ-2 содержится 4096 процессорных кристаллов, 2048 кристаллов для операций с плавающей запятой, 2048 кристаллов для интерфейса с плавающей запятой и оперативная память емкостью 512 Мбайт.

В качестве ВУ выступает конфигурация из 16 ЭП и память емкостью 1 Мбит. При этом следует отметить, что два узла имеют связь с акселератором (кристаллом для операций с плавающей запятой и кристаллом интерфейса).

Сеть межпроцессорных связей СМ-2. Сеть межпроцессорных связей в модели СМ-2 почти такая же, как в ВС СМ-1, однако в ней имеются две особенности. Во-первых, маршрутизатор процессорного кристалла СМ-2 обеспечивает связь с 12-ю соседними маршрутизаторами (в пределах 12D-структуры). Он более производительный и надежный и поддерживает диагностику. Во-вторых, в пределах процессорного кристалла (вершины гипер-

куба) допускается формирование n-мерных структур из элементарных процессоров. Последнее позволяет пользователям в зависимости от решаемой задачи применять для межпроцессорных связей наиболее адекватные структуры.

5.4.3. Вычислительная система CM-5

При разработке системы CM-5 в качестве рабочей была взята парадигма научных вычислений (а не парадигма искусственного интеллекта, как это имело место при создании CM-1).

Модель CM-5 — результат архитектурного развития семейства CM и применения более совершенных микропроцессорных БИС. Основной задачей ее создателей было удовлетворение заказа на супервычисления, на решение задач, требующих терафлопсной производительности. Работы по созданию модели CM-5 были завершены в октябре 1991 г., а через месяц было осуществлено ее анонсирование.

Модель CM-5 — масштабируема; в ее составе может быть от 16 до 16384 процессорных узлов. Максимальная конфигурация характеризуется производительностью порядка 1 TFLOPS (10^{12} операций с плавающей запятой в секунду), емкостью памяти не менее 512 Гбайт и пропускной способностью не менее 1 Тбит/с для устройств ввода-вывода информации.

Наибольшее распространение получили конфигурации CM-5 с числом процессорных узлов в диапазоне от 32 до 1024 (с быстродействием 3...100 GFLOPS и емкостью памяти 1...32 Гбайт).

Особенности архитектуры системы CM-5. В вычислительной системе CM-5 получила дальнейшее развитие идея совмещения архитектур SIMD и MIMD. Система CM-5 представляется композицией из множества процессорных узлов и гиперкубической коммуникационной среды. Данная система масштабируема; варьирование количества процессорных узлов может быть осуществлено в пределах от 16 до 16384, а межузловой структуры — от 4D-куба до 14D-куба.

В свою очередь, множество процессорных узлов образуется из подмножеств управляющих (control) и вычислительных (computational) процессоров. Такая композиция средств позволяет в пределах CM-5 выделять подсистемы (Partitions) с архитектурой SIMD, в каждой из которых имеется управляющий узел-процессор и подмножество вычислительных узлов-процессоров, взаимодействующих между собой через коммуникационную среду. Эти подсистемы масштабируемы и способны функционировать асинхронно. Количество SIMD-подсистем определяется числом управляющих узлов (называемых также менеджерами подсистем — Partition Managers).

Ясно, что композиция SIMD-подсистем и есть реализация возможностей MIMD-архитектуры.

Более детальное рассмотрение системы CM-5 позволяет сделать заключение о том, что ее архитектура сочетает в себе достоинства параллельно-векторных (PVP) и массово-параллельных (MPP) архитектур (см. § 4.3 и 4.5). Система CM-5 имеет распределенную оперативную память.

Управляющий и вычислительный узлы ВС CM-5. Управляющий узел в модели CM-5 предназначен для реализации функций операционной системы, в частности, он обрабатывает запросы с устройств ввода-вывода, управляет решением задач на множестве ВУ и межузловыми обменами информацией.

Функциональные структуры управляющих и вычислительных узлов почти идентичны. В состав каждого из этих узлов обязательно входят микропроцессор и локальная память (в управляющем она имеет больший объем). В модели CM-5 и управляющий, и вычислительный узлы (рис. 5.12) формируются на базе 64-разрядного RISC-микропроцессора (именно SPARC-микропроцессора фирмы SUN Microsystems). Ясно, что управляющий узел может выполнять и вычислительные функции.

Минимальная конфигурация ВУ состоит из средств для межузловых взаимодействий, SPARC-процессора и локальной памяти емкостью 32 Мбайт (см. рис. 5.12). В расширенных конфигурациях узла дополнительно может быть до четырех векторных процессоров (ВП) типа Cray (см. § 4.3). Каждый из ВП имеет 128 32-разрядных регистров. Все ВП имеют прямой доступ к локальной памяти через свои порты. Порт каждого ВП обладает пропускной способностью 0,5 Гбайт/с и подключен к 64-разрядной шине. Максимальная конфигурация узла обеспечивает производительность 128 MFLOPS при 64-разрядной точности.

В состав управляющего узла CM-5 обязательно входит интерфейс ввода-вывода (см. рис. 5.12). Реально, в качестве этого узла используется рабочая станция SUN SPARCstation.

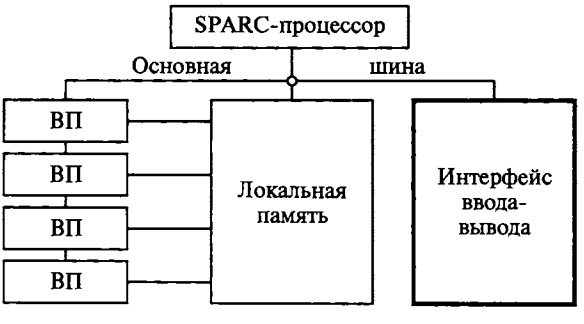


Рис. 5.12. Функциональная структура процессорного узла CM-5:
ВП — векторный процессор

В модели CM-5 в качестве базового используется массово выпускаемый микропроцессор, а не заказной одноразрядный (Custom-built Bit-serial Processor).

Коммуникационная среда ВС CM-5. Взаимодействия между процессорными узлами (и управляющими, и вычислительными) в системе CM-5 осуществляются через коммуникационную среду. Эта среда в CM-5 представлена тремя сетями: для управления, для передачи данных и для диагностики ВС. По первой сети передается управляющая информация, которая инициирует выполнение в системе глобальных действий, таких, например, как прерывание и синхронизация или реализация трансляции (Broadcasting) информации из управляющего узла во все вычислительные узлы (в пределах подсистемы). По второй сети осуществляются обмены данными между вычислительными узлами при их параллельной работе в SIMD-подсистеме. Сеть передачи данных CM-5 обеспечивает обмен информацией между соседними узлами со скоростью до 20 Мбайт/с. Передача информации по сети управления и сети передачи данных осуществляется 64-разрядными словами (по 64-разрядным шинам). Диагностическая сеть используется только при системном администрировании и скрыта от пользователей.

Программное обеспечение ВС CM-5. Модель ВС CM-5 работает под управлением распределенной операционной системы (ОС), которая позволяет нескольким пользователям работать на ресурсах ВС CM-5 одновременно (или даже ее подсистемы, но в режиме разделения времени). Каждый пользовательский процесс может исполняться только на одной SIMD-подсистеме и ему предоставляется доступ ко всем ресурсам этой подсистемы. Процессы, реализующиеся на различных подсистемах, могут взаимодействовать друг с другом.

В модели CM-5 используется ОС CMOST. Это расширенная версия ОС UNIX (UNIX — многопользовательская многозадачная ОС; считается, что она менее компьютерно-зависима, чем остальные ОС). Операционная система CMOST относится к распределенным, в каждом ВУ модели CM-5 размещается и исполняется только микроядро (Microkernel), для чего в узле необходимо 5 Мбайт локальной памяти. Все остальные компоненты CMOST находятся и реализуются в каждом из управляющих узлов (в каждой SIMD-подсистеме) модели CM-5.

Средства параллельного программирования CM-5 представлены компиляторами для языков CM FORTRAN (подобного FORTRAN 90), C* и *LISP (соответственно языков C и LISP с расширениями для параллельных вычислений; LISP — LISt Processing language — язык обработки списков).

Компиляторы реализуются специальным сервером (Front-end Compile Server). Компилирующий сервер построен на базе SPARC-микропроцессора. В состав программного обеспечения входит также пакет диагностических тестов.

5.4.4. Области применения систем семейства Connection Machine

Изначальный замысел Connection Machine — это создание в условиях технических и технологических ограничений 1980-х годов высокопроизводительного инструмента для решения проблем искусственного интеллекта. Этот замысел нашел воплощение в архитектуре ВС СМ-1 и СМ-2; первоначальную область их применения составили информационно-логические задачи. Первые параллельные алгоритмы сортировки и ассоциативного поиска, использующие возможности программирования структуры СМ-1, были разработаны В.Д. Хиллисом.

Вычислительные системы СМ-2 эффективно эксплуатировались в областях, в которых требовалось решать сложные задачи, связанные с переработкой большого количества данных. Это, в частности, поиск информации в базах данных, картография, обработка изображений, восстановление документов; автоматизированное проектирование, моделирование БИС; проектирование лекарственных препаратов; гидро- и газодинамика, моделирование прохождения волн через Землю; ядерная физика.

Было установлено свыше 30 ВС СМ-2 в различных организациях, включая Национальную лабораторию в Лос-Аламосе (Los-Alamos National Labs) и Национальный центр для исследования атмосферы (the National Center for Atmospheric Research).

Вычислительные системы СМ-5 в отличие от СМ-2 проектировались для супервычислений. Достигнутые их технические характеристики (производительность порядка 1 TFLOPS) позволили существенно раздвинуть границы применения семейства Connection Machine. В качестве областей применения систем СМ-5 можно указать, в частности, фармацевтическую промышленность (конструирование сложных химических соединений), ядерную физику, космонавтику, геофизику и геологию (моделирование геологических образований Земли).

Системы СМ-5 нашли широкое применение в исследованиях, проводимых в правительственных организациях и университетах, а также в международном бизнесе. В частности, система СМ-5 установлена в Лаборатории реактивных двигателей (Jet Propulsion Laboratory, NASA). Система используется для обработки данных, поступающих от космических кораблей-челноков (для получения с высоким разрешением цветных изображений поверхности Земли).

Конфигурация системы СМ-5 Национального центра по супервычислениям (NCSA — National Center for Supercomputing Applications), эксплуатировавшаяся несколько лет с мая 1993 г., состояла из 512 процессорных узлов. Она обладала производительностью 64 GFLOPS, имела распределенную память емкостью 16 Гбайт и параллельную дисковую память (SDA — Scalable Disk Array) емкостью 140 Гбайт.

Лос-Аломосская национальная лаборатория эксплуатирует конфигурацию СМ-5, состоящую не менее чем из 1024 процессорных узлов. Находятся в эксплуатации также конфигурации системы СМ-5Е, которая является развитой модификацией СМ-5 и компонуется из более производительных микропроцессоров (SuperSPARC processors) и векторных процессоров.

5.5. Семейство вычислительных систем nCube

Вычислительные системы семейства nCube (так же, как и модели Connection Machine) являются средствами обработки информации с массовым параллелизмом, т. е. MPP-системами (MPP — Massively Parallel Processing). Технико-экономическая эффективность MPP-систем была достигнута в 1980-х годах прежде всего за счет успехов в технологии БИС (VLSI — Very Large Scale Integration).

В состав семейства nCube входят следующие модели: nCube-1 (1985), nCube-2 (1989) и nCube-3 (1995). Все модели созданы одним из пионеров в области MPP-систем — компанией nCube, основанной в 1983 г. Системы семейства nCube показали рекордные для 1980-х годов результаты как по производительности, так и по показателю «цена/производительность» при решении сложных вычислительных задач.

5.5.1. Особенности архитектуры и функциональной структуры ВС семейства nCube

Архитектура ВС nCube относится к классу MIMD. Следовательно, они характеризуются высокой эффективностью использования оборудования (множества элементарных процессоров) при мультипрограммировании; в каждой из них допускается одновременная реализация нескольких параллельных программ на своих ресурсах. Кроме того, системы nCube поддерживают режим обработки информации, характерный для SIMD-архитектур (для матричных процессоров).

Структуры сетей межпроцессорных связей в системах nCube представлены гиперкубами (см. разд. 3.1.2 и 5.4.1). В системах nCube допускается настройка популярных (виртуальных) структур, включая кольца, решетки, деревья и тороидальные образования.

Максимальное число элементарных процессоров в конфигурациях ВС семейства nCube составляет 65 536; быстродействие — до нескольких TFLOPS; емкость памяти — до 64 Тбайт; масштабируемость гиперкубических конфигураций систем — от 8 до 65 536 ЭП.

Любая из систем nCube обладает *распределенной памятью*. В самом деле, каждый элементарный процессор ВС располагает своей собственной локальной памятью, и он имеет доступ через сеть межпроцессорных связей к памяти любого другого ЭП. Путем изменения числа ЭП в конфигурации ВС достигается *масштабирование* не только вычислительной мощности, но и емкости памяти и пропускной способности канала «процессор — память».

Элементарные процессоры (процессорные узлы) во всех системах семейства nCube реализованы в виде VLSI-схем.

Работоспособность систем nCube поддерживается с помощью методов контроля по четности и ECC-коррекции ошибок (ECC — Error Correction Code — код коррекции ошибок).

С развитием технологи БИС (с увеличением возможностей микропроцессорных VLSI-схем) осуществлялось эволюционное совершенствование моделей семейства nCube. Однако все модели данного семейства — по сути, масштабируемые ВС, любая из моделей могла состоять из 2^k элементарных процессоров, причем для nCube-1 и nCube-2 $k = \overline{3, 13}$, а для nCube-3 $k = \overline{3, 16}$.

Модель nCube-1 была построена в 1985 г. Она показала перспективность применения VLSI-схем и гиперкубических структур для межпроцессорных соединений. Эта модель позволила достичь рекордных для 1985 г. результатов по производительности и по показателю «цена/производительность» при решении сложных научных проблем.

Модель nCube-2 выпускалась серийно с 1989 г. в течение десятилетия. Максимальная конфигурация nCube-2 могла иметь в своем составе 8192 ЭП, представляющих 13-мерный гиперкуб. Для данной модели производительность составляла 34 GFLOPS и 123 000 MIPS.

Модель nCube-3 завершает ряд nCube, данная ВС разработана в середине 1990-х годов. Диапазон масштабирования ВС по числу элементарных процессоров: от 8 до 65 536. Производительность максимальной конфигурации nCube-3 оценивается несколькими TFLOPS, а ее структура представлена 16-мерным гиперкубом (16D-кубом).

В последующих разделах будут подробнее рассмотрены архитектурные решения в вычислительных системах nCube-2 и nCube-3.

5.5.2. Вычислительная система nCube-2

Вычислительные системы nCube-2 выпускались в двух модификациях: nCube-2 и nCube-2S. Первая модификация ВС базируется на ЭП с тактовой частотой 20 МГц, а вторая — 25 МГц. Для конфигураций nCube-2 доступ-

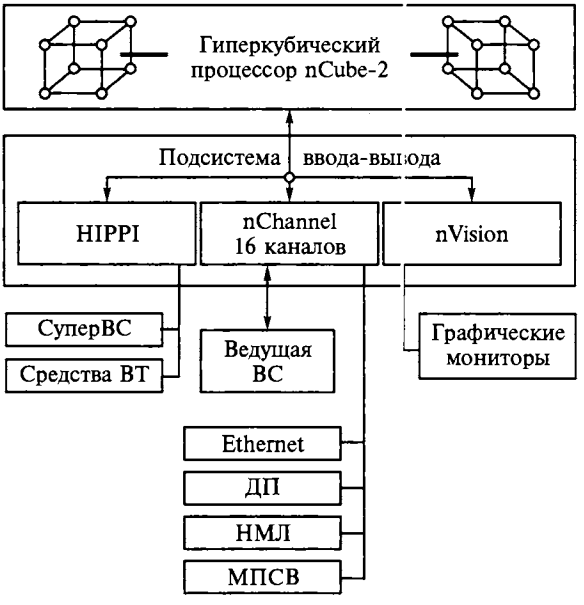


Рис. 5.13. Функциональная структура ВС nCube-2:

ДП — дисковая память; НМЛ — накопитель на магнитной ленте; МПСВ — модуль преобразования сигналов и видеообразов

ны следующие диапазоны производительности: 26 MFLOPS...27 GFLOPS, 60 MIPS...60 GIPS, а для nCube-2S — 33 MFLOPS...34 GFLOPS, 120 MIPS...123 GIPS. При масштабировании ВС варьируется не только производительность, но и емкость оперативной памяти — от 32 Мбайт до 32 Гбайт.

Функциональная структура ВС nCube-2. Вычислительная система nCube-2 представляет собой композицию (рис. 5.13) из гиперкубического процессора nCube-2, подсистемы ввода-вывода и ведущей ВС (см. функциональную структуру ILLIAC IV, разд. 5.2.1).

Процессор (или собственно ВС) nCube-2 — это множество элементарных процессоров, структура сети связей между которыми является гиперкубом. Данный процессор масштабируем, число ЭП в его конфигурациях может составлять 8–8192. В зависимости от количества ЭП в конфигурациях nCube используются гиперкубы от 3D-куба до 13D-куба.

Конструкция процессора nCube-2 формируется из стоек. В каждой стойке содержится 16 плат, на одной плате размещается 16 ЭП.

Подсистема ввода-вывода в системе nCube-2 параллельного действия. Она позволяет осуществлять параллельные вычисления в процессоре nCube-2 одновременно с параллельным обменом с внешними устройствами.

Подсистема ввода-вывода nCube-2 формируется из следующих плат: nChannal, HIPPI и nVision. Плата nChannal используется для реализации взаимодействий между системой nCube-2 и внешней средой. Она имеет 16 независимых каналов ввода-вывода, каждый из которых управляется одним ЭП (таким же, как в процессоре nCube-2). Пропускная способность каждого из каналов платы nChannal достигает 20 Мбайт/с. С помощью каналов платы nChannal система nCube-2 соединяется, в частности с ведущей ВС, сетью Ethernet, дисковой памятью (ДП), накопителями на магнитных лентах (НМЛ), модулями преобразования сигналов и видеообразов (МПСВ).

Плата HIPPI обеспечивает связь системы nCube-2 с другими высокопроизводительными ВС или иными средствами ВТ, поддерживающими интерфейс HIPPI (HIgh Performance Parallel Interface — высокопроизводительный параллельный интерфейс). На плате HIPPI содержится 16 ЭП, которые служат для обработки сетевого трафика (т. е. для организации и реализации обменов информацией; по 8 ЭП для управления и вводом, и выводом). Плата HIPPI обеспечивает пропускную способность при передаче данных до 100 Мбайт/с.

Плата nVision позволяет графически отображать большие объемы данных в реальном времени. На плате nVision размещены 16 ЭП, специальный текстовый процессор, оперативная память емкостью 16 Мбайт, буферная память емкостью 2 Мбайт, два контроллера.

Ведущая ВС (Host Computer) предназначается для реализации некоторых функций операционной среды: для загрузки, инициализации и «сброса» системы nCube-2, для разработки и запуска прикладных параллельных программ, для контроля за функционированием nCube-2 и, в частности подсистемы ввода-вывода. Ведущая ВС обеспечивает также доступ в интерактивном (on-line) режиме к документации по архитектуре и программному обеспечению nCube-2.

В качестве стандартной ведущей ВС в конфигурациях системы nCube-2 может использоваться любая рабочая станция компании Sillicon Graphics. Такие станции, по сути, являются многопроцессорными системами с общей памятью, которые работают под управлением операционной системы UNIX.

Элементарный процессор nCube-2. В системе nCube-2 ЭП — однокристальный, в состав его функциональной структуры входит:

- центральный процессор;
- блок для вычислений с плавающей запятой;
- кэш-память;
- оперативная память;
- 14 каналов прямого доступа к памяти (DMA — Direct Memory Access);
- устройство маршрутизации сообщений.

Центральный процессор — 64-разрядный, он производит целочисленные вычисления. Блок для вещественных вычислений представляет собой двухступенчатый конвейер, способный работать с 64-разрядными словами.

Кэш-память ЭП состоит из двух частей емкостью по 16 К байт каждая. Одна из этих частей предназначается для команд, а другая — для данных. При обращении к кэш-памяти поддерживается скорость 1,6 Гбайт/с. В ЭП используется память динамического типа емкостью 4...64 Мбайт.

При тактовой частоте 25 МГц ЭП имеет быстродействие при обработке целочисленных данных 15 MIPS и производительности 4,1 MFLOPS и 3 MFLOPS в вычислениях с плавающей запятой при одинарной и двойной точности соответственно.

Из 14 DMA-каналов ЭП 13 используются для взаимодействий с соседними ЭП (в 13D-кубе) и один служит для ввода-вывода информации. Это позволяет варьировать число процессоров от 8 до 8192, причем при этом адекватно масштабируется не только производительность и емкость памяти, но и пропускная способность канала ввода-вывода ВС.

Устройство маршрутизации ЭП обеспечивает транзитное прохождение сообщений по сети межпроцессорных связей без прерывания работы центрального процессора. Скорость передачи сообщений в nCube-2 составляет 2,75 Мбайт/с на один DMA-канал. Реализованный в ВС «механизм» маршрутизации позволяет автоматически отыскивать кратчайшие пути между взаимодействующими ЭП. Он предотвращает также взаимную блокировку сообщений. Следовательно, пользователи ВС освобождены от необходимости планировать маршруты для передачи сообщений по гиперкубической сети.

Таким образом, nCube-2 — система с программируемой структурой. Она, в частности, может быть разбита на подсистемы, каждая из которых способна реализовать свою параллельную программу одновременно с работой других подсистем.

Программное обеспечение nCube-2. Конфигурации ВС nCube-2 оснащаются параллельной программной средой PSE (Parallel Software Environment), интегрированной с операционной средой UNIX ведущей ВС. Среда PSE поддерживает и моно-, и мультипрограммный режимы функционирования системы nCube-2. Следовательно, множество пользователей могут одновременно работать на ВС, причем каждому из них предоставляется возможность автономного контроля процесса выполнения своей программы.

Среда PSE обеспечивает совместимость (см. § 2.4) конфигураций ВС nCube-2, состоящих из различного числа ЭП. Совместимость «снизу вверх» в nCube-2 выражается в том, что программы решения сложных задач (см. разд. 3.3.4), разработанные для конфигураций ВС с небольшим числом ЭП, не требуют переделок при их переносе на конфигурацию системы большего

размера. Наоборот, совместимость «сверху вниз» в ВС достигается при условии, что сложность (трудоемкость) задачи допускает ее вложение в конфигурацию с меньшими ресурсами по сравнению с первоначальной.

Среда PSE системы nCube-2 включает в себя следующие компоненты:

- распределенную ОС nCX;
- библиотеку стандартных UNIX-функций (UNIX — многопользовательская многозадачная ОС);
- набор драйверов (управляющих программ) для хост-интерфейса и интерфейса ввода-вывода;
- средства параллельного программирования (для написания, компиляции, отладки, запуска и анализа параллельных программ);
- библиотеку подпрограмм, обеспечивающих межпроцессорные взаимодействия;
- библиотеки подпрограмм для решения задач вычислительной математики;
- утилиты (обслуживающие программы) системного администратора;
- подсистему интерактивной (on-line) справочной документации.

Операционная система nCX осуществляет управление параллельными процессами и распределенной памятью, производит вызовы системных UNIX-функций, реализует передачи сообщений между элементарными процессорами, поддерживает мультипрограммный режим работы nCube-2 и т. п. Данная ОС построена по «микроядерной» технологии. Она представляется множеством идентичных микроядер, распределенным по всем ЭП системы nCube-2 (включая гиперкубический процессор и подсистему ввода-вывода). Микроядро (Microkernel) обеспечивает выполнение на ЭП такого минимума операционных функций, которого достаточно для достижения функциональной целостности множества ЭП как единой системы nCube-2. Объем микроядра для ВС nCube-2 не превышает 128 К байт (для сравнения: в системе СМ-2 он был равен 5 Мбайт).

Драйвер хост-интерфейса используется для взаимодействия системы nCube-2 с ведущей ВС. Драйверы интерфейса ввода-вывода служат для работы nCube-2 с внешней памятью, устройствами ввода-вывода информации и сетью Ethernet. Драйверы распределяются по элементарным процессорам как гиперкубического процессора, так и подсистемы ввода-вывода nCube-2.

Среди компонентов параллельного программирования системы nCube-2 имеются:

- набор средств Express для распараллеливания программ, написанных на языках FORTRAN и C;
- пакет Linda стандартных программ для передачи сообщений и параллельной обработки, используемый в качестве расширения для обычных последовательных языков программирования;

• языки параллельного программирования High Performance FORTRAN и Dataparallel C;

• компиляторы для языков программирования;

• отладчик параллельных программ;

• набор программ для графических мониторов.

В вычислительной системе nCube-2 для поддержки моно- и мультипрограммного режимов используются две парадигмы параллельного программирования: SPMD и MPMD соответственно.

Парадигма SPMD (Single Program — Multiple Data, одна программа — множество данных) предопределяет:

• загрузку в систему nCube-2 одной параллельной программы, состоящей из множества идентичных ветвей;

• размещение в каждом ЭП своей ветви программы и своей порции данных;

• одновременное выполнение в каждом ЭП своей ветви программы над своими данными и обмены информацией между процессорами.

Парадигма MPMD (Multiple Program — Multiple Data, множество программ — множество данных) предусматривает:

• загрузку в систему набора параллельных программ;

• реализацию каждой программы на своей подсистеме (на своем связном подмножестве ЭП).

5.5.3. Вычислительная система nCube-3

Система nCube-3 полностью совместима с ВС nCube-2, однако ее архитектурные характеристики существенно улучшены. Для ВС nCube-3 доступна производительность в несколько TFLOPS. Емкость оперативной памяти в максимальной конфигурации ВС оценивается значением от 1 Гбайт до 64 Тбайт.

Функциональная структура nCube-3 (в сравнении с nCube-2) характеризуется значительно большим диапазоном масштабирования: от 8 до 64 К элементарных процессоров, от 3D-куба до 16D-куба.

Система nCube-3 располагает более эффективной конфигурацией подсистемы ввода-вывода информации, называемой Parachannel. Эта подсистема представляется гиперкубической сетью процессоров ввода-вывода. Подсистема Parachannel допускает масштабирование от 3D-куба до 13D-куба, в ней может быть от 1 до 8 К процессоров ввода-вывода. Она связана с процессором nCube-3 каналами так, что каждый ее узел может обслуживать до 8 ЭП.

В системе nCube-3 реализован механизм виртуальной памяти. Он предоставляет каждому процессу адресное пространство в пределах до 256 Тбайт.

Элементарный процессор nCube-3 по своей производительности превосходит в несколько раз ЭП системы nCube-2, а емкость его локальной памяти может составлять от 16 Мбайт до 1 Гбайт (вместо 4...64 Мбайт).

5.6. Анализ матричных вычислительных систем

Матричные ВС, начиная с 60-х годов XX в., относятся к основным концепциям построения сверхмощных средств ВТ. Матричный способ обработки информации в отличие от конвейерного в принципе позволяет осуществлять неограниченное количество вычислительных процессов, следовательно, достичь любого уровня быстродействия вычислительных средств.

Матричные ВС — вариант технической реализации модели коллектива вычислителей. В таких системах в высокой степени воплощены фундаментальные архитектурные принципы.

1. *Параллельность выполнения операций* в матричных ВС обеспечивается на нескольких функциональных уровнях. На макроуровне параллельность достигается за счет одновременной работы нескольких матричных процессоров (квадрантов — в ILLIAC-IV и процессорных подсистем — в системах семейства Connection Machine).

На микроуровне параллельность выражается в возможности одновременной работы большого количества элементарных процессоров (64 ЭП в квадранте и 256 ЭП в системе ILLIAC-IV; 16 384 ЭП в подсистеме и 65 536 ЭП в моделях СМ-1 и СМ-2 и 16384 ЭП в модели СМ-5 семейства Connection Machine; 8192 ЭП в моделях nCube-1 и nCube-2 и 65 536 ЭП в модели nCube-5).

В современных суперсистемах выделяются промежуточные функционально-конструктивные образования из ЭП (обусловленные, в частности, технологическими возможностями элементарной базы). Так, в моделях СМ-1 и СМ-2 в качестве таких образований выступают процессорные кристаллы (узлы гиперкубической структуры). Эти кристаллы-узлы — матричные процессоры, в каждом из которых все ЭП (их 16 в СМ) взаимодействуют друг с другом через n-мерную (двумерную в СМ-1) решетчатую структуру. На рассмотренном уровне параллелизм также поддерживается: осуществляется параллельная работа элементарных процессоров в пределах каждого из кристаллов-узлов.

С развитием технологии БИС стало возможным построение параллельных суперВС на основе мощных 64-разрядных микропроцессорных узлов. Например, в ВС СМ-5 в качестве основного функционально-конструктивного образования выступает узел — конфигурация из серийного SPARC-

микропроцессора, векторных конвейеров, памяти и средств для межузловых обменов информацией. В моделях семейства ВС nCube узел является заказным однокристальным ЭП (или композицией из процессора для целочисленных вычислений, двухступенчатого конвейера для вещественных вычислений, памяти и средств для межпроцессорных взаимодействий). Следовательно, и в новейших суперВС параллельность выполнения операций также поддержана и на уровне отдельного узла.

Таким образом, современные суперВС, архитектура которых является результатом эволюционного развития канонического матричного процессора, являются системами с массовым параллелизмом (MPP-системами). В таких суперВС число параллельно выполняемых операций достигает порядков $10–10^5$ Кроме того, вычислительные суперсистемы, точнее их функционально-конструктивные образования, впитали в себя решения, характерные для параллельно-векторных систем (PVP-систем).

2. *Программируемость структуры* в матричных системах изначально проявлялась более сильно, чем в конвейерных. В самом деле, матричная ВС может быть так настроена, что ее различные квадранты или подсистемы будут одновременно решать различные задачи. Кроме того, в пределах квадранта или подсистемы имеется возможность программировать направление передачи информации от каждого ЭП и, следовательно, настраивать канал связи между любыми ЭП. В матричных ВС заложены средства программного управления состоянием каждого ЭП. Последнее позволяет матрицу или подсистему ЭП разбивать на группы, каждая из которых может реализовать свой режим обработки данных. Следовательно, на различных группах ЭП можно выполнять различные программы. Однако в силу того, что в квадранте или подсистеме имеется только одно устройство управления, группы ЭП должны работать последовательно.

Итак, если в мультипрограммном режиме в однопроцессорной ЭВМ имеет место разделение времени, то в матричной ВС — разделение «пространства» ЭП и системного времени. При этом различные задачи решаются на различных группах ЭП (состоящих из различного числа процессоров) и в различные (последовательные) отрезки времени. При такой организации мультипрограммного режима резко падает производительность всей системы, так как вместо параллельных вычислений проводятся последовательно-параллельные и в пределе последовательные.

Матричные ВС располагают функционально гибкой структурой сети межпроцессорных связей. В первых системах (таких, как ILLIAC-IV), предельно близких к канонической структуре матричного процессора, сеть связей между ЭП представлялась двумерными решетками.

В системах 1980-х и 1990-х годов двумерные решетки использовались для организации сети межпроцессорных связей только в пределах одного

кристалла-узла, причем число ЭП в нем соответствовало текущему уровню технологии БИС. Более того, при эволюционном развитии архитектуры ВС такие двумерные решетки претерпели трансформацию в более совершенные *n*-мерные (например, это имело место при переходе от модели СМ-1 к СМ-2). Процессорные кристаллы брались в качестве функционально-конструктивных элементов, а сама система формировалась как композиция множества кристаллов и сети связей между ними. Технология БИС и техника конструирования систем уже позволила формировать из таких кристаллов гиперкубы.

Таким образом, программируемые структуры сетей межпроцессорных связей позволяют ВС адаптироваться под области применения и структуры решаемых задач, они делают локальную оперативную память любого ЭП общедоступной для других ЭП системы.

3. *Однородность состава и структуры* ВС видна на всех функциональных уровнях. На макроуровне однородность выражена тем, что все матричные процессоры (или квадранты в ILLIAC-IV, или подсистемы в моделях СМ) и устройства управления, входящие в них, одинаковы. На микроуровне однородность ВС достигнута за счет применения множества идентичных элементарных процессоров.

Сети межпроцессорных связей в матричных ВС однородные — это и двумерные решетки, и гиперкубы. Однородность проявляется и в конструкции матричных ВС, они формируются из конструктивно однотипных элементарных процессоров или процессорных кристаллов-узлов.

Архитектура матричных ВС не лишена существенных недостатков. Так, единственное устройство управления в квадранте или подсистеме резко снижает надежность и живучесть, а также производительность при мультипрограммной работе и, следовательно, ограничивает сферу применения матричных ВС. Эти недостатки и сравнительно небольшие экономические преимущества, полученные за счет общего устройства управления в матричном процессоре (в квадранте или подсистеме), заставляют разработчиков постепенно отходить от принятой ими архитектурной концепции и переходить к сформулированной в Сибирском отделении РАН модели коллектива вычислителей [5]. Кардинальным шагом стало введение в функциональную структуру матричной ВС множества устройств управления (управляющих узлов, см. разд. 5.4.3). Очевиден и предельный вариант развития функциональной структуры ВС, он предопределяет каждому ЭП свое устройство управления (см. разд. 5.5.2). Это и является констатацией о полном переходе к модели коллектива вычислителей.

В заключение следует подчеркнуть, что современные высокопроизводительные ВС являются «симбиозом» архитектур. Действительно, на макроуровне модели семейства СМ являются ничем иным, как MIMD-систе-

мами, а их подсистемы имеют архитектуры класса SIMD; узлы моделей СМ-1 и СМ-2 являются «SIMD-матрицами», состоящими из последовательных ЭП с классической SISD-архитектурой. Далее, в модели СМ-5 семейства Connection Machine и MIMD-системах семейства nCube процессорные узлы используют конвейеры, архитектура которых относится к классу MISD.

Характерным стало построение ВС из универсальных промышленных изделий, а не из уникальных заказных БИС. Так, например, в ВС СМ-5 (в отличие от СМ-1 и СМ-2) применен массово-производимый высокопроизводительный многоразрядный микропроцессор (а не заказные БИС из 16 последовательных элементарных процессоров). Это позволило достичь производительности 1 TFLOPS и улучшило технико-экономические показатели системы.

Таким образом, матричные вычислительные системы с канонической архитектурой относятся к важнейшим вехам компьютерной истории. Матричные ВС с момента своего зарождения обеспечивали уровень производительности, адекватный потребностям в высокопроизводительных вычислениях и технико-экономическим возможностям общества. Промышленные матричные ВС, «стартовавшие» от производительности 10^8 опер./с, достигли в результате своего архитектурного развития и применения микропроцессорных БИС производительности порядка 10^{12} опер./с.

6. МУЛЬТИПРОЦЕССОРНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Семантика термина «Мультипроцессорные вычислительные системы» предопределяет средства обработки информации, каждое из которых имеет в своем составе множество процессоров. Если ограничиться только такой интерпретацией, тогда в класс мультипроцессорных попадут все параллельные ВС, независимо от их архитектурных особенностей (в частности, конвейерные и матричные ВС, см. гл. 4 и 5).

Мультипроцессорные (или многопроцессорные) ВС — класс параллельных средств обработки информации, которые характеризуются тремя особенностями:

- *MIMD-архитектурой;*
- *множеством процессоров;*
- *единым общедоступным ресурсом (как правило, общей оперативной памятью).*

С развитием архитектуры ВС границы между различными каноническими классами систем стираются, уже сейчас они в достаточной степени условны.

6.1. Каноническая функциональная структура мультипроцессора

По определению *мультипроцессорная ВС — средство обработки информации, в котором имеется множество процессоров, взаимодействующих между собой через единый ресурс.* В качестве единых ресурсов выступают машины-посредники, внешние запоминающие устройства, оперативная память, коммутаторы, общие шины и т. п.

Анализ мультипроцессорных ВС и тенденций их развития позволяет считать в качестве канонической функциональную структуру мультипроцессора, представленную на рис. 6.1. Мультипроцессор — это композиция, в которой выделяются подмножество элементарных процессоров (ЭП), подмножество модулей памяти (МП) и коммутатор, обеспечивающий взаимодействие между любыми элементами различных подмножеств. Подмножество модулей памяти $МП_1$– $МП_m$ является общей памятью для всех про-

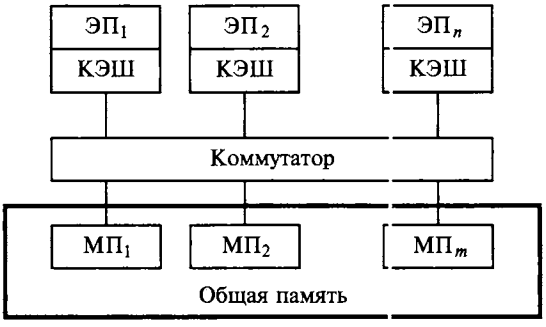


Рис. 6.1. Каноническая функциональная структура мультипроцессора:
ЭП — элементарный процессор; МП — модуль памяти; КЭШ — кэш-память

цессоров $ЭП_1$–$ЭП_n$, обычно $n \leqslant m$. Взаимодействие между ЭП осуществляется не через коммутатор, а через общую память. Все ЭП, как правило, идентичны.

Коммутатор может быть сосредоточенным или распределенным. В последнем случае он представляется композицией из локальных коммутаторов, каждый из которых находится во взаимно однозначном соответствии либо с ЭП, либо с модулем памяти.

В мультипроцессорах вместо коммутатора, обеспечивающего доступ ЭП к общей памяти, можно использовать шину или иерархическую композицию шин.

В теоретических исследованиях параллельных алгоритмов часто используется идеализированный мультипроцессор — PRAM-модель (Parallel Random Access Machine — параллельная машина с произвольным доступом к памяти). Допускается, что в PRAM-модели любой ЭП может обращаться к любой ячейке общей памяти за одно и то же время. На практике же это далеко не так. Необходимость масштабирования мультипроцессора однозначно приводит к иерархии памяти, т. е. к организации кэш (Cache) — сверхоперативной «надстройки» памяти. Кэш-память предназначена для хранения копий часто используемых данных, она сокращает частоту обращения ЭП к общей памяти. Доступ ЭП к кэш-памяти осуществляется намного быстрее, чем к общей памяти мультипроцессора. Следовательно, введение кэш-памяти повышает эффективность мультипроцессора. Как и коммутатор, кэш-память мультипроцессора может быть распределенной (см. рис. 6.1).

Средства обработки информации, основанные на мультипроцессоре с канонической функциональной структурой, называют *ВС с общей (разделяемой) памятью* (True Shared Memory).

В первых мультипроцессорных ВС (60-е и 70-е годы XX в.) число n ЭП достигало десяти. Современные мультипроцессорные ВС представляют

собой системы с массовым параллелизмом (MPP Systems, Massively Parallel Processing Systems); количество ЭП в них составляет $10–10^5$

Достаточно обширные экспериментальные исследования в области мультипроцессорных систем были выполнены в 70-х годах XX в. в Университете Карнеги-Меллона (Carnegie-Mellon University, США). Исследователи прошли путь от канонической структуры мультипроцессора до распределенных ВС и убедились в перспективности последних. Они независимо от результатов исследований, выполненных в СССР, дали технико-экономическое обоснование нашей отечественной концепции распределенных ВС.

Значительные достижения по развитию ВТ принадлежат научной школе по многопроцессорным системам со структурно-процедурной организацией вычислений Таганрогского государственного технического университета.

Большой вклад в архитектуру и практику мультипроцессорных ВС сделан Институтом точной механики и вычислительной техники им. С.А. Лебедева АН СССР и Burroughs Corp.

6.2. Вычислительная система C.mmp

Вычислительная система C.mmp (Carnegie-Mellon Multi-Processor) была создана в университете Карнеги-Меллона (США). Работы в области архитектуры мультипроцессорных ВС, начатые университетом в 1970 г., преследовали такие общие цели* [6]:

1) достижение высокой производительности (большой полосы пропускания канала «процессор — память»);

2) проведение экспериментальных исследований по эффективности параллельной обработки данных;

3) экспериментальное изучение и обеспечение надежности;

4) достижение приемлемых технико-экономических показателей;

5) воплощение принципа максимального использования аппаратурно-программных средств мини-ЭВМ.

Последний принцип позволил:

- свести разработку системы к работам по созданию лишь системных компонентов, тем самым не расходовать материальных ресурсов на проектирование и изготовление процессоров, памяти и устройств ввода-вывода информации;

---

* Исследование систем C.mmp, Cm* и C.vmp. I. Опыт обеспечения отказоустойчивости в мультипроцессорных системах / Д.П. Северек, В. Кини, Х. Мешберн и др. // ТИИЭР. 1978. Т. 66. № 10. С. 89–117.

• использовать ПО (в частности, контрольно-диагностические программы) серийной аппаратуры;

• достичь бóльшей надежности в работе ВС как совокупности взаимосвязанных модулей обработки и хранения информации (благодаря их массовому производству).

6.2.1. Функциональная структура мини-ВС C.mmp

Вычислительная система C.mmp принадлежала к мини-машинным, т. е. мини-ВС, и формировалась из средств мини-ЭВМ. Из-за высокой стоимости мини-ЭВМ 1970-х годов была выбрана достаточно простая функциональная структура системы (рис. 6.2), а проблема поддержки ее надежности была решена программными средствами. Система состояла из 16 ЭП, общей памяти и матричного коммутатора. В состав ЭП (мини-ЭВМ корпорации DEC, Digital Equipment Corp.) входили незначительно модифицированный процессор PDP-11/40, локальная (или местная, или индивидуальная) память, блок отображения адреса (который преобразовывал генерировавшиеся процессором 18-разрядные адреса в 25-разрядные физические адреса) и контроллер межпроцессорного интерфейса (который обеспечивал подключение процессора к межпроцессорной шине). Кроме указанных компонентов в состав ЭП могли входить память на магнитных дисках, страничная память на дисках, внешние устройства и др. Матричный коммутатор 16 × 16 позволял установить связь между любым процессором $ЭП_i$ и любым портом $МП_j$ ($i, j \in \{1, 2, \ldots, 16\}$) памяти общего доступа.

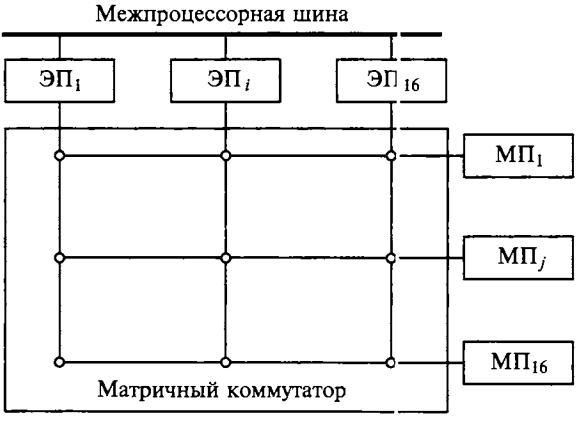


Рис. 6.2. Функциональная структура системы C.mmp:
ЭП — элементарный процессор; МП — модуль памяти

6.2.2. Анализ надежности мини-ВС C.mmp

Простота процесса построения вероятностных моделей для мини-ВС C.mmp и их исследования обусловлена принципом однородности модулей (ЭП и модулей памяти). Легко заметить, что нет каких-либо принципиальных трудностей в представлении системы C.mmp в виде модели ВС со структурным резервом и с восстановлением. Однако исследователи из Университета Карнеги—Меллона ограничились изучением модели ВС с резервом, но без восстановления. В рамках этой простейшей модели предполагалось, что восстановление начинается только после того, как отказ очередного модуля приводит к отказу системы в целом. Говоря иначе, считалось, что ВС исправно работает при отказах ЭП и модулей памяти вплоть до истощения резервных ресурсов.

При использовании такой модели ВС (с резервом и с отказами) считается, что вероятность безотказной работы в течение времени t любой подсистемы (любого подмножества) из N идентичных модулей равна

$$R(t) = \sum_{i=0}^{N-n} \frac{N!}{i!(N-i)!} r^{N-i}(t)[1-r(t)]^i, \qquad (6.1)$$

где $r(t) = \exp(-\lambda t)$ — функция надежности (вероятность безотказной работы модуля); λ — интенсивность отказов модуля (см. разд. 2.8.2); n — допустимое число исправных модулей; $(N-n)$ — резерв (следовательно, изучаемая подсистема устойчива к $(N-n)$ отказам модулей).

Реальная ВС C.mmp содержала несколько подсистем одинаковых модулей (процессоров, модулей локальной памяти для каждого процессора, модулей памяти общего доступа для каждого из портов, блоков отображения, контроллеров межмашинного интерфейса и др.) и единственный коммутатор. Эти структурные особенности требуют преобразований формулы (6.1), которые заметно усложняют счет. Надежность ВС C.mmp существенно определял способ организации коммутатора. При этом разработчиками системы использовались две модели коммутатора. В простейшем случае (*сосредоточенный коммутатор*) коммутатор рассматривался как единый элемент, выход которого из строя вызывал отказ всей системы. Вторая модель (*распределенный коммутатор*) отражала потенциальные возможности структуры коммутатора (далеко не все отказы коммутатора приводили к отказу системы).

Из анализа значений функции $R(t)$ (6.1) было установлено, что для задач, требующих исправности восьми процессоров, отношение интервала времени, в течение которого вероятность безотказной работы ВС C.mmp при использовании распределенной модели коммутатора превышает уро-

вень 0,9, к соответствующему интервалу для сосредоточенной модели коммутатора составляет 2700:350, или примерно 7,7. Таким образом, даже не выходя из рамок сформулированных моделей, анализ надежности ВС C.mmp показал, что сосредоточенный коммутатор является критическим источником отказов в мини-ВС.

Система C.mmp и при отсутствии средств восстановления обладала надежностью, допускавшей решение сложных задач (при этом резерв не превышал 25 % общего числа процессоров в системе). Дальнейшее повышение надежности ВС C.mmp могло быть обеспечено повышением надежности компонентов и (или) применением средств восстановления.

Информация о надежности ВС C.mmp была получена на основе статистики, не учитывающей перемежающихся отказов (сбоев) модулей системы. В реальных условиях их следовало учитывать. Согласно статистическим данным, среднее время безотказной работы системы в целом (и без резерва) составляет всего лишь 9,2 ч, а среднее время ее восстановления — немного более 5 мин.

6.2.3. Недостатки архитектуры мини-ВС C.mmp

1. Сосредоточенный коммутатор в мини-ВС C.mmp являлся критическим источником отказов. Как показали исследования, вероятность безотказной работы системы резко убывает на начальном участке времени. Это нейтрализует увеличение надежности мини-ВС за счет введения резерва ЭП и модулей памяти. В то же время, если использовать распределенный коммутатор, то будет достигнуто повышение надежности мини-ВС (благодаря резерву процессоров и модулей памяти).

2. Существуют границы для количества резервных модулей. Увеличение количества резервных модулей сверх этих границ практически не улучшает надежности систем. Так, в мини-ВС C.mmp такой границей для числа процессоров является $N - n = 8$.

3. Заметное повышение надежности мини-ВС обеспечивается применением высоконадежных компонентов. Однако такой путь имеет свои физические и технические пределы.

4. Кардинальный путь повышения надежности ВС состоит не в увеличении надежности компонентов, а в применении перспективных архитектурных и структурных решений, новых принципов обработки информации.

Перспективными представляются архитектуры, которые допускают автоматическое изменение (точнее, программирование) структуры и параметров ВС с целью установления такого соотношения между производительностью и надежностью, которое наиболее адекватно сфере применения.

Таким образом, принципиальным недостатком структуры мини-ВС C.mmp является матричный коммутатор, выход которого из строя приводит к отказу системы как единого ансамбля модулей. Архитектура ВС с общим коммутатором в современных условиях представляется неперспективной. *Распределенный (а не сосредоточенный) коммутатор должен стать средством обеспечения взаимодействий между элементарными процессорами в ВС.*

6.3. Семейство вычислительных систем Burroughs

Фирма «Бэрроиз» (Burroughs Corp.) в 1961 г. начала работы по созданию своего семейства многопроцессорных ВС. В 1963 г. была выпущена первая ВС В 5000, а в 1964 г. ее модификация В 5500; в 1969 г. была создана В 6500, в 1971 г. — В 6700 и, наконец, в 1973 г. — В 7700.

В этих ВС нашли воплощение новые архитектурные и структурные решения, которые радикально отличались от концептуальных решений ЭВМ Дж. фон Неймана. Так, например, даже в ВС В 5000 было реализовано следующее:

- «ручная» реконфигурируемость состава (в ВС могло быть один или два центральных процессора и до восьми модулей оперативной памяти);
- механизм виртуальной памяти;
- аппаратурная реализация функций, выполнявшихся ранее программно;
- операционная система — главная управляющая программа (Master Control Program);
- языки высокого уровня ALGOL 60 и COBOL.

В семействе ВС Burroughs, начиная с В 5500, воплощена концепция виртуальной машины.

Архитектура ВС семейства Burroughs относится к типу MIMD [13].

6.3.1. Вычислительная система В 6700

Вычислительная система В 6700 (1971 г.) — это композиция (см. рис. 6.1 и 6.3) ЭП, модулей памяти, коммутатора и периферийного оборудования (процессоров передачи данных, каналов, контроллеров и др.). Подмножество ЭП составляли 1–3 центральных процессоров и 1–3 процессоров ввода-вывода.

Центральный процессор (ЦП) системы В 6700 обладал быстродействием порядка 1 млн опер./с (над 48-разрядными числами). Оперативная память состояла из 1–64 модулей (МП), обладала емкостью до 6 Мбайт и име-

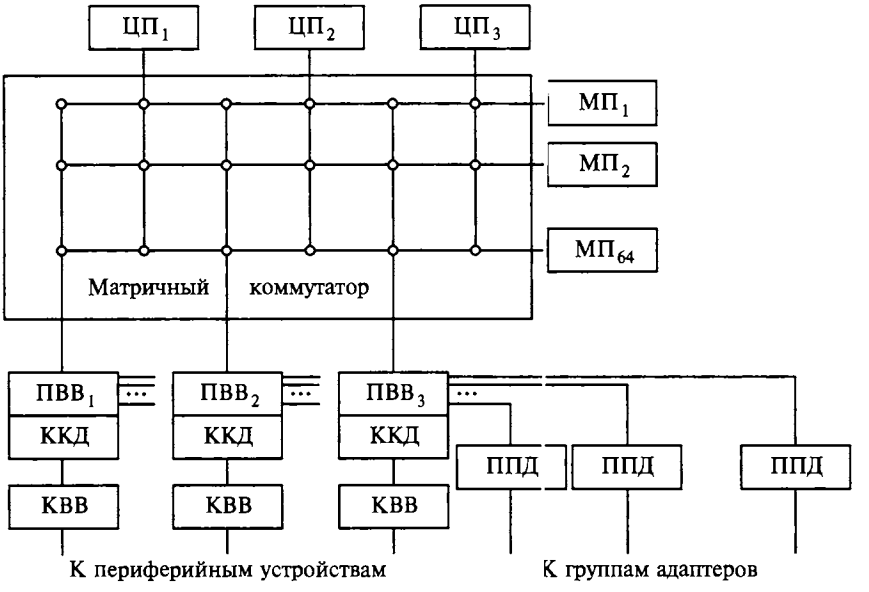


Рис. 6.3. Функциональная структура ВС В 6700:

ЦП — центральный процессор; ПВВ — процессор ввода-вывода; ККД — канал коммутации данных; КВВ — контроллер ввода-вывода; ППД — процессор передачи данных; МП — модуль памяти

ла шесть входов. Для формирования памяти использовались модули емкостью 6 К или 64 К слов, цикл обращения к модулю (в зависимости от его типа) был равен 1,5 или 1,2 мкс, или 500 нс. Скорость обмена информацией между процессорами (центральными и/или ввода-вывода) и оперативной памятью по любому из шести входов составляла 6750 К байт/с.

Процессоры ввода-вывода (ПВВ) предназначались для подключения периферийного оборудования, в каждом из процессоров имелось 4–12 каналов коммутации данных (ККД). Три процессора ввода-вывода могли одновременно выполнять до 36 операций ввода-вывода. К каждому каналу могло подключаться до 20 контроллеров ввода-вывода (КВВ), а к каждому контроллеру — до 10 периферийных устройств. Максимальное количество адресуемых периферийных устройств составляло 128, среди этих устройств имелись запоминающие устройства на магнитных лентах и дисках, графопостроители, алфавитно-цифровые печатающие устройства, перфокартные и другие устройства.

Каждый ПВВ был соединен с 1–4 процессорами передачи данных (ППД), каждый из которых был связан с 1–16 группами адаптеров, а каждая из них, в свою очередь, соединялась 1–16 линиями связи.

6.3.2. Вычислительная система B 7700

Вычислительная система B 7700 (1973 г.) в отличие от B 6700 могла иметь в своем составе 1–7 ЦП и от 1–7 процессоров ввода-вывода информации, при этом общее количество процессоров не превышало восьми. Быстродействие ЦП составляло 4...5 млн опер./с. Процессор ввода-вывода обслуживал 32 канала и четыре процессора передачи данных. К каждому каналу подключалось периферийное оборудование; максимальное количество адресуемых внешних устройств составляло 255. Процессор передачи данных был рассчитан на подключение до 256 линий связи. Оперативная память ВС B 7700 состояла из восьми модулей, была 8-входовой и имела емкость 12 288 К байт. Слово состояло из 52 разрядов, из которых 48 были информационными, 3 — управляющими и 1 — для контроля по четности. Все разряды слова были доступны процессорам, как центральным, так и ввода-вывода и передачи данных.

Программное обеспечение системы B 7700 представляло собой модификацию ПО B 6700. Все системные программы (кроме ОС) и все прикладные программы могли работать как на B 6700, так и на B 7700 без каких-либо изменений. Программное обеспечение в комплексе с аппаратурой выполняло функции обнаружения, локализации и устранения неисправностей и ошибок. При наличии отказов осуществлялось динамическое изменение рабочей конфигурации ВС. Операционная система после обнаружения и локализации отказа или ошибки исключала из рабочей конфигурации неисправные модули или программы с ошибками (без влияния на другие программы).

Опыт, накопленный при создании ВС B 6700 и B 7700, был положен фирмой Burroughs в основу последующих разработок мультипроцессорных ВС.

6.4. Семейство вычислительных систем «Эльбрус»

Работы по проектированию семейства ВС «Эльбрус» проводились в 1970-х и 1980-х годах под руководством В.С. Бурцева (1927–2005; академик РАН с 1992 г.) в Институте точной механики и вычислительной техники (ИТМиВТ) им. С.А. Лебедева АН СССР. Аванпроект ВС «Эльбрус» был разработан в 1970 г., модель «Эльбрус-1» была принята Госкомиссией в 1980 г., а «Эльбрус-2» — в 1985 г. [3, 7, 13]. Обе модели выпускались в СССР более 15 лет.

В рамках работ по проекту «Эльбрус» преследовалась цель — создать семейство высокопроизводительных и надежных ВС. Для моделей семейства «Эльбрус» характерны:

- MIMD-архитектура;

- распределенное управление;
- однородность, модульность и масштабируемость структуры;
- надежность и самоконтроль;
- аппаратурная поддержка функций ОС и средств языка высокого уровня;
- разрядность слов — 32, 64, 128;
- многоуровневая память;
- спецпроцессоры приема-передачи данных;
- производительность — до 125 MFLOPS.

6.4.1. Функциональная структура семейства вычислительных систем «Эльбрус»

Любая из моделей семейства «Эльбрус» представляет собой распределенную вычислительную систему, построенную по модульному принципу (рис. 6.4). Система обладала свойством масштабируемости и допускала формирование конфигураций, адекватных сферам применения и/или финансовым возможностям потребителей. В состав системы «Эльбрус» могло входить от 1 до 10 ЦП, от 4 до 32 модулей оперативной памяти, от 1 до 4 процессоров ввода-вывода, от 1 до 16 процессоров приема-передачи данных, необходимое количество устройств внешней памяти (УВП; накопителей на магнитных лен-

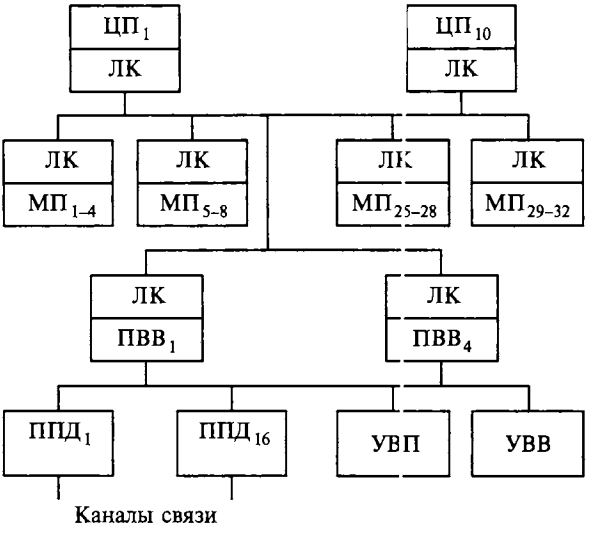


Рис. 6.4. Функциональная структура системы «Эльбрус»:

ЦП — центральный процессор; ЛК — локальный коммутатор; МП — модуль памяти; ПВВ — процессор ввода-вывода; ППД — процессор передачи данных; УВП — устройство внешней памяти; УВВ — устройство ввода-вывода

тах, барабанах и дисках) и устройств ввода-вывода информации (УВВ). Предусматривалась возможность подключения УВВ либо непосредственно к ПВВ, либо через каналы связи к ППД. Взаимодействие между подмножествами ЦП, МП и ПВВ осуществлялось через распределенный коммутатор, представлявший собой композицию локальных коммутаторов (ЛК).

Функционирование каждого из компонентов (ЦП, ПВВ, четырех модулей памяти и ЛК) системы «Эльбрус» поддерживалось средствами аппаратного контроля. Эти средства при появлении даже одиночной ошибки вырабатывали сигнал неисправности. По этому сигналу ОС через аппаратно-реализованную среду реконфигурации исключала неисправный компонент из рабочей конфигурации. Отключенный компонент попадал в ремонтную конфигурацию, где он при помощи контрольно-диагностических программ и специальной аппаратуры ремонтировался, после чего мог быть включен ОС в рабочую конфигурацию.

Средства реконфигурации позволяли организовать конфигурации ВС повышенной надежности. Так, например, допускались конфигурации ВС с резервом на уровне однотипных компонентов. Время включения резервного компонента в рабочую конфигурацию ВС не превышало 0,01 с. Надежность ВС выражалась в возможности передачи функций отказавшего компонента другому такого же типа, что, конечно, приводило к снижению рабочих характеристик (производительности, в частности), но не к отказу ВС в целом.

Средства реконфигурации предоставляли также возможность настраивать ВС на решение задач различных классов.

Помимо общедоступной оперативной памяти ($МП_1$–$МП_{32}$) в системе «Эльбрус» имелась также и сверхоперативная память, распределенная по центральным процессорам ($ЦП_1$–$ЦП_{10}$).

Каждое слово в ВС сопровождалось тегом, указывающим тип данных (целое, вещественное, адрес, метка процедуры и т. д.) и формат данных (32, 64 или 128 разрядов).

Работа ВС «Эльбрус» осуществлялась под управлением ОС. В этой ВС был реализован механизм управления вычислительными процессами, учитывающий состав и состояние ресурсов ВС. Вычислительный процесс мог быть активным или пассивным. Одновременно несколько процессов могли быть активными, при этом любой процессор мог работать с любым из них. Для синхронизации процессов использовались семафоры; система команд содержала операции «открыть семафор» и «закрыть семафор». Если все процессоры были заняты работой, то могла образовываться очередь готовых к выполнению процессов. После освобождения какого-либо процессора (в частности, если реализуемый в нем активный процесс по тем или иным причинам становился пассивным или заканчивался) происходило об-

ращение к очереди готовых процессоров и этот процессор начинал выполнять первый из них. Процессы могли перераспределяться в очереди в зависимости от их приоритетов.

6.4.2. Вычислительная система «Эльбрус-1»

Опытные образцы модели «Эльбрус-1» были изготовлены в начале 1978 г., а промышленная система «Эльбрус-1» была принята Госкомиссией в 1980 г. Система «Эльбрус-1» имела производительность от 1,5 до 15 MFLOPS и емкость оперативной памяти от 576 до 4608 К байт в зависимости от конфигурации.

Центральный процессор ВС «Эльбрус-1» оперировал с числами, имевшими формат в 32, 64 и 128 разрядов, и с алфавитно-цифровой информацией. Он обеспечивал выполнение операций сложения с фиксированной и с плавающей запятой за 520 и 780 нс соответственно, операций умножения 32-разрядных и 64-разрядных чисел за 780 и 1300 нс соответственно, а логических операций за 520 нс. Максимальное быстродействие центрального процессора оценивалось величиной 1,5 млн опер./с. Для построения процессора были использованы интегральные схемы (с транзисторно-транзисторной логикой — ТТЛ), имевшие задержку 10...20 нс на вентиль.

Для обеспечения программной совместимости «Эльбрус-1» с БЭСМ-6 был разработан спецпроцессор (реализующий систему команд БЭСМ-6). Этот спецпроцессор подключался к ВС «Эльбрус-1» вместо одного из ЦП. Спецпроцессор имел быстродействие 3 млн опер./с, выполнял сложение за 240 нс, умножение — за 400 нс и логические операции — за 80 нс.

Оперативная память имела в своем составе от 4 до 32 модулей, причем для группы из 4 модулей предназначался свой локальный коммутатор. Коммутатор обеспечивал связь группы из 4 модулей памяти с 10 ЦП и 4 процессорами ввода-вывода (через локальные коммутаторы). Группа из 4 модулей памяти имела емкость 64 К 72-разрядных слов, модули работали с перекрытием циклов обращения. Оперативная память была реализована на ферритовых сердечниках, ее время цикла составляло 1,2 мкс, а время доступа 0,5 мкс.

В процессорах ввода-вывода были аппаратно реализованы алгоритмы управления периферийным оборудованием, а также те основные алгоритмы операционной системы, которые осуществляли диспетчеризацию ввода-вывода. Максимальная скорость обмена информацией одного ПВВ с оперативной памятью достигала 36 млн байт/с, а скорости обмена данными через быстрый канал (для накопителей на магнитных барабанах и дисков), стандартный канал (для накопителей на магнитных лентах и устройств ввода-вывода) и канал для ППД составляли до 4, 1,3 и 1 млн байт/с соответственно.

В состав программного обеспечения ВС «Эльбрус-1» входили ОС, система программирования, комплекс прикладных и сервисных программ, система телеобработки, контрольные и диагностические программы. Операционная система позволяла ВС работать в режимах пакетной обработки, разделения времени и терминальной обработки. Она осуществляла динамическое распределение всех ресурсов ВС; управляла работой процессоров и обеспечивала их синхронизацию; реализовывала виртуальную память и расчленение физической памяти на сегменты (без жестких границ и с точностью до слова). Операционная система обеспечивала также автоматическую реконфигурацию ВС, восстановление файлов и перезапуск задач и системы (в целях поддержки надежности). Общий объем ОС и транслятора с автокода «Эльбрус» составлял всего лишь 250 тыс. слов.

Система программирования ВС «Эльбрус-1» включала в себя следующие языки: ALGOL 60, ALGOL 68, FORTRAN, COBOL, PL/1, SIMULA 67, PASCAL и автокод «Эльбрус». Она обеспечивала также: отладку программ на исходном языке; диагностику ошибок при компиляции и исполнении программ; комплексирование и сегментацию программ; повторное вхождение в программы; защиту и редактирование программ и др.

6.4.3. Вычислительная система «Эльбрус-2»

Вычислительная система «Эльбрус-2» (вторая архитектурная генерация семейства «Эльбрус») была сдана Госкомиссии в 1985 г. Максимальная производительность ВС «Эльбрус-2» составляла 125 млн опер./с, емкость оперативной памяти — 160 Мбайт, пропускная способность распределенного коммутатора — 2 Гбайт/с. Для построения данной модели была использована элементная база, которая характеризовалась временем задержки на вентиль 2...3 нс.

Архитектурные особенности ВС «Эльбрус-2»:

- система команд обеспечивала эффективную реализацию автокода «Эльбрус» (являющегося языком высокого уровня);
- часто встречающиеся программные конструкции языка высокого уровня имели аппаратурную поддержку;
- диспетчер внешних объектов был реализован аппаратно в ПВВ;
- система команд и программное обеспечение ППД обеспечивали простую адаптацию к различным вычислительным сетям и внешним объектам;
- допускалось формирование конфигураций ВС, в которых вместо ЦП включался спецпроцессор, совместимый с машиной БЭСМ-6 (с производительностью, превышающей более чем в 6 раз быстродействие указанной ЭВМ).

6.4.4. Перспективы развития семейства вычислительных систем «Эльбрус»

Легко заметить, что в моделях «Эльбрус-1» и «Эльбрус-2» содержатся архитектурные решения, существенно отличающиеся от канонических и приближающие мультипроцессорные ВС к классу распределенных средств обработки информации.

В 1995 г. была создана в единственном экземпляре 16-процессорная ВС «Эльбрус-3», в которой принцип параллелизма получил более глубокое воплощение.

Дальнейшее архитектурное развитие семейства ВС «Эльбрус» было связано с проектированием микропроцессора E2k (E2k — сокращение от Elbrus 2000). Планировалось, что данный микропроцессор будет создан группой компаний «Эльбрус» (организованной в ноябре 1998 г.).

Архитектура E2k ориентирована на работу с длинным командным словом (VLIW — Very Long Instruction Word). В специально отведенных полях команды каждому из параллельно работающих функциональных устройств предписываются свои действия. Предполагается, что существуют компиляторы с языков высокого уровня, которые готовят программы, загружаемые в микропроцессор. Итак, архитектура VLIW является противоположностью суперскалярной, которая в системе команд не предусматривает каких-либо указаний на параллельную обработку внутри процессора. Архитектура VLIW позволяет осуществить параллельную работу в процессоре большого числа функциональных устройств.

Возможности микропроцессора E2k характеризуют следующие показатели:

- тактовая частота — 1,2 ГГц;
- технологические нормы — 0,18 мкм;
- площадь кристалла — 126 мм^2;
- мощность тепловыделения — 35 Вт.

За счет параллелизма архитектуры микропроцессор E2k должен был обладать пиковой производительностью 4,8 GFLOPS. Этот уровень производительности был самым высоким из доступных микропроцессорам, анонсированным к началу 1999 г.

Чтобы оценить возможности микропроцессора E2k полнее, воспользуемся современными технологиями измерения производительности средств микропроцессорной техники. Среди наиболее распространенных имеются наборы тестов корпорации SPEC (Standard Performance Evaluation Corporation), например SPEC 95. Этот набор состоит из двух частей: SPECint95 совокупность тест-программ для оценки быстродействия (в MIPS) микропроцессоров при выполнении операций с фиксированной за-

пятой и SPECfp95 — для определения производительности (в MFLOPS) для операций с плавающей запятой.

Разработчики микропроцессора получили следующую оценку производительности E2k: SPECint95/fp95 = 135/195. Эту оценку уместно сравнить с возможностями самого быстродействующего в 1999 г. микропроцессора Merced фирмы Intel: SPECint95/fp95 = 45/70. Следовательно, можно было ожидать, что микропроцессор E2k будет почти в 3 раза быстрее Merced (частота Merced — 800 МГц, площадь кристалла — 300 мм^2, мощность тепловыделения — 60 Вт). Планировалось, что промышленный выпуск E2k будет налажен в 2002 г. Однако этим планам не суждено было сбыться.

В настоящее время Институтом микропроцессорных вычислительных систем РАН выполняются работы в рамках госзаказа по созданию ВС «Эльбрус-3М». Данная мультипроцессорная ВС будет компоноваться из однокристальных высокопроизводительных микропроцессоров и иметь распределенную общую когерентную память.

6.5. Предпосылки совершенствования архитектуры мультипроцессорных вычислительных систем

Мультипроцессорные ВС — результат развития архитектуры ЭВМ. Они создавались и создаются с целью достижения показателей производительности и надежности, которые были не доступны компьютерам первого-третьего поколений. Первые мультипроцессорные ВС появились еще в середине 1960-х годов. Например, в 1955 г. фирмой IBM была создана специализированная дуплексная система SAGE (Semi-Automatic Ground Equipment, Полуавтоматическое наземное оборудование) для обработки информации в целях противовоздушной обороны США (такие системы располагались по периметру Северной Америки). В 1958 г. была создана дуплексная система SABRE для резервирования билетов на американских авиалиниях. Она представляла собой комплекс из двух машин IBM 7090 и множества терминалов, в котором одна из ЭВМ составляла горячий резерв.

Мультипроцессорные ВС начала 1960-х годов представляли собой композицию из нескольких ЭВМ, процессоры которых имели доступ к общей внешней памяти на магнитных лентах или дисках. Позднее в системах в качестве общего ресурса стала выступать оперативная память. К 1970-м годам мультипроцессорные ВС приобретают архитектуру, в основе которой лежит канонический мультипроцессор (см. § 6.1, рис. 6.1).

Мультипроцессорные ВС как класс средств обработки информации в архитектурном плане совершеннее конвейерных и матричных ВС. Они яв-

ляются носителями MIMD-архитектуры и основаны на модели коллектива вычислителей.

Производительность, недоступная ЭВМ, достигается в мультипроцессорных ВС благодаря параллелизму (одновременной работе множества процессоров). Программируемость структуры или адаптируемость систем под структуры и параметры реализуемых алгоритмов достигается средствами коммутации (например такими, как единый матричный коммутатор, схемы голосования, распределенный коммутатор). Эти средства коммутации позволяют задать маршруты передачи информации при взаимодействиях между основными элементами мультипроцессора, т. е. процессорами и модулями (или областями) памяти. Однородность проявляется в идентичности процессоров (центральных, ввода-вывода, передачи данных) и модулей памяти и в регулярности структуры средств коммутации.

В мультипроцессорных ВС принципы модели коллектива вычислителей реализованы далеко не с исчерпывающей полнотой. Причиной этого является единый ресурс, через который взаимодействуют средства обработки и хранения информации. Единый ресурс ограничивает возможности масштабирования, наращивания производительности и емкости памяти, снижает надежность ВС в целом.

В сравнении с системами, использующими каноническую структуру мультипроцессора (см. рис. 6.1), архитектурно более совершенными и более технологичными будут ВС, базирующиеся на модифицированном мультипроцессоре (рис. 6.5). В усовершенствованном мультипроцессоре в качестве единиц ресурсов выступают не элементарные процессоры (ЭП), а более мощные средства — элементарные машины (ЭМ). Каждая ЭМ включает в свой состав ЭП и локальную память (ЛП), взаимодействие между ЭП и ЛП осуществляется через локальный коммутатор (ЛК). Допустимы модульная организация локальной памяти ЭМ и, более того, использование нескольких процессоров в одной ЭМ. Связь между ЭМ реализуется через общий коммутатор. В простейшем случае функции этого коммутатора могут быть пре-

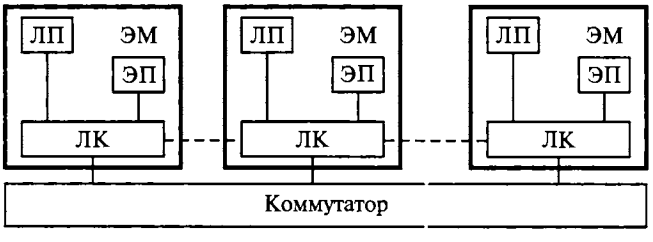


Рис. 6.5. Модифицированная функциональная структура мультипроцессора:

ЛП — локальная память; ЭМ — элементарная машина; ЛК — локальный коммутатор; ЭП — элементарный процессор

дельно упрощены и возложены на линию связей между локальными коммутаторами ЭМ (см. пунктирную линию на рис. 6.5). В более совершенных мультипроцессорах общий коммутатор может быть основан на принципе модульности. В архитектурно развитых ВС такой коммутатор может быть распределенным (когда его модули рассредоточены по ЭМ). Тогда такие ВС не будут различимы с распределенными системами (т. е. с ВС, полностью основанными на модели коллектива вычислителей).

Средства обработки информации, базирующиеся на мультипроцессоре с модифицированной функциональной структурой, могут быть названы *вычислительными системами с виртуальной общей (разделяемой) памятью* (Virtual Shared Memory). Подчеркнем, что в данных мультипроцессорных ВС общая память как таковая отсутствует. Существуют лишь локальные памяти в элементарных машинах ВС, и к каждой из них имеет непосредственный доступ только свой ЭП. Однако очевидно и то, что средства коммутации ВС «обобществляют» локальные памяти машин и предоставляют доступ любой ЭМ к локальным памятям остальных ЭМ.

Современные мультипроцессорные ВС — системы с массовым параллелизмом (MPP Systems). Примерами MPP-систем могут служить Cray T3D и Cray T3E, которые берут свое начало от канонических конвейерных ВС (см. гл. 4). Массовым параллелизмом характеризуются и модели семейств Connection Machine и nCube, являющиеся результатом эволюции классических матричных ВС (см. гл. 5). Количество элементарных процессоров в MPP-системах достигает 10^5

Развитие мультипроцессорных ВС идет в направлении не только «распределенности» (памяти, коммутатора и средств управления), но и все более полного воплощения принципов программируемости и однородности. Далее будут рассмотрены американские академические системы, в которых глубже проявляются свойства коллектива вычислителей и которые в своем архитектурном развитии вплотную подошли к распределенным ВС с программируемой структурой (см. гл. 7 и [5]).

6.6. Вычислительная система Cm*

Вычислительная система Cm* была разработана Университетом Карнеги—Меллона и относилась к микроВС. Целью разработки являлось создание ВС из сравнительно большого числа (до 100) микропроцессоров (построение микроВС) и исследование аппаратурно-программных решений в области архитектуры средств на базе БИС. Существенное внимание было уделено вопросам обеспечения надежности ВС и снижения их показателя стоимость/производительность. При построении ВС были исполь-

зованы простые микропроцессоры LSI-11 фирмы DEC, совместимые с мини-ЭВМ PDP-11.

6.6.1. Архитектура микроВС Cm*

Система Cm* (рис. 6.6–6.8) в сравнении с системой C.mmp (см. рис. 6.2) приобрела заметные архитектурные усовершенствования, в ней полнее были реализованы принципы модели коллектива вычислителей. Архитектура ВС Cm* стала более близкой к архитектуре ВС с программируемой структурой (см. гл. 7 и [5]). В самом деле, в микроВС Cm* пара «элементарный процессор — локальная память» выполняла функции вычислительного модуля (Computer Module), а вычислительный модуль в совокупности либо с контроллером К отображения адресов (рис. 6.6, *а*), либо с локальным коммутатором ЛК (рис. 6.6, *б*) реализовывали функции ЭМ. В первом случае ЭМ выглядела как двухполюсник, а во втором — как многополюсник (не менее двух полюсов). В состав ЭМ могли включаться внешние устройства (со своими контроллерами).

Взаимодействие между вычислительными модулями осуществлялось через «распределенный коммутатор» — сеть связи, образуемую подсоединением контроллеров отображения адресов к межмодульным шинам (см. рис. 6.7) или (и) отождествлением полюсов локальных коммутаторов различных элементарных машин (см. рис. 6.8). Контроллер — это специальный процессор, который выполнял все функции, связанные с передачей сообщений. Локальный коммутатор имел простую структуру, обеспечивающую параллельную передачу слова между процессорами.

Отметим характерные особенности архитектуры ВС Cm*.

Реконфигурируемость — способность микроВС к априорной адаптации своего состава и структуры сети межмашинных связей к конкретной области применения.

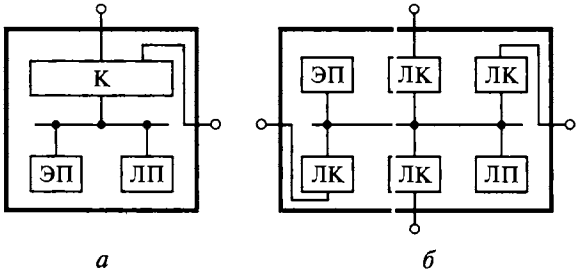


Рис. 6.6. Элементарная машина системы Cm*:

а — на базе контроллера; *б* — на базе локального коммутатора; К — контроллер; ЭП — элементарный процессор; ЛК — локальный коммутатор; ЛП — локальная память

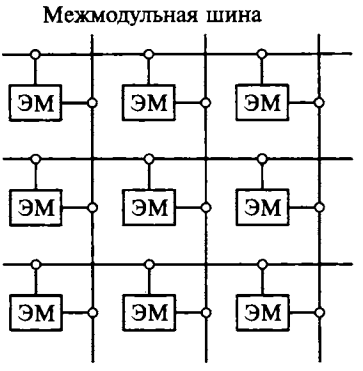


Рис. 6.7. Каноническая структура системы Cm*:
ЭМ — элементарная машина

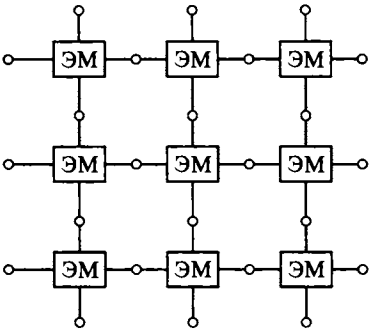


Рис. 6.8. Фрагмент двумерной структуры системы Cm*:
ЭМ — элементарная машина

Масштабируемость (наращиваемость) — способность микроВС Cm* к развитию в целях увеличения производительности, объема памяти и полосы пропускания каналов связи. В вычислительной системе нет принципиальных ограничений на число ЭМ и число связей между ними.

Общедоступность и распределенность памяти. Память системы Cm* состояла из общей памяти и локальной памяти элементарных машин. Вся память системы была потенциально доступна для всех процессоров. За счет наращивания числа ЭМ имелась принципиальная возможность обеспечить превалирование по емкости локальной памяти всех ЭМ сравнительно с общей памятью. В этом смысле можно считать память системы Cm* виртуально общей и распределенной.

«Локальность» программы — свойство, положенное в основу архитектуры микроВС Cm*. Эффективное функционирование системы достигалось, если большая часть программы и данных, с которыми работала каждая ЭМ, хранилась в ее локальной памяти. Для системы Cm* коэффициент обращения к локальной памяти, определяемый отношением «число обращений к локальной памяти/число обращений к общей памяти», был равен 0,8–0,9. Ясно, что «локальность» программы могла быть обеспечена при применении методики крупноблочного распараллеливания сложных задач (см. § 3.3).

Эффективность использования ресурсов системы обеспечивалась возможностью организации параллельной обработки исходных данных элементарными машинами и параллельных межмашинных обменов информацией.

Экономичность ВС, т. е. удовлетворительные значения отношения стоимость/производительность — результат применения большого числа недорогих серийных микропроцессоров.

Надежность функционирования микроВС достигалась за счет распределенности ее ресурсов. В системе не было критического ресурса, отказ которого приводил бы к отказу ВС в целом. Аппаратурно-программные средства позволяли «удалять» из структуры неисправные компоненты. Контроль по четности, дистанционное диагностирование и повторение команд обеспечивали обнаружение и исправление не только устойчивых, но и перемежающихся отказов аппаратуры.

Следует отметить, что архитектура системы Cm* допускала формирование структур, например, подобных структуре на рис. 6.8, но в которых в узле устанавливалась не простейшая конфигурация элементарной машины (вычислительный модуль с контроллером), а группа ЭМ (как композиция вычислительных модулей и локальных коммутаторов). Связь между ЭМ группы и с контроллером отображения адреса (общим на группу) осуществлялась специальной шиной. Количество ЭМ в группе не превышало 14. При этом никакие ограничения на топологию межгрупповой сети связи и на количество групп не накладывались. Вычислительная система с такой функциональной структурой являлась суперсистемой, в качестве ее элементов выступали микроВС Cm* В суперсистеме были потенциально допустимы реализация большого набора функций, адаптация под классы решаемых задач и сферы применения с учетом показателей производительности, надежности и экономической эффективности.

6.6.2. Средства обеспечения надежности микроВС Cm*

Для достижения надежного функционирования микроВС Cm* использовались специальные аппаратурно-программные средства. В микроВС Cm* реализован подход, основанный на введении в контроллер отображения адреса специальной аппаратуры — ловушек (Hooks). За счет возможности организации взаимодействия этих ловушек с процессором LSI-11 была обеспечена хорошая программная поддержка при разработке системной аппаратуры и ее диагностировании, а также созданы условия для отладки микропрограмм на реальной аппаратуре.

Ловушка предоставляла микропроцессору LSI-11 (названному hooks-процессором) возможность тщательного исследования и изменения внутреннего состояния контроллера. Она позволяла загружать в управляющую память процессора микропрограмму, считывать значения сигналов на шинах и управлять генератором тактовых импульсов процессора (выполнять операции «пуск», «останов», «один цикл»). Всякий раз при остановке тактового генератора (вследствие либо останова, задаваемого микропрограммой, либо нарушения четности в управляющей памяти или памяти данных) hooks-

процессору выдавался сигнал прерывания. Сложность аппаратуры ловушек оценивалась 10 % от стоимости процессора LSI-11.

Кроме того, в локальный коммутатор были введены регистры, которые предназначались для хранения информации при диагностировании и восстановлении после обнаружения программного и аппаратурного отказов. Почти обо всех отказах, включая отказы, обнаруживаемые контроллером при внешних обращениях, процессор извещался через систему прерывания.

Все внешние обращения к памяти любого вычислительного модуля выполнялись с помощью коммутации сообщений. До получения сигнала подтверждения каждое сообщение хранилось в буфере. По истечении периода ожидания или при получении извещения об ошибке контроллер пытался вновь передать требуемое сообщение, возможно, даже по другому физическому тракту. Пользователь об этом извещался лишь в том случае, если память, к которой делалось обращение, оказывалась изолированной или находящейся в состоянии устойчивого отказа.

Вся память системы была защищена контролем на четность. При нарушении четности адрес блокировался и поступал соответствующий сигнал в контроллер, с тем чтобы он мог повторить обращение к памяти.

Контроль информационных трактов на четность (в сочетании с коммутацией сообщений) обеспечивал обнаружение и исправление ошибки в одном разряде. Сообщение передавалось с вертикальным признаком четности. В узле приема сообщения проверялись горизонтальный и вертикальный признаки четности. В случае одиночной ошибки в результате пересечения двух признаков четности однозначно идентифицировался разряд, содержащий ошибку.

Установлено, что большинство ошибок в системе составляли ошибки обращения к памяти. Поэтому ОС была ориентирована на анализ ошибок отмеченного класса.

6.6.3. Система самодиагностики микроВС Cm*

Контроль и диагностирование свободных от работы вычислительных модулей в микроВС Cm* выполнялся системой самодиагностики, представляющей собой последовательность из четырех диагностических программ.

Диагностическая программа для памяти состояла из 13 тестов, в том числе теста «галоп» (Gallop Test), теста бегущих «0» и «1» (Marching Ones and Zeros) и теста сдвига. Один прогон такой программы для динамической МОП-памяти произвольной выборки емкостью 56 К байт занимал приблизительно 13 мин.

Диагностические программы системы команд и системы прерывания были предназначены для проверки функционирования микропроцессора

LSI-11. Поскольку эти две программы были короткими, то перед переходом к следующей диагностической программе они пропускались несколько раз.

Диагностическая программа системной аппаратуры проверяла, во-первых, регистры и тракты данных локального коммутатора, во-вторых, часть контроллера отображения адреса и содержала несколько тестов для проверки отдельных частей памяти.

Система самодиагностики размещалась в машине-диспетчере (в мини-ЭВМ PDP-11) на магнитной ленте. Среднее время $\bar{t}$ между загрузкой двух очередных диагностических последовательностей в свободные вычислительные модули выбиралось экспериментально. На начальном этапе эксплуатации Cm* было принято $\bar{t}$ = 7 мин. Для такой загрузки использовался последовательный канал связи между вычислительными модулями микроВС Cm* и машиной-диспетчером.

Процесс самодиагностики шел непрерывно и заключался в следующем. Сначала машине-диспетчеру сообщалось о том, что система самодиагностики берет управление на себя. После этого машина-диспетчер рассматривала вычислительный модуль, реализующий программу самодиагностики, как обычный терминал и позволяла ему «войти» в ВС (подобно пользователю) и затребовать в машину-диспетчер отчет о состоянии Cm* в целом. В этом отчете указывалось, какие вычислительные модули находились в рабочем состоянии, какие из них были свободны от работы. Затем машина-диспетчер по запросу от системы самодиагностики подключалась к свободным от работы модулям, загружала в них диагностические программы и подавала команду «пуск». Запросы системы самодиагностики на подключение свободных от работы модулей поступали через $\bar{t}$ единиц времени. В случае увеличения потребности в ресурсах микроВС Cm* со стороны пользователей система самодиагностики отказывалась от диагностирования отдельных модулей.

Диагностические программы загружались поочередно; условием загрузки очередной программы в свободный от работы вычислительный модуль было либо успешное завершение заданного числа прогонов, либо обнаружение отказов после определенного количества прогонов текущей диагностической программы. В последнем случае предпринималась попытка получить от всех диагностических программ максимально возможное количество информации об условиях проявления отказов.

Система самодиагностики была способна реагировать на некоторые ситуации автоматически. Одна из таких ситуаций — зацикливание модуля. Если модуль в течение определенного времени не реагировал на внештатную ситуацию, то производились его повторная загрузка и пуск. Вторая ситуация — превышение времени ожидания — фиксировалась в том случае, если ожидаемый от машины-диспетчера символ не поступал в течение оп-

ределенного времени. При этом печаталось сообщение о превышении времени, а выполняемая задача ставилась в очередь на повторное решение.

Система самодиагностики предоставляла два вида информации:

1) извещение об ошибке в том или ином модуле, печатавшееся при ее обнаружении и содержавшее: обозначение модуля; наименование выполняемой диагностической программы; время, прошедшее после предыдущей ошибки в данном модуле; сведения об ошибке;

2) извещение о состоянии ВС в целом, которое содержало время начала работы системы самодиагностики, время выдачи извещения и сведения о состоянии всех модулей.

Диагностические программы были ориентированы на обнаружение устойчивых отказов в микроВС Cm*. Однако эти программы (с некоторой вероятностью) могли обнаруживать и перемежающиеся отказы. Предложенный способ самодиагностики микроВС Cm*, несмотря на свою гибкость, все же нельзя было назвать вполне отвечающим современным требованиям. Его недостаток состоял в том, что система самодиагностики работала с помощью машины-диспетчера (что являлось узким местом), а не на основе взаимного тестирования элементарными машинами друг друга.

6.6.4. Анализ и модификация архитектуры микроВС Cm*

Статистическая обработка информации по применению системы самодиагностики показала, что среднее время ϑ безотказной работы ЭМ (состоящей из процессора, динамической МОП-памяти емкостью 48 К байт, локального коммутатора и периферийных устройств) составляет $\vartheta = 127{,}7$ ч при $\overline{t} = 7$ мин. Приведенное значение ϑ следует рассматривать как верхнюю оценку для среднего времени безотказной работы ВС. Установлено также, что одновременное появление перемежающихся отказов в нескольких модулях было возможно с вероятностью 0,17, и такие ситуации возникают в среднем через 1000 ч. Отношение интенсивности перемежающихся отказов к интенсивности устойчивых отказов было равно 100:1 при $\overline{t} = 7$ мин.

Очевидно, что вероятностная модель микроВС Cm* существенно сложнее модели для мини-ВС C.mmp, поэтому разработчиками были подвергнуты анализу лишь конкретные конфигурации системы. Количественный и качественный анализ убедил разработчиков ВС из Университета Карнеги—Меллона в том, что с позиций надежности и живучести архитектура системы Cm* более перспективна, чем архитектура системы C.mmp (и конечно, систем семейства Burroughs).

Главным недостатком микроВС Cm* являлось использование нераспределенного диспетчера (мини-ЭВМ PDP-11). Отказ мини-ЭВМ PDP-11

приводил к невозможности реализации функций ОС и, следовательно, к отказу системы как единого аппаратурно-программного ансамбля. Этот недостаток системы Cm*, безусловно, можно было устранить. В самом деле, возможности архитектуры системы Cm* значительны: в частности, она допускала такую модификацию, когда исчезал главный недостаток — единый аппаратурный диспетчер. Мини-ЭВМ PDP-11 могла быть исключена из структуры Cm*, а ОС реализована в распределенном виде, т. е. ее компоненты могли быть размещены в элементарных машинах. Системное ПО микроВС Cm* (даже если не использовать машину-диспетчер) можно было построить на основе стандартного ПО для PDP-11 и LSI-11.

Описанная модификация архитектуры микроВС Cm* могла бы улучшить количественные характеристики микроВС как единого аппаратурно-программного комплекса и, в частности, положительно сказалась бы на надежности и живучести системы. Эта модификация превратила бы системы Университета Карнеги—Меллона в ВС с программируемой структурой (см. гл. 7 и [5, 6]).

6.7. Мультипроцессорные системы со структурно-процедурной организацией вычислений*

Научная школа по многопроцессорным ВС Южного федерального университета выделяет две парадигмы [15, 16]:

- программируемость архитектуры;
- структурно-процедурную организацию вычислений.

Многопроцессорные ВС, основанные на этих парадигмах, характеризуются высокой эффективностью. Первые результаты данной школы были получены еще в 1960-х годах. Так, например, в 1964 г. была построена ВС «Метеор-3», состоящая из 100 цифровых интегрирующих процессоров. Основателем научной школы по многопроцессорным ВС со структурно-процедурной организацией вычислений является А.В. Каляев (1922–2004; академик РАН с 2000 г.).

6.7.1. Программирование виртуальных архитектур в суперкомпьютерах

В области суперВС с массовым параллелизмом остается еще много принципиально важных нерешенных проблем, которые существенно тормозят их развитие. К важнейшим относится проблема обеспечения соответствия

* По просьбе автора данный параграф написан А.В. Каляевым.

архитектуры (точнее, функциональной структуры) параллельной вычислительной системы структуре решаемой задачи. Без решения этих проблем или при неэффективном их решении невозможно получить реальную производительность суперВС с массовым параллелизмом, близкую к пиковой для очень многих классов задач, и, кроме того, трудно обеспечить существенный рост производительности при наращивании числа процессоров.

В проблемно-ориентированных многопроцессорных суперВС эти проблемы решаются относительно легко за счет конструирования специальной архитектуры, соответствующей структуре ограниченного класса задач. Но достигается это ценой того, что другие классы задач с помощью таких ВС не могут быть решены.

Проблема обеспечения адекватности между архитектурами универсальных суперВС и структурами решаемых задач относится к первостепенно важным и далеко нетривиальным. Эта проблема может быть решена только в том случае, если пользователю будет обеспечена возможность быстро программировать, а также автоматически в динамике вычислений перепрограммировать архитектуру универсального суперкомпьютера под структуры решаемых задач. Фактически это означает, что пользователю должна быть предоставлена возможность программировать в рамках реальной универсальной многопроцессорной суперВС виртуальные проблемно-ориентированные системы. Программирование архитектуры виртуальных систем (соответствующих классам решаемых задач) должно осуществляться как в статическом, так и в динамическом режимах (в процессе решения задач). Только в этом случае можно достичь для многих классов задач пиковой производительности универсальной суперВС с массовым параллелизмом и обеспечить линейный рост производительности при увеличении числа одновременно работающих процессоров.

В Научно-исследовательском институте многопроцессорных вычислительных систем (НИИ МВС) Южного федерального университета была разработана концепция многопроцессорных ВС, которая состоит в нижеследующем. В процессе разработки и конструирования суперВС с массовым параллелизмом ее архитектура не формируется окончательно, а остается в определенном смысле незавершенной и открытой. Такую архитектуру можно назвать рамочной, или фрейм-архитектурой (Frame-architecture, рис. 6.9).

Фрейм-архитектура суперВС содержит поля макропроцессоров, макрокоммутаторов, распределенной памяти и интерфейса, архитектура каждого из которых исходно не синтезирована. Наряду с этим во фрейм-архитектуру суперВС входят аппаратно-программные средства управления полями макропроцессоров, макрокоммутаторов и распределенной памяти, которые позволяют программировать виртуальные проблемно-ориентированные многопроцессорные системы с архитектурами, необходимыми для

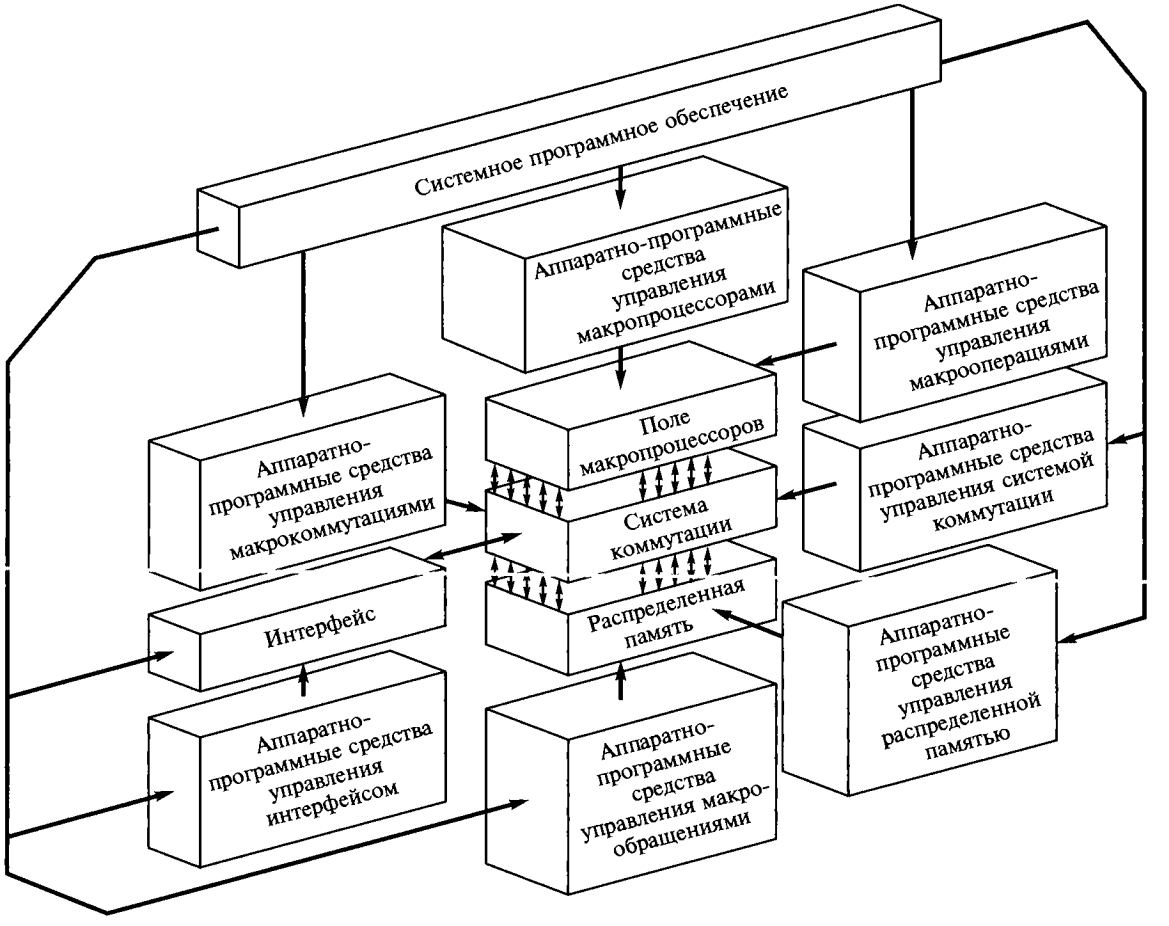


Рис. 6.9. Фрейм-архитектура многопроцессорной ВС

решения конкретных классов задач. Таким образом, фрейм-архитектура основывается на аппаратных и программных средствах, которые дают пользователю возможность очень быстро (как до начала решения задачи, так и в процессе ее решения) программировать и настраивать архитектуру универсальной ВС.

Очевидно, что в суперВС с массовым параллелизмом легко настраивать виртуальные архитектуры, адекватные структурам решаемых задач. Настройка архитектуры универсальной суперВС включает программирование:

- информационных каналов связи между параллельно работающими процессорами;
- поля коммутаторов (системы коммутации);
- структуры каждого из процессоров для параллельного выполнения макроопераций;
- наборов макроопераций;
- структуры распределенной памяти;
- процедуры параллельного бесконфликтного доступа процессоров к распределенной памяти;
- внутреннего машинного языка высокого уровня;
- каналов внешних связей;
- структуры распределенного системного программного обеспечения.

Для всех этих процедур программирования и для аппаратной настройки многопроцессорных суперВС на необходимые архитектуры в НИИ МВС Южного федерального университета были проработаны соответствующие методы и механизмы, которые проверены в реальных системах.

6.7.2. Структурно-программируемая макропроцессорная элементная база

СуперВС с массовым параллелизмом, программируемой архитектурой и структурно-процедурной организацией вычислений основана на новой макропроцессорной элементной базе. Такая элементная база содержит три основных компонента:

- многоканальный макропроцессор с программируемой на выполнение макроопераций структурой;
- многоканальный макрокоммутатор, программируемый на выполнение операций макрокоммутаций;
- макропамять — элемент распределенной памяти с параллельными каналами доступа, программируемый на реализацию операций макрообращений.

Макропроцессор является основным функциональным и конструктивным элементом суперВС с массовым параллелизмом и программируе-

мой архитектурой. Он предназначен для выполнения макроопераций (из их базового набора, либо дополнительно вводимых). Функциональная структура макропроцессора (рис. 6.10) представляется композицией из множеств элементарных процессоров $\{ЭП_0, ЭП_1, ..., ЭП_{n-1}\}$ и модулей памяти $\{МП_0, МП_1, ..., МП_{m-1}\}$, коммутатора (К) и блока управления (БУ). Следовательно, макропроцессор — это мультипроцессор с канонической функциональной структурой (см. рис. 6.1).

Макропроцессор обладает возможностью программирования своей архитектуры. Это достигается с помощью блока управления и коммутатора. Макропроцессор может быть настроен на выполнение любой макрооперации из их базового набора в соответствии с макрокомандой, которая поступает на управляющий вход. Макропроцессор выполняет любую макрооперацию параллельным структурным методом (вместо последовательного процедурного метода, который используется в обычных классических микропроцессорах).

Любые макрооперации реализуются в макропроцессорах суперВС в одинаковые интервалы времени. Это обеспечивает согласованную работу всех

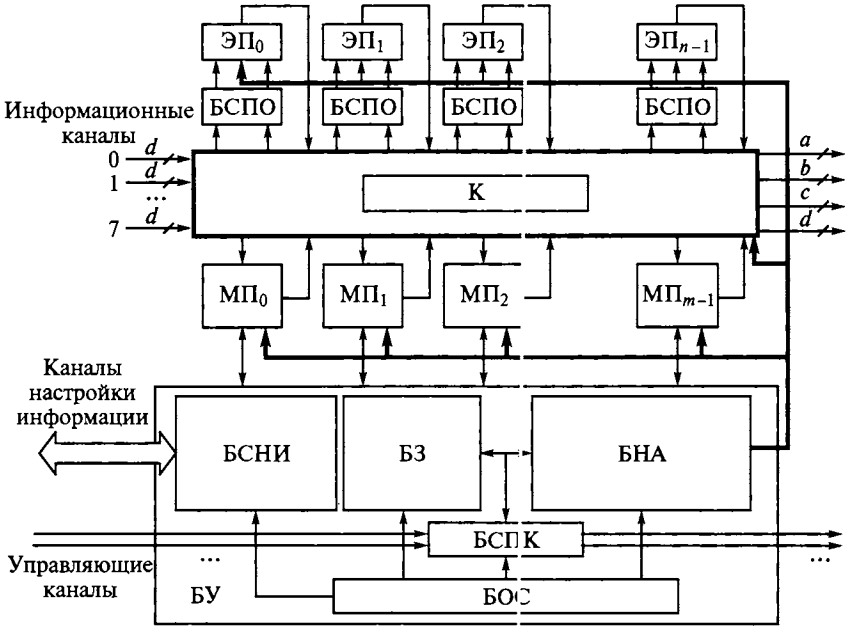


Рис. 6.10. Функциональная структура макропроцессора:

ЭП — элементарный процессор; МП — модуль памяти; БУ — блок управления; БСПО — блок синхронизации потоков операндов; БСПК — блок синхронизации потоков кадров; К — коммутатор; БСНИ — блок сопряжения с носителями информации; БЗ — блок заданий; БНА — блок настройки архитектуры; БОС — блок ОС

макропроцессоров суперВС и высокую скорость обмена данными между ними и, как следствие, высокую системную производительность. Такая функциональная структура макропроцессора позволяет решить очень трудные и важные проблемы функционирования суперВС с массовым параллелизмом, в том числе совпадения времени обработки информации в различных параллельно работающих макропроцессорах. Решение этой проблемы необходимо для синхронизации обработки данных в различных макропроцессорах и для организации быстрых одновременных обменов информацией между очень большим числом параллельно работающих макропроцессоров в суперсистеме.

Механизм синхронизации в суперВС реализуется коллективно макропроцессорами, в частности их блоками синхронизации потоков операндов (БСПО) и кадров (БСПК). Под «кадром» разработчики понимают текущую виртуальную структуру, запрограммированную в макропроцессоре под структуру решаемой задачи.

Макропроцессор имеет достаточно большое количество входных и выходных каналов (линков), которые обеспечивают для него прямой высокоскоростной обмен информацией через коммутационную среду, как с любыми другими макропроцессорами, так и с элементами распределенной памяти суперВС. Макропроцессор реализуется на одном кристалле с высокой степенью интеграции.

Следует отметить, что макропроцессор имеет также память для хранения данных и простых арифметических, логических, и коммутационных операций (которые необходимы для выполнения макроопераций). Наконец, он содержит управляющие блоки для выполнения макрокоманд, поступающих на его вход, и интерфейс, который поддерживает операции входных и выходных линков.

Макропроцессор может работать в двух режимах. В первом режиме он выполняет на каждом последовательном шаге вычислений одну и ту же макрооперацию над данными поступающего потока. Во втором режиме макропроцессор реализует на каждом временном шаге вычислений новую макрокоманду и, следовательно, выполняет поток различных макроопераций, каждая из которых обрабатывает пакет данных, поступивший на этом шаге.

Результаты обработки данных макропроцессор может передавать через множество выходных линков и через коммутационную среду суперВС как на входы других макропроцессоров, так и на входы модулей распределенной памяти. Кроме этого каждый макропроцессор может вырабатывать макрокоманды для других макропроцессоров системы.

В случаях, когда при решении задач на суперВС стандартных макроопераций базового набора недостаточно, структура макропроцессора может быть запрограммирована (программистом или автоматически) и на реализацию специальных макроопераций. Для этого пользователь имеет возмож-

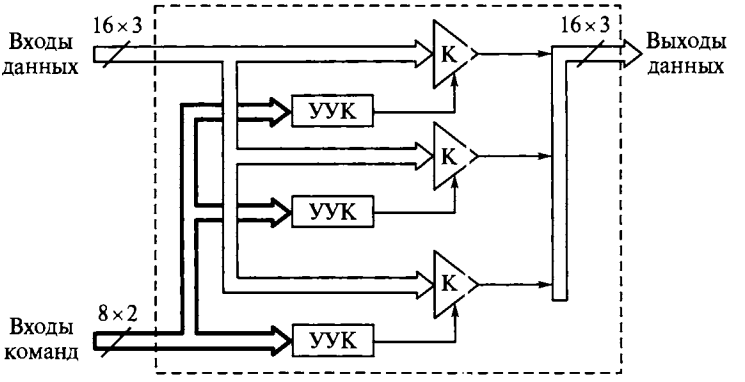


Рис. 6.11. Функциональная структура макрокоммутатора:
К — коммутатор; УУК — устройство управления коммутатором

ность синтезировать и ввести в память макропроцессора набор необходимых специальных макроопераций, ориентированных на класс проблем, в решении которых заинтересован пользователь.

Макрокоммутатор (рис. 6.11) — однокристальный цифровой программируемый переключатель, который может реализовать программным методом базовый набор операций многоканальных коммутаций (или кратко «макрокоммутаций»). Макрокоммутатор обеспечивает при этом программную настройку соединений любого своего входного линка с любой группой его выходных линков. Программирование макрокоммутатора выполняется в соответствии с макрокомандой, которая поступает на его управляющий вход. Стандартные макрокоммутации хранятся в памяти макрокоммутатора, что обеспечивает быструю перестройку его структуры с одной макрокоммутации на другую. Макрокоммутатор имеет достаточно большое количество входных и выходных линков.

Макрокоммутаторы необходимы для создания универсальной коммутационной среды, которая дает возможность настраивать любые прямые каналы передачи данных в многопроцессорной системе. Данная среда обеспечивает быструю настройку каналов между макропроцессорами суперВС для передачи потоков как данных, так и макрокоманд.

Макропамять является многоканальной памятью. Она используется для формирования параллельной распределенной памяти суперВС. Макропамять позволяет реализовать базовый набор макроопераций обращения макропроцессоров суперВС к необходимой информации.

Макропамять (рис. 6.12) состоит из сверхбольшой интегральной микросхемы мультиконтроллера и нескольких стандартных кристаллов памяти (П). Мультиконтроллер представлен множеством контроллеров (Кон), устройством управления контроллерами (УУКон) и памятью программ (ПП).

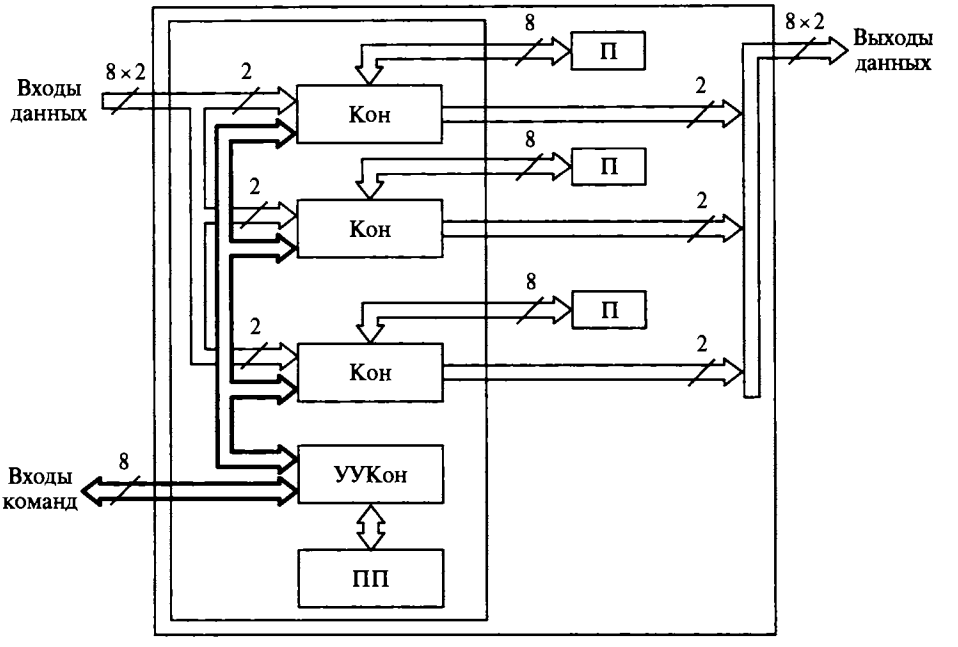


Рис. 6.12. Функциональная структура макропамяти:

Кон — контроллер; УУКон — устройство управления контроллерами; ПП — память программ; П — память

Следует отметить, что множество памятей используется для хранения массивов данных, подлежащих обработке, а память программ — для запоминания системных и пользовательских программ. Макропамять содержит несколько элементарных страниц.

Распределенная память суперВС (формируемая из кристаллов макропамяти) обеспечивает параллельный доступ макропроцессоров к требуемым данным без задержек, очередей и конфликтов. Каждая макропамять может с помощью коммутационной среды обмениваться информацией с другими кристаллами распределенной памяти многопроцессорной суперВС.

Таким образом, основными микросхемами, предназначенными для создания суперВС с массовым параллелизмом и программируемой архитектурой, являются макропроцессор, макрокоммутатор и макропамять. Все эти базовые компоненты могут быть реализованы как в виде заказных монолитных микросхем, так и на основе ПЛИС-технологии (ПЛИС — Программируемая Логическая Интегральная Схема; EPLD — Electrically Programmable Logic Device). Опыт разработки макропроцессоров, макрокоммутаторов и макропамяти на ПЛИС-технологии показывает, что такая элементная база обладает достаточно высокой производительностью и архитектурной гибкостью.

6.7.3. Модульная организация суперВС с массовым параллелизмом и программируемой архитектурой

СуперВС целесообразно строить на основе унифицированных базовых модулей. Базовый модуль представляет собой функционально завершенный параллельный многопроцессорный супервычислитель, построенный на основе макропроцессорной элементной базы. Ясно, что такой модуль обладает всеми свойствами и характеристиками многопроцессорной системы с программируемой архитектурой.

Базовый модуль (рис. 6.13) имеет незавершенную (рамочную) фрейм-архитектуру, которая может программироваться на любую виртуальную архитектуру, адекватную структуре решаемой задачи. В состав базового мо-

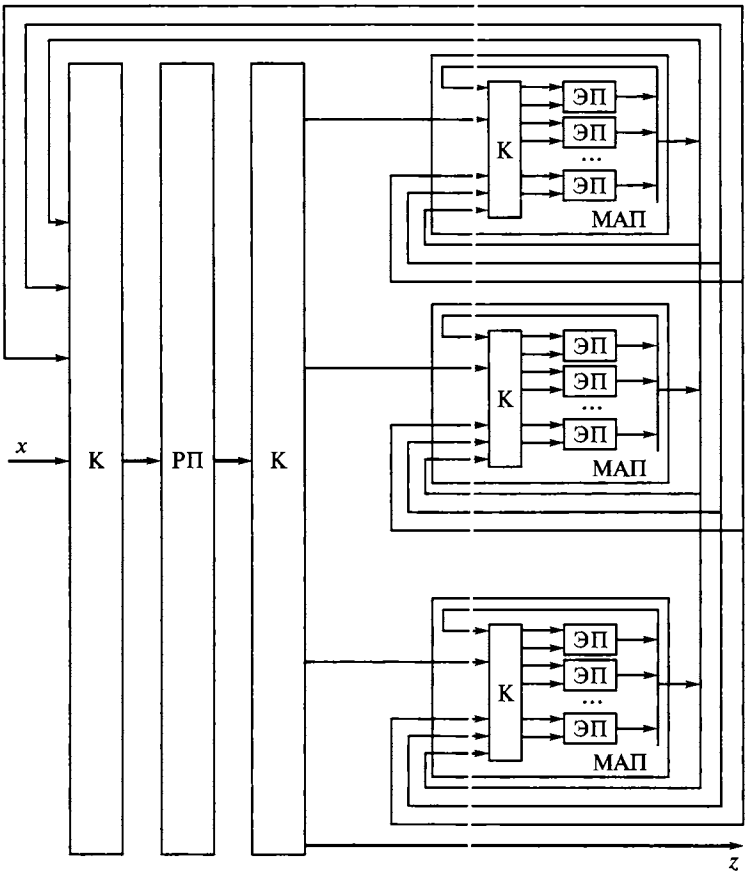


Рис. 6.13. Функциональная структура базового модуля суперВС:

МАП — макропроцессор; РП — распределенная память; К — коммутатор; ЭП — элементарный процессор

дуля входят несколько макропроцессоров (МАП), распределенная память (РП), коммутаторы (К) и коннекторы. Коммутаторы служат для динамической перестройки топологии связей между макропроцессорами данного базового модуля, а также топологии связей макропроцессоров данного базового модуля с макропроцессорами других базовых модулей. Коннекторы обеспечивают однотипное соединение модулей между собой в пределах суперВС.

Таким образом, любая конфигурация многопроцессорной ВС с программируемой архитектурой представляется совокупностью базовых модулей, взаимодействующих между собой через коммутационную среду (распределенную по модулям) и сеть коннекторов. Для операционной системы базовые модули являются программно-неделимыми единицами, число которых в многопроцессорной ВС может варьироваться в зависимости от потребностей пользователя. Принцип модульной наращиваемости ВС и принцип функциональной законченности и программной неделимости базового модуля обеспечивают решение пользовательских задач на любом количестве базовых модулей и в любом их сочетании. При этом параллельная программа является параметризуемой относительно указанных параметров. Модульная наращиваемость позволяет осуществить ресурсонезависимое программирование задачи, при котором выход из строя одного или нескольких базовых модулей не приводит к выходу из строя всей ВС, а лишь замедляет решение задачи. Кроме того, принцип модульной наращиваемости позволяет ВС функционировать в многозадачном режиме с разделением ресурсов, когда ОС в соответствии с потоком входных заданий выделяет разным пользователям группы базовых модулей для решения их задач. В системе с программируемой архитектурой гарантируется линейная зависимость ее производительности от числа модулей.

6.7.4. Структурно-процедурная организация вычислений в суперВС

В некоторых известных ВС с массовым параллелизмом используется принцип мультипроцедурного параллелизма. В соответствии с этим принципом распараллеливание осуществляется по элементам структуры данных, а каждый процессор системы работает по своей процедуре — ветви параллельной программы. Обмен данными между ветвями программы (или процессорами ВС) реализуется с помощью соответствующих специальных процедур. При этом структура параллельного процесса, происходящего в многопроцессорной системе, может не совпадать со структурой решаемой задачи или процесса, имеющего место в реальной моделируемой системе; это бывает, когда ВС не обладает способностью к программированию своей архитектуры.

Любую решаемую задачу или любой сложный процесс, происходящий в реальной моделируемой системе, можно описать с помощью некоторого множества математических и логических зависимостей или в форме некоторого графа $G^*\left[V^*, D^*, P^*, Z^*, X^*, Y^*\right]$. Граф G^* содержит множество вершин V^*; множество дуг D^*; множество процессов P^*, происходящих в вершинах; множество информационных потоков Z^*, вырабатываемых процессами из P^* и передаваемых по дугам D^*; а также множества X^* и Y^* соответственно входных и выходных информационных потоков.

При моделировании процессов, происходящих в моделируемой системе с помощью многопроцессорной ВС, в последней организуется соответствующий вычислительный процесс, который описывается в большинстве случаев совершенно другим графом $G[V, D, P, Z, X, Y]$. В последнем графе множество V вершин соответствует множеству процессоров ВС. Множество дуг D отражает множество физических или виртуальных каналов коммуникаций между процессорами. Множество процессов P реализуется в форме процедур обработки данных в процессорах из V. Далее, Z является множеством информационных потоков, передаваемых от одного процессора к другому через коммуникационные каналы D. Наконец, X и Y представляют собой множества соответственно входных и выходных информационных потоков вычислительной системы.

Как правило, если многопроцессорная ВС имеет жесткую архитектуру, то графы G и G^* существенно отличаются один от другого. Это объясняется тем, что в такой ВС нельзя организовать вычислительный процесс, который совпадал бы как процедурно, так и структурно с процессом в моделируемой системе. В результате для моделирования некоторой системы на ВС с жесткой архитектурой приходится организовывать некоторый вычислительный процесс, который моделирует систему процедурно, а структурно с ней не совпадает. Процесс описывается графом G, существенно отличающимся от графа G^* реальной моделируемой системы.

В большинстве случаев граф G многопроцессорной ВС с жесткой архитектурой значительно более сложен, чем исходный граф G^* моделируемой системы или решаемой задачи, и имеет более сложную структуру внутренних коммуникационных каналов D, чем система дуг D^* в графе G^*. Кроме того, процессы множества P, реализуемые в процессорах ВС, выполняются асинхронно и плохо согласованы между собой во времени, в то время как реальные процессы P^* в моделируемой системе развиваются параллельно и синхронно. Все это приводит при процедурной организации вычислений к существенным потерям времени и за счет операций распределения заданий и

процедур между процессорами ВС, и из-за усложнения процедур обмена информацией между процессорами и необходимости их синхронизации, и вследствие потерь времени при передаче данных через промежуточные процессоры; и из-за очередей и конфликтов при обращении процессоров к данным, хранящимся в памяти, и по другим причинам. В результате производительность многопроцессорной системы при процедурной организации вычислений резко уменьшается по сравнению с пиковой.

Возможность организации виртуальных проблемно-ориентированных конфигураций, адекватных структурам решаемых задач, позволяет избавиться в многопроцессорных системах с программируемой архитектурой от перечисленных выше недостатков (за счет перехода к структурной или структурно-процедурной организации вычислений). При структурной организации вычислительного процесса распараллеливание осуществляется по операционным вершинам информационного графа задачи. При этом имеет место соответствие между вершинами информационного графа и процессорами ВС, через которые в соответствии с множеством информационных связей (дуг графа) циркулируют элементы данных. Ясно, что при достаточной трудоемкости задач производительность ВС при структурной реализации параллелизма будет значительно выше, чем в случае процедурной. Если структура ВС полностью покрывает информационный граф (число процессоров равно количеству операционных вершин, а число дуг — числу каналов коммутационной системы), задача может быть решена структурно. Если оборудования недостаточно для структурной реализации вычислений, задача может быть представлена в структурно-процедурном виде.

При структурно-процедурной организации вычислений информационный граф задачи разбивается на функционально законченные фрагменты (кадры), каждый из которых аппаратно (структурно) реализуется в ВС с программируемой архитектурой (рис. 6.14). При этом имеет место соответствие вершин информационного графа задачи элементарным процессорам ВС. Множество дуг информационного графа реализуется коммутационной средой. Настройка на кадры производится единой управляющей программой, что обеспечивает фон-неймановский детерминизм вычислительной процедуры. Процедура представляет собой последовательность вызовов кадров. Вычисления в теле кадра выполняются по принципу управления потоком данных и не требуют синхронизации. Организация вычислительного процесса в кадре осуществляется по принципу машины потока данных. Можно сказать, что ВС с программируемой архитектурой является своеобразным гибридом фон-неймановской ЭВМ и машины потоков данных и сочетает в себе их достоинства: детерминизм программирования, присущий фон-неймановским ЭВМ, и высокую реальную производительность, характерную для конвейерных машин потоков данных.

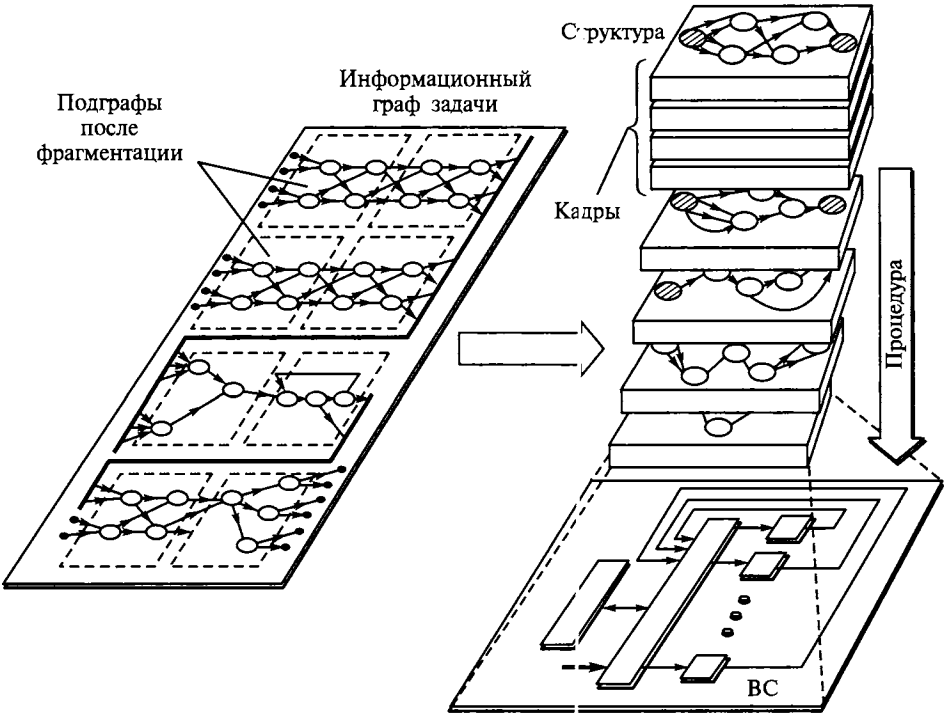


Рис. 6.14. Процесс преобразования задачи в структурно-процедурную форму

В системе с программируемой архитектурой и со структурно-процедурной организацией вычислений ЭП могут соответствовать операционным вершинам информационного графа задачи (или могут объединяться в макропроцессоры, которые соответствуют операционным вершинам графа). Множество информационных связей между вершинами графа отображается на множество каналов коммутационной среды и множество параллельно-последовательных процедур обращения к каналам распределенной памяти.

6.7.5. Аппаратно-программные средства поддержки синтеза виртуальных конфигураций суперВС

СуперВС с программируемой архитектурой является, с одной стороны, универсальной, поскольку она может быть использована для решения сложных задач, составляющих достаточно широкий класс. С другой стороны, такая система является и проблемно-ориентированной, так как она позволяет достичь максимально высокой производительности (близкой к пиковой) при решении отдельных задач. Это поддерживается тем, что ВС мо-

жет быть запрограммирована адекватно структуре решаемых задач. Для возможности настройки архитектуры суперВС оснащается соответствующими средствами. В состав суперВС включаются аппаратные и программные подсистемы, которые гарантируют пользователю возможность программировать и настраивать его архитектуру таким образом, чтобы можно было получить любые виртуальные проблемно-ориентированные конфигурации, архитектура которых адекватна соответствующим структурам решаемых задач.

В состав многопроцессорной ВС (см. рис. 6.9) входят аппаратно-программные подсистемы управления макрооперациями, макрокоммутациями и макрообращениями. Кроме того, необходимы еще три подсистемы, а именно: подсистемы программирования и настройки поля макрокоммутаторов и распределенной памяти. В число этих подсистем входят: аппаратно-программная подсистема организации в поле макропроцессоров вычислительного графа решаемой задачи или его подграфов (кадров); аппаратно-программная подсистема программирования и настройки в коммутационной структуре в соответствии с кадром прямых каналов связи между макропроцессорами; аппаратно-программная подсистема управления организацией структуры распределенной памяти; и, наконец, аппаратно-программная подсистема управления организацией интерфейса. Все эти подсистемы, в свою очередь, находятся под управлением распределенного системного программного обеспечения.

6.7.6. Принципы синтеза суперВС с массовым параллелизмом и программируемой архитектурой

Исходя из вышесказанного, в основу синтеза ВС с массовым параллелизмом и программируемой архитектурой должен быть положен ряд фундаментальных принципов, обеспечивающих выполнение всех современных требований.

Принцип адекватности структуры задачи и архитектуры вычислительной системы. Этот принцип требует максимально адекватного отображения внутренней структуры любой решаемой задачи или любого моделируемого объекта в архитектуру многопроцессорной ВС с массовым параллелизмом.

Принцип использования внутренней параллельности решаемой задачи. Принцип обеспечивает максимально широкое использование внутренней естественной параллельности решаемой задачи или моделируемого объекта при организации параллельных вычислений в ВС с массовым параллелизмом.

Принцип обмена информацией без посредников. Согласно этому принципу, обеспечивается преимущественно прямая передача информации от

процессора к процессору по прямым каналам связи без промежуточных процессоров, памяти или других элементов многопроцессорной системы.

Принцип калиброванного времени выполнения макроопераций. Этот принцип призван поддержать выполнение разнотипных макроопераций (крупных операций), реализуемых в разных параллельно работающих процессорах ВС, в одинаковые промежутки времени и тем самым обеспечить согласованность работы всех процессоров.

Принцип бесконфликтного параллельного доступа к информации. Принцип обеспечивает условия для параллельного, бесконфликтного доступа без очередей всех одновременно работающих процессоров системы к необходимой информации, как хранящейся в распределенной памяти, так и к вырабатываемой другими процессорами.

Принцип структурно-процедурного кадрового процесса вычислений. Данный принцип обеспечивает организацию процесса вычислений в многопроцессорной ВС на основе информационного графа задачи в кадровой структурно-процедурной форме. При этом формирование кадров осуществляется путем такого разрезания графа задачи на непересекающиеся подграфы, когда каждый подграф заполняет весь процессорный ресурс ВС и в каждом кадре осуществляются параллельно-конвейерные структурные вычисления, а обработка всех кадров информационного графа задачи ведется в виде последовательной процедуры с обменом информацией между кадрами.

Принцип модульности архитектуры. Этот принцип предписывает синтез архитектуры ВС с массовым параллелизмом на основе однотипных стандартных модулей, каждый из которых, в свою очередь, является многопроцессорной системой в минимальной конфигурации. Объединение модулей осуществляется стандартными однотипными коннекторными соединениями. При объединении ресурсы всех модулей (включая процессорные поля, коммутационные структуры, распределенную память, интерфейс и ОС) порождается многомодульная ВС с массовым параллелизмом и программируемой архитектурой.

Принцип масштабируемости структурно-процедурных кадровых программ. Принцип обеспечивает эффективное выполнение структурно-процедурных программ с максимальным использованием имеющихся вычислительных ресурсов при любых модульных конфигурациях многопроцессорных ВС с массовым параллелизмом и программируемой архитектурой.

6.7.7. Реализация многопроцессорных вычислительных систем со структурно-процедурной организацией вычислений

Научно-исследовательский институт многопроцессорных вычислительных систем Южного федерального университета в 2007 г. завершил

разработку модульно-наращиваемой многопроцессорной системы с программируемой архитектурой. Эта система компонуется из функционально и конструктивно законченных элементов — вычислителей. Каждый такой вычислитель состоит из двух базовых модулей (см. рис. 6.13) и имеет следующие характеристики:

- количество микропроцессоров — 32;
- число элементарных процессоров — 512;
- емкость памяти — 1 Гбайт;
- производительность — 50 GFLOPS;
- потребляемая мощность — 0,44 кВт;
- габариты – 482,6×595×265,9 мм^3

Макропроцессор ВС реализован на одной ПЛИС серии Virtex II-Pro фирмы Xilinx.

Любая конфигурация ВС работает под управлением хост-компьютера.

Архитектура многопроцессорной ВС и технология ресурсонезависимого параллельного программирования позволяют выполнять параллельную программу на любом количестве базовых модулей. Создана интегрированная среда программирования, обеспечивающая возможность оперативного создания компонентов адаптирующихся (масштабируемых) параллельных программ.

6.8. Сверхвысокопроизводительные вычислительные системы семейства IBM Blue Gene

Проект создания вычислительной масштабируемой суперсистемы Blue Gene был анонсирован корпорацией IBM в декабре 1999 г. Проект направлен на создание суперВС с производительностью 1 PetaFLOPS и предусматривает поэтапное конфигурирование ряда моделей с производительностью в диапазоне 10...10^3 TeraFLOPS.

Первая поставка простой конфигурации системы — модели IBM Blue Gene/L была осуществлена в 2000 г. в Ливерморскую национальную лабораторию им. Лоуренса (LLNL — Lawrence Livermore National Laboratory). Неоднократное наращивание первоначальной конфигурации ВС превратило ее уже в 2004 г. в самый мощный суперкомпьютер в мире. В 24-й редакции списка 500 наиболее мощных компьютеров мира (Top500) было указано, что конфигурация IBM Blue Gene/L из 32 768 процессорных ядер (16 384 двухъядерных процессоров IBM PowerPC 440 с тактовой частотой 700 МГц) обладала пиковой производительностью 91,75 TeraFLOPS (70,72 TeraFLOPS — на тестовом наборе LINPACK).

Конфигурация IBM Blue Gene/L, эксплуатировавшаяся в 2006–2007 гг. в LLNL (см. 28-ю и 29-ю редакции списка Top500, 2006–2007 гг.; http://www.top500.org),

характеризовалась следующими показателями: число процессорных ядер — 131 072 (двухъядерных процессоров PowerPC 440 — 65 536), пиковая производительность — 367 TeraFLOPS (280,6 TeraFLOPS — на наборе тестов LINPACK).

Наращивание ресурсов IBM Blue Gene/L, произведенное к ноябрю 2007 г. в Ливерморской лаборатории, позволило преодолеть рубеж 0,5 PetaFLOPS (см. 30-ю редакцию Top500). Конфигурация модели IBM Blue Gene/L 2008 г. обладает следующими показателями: число процессорных ядер — 212 992, пиковая производительность — 596,38 TFLOPS (478,2 TFLOPS — на LINPACK), емкость памяти — 53,248 Тбайта, пропускная способность канала «процессор-память» — 5,5 Гбайт/с, энергопотребление — около 3,25 Мегаватт, занимаемая площадь — 4060 квадратных футов (377,2 $м^2$), цена — 208 млн долл.

В списке Top10 — десяти самых высокопроизводительных суперВС мира — модели семейства IBM Blue Gene занимают четыре позиции (2008 г.). В ближайших планах корпорации IBM предусмотрено построение моделей Blue Gene/P и Blue Gene/Q с пиковой производительностью 1 PFLOPS и 3 PFLOPS соответственно. Реализованная конфигурация модели IBM Blue Gene/P, состоящая из 65 536 процессорных ядер (из 32 768 двухъядерных микропроцессоров Power PC 450 с тактовой частотой 850 МГц), обладает пиковой производительностью 222,8 TFLOPS (167,3 TFLOPS — на LINPACK). Эта конфигурация является второй в 30-й редакции Top500 (2007 г.) и самой мощной суперВС Европы. Она установлена в Научно-исследовательском консорциуме FZJ (Forschungszentrum Juelich) в Германии.

6.8.1. Особенности архитектуры IBM Blue Gene

Для интенсификации работ по проекту Blue Gene разработчики IBM вынуждены были пересмотреть свою архитектурную платформу. В результате ими была декларирована «новая» (для IBM, но не для российских разработчиков, см. гл. 3, [5, 6]) архитектурная концепция SMASH (Simple, Many, Self-Healing — простая, множественная и самовосстанавливаемая). Эта концепция ориентирована на использование таких архитектурных и функциональных решений, которые позволяют интегрировать большое число процессоров (до 1 млн, для достижения петафлопсной производительности) и, следовательно, обеспечивают автоматическое устранение проблем, вызванных сбоями.

При разработке данной суперВС принцип масштабирования воплощался на всех уровнях и аппаратурного, и программного обеспечения.

Все модели семейства IBM Blue Gene относятся к классу распределенных систем с массовым параллелизмом. Несмотря на то, что они ориен-

тированы на решение суперсложных задач, заложенные в них решения позволяют характеризовать их архитектуру как MIMD.

Функциональная структура суперВС рассчитана на использование хост-компьютерной системы — множества хост-компьютеров. Эти компьютеры реализуют файловую систему и выполняют следующие функции: анализ функционирования, контроль, диагностику и восстановление суперВС, а также компиляцию и сервисное обслуживание. Выбор хост-компьютеров для IBM Blue Gene определяется областью применения (требуемыми производительностью и полосой пропускания, в частности).

Важной особенностью архитектуры IBM Blue Gene является возможность одновременной работы множества пользователей. Это достигается путем выделения каждому пользователю требуемых ресурсов — подсистемы из необходимого количества вычислительных узлов. Следовательно, в данной суперВС реализована возможность разбиения «пространства» вычислительных узлов на подпространства и доступа к ним пользователей. Выделение подсистем в суперВС выполняется хост-компьютерной системой.

При разработке функциональной структуры суперВС и основных ее элементов значительное внимание уделялось обеспечению надежности (RAS — Reliability, Availability, Serviceability — надежности, готовности, обслуживаемости). При этом свою роль сыграли принципы простоты, однородности и модульности, а также введение избыточности, средств контроля, диагностики и восстановления.

Фактически разработчики суперВС IBM Blue Gene реализовали платформу ВС с программируемой структурой (см. разд. 3.4.2), они достаточно полно воплотили и архитектурные принципы, и принципы технической реализации модели коллектива вычислителей (см. разд. 3.1.1, 3.2.1).

6.8.2. Функциональные узлы IBM Blue Gene

Для формирования конфигураций суперВС IBM Blue Gene используют вычислительные узлы, узлы ввода-вывода информации и сервисные узлы.

Вычислительный узел IBM Blue Gene. Изначально планировалось, что вычислительный узел (Compute Node) будет реализован в виде одной микросхемы, содержащей 32 гигафлопсных процессора с DRAM-памятью (DRAM — Dynamic Random Access Memory — динамическая память с произвольной выборкой). Такие чипы должны были использоваться для формирования модели IBM Blue Gene/C (C — Cyclops – циклоп).

На практике при формировании ВС нашли применение не сверхплотно упакованные 32-процессорные чипы, а двухъядерные 64-разрядные мик-

ропроцессоры IBM PowerPC 440. Именно эти микропроцессоры использованы в вычислительных узлах конфигураций модели IBM Blue Gene/L (L — возможно от Livermore).

Главная особенность микропроцессора IBM PowerPC 440 заключается в том, что он создан на основе технологии «система-на-кристалле» (System-on-Chip), позволившей объединить в единой проблемно-ориентированной интегральной схеме (ASIC — Application-Specific Integrated Circuit) вычислительное и коммуникационное ядра, три уровня КЭШ-памяти, пять сетевых интерфейсов и контроллер для подключения внешней памяти. Микропроцессор производится по 0,13-микронной технологии.

Тактовая частота IBM PowerPC 440 составляет всего лишь 700 МГц, но каждое ядро микропроцессора способно выполнять за такт четыре операции с плавающей запятой. Следовательно, суммарная пиковая производительность вычислительного узла IBM Blue Gene/L оценивается значением 5,6 GFLOPS. В штатном режиме одно из ядер PowerPC используется для вычислений, а второе реализует функции межузловых обменов. При отсутствии обменов оба ядра могут работать как вычислительные, обеспечивая производительность (2,8×2) GFLOPS.

Вычислительный узел IBM Blue Gene — это композиция микропроцессора IBM PowerPC и чипов памяти DDR DRAM (DDR — Double Data Rate — удвоенная скорость обмена данными). Емкость памяти вычислительного узла суперВС достигает 2 Гбайт (плюс 4 Мбайта DRAM в микропроцессоре), а полоса канала между микропроцессором и памятью — 5,5 Гбайт/с (в IBM Blue Gene/L емкость памяти узла — 512 Мбайт).

Каждый вычислительный узел оснащается небольшой операционной системой (ядром ОС), что позволяет выполнять основные функции по вводу и обработке данных.

Узел ввода-вывода IBM Blue Gene. Через узлы ввода-вывода (Input/Output Nodes) осуществляется связь суперВС с глобальной параллельной файловой системой и с хост-компьютерами. Узлы ввода-вывода (как и вычислительные) основаны на двухъядерных интегральных схемах (ASICs), но они имеют расширенную внешнюю память и гигабитный сетевой интерфейс Ethernet.

Каждый узел ввода-вывода помимо ядра ОС содержит программный компонент, осуществляющий взимодействие с хост-компьютером.

Сервисный узел IBM Blue Gene. Для обеспечения доступа к суперВС в целях осуществления функций управления и мониторинга применяется специальный сервисный узел (Service Node or Front-end Node). Этот узел является внешним по отношению к собственно суперВС, и в качестве такового используется серийно производимая ЭВМ. Сервисный узел является частью хост-компьютерной системы.

Конфигурация модели IBM Blue Gene/L, эксплуатировавшаяся в 2006–2007 гг. в LLNL, в своем составе имела 65 536 и 12 вычислительных и сервисных узлов соответственно и 1024 узла ввода-вывода.

6.8.3. Коммуникационные сети IBM Blue Gene/L

В системе IBM Blue Gene используют пять коммуникационных сетей: управляющую, трехмерную тороидальную и древовидную сети, сети синхронизации и ввода-вывода. Сети тороидальная, древовидная и синхронизации характеризуются высокой пропускной способностью и низкой латентностью, они обеспечивают связность вычислительных узлов суперВС при реализации межузловых обменов данными.

Управляющая сеть IBM Blue Gene/L. Сеть Gigabit Ethernet–JTAG служит для реализации широкого спектра функций управления и мониторинга, среди которых: инициализация системы и загрузка вычислительных узлов, текущий контроль за состоянием, диагностика и устранение неисправностей в суперВС.

Управляющая сеть IBM Blue Gene/L является интрасетью (Intranet), основанной на стандартах Ethernet и JTAG (JTAG — Joint Test Automation Group — объединенная группа по автоматизации тестирования). Она обеспечивает связь сервисного узла со всеми оконечными точками вычислительных ресурсов IBM Blue Gene/L (рис. 6.15). В 64К-узловой конфигурации модели IBM Blue Gene/L имеется более 250 000 оконечных точек интегральных схем (ASICs), древовидных схем генераторов тактовой частоты, температурных сенсоров, источников электропитания, вентиляторов, светоизлучающих диодов для индикации состояний и т. п. Именно через эти точки сервисный узел осуществляет инициализацию, управление и контроль суперВС.

Адаптация информационных пакетов 100-мегабитной Ethernet к различным управляющим цепям (оконечным точкам) осуществляется специальным FPGA-чипом (FPGA — Field Programmable Gate Array) — программируемой вентильной матрицей. В частности, этот чип адаптирует Ethernet к интерфейсу JTAG вычислительных узлов и узлов ввода-вывода. Интерфейс JTAG обеспечивает начальную загрузку узлов и доступ к каждому из них при контроле, диагностике и устранении неисправностей.

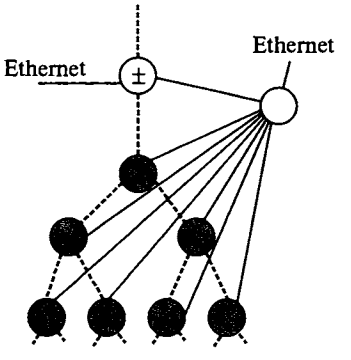


Рис. 6.15. Фрагменты сетей древовидной, управляющей и ввода-вывода IBM Blue Gene/L: ● — вычислительный узел; ⊕ — узел ввода-вывода; ○ — сервисный узел; ------ — линк древовидной сети; —— — линк управляющей сети

Интерфейс JTAG предоставляет доступ хост-компьютерной системе к регистрам каждого процессорного ядра суперВС.

Управляющий FPGA-чип в любой конфигурации суперВС размещается на каждой плате — конструктивном объединении из 32-х вычислительных узлов и 4-х узлов ввода-вывода.

Средства управляющей сети IBM Blue Gene/L позволяют предельно просто осуществлять взаимодействия между хост-компьютерной системой и ресурсами суперВС.

Трехмерная тороидальная сеть IBM Blue Gene/L. Для организации этой сети служат шесть специальных двусторонних связей (линков) каждого вычислительного узла суперВС. В сети обеспечивается трехмерная связность вычислительных узлов, причем структура данной суперВС представляется 3D-тором (см. рис. 4.7): $N(x, y, z)$, где N — число вычислительных узлов; x, y, z — число узлов, составляющих кольца по трем направлениям X, Y, Z. Тор 8 (2×2×2).

Трехмерная тороидальная сеть (Three-dimensional Torus Network) IBM Blue Gene обеспечивает пересылки информации от одного вычислительного узла к другому соседнему (передачи сообщений типа point-to-point). Через эту сеть любой из вычислительных узлов может взаимодействовать с любым другим узлом суперВС. Взаимодействие узла с удаленными узлами осуществляется через множество транзитных узлов.

В модели IBM Blue Gene/L из 64 К узлов реализован 3D-тор вида: 65 536 (32×32×64). Скорость передачи информации от узла к ближайшему соседнему узлу по каждому из трех направлений в данной модели составляет 175 Мбайт/с. Каждый узел поддерживает шесть независимых двунаправленных связей с ближайшими соседними узлами. Следовательно, суммарная пропускная способность узла достигает 2,1 Гбайт/с. Латентность узла при транзитном прохождении информации через него равна 10 нс. Для 64 К-узловой модели IBM Blue Gene/L максимальное число пересылок (Hops) данных через транзитные узлы равно 32 + 16 + 16 = 64, значит, в наихудшем случае сетевая латентность составит 6,4 мкс.

Древовидная сеть IBM Blue Gene/L. Для формирования этой сети используются три специальных связи (линка) каждого из вычислительных узлов. Сеть обеспечивает двустороннюю связь вычислительного узла или с одним, или с двумя, или с тремя соседними узлами (см. рис. 6.15). Древовидная сеть (Global Tree Network or Global Collective Network) поддерживает глобальные коллективные взаимодействия вычислительных узлов в пределах всей суперВС.

К глобальным взаимодействиям, реализуемым множеством вычислительных узлов суперВС с использованием древовидной сети, относятся трансляционный обмен данными (Broadcast — широковещательная рассылка) и арифметические и логические операции.

При трансляционном обмене данные передаются из (любого) вычислительного узла всем остальным узлам или узлам некоторого подмножества (см. рис. 3.6, *б*). Древовидная сеть при таких передачах обеспечивает латентность менее 5 мкс. Межузловая связь (линк) в этой сети характеризуется пропускной способностью 350 Мбайт/с (4 бита за один процессорный такт) и при передаче, и при приеме данных.

Трансляционный обмен в суперВС может быть реализован и через тороидальную сеть. Однако она для этих целей используется реже, чем древовидная сеть. Это объясняется тем, что осуществление трансляционного обмена через тороидальную сеть связано со значительными затратами времени на синхронизацию. Кроме того, древовидная сеть имеет меньшую латентность, чем тороидальная. Вместе с этим следует отметить, что для больших сообщений время передачи по тороидальной сети может быть меньше, чем по древовидной сети.

Арифметическая и логическая аппаратура древовидной сети поддерживает такие глобальные целочисленные операции, как отыскание минимума или максимума, вычисление суммы или разности, поразрядные логические операции: И, ИЛИ, исключающее ИЛИ (сложение по модулю 2). Латентность древовидной сети IBM Blue Gene/L при выполнении таких операций в 10–100 раз меньше, чем у сетей других суперВС.

Глобальная операция суммирования с плавающей запятой в пределах всего пространства вычислительных узлов IBM Blue Gene/L выполняется примерно за 10 мкс. При этом требуется двукратное использование древовидной сети: первый раз — для отыскания максимального порядка чисел, второй раз — для суммирования «сдвинутых» мантисс.

Аппаратура древовидной сети поддерживает два виртуальных канала, что позволяет одновременно реализовать две неблокируемые глобальные операции.

Важной особенностью коммуникационной среды IBM Blue Gene/L является то, что обе сети — и тороидальная, и древовидная — могут работать параллельно. При формировании пользовательских подсистем автоматически обеспечивается связность их вычислительных узлов и по тороидальной, и по древовидной сетям.

Сеть синхронизации IBM Blue Gene/L. Эта сеть, называемая также сетью «барьер» (Barrier Network), используется для реализации в суперВС глобальных операций «барьер» (Barrier) и прерывание (Interrupt). Операция «барьер» необходима для синхронизации процессов, протекающих в различных ядрах вычислительных узлов. Суть ее заключается в том, что любой процесс, достигший точки синхронизации, не может быть продолжен до тех пор, пока все остальные процессы не достигнут своих точек синхронизации. Операция «барьер» всегда должна предшествовать коллективным обменам информацией в суперВС (см. разд. 3.3.6, 4.4.1).

Прерывание требуется при обнаружении неисправности в системе в целом или в ее подсистеме и для запуска процедур диагностики и восстановления.

Операции «барьер» и прерывание реализуются в логических схемах соответственно глобального И и глобального ИЛИ (охватывающих вычислительные узлы в пределах или всей суперВС, или ее подсистемы).

Сеть ввода-вывода IBM Blue Gene/L. Сеть для ввода-вывода информации является внешней по отношению к суперВС. В IBM Blue Gene/L она представляет собой Gigabit Ethernet (см. рис. 6.15). Эта сеть обеспечивает связь суперВС с глобальной файловой системой и хост-компьютерами.

Подключение Ethernet-коммутаторов к узлам ввода-вывода суперВС осуществляется через гигабитный Ethernet-интерфейс.

Сеть ввода-вывода и глобальная древовидная сеть IBM Blue Gene/L функционально связаны. Роль посредников такой связи играют узлы ввода-вывода, причем каждый из них способен взаимодействовать с 64-мя вычислительными узлами по древовидной сети. Следовательно, древовидная сеть используется и при загрузке суперВС. Скорость трансляционной передачи информации (Broadcast) от узла ввода-вывода к 64-м вычислительным узлам составляет 2,8 Гбит/с. Скорость обмена узла ввода-вывода с файловой системой по одному линку Ethernet равна 1 Гбит/с.

Число узлов ввода-вывода в суперВС не фиксировано. Их максимальное число, отнесенное к числу вычислительных узлов, составляет 1:8. Если бы конфигурирование производилось с отношением 1:64, то 64К-узловая IBM Blue Gene/L имела бы 1024 узла ввода-вывода с пропускной способностью более 1 Тбит/с (что превышало бы возможности древовидной сети).

6.8.4. Конструкция системы IBM Blue Gene/L

Для формирования конфигураций вычислительной системы IBM Blue Gene/L применяются стойки, в каждой из которых размещается 1024 вычислительных узла (двухъядерных микропроцессоров PowerPC). Полностью укомплектованная стойка характеризуется следующими показателями: производительность — 5,73 TFLOPS, емкость памяти — 512 Гбайт, энергопотребление — до 27,6 кВт, цена — 2 млн долл. Модель IBM Blue Gene/L, занимающая в 2008 г. первую позицию списка Top500, состоит из 104 стоек. Структура межузловых связей в любой стойке модели IBM Blue Gene/L — 3D-тор: 1024 (2×16×32).

При формировании конфигураций IBM Blue Gene/L используется иерархия конструктивов (табл. 6.1).

Таблица 6.1

Количественная характеристика	Конструктивный элемент			
	Вычислительный узел	Вычислительная карта	Плата	Стойка
Число процессорных ядер	2 (PowerPC 770 МГц)	4 (2 узла)	64 (16 вычислительных карт)	2048 (32 платы)
Производительность	5,6 GFLOPS	11,2 GFLOPS	179,2 GFLOPS	5,73 TFLOPS
Емкость памяти	512 Мбайт	1 Гбайт	16 Гбайт	512 Гбайт
Число узлов (ядер) ввода-вывода	–	–	4 (8)	16–128 (32–156)

Два узла ввода-вывода составляют карту. В каждой плате может находиться до двух карт ввода-вывода (вместе с 16-ю вычислительными картами).

В стойках применена воздушная система охлаждения.

6.8.5. Программное обеспечение IBM Blue Gene/L

Модель IBM Blue Gene/L функционирует под управлением специальной версии операционной системы Linux. Каждый из функциональных узлов оснащается своей небольшой ОС. Вычислительный узел имеет ядро ОС, узел ввода-вывода и сервисный узел — ядро ОС и специализированные программные компоненты.

Инструментарий параллельного программирования IBM Blue Gene/L включает: библиотеку MPI функций для поддержки параллельных процессов (Message Passing Interface), ориентированную на языки C, C++, FORTRAN; языки CAF (Co-Array FORTRAN) и UPC (Unified Parallel C); библиотеку ESSL подпрограмм для инженерных и научных расчетов (Engineering and Scientific Subroutine Library); подсистему MASS для ускоренных математических вычислений (Mathematical Acceleration Subsystem); алгоритмы FFT и 3D-FFT быстрого преобразования Фурье (Fast Fourier Transform); неспециализированную параллельную файловую систему GPFS (General Parallel File System); менеджер LoadLeveler, осуществляющий загрузку ресурсов ВС. Библиотека ESSL включает в себя более 150 оптимизированных программ для вычисления математических функций; подсистема MASS содержит набор оптимизированных программ для вычисления дробей и квадратных корней с одинарной и двойной точностью.

6.8.6. Области применения системы IBM Blue Gene

Создание петафлопсной системы IBM Blue Gene («Голубой ген») изначально было продиктовано необходимостью решения фундаментальной проблемы — исследования процесса синтеза и свертывания белков. Без моделирования этого процесса невозможно, в частности, понять особенности ряда заболеваний (например, Паркинсона) и создать лекарства узконаправленного действия.

Разработчиками указаны три обширные области применения IBM Blue Gene:

- моделирование физических явлений;
- обработка данных в реальном масштабе времени;
- автономный (offline) анализ данных.

Первая область применения суперВС стала доминирующей. Сюда относятся исследование проблем наук о жизни, моделирование климата и предсказание погоды, энергетические исследования, моделирование и проектирование автомобильных и аэрокосмических конструкций.

Суперкомпьютеры необходимы для исследований по материаловедению и нанотехнологиям. Они успешно применяются для моделирования ядерных процессов и в исследованиях по ядерным вооружениям.

Работы по проекту IBM Blue Gene так же, как и по проекту Cray X, ведутся прежде всего в интересах национальной безопасности США (!).

6.9. Анализ мультипроцессорных вычислительных систем с усовершенствованной структурой

Классический подход на основе последовательной машины Дж. фон Неймана, подход на основе модели вычислителя, уже в 1960-х годах вступил в противоречие с требованиями, предъявляемыми к высокопроизводительным средствам обработки информации. Такая ситуация инициировала исследования по поиску новых принципов переработки информации, новых структурных и архитектурных решений, которые бы адекватно учитывали возможности технологии микроминиатюризации.

В настоящее время наблюдается необычайный подъем в исследованиях и опытно-конструкторских работах по мультипроцессорным ВС с усовершенствованной структурой. Такие ВС вплотную подошли к средствам, полностью основанным на модели коллектива вычислителей, т. е. к ВС с программируемой структурой.

В отличие от систем с канонической структурой мультипроцессора в ВС с распределенной памятью принцип программируемости (автоматиче-

ской настраиваемости) структуры получил внедрение в наиболее завершенном виде. Системы стали обладать способностью к автоматической реконфигурации структуры в процессе решения задач, представленных в параллельной форме, они стали допускать возможность разбиения на подсистемы для целей мультипрограммирования (одновременного решения нескольких задач на различных подсистемах). Вычислительные системы приобрели бóльшую способность к статической (априорной) реконфигурации структур и состава для адаптации под область применения и условия эксплуатации. Такие ВС становятся все более распределенными и в реализации ориентированы на современную технологию микропроцессорных БИС. Недалеко то время, когда системы как ансамбли ЭП будут производиться в едином технологическом цикле.

Подтверждением сказанному служит архитектура Connection Machine (см. § 5.4). Эта ВС (на макроуровне) имеет архитектуру MIMD и представляется как композиция из суперпроцессоров. В качестве суперпроцессора выступает SIMD-подсистема из 1024 вычислительных узлов, а в качестве последних — кристаллы, содержащие по 16 ЭП. Таким образом, Connection Machine представляет собой «мультисуперпроцессорную» систему.

Приведем еще один пример современной мультипроцессорной ВС. Как уже отмечалось (см. § 6.5), в фирме IBM были созданы первые мультипроцессорные ВС уже в середине 1950-х годов. Конечно, эти первые системы были достаточно просты, число процессоров в них имело порядок 10 (т. е. они не были многопроцессорными в современном понимании). Сейчас ВС рассматриваемого архитектурного класса действительно можно отнести к многопроцессорным, они состоят из сотен и сотен тысяч процессоров. Последнее послужило основанием называть их системами с массовым параллелизмом. В качестве примера такой ВС служит одна из разработок 1990-х годов фирмы IBM — система SP2.

Отметим архитектурные особенности ВС IBM RS/6000 SP2 (серии RS/6000). Тип архитектуры ВС — MIMD, это масштабируемая система с массовым параллелизмом: число элементарных процессоров (или узлов) в ней составляет от 2 до 512. Память системы — распределенная, т. е. каждый ЭП имеет свою локальную оперативную и дисковую память. Связь между ЭП осуществляется через высокопроизводительный коммутатор (IBM high-performance switch). Таким образом, функциональная структура системы IBM RS/6000 SP2 — это мультипроцессор с распределенной памятью (см. рис. 6.5). Программное обеспечение системы IBM RS/6000 SP2 включает, в частности, операционную систему AIX, систему IBM LoadLeveler, реализующую режим пакетной обработки информации, операционную среду POE (Parallel Operating Environment), поддерживающую параллельные вычисления.

Начало XXI в. ознаменовалось созданием самой мощной в мире суперВС IBM Blue Gene (см. § 6.8), которая полностью основывается на принципах модели коллектива вычислителей (см. § 3.1).

Анализ архитектуры мультипроцессорных ВС позволяет сделать нижеследующие выводы.

1. Основная тенденция в области архитектуры мультипроцессорных ВС — повышение степени полноты воплощения принципов модели коллектива вычислителей (параллелизма, программируемости структуры и конструктивной однородности).

2. Архитектурные возможности ЭП неуклонно наращиваются, их структура претерпела трансформацию от простейших конфигураций без памяти до элементарных машин — композиций из мощных микропроцессоров, оперативной памяти и внешних запоминающих устройств (и даже устройств ввода-вывода информации).

3. Мультипроцессорные ВС, начавшие свою историю как композиции из нескольких процессоров, превратились в системы с массовым параллелизмом (с количеством процессоров порядка 10^2–10^5).

4. Современные мультипроцессорные ВС — это распределенные средства обработки информации, они имеют множество процессоров и распределенную память. Более того, в них и коммутатор (или другой ресурс), через который осуществляется взаимодействие процессоров, может быть распределенным. Программное обеспечение таких ВС также является распределенным.

5. Высокопроизводительные ВС рассматриваемого класса представляют собой суперсистемы: это множество мощных микропроцессоров-конвейеров или матричных процессоров или даже объединение мультипроцессоров.

Эволюционное развитие архитектуры мультипроцессорных ВС так же, как конвейерных и матричных ВС (см. § 4.4, 5.4 и 5.5), привело к «революционной» модернизации первоначальных канонов. Любая современная высокопроизводительная ВС в зависимости от функционального уровня рассмотрения может выглядеть как MISD- или SIMD-, или MIMD-система, более того, ее функциональная структура одновременно обладает свойствами конвейерных, матричных и мультипроцессорных систем. Исследователи параллельных вычислительных технологий, архитекторы и создатели промышленных ВС (независимо от их первоначальных архитектурных концепций) пришли к необходимости создания распределенных ВС с программируемой структурой.

7. ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ С ПРОГРАММИРУЕМОЙ СТРУКТУРОЙ

Десять лет, прошедшие с момента появления первой ЭВМ, позволили выявить пределы в развитии средств обработки информации на основе концептуальной машины Дж. фон Неймана и на основе модели вычислителя (см. § 2.10). Исследователи и проектировщики средств обработки информации уже в начале 60-х годов XX в. пришли к пониманию необходимости технической реализации новых принципов обработки информации.

Исследования по ВС, основанным на модели коллектива вычислителей, были начаты в Институте математики Сибирского отделения АН СССР в начале 1960-х годов по инициативе математика и механика С.Л. Соболева (1908–1989; академик АН СССР с 1939 г.). Результатом этих исследований явилось научно-техническое направление «Вычислительные системы с программируемой структурой». Такие системы обладают способностью адаптации под структуры и параметры решаемых задач.

В главе 7 приведены основные понятия ВС с программируемой структурой и описаны архитектуры систем «Минск-222», МИНИМАКС, СУММА, МИКРОС, МВС.

7.1. Понятие о вычислительных системах с программируемой структурой

В гл. 4–6 было показано, что в конвейерных, матричных и мультипроцессорных ВС принципы модели коллектива вычислителей воплощены недостаточно полно. Пожалуй, самым главным недостатком архитектуры таких ВС является наличие единого ресурса (устройства управления, управляющей ЭВМ, коммутатора и т. п.). Отказ единого ресурса приводит к отказу ВС в целом, что неприемлемо даже с позиций надежности и живучести. Единый ресурс не позволяет организовать мультипрограммную работу, при которой на различных подсистемах — связных подмножествах технических ресурсов (элементарных процессорах) — будут выполняться одновременно различные программы.

Значительный прогресс в практической реализации принципов модели коллектива вычислителей обеспечивают ВС с распределенными средствами управления. В отечественных научно-технических работах к таким системам относят ВС с программируемой структурой.

7.1.1. Этапы исследований

Работы в области высокопроизводительных средств обработки информации выполнялись в Институте математики (ИМ) СО АН СССР в 1960-х годах под руководством специалиста по вычислительной технике Э.В. Евреинова (1928, профессор с 1972 г.). Первая работа сотрудников ИМ СО АН СССР [14] о возможности построения ВС высокой производительности опередила американские публикации в данной области примерно на 6 месяцев. В середине 1960-х годов вышла в свет монография [17], обобщающая первые результаты работ ИМ СО АН СССР по функциональным структурам ВС и параллельному программированию. Во второй половине 1960-х годов были созданы и первые ВС [5]: «Минск-222» (1965–1966) и управляющая ВС для автоматизации научных исследований (1964–1967). К началу 1970-х годов завершилось формирование концепции ВС с программируемой структурой, как средств обработки информации, основанных на модели коллектива вычислителей. Следует отметить, что первоначальное название рассматриваемых средств — «Однородные вычислительные системы» [14]; в конце 1970-х годов закрепляется название «ВС с программируемой структурой» (см. [5], стр. 26), так как оно точнее отражает архитектурные возможности систем — коллективов вычислителей. «Однородные ВС» и «ВС с программируемой структурой» следует рассматривать как синонимические термины.

Начиная с 70-х годов XX в. теоретические и проектные работы в Сибирском отделении АН СССР (ныне СО РАН) по ВС с программируемой структурой ведутся под руководством одного из разработчиков первой ВС с программируемой структурой «Минск-222» В.Г. Хорошевского (1940; д-р техн. наук с 1974 г., чл.-кор. РАН с 2000 г.).

Главными направлениями работ становятся:

- архитектура ВС;
- теория структур ВС: анализ и синтез структур сетей межмашинных связей;
- теория функционирования ВС: организация оптимального (субоптимального, стохастически оптимального) функционирования ВС в моно- и мультипрограммных режимах (обработки наборов и обслуживании потоков параллельных задач);

- надежность и живучесть (потенциальная и структурная) ВС;
- самодиагностика и самоконтроль ВС;
- отказоустойчивые параллельные вычислительные технологии;
- проектирование ВС;
- распределенные ОС;
- системы параллельного программирования;
- параллельные алгоритмы и программы для решения прикладных задач.

Работы по ВС из академической сферы распространяются в промышленность, под руководством автора данной книги создается ряд систем: МИНИМАКС (1975), СУММА (1976), МИКРОС-1 (1986), МИКРОС-2 (1992), МИКРОС-Т (1996). Выходит в свет большое число публикаций как сотрудников СО АН СССР (СО РАН), так и других организаций [5, 6, 13, 14, 17, 18].

В 90-х годах XX в. активизируются работы по построению отечественных промышленных ВС с массовым параллелизмом. Работы по созданию систем семейства МВС (генераций МВС-100 и МВС-1000) выполнены в кооперации научно-исследовательских институтов РАН и промышленности.

Вычислительные масштабируемые системы генерации МВС-100 эксплуатируются с 1992 г. Количество процессоров в различных конфигурациях составляет от 4 до 128; производительность конфигураций ВС — от 400 MFLOPS до 10 GFLOPS. Системы генерации МВС-1000 поставляются с 1998 г.; их большемасштабные конфигурации ВС позволяют достичь производительности $10...10^3$ GFLOPS.

Вычислительные системы с программируемой структурой — это распределенные средства обработки информации. В таких ВС нет единого функционально и конструктивно реализованного устройства: все компоненты (устройство управления, процессор и память) являются распределенными. Тип архитектуры ВС — MIMD; в системах заложена возможность программной перенастройки архитектуры MIMD в архитектуры MISD или SIMD (см. разд. 3.4.2).

Основной функционально-структурной единицей вычислительных ресурсов в системах рассматриваемого класса является *элементарная машина* (ЭМ). Допускается конфигурирование ВС с произвольным числом ЭМ. Следовательно, ВС с программируемой структурой относятся к масштабируемым средствам обработки информации и допускают формирование конфигураций с массовым параллелизмом (Scalable Massively Parallel Architecture Computing Systems).

Вычислительные системы с программируемой структурой сочетают в себе достоинства универсальных и специализированных средств обработки информации, в них допускается автоматическое формирование виртуальных проблемно-ориентированных конфигураций. В таких ВС с достаточной пол-

нотой воплощены перспективные архитектурные принципы, системы основаны на модели коллектива вычислителей (см. § 3.1), обладают большими потенциальными возможностями по обеспечению высоких значений показателей эффективности функционирования.

7.1.2. Определение вычислительной системы

При построении ВС с программируемой структурой доминирующими являются следующие три принципа:

1) массовый параллелизм (параллельность выполнения большого числа операций);

2) программируемость (автоматическая перестраиваемость или реконфигурируемость) структуры;

3) конструктивная однородность.

Следует подчеркнуть, что принцип программируемости структуры ВС является таким же важным, каким в свое время было предложение Дж. фон Неймана относительно организации в ЭВМ автоматической модификации программ. (Он предложил хранить программу в памяти машины вместе с данными, что позволило модифицировать программу с помощью самой ЭВМ.) *Принцип программируемости структуры требует, чтобы в ВС была реализована возможность хранения программного описания функциональной структуры и программной ее модификации (перенастройки) с целью достигнуть адекватности структурам и параметрам решаемых задач.*

Под ВС с программируемой структурой понимается совокупность элементарных машин, функциональное взаимодействие между которыми осуществляется через программно настраиваемую сеть связи. Элементарная машина — это композиция из вычислительного модуля и системного устройства. *Вычислительный модуль* (ВМ) служит как для переработки и хранения информации, так и для выполнения функций по управлению системой в целом. *Системное устройство* (СУ) — аппаратурная часть ЭМ, предназначенная только для обеспечения взаимодействия данной ЭМ с ближайшими соседними машинами (точнее, с системными устройствами, с которыми имеется непосредственная связь). Фрагмент возможной структуры ВС рассматриваемого класса показан на рис. 3.7 и 6.8.

В качестве вычислительного модуля в общем случае рассматривается аппаратурно-программный комплекс, структура и состав которого допускают варьирование в широких пределах. Далее под ВМ будем понимать лишь аппаратурную часть этого комплекса. Минимальный состав ВМ представляется (элементарным или локальным) процессором с (локальной) оперативной памятью; промежуточный — ЭВМ с каналами (например, прямого дос-

тупа к памяти, селекторными, мультиплексными), внешними запоминающими устройствами, устройствами ввода-вывода информации, терминалами и т. п.; максимальный — вычислительной системой (или средством, основанным на модели коллектива вычислителей). Вычислительные модули могут быть произвольного состава, но для компоновки каждого из них в пределах одной ВС, как правило, используется один и тот же тип процессора. При более широкой трактовке класса ВС с программируемой структурой можно ограничиться лишь требованием программной совместимости вычислительных модулей.

Простейшей и развитой конфигурациям СУ соответствуют локальный коммутатор и коммуникационный процессор. Простейшей конфигурацией ЭМ является ЭП, т. е. композиция локального процессора (АЛУ и локальной памяти) и предельно простого локального коммутатора (см. рис. 3.7).

Системы с программируемой структурой предназначены для решения задач произвольной сложности (с произвольным объемом вычислений) как в моно-, так и в мультипрограммных режимах, они могут использоваться как вычислительные средства общего назначения либо как проблемно-ориентированные, могут эксплуатироваться автономно либо в составе сложной системы.

7.1.3. Сосредоточенные и распределенные вычислительные системы

В разд. 7.1.1 упоминалось, что ВС с программируемой структурой являются распределенными средствами и в архитектурном, и в функциональном планах. В самом деле, в них любые ресурсы (процессоры, память и устройства управления, и системные устройства) представляются множествами соответствующих модулей. Тем не менее в классе систем с программируемой структурой выделяют (пространственно) сосредоточенные и распределенные ВС. Характерной особенностью сосредоточенных ВС является компактное пространственное размещение средств обработки и хранения информации, при котором нет необходимости использовать телекоммуникационные сети (существующие абонентские или специально разработанные сети передач данных). В сосредоточенных ВС нет линий связи, вносящих существенную задержку в работу ВС, нет жестких ограничений на топологию сети связи, на возможность параллельной передачи информации между функциональными модулями (процессорами, модулями памяти, ЭМ и др.). В таких ВС среднее время передачи слова между функциональными модулями соизмеримо со средним временем выполнения одной операции в процессоре.

К распределенным ВС относят макросистемы — системы сложной конфигурации, в которых в качестве функциональных элементов выступают

пространственно-рассредоточенные вычислительные средства, основанные на моделях вычислителя и коллектива вычислителей, и сети связи, обеспечивающие взаимный теледоступ между средствами обработки информации.

Распределенная ВС — объединение пространственно удаленных друг от друга сосредоточенных ВС, основанное на принципах:

- *параллельности* функционирования сосредоточенных ВС (т. е. способности нескольких или всех сосредоточенных систем совместно и одновременно решать одну сложную задачу, представленную параллельной программой);
- *программируемости структуры* (т. е. возможности автоматически настраивать сеть связи между сосредоточенными ВС);
- *гомогенности состава* (т. е. программной совместимости различных сосредоточенных ВС и однотипности элементарных машин в каждой из них).

Распределенные ВС в общем случае предназначаются для реализации параллельных программ решения задач (произвольной сложности или с произвольным объемом вычислений, в монопрограммном и мультипрограммном режимах) на рассредоточенных в пространстве вычислительных ресурсах. Они должны быть приспособленными и для выполнения функций, присущих вычислительным сетям, и для реализации последовательных программ. В распределенных ВС допустимо централизованное и децентрализованное управление вычислительными процессами.

Любое функциональное взаимодействие между сосредоточенными ВС, составляющими распределенную систему, осуществляется через сеть связи. В сеть связи включаются каналы и системные устройства, распределенные по сосредоточенным ВС. В распределенных ВС нет принципиальных ограничений на физическую природу каналов; в них могут использоваться двухпроводные, коаксиальные и радиорелейные линии связи, средства коммуникации на основе волоконной оптики, микроволновой и спутниковой связи и др. На системное устройство возлагается выполнение функций, свойственных СУ для сосредоточенных ВС и устройствам связи для вычислительных сетей. В элементарные функции любого СУ входят: обмен информацией между сосредоточенной ВС и каналом, а также обеспечение транзитной передачи информации. В мощных распределенных ВС системное устройство может быть реализовано как параллельный сервер.

Предельным по простоте вариантом сосредоточенной ВС является ЭМ, следовательно, простейшим вариантом распределенной ВС является совокупность однотипных и регулярно соединенных ЭМ, использующая «длинные» программно-коммутируемые каналы межмашинной связи. В этом случае в состав ЭМ входят однотипные СУ.

Более развитым вариантом распределенных ВС являются системы, структура сети связи которых в целом является нерегулярной и которые

компонуются из программно-совместимых ЭМ. В данном случае сеть связи организуется посредством программно-совместимых системных устройств. В топологическом плане такие ВС классифицируются как централизованные, децентрализованные, кольцевые и радиально-кольцевые системы. Топологическое изображение названных систем аналогично схемам, общепринятым для вычислительных сетей.

Распределенные системы, в которых выполняется условие программной совместимости ЭМ, относятся к гомогенным ВС. В теоретическом плане можно рассматривать и гетерогенные распределенные ВС, состоящие из программно-несовместимых ЭМ и сосредоточенных ВС и имеющие сеть связи с нерегулярной структурой. Однако реализация гетерогенных систем наталкивается на большие трудности, связанные прежде всего с организацией их эффективного функционирования.

В подкласс распределенных ВС входят вычислительные сети. Несмотря на внешнюю схожесть распределенных систем и сетей, существует принципиальное отличие в их функционировании. Вычислительные сети выполняют обработку потоков задач, как правило, представленных последовательными программами. В сетях параллельные программы также могут быть реализованы, если в их состав входят сосредоточенные ВС (например, в сети ARPA такой системой была ILLIAC-IV; в глобальной сети Internet функционирует большое разнообразие конвейерных, матричных, кластерных и мультипроцессорных ВС). Решение каждой задачи выполняется на отдельной ЭВМ или сосредоточенной ВС, совместное использование нескольких ЭВМ и (или) ВС для решения одной задачи не допускается. При обслуживании потоков задач в вычислительных сетях осуществляются распределение нагрузки на ресурсы, передача информации, распределение программ и данных, дистанционная обработка и т. п. В вычислительных сетях основным режимом является диалоговый, при котором взаимодействие происходит между двумя абонентами (пользователь — ЭВМ или ВС, пользователь — пользователь). Естественно, в сетях одновременно может происходить большое число диалогов. Работа вычислительной сети в этом отношении напоминает функционирование телефонной (проводной или сотовой) сети.

В распределенных ВС наряду с режимом работы, характерным для вычислительных сетей, реализуется также режим решения общей сложной задачи. Задача представляется параллельной программой, каждая макроветвь (или ветвь) которой реализуется на своей сосредоточенной ВС (или ЭМ). Для распределенных ВС при решении одной сложной задачи характерен режим группового обмена информацией между абонентами (машины — машины, машины — пользователь).

Таким образом, *распределенные системы относятся к более общему классу средств обработки информации по сравнению с вычислительными*

сетями и качественно от них отличаются возможностью решения одной сложной задачи на распределенных вычислительных ресурсах.

На производительность распределенных ВС (при реализации параллельных программ сложных задач) сильно влияют скорость передачи информации в каналах и запаздывание сигналов при передаче информации из одной сосредоточенной ВС (в простейшем варианте системы из одной ЭМ) в другую. Для уменьшения этого влияния используется совмещение счета с обменом информацией и применение в полной мере методики крупноблочного распараллеливания сложных задач (см. § 3.3).

При распараллеливании процесса вычислений (разбиении задачи на параллельные ветви — информационно связанные подзадачи — и размещении их в ЭМ системы) нужно стремиться к укрупнению частей задачи, решаемых автономно каждой ЭМ, и к уменьшению количества информации, передаваемой по каналам между машинами. Практика распараллеливания сложных задач показывает, что по мере укрупнения «автономных» частей задачи имеет место уменьшение количества информации, передаваемой при межмашинных взаимодействиях. Поэтому даже при относительно малой скорости передачи информации по межмашинным каналам существует реальная возможность эффективно решать задачу не только на сосредоточенных, но и на распределенных ВС (если эта задача может быть разбита на такие крупные подзадачи, время параллельного счета для которых будет много больше времени, расходуемого на обмен информацией между машинами, реализующими подзадачи).

При построении распределенных ВС большое значение приобретают экономические факторы. Так, стоимость распределенной ВС (в сравнении с сосредоточенной системой) в существенной мере определяется стоимостью достаточно дорогих физических каналов связи между рассредоточенными в пространстве вычислительными ресурсами (сосредоточенными системами или в простейшем случае — ЭМ). Следовательно, для снижения стоимости системы в целом приходится при построении ВС использовать минимальное число каналов связи между ЭМ. В простейших приложениях достаточно ограничиться топологией сети межмашинных связей, обеспечивающей непосредственную связь любой ЭМ с двумя ЭМ (достаточно использовать одномерные или кольцевые структуры), а также применить способ межмашинной передачи информации в последовательной форме.

Появившиеся в конце XX в. так называемые кластерные ВС или вычислительные кластеры представляют варианты сосредоточенных и распределенных систем. Различают физические и логические кластеры. Первые кластеры являются специально сконфигурированными ВС из серийных аппаратурно-программных средств ЭВМ. Логические (или виртуальные) кластеры организуются в аппаратурно-программной среде вычислительных се-

тей. Для такой организации используются дополнительные системные средства (как правило, программные), которые превращают вычислительную сеть в средство для реализации параллельных вычислений.

Материал, изложенный в последующих разделах книги, относится в основном к сосредоточенным ВС. Здесь мы ограничимся ссылкой на книгу [5], в которой описана первая распределенная ВС с программируемой структурой АСТРА. Эта система создана ИМ СО АН СССР и Новосибирским электротехническим институтом МВ и ССО РСФСР; работы по проектированию ВС были начаты в 1970 г., а первая ее конфигурация была сдана в эксплуатацию в 1972 г. Среди конфигураций ВС АСТРА имелись городские и междугородные (Новосибирск — Москва). Они формировались из средств ЭВМ «Минск-32» и использовали телефонные каналы связи. Были выполнены проекты распределенных ВС и на базе машин третьего поколения семейства ЕС ЭВМ [5].

Целесообразно подчеркнуть, что современное информационное пространство — это локальные и распределенные корпоративные вычислительные сети и глобальная сеть Internet. Дальнейшим шагом в развитии архитектуры сетей должны стать распределенные ВС, способные реализовать параллельный алгоритм решения сложной задачи на рассредоточенных ЭМ и ВС. Наиболее перспективны распределенные ВС, основанные на принципах программируемости межмашинных связей и конструктивной однородности.

Опыт 70-х годов XX столетия в создании и эксплуатации распределенных ВС убеждает в том, что такие системы являются эффективным инструментарием решения сложных задач. Установлено, что при реализации параллельных программ сложных задач на n машинах распределенной ВС достигается производительность $A_n n\omega$ (A_n — коэффициент, $A_n \geq 1$; ω — быстродействие одной ЭМ). В распределенных ВС эффективно реализуются все режимы работы, характерные для вычислительных сетей.

Создание распределенных ВС из средств серийно выпускаемых ЭВМ и ВС не вызывает существенных инженерных и математических трудностей. Как правило, требуются незначительные доработки базового ПО, связанные с реализацией системных операций в виде подпрограмм и с введением системного диспетчера в ОС. Это же самое можно сказать и относительно организации (логических) вычислительных кластеров на базе существующих компьютерных сетей.

Таким образом, распределенные ВС в сравнении с компьютерными сетями являются более общим классом средств переработки информации.

В заключение следует отметить, что в индустрии обработки информации в конце XX в. получили развитие так называемые GRID-технологии (GRID — Global Resource Information Distribution). На основе этих технологий создаются большемасштабные пространственно-распределенные системы обработки информации, способные реализовать параллельные алгоритмы решения супер-

сложных задач на своих рассредоточенных ресурсах. Такая GRID-система представляет собой композицию множества ЭВМ и ВС, пространственно-распределенной коммуникационной сети и программных компонентов для осуществления параллельных вычислений. В GRID-системе могут использоваться гетерогенные и несовместимые вычислительные средства, однако для их совместной работы над параллельными алгоритмами должны быть применены специальные механизмы (например, протоколы) поддержки. Очевидно, что введение в Internet программных компонентов для реализации параллельных алгоритмов на ее распределенных вычислительных ресурсах превращает сеть в GRID-систему. Из сказанного следует, что GRID-система — это ничто иное, как распределенная ВС в том понимании, которое представлено в данном разделе.

7.2. Архитектурные особенности вычислительных систем с программируемой структурой

В архитектурном плане ВС с программируемой структурой — это аппаратурно-программные объекты, основанные на модели коллектива вычислителей. Первоначальное представление о функциональной организации ВС дано в гл. 3 и § 7.1. В этом параграфе будут изложены детализация структуры ВС, элементы теории структур, описаны режимы функционирования ВС и способы обработки информации в них и, наконец, отмечены аспекты, связанные с организацией ОС. Материал § 7.2 вместе с содержанием гл. 3 позволит достаточно полно судить об архитектуре ВС с программируемой структурой.

7.2.1. Структура вычислительной системы

Как известно (см. разд. 3.1.2), (макро)структура ВС описывается графом G, множеству вершин которого сопоставлены ЭМ (или системные устройства, или локальные коммутаторы), а множеству ребер — линии межмашинных связей.

Какие же структуры следует использовать в ВС, особенно в масштабируемых и большемасштабных ВС? Это достаточно сложный вопрос, однако аксиоматически ясно, что полносвязные структуры (когда любая ЭМ непосредственно связана с каждой ЭМ) практически не приемлемы (хотя бы из-за ограничений существующих технологий БИС). Нужны структуры, которые были бы существенно проще полных графов и которые бы позволяли достичь «эффективного» решения задач (с учетом ненадежности компонентов ВС). Рассмотрим подробнее структурные аспекты, прежде всего связанные с производительностью и живучестью ВС.

Требования, предъявляемые к структуре ВС. Остановимся на требованиях, которые предъявляются к структурам современных ВС.

1. *Простота вложения параллельного алгоритма решения сложной задачи в структуру ВС.* Структура ВС должна быть адекватна достаточно широкому классу решаемых задач; настройка проблемно-ориентированных виртуальных конфигураций и реализация основных схем обмена информацией между ЭМ (см. разд. 3.3.5) не должны быть связаны со значительными накладными расходами (например, с большим временем работы операционной системы по вложению параллельного алгоритма).

2. *Удобство адресации элементарных машин и «переноса» подсистем в пределах ВС.* Вычислительная система должна предоставлять возможность пользователям создавать параллельные программы с виртуальными адресами ЭМ. Следовательно, структура ВС должна позволять реализовать простейший механизм преобразования виртуальных адресов ЭМ в реальные (физические) адреса машин ВС. Необходимость организации одновременного решения нескольких задач на ВС (т. е. необходимость разделения пространства элементарных машин между задачами) обосновывает требование простоты перемещения подсистем в пределах системы (при сохранении их топологических свойств). При выполнении данных требований будет достигнута эффективность ВС при работе как в моно-, так и мультипрограммных режимах. Кроме того, следует отметить, что данные требования являются необходимыми условиями для создания отказоустойчивых параллельных программ.

3. *Осуществимость принципа близкодействия и минимума задержек при межмашинных передачах информации в ВС.* Принцип близкодействия предопределяет реализацию обменов информацией между «удаленными» друг от друга ЭМ через промежуточные машины системы (см. разд. 3.2.1). Следовательно, в условиях ограниченности числа связей у каждой ЭМ структура должна обеспечивать минимум задержек при транзитных передачах информации.

4. *Масштабируемость и большемасштабность структуры ВС.* Для формирования конфигураций ВС с заданной эффективностью необходимо, чтобы структура обладала способностью к наращиванию и сокращению числа вершин (машин). Изменение числа ЭМ в ВС не должно приводить к коренным перекоммутациям между машинами и (или) к необходимости изменения числа связей для любых ЭМ.

Для достижения высокой производительности ВС при существующих возможностях микропроцессорной техники требуется количество ЭМ порядка $10–10^6$. Для поддержки большемасштабности (такого массового параллелизма) необходимо, чтобы структура ВС обладала способностью эффективно осуществлять межмашинные обмены информацией в условиях

невозможности реализации связей по полному графу (например, из-за ограниченности числа выводов с корпусов БИС).

5. *Коммутируемость структуры ВС.* Как уже было показано (см. разд. 3.3.5, 3.3.6), ВС должна быть приспособлена к реализации групповых межмашинных обменов информацией. Следовательно, структура ВС должна обладать способностью осуществлять заданное число одновременных непересекающихся взаимодействий между элементарными машинами.

6. *Живучесть структуры ВС.* Важным требованием к ВС в целом является обеспечение работоспособности при отказе ее компонентов или даже подсистем. Основой функциональной целостности ВС как коллектива ЭМ является живучесть структуры. Под последним понимается способность структуры ВС обеспечить связность требуемого числа работоспособных ЭМ в системе при ненадежных линиях межмашинных связей.

7. *Технологичность структур ВС.* Структура сети межмашинных связей ВС не должна предъявлять особых требований к элементной базе, к технологии изготовления микропроцессорных БИС. Системы должны быть восприимчивы к массовой технологии, их «вычислительное ядро» должно формироваться из массовых микропроцессорных БИС. Последнее позволит достичь приемлемых значений технико-экономических показателей ВС.

Анализ путей удовлетворения перечисленным требованиям приводит к безальтернативному выбору *однородных* (или регулярных, т. е. описываемых однородными графами) *структур* для формирования ВС. Заметим, что аналогичный вывод сделан в гл. 3 на основе опыта применения методики крупноблочного распараллеливания трудоемких задач.

Структурные характеристики ВС. Структура ВС представляет собой граф G (как правило, однородный для масштабируемых и большемасштабных ВС). Следовательно, *структурные задержки* при передачах информации между машинами ВС определяются расстоянием (в смысле теории графов) между вершинами структуры, сопоставленными взаимодействующим машинам. Для оценки структурных задержек в ВС используются диаметр d и средний диаметр $\overline{d}$ структуры. *Диаметр* — максимальное расстояние, определенное на множестве кратчайших путей между двумя вершинами структуры ВС:

$$d = \max_{i,j}\{d_{ij}\}, \tag{7.1}$$

средний диаметр

$$\overline{d} = (N-1)^{-1}\sum_{l=1}^{d} l n_l, \tag{7.2}$$

где d_{ij} — расстояние, т. е. минимальное число ребер, образующих путь из вершины i в вершину j; $i, j \in \{0, 1, \ldots, N-1\}$; n_l — число вершин, находя-

щихся на расстоянии l от любой выделенной вершины (однородного) графа G.

Показателем, оценивающим *структурную коммутируемость* ВС, является вектор-функция

$$\mathcal{K}(G, s, s') = \{\mathcal{K}_h(G, s, s')\}, \quad h \in \{1, 2, \ldots, [N/2]\}, \tag{7.3}$$

в которой координата $\mathcal{K}_h(G, s, s')$ есть вероятность реализации в системе при заданных структуре G и коэффициентах готовности s и s' соответственно одной ЭМ и линии связи h (см. § 2.8.4) одновременных непересекающихся межмашинных взаимодействий (обменов информацией между ЭМ); $[N/2]$ — целая часть числа $N/2$.

Структурная живучесть ВС оценивается вектор-функцией

$$\mathcal{L}(G, s, s') = \{\mathcal{L}_r(G, s, s')\}, \quad r \in E_2^N = \{2, 3, \ldots, N\}. \tag{7.4}$$

Здесь $\mathcal{L}_r(G, s, s')$ — вероятность существования подсистемы ранга r (т. е. подмножества из r работоспособных ЭМ, связность которых устанавливается через работоспособные линии связи) при заданных структуре G, коэффициентах готовности s и s' ЭМ и линии связи соответственно.

Введенные показатели позволяют осуществить с достаточной полнотой анализ структурных возможностей ВС и анализ структурной живучести ВС, в частности. Отметим прикладное значение показателей (7.1)–(7.4).

Прежде всего подчеркнем, что диаметр (7.1) и средний диаметр (7.2) — это структурные характеристики, связанные с производительностью ВС. Диаметр структуры ВС определяет максимально необходимое число транзитных (или ретранслирующих) вершин при межмашинных обменах информации, следовательно, он является количественной характеристикой для максимальных структурных задержек. Средний диаметр структуры ВС можно использовать в качестве показателя, оценивающего средние задержки при выполнении межмашинных взаимодействий.

По координатам *вектор-функции структурной коммутируемости* $\mathcal{K}(G, s, s')$ можно характеризовать ВС относительно ее возможностей по реализации обменов информацией между ее машинами. Эта характеристика важна для анализа структур в интересах и монопрограммного, и мультипрограммных режимов работы ВС. В монопрограммном режиме функционирования системы можно ограничиться тремя известными схемами обмена информацией (см. разд. 3.3.5). Тогда от структуры ВС требуется, чтобы:

1) при дифференцированном обмене имелась возможность реализации одного взаимодействия между любыми двумя ЭМ;

2) при трансляционном обмене реализовывалась одновременная передача информации из одной ЭМ во все остальные;

3) при конвейерно-параллельном обмене выполнялось одновременно $[N/2]$ взаимодействий между машинами $[N/2]$ пар.

Вектор-функция (7.3) дает полную характеристику структуры ВС по реализации межмашинных взаимодействий при работе систем в мультипрограммном режиме (когда все ЭМ используются для решения одной задачи).

В мультипрограммных режимах функционирования система программным способом разбивается на подсистемы. Максимальное число подсистем в ВС определяется, очевидно, величиной $[N/2]$. В случае мультипрограммирования обмены совершаются лишь в пределах подсистем. Тогда, например, координата $\mathcal{K}_h(G, s, s')$ вектор-функции (7.3) будет информировать о приспособленности структуры к генерации в пределах ВС h подсистем или к одновременному решению на ВС h задач, $1 \leqslant h \leqslant [N/2]$.

При более общей трактовке мультипрограммирования, когда вычислительные ресурсы (элементарные машины) делятся не только в пространстве (между подсистемами), но и во времени, говорят о виртуальных подсистемах, количество которых может быть произвольным, в частности, превышающим $[N/2]$. В этом случае вводят и виртуальные линии межмашинных связей ВС. Применительно к такой мультипрограммной работе ВС координата $\mathcal{K}_h(G, s, s')$ вектор-функции структурной коммутируемости должна определяться как условная вероятность описанного события (7.3), причем в качестве условия выдвигается то, что ни одна из физических линий связи (ни одно из ребер графа G) не должна обслуживать более чем l взаимодействий (разделяться более чем на l виртуальных линий); $l \in \{1, 2, \ldots, h\}$; $h \in \{1, 2, \ldots, [N/2]\}$.

Координаты *вектор-функции структурной живучести* (7.4) характеризуют приспособленность ВС в условиях отказов ЭМ и линий связи к порождению подсистем тех или иных рангов, следовательно, приспособленность ВС к решению задач заданной сложности. В частности, координата $\mathcal{L}_r(G, s, s')$ вектор-функции структурной живучести (7.4) определяет возможности структуры по реализации на ВС задач ранга r, т. е. сложных задач, представленных параллельными программами с числом ветвей, равным $r \in \{2, 3, \ldots, N\}$.

Подмножество координат

$$\{\mathcal{L}_{r^\circ}(G, s, s'), \mathcal{L}_{r^\circ+1}(G, s, s'), \ldots, \mathcal{L}_{r^\cdot}(G, s, s')\}$$

вектор-функции (7.4) характеризует приспособленность ВС по выполнению на ней *адаптирующихся параллельных программ*, допускающих автоматиче-

ское изменение своих рангов от r° до r^{*}, где $1 < r^{\circ}, r^{*} \leqslant N$. При $r^{\circ} = n$ и $r^{*} = N$ вектор-функция (7.4) будет характеризовать способность ВС с программируемой структурой к организации в ней виртуальных образований, обладающих живучестью (гл. 10, [5, 6]).

Перспективные структуры ВС. В гл. 3 (см. разд. 3.1.2) рассматривались структуры, перспективные для формирования ВС. В гл. 4–6 описаны структуры промышленных ВС с массовым параллелизмом. Здесь же рассмотрим структуры, удовлетворяющие требованиям, изложенным выше, т. е. перспективные для формирования масштабируемых и большемасштабных вычислительных систем (в частности, ВС с программируемой структурой).

В компьютерной индустрии получили распространение n-мерные структуры ВС, известные сейчас как циркулянтные (Circulant Structures). Впервые они были определены и исследованы в Отделе вычислительных систем ИМ СО АН СССР в начале 1970-х годов и первоначально назывались *D_n-графами* [5]. По определению *D_n-граф*, или *циркулянтная структура*, есть граф G вида: $\{N; \omega_1, \omega_2, \ldots, \omega_n\}$, в котором:

- N — количество вершин или порядок графа;
- вершины помечены целыми числами i по модулю N, следовательно, $i \in \{0, 1, \ldots, N-1\}$;
- вершина i соединена ребром (или является смежной) с вершинами $i \pm \omega_1, i \pm \omega_2, \ldots, i \pm \omega_n (\mathrm{mod}\ N)$;
- $\{\omega_1, \omega_2, \ldots, \omega_n\}$ — множество целых чисел, называемых образующими, таких, что $0 < \omega_1 < \omega_2 < \ldots < \omega_n < (N+1)/2$, а для чисел $N; \omega_1, \omega_2, \ldots, \omega_n$ наибольшим общим делителем является 1;
- n — размерность графа;
- $2n$ — степень вершины в графе.

В качестве примера рассмотрим D_2-граф, или двумерный циркулянт вида: {12; 3, 4}, представленный на рис. 7.1.

Графы G вида $\{N; 1, \omega_2, \ldots, \omega_n\}$, т. е. D_n-графы или циркулянты с единичной образующей (Loop Networks — петлевые структуры) интенсивно изучаются в последнее время. Циркулянтные структуры $\{N; 1, \omega_2\}$ широко внедрены в практику ВС, и читатели уже имели с ними дело при изучении матричных ВС (см. гл. 5). Так, например, на рис. 5.3 изображен циркулянт {64; 1, 8}, отражающий структуру квадранта ВС ILLIAC-IV.

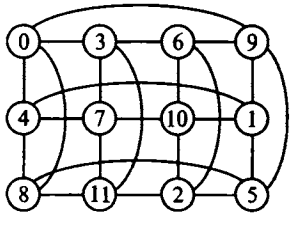
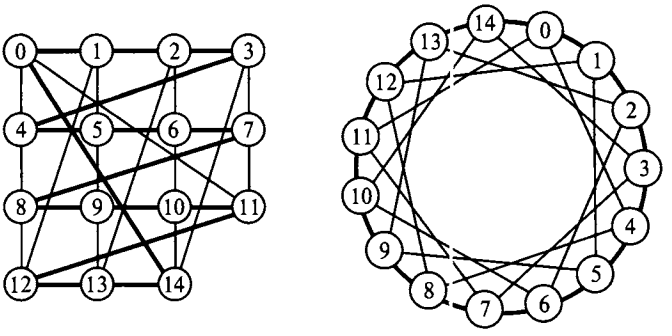


Рис. 7.1. D_2-граф: {12; 3, 4}



а *б*

Рис. 7.2. D_2-граф: {15; 1, 4}:

а — двумерная матрица; *б* — хордовое кольцо

Структуры, изображенные на рис. 5.3 и 7 1, представлены в виде двумерных матриц. На практике часто используется изображение структур как хордовых колец (рис. 7.2).

Целые числа $i \in \{0, 1, 2, \ldots, N-1\}$, отмечающие вершины D_n-графа, называют *адресами. Адресация* вершин в таких структурах *называется диофантовой* (в честь древнегреческого математика из Александрии Диофанта, Diophantos, III в.). *В циркулянтных структурах при полном переносе какой-либо подструктуры* (всех вершин подструктуры на одно и то же расстояние в одном из направлений) *сохраняются все ее свойства и адресация вершин*. Таким образом, при диофантовой адресации элементарных машин ВС можно простыми средствами реконфигурации осуществить виртуальную адресацию вершин-машин и, следовательно:

1) создавать отказоустойчивые параллельные программы, неориентированные на физические номера машин;

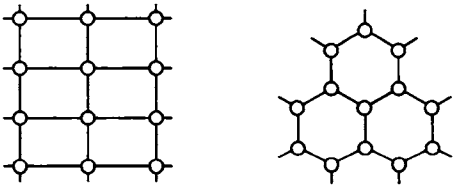
2) реализовывать мультипрограммные режимы обработки информации;

3) исключать отказавшие вершины-машины из подсистем, а значит, обеспечить живучесть ВС.

Пусть параллельная программа реализуется на некоторой подсистеме ВС (т. е. на подграфе в пределах D_n-графа). Далее, пусть i — физический номер элементарной машины (вершины D_n-графа), исключаемой из подсистемы (подграфа), а j — номер ЭМ, включаемой в нее, $i, j \in \{0, 1, \ldots, N-1\}$. Тогда, очевидно, алгоритм преобразования виртуального адреса α машины, используемого в параллельной программе, сводится к его изменению по формуле

$$\alpha := [\alpha + (j - i)] \mod N,$$

где $\alpha \in \{0, 1, \ldots, N-1\}$, $i, j \in \{0, 1, \ldots, N-1\}$.



а *б*

Рис. 7.3. Фрагменты $Л(N, v, g)$-графов:
а — $v = 4$, $g = 4$; *б* — $v = 3$, $g = 6$

В качестве структур ВС, допускающих масштабирование (изменение числа машин) без коренной перекоммутации уже имеющихся межмашинных связей, используются $Л(N, v, g)$-*графы*. В такие графы [18] вкладываются D_n-графы; $Л(N, v, g)$-граф — это неориентированный однородный граф с числом и степенями вершин соответственно N и v и значением обхвата g (рис. 7.3).

В $Л(N, v, g)$-графах каждая вершина при $v \geqslant 3$ входит в не менее v кратчайших простых циклов длиной g (длина кратчайшего цикла в графе называется *обхватом*). При $v = 2$ $Л(N, v, g)$-граф является простым циклом с N вершинами.

Анализ и синтез структур ВС. Введенные показатели (7.1)–(7.4) в равной степени пригодны и для анализа, и для синтеза структур ВС. Как в том, так и в другом случае требуется осуществлять расчет значений структурных показателей ВС. Получение аналитических выражений для координат вектор-функций структурной коммутируемости ВС (7.3) и структурной живучести (7.4) является сложной задачей, разрешимой лишь для частных случаев. Для расчета этих показателей используют метод статистического моделирования.

Проблема синтеза структур заключается в поиске таких графов G^*, которые бы реальные (физические) конфигурации ВС делали максимально приспособленными для программирования виртуальных конфигураций. Если при проектировании ВС преследуется цель ее адаптации под какую-либо область применения, под класс решаемых задач, то физическая структура G^* должна быть максимально приспособлена для программной настройки проблемно-ориентированных виртуальных конфигураций. Если же требуется достичь живучести ВС, то G^* должна быть адаптирована под программирование живучих виртуальных конфигураций. Или же, если при создании ВС требуется максимизировать эффективность использования ЭМ, то определяется струк-

тура G^*, которая минимизирует задержки (Latency — латентность) при транзитных передачах информации между ЭМ.

На этапе выбора структур ВС заметную роль играют интуитивные соображения. Так, например, было очевидно, что кольцевые и тороидальные структуры ВС обладают большей живучестью, чем «линейка» и «решетка» соответственно.

Проблема синтеза структур ВС может быть сформулирована с ориентацией на любой из структурных показателей (7.1)–(7.4). Здесь мы дадим следующую постановку: найти структуру G^* которая обеспечивала бы максимум координаты вектор-функции структурной живучести (7.4), т. е.

$$\max_{G} \mathcal{L}_r(G, s, s') = \mathcal{L}_r(G^*) \qquad (7.5)$$

при заданных значениях N, r, v, s, s'. Структура G^*, для которой выполняется (7.5), называется *оптимальной*. В упрощенной постановке можно ограничиться поиском G^* в некотором классе структур, например в классе D_n- или $Л(N, v, g)$-графов.

К сложным относится проблема синтеза оптимальных структур большемасштабных ВС, она практически решается при помощи статистического моделирования (методом Монте-Карло) и, следовательно, с использованием мощных вычислительных средств. Трудоемкость поиска G^* можно заметно снизить, если воспользоваться двумя нижеприведенными гипотезами.

***Гипотеза* 1**. Структура G^*, при которой достигается $\mathcal{L}_N(G^*)$ — максимум живучести ВС, обеспечивает и $\mathcal{L}_r(G^*)$ — максимум живучести подсистем ранга $r < N$.

***Гипотеза* 2**. Структура с минимальным (средним) диаметром относится к G^*, т. е. обладает максимальной структурной живучестью.

Справедливость гипотез, высказанных автором в 1970-х годах, подтверждена результатами статистического моделирования структур ВС.

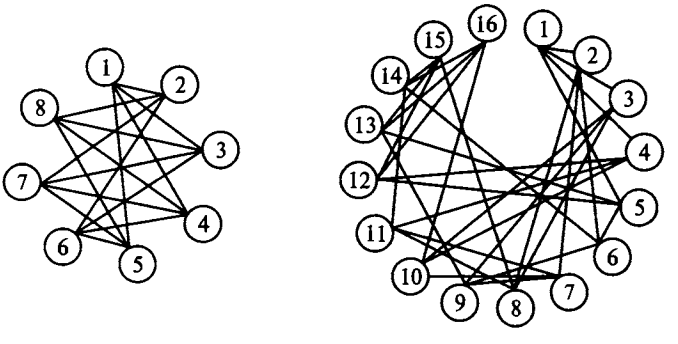
Ниже *оптимальными* будем называть структуры G^*, имеющие при заданных порядке N и степени v вершин минимальный диаметр. Создание общего алгоритма синтеза оптимальных структур является сложной задачей. Существуют алгоритмы синтеза для конкретных классов графов. Для практических целей созданы и пополняются каталоги оптимальных структур. Пользование каталогами так же просто, как таблицами элементарных функций. Фрагмент каталога оптимальных D_n-графов отражен в табл. 7.1.

Структуры ВС в виде оптимальных $Л(N, v, g)$-графов показаны на рис. 7.4. В этих структурах достигнуты минимумы диаметра d (7.1) и сред-

него диаметра $\bar{d}$ (7.2) и, следовательно, минимумы задержек при передаче информации между ЭМ в ВС.

Таблица 7.1

D_n-граф	Характеристика					
	N	ω_1	ω_2	ω_3	ω_4	ω_5
D_2-граф	16	1	6			
	32	1	7			
	64	1	14			
	128	1	15			
	256	1	92			
D_3-граф	16	1	2	6		
	32	1	4	10		
	50	1	8	12		
	2048	37	116	202		
		48	407	615		
		349	390	686		
D_4-граф	16	1	2	3	4	
	32	1	2	8	13	
	64	1	4	10	17	
D_5-граф	16	1	2	3	4	5
	32	1	2	3	4	12
	50	1	3	8	16	20
	1024	22	189	253	294	431
		30	133	230	253	485
		6	317	403	425	475



а *б*

Рис. 7.4. Оптимальные $Л(N, v, g)$-графы:

а — $Л(8, 4, 4)$-граф, $d = 2$, $\bar{d} = 1,43$; *б* — $Л(16, 4, 4)$-граф, $d = 3$, $\bar{d} = 1,91$

Графы $Л(N, v, g)$ можно описать в виде матриц смежности. Пусть запись $i, j, k, \ldots, l$ представляет элементы i-й строки матрицы смежности $Л(N, v, g)$-графа, которые равны 1. В качестве примера оптимальных $Л(N, v, g)$-графов ниже приведены описания матриц смежности для $N = 32$:

$Л(32, 3, 7)$-граф, $d = 5$, $\overline{d} = 2,94$

1. 2,3,4	9. 4,19,20	17. 8,12,26	25. 13,16,26
2. 1,5,6	10. 4,21,22	18. 8,19,29	26. 17,20,25
3. 1,7,8	11. 5,22,29	19. 9,18,3	27. 14,20,28
4. 1,9,10	12. 5,17,23	20. 9,26,27	28. 21, 27,32
5. 2,11,12	13. 6,24,25	21. 10,23,28	29. 11,18,32
6. 2,13,14	14. 6,15,27	22. 10,11,30	30. 15,22,31
7. 3,15,16	15. 7,14,30	23. 12,21,24	31. 19,24,30
8. 3,17,18	16. 7,25,32	24. 13,23,31	32. 16,28,29

$Л(32, 4, 5)$-граф, $d = 4$, $\overline{d} = 2,38$

1. 2,3,4,5	12. 4,8,11,26	23. 10,22,26,30
2. 1,6,7,8	13. 4,9,27,28	24. 10,25,27,31
3. 1,9,10,11	14. 4,15,18,29	25. 11,19,24,29
4. 1,12,13,14	15. 5,11,14,30	26. 12,20,23,27
5. 1,15,16,17	16. 5,18,20,31	27. 13,24,26,32
6. 2,17,18,32	17. 5,6,21,22	28. 13,21,29,31
7. 2,10,19,20	18. 6,14,16,19	29. 14,25,28,32
8. 2,9,12,21	19. 7,18,22,25	30. 15,23,31,32
9. 3,8,13,22	20. 7,16,21,26	31. 16,24,28,30
10. 3,7,23,24	21. 8,17,20,28	32. 6,27,29,30
11. 3,12,15,25	22. 9,17,19,23	

$Л(32, 4, 6)$-граф, $d = 4$, $\overline{d} = 2,36$

1. 2,3,4,5	12. 4,20,24,29	23. 9,13,29,32
2. 1,6,7,8	13. 4,23,27,28	24. 12,25,26,31
3. 1,9,10,11	14. 4,21,22,32	25. 9,15,24,30
4. 1,12,13,14	15. 5,21,25,27	26. 7,10,17,24
5. 1,15,16,17	16. 5,6,28,29	27. 10,13,15,22
6. 2,16,18,22	17. 5,18,19,26	28. 13,16,30,31
7. 2,21,26,29	18. 6,17,31,32	29. 7,12,16,23
8. 2,19,30,32	19. 8,9,17,20	30. 8,10,25,28
9. 3,19,23,25	20. 11,12,19,22	31. 11,18,24,28
10. 3,26,27,30	21. 7,11,14,15	32. 8,14,18,23
11. 3,20,21,31	22. 6,14,20,27	

Для наглядности на рис. 7.5 представлен граф для последней матрицы смежностей.

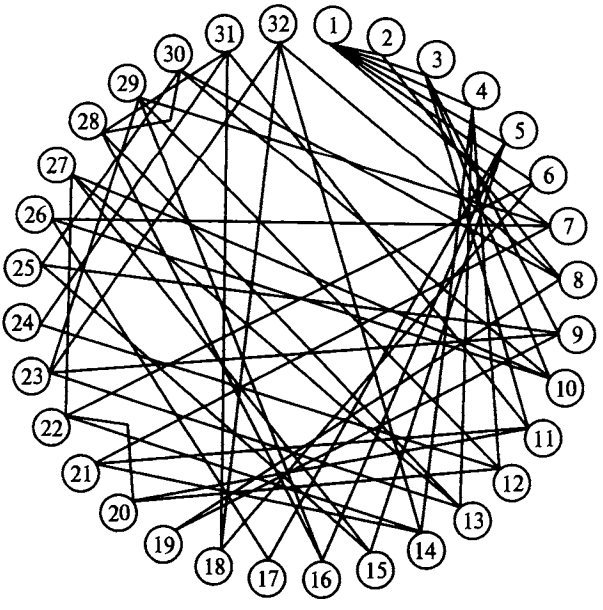


Рис. 7.5. Оптимальный $Л(32, 4, 6)$-граф: $d = 4$; $\overline{d} = 2,36$

Очевидно, что в рассматриваемых графах с 32 вершинами увеличение степени вершины v от 3 до 4 приводит к уменьшению диаметра d от 5 до 4, а среднего диаметра $\overline{d}$ — от 2,94 до 2,38. При фиксации степени вершины $v = 4$ увеличение обхвата в графе приводит к некоторому уменьшению среднего диаметра от $\overline{d} = 2,38$ до $\overline{d} = 2,36$.

Как уже отмечалось (см. разд. 3.1.2), гиперкубы весьма популярны при формировании сетей межмашинных связей в большемасштабных ВС (см. § 5.4). Представляет интерес сравнить структурные показатели гиперкубов, оптимальных D_n- и $Л(N, v, g)$-графов. В табл. 7.2 отражены результаты сравнения названных структур по степени v, диаметру d (7.1) и среднему диаметру $\overline{d}$ (7.2) для одинаковых чисел $N = 2^v$ вершин графов. При этом следует заметить, что степень вершины D_n-графа (циркулянта) равна степени гиперкуба или (в случае нечетной степени) меньше на единицу. На рис. 7.6 представлены графики зависимости диаметров гиперкубов D_n- и $Л(N, v, g)$-графов от количества вершин N.

Заметим, что диаметры D_n- и $Л(N, v, g)$-графов значительно меньше диаметров гиперкубов даже при одинаковых (или меньших на единицу) степенях их вершин. Более того, D_n- и $Л(N, v, g)$-графы обладают и меньшими

Таблица 7.2

$N = 2^v$	Гиперкубы		Циркулянты			$Л(N, v, g)$-графы			
	$v = d$	$\bar{d}$	v	d	$\bar{d}$	v	g	d	$\bar{d}$
64	6	3,0	6	4	2,5	6	6	3	2,29
256	8	4,0	8	4	3,3	8	6	3	2,7
512	9	4,5	8	5	4,02	9	6	3	2,81
1024	10	5,0	10	5	4,04	10	6	4	3,01
2048	11	5,5	10	6	4,70	11	6	4	3,47
4096	12	6,0	12	6	4,68	12	6	4	3,57
8192	13	6,5	12	6	5,34	13	6	4	3,78
16384	14	7,0	14	6	5,38	14	6	4	3,83
32768	15	7,5	14	7	6,09	15	6	4	3,89
65536	16	8,0	16	7	6,12	16	6	5	4,06
131 072	17	8,5	16	8	6,73	17	6	5	4,39
262 144	18	9,0	18	8	6,75	18	6	5	4,62
1 048 576	20	10,0	20	8	7,41	20	6	5	4,85
16 777 216	24	12,0	24	10	8,76	24	6	6	5,56
268 435 456	28	14,0	28	11	10,1[illegible]	28	6	6	5,94

средними диаметрами по сравнению с гиперкубами. Рассматриваемые показатели для $Л(N, v, g)$-графов при $g > 4$ являются самыми лучшими: так, диаметры для оптимальных $Л(N, v, g)$-графов оцениваются величиной $0{,}21 \log_2 N$, в то время как в гиперкубах — $\log_2 N$. Кроме того, $Л(N, v, g)$-графы характеризуются логарифмической зависимостью диаметров от ко-

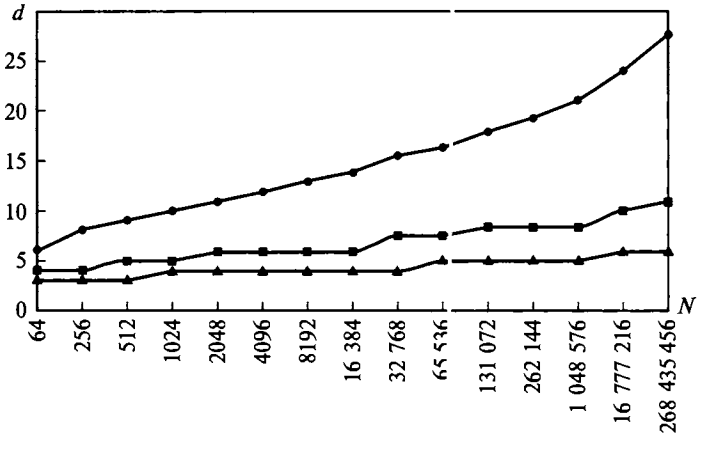


Рис. 7.6. Зависимость диаметра структуры ВС от количества вершин:
• — гиперкуб; ▪ — D_n-граф; ▲ — $Л(N, v, g)$-граф

личества N вершин при фиксированной степени вершин. Из сказанного следует, что в ВС, структура которых является D_n- и $Л(N, v, g)$-графами, время межмашинных обменов информацией значительно меньше по сравнению с временем гиперкубических ВС.

Таким образом, D_n- и $Л(N, v, g)$-графы более перспективны для формирования сетей межмашинных (межпроцессорных) связей в ВС, чем гиперкубы*.

Численный анализ показал, что ВС с программируемой структурой при существующей физико-технологической базе могут обладать высокой структурной живучестью. Учет топологии межмашинных связей и надежности линий связи не приводит к существенной разнице между реальной и потенциально возможной живучестью ВС. Потенциальная живучесть ВС оценивается в предположении, что структура сети связей идеальна (т. е. она абсолютно надежна и обеспечивает связность любых подмножеств элементарных машин). Высокая живучесть ВС с программируемой структурой объясняется тем, что в них достигается близкая к 1 вероятность существования связной подсистемы исправных машин, включающей 80...90 % их общего числа, уже при четырех межмашинных связях в каждой ЭМ.

7.2.2. Режимы функционирования вычислительной системы и способы обработки информации

В зависимости от сложности задач и характера их поступления можно выделить следующие *основные режимы работы ВС с программируемой структурой:*

- решение одной сложной задачи;
- обработка набора задач;
- обслуживание потока задач.

Первый режим — *монопрограммный*, т. е. для решения задачи используются все ресурсы ВС. Задача представляется в виде параллельной программы, число ветвей в которой либо фиксировано, либо допускает варьирование в заданном диапазоне. В качестве единицы ресурса выступает элементарная машина ВС. Все машины используются для решения задачи. Если максимальное количество ветвей в параллельной программе менее общего количества ЭМ в системе, то избыточные машины используются для повышения надежности функционирования ВС.

---

* *Монахов О.Г. Монахова Э.А.* Исследование топологических свойств регулярных параметрически описываемых структур вычислительных систем // Автометрия, 2000. № 2. С. 70–82.

Второй и третий режимы функционирования ВС относятся к *мультипрограммным*. При работе ВС в этих режимах одновременно решается несколько задач, следовательно, ресурсы системы делятся между несколькими задачами.

При организации функционирования ВС в случае набора задач учитывается не только количество задач, но их параметры: число ветвей в программе (точнее, число машин, на которых она будет выполняться), время решения или вероятностный закон распределения времени решения и др. Алгоритмы организации функционирования ВС задают распределение задач по машинам и последовательность выполнения задач на каждой машине. В результате становится известным, в какой промежуток времени и на каких машинах (или на какой подсистеме) будет решаться любая задача набора. Этот режим, безусловно, является обобщением мультипрограммных режимов для ЭВМ, и он более сложный. В самом деле, при мультипрограммировании ресурсы ЭВМ (прежде всего процессор) делятся между несколькими последовательными программами. При обработке наборов параллельных задач ресурсы ВС (множество элементарных машин) также распределяются между задачами, однако в любой момент времени задачи решаются на непересекающихся подмножествах машин. Следовательно, мультипрограммные режимы работы ЭВМ реализуются путем разделения времени процессора, в то время как обработка наборов задач на ВС осуществляется посредством разделения «пространства» машин.

Третий режим — обслуживание потока задач на ВС — принципиально отличается от обработки задач набора: задачи поступают в случайные моменты времени, их параметры случайны, следовательно, детерминированный выбор подсистем для решения тех или иных задач исключен. Для режима потока задач созданы методы и алгоритмы [5], обеспечивающие стохастически оптимальное функционирование ВС.

Следует подчеркнуть, что при работе ВС в любом из мультипрограммных режимов система представляется в виде композиции подсистем различных рангов. По мере решения задач эта композиция «автоматически» (с помощью ОС) реконфигурируется так, чтобы обеспечить ее адекватность текущей мультипрограммной ситуации. Любая подсистема обладает всеми архитектурными свойствами системы, поэтому ее организация при решении выделенной ей задачи может осуществляться теми же методами, что и организация работы всей ВС в первом режиме.

Технология решения произвольной задачи на ВС (или на ее части — подсистеме) предусматривает следующие этапы:

- выбор способа обработки данных;
- разработку параллельного алгоритма (в общем случае отказоустойчивого, способного адаптироваться на количество работоспособных ЭМ как на

параметр), который эффективно реализуется на ВС при заданной ее структуре и выбранном способе обработки данных;

- запись (параллельного) алгоритма решения задачи на языке (высокого уровня);
- получение объектной программы решения задачи на системе.

Различают *распределенный, матричный и конвейерный способы обработки информации*. Последние два способа обработки информации получили наибольшее распространение в виде высокопроизводительных (порядка $10^8 \ldots 10^{12}$ опер./с) промышленных матричных и конвейерных ВС (см. гл. 4 и 5). Принципы, положенные в основу ВС с программируемой структурой (см. гл. 3), позволяют реализовать в них любой из названных выше способов обработки данных.

При *распределенной обработке* параллельные программы и данные рассредотачиваются по элементарным машинам ВС (рис. 7.7, *а*). Допустимо построение адаптирующихся параллельных программ, число ветвей в которых в процессе их реализации соответствует числу (работоспособных) ЭМ в системе. Способ распределенной обработки данных был теоретически и экспериментально исследован в широком диапазоне классов сложных задач.

Опыт применения методики крупноблочного распараллеливания при решении сложных задач на действующих ВС показал высокую эффективность параллельных программ для распределенной обработки информации и позволил сделать выводы, приведенные в разд. 3.3.6.

В случае *матричной обработки данных* (рис. 7.7, *б*) программа вычислений содержится в одной (управляющей) ЭМ, а данные однородно распределяются по всем машинам ВС (или подсистемы). Процесс решения задачи состоит из чередующихся процедур: рассылки команд из управляющей ЭМ остальным машинам и исполнения этих команд всеми машинами, но каждой над своими операндами.

Матричный способ в сравнении с распределенным дает экономию в использовании (распределенной по ЭМ) памяти ВС. Однако данному способу присущ недостаток, заключающийся в неоднородном использовании машин и, в частности, в неоднородной нагрузке на их память. Этого недостатка лишен *обобщенный матричный способ обработки информации* (рис. 7.7, *в*). При этом способе программа не целиком помещается в одной ЭМ, а предварительно сегментируется (не распараллеливается, а сегментируется!) и затем посегментно размещается в памяти машин. Последовательность сегментов, составляющих программу, может быть размещена в памяти машин, например так, что номер распределенного в машину сегмента будет равен ее номеру. И для распределенного, и для матричного способов обработки информации характерно то, что в процессе решения задачи имеют место обмены данными между ЭМ системы.

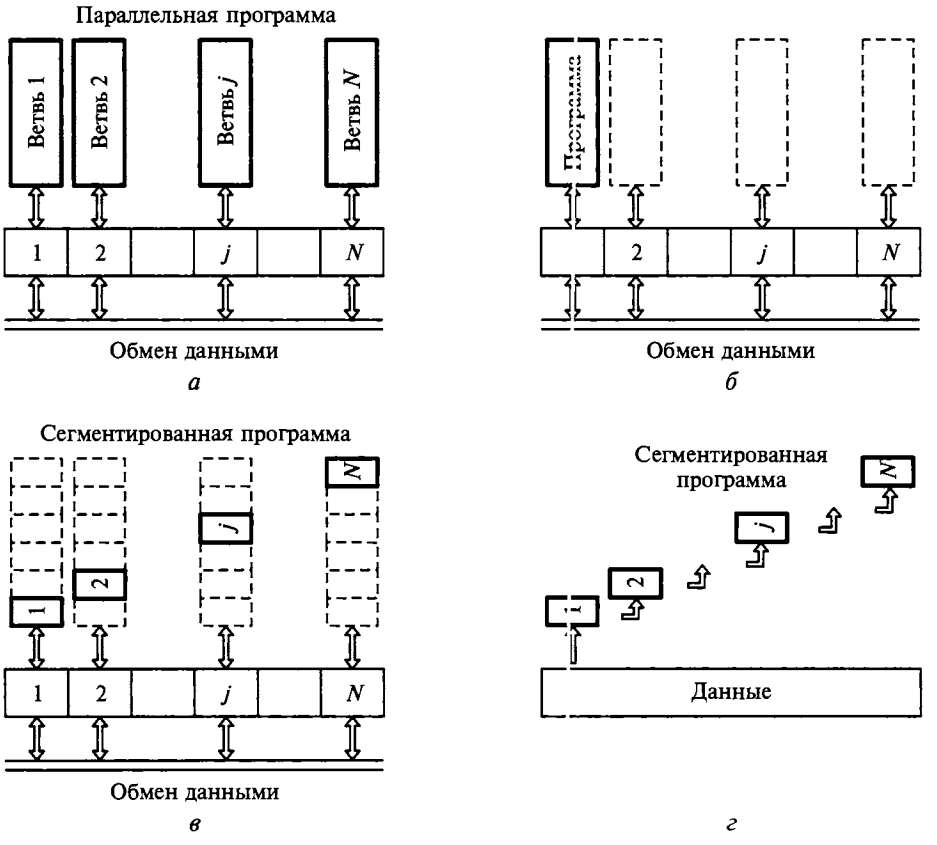


Рис. 7.7. Способы обработки информации:

а — распределенный; *б* — матричный; *в* — обобщенный матричный; *г* — конвейерный; ⇨ — направление потоков данных

При *конвейерном способе обработки данных* (рис. 7.7, *г*) структура ВС предварительно настраивается так, что машины образуют конвейер (или «линейку», или «кольцо»). Затем осуществляются сегментирование программы и размещение в машинах ВС последовательности полученных сегментов в соответствии со структурой конвейера. Размещение данных может быть сосредоточенным (например, на внешней памяти одной ЭМ) или распределенным (по памятям всех машин конвейера). В процессе решения задачи данные проходят через последовательность машин, составляющих конвейер.

Рассмотренные способы обработки информации на ВС иллюстрирует рис. 7.7. Номер $j \in \{1, 2, \ldots, N\}$, приписанный к блокам данных и программ (к ветви или сегменту), соответствует номеру ЭМ, в памяти которой они размещены. Штриховой линией изображены недостающие части програм-

мы, которые будут получены машиной в процессе решения задачи от ЭМ, хранящих программы или ее сегменты.

Таким образом, при распределенном способе обработки информации полностью используются возможности ВС с архитектурой MIMD. Матричный и конвейерный способы обработки информации обеспечивают частичное использование возможностей архитектуры ВС. Архитектура MIMD при первом способе трансформируется в архитектуру SIMD, а при втором — в архитектуру MISD (см. § 3.4).

7.2.3. Архитектурные аспекты при создании операционных систем ВС

Отметим те из архитектурных особенностей ВС с программируемой структурой, которые оказывают определяющее влияние на ОС — комплекс средств для управления работой ВС. Архитектура ВС должна удовлетворять главнейшему требованию — реализации в системе параллельных вычислений во всех режимах. Рассмотренная в § 3.3 методика крупноблочного распараллеливания позволяет для сложных задач представить вычисления в виде совокупности большого числа слабо связных ветвей. Анализ показал, что круг задач, допускающих такое представление, достаточно широк. Более того, методика, как правило, обеспечивает построение идентичных ветвей и таких, что каждая из них состоит в основном из операторов-преобразователей данных, находящихся в памяти выделенной для этой ветви ЭМ, и относительно редко требует обращений к памяти других ЭМ. Данное свойство снижает влияние структуры ВС — сети межмашинных связей и, в частности, системного устройства на время обращения к общей рассредоточенной по ЭМ памяти системы.

Опыт распараллеливания сложных задач, составления параллельных программ и их вложения в структуры ВС показал, что около 90 % всех обращений ЭМ к памяти других машин составляют групповые и регулярные схемы обмена информацией, а именно: трансляционная схема («одна — всем», т. е. когда информация передается из одной ЭМ всем остальным машинам, участвующим в реализации параллельной программы), трансляционно-циклическая схема («каждая — всем») и конвейерно-параллельная («каждая — своим соседям»). Групповые схемы обмена информацией выдвигают требование простоты их реализации в структуре ВС (достаточной приспособленности системных устройств к их выполнению), а также предопределяют наличие в ОС средств организации этих обменов.

В основе организации параллельных вычислений в ВС лежит представление вычислений в виде совокупности *совместно протекающих асинхронных взаимодействующих процессов*. Под процессом здесь понимается совокупность последовательных действий (операций) при реализации в ре-

альном времени некоторого алгоритма, использующего часть ресурсов системы. *Совместность процессов* означает не только обычную для мультипрограммных систем (в частности, систем разделения времени) одновременность реализации алгоритмически независимых процессов (разделение ресурсов ВС), но и существование связи между отдельными процессами, которая обусловлена тем, что они представляют собой части одного сложного алгоритма (объединение ресурсов ВС).

В общем случае для ВС характерен мультипрограммный режим работы даже при решении задач, представленных в параллельной форме. Мультипрограммирование является следствием разделения ресурсов ВС между:

- процессами, относящимися к различным параллельным программам пользователей;
- процессами пользовательских программ и процессами операционной системы;
- отдельными процессами, относящимися к одной программе решения задачи.

Управление работой ВС в мультипрограммном режиме возлагается на ядро (резидентную часть) ОС и непосредственно осуществляется с помощью операций управления процессами.

Базовый набор средств, обеспечивающий управление процессами (как процессами пользователей, так и процессами самой ОС) каждой ЭМ ВС, составляют средства для порождения и уничтожения процессов и для реализации взаимодействий между ними. На основе базового набора строятся все процессы ОС элементарной машины ВС: управление распределением ресурсов, связь с внешними устройствами и с пользователями, связь с ОС других ЭМ. Последние обеспечивают слияние идентичных операционных систем ЭМ в операционную систему ВС. Ясно, что такой способ построения для ВС операционной системы делает ее распределенной, приводит к высокой живучести ВС как единого аппаратурно-программного комплекса, делает ВС приспособленной к развитию и сокращению, следовательно, позволяет просто подобрать конфигурацию ВС, которая будет адекватна сфере применения.

Временно́е и пространственное распределение аппаратурно-программных ресурсов системы (процессоров, оперативной памяти, внешних устройств, системных устройств, линий связи, программ, данных) между совместно протекающими процессами обеспечивается благодаря полноте реализации в ВС трех канонических принципов модели коллектива вычислителей. Так, программируемость структуры позволяет уже на этапе программирования сложной задачи не только указывать последовательность реализации процессов (подчиненность процессов по данным и по управлению), но и задавать размещение этих процессов в пространстве ресурсов ВС. Это размеще-

ние осуществляется с учетом характера взаимодействия процессов, свойств самих процессов и структуры обрабатываемых данных. Оно определяет структуру подсистемы — области ВС, выделяемой для решения задачи. Программируемость структуры ВС позволяет и в условиях мультипрограммности осуществлять временно́е и пространственное распределение ресурсов между совокупностями процессов, относящимися к различным задачам.

Программируемость структуры ВС достигается аппаратурно-программными средствами, среди которых уже известные системные устройства и специальные программные блоки ОС. Последние обеспечивают управление структурой ЭМ, управление структурой связей между ними (настройку сети межмашинных связей), а также организацию взаимодействий между процессами, реализуемыми в разных ЭМ системы (межмашинных взаимодействий).

Межмашинные взаимодействия в ВС осуществляются с помощью системных операций (представляемых в виде либо команд, либо микропрограмм, либо подпрограмм). Один из возможных полных наборов системных операций дифференцирует все взаимодействия между ЭМ системы на:

- обмен информацией, предназначенной для указания отношений между машинами ВС и режимов их функционирования (настройку ВС — задание для системных ресурсов состояний, которые используются при реализации взаимодействий между процессами);
- обмен рабочей информацией (пересылку данных или программ между оперативными памятями ЭМ — обмен между процессами или порождение процессов);
- обмен информацией, предназначенной для выработки значения некоторой булевой функции (которая используется для определения отношения машин системы к выполняемой программе, следовательно, для синхронизации процессов и осуществления условных переходов при протекании процессов);
- обмен командами, позволяющий из любой ЭМ осуществлять управление работой других машин (изменение состояний процессов, уничтожение процессов).

Все отмеченные архитектурные возможности достигаются в ВС с программируемой структурой с помощью специальных средств, которые в комплексе с традиционными средствами организации функционирования и составляют ОС. Функции ОС, как правило, реализуются с помощью программных средств. Однако уже в современных условиях технологии микропроцессорных БИС возможна и аппаратурная реализация основных функций ОС.

В общем случае ОС имеет иерархическую структуру. За начальное условие для решения проблемы управления берется описание операционной

обстановки в ВС. Оно включает сведения о текущем состоянии реализуемых процессов и о распределении между ними ресурсов системы. Описание операционной обстановки разбивается на уровни, соответствующие функциональной обособленности ресурсов ВС (например, ресурсы любой ЭМ обособлены от ресурсов остальных машин; ресурсы подсистемы — от ресурсов других подсистем). Каждому уровню функциональной обособленности ресурсов ВС соответствует свой уровень ОС. За ним закрепляются параметры, значения которых вырабатываются на основе анализа соответствующей части описания операционной обстановки и служат исходными данными для вышестоящего уровня. Каждый уровень ОС допускает представление в виде совокупности процессов, реализуемых в различных ЭМ ВС.

7.3. Вычислительная система «Минск-222»

Интерес к практической реализации ВС с программируемой структурой постоянно проявлялся, начиная с 60-х годов XX в. Первоначально он поддерживался прежде всего необходимостью проверки теоретических основ построения ВС, необходимостью отработки архитектурных решений и функциональной структуры ВС, а также параллельных вычислительных технологий. Позднее возрастающую роль стал играть утилитарный компонент целей создания ВС, в 1970-х годах этот компонент стал превалировать над исследовательским. Последнее обосновывается потребностью в ВС, обладающих и высокой производительностью, надежностью и живучестью.

Работы по построению ВС, основанных на принципах коллектива вычислителей, были инициированы в Институте математики (ИМ) Сибирского отделения АН СССР в 1964 г. Вскоре в ИМ СО АН СССР было организовано и мини-производство ВС.

Ниже будет описана первая ВС с программируемой структурой «Минск-222». В проекте «Минск-222» были отработаны архитектурные, технические и программные решения, значительная часть из которых была «канонизирована» разработчиками не-фон-неймановских вычислительных средств. Описание архитектуры системы «Минск-222» будет сделано с полнотой, достаточной для учебных целей.

Система «Минск-222» была разработана и построена Отделением вычислительной техники ИМ СО АН СССР совместно с Конструкторским бюро завода им. Г.К. Орджоникидзе Министерства радиопромышленности СССР (г. Минск). Руководитель работ по созданию ВС «Минск-222» — Э.В. Евреинов; основные разработчики: В.Г. Хорошевский, Б.А. Сидристый, Г.П. Лопато (1924–2001; чл.-кор. РАН с 1979 г.), А.Н. Василевский. Работы по проектированию ВС «Минск-222» были начаты в 1965 г., а первый ее образец был

установлен в апреле 1966 г. в Институте математики АН БССР. Системы «Минск-222» были смонтированы в нескольких организациях Советского Союза и эксплуатировались более 15 лет.

Архитектура ВС «Минск-222»:

- MIMD-архитектура, распределенность ресурсов;
- параллелизм, однородность, программируемость структуры;
- одномерная (кольцевая) топология;
- масштабируемость: 1–16 элементарных машин (ЭМ);
- быстродействие: $\Omega = AN\omega$, где N — число ЭМ, ω — быстродействие одной ЭМ, $A \geqslant 1$ (при крупноблочном распараллеливании сложных задач);
- использование промышленных ЭВМ второго поколения.

7.3.1. Функциональная структура вычислительной системы «Минск-222»

По структуре ВС «Минск-222» относилась к одномерным (точнее, кольцевым) системам (рис. 7.8). Для связи между элементарными машинами в ВС применялись двусторонние каналы. Количество N ЭМ в системе могло изменяться в пределах от 1 до 16. Каждая ЭМ системы «Минск-222» имела свой абсолютный номер $i \in \{0, 1, ..., N-1\}$.

Элементарная машина состояла из вычислительного модуля и системного устройства (СУ). В качестве ВМ были использованы серийные ЭВМ «Минск-2» или «Минск-22», выпускавшиеся заводом им. Г.К. Орджоникидзе (г. Минск). Указанные ЭВМ имели одну и ту же архитектуру и были не только совместимы, а, по сути, являлись конфигурациями одной и той же двухадресной машины. Машина «Минск-22» в сравнении с «Минск-2» обладала магнитной памятью удвоенной емкости (8 К 37-разрядных слов) и имела дополнительный набор устройств ввода-вывода информации.

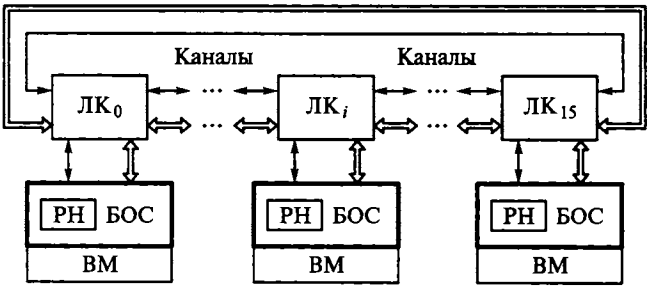


Рис. 7.8. Функциональная структура системы «Минск-222»:

ЛК — локальный коммутатор; БОС — блок операций системы; РН — регистр настройки; ВМ — вычислительный модуль; ⇨ — рабочий канал; → — управляющий канал

Обращаем внимание читателей на то, что подход к построению параллельных ВС, ориентированный на применение серийных ЭВМ, был впервые применен в Сибирском отделении АН СССР [5], а не за рубежом (см., например, разработки 70-х годов XX в. Университета Карнеги—Меллона, § 6.2 и 6.6, а также современные кластерные ВС).

В состав СУ входили локальный коммутатор (ЛК) каналов связи и блок операций системы (БОС). Коммутатор $ЛК_i$ состоял из вентилей, которые открывали или закрывали канал связи, идущий к соседней справа ЭМ, т. е. к коммутатору $ЛК_j$, где $j = i + 1 (\mathrm{mod}\ N)$. (Запись $j(\mathrm{mod}\ N)$ означает число j по модулю N.) Вентили управлялись сигналами, поступавшими из БОС.

Блок операций системы включал в себя регистр настройки (РН) и узел, реализующий системные команды. Содержимое РН определяло вид соединительной функции коммутатора и степень участия ЭМ при системных взаимодействиях. Регистр настройки состоял из трех разрядов: TR, TQ, $T\Omega$.

Триггер TR позволял разбивать систему на функционально изолированные подсистемы. Для того чтобы образовать подсистему из машин с номерами $i + 1(\mathrm{mod}\, N)$, $i + 2(\mathrm{mod}\, N)$, ..., $i + m(\mathrm{mod}\, N)$, где $m \leqslant N$, достаточно было в разряды TR регистров настройки элементарных машин i и $i + m(\mathrm{mod}\, N)$ и машин $i + 1(\mathrm{mod}\, N)$, $i + 2(\mathrm{mod}\, N)$, ..., $i + m - 1(\mathrm{mod}\, N)$ записать соответственно 1 и 0.

Триггеры TQ и $T\Omega$ конкретизировали степень участия машин в выполнении некоторых системных команд. В частности, триггеры $T\Omega$ использовали для выделения машин, участвовавших в выработке обобщенного признака Ω_k ($k = 1, 2, 3$), который управлял ходом вычислений. Признак

$$\Omega_k = \underset{i \in E}{\&}\, \omega_{ki}, \quad k = 1, 2, 3, \tag{7.6}$$

где E — подмножество номеров машин, управлявших ходом вычислений (т. е. отмеченных единицей в разряде $T\Omega$), $E \subseteq \{0, 1, \ldots, N-1\}$; ω_{ki} — признаки, вырабатываемые ЭМ с номером i, причем $\omega_{1i} = 1$, если знак последнего вычислительного результата был меньше нуля, $\omega_{2i} = 1$, если происходило переполнение, $\omega_{3i} = 1$, если последний вычислительный результат арифметико-логического устройства был равен нулю. В конфликтных ситуациях приоритет имела машина с меньшим номером.

Системное устройство было реализовано на 80 стандартных элементах и составляло менее 1,5 % объема оборудования АЛУ и устройства управления ЭВМ «Минск-22».

7.3.2. Системные команды вычислительной системы «Минск-222»

К системным относят команды, обеспечивающие функциональную целостность множества ЭМ как коллектива. С другой стороны, системные команды — это средства для организации и реализации параллельных вычислительных процессов, в частности, обменов управляющей информацией и данными между ветвями параллельной программы (см. § 3.3). Набор системных команд ВС «Минск-222» составляли *команды настройки, обмена, обобщенных безусловного и условного переходов.* В методическом плане целесообразно дать детальное описание системных команд.

Команды в ЭВМ «Минск-222» представлялись 37-разрядными двоичными числами и имели следующую структуру:

0	1 ... 6	7 ... 12	13 ... 24	25 ... 36
±	КОП	Θ	A1	A2

где ±КОП — код операции, Θ — 6-разрядное поле, два разряда которого определяли номер блока памяти, а четыре остальных — адрес индекс-ячейки, A1 и A2 — соответственно первый и второй адреса.

Команда настройки (Н): — 01 00A1A2. Команда **Н** имела три модификации: $\mathbf{H}_0$, $\mathbf{H}_1$, $\mathbf{H}_2$, отличавшиеся содержимым a_{29} и a_{34} соответственно 29-го и 34-го разрядов (табл. 7.3).

Модификация $\mathbf{H}_0$ изменяла содержимое только РН той ЭМ, в которой она находилась и записывала по A1 прежнее содержимое $RQ\Omega$ регистра настройки, содержимое триггера P режима округления и значение ω_3 — признака нуля. Модификация $\mathbf{H}_2$ выполняла все, что и $\mathbf{H}_0$, и, кроме того, изменяла содержимое триггеров режима округления и признака ω_3.

Модификация $\mathbf{H}_1$ изменяла содержимое РН тех ЭМ подсистемы, которые были отмечены единицами в ее соответствующих разрядах. Соответствие между разрядом j ($13 \leqslant j \leqslant 28$) команды $\mathbf{H}_1$ и номером i настраиваемой ЭМ определялось формулой $j = i + 13$. Ни при каких условиях команда $\mathbf{H}_1$ не могла изменять содержимое РН той ЭМ, в которой она выполнялась.

Информация, предназначавшаяся для РН, являлась содержимым a_{31}, a_{32}, a_{33} разрядов 31, 32, 33 команды **Н**. При выполнении **Н** настроечная информация либо засылалась в РН без изменения (при $a_{30} = 0$), либо поразрядно логически складывалась с содержимым РН при ($a_{30} = 1$). При выполнении $\mathbf{H}_2$, кроме того, засылались a_{35} и a_{36} в триггеры режима округления и признака ω_3 соответственно.

При выполнении $\mathbf{H}_0$ и $\mathbf{H}_2$ предыдущее содержимое R, Q, Ω, P, ω_3 запоминалось соответственно в разрядах 31, 32, 33, 35, 36 ячейки памяти с адресом A1; остальные разряды этой ячейки содержали нули.

Таблица 7.3

Команда	КОП	a_{29}	a_{34}	Модификация	Запись по А1					Условие настройки машины	Настройка		Время, мкс
					a_{31}	a_{32}	a_{33}	a_{35}	a_{36}		при $a_{30}=0$	при $a_{30}=1$	
											Дополнительно		
Настройка (Н)	–01	0	0	$\mathbf{H_0}$	R	Q	Ω	P	ω_3	Всегда	$R := a_{31}$ $Q := a_{32}$ $\Omega := a_{33}$	$R := R \vee a_{31}$ $Q := Q \vee a_{32}$ $\Omega := \Omega \vee a_{33}$	48
		0	1	$\mathbf{H_2}$	R	Q	Ω	P	ω_3	Всегда	$P := a_{35}$ $\omega_3 := a_{36}$	$P := a_{35}$ $\omega_3 := a_{36}$	48
		1	–	$\mathbf{H_1}$	Записи по А1 нет					$a_{13+i}=1$	$R^i := a_{31}$ $Q^i := a_{32}$ $\Omega^i := a_{33}$	$R^i := R^i \vee a_{31}$ $Q^i := Q^i \vee a_{32}$ $\Omega^i := \Omega \vee a_{33}$	80

Таким образом, с помощью команд $\mathbf{H_0}$ и $\mathbf{H_2}$ осуществлялась *самонастройка* элементарной *машины*, а с помощью $\mathbf{H_1}$ — настройка из данной ЭМ остальных машин ВС. Следовательно, команды $\mathbf{H_0}$, $\mathbf{H_1}$, $\mathbf{H_2}$ представляют собой средства самонастройки ВС. С помощью этих команд программировались структура ВС (связность машин, т. е. канал межмашинных связей) и участие каждой ЭМ в выполнении системных функций. Они позволяли разбивать систему на подсистемы, а значит, и создавать «среду» для мультипрограммной обработки данных.

Команды обмена — это команда передачи (**П**): — **56 00** ***l*** **б** и команда приема (**ПР**): — **57 00** ***h*** **в**. Команда **П** (табл. 7.4) предназначалась для выдачи из передающей ЭМ в канал связи l кодов начиная с кода, расположенного в ячейке α памяти. После передачи l кодов в передающей ЭМ выполнялась очередная команда. По команде **ПР** осуществлялся прием из канала h кодов в ячейки $\beta, \beta+1, \ldots, \beta+h-1$ оперативной памяти принимающей ЭМ. Прием из канала разрешался лишь при условии, что в него поступило очередное слово из передающей ЭМ. Машина, выполнявшая команду приема, могла приступить к выполнению следующей команды только после приема h кодов. В случае $h > l$ ЭМ находилась в состоянии приема, пока не поступали остальные $(h-l)$ кодов. Очевидно, что перед началом обмена требовались настройка межмашинного канала и синхронизация.

Такой способ организации обменов информацией между машинами подобен функционированию радио- и телепередающих систем. В самом деле, передатчик осуществляет безадресную выдачу информации в эфир (передающая ЭМ направляет данные в межмашинный канал без указания адресов ЭМ), а прием этой информации из эфира будет выполняться только теми приемниками, которые настроены на частоту электромагнитных волн передатчика (а прием данных из канала будет реализовываться только ЭМ, выполняющими команду приема).

Предлагаемый способ реализации межмашинных обменов информацией не зависит от числа ЭМ в системе и позволяет избежать трудностей с адресацией машин. Это особенно важно для ВС с массовым параллелизмом. Как уже отмечалось ранее, современные ВС обладают, в частности, двумя свойствами:

- большемасштабности (количество ЭМ в системе может иметь порядок 10^6);
- масштабируемости (архитектура ВС должна быть рассчитана на переменное количество ЭМ, в современных условиях в пределах от 10 до 10^6).

Следовательно, при введении прямой адресации машин в системе потребовались бы специальное адресное слово длиной в 20 двоичных разрядов

Таблица 7.4

Команда	КОП	a_7	a_{13}	a_{14}	a_{15}	Модификация	Отличие	Условие выполнения	Пояснение		Время, мкс
Передача (**П**)	–56	A1 = l — число передаваемых кодов, A2 = α — начальный адрес памяти									$48l + 52$
Прием (**ПР**)	–57	A1 = h — число принимаемых кодов, A2 = β — начальный адрес памяти									$48h + 76$
Обобщенный безусловный переход (**ОБП**)	–02	0	–	–	–	**ОБП**$_0$	ЭМ i ожидает	$Q^i = 1$	ЭМ i исполняет команду из ячейки A2 управляющей ЭМ		96
		1	–	–	–	**ОБП**$_1$	ЭМ i не ожидает	$Q^i = 1$			100
Обобщенный условный переход (**ОУП**)	–65	–	0	0	0	**ОУП**$_0$	Синхронизация	Есть **ОУП** во всех ЭМ, в которых $\Omega = 1$	Передача управления		48
									дальше	по A2	
		–	1	0	0	**ОУП**$_1$	ω_1		Всегда	–	48
		–	0	1	0	**ОУП**$_2$	ω_2		$\Omega_1 = 0$	$\Omega_1 = 1$	48
		–	0	0	1	**ОУП**$_3$	ω_3		$\Omega_2 = 0$	$\Omega_2 = 1$	48
									$\Omega_3 = 0$	$\Omega_3 = 1$	48

Примечание. i — номер управляемой ЭМ.

и достаточно сложная аппаратурная поддержка (особенно в многомерных ВС). Создание средств прямой адресации, рассчитанных на предельное количество ЭМ, конечно, обеспечивает масштабируемость «вниз» (в направлении уменьшения числа ЭМ). Однако ясно, что конфигурации ВС с числом ЭМ, менее предельного, обладают бесполезной аппаратурной избыточностью. С другой стороны, такой «предел» с развитием ВТ и технологии вычислений постоянно отодвигается «вверх» (в сторону увеличения числа ЭМ). Описанные команды межмашинного обмена информацией не требуют сложной аппаратурной поддержки, они адекватны большемасштабным и масштабируемым ВС.

Команда обобщенного безусловного перехода (ОБП): — 02 $a_7$0 0000 A2. Команда **ОБП** предназначалась для безусловного управления параллельными вычислительными процессами. Она имела две модификации: **ОБП$_0$** и **ОБП$_1$**, отличавшиеся значением a_7 (см. табл. 7.4). Управляющая машина при выполнении **ОБП** засылала в канал содержимое ячейки A2 своей памяти, которое воспринималось как команда только ЭМ подсистемы, содержащими единицу в разрядах TQ регистров настройки. В случае **ОБП$_0$** поступившая команда выполнялась после окончания текущей, а в случае **ОБП$_1$** — немедленно (прерывая выполнение текущей).

В управляющей ЭМ после выполнения команды **ОБП** осуществлялась реализация следующей команды. Во всех управляемых ЭМ поступившей команде присваивались номера последних из исполнявшихся команд (т. е. в них происходила блокировка изменения счетчика адресов).

Команда **ОБП** позволяла осуществлять инициирование работы ВС и ее загрузку данными из любой ЭМ, а также вмешиваться в параллельные вычислительные процессы и принудительно управлять работой подмножеств машин системы из любой ЭМ. Следовательно, при помощи команды **ОБП** из любой ЭМ можно было и инициировать программирование структуры ВС. В самом деле, при выполнении команды **ОБП** из данной машины могла быть передана для исполнения любая команда (в том числе и системная) в любые элементарные машины ВС.

Команда обобщенного условного перехода (ОУП): — 65 00 A1 A2. Эта команда служила для управления параллельными вычислительными процессами по обобщенному условию (7.6). Команда **ОУП** употреблялась в четырех модификациях: **ОУП$_k$**, $k \in \{0, 1, 2, 3\}$, которые отличались значением вектора $a_{13}a_{14}a_{15}$ (см. табл. 7.4). Команда **ОУП** могла быть выполнена в машине, в регистре настройки которой было содержимое $\Omega = 1$; в противном случае команда **ОУП** воспринималась как пустая команда +00. Реализация перехода по команде **ОУП** осуществлялась одновременно во всех машинах (имевших $\Omega = 1$) после их выхода на команду **ОУП**. При выполнении модификации **ОУП$_0$** всегда осуществлялся переход к следующей

команде. Следовательно, **ОУП$_0$** служила для *синхронизации машин* подсистемы. При реализации **ОУП$_k$**, $k = 1, 2, 3$, в машине происходила передача управления либо следующей команде при $\Omega_k = 0$, либо команде по адресу А2 при $\Omega_k = 1$.

Команда **ОУП** использовалась, в частности, для организации ветвлений в параллельных вычислениях, для реализации параллельных циклов.

7.3.3. Программное обеспечение вычислительной системы «Минск-222»

Программное обеспечение вычислительной системы «Минск-222» — это расширенное ПО ЭВМ «Минск-22». Возможности ЭВМ «Минск-22», являвшейся машиной второго поколения, были весьма ограничены. Поэтому расширение ПО было ориентировано лишь на реализацию в монопрограммном режиме параллельных программ (*P*-программ) решения сложных задач. Программное обеспечение ВС «Минск-222» состояло из двух частей (рис. 7.9): системы *P*-программирования и пакета прикладных адаптирующихся *P*-программ.

Система *P*-программирования. Система параллельного программирования включала средства автоматизации *P*-программирования, отладки, редактирования и анализа *P*-программ.

Рис. 7.9. Программное обеспечение ВС «Минск-222»

Средства автоматизации Р-программирования — языки и трансляторы. В качестве входных в системе «Минск-222» использовались расширенные языки: автокод АКИ, ЛЯПАС, ALGOL, BASIC. Язык АКИ — АвтоКод «Инженер» — относится к машинно-ориентированным. Язык ЛЯПАС — Логический Язык Программирования Алгоритмов Синтеза — оригинальный язык, ориентированный для описания алгоритмов решения задач математической логики, теории автоматов, булевой алгебры, теории графов и теории кодирования (создан в 1966 г. школой специалистов-логиков Томского государственного университета). Язык ALGOL (ALGOrithmic Language) — язык программирования высокого уровня, предназначенный для описания алгоритмов решения вычислительных задач. Язык BASIC (Beginner's All-purpose Symbolic Instruction Code) — это язык программирования высокого уровня, отличается простотой, легко усваивается начинающими программистами. В расширенные языки были включены средства для описания взаимодействий между параллельными ветвями вычислений. Каждый транслятор для ВС имел два блока: машинный и системный. Машинный блок представлял собой обычный транслятор для ЭВМ, системный служил для реализации межмашинных взаимодействий в ВС.

Разработка трансляторов для расширенных языков АКИ, ЛЯПАС и ALGOL свелась к созданию системных блоков. Это было обеспечено тем, что имевшиеся трансляторы для ЭВМ «Минск-22» с языков АКИ, ЛЯПАС и ALGOL (транслятор ТАМ-2/22) предусматривали возможность использования библиотеки стандартных программ, записанных в машинных кодах.

Любой системный блок представлял собой совокупность программ для реализации операций настройки, обмена, ОБП и ОУП, которые были включены в библиотеки трансляторов. При использовании языка АКИ обращение к программе, реализующей требуемую системную операцию, осуществлялось при помощи оператора КОД; при применении ЛЯПАС — посредством оператора «↓» перехода к машинному языку; наконец, при применении ALGOL — при помощи оператора STANDARD.

Системные блоки, добавляемые к трансляторам ЭВМ, имели относительно небольшое количество команд (см. табл. 7.5). Так, отношение объема системного блока к общему для трансляторов с АКИ и ЛЯПАС не превышало 0,1, а для транслятора с ALGOL (ТАМ-2/22) — 0,025. Следовательно, качество параллельных программ в основном определялось машинными блоками трансляторов ВС «Минск-222».

Язык BASIC включал обычные операторы, системные операторы и системные стандартные функции. Он позволял описывать как последовательные, так и параллельные вычислительные процессы. Реализованный транслятор (вместе с сервисными программами) имел объем 6600 команд, скорость трансляции 700–800 команд готовой программы в минуту. Транслятор допускал ввод и исправление исходных программ в режиме диалога.

Таблица 7.5

Компоненты программного обеспечения ВС «Минск-222»		Общий объем	Объем системного блока
Система *Р*-программирования	Трансляторы для языков:		
	– АКИ	4000	300
	– ЛЯПАС	4000	300
	– ALGOL (ТАМ-2/22)	16 000	300
	– BASIC	6600	300
	Средства отладки и редактирования *Р*-программ	2200	640
	Средства анализа *Р*-программ	850	460
Пакет *Р*-программ	Адаптирующиеся *Р*-программы для решения:		
	– задач линейной алгебры и математического программирования	3000	250
	– систем дифференциальных уравнений	2200	80
	– информационно-логических задач	3500	160
	– статистического моделирования сложных процессов	4000	75
Итого слов:		46 350	2865

Средства отладки и редактирования Р-программ — совокупность четырех стандартных программ. Первая программа преобразовывала отлаживаемую *Р*-программу в последовательную и выявляла ошибки, не связанные с использованием системных команд. Вторая программа служила для моделирования на одной машине выполнения *Р*-программы из двух ветвей. Всевозможные (допустимые и недопустимые) взаимодействия ветвей были представлены матрицей переходов к моделирующим или аварийным подпрограммам. Эта же программа могла определять время простоев машин, время работы отдельных блоков *Р*-программы и точность вычислений. Третья программа позволяла вывести на печать заданное количество раз содержимое интересующих областей памяти перед выполнением команд обмена в процессе контрольной реализации параллельной программы на ВС «Минск-222». Четвертая программа служила для корректировки *Р*-программ.

Средства анализа Р-программ были представлены тремя стандартными программами. Первая программа служила для анализа распределения памяти между блоками исследуемой программы. Вторая программа предназначалась для измерения времени простоев машин ВС; она заставляла простаивающую машину считать время своего простоя. Третья программа применялась для измерения времени работы участков *Р*-программы; она использовала одну из машин системы как часы.

Пакеты прикладных адаптирующихся *Р*-программ. Пакеты были ориентированы на решение задач повышенной сложности. Параметры таких задач, как правило, не позволяли решать их на ЭВМ «Минск-22» за удовлетворительное время. В пакеты были включены адаптирующиеся *Р*-программы, т. е. такие программы, которые могли настраиваться на число машин ВС как на параметр. Параллельные программы пакета были разделены на четыре группы (см. рис. 7.9).

Из опыта создания ПО для системы «Минск-222» установлено, что его объем отличается от объема программного обеспечения ЭВМ «Минск-22» не более чем на 10 % (табл. 7.5).

7.3.4. Области применения и эффективность вычислительной системы «Минск-222»

Система «Минск-222», имевшая программируемую структуру, позволяла решать задачи, представленные как последовательными, так и параллельными программами с различными количествами ветвей. Следовательно, в число областей применения ВС «Минск-222» включались все области, где использовалась ЭВМ «Минск-22». Кроме того, на ВС «Минск-222» можно было решать научно-технические задачи повышенной трудоемкости. Спектр таких задач достаточно широк; об этом свидетельствует следующий список:

1) решение систем линейных алгебраических уравнений методом простой итерации;

2) решение систем линейных алгебраических уравнений методом Жордана (варианты *а* и *б*);

3) решение систем линейных алгебраических уравнений методом Зейделя;

4) решение систем линейных алгебраических уравнений модифицированным методом Зейделя;

5) решение систем линейных алгебраических уравнений методом Самарского;

6) обращение матриц методом пополнения;

7) отыскание коэффициентов характеристического полинома;

8) умножение матриц;

9) решение систем нелинейных алгебраических уравнений;

10) решение общей задачи линейного программирования;

11) решение транспортной задачи;

12) численное интегрирование;

13) численное дифференцирование;

14) решение задачи интерполирования;

15) решение систем обыкновенных дифференциальных уравнений методом Рунге—Кутта;

16) расчет подвесок автопоездов;
17) решение граничных задач для линейных систем;
18) решение граничных задач для нелинейных систем;
19) решение задачи Дирихле для эллиптических уравнений;
20) решение краевой задачи для параболических уравнений;
21) решение задачи Коши для гиперболических уравнений;
22) решение задачи обтекания тела потоком жидкости;
23) моделирование работы группового реактора;
24) моделирование термодинамических циклов газовых атомных электростанций;
25) моделирование неупругого рассеяния электронов (варианты *а* и *б*);
26) моделирование рассеяния энергии ионизирующего излучения;
27) перспективное изображение непрозрачных объектов;
28) случайный поиск с адаптацией;
29) решение проблем теории статистических решений;
30) решение задачи таксономии;
31) упорядочение массивов;
32) перекрещивание массивов (варианты *а* и *б*);
33) поиск информации в больших массивах (вариантах *а* и *б*);
34) составление ведомостей (варианты *а* и *б*).

Как видно из списка, в него были включены задачи, которые охватывают важнейшие математические разделы и которые в совокупности для своего решения используют основной арсенал вычислительной математики.

В параллельных программах решенных задач были использованы известные схемы обмена информацией между ветвями: трансляционная — Т («одна — всем»), трансляционно-циклическая — ТЦ («каждая—всем»), конвейерно-параллельная — КП («каждая — своим соседям»), коллекторная — К («все — одной») и дифференцированная — Д («любая — любой»). Классификация задач по типам схем обмена приведена в табл. 7.6, из которой видно, что не менее 90 % всех схем составляют первые три схемы обмена. Это обеспечивает в основном параллельную работу ЭМ системы при обменах. Сравнительная редкость применения коллекторной и дифференцированной схем обменов гарантирует незначительные простои машин ВС при реализации параллельных программ.

Примененные схемы обмена между ветвями *Р*-программы приводят к их простой программной реализации. Благодаря этому при решении задач на ВС «Минск-222» системные команды в *Р*-программах составляли, как правило, менее 10 % их общего объема (табл. 7.7). Следовательно, можно считать, что затраты при составлении параллельных и эквивалентных им последовательных программ имеет один и тот же порядок.

Таблица 7.6

Номера задач	Тип схемы обмена				
	Т	ТЦ	КП	К	Д
4, 10, 32б, 33б, 34б	5	0	0	0	0
1, 2а, 3, 6, 9, 11, 15, 16, 18, 25а, 28, 29, 30, 31	0	14	0	0	0
8, 13, 19, 21, 23, 26, 32а, 33а, 34а	0	0	9	0	0
12, 25б, 27	0	0	0	3	0
24	0	0	0	0	1
14, 17, 20, 22	0	4	4	0	0
2б, 7	2	0	2	0	0
5	1	1	1	0	0
Итого:	8	19	16	3	1

Было установлено, что для ВС «Минск-222» доля затрат времени на системные взаимодействия (включая синхронизацию) составляет, как правило, несколько процентов, что является следствием применения методики крупноблочного распараллеливания задач. Кроме того, выяснилось, что за счет большей емкости оперативной памяти в системе «Минск-222» по сравнению с одной ЭВМ «Минск-22» и за счет быстродействия каналов связи, сравнимого с быстродействием ЭВМ при выполнении операций, получается дополнительный значительный выигрыш во времени решения задач на ВС (см. табл. 7.7).

Таблица 7.7

Номер задачи	Число команд		Время τ_N решения на ВС из N машин, мин			$E = \dfrac{\tau_1}{N\tau_N}$
	общее	системных	1	2	3	
1	277	16	3–5	1,5	–	1,0–1,7
6	656	51	42	21,8	5,7	1,0–2,5
7	272	104	144	36	–	2,0
16	857	45	67	19	14	1,6–1,8
22	800	29	65	25	16,7	1,3
23	1167	50	195	34	–	2,9
24	1000	16	10–23	0,25–0,3	–	20–40
25б	400	10	1680	840	–	1,0
26	890	8	–	600	–	

Следует обратить особое внимание на *парадокс параллелизма* (см. табл. 7.7) — нелинейный рост производительности ВС при повышении количества N ЭМ (что противоречит якобы здравому смыслу, см. разд. 3.3.3). Парадокс параллелизма был впервые обнаружен при работе на ВС «Минск-222».

Реакцию на обнаруженный факт легко восстановить, если учесть, что в 60-х годах XX в. существовало устойчивое мнение о невозможности создания параллельных средств обработки информации с большим количеством АЛУ. Считалось и «доказывалось», что увеличение быстродействия ВС происходит только при наращивании количества АЛУ до 10, после чего наблюдается замедление, обусловленное накладными расходами на организацию совместной работы нескольких АЛУ. Да, все это было правильно, только неверными были концептуальные подходы к организации параллельных ВС.

Экспертные оценки показали, что сложность программирования для ВС «Минск-222» (по сравнению со сложностью программирования для одной ЭВМ «Минск-22») возрастает на 10...20 %, а при развитой библиотеке стандартных параллельных программ — на 5...10 %.

Вычислительные системы «Минск-222» в течение многолетней эксплуатации в различных организациях СССР показали высокую эффективность при решении широкого круга задач.

Следует особо подчеркнуть, что архитектурные решения, реализованные в ВС «Минск-222», стали, по сути, каноническими. Схемы обмена информацией между ветвями *P*-программ (см. разд. 3.3.5) и рассмотренные в данном параграфе системные команды нашли отражение в современном инструментарии, используемом при построении распределенных и параллельных ВС. Так, в MPI (Message Passing Interface) — библиотеке функций, предназначенной для поддержки параллельных процессов — применяются как дифференцированный (Point-to-point Communication), так и коллективные взаимодействия (Collective Communications). В табл. 7.8 приведены основные виды системных взаимодействий и реализующие их команды ВС «Минск-222» и функции MPI.

Таблица 7.8

Вид взаимодействия	Команды ВС «Минск-222»	Функции MPI
Дифференцированный обмен	**П, ПР**	MPI_Send, MPI_Recv
Трансляционный обмен	**П, ПР**	MPI_Bcast
Трансляционно-циклический обмен	**П, ПР,** *n* итераций	MPI_Alltoall или MPI_Allscatter
Коллекторный обмен	**П, ПР,** n – 1 итерация	MPI_Gather
Синхронизация элементарных машин	**ОУП$_0$**	MPI_Barrier
Разбиение ВС на подсистемы	**Н$_1$**	MPI_Comm_group MPI_Group_incl MPI_Comm_create MPI_Cart_create

Полученный опыт по проектированию, математической и технической эксплуатации «Минск-222» был использован в последующих проектах вычислительных систем с программируемой структурой.

7.4. Вычислительная система МИНИМАКС

Вычислительные системы, которые формировались из аппаратурно-программных средств мини-ЭВМ, относились к группе мини-ВС. Построение таких ВС было одной из основных тенденций развития ВТ 1970-х годов (см. § 6.2 и 6.6). Опыт эксплуатации показал, что при решении большого круга задач мини-ВС были более эффективны с точки зрения производительности, надежности, живучести и стоимости, чем одна или даже несколько больших ЭВМ третьего поколения.

Работы по созданию ВС из мини-машин достаточно интенсивно велись в США. Однако общей концепции построения таких систем американские специалисты не выработали. Анализ проектов показывает, что использовались в основном три способа организации ВС: системы с общей памятью; ВС с общей шиной (или системой шин), к которой подключались процессоры, запоминающие и другие устройства; системы, в которых машины взаимодействовали через общую группу устройств ввода-вывода информации. Как правило, системы не имели программируемой структуры и обладали ограниченными возможностями к наращиванию.

При создании мини-ВС в Советском Союзе за основу была взята концепция ВС с программируемой структурой (см. разд. 3.4.2, § 7.1, 7.2, [5]).

Архитектурные решения в области мини-ВС, опыт их проектирования, разработки системного и прикладного ПО нашли массовое применение только в конце XX в. Именно вычислительные кластеры являются по существу многопроцессорными или многомашинными ВС, конфигурируемыми из микропроцессоров или персональных ЭВМ (например, IBM PC). При этом заметна архитектурная близость мини-ЭВМ и современных персональных компьютеров.

Ниже описывается МИНИМАшинная программно Коммутируемая Система (МИНИМАКС), созданная Институтом математики СО АН СССР (Отделом вычислительных систем) и Научно-производственным объединением «Импульс» Министерства приборостроения, средств автоматизации и систем управления СССР (г. Северодонецк). Технический проект МИНИМАКС разработан в 1974 г., а опытно-промышленный образец системы был изготовлен и отработан в 1975 г.

Архитектура системы МИНИМАКС:

- MIMD-архитектура;

- распределенность средств управления, обработки и памяти;
- параллелизм, однородность, модульность;
- программируемость структуры;
- двумерная (циркулянтная) топология;
- масштабируемость;
- живучесть;
- максимальное использование промышленных средств мини-ЭВМ.

7.4.1. Функциональная структура мини-ВС МИНИМАКС

Функциональная структура мини-ВС МИНИМАКС — композиция из произвольного количества элементарных машин и программно настраиваемой сети связей между ними. Мини-ВС МИНИМАКС обладала свойством масштабируемости: в ней количество ЭМ не было фиксировано и определялось сферой применения.

Взаимодействия между ЭМ в системе МИНИМАКС осуществлялись через сеть связей (рис. 7.10), которая формировалась из одномерных *1* и двумерных *2* полудуплексных каналов. Одномерные каналы связи *1* были управляющими; они служили для программирования соединений между ЭМ по каналам связи *2*, а также для передачи между ЭМ управляющей информации, регламентирующей использование общих ресурсов (внешних устройств, сервисных программ, файлов и т. п.). Двумерные каналы связи *2* являлись рабочими; они применялись для следующих целей: реализации основных межма-

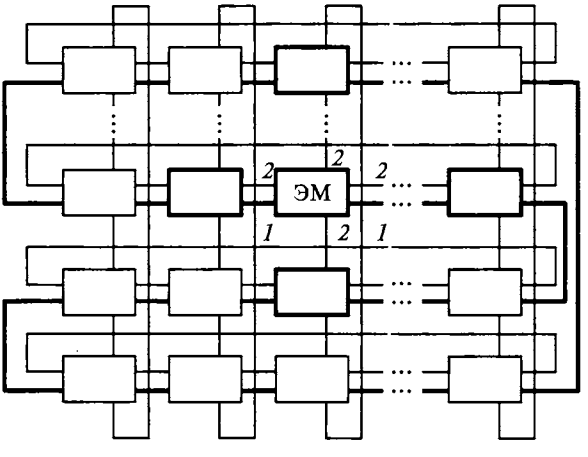


Рис. 7.10. Функциональная структура мини-ВС МИНИМАКС:

ЭМ — элементарная машина; *1* — одномерные управляющие каналы; *2* — двумерные рабочие каналы

шинных взаимодействий, пересылки массивов данных между памятями передающей ЭМ и одной или нескольких принимающих ЭМ, передачи адресов из одной ЭМ в другую и обмена логическими переменными между машинами.

Системные взаимодействия, осуществлявшиеся в мини-ВС, были следующих типов:

1) программное изменение структуры мини-ВС и степени участия ЭМ в системных взаимодействиях (настройка);

2) передача информации из оперативного запоминающего устройства одной ЭМ в памяти других (обмен);

3) выполнение ОБП;

4) синхронизация работы ЭМ;

5) реализация ОУП по значению признака Ω.

Выбором размерности связей между ЭМ достигался компромисс между архитектурной гибкостью и живучестью системы, с одной стороны, и сложностью системных устройств ЭМ — с другой. Именно учет интенсивностей взаимодействий типов 2–5 при функционировании мини-ВС и стремление обеспечить гибкость и живучесть системы обусловили двумерность каналов связи *2*. Сравнительная редкость взаимодействий 1-го типа позволила ограничиться для их выполнения одномерными каналами связи *1*. Необходимая живучесть мини-ВС в целом обеспечивалась замыканием связей *1* в кольцо и возможностью обхода дефектов связей *1* по связям *2* (см. рис. 7.10).

Межмашинные взаимодействия при функционировании мини-ВС реализовывались с помощью специальных подпрограмм — *системных драйверов*, которые, в свою очередь, использовали специальные команды (занесение кода на регистр настройки, считывание его содержимого, занесение информации в СУ о начальном адресе передаваемого массива данных и т. п.).

Структура системы МИНИМАКС, приведенная на рис. 7.10, была удобна для реализации параллельных программ сложных задач. Однако в определенных сферах применения мини-ВС МИНИМАКС (таких, как телеавтоматические системы массового обслуживания, автоматизированные системы управления технологическими процессами) требовались оптимальные структуры (в смысле структурной живучести, см. разд. 7.2.1). Поэтому при компоновке мини-ВС, состоявшей из N ЭМ, использовались для отождествления полюсов связей *1* (см. рис. 7.10) одномерные структуры, а для отождествления полюсов связей *2* — оптимальные двумерные структуры, точнее, D_2-графы. Оптимальные структуры мини-ВС МИНИМАКС при количестве ЭМ от 5 до 10 показаны на рис. 7.11, а при числе ЭМ более 10 могут быть выбраны из каталога оптимальных структур (см. табл. 7.1, 7.2 и 2.4 в [5]).

В пределах мини-ВС МИНИМАКС допускалось формирование произвольного числа подсистем из любого количества ЭМ. Подсистему составляли

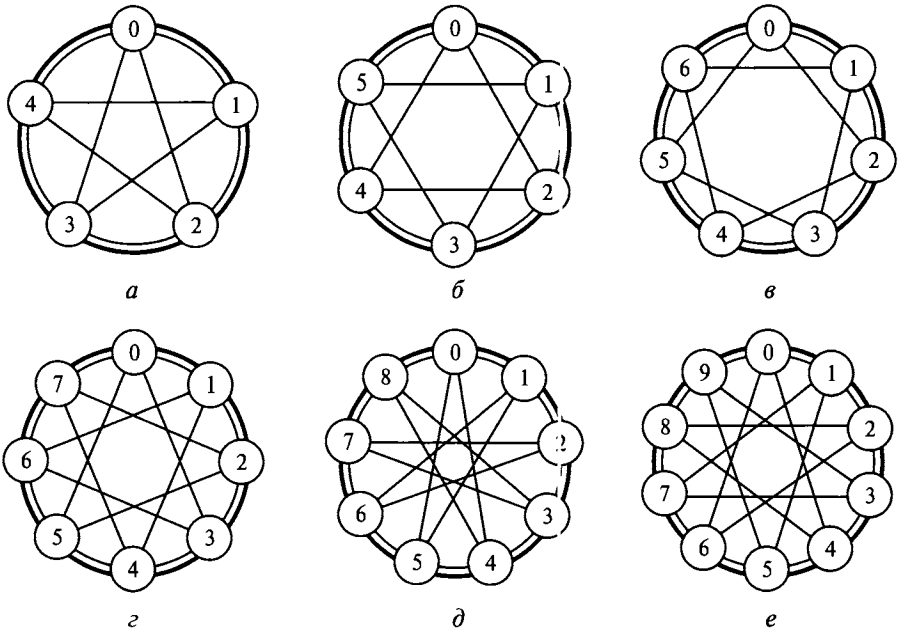


Рис. 7.11. Оптимальные структуры мини-ВС МИНИМАКС, D_2-графы вида:
а — {5; 1,2}; *б* — {6; 1,2}; *в* — {7; 1,2}; *г* — {8; 1,3}; *д* — {9; 1,4}; *е* — {10; 1,4}

взаимодействовавшие друг с другом ЭМ вместе с машинами, которые использовались в качестве транзитных пунктов передачи информации. Каждая ЭМ могла входить только в одну из подсистем (ПС2), образованных по связям *2*. Вместе с тем она могла входить и в одну из подсистем (ПС1), образованных по связям *1*. Машина, принадлежавшая подсистемам ПС1 и ПС2, не могла одновременно участвовать в нескольких взаимодействиях. Подсистемы ПС2 могли сохраняться в течение нескольких следующих друг за другом взаимодействий. Подсистемы ПС1 образовывались только на время одного взаимодействия и разрушались после его выполнения.

7.4.2. Элементарная машина мини-ВС МИНИМАКС

Вычислительная система МИНИМАКС компоновалась из ЭМ-многополюсников (рис. 7.12). Каждая ЭМ — это композиция из ВМ и СУ. Структура ЭМ данной мини-ВС не была жестко заданной и определялась областью применения. Состав каждой ЭМ допускал варьирование; компоновка ЭМ проводилась по правилам, которые были приняты для агрегатных средств ВТ на микроэлектронной основе (АСВТ-М) или для средств систе-

мы малых ЭВМ (СМ ЭВМ). В качестве ВМ могли быть использованы любые конфигурации мини-ЭВМ на базе процессоров М-6000, М-7000, СМ-1П. Архитектура системы МИНИМАКС была рассчитана также на применение мини-ЭВМ моделей HP 2114—2116 семейства Hewlett—Packard.

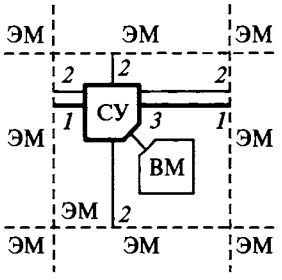


Рис. 7.12. Функциональная структура ЭМ мини-ВС МИНИМАКС:
ВМ — вычислительный модуль; ЭМ — элементарная машина; СУ — системное устройство; *1–3* — связи

Основные технические характеристики мини-ЭВМ семейств АСВТ-М, СМ ЭВМ и Hewlett—Packard:

- базовый набор команд — 70 инструкций;
- структура команд машины — одноадресная;
- система счисления — двоичная;
- длина слов — 16 двоичных разрядов;
- способы представления чисел — с фиксированной и плавающей запятыми (режим работы с плавающей запятой реализовывался программно и аппаратурно, специальным процессором);
- многоуровневая непрямая адресация памяти;
- емкость магнитной оперативной памяти — 4…32 К слов;
- цикл оперативной памяти — 1,2…2 мкс;
- многоуровневая система прерываний (для обслуживания запросов различных приоритетов);
- быстродействие при выполнении операций типа «сложение» — 250…400 тыс. опер./с;
- время выполнения команд:

  сложения (регистр-память) — 2,5…5 мкс,
  умножения (по подпрограмме) — 150…187 мкс,
  деления (по подпрограмме) — 310…387 мкс,
  умножения (спецпроцессором) — 19,2…24 мкс,
  деления (спецпроцессором) — 20,8…26 мкс.

Системное устройство было спроектировано как автономное устройство АСВТ-М. Оно подключалось к ВМ через связи *3* (рис. 7.13). При выборе способа реализации связей *3* учитывались принципы построения АСВТ-М и следующие отсюда ограничения:

- целесообразность построения системного устройства в виде отдельного модуля;
- недопустимость изменений в схемах и конструкции процессоров АСВТ-М.

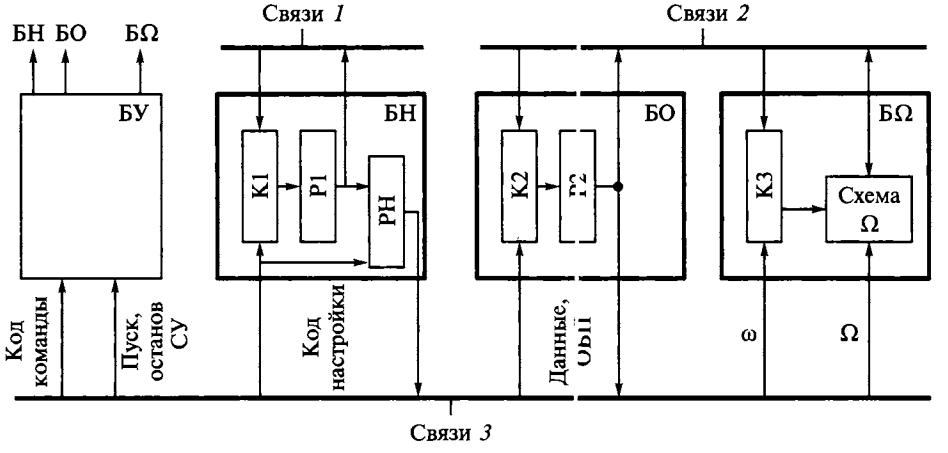


Рис. 7.13. Функциональная структура системного устройства мини-ВС МИНИМАКС:

БН — блок настройки; БО — блок обмена информацией; БУ — блок управления; БΩ — блок выработки обобщенного признака; $К_1$, $К_2$ — коммутаторы; $РН_1$, $РН_2$ — регистры настройки

В мини-ВС МИНИМАКС связь ВМ с СУ реализовывалась двумя способами:

- через программный канал;
- через канал прямого доступа к памяти.

При этих способах использовался стандартный интерфейс ввода-вывода АСВТ-М.

В состав СУ входили: блок управления БУ, блок настройки БН, блок обмена информацией БО, блок выработки обобщенного признака Ω БΩ (см. рис. 7.13).

Блок управления служил для реализации алгоритма работы системного устройства в соответствии с кодами системных операций. Управляющие сигналы данного блока координировали работу остальных — БН, БО, БΩ.

Блок настройки. Настройка ЭМ заключалась в изменении кода в ее РН. Содержимое 3-разрядного РН указывало, выполняет ли данная ЭМ взаимодействия по связям *2* и задает ли направление приема информации для взаимодействий *2* и *3* (два разряда для кода соединительной функции). Настройка могла быть выполнена из собственного ВМ через связи *3* (настройка машины) или из любой другой ЭМ через связи *1* (настройка системы).

Настройка системы могла осуществляться в любом из направлений связей *1*. Использовалась пошаговая настройка со сдвигом. На каждом шаге к подсистеме ПС1, сформированной ранее, подсоединялась ЭМ, соседняя с подсистемой в направлении выполняемой настройки; выдавался очередной код из ВМ настраивающей ЭМ на ее регистр Р1; осуществлялся параллель-

ный сдвиг содержимого Р1 машин подсистемы на одну позицию в направлении настройки. Общее количество шагов настройки системы было равно $(l+2)$, где l — число ЭМ, находящихся между настраивающей и настраиваемой машинами. После выдачи $(l+2)$ кодов настраивающая ЭМ посылала исполнительный сигнал, который осуществлял в каждой ЭМ ПС1 перепись кода из Р1 на РН, а затем разрушал подсистему. В машинах ПС1, использовавшихся для транзитной передачи кодов, перепись содержимого Р1 в РН блокировалась. Транзитные ЭМ распознавались по кодам, находившимся в их регистрах Р1 (см. рис. 7.13).

Направление приема информации по связям *1* для ЭМ определялось с помощью коммутатора К1. Он реализовывал функцию ИЛИ от входной информации, если ЭМ не была включена в ПС1. Для ЭМ, принадлежавшей ПС1, направление приема определялось направлением, с которого она получила первый код.

Пошаговая настройка позволяла образовывать по связям *1* несколько одновременно работающих подсистем, что уменьшало общее время настройки заданной области мини-ВС.

Время, в течение которого связи *1* были свободны, можно было использовать для передачи в ЭМ кода настройки, необходимого для ее дальнейшей работы. При предварительной настройке перепись информации из Р1 на РН в настраиваемых ЭМ не выполнялась. Код на Р1 сохранялся, а в ЭМ посылался сигнал готовности Р1.

Блок обмена. Этот блок предназначался для выполнения передач информации между оперативными памятями машин, выделенных при настройке, и для пересылок адресов передачи управления при операциях обобщенного безусловного перехода.

Передача информации из ЭМ осуществлялась по всем четырем направлениям связей *2*. Направление приема информации определялось коммутатором К2, соединительная функция которого задавалась содержимым регистра настройки (см. рис. 7.13).

Передача информации по связям *2* осуществлялась с использованием сигналов готовности, что позволяло автоматически учитывать колебания циклов работы взаимодействующих ЭМ. Каждая передающая ЭМ начинала цикл выборки на регистр Р2 очередного кода после получения сигналов окончания приема текущего содержимого Р2 от всех соседних с нею принимающих ЭМ данной подсистемы. Передающая ЭМ сигнализировала соседним с нею принимающим ЭМ и собственному ВМ об изменении кода в Р2 посылкой управляющего сигнала.

Блок Ω. Блок Ω служил для синхронизации машин и осуществления обобщенного условного перехода в мини-ВС. Оба этих взаимодействия сопровождались выработкой обобщенного признака Ω. Признак Ω вырабаты-

вали и реагировали на его значение только ЭМ, выделенные при настройке. Обобщенный признак Ω представлял собой конъюнкцию индивидуальных признаков ω, поступавших из ВМ. Значение Ω вырабатывалось по схеме И, распределенной по машинам системы. Интервал времени, в течение которого сигнал на выходах блока Ω соответствовал значению признака Ω, выделялся программно.

Регистр кода операций блока местного управления, регистры Р1 и Р2 блоков настройки и обмена, триггер обобщенного признака Ω были программно доступны.

Контроль. Информация, передаваемая по связям *1–3*, контролировалась по паритету. Следовательно, вычисления по модулю *2* контрольных сумм кодов осуществлялись до их передачи и после приема. Если эти контрольные суммы не совпадали (одна была четной, а другая — нечетной), то имела место ошибка. Она вызывала выработку сигнала прерывания в соответствующем ВМ. Реакция мини-ВС на возникновение ошибки определялась содержимым регистра настройки и источником ошибки (связи *1*, *2* и т. д.).

Системное устройство было включено в состав агрегатных модулей АСВТ-М. По аппаратурным затратам оно было не сложнее канала прямого доступа к памяти. Отношение стоимости системного устройства к стоимости процессора не превышало 0,5.

Следует заметить, что для мультипроцессорных систем с общей шиной и вычислительных сетей фирмы Digital Equipment Corporation отношение стоимости соединительного модуля к стоимости процессора ЭВМ PDP-11 составляло 1,2 и 2,1 соответственно.

Логическая сложность системных устройств ВС МИНИМАКС и «Минск-222» имели один и тот же порядок. Так, число вентилей на два входа в системных устройствах для вычислительных систем МИНИМАКС и «Минск-222» было равно соответственно 830 и 440. При этом функциональные возможности мини-ВС МИНИМАКС, имевшей двумерные рабочие каналы межмашинной связи, были существенно шире, чем одномерной системы «Минск-222».

7.4.3. Системные команды мини-ВС МИНИМАКС

Межмашинные взаимодействия при функционировании мини-ВС реализовывались при помощи *системных драйверов* и *команд*. Системный драйвер — специальная подпрограмма, которая выполнялась в ВМ и осуществляла:

- формирование кода системной команды в соответствии с требованиями к ее формату;
- выдачу в СУ кода команды и сигналов инициирования ее выполнения;

• контроль за ходом выполнения системной команды (включая реакцию на сигналы системного устройства).

Кратко опишем системные команды мини-ВС МИНИМАКС.

1. *Настройка* подразделялась на настройку машины и настройку системы.

Настройка машины проводилась с помощью одной из двух команд. Первая осуществляла занесение кода, заданного в команде, на регистр настройки, вторая — перепись содержимого регистра Р1 на регистр настройки.

Настройка системы выполнялась при помощи четырех команд, определявших направление настройки (вправо или влево от настраивающей ЭМ) и ее характер (занесение кода настройки на регистр настройки или Р1).

2. *Обмен* осуществлялся при помощи *команд передачи* и *приема*. Начальный адрес участка памяти, используемого для хранения передаваемого (или принимаемого) массива, и объем массива задавался путем выполнения соответствующих драйверов в ЭМ. Машина могла перейти к выполнению очередной системной команды только после завершения передачи (приема) заданного количества слов.

3. *Обобщенный безусловный переход* использовался для передачи управления в машинах по адресу, поступавшему из управляющей машины. Для реализации обобщенного безусловного перехода использовались команды передачи адреса и приема адреса. Первая команда выполнялась в управляющей машине, вторая — в управляемой ЭМ.

4. *Синхронизация* работы машин осуществлялась с помощью специальной команды. Машины, выполнявшие эту команду, вырабатывали значения индивидуальных признаков $\omega = 1$. Синхронизация достигалась, когда все синхронизируемые машины вырабатывали значения обобщенного признака $\Omega = 1$.

5. *Обобщенный условный переход* реализовывался по специальной команде. Эту команду обязаны были выполнить все машины, в которых требовалось осуществить условную передачу управления по значению обобщенного признака Ω (конъюнкции индивидуальных признаков ω). Перед выполнением данной команды требовалась синхронизация работы машин.

При реализации обмена, синхронизации, обобщенных условного и безусловного переходов после окончания основной команды обязательно выполнялась и вспомогательная. Функции последней команды, использовавшейся для завершения любых взаимодействий по связям *2*, заключались в следующем:

• передаче информации о коллизиях в ВМ;

• приведении в исходное состояние некоторых узлов СУ.

Обмен, синхронизация, обобщенный условный и безусловный переходы выполнялись в пределах ПС2 — подсистемы, организованной по связям *2*. (Связность машин в подсистеме определялась хранившимися на регистрах настройки кодами соединительных функций.) Участвовали в этих взаимодействиях только машины, отмеченные в регистрах настройки.

Коллизии, возникавшие в мини-ВС МИНИМАКС при осуществлении межмашинных взаимодействий, разрешались программно-аппаратурными средствами.

7.4.4. Программное обеспечение мини-ВС МИНИМАКС

В основу программного обеспечения системы МИНИМАКС было положено ПО мини-ЭВМ. Программное обеспечение мини-ВС состояло из управляющей системы, системы параллельного программирования, пакетов прикладных *Р*-программ и комплекса программ технического обслуживания (рис. 7.14).

Управляющая система обеспечивала связь мини-ВС с пользователями и внешними объектами, осуществляла рациональное использование ресурсов системы с учетом неисправных компонентов и входной мультипрограммной ситуации. Управляющая система имела иерархическую структуру и содержала блоки: главный диспетчер, старший диспетчер и диспетчер.

Главный диспетчер осуществлял программирование структуры и определял динамику ее перенастройки в процессе функционирования мини-ВС МИНИМАКС. Следовательно, он выполнял разбиение системы на подсистемы с требуемым количеством машин и внешних устройств, формировал топологию и определял режимы работы образуемых подсистем.

Каждая подсистема могла функционировать в одном из четырех режимов: автономной работы, параллельной обработки, профилактики, управления. В режиме автономной работы осуществлялась непосредственная связь пользователя с одной машиной. При этом обеспечивались возможности счета, трансляции, отладки, редактирования программ на одной машине, а также доступ к системной информации, формирование заданий системе, связь пользователей, работавших за различными терминалами. В режиме параллельной обработки реализовывались *Р*-программы и обеспечивалось продолжение счета при выходе машин из строя. В режимах профилактики и управления выполнялись функции соответственно тестирования и организации работы системы.

Динамика перенастройки структуры мини-ВС и ее конфигурация в каждый момент времени определялись главным диспетчером на основе ин-

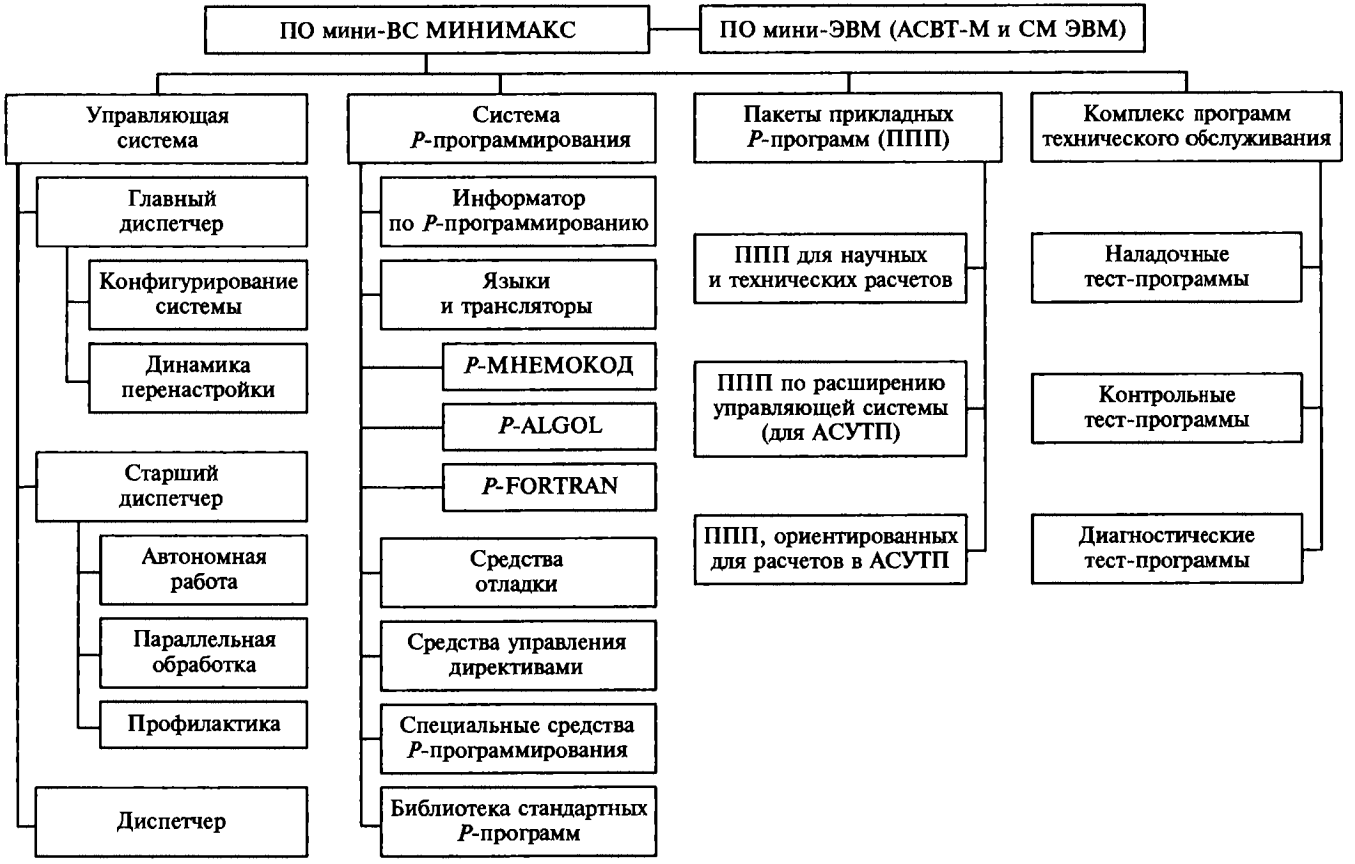


Рис. 7.14. Программное обеспечение мини-ВС МИНИМАКС

формации о суммарных ресурсах и об отказах компонентов, директив человека-оператора, характеристик мультипрограммной ситуации на входе мини-ВС. Мультипрограммная ситуация описывалась параметрами программ и способом их поступления. Различали два способа поступления программ: в виде пакета и потока.

Старший диспетчер организовывал функционирование подсистем в режимах автономной работы, параллельной обработки и профилактики. Он взаимодействовал с главным диспетчером.

Функции главного и старшего диспетчеров выполнялись подсистемой, работавшей в режиме управления. Диспетчеры имели столько идентичных параллельных ветвей, сколько было машин в подсистеме. Каждая ветвь обслуживала выделенную ей сферу влияния. Основные взаимодействия ветвей были связаны:

- с дублированием некоторой информации;
- с обеспечением доступа к информации данной сферы по запросам из других сфер влияния;
- с перераспределением сфер влияния в связи с изменением мультипрограммной ситуации.

У старшего диспетчера имелась еще одна возможность. Он хранился в той же подсистеме, что и главный диспетчер, но его функции реализовывались в других подсистемах. Последнее обеспечивалось путем соответствующей пересылки его частей в подсистемы, сформированные главным диспетчером.

Диспетчер управлял работой элементарных машин. Он участвовал в реализации системных взаимодействий (настройки, обмена, синхронизации, обобщенных условного и безусловного переходов), анализировал коллизии и аварийные ситуации, распознавал директивы пользователя, управлял индивидуальными внешними устройствами. Функции диспетчера реализовывались каждой ЭМ.

Система Р-программирования обеспечивала достаточно широкие возможности при производстве программ для мини-ВС МИНИМАКС и включала в себя шесть нижеследующих компонентов.

1. Информатор по параллельному программированию являлся средством обучения пользователей способам представления вычислительных процессов в виде совокупностей ветвей, взаимодействующих между собой с помощью системных операторов.

2. Языки *Р*-МНЕМОКОД, *Р*-ALGOL, *Р*-FORTRAN, полученные путем расширения соответствующих языков операторами системных взаимодействий, применялись для записи параллельных алгоритмов (FORTRAN — FORmula TRANslation language — первый процедурно-ориентированный язык программирования высокого уровня, предназначенный для описания

алгоритмов решения вычислительных задач). Трансляторы с ВС-языков позволяли при автоматизации параллельного программирования использовать трансляторы для мини-ЭВМ.

3. Средства отладки *Р*-программ служили для анализа качества программ и выявления ошибок при взаимодействии *Р*-ветвей (путем моделирования параллельного процесса на одной ЭМ).

4. Средства управления заданиями (директивами) позволяли пользователю запускать и снимать *Р*-программы, создавать и уничтожать *Р*-файлы, задавать график работы подсистем, переводить машины в режим тестирования и т. д.

5. Средства специальной организации параллельных программ включали:

а) сегментатор, дававший экономию оперативной памяти при хранении программы (он производил собственно сегментирование *Р*-ветви и распределение сегментов по памятям машин и организовывал последовательность реализации сегментов; следовательно, он применялся при обобщенной матричной обработке);

б) блок отказоустойчивых вычислений, обеспечивавший продолжение счета при сбоях ЭМ и выходе их из строя.

6. Библиотека стандартных параллельных программ упрощала процесс параллельного программирования.

Пакеты прикладных Р-программ (ППП) были ориентированы на определенные области применения мини-ВС. Среди пакетов имелись:

1) пакет *Р*-программ общего назначения для решения научных, экономических и технических задач;

2) пакет программ для расширения управляющей системы функциями, реализуемыми в автоматизированных системах управления технологическими процессами (АСУТП);

3) пакет *Р*-программ, ориентированный на применение в АСУТП (например, расчет сложных поверхностей, определение технологии обработки деталей и т. д.).

Комплекс программ технического обслуживания мини-ВС МИНИМАКС включал наладочные, контрольные и диагностические тест-программы.

7.4.5. Области применения мини-ВС МИНИМАКС

Ориентация агрегатных средств АСВТ-М и системы малых машин СМ ЭВМ определила области применения мини-ВС МИНИМАКС. Поскольку системы МИНИМАКС имели программируемую структуру и могли состоять из произвольного количества машин, им были доступны задачи со значительным объемом вычислений.

Вычислительные системы МИНИМАКС могли применяться:

- в химической, нефтехимической, нефтеперерабатывающей, металлургической, металлообрабатывающей, приборостроительной, радиотехнической, электронной и других отраслях промышленности;
- на тепловых и атомных электростанциях и в системах энергоснабжения;
- в научно-исследовательских учреждениях, занимающихся исследованием проблем физики, гидроаэродинамики, химии, биологии, медицины и др.

Системы МИНИМАКС использовались:

- для решения научных, экономических и технических задач;
- для сбора и первичной переработки информации в сложных иерархических системах;
- для управления научными экспериментами;
- для исследования технологических процессов и расчета технико-экономических показателей;
- для управления технологическими процессами;
- для контроля качества промышленной продукции (полупроводниковых приборов, электронных схем и др.);
- в качестве центра коммутации сообщений.

Мини-ВС МИНИМАКС могли функционировать автономно, в качестве вспомогательных подсистем мощных сосредоточенных ВС, в составе распределенных ВС или сетей.

7.5. Вычислительная система СУММА

В 1970-х годах для управления процессами в реальном времени широко применялись не только мини-машины, но и вычислительные сети и системы из мини-ЭВМ. В данном параграфе описывается вторая отечественная мини-ВС: Система Управляющая Мини–МАшинная (СУММА). Эта мини-ВС, как и система МИНИМАКС, имела программируемую структуру и свои архитектурные особенности.

Система СУММА была разработана Институтом математики СО АН СССР (Отделом вычислительных систем) совместно с Производственным объединением «Кварц» Министерства электронной промышленности СССР (г. Калининград). Техническое проектирование мини-ВС было выполнено в 1975 г., опытно-промышленный образец был изготовлен и отработан в 1976 г.

Архитектура системы СУММА:

- MIMD-архитектура;
- распределенность средств управления, обработки и памяти;

- параллелизм, однородность, модульность;
- программируемость структуры;
- масштабируемость;
- живучесть;
- единый канал для управляющей и рабочей информации;
- аппаратурно-программная реализация системных взаимодействий.

7.5.1. Функциональная структура мини-ВС СУММА

Мини-ВС СУММА формировалась из ЭМ-трехполюсников, количество которых не было фиксировано. Система характеризовалась большой архитектурной гибкостью. Ее можно было легко расширить или сократить в соответствии с предъявляемыми требованиями.

Принципиальные ограничения на структуру мини-ВС (количество ЭМ и порядок их соединения) не накладывались, однако при любой структуре каждая ЭМ могла взаимодействовать не более чем с тремя соседними машинами с помощью полудуплексных каналов. В мини-ВС была заложена возможность «программировать» адресацию ЭМ, в частности, система могла быть настроена на относительную адресацию ЭМ.

Системы управления, на применение в которых была рассчитана мини-ВС СУММА, характеризуются стабильностью решаемых задач, нежесткими требованиями к реактивности на изменение операционной обстановки (преимущественно детерминированный поток запросов на обслуживание). Следовательно, в системах управления перепрограммирование структуры мини-ВС требовалось выполнять редко, и время обмена управляющей информацией в общем времени работы машин системы составляло незначительную часть. Эти факторы позволили ограничиться единым каналом для обмена управляющей (настроечной) информацией и данными между ЭМ мини-ВС.

Единый канал обмена управляющей и рабочей информацией между машинами системы СУММА вместе с программной реализацией некоторых функций позволили по сравнению с системой МИНИМАКС существенно упростить СУ (например, программными средствами в системе СУММА реализовывалась выработка обобщенного признака Ω).

Из-за использования для всех взаимодействий одних и тех же связей перепрограммирование структуры мини-ВС можно было осуществлять только в границах сформированных подсистем. Снятие границ образованной подсистемы производилось только «изнутри».

К системам управления предъявляются повышенные требования по живучести, следовательно, их вычислительные средства должны обладать структурной живучестью. Для формирования мини-ВС СУММА использо-

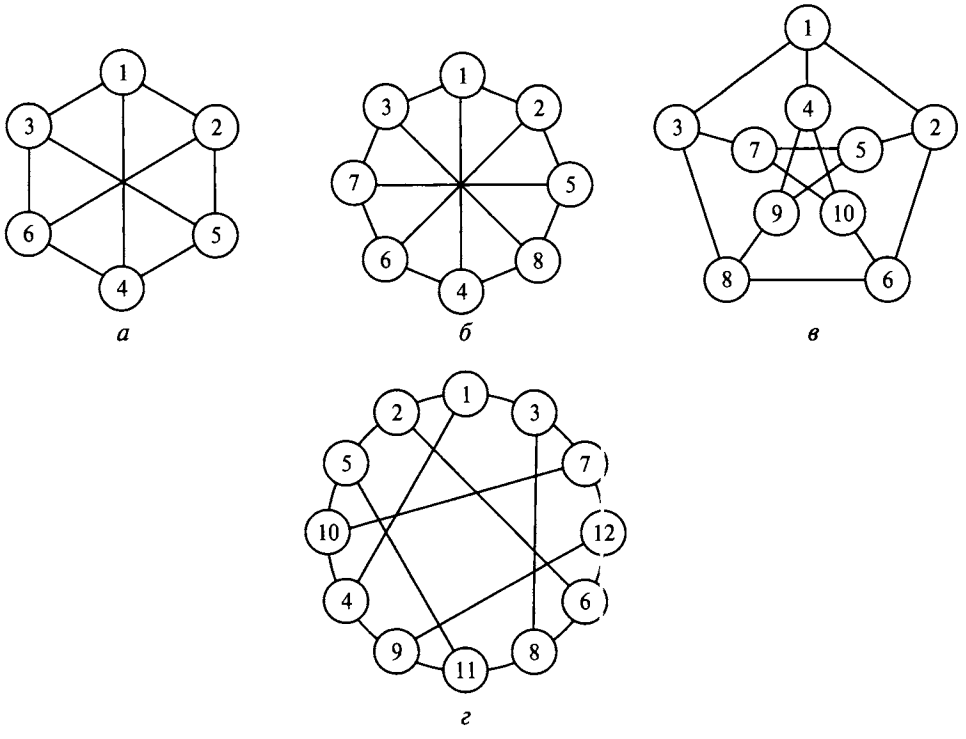


Рис. 7.15. Оптимальные структуры мини-ВС СУММА:
а — $L(6, 3, 4)$; *б* — $L(8, 3, 4)$; *в* — $L(10, 3, 5)$; *г* — $L(12, 3, 5)$

вались оптимальные $L(N, 3, g)$-графы (т. е. обеспечивающие максимум координат вектор-функции структурной живучести, см. разд. 7.2.1). На рис. 7.15 представлены примеры оптимальных структур мини-ВС СУММА для количества машин 6, 8, 10 и 12.

7.5.2. Элементарная машина мини-ВС СУММА

Элементарная машина системы СУММА формировалась как «трехполюсник», или, точнее, композиция из ВМ и СУ, рассчитанного на три межмашинные связи (рис. 7.16).

Вычислительный модуль предназначался для выполнения всех операций, связанных с переработкой информации, в частности для инициирования реализации системных операций. *Системное устройство* использовалось для реализации системных взаимодействий машин, в частности для программирования структуры мини-ВС.

В качестве ВМ использовали произвольные конфигурации мини-ЭВМ «Электроника-100 И». Следует заметить, что архитектура системы СУММА была ориентирована также на применение мини-ЭВМ PDP-8 фирмы Digital Equipment Corp.

Минимальная конфигурация ЭВМ «Электроника-100 И» включала процессор, ферритовую оперативную память и средства ввода-вывода информации. Состав периферийного оборудования, а также тип (магнитные ленты, диски) и объем внешней памяти определялись конкретным применением мини-машины.

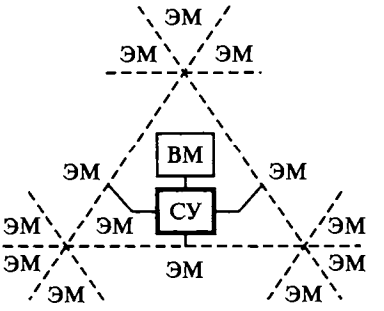


Рис. 7.16. Функциональная структура ЭМ мини-ВС СУММА:

ЭМ — элементарная машина; ВМ — вычислительный модуль; СУ — системное устройство

Основные технические характеристики ЭВМ:

- структура команд машины — одноадресная;
- система счисления — двоичная;
- длина слов — 12 двоичных разрядов;
- способ представления чисел — с фиксированной запятой (режим работы с плавающей запятой реализовывался программно);
- цикл оперативной памяти — 1,5 мкс;
- емкость оперативной памяти — 4...32 К слов;
- быстродействие при выполнении операций типа «сложение» — 300 тыс. опер./с;
- время выполнения команд:
  сложения (регистр-память) — 3 мкс,
  умножения с фиксированной запятой — 10 мкс.

Система команд включала команды обращения к памяти, микрокоманды, команды расширенной арифметики, команды обращения к внешним устройствам. Было предусмотрено прерывание программ. Аппаратура обеспечивала глубину прерывания, равную единице, однако в мини-ЭВМ была заложена возможность организации многоуровневого прерывания программным способом.

Обмен информацией с внешними устройствами осуществлялся через программный канал или через канал разрыва данными. Конкретное внешнее устройство, к которому проводилось обращение из программ, определялось селекторным кодом. Канал разрыва данными являлся вырожденным каналом прямого доступа к памяти и позволял считывать или заносить массивы информации в оперативную память машины. На время обмена с внешним устройством в режиме разрыва данными программная работа машины при-

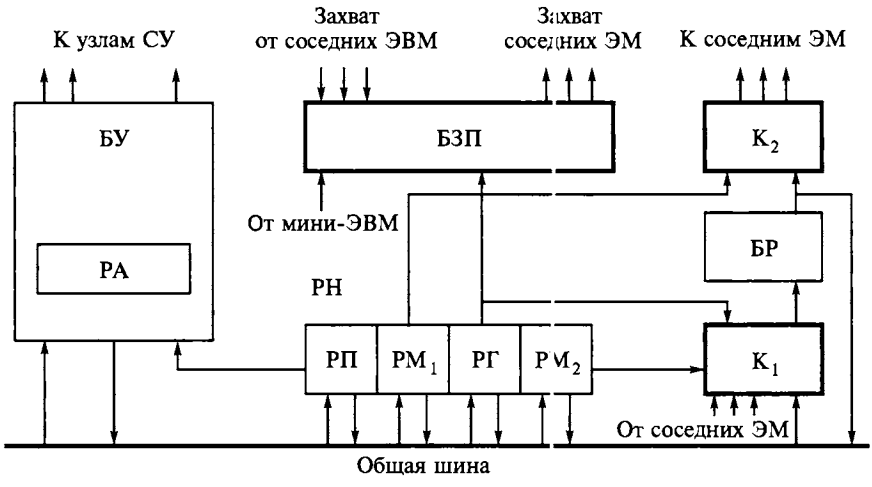


Рис. 7.17. Функциональная структура системного устройства мини-ВС СУММА:

БУ — блок управления; $К_1$, $К_2$ — коммутатор; РН — регистр настройки; БЗП — блок захвата и приоритетов; СУ — системное устройство; ЭМ — элементарная машина; БР — буферный регистр; РП — регистр признаков; $РМ_1$, $РМ_2$ — регистры маски; РГ — регистр границ; РА — регистр адреса

останавливалась, содержимое рабочих регистров сохранялось аппаратно; после завершения обмена продолжалось выполнение программы. Максимальная скорость обмена достигала $6 \cdot 10^6$ бит/с.

Системное устройство конструктивно было оформлено в виде отдельного модуля. К мини-ЭВМ оно подключалось через общую шину (как и внешнее устройство), а к СУ трех соседних ЭМ — через каналы межмашинной связи. Данное устройство было не сложнее СУ мини-ВС МИНИМАКС.

Рассмотрим подробнее функциональную структуру СУ мини-ВС СУММА.

Системные устройства мини-ВС СУММА обеспечивали передачу информации между машинами по способу коммутации сообщений. В состав каждого СУ входили: блок управления (БУ), входной и выходной коммутаторы ($К_1$ и $К_2$), буферный регистр (БР), регистр настройки (РН), блок захвата и приоритетов (БЗП) (рис. 7.17).

Блок управления координировал работу всех остальных схем СУ при реализации его взаимодействий с собственной мини-ЭВМ и соседними ЭМ. В составе БУ имелся 12-разрядный регистр адреса (РА). Этот регистр использовался для указания ячейки оперативной памяти мини-ЭВМ, к которой осуществлялось очередное обращение СУ. Содержимое РА можно было устанавливать либо из данной мини-ЭВМ, либо из соседней ЭМ. Содержимое РА наращивалось автоматически на 1 после каждого обращения к памяти.

Коммутаторы $К_1$ и $К_2$ системных устройств предназначались для управления потоками информации при межмашинных взаимодействиях в мини-ВС. Входной коммутатор $К_1$ определял направления приема информации из межмашинных каналов, а выходной коммутатор $К_2$ — направления передачи в межмашинные каналы. Связь между коммутаторами $К_1$ и $К_2$ осуществлялась через буферный регистр. Композиция из $К_1$, БР и $К_2$ обеспечивала не только обмен информацией между данной ЭМ и межмашинными каналами, но и необходимые транзитные пересылки информации при взаимодействиях между другими ЭМ мини-ВС. Состояния коммутаторов $К_1$ и $К_2$ задавались при помощи специальных разрядов регистра настройки.

Регистр настройки — представлял собой композицию из четырех регистров: РП, $РМ_1$, $РМ_2$, РГ. Все регистры были программно доступны для мини-ЭВМ. Регистр признаков РП был 13-разрядным, он предназначался для хранения признаков $\{\pi_0, \pi_1, \ldots, \pi_i, \ldots, \pi_{12}\}$. Признаки позволяли задать функции ЭМ при выполнении межмашинных взаимодействий. В частности, признак π_0 использовался для отметки ЭМ при ее включении в систему: $\pi_0 = 1$, если ЭМ входила в состав мини-ВС. Семантика признаков $\{\pi_1, \pi_2, \ldots, \pi_{12}\}$ заранее не была определена, значение этих признаков программист мог задать сам с учетом особенностей конкретной задачи. Например, с помощью признака π_i можно было выделить машины, участвующие в выработке обобщенного условия перехода, а с помощью π_j — машины, реагирующие на значение этого условия, $j \neq i$, $i, j \in \{1, 2, \ldots, 12\}$. В других применениях мини-ВС содержимое РП могло быть использовано в качестве адреса ЭМ.

Регистр маски $РМ_1$ состоял из трех триггеров T_k^- ($k \in \{1, 2, 3\}$), каждый из которых предназначался для маскирования одного из направлений межмашинных связей. Состояние триггера T_k^- определяло возможность выдачи информации из данной ЭМ на k-й выход межмашинных связей ($k \in \{1, 2, 3\}$). Регистр маски $РМ_2$ включал в себя также три триггера: T_k^+ ($k \in \{1, 2, 3\}$), а его состояние определяло направления приема данных в ЭМ из межмашинных каналов.

Регистр границ РГ — 3-разрядный – использовался при формировании «границ» между машинами, т. е. при разбиении мини-ВС на подсистемы. Содержимое этого регистра определяло направления межмашинных связей ЭМ, из которых запрещался прием любой информации (в том числе и управляющей). Точнее, установка в единичное состояние тригера «границы», сопоставленного с k-м направлением межмашинных связей, приводила к функциональному разрыву всех связей системного устройства ЭМ в k-м направлении.

При функционировании мини-ВС СУММА любое обращение к любому СУ начиналось с выдачи в него сигнала захвата. Доступ к СУ получало то средство, заявка которого на обслуживание принималась *блоком захвата и приоритетов*. Однако в процессе работы мини-ВС в каждой ЭМ одновременно в состоянии разрешения приема/передачи информации могло находиться несколько триггеров масок. Конфликты при одновременном обращении к СУ с нескольких направлений разрешались также БЗП. Связям СУ с собственной мини-ЭВМ отводился наивысший приоритет.

7.5.3. Системные команды мини-ВС СУММА

Системные команды мини-ВС СУММА были разделены на три группы. Рассмотрим подробно функциональные особенности реализации команд этих групп.

Первую группу составляли команды обращения из мини-ЭВМ в собственное СУ. Эти команды являлись командами обращения к внешним устройствам (но с селекторным кодом, присвоенным СУ). Команды позволяли:

- установить (или сбросить) заявку на обслуживание процессора;
- сбросить все заявки на обслуживавание, кроме заявки собственного процессора;
- осуществить обмен с регистрами СУ;
- сбросить триггер признака готовности СУ;
- пропустить очередную команду по значению триггера готовности СУ.

Команды второй и третьей групп выполнялись совместно и позволяли осуществить обмен информацией между любыми ЭМ подсистемы. Процесс передачи, инициированный передающей ЭМ, начинался с «захвата» собственного СУ. Применялось два режима захвата: мягкий и жесткий. При мягком режиме процессор устанавливал в СУ заявку на обслуживание и ждал освобождения СУ от текущей работы. При жестком режиме захват СУ происходил независимо от текущего состояния СУ. Передача процессором необходимой информации в СУ осуществлялась лишь после подтверждения, что захват СУ произошел.

Обмен между ЭМ выполнялся 13-разрядными словами, все разряды слова передавались параллельно. Тринадцатый разряд приформировывался СУ к каждому слову, полученному от процессора, и указывал назначение слова. Единичное значение этого разряда соответствовало передаче кода настройки, а нулевое — передаче операнда или адреса: последний посылался в регистр адреса СУ. Любой передаваемый между ЭМ массив информации начинался со слова настройки. Передача осуществлялась по всем на-

Рис. 7.18. Формат слова настройки мини-ВС СУММА

правлениям, определенным состоянием регистра выходной маски. Слово настройки предназначалось для задания функционирования СУ всех машин, принимавших данный массив, и состояло из двух частей (рис. 7.18).

1. *Указатели*:

а) запроса прерывания (1 соответствовала переходу к подпрограмме обработки прерываний после приема всего массива информации; 0 запрещал переход);

б) второго слова массива передаваемой информации (1 означала, что второе слово являлось начальным адресом следующего за ним массива; следовательно, оно заносилось в регистр адреса системного устройства; 0 соответствовал операнду, который заносился в память ЭВМ по адресу, определяемому текущим содержимым регистра адреса СУ);

в) приема кодов (1 настраивала ЭМ на прием операндов или адреса для регистра адреса СУ; 0 запрещал прием);

г) ретрансляции настройки (1 разрешала ретрансляцию слова настройки в выходные каналы межмашинной связи; 0 не разрешал ретрансляцию);

д) конца обмена (1 означала: завершить обмен информацией и освободить системные устройства принимающих ЭМ; 0 — обмен продолжить);

е) типа обмена (1 соответствовала быстрому обмену, при котором принимающие ЭМ были либо заняты приемом кодов, либо находились в состоянии ожидания; 0 означал медленный обмен, при котором принимающие ЭМ после завершения приема в память очередного кода, возвращались к своей программной работе).

2. *Идентификатор* ЭМ, участвующих в системном взаимодействии, определяемом указателями. Элементарные машины, отмеченные признаком $\pi_i = 1,\ i \in \{1, \ldots, 12\}$, осуществляли настройку своих СУ в соответствии со значениями указателей; машины, в которых $\pi_i = 0$, сохраняли состояния СУ.

Идентификация ЭМ в системе СУММА осуществлялась с помощью программно-задаваемых идентификаторов, а связность между ЭМ области,

выделенной признаками разделения, устанавливалась лишь на время взаимодействия. Имя ЭМ в системе СУММА задавалось содержимым регистра признаков, а ее адрес — содержимым выделенной для этой цели ячейки оперативной памяти. Вид адресации и характер преобразования адресов определялся программой, которую можно было вызывать через систему прерывания. Эта программа являлась частью «путевой процедуры», выполняемой машиной при реализации взаимодействия ЭМ. Идентификация по имени является основным способом выделения взаимодействующих ЭМ системы СУММА.

Из-за отсутствия аппаратурно закрепленных за ЭМ идентификаторов в системе СУММА предусматривалась процедура разметки системы, которая заключалась в присвоении машинам групповых или индивидуальных идентификаторов. Разметка выполнялась как при первоначальном включении электропитания системы, так и в процессе решения ОС задачи планирования и загрузки мини-ВС. Последний случай являлся более общим. Ниже рассмотрен реализованный в системе СУММА вариант разметки. Он был предназначен для организации мультидоступа к системе при решении задач, представленных как последовательными, так и параллельными алгоритмами.

Разметка выполнялась той машиной, которая получила очередную заявку на решение задачи (ведущей ЭМ). *Ведущая ЭМ* создавала компонент связности, включавший в себя все доступные ей машины при данной загруженности системы (т. е. те ЭМ, которые были не отделены от ведущей ЭМ установленными в них признаками границ). В процессе разметки ведущая ЭМ передавала во все выходные каналы слово настройки с указателем ретрансляции, равным 1, и с идентификатором $\pi_0 = 1$. Машины, соседние с ведущей, принимали и одновременно ретранслировали слово настройки в другие ЭМ системы. В результате выполнения описываемого «волнового» алгоритма все исправные ЭМ, доступные ведущей машине, образовывали компонент связности. Компонент связности имел вид дерева, корнем которого была ведущая ЭМ (рис. 7.19).

После завершения формирования компонента связности реализовывался режим сбора информации. Суть этого режима заключалась в передаче сообщения-квитанции от каждой ЭМ компонента связности к той соседней машине, от которой она получала прямое сообщение. Процесс передачи квитанций начинался с вершин дерева, а сообщения адресовались ведущей ЭМ. При этом каждая ЭМ, получавшая сообщение-квитанцию, перед дальнейшей передачей включала в него сведения о своем состоянии и о реализованной в процессе передачи прямого сообщения соединительной функции.

Далее, в ведущей ЭМ на основе собранной в процессе стока информации делалось диспетчерское заключение о возможности обслуживания заявки пользователя на доступных ресурсах мини-ВС. Если ресурсов компонента

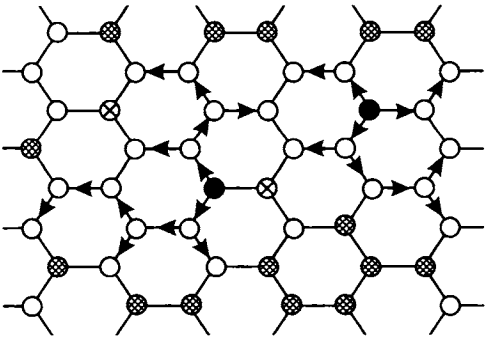


Рис. 7.19. Формирование компонентов связности в мини-ВС СУММА:
● — ведущая ЭМ; ⊗ — неисправная ЭМ; ○ — свободная ЭМ; ◍ — ЭМ, занятая в других подсистемах

связности было недостаточно, то дальнейшее обслуживание заявки определялось выбранной дисциплиной. Она, например, могла быть либо снята с обслуживания, либо поставлена в очередь на ожидание недостающих ресурсов. Если ресурсы компонента связности были достаточны, то происходила собственно разметка машин компонента связности. Процесс разметки осуществлялся ведущей ЭМ и сводился к приписыванию необходимых идентификаторов принимающим ЭМ. Принимающие ЭМ выбирались на основе заданной в задаче пользователя структуры связей между отдельными ветвями и с учетом вложения этой структуры в структуру компонента связности. После разметки проводилась настройка и загрузка размеченных ЭМ. Остальные ЭМ идентификаторов не получали и возвращались в фонд общих ресурсов мини-ВС. Разметка уничтожалась после решения загруженной задачи.

7.5.4. Программное обеспечение мини-ВС СУММА

Программное обеспечение мини-ВС СУММА было ориентировано на управление процессами в реальном масштабе времени (рис. 7.20). Вместе с тем оно содержало компоненты, позволявшие использовать систему СУММА в качестве вычислительного средства общего назначения.

Программное обеспечение мини-ВС СУММА включало в себя супервизор, систему *Р*-программирования, управляющие системы для автоматизированных систем управления технологическими процессами (АСУТП), комплекс программ технического обслуживания.

Супервизор являлся резидентной программой управления процессами в реальном масштабе времени. Он состоял из подсистем управления процессами и межмашинных взаимодействий.

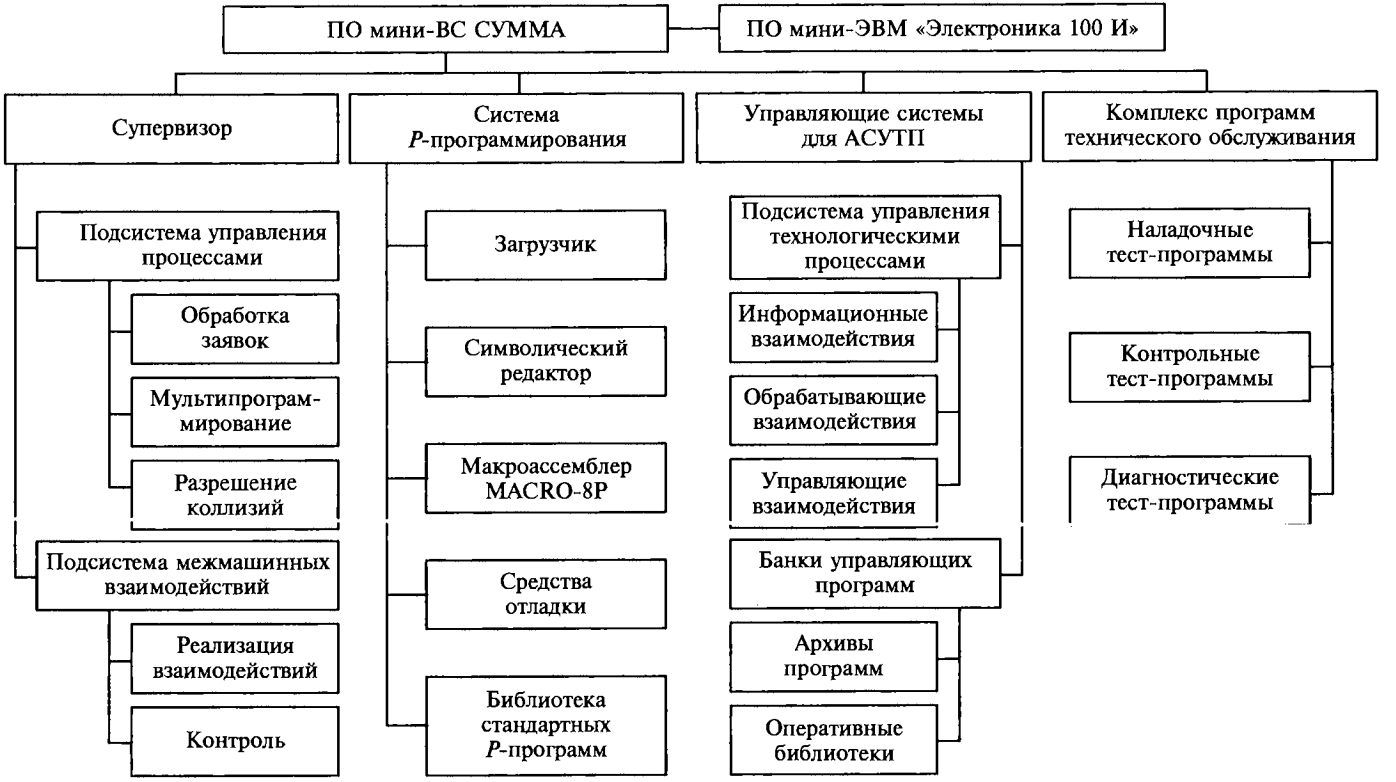


Рис. 7.20. Программное обеспечение мини-ВС СУММА

Первая подсистема осуществляла:

- анализ и обработку заявок от внешних устройств и от объектов управления;
- режим мультипрограммирования в реальном масштабе времени;
- накопление запросов на системные межмашинные взаимодействия;
- разрешение коллизий при системных взаимодействиях.

Вторая подсистема выполняла:

- реализацию системных взаимодействий;
- контроль правильности выполнения системных взаимодействий в мини-ВС.

Супервизор не зависел от конкретного применения системы СУММА.

Система Р-программирования мини-ВС СУММА включала в себя комплекс модифицированных средств программирования мини-ЭВМ «Электроника-100 И»:

- загрузчик для ввода объектных программ в оперативную память мини-ВС:
- редактор для приготовления (с клавиатуры пишущей машинки) символических *Р*-программ (в частности, *Р*-программ на системном макроассемблере MACRO-8P);
- транслятор MACRO-8P для трансляции исходной *Р*-программы в объектную, готовую для реализации на мини-ВС;
- средства отладки объектных программ;
- библиотеку стандартных параллельных программ, включавшую программы для реализации сложных системных взаимодействий (например, обобщенных условного и безусловного переходов, конвейерной обработки и др.) и *Р*-программы для научно-технических расчетов.

Управляющие системы для АСУТП определяли конкретные применения мини-ВС СУММА. Каждая такая система подразделялась на подсистему управления технологическим процессом и банк управляющих программ.

Подсистема управления технологическим процессом обеспечивала следующие виды взаимодействий:

- информационное (реализация всех необходимых видов и способов ввода и вывода информации);
- обрабатывающее (выполнение требуемых видов переработки информации: интерполирования, расчета геометрии детали и т. п.);
- управляющее (реализация организационных, технологических и обслуживающих функций по управлению процессом).

Банк управляющих программ включал:

а) архив программ управления технологическим процессом (на магнитных лентах);

б) оперативную библиотеку программ управления технологическим процессом (на магнитных дисках).

Комплекс программ технического обслуживания мини-ВС СУММА обеспечивал выполнение работ по наладке, контролю и диагностике технических средств. В комплекс входили наладочные, контрольные и диагностические программы.

7.5.5. Области применения мини-ВС СУММА

Система СУММА в 1970-х годах была перспективным вычислительным средством для АСУТП. Для АСУТП, построенных на ее основе, были характерны:

- простота компоновки и настройки на заданный парк оборудования и объектов управления;
- модульная и адекватная наращиваемость вычислительной мощности при развитии производства;
- высокая надежность и живучесть;
- высокая технико-экономическая эффективность;
- длительный срок эффективной эксплуатации (медленное моральное старение).

Применение мини-ВС СУММА было эффективно и при решении широкого класса задач, представленных параллельными программами. Кроме того, она могла быть использована в качестве вычислительного ядра «интегрированных» АСУТП. В таких АСУТП реализовывались функции не только собственно управления, но и планирования производства, и «проектирования» процесса (например, расчета технологии обработки или расчета поверхностей деталей, если система предназначалась для работ со станками с числовым программным управлением).

Систему СУММА можно было использовать и как автономное средство для решения задач повышенной сложности, а также для моделирования архитектур ВС и параллельных вычислительных технологий.

Функциональная организация СУ позволяла просто адаптировать систему СУММА к конкретным областям ее применения.

7.6. Вычислительные системы семейства МИКРОС

Прогресс в ВТ неразрывно связан с достижениями в области элементной базы и в интегральной технологии. В конце 1970-х годов мини-процессоры вытесняются микропроцессорами, на смену мини-ЭВМ пришли микроЭВМ; создаются параллельные ВС как коллективы микропроцессоров (см., например,

систему Cm*; § 6.6). Параллельные средства, построенные из микропроцессоров, первоначально назывались микроВС. Использовать этот термин сейчас нет необходимости: все современные ЭВМ и ВС — микропроцессорные.

В начале 1980-х годов в качестве базы для построения распределенных ВС с программируемой структурой стали использовать аппаратурно-программные средства микроЭВМ. В Отделе вычислительных систем Сибирского отделения АН СССР проводились работы по научно-исследовательскому проекту МИКРОС*, целью которых было создание МИКРОпроцессорных Систем с программируемой структурой (МИКРОС). Результатом работ явилось семейство МИКРОС, включающее модели МИКРОС-1 (1986); МИКРОС-2 (1992); МИКРОС-Т (1996). Разработка моделей семейства МИКРОС осуществлялась Отделом вычислительных систем СО АН СССР (СО РАН) в содружестве с подразделениями Научно-производственного объединения «Алмаз» и Научно-исследовательского института «Квант» Министерства радиопромышленности СССР (г. Москва).

Архитектура систем семейства МИКРОС:

- MIMD-архитектура;
- распределенность средств управления, обработки и памяти;
- массовый параллелизм (при обработке данных и управлении процессами);
- программируемость структуры сети межмашинных связей;
- возможность программной трансформации MIMD-архитектуры в SIMD и MISD;
- децентрализация ресурсов;
- асинхронность и близкодействие;
- масштабируемость, модульность и однородность.

7.6.1. Функциональная структура вычислительной системы МИКРОС

Возможности функциональных структур систем семейства МИКРОС определяются количеством ЭМ, входящих в их состав, конфигурациями ЭМ и топологией сетей межмашинных связей. Количество ЭМ в любой из моделей (МИКРОС-1, МИКРОС-2, МИКРОС-Т) не фиксировано. Каждая ЭМ — это многополюсник, число полюсов v в первых моделях систем составляло от 2 до 8, а в модели МИКРОС-Т $v = 4$.

Каждая генерация ВС семейства МИКРОС адекватно учитывала текущие возможности ВТ и интегральной технологии. Для формирования конфигураций ЭМ моделей МИКРОС-1 и МИКРОС-2 использовались средства микроЭВМ отечественного семейства «Электроника». Элементарная

* *Хорошевский В.Г* Вычислительная система МИКРОС// Препринт. Новосибирск: ИМ СО АН СССР, 1981. № 38 (ОВС-19).

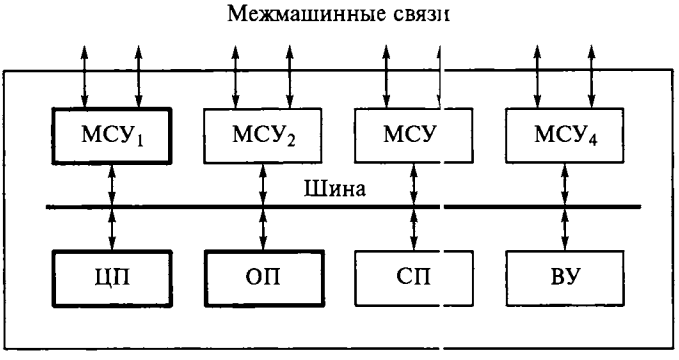


Рис. 7.21. Функциональная структура ЭМ систем МИКРОС-1 и МИКРОС-2:
МСУ — модуль системного устройства; ЦП — центральный процессор; ОП — оперативная память; СП — специальный процессор; ВУ — внешнее устройство

машина представлялась композицией из микроЭВМ (вычислительного модуля) и системного устройства (которое, в свою очередь, формировалось из модулей) (рис. 7.21).

Свойством масштабируемости обладали не только модели семейства МИКРОС, но и их ЭМ. Простейшая конфигурация ЭМ состоит из модуля системного устройства (МСУ), центрального процессора (ЦП) и оперативной памяти (ОП). Модуль СУ обеспечивал реализацию системных операций в ВС и непосредственную связь данной ЭМ с двумя соседними машинами через полудуплексные каналы. Модуль СУ позволял использовать в качестве каналов различные средства, в частности, экранированные провода (при расстоянии между ЭМ до 30 м), либо радиочастотные кабели (если расстояние между ЭМ не превышало 300 м), либо коммутируемые или выделенные телефонные каналы связи (с использованием аппаратуры передачи данных независимо от расстояния между ЭМ). Заложенная в модуль СУ схема обеспечения связности машин была равно пригодна для формирования как сосредоточенных, так и пространственно распределенных ВС.

В моделях ВС МИКРОС-1 и МИКРОС-2 в качестве базовых машин были использованы микроЭВМ «Электроника 60М» и «Электроника 60-1» соответственно. Технические характеристики микроЭВМ, точнее их ЦП, отражены в табл. 7.9.

Расширенные конфигурации ЭМ (см. рис. 7.21) систем МИКРОС-1 и МИКРОС-2 могли иметь до четырех модулей СУ, специальный процессор (СП), дополнительные модули оперативной памяти, набор внешних устройств (ВУ). *Специальные процессоры* «Электроника МТ-70» или «Электроника 1603» расширяли вычислительные возможности ЦП при решении научно-технических задач, связанных с обработкой значительных массивов данных и с выполнением больших объемов однородных вычислений (табл. 7.10).

Таблица 7.9

Техническая характеристика	МикроЭВМ	
	«Электроника 60М»	«Электроника 60-1»
Разрядность слов, дв. разр.	16	16
Разрядность чисел с плавающей запятой, дв. разр.	32	32
Объем адресного пространства, К слов	32	128
Максимальная емкость ОЗУ, К слов	28	124
Число команд	81	130
Быстродействие, 10^3 опер./с	250	500
Число уровней прерывания	2	4

Таблица 7.10

Техническая характеристика	Спецпроцессор	
	«Электроника МТ-70»	«Электроника МС 1603»
Разрядность чисел с фиксированной запятой, дв. разр.	16	16
Емкость памяти данных, К слов	32	32–256
Емкость памяти микропрограмм, бит	512 × 56	512 × 32
Число операций над массивами данных	32	32
Время выполнения операций сложения, нс	200	200
Время выполнения операций умножения, нс	400	200
Время выполнения быстрого преобразования Фурье (1024 комплексные точки), мс	30,1	Менее 11

Функциональная организация спецпроцессора семейства «Электроника» основана на микропрограммировании. Спецпроцессор был способен реализовать 32 стандартных операции над массивами (векторами) данных. Алгоритмы этих 32-х операций интерпретировались 12-ю микропрограммами, заложенными в память микропрограмм. Спецпроцессор «Электроника МТ-70» был ориентирован на выполнение большого числа итеративных операций, широко используемых в задачах статистического анализа, численного моделирования, цифровой обработки сигналов и матричной арифметики.

Процессор достигал максимального быстродействия 7,5 млн опер./с при регистровом способе адресации и при выполнении последовательных операций сложения и умножения 16-разрядных чисел с фиксированной запятой.

Набор стандартных операций спецпроцессора «Электроника МТ-70»

1. Логические операции над массивами из 10 элементов:
сложение двух массивов,
сложение массива с константой,

исключающее ИЛИ для двух массивов,
исключающее ИЛИ для массива и константы,
умножение двух массивов,
умножение массива на константу.
2. Арифметические операции:
сложение двух массивов из 20 элементов,
сложение массива из 20 элементов с константой,
вычитание массива из 20 элементов из массива из 20 элементов,
вычитание константы из 20-элементного массива,
умножение двух массивов из 38 элементов,
умножение массива из 38 элеменов на константу,
умножение с двойной точностью двух массивов из 19 элементов,
умножение с двойной точностью массива из 19 элементов на константу.
3. Комплексное умножение:
двух массивов из 20 элементов,
массива из 20 элеменов на сопряженный (два элемента представляют одно комплексное число).
4. Вычисление спектральных амплитуд массива из 20 элементов.
5. Операции:
свертки оператора из четырех элементов с операндом из 12 элементов,
корреляции оператора из четырех элементов с операндом из 12 элементов.
6. Операции, связанные с быстрым преобразованием Фурье (БПФ):
прямое БПФ для массива из 64-х элементов,
обратное БПФ для массива из 64-х элементов,
упаковка двух массивов из 18 элементов для прямого БПФ,
упаковка двух массивов из 18 элементов для обратного БПФ,
распаковка массива из 34-х элементов для прямого БПФ,
распаковка массива из 34-х элементов для обратного БПФ.
7. Сканирование массива из 38 элементов:
по максимуму,
по минимуму.
8. Операции преобразования массивов:
сдвиг массива из 18 элементов вправо на три разряда,
сдвиг массива из 18 элементов влево на три разряда,
двоичная инверсия массива из 64-х элементов,
перемещение двух массивов из 16 элементов,
очистка массива из 20 элементов.

Спецпроцессор «Электроника МС 1603» обладал улучшенными характеристиками по сравнению с процессором «Электроника МТ-70».

Характерные особенности СП «Электроника МС 1603»:

• повышенная производительность (почти втрое выше, чем у процессора «Электроника МТ-70»);

• увеличенный до 256 К слов объем памяти данных при цикле обращения к памяти 200 нс (вместо 600 нс при чтении и 800 нс при записи);

• оперативная (перезагружаемая) память команд емкостью 512 32-разрядных слов;

• встроенные диагностические средства;

• отдельный канал памяти и устройств пользователя.

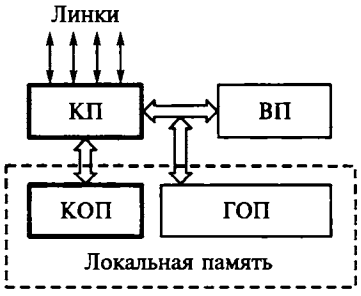


Рис. 7.22. Функциональная структура ЭМ МИКРОС-Т:

КП — коммуникационный процессор; ВП — вычислительный микропроцессор; КОП — коммуникационная память; ГОП — главная оперативная память

Модули системного устройства для системы МИКРОС-2 обладали бóльшими функциональными возможностями, чем в системе МИКРОС-1. Их аппаратура, в частности, позволяла осуществлять: обработку входных/выходных запросов для межмашинных связей (линков); анализ семафоров; формирование пакетов выходных сообщений; управление входными и выходными портами при выполнении системных команд; мультиадресные передачи информации; совмещение межмашинных обменов информацией с вычислениями.

Система МИКРОС-Т базируется на транспьютерных технологиях*. Такие технологии позволяют формировать двумерные ВС с массовым параллелизмом. Двумерные структуры ВС формируются путем отождествления полюсов-линков (Link — связь).

Простейшая конфигурация ЭМ представляется транспьютером (например, Inmos T 805) с памятью, развитые конфигурации ЭМ могут включать в себя: высокопроизводительные микропроцессоры — Intel 860 (компания Intel), PowerPC (альянс компаний IBM, Apple и Motorola), Alpha (компании DEC и Compaq) и др. Для формирования ЭМ системы МИКРОС-Т могут быть использованы стандартные решения зарубежных и отечественных фирм-производителей транспьютерных модулей.

На рис. 7.22 представлена одна из возможных функциональных структур ЭМ системы МИКРОС-Т, включающая в себя коммуникационный процессор (КП, например транспьютер Inmos T805), высокопроизводительный вычислительный микропроцессор (ВП), коммуникационную память (КОП)

* Транспьютер (Transputer) — это элементарная машина-четырехполюсник в интегральном исполнении.

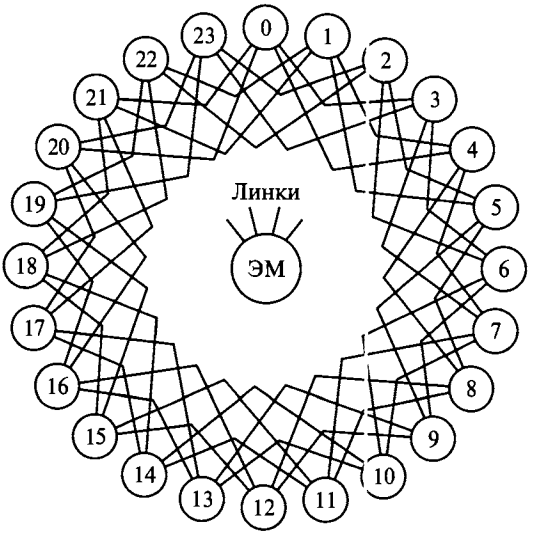


Рис. 7.23. Оптимальная структура 24-машинной ВС семейства МИКРОС в виде D_2-графа {24; 3, 4}:

ЭМ — элементарная машина

и главную оперативную память (ГОП). Производительность такой ЭМ составляет 10^2 MFLOPS...10 GFLOPS, а емкость памяти — 10^8...10^{10} байт.

Путем отождествления полюсов (или линков) ЭМ достигается связность последних. Достаточные масштабируемость размерности и живучесть структур ВС достигаются при числе связей (линков) $v = 2–8$. Рекомендуемые виды структур сетей межмашинных связей ВС семейства МИКРОС — оптимальные D_n- и $L(N, v, g)$-графы (см. разд. 7.2.1), могут быть использованы произвольные (нерегулярные) графы. На рис. 7.23 показана оптимальная структура 24-машинной ВС семейства МИКРОС, являющаяся D_2-графом {24; 3, 4}. Эта структура обладает минимальным средним диаметром — 2, 21 среди D_2-графов, имеющих 24 вершины. Следовательно, в системе с такой структурой обеспечивается минимум задержек при передаче информации между ЭМ.

Таким образом, в ВС семейства МИКРОС заложена возможность статической реконфигурации структуры путем:

1) изменения вида графа, представляющего сеть связи между ЭМ (от простейших однородных до произвольных);

2) варьирования числа вершин в структурном графе (или числа машин) системы;

3) изменения степени вершины в структурном графе (или числа соседних машин для каждой) от 2 до 8;

4) подбора состава ЭМ в широком диапазоне;

5) формирования как сосредоточенных, так и пространственно распределенных звеньев системы.

Отмеченные возможности позволяют на стадии компоновки адаптировать системы семейства МИКРОС к областям применения, классам и сложности решаемых задач.

7.6.2. Программное обеспечение МИКРОС

Эффективная работа ВС и ее пользователей немыслима без ОС и средств параллельного программирования. Любая система семейства МИКРОС, как и ее ПО, открыты к совершенствованию. Ряд моделей семейства ВС — МИКРОС-1, МИКРОС-2 и МИКРОС-Т — породил и соответствующий ряд генераций ПО. Рассмотрим ПО ВС МИКРОС-Т.

Период разработки ПО ВС семейства МИКРОС пересекается с промежутком времени, в течение которого осуществлялось создание программных средств для западных транспьютерных ВС. Широкое распространение получили ОС Helios и средства параллельного программирования 3L. Одним из недостатков средств Helios и 3L является необходимость предварительного описания конфигурации транспьютерной системы и структуры параллельной программы. При конфигурировании параллельной программы процессам и информационным связям между ними ставятся в однозначное соответствие их физические носители: транспьютеры и линки. Подобный статический подход к конфигурированию исключает возможность оперативной автоматической перенастройки системы в случае отказов ее элементов, следовательно, снижает живучесть ВС.

В основу *операционной системы* МИКРОС положены следующие принципы:

- независимость от структуры ВС и числа машин в ней;
- модульность построения;
- распределенность и децентрализованность модулей по машинам ВС;
- локальность связей между модулями;
- асинхронность взаимодействий модулей;
- развиваемость (изменяемость и пополняемость состава модулей, в частности возможность замены программных модулей на аппаратурные);
- иерархичность построения: стратификация системы на уровни, каждый из которых строится на основе предыдущих и освобождает пользователя от специфических для уровня операций по погружению задачи в систему;

• преемственность с ОС базовых микропроцессорных средств (либо микроЭВМ «Электроника», либо транспьютеров, в зависимости от моделей семейства ВС МИКРОС).

Все созданные генерации ОС (МИКРОС-1, МИКРОС-2, МИКРОС-Т) являются распределенными и децентрализованными. Децентрализованная распределенная ОС МИКРОС способна функционировать в ВС произвольной конфигурации; ОС создает в каждой ЭМ «окружение», позволяющее осуществлять динамическую настройку адаптирующейся параллельной программы на существующую конфигурацию ВС (или подсистемы). Децентрализованные процедуры маршрутизации обеспечивают передачу сообщений между любыми ЭМ системы. Указанные свойства ОС МИКРОС являются основой для поддержки живучести ВС (и, следовательно, для организации отказоустойчивых вычислений). Следует подчеркнуть, что ОС МИКРОС-Т имеет иерархическую структуру (рис. 7.24). Каждый уровень ОС строится на основе предыдущих.

Нижний (нулевой) уровень ОС МИКРОС-Т включает средства инициирования работы ВС, приводящие, в частности, все ЭМ в исходное состояние, и драйвер СУ, используемый для организации обменов данными и командами между соседними ЭМ (т. е. машинами, непосредственно соединенными линком). Следующий уровень *1* (ядро ОС) интерпретирует примитивы ОС, предназначенные для динамического распределения памяти ЭМ, а также для порождения, уничтожения процессов и организации их взаимодействия (синхронизации и обмена данными) в пределах одной машины. В совокупности с драйвером СУ ядро ОС позволяет порождать и уничтожать процессы в соседней машине и выполнять взаимодействия процессов, протекающих в этих машинах.

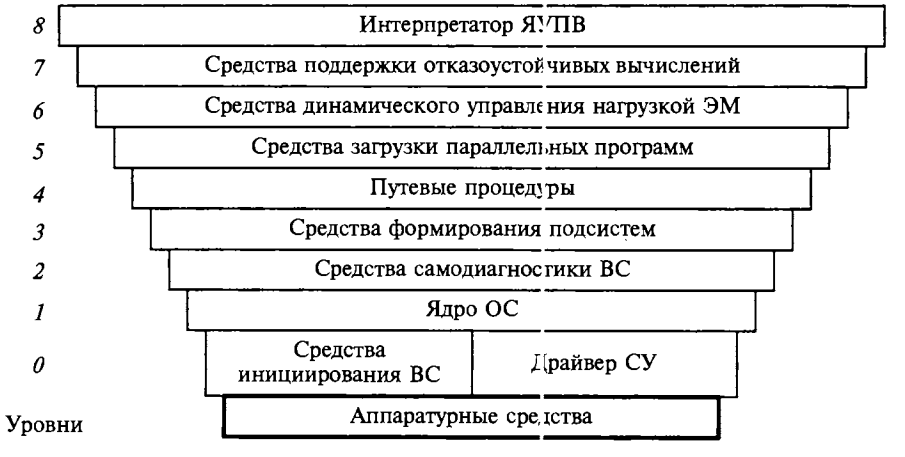


Рис. 7.24. Операционная система МИКРОС-Т

Уровень *2* ОС образован средствами самоконтроля и самодиагностики ВС, которые обеспечивают проверку работоспособности компонентов ВС и локализацию неисправных ресурсов. Самоконтроль и самодиагностика реализуются как параллельные процессы, выполняемые всеми ЭМ системы.

Средства формирования виртуальных подсистем (уровень *3*) используются в мультипрограммных режимах работы системы. Они выделяют машины, входящие в подсистему, посредством созданных в этих машинах «окружений». Окружение содержит информацию о принадлежности машины подсистеме и другие параметры, в частности задающие вид связности машин: «линейку», «кольцо», «дерево», «решетку» и др. Значения элементов окружения используются путевыми процедурами (уровень *4*) при выполнении обменов между машинами подсистемы. Набор путевых процедур предназначается для реализации схем межмашинных обменов в пределах подсистем и всей системы в целом. Фактически путевые процедуры распространяют функции ядра ОС на всю систему.

После формирования окружений в машины подсистемы загружаются предназначенные им программы и данные. Загрузка машин осуществляется с помощью специальных средств уровня *5*.

Уровни *0–5* (за исключением программы инициирования) составляют резидентную часть ОС, содержащуюся в каждой ЭМ вычислительной системы.

Средства динамического управления нагрузкой ЭМ (уровень *6*) осуществляют перераспределение программ и данных между ЭМ подсистемы по завершении ее формирования или реконфигурации. На уровне *7* поддержки отказоустойчивых вычислений выполняются операции, связанные с перезапуском параллельных процессов вычислений с заданных точек возврата. Методы управления нагрузкой и восстановления вычислений могут быть как универсальными, так и специализированными, определяемыми областью применения ВС. Следовательно, уровни *6* и *7* могут быть включены либо в резидентную часть ОС, либо в загрузочный модуль параллельной программы пользователя.

Интерпретатор языка управления параллельными вычислениями (ЯУПВ, уровень *8*) по командам с терминала порождает процессы, осуществляющие: генерацию подсистемы необходимого типа («дерево», «линейка», «кольцо» и т. п.) из требуемого числа работоспособных ЭМ; загрузку параллельной программы в сформированную подсистему и инициирование ее выполнения. Интерпретатор загружается в ЭМ, непосредственно связанную с терминалом.

При разработке *средств программирования* соблюдались следующие положения:

1) независимость от структуры и числа машин ВС;

2) адекватность между языковыми средствами для задания параллелизма и синхронизации вычислений и структурной, и функциональной организацией ВС;

3) развиваемость (пополняемость как языковыми средствами, так и средствами отладки, анализа программ, распараллеливателями последовательных программ, диалоговыми системами обучения параллельному программированию, пакетами параллельных программ);

4) использование языков высокого уровня как для прикладного, так и для системного программирования;

5) преемственность со средствами последовательного программирования, обеспечение постепенного перехода программиста от привычной для него идеологии последовательного программирования к параллельному программированию.

В версии средств программирования МИКРОС-Т имеются языки параллельного программирования *Р*-ФОРТРАН и *Р*-С. Эти языки построены путем расширения соответствующих традиционных языков FORTRAN и С примитивами организации межмашинных взаимодействий и примитивами оценки параметров подсистем, на которых исполняются параллельные программы. Первые позволяют организовать взаимодействия между любыми ветвями программы, вторые дают возможность использовать параметры подсистемы для адаптации программы к текущей конфигурации последней. Это свойство существенно с двух точек зрения: простоты организации параллельных вычислений и отказоустойчивости. Реализация данных примитивов основывается на средствах распределенной децентрализованной операционной системы МИКРОС-Т.

7.6.3. Архитектурные свойства систем семейства МИКРОС

Опишем архитектурные свойства ВС семейства МИКРОС.

Класс архитектуры любой модели ВС — это MIMD; допустима трансформация архитектуры MIMD в архитектуру MISD или SIMD путем программной перенастройки системы.

Класс ВС — система с программируемой структурой и с распределенным управлением.

Характер пространственного размещения вычислительных ресурсов — сосредоточенный или распределенный.

Основная функционально-структурная единица вычислительных ресурсов — ЭМ.

Функции ЭМ — традиционные для ЭВМ функции по переработке информации плюс функции, связанные с управлением ВС в целом как коллектива (ансамбля) машин.

Масштабируемость ВС поддерживается аппаратурными средствами (системным устройством либо транспьютером) и программным обеспечением.

Количество N элементарных машин не фиксировано, что обеспечивает принципиально неограниченное наращивание производительности ВС.

Виды структуры сети межмашинных связей — произвольные (нерегулярные) графы.

Рекомендуемые структуры ВС:

• для сосредоточенных ВС: оптимальные D_n- и $L(N, v, g)$-графы, т. е. графы, в которых, в частности, обеспечивается минимум среднего диаметра (задержек при межмашинных передачах информации):

D_n-графы имеют параметрическое описание $\{N; \omega_0, \omega_1, \ldots, \omega_{n-1}\}$, где N — порядок; n — размерность графа, числа ω_k таковы, что две вершины с номерами i и j соединены ребром, если выполняется сравнение $i - j \equiv \omega_k \bmod N$, $i, j \in \{0, 1, .., N-1\}$, $k \in \{0, 1, \ldots, n-1\}$;

$L(N, v, g)$-графы — неориентированные однородные графы, в которых N — число вершин, v — степень вершины (число межмашинных связей для каждой ЭМ), g — обхват (длина кратчайшего цикла в графе);

• для пространственно распределенных ВС: в условиях отсутствия жестких технико-экономических ограничений те же структуры, что и для сосредоточенных систем, в противном случае любые структуры реально существующих сетей передачи информации или сетей, обеспечивающих при заданных ограничениях связность вычислительных ресурсов.

Наращиваемость (масштабируемость) *размерности* структуры ВС — $v = 1-8$.

Тип оперативной памяти — распределенная и общедоступная.

Аппаратурно-программная база системы:

• для моделей МИКРОС-1 и МИКРОС-2 — средства микромашинной техники и спецпроцессоров семейства «Электроника»;

• для модели МИКРОС-Т — транспьютерные средства семейства Inmos T800 (компании SGS-Thomson) и средства высокопроизводительных микропроцессоров.

Конфигурации ЭМ:

• для моделей МИКРОС-1 или МИКРОС-2 — всевозможные допустимые комплексы на основе микроЭВМ «Электроника 60М» или «Электроника 60-1» (или совместимых с ними других микроЭВМ), которые могут иметь в своем составе, в частности, спецпроцессоры «Электроника МТ-70» и «Электроника 1603»;

• для модели МИКРОС-Т — либо один из транспьютеров T805 или T800 (простейшая конфигурация), либо транспьютер, один из высокопроиз-

водительных микропроцессоров: Intel 860, PowerPC, Alpha и дополнительная оперативная память (расширенная конфигурация).

Коммуникационные средства ЭМ для реализации функций управления системами:

- МИКРОС-1 или МИКРОС-2 — модули СУ (СУ-1 или СУ-2), выполненные на полных платах конструктивов для микроЭВМ «Электроника 60М» или «Электроника 60-1»;
- МИКРОС-Т — транспьютер Inmos T805 (или T800).

Число коммуникационных средств в одной ЭМ в системах:

- МИКРОС-1 или МИКРОС-2 — одно СУ в составе от одного до четырех модулей СУ;
- МИКРОС-Т — один транспьютер Inmos T805 (или T800).

Программное обеспечение ВС:

- МИКРОС-1 или МИКРОС-2:

– распределенные децентрализованные ОС, являющиеся расширением ОС микроЭВМ «Электроника 60М» или «Электроника 60-1»;

– языки параллельного программирования *P*-FORTRAN и *P*-PASCAL, являющиеся языками семейства микроЭВМ «Электроника», дополненными средствами организации системных взаимодействий.

- МИКРОС-Т:

– распределенная децентрализованная ОС МИКРОС-Т (см. рис. 7.24);

– языки параллельного программирования *P*-FORTRAN и *P*-C.

Режимы функционирования ВС:

- монопрограммный, обеспечивающий решение сложной задачи, при котором все ресурсы ВС используются для реализации параллельных программ и обеспечения требуемого уровня надежности и живучести;
- мультипрограммные (обработка наборов и обслуживание потоков параллельных задач, разделение «времени и/или пространства» и др.), при которых для решения любой задачи или для обслуживания любого задания используется лишь часть ресурсов системы.

Способы обработки данных в ВС:

- распределенный (параллельный), когда однородно расчлененные данные и ветви параллельной программы их обработки рассредоточиваются по ЭМ;
- матричный, при котором программа вычислений размещается в одной или нескольких ЭМ, а данные (однородно) распределяются по всем ЭМ;
- конвейерный, когда сегментированная программа распределяется по машинам предварительно настроенного конвейера (или «кольца» или «линейки») и обеспечивается последовательное «пропускание» данных через все ЭМ конвейера.

Рекомендуемая методика распараллеливания сложных задач — крупноблочное распараллеливание, позволяющее за счет минимизации затрат на

межмашинные взаимодействия достичь линейной зависимости производительности ВС от числа ЭМ.

Требуемые уровни производительности, емкости памяти, надежности и живучести ВС достигаются путем подбора количества ЭМ и их состава, выбора структуры сети межмашинных связей, использования широких возможностей системных аппаратурно-программных средств по статической и динамической реконфигурации структуры и по варьированию состава системы.

Области применения ВС:

- традиционные сферы применения ЭВМ и векторных процессов, в которых возросли требования по обеспечению производительности, емкости памяти, надежности и живучести и где целесообразно сохранить совместимость вычислительных средств;
- сферы применения, связанные с решением трудоемких задач, таких как сложные задачи физики, механики сплошной среды, аэродинамики, баллистики, метеорологии, обработки изображений и речевых данных, задачи организации баз знаний, искусственного интеллекта;
- сложные большемасштабные системы, среди которых системы управления энергетическими установками, системы управления динамическими объектами и другие системы, характеризуемые высокой эффективностью, безотказностью, живучестью, развиваемостью, компактностью либо распределенностью своих ресурсов и т. п.

Таким образом, ВС семейства МИКРОС основываются на перспективных принципах обработки информации, строятся из аппаратурно-программных средств микропроцессорной техники, обладают гибкими возможностями по статической и динамической реконфигурации своих структур, позволяют достичь высокой производительности, надежности и живучести в широкой области применения.

Продолжением ряда ВС МИКРОС-1, МИКРОС-2 и МИКРОС-Т являются высокопроизводительные ВС с массовым параллелизмом семейства МВС.

7.7. Вычислительные системы семейства МВС

Вычислительные системы семейства МВС (МВС-100 и МВС-1000) созданы Научно-исследовательским институтом «Квант» (Москва) в содружестве с институтами РАН; руководитель работ — В.К. Левин (1929; академик РАН с 2002 г.). Они вобрали в себя отечественные и зарубежные достижения в области архитектуры ВС и производства микропроцессорных БИС.

Современная элементная база индустрии обработки информации — это большие интегральные схемы, среди которых выделяются: микропроцессоры

и микросхемы памяти (статические и динамические). В 1980-х и 1990-х годах электронная промышленность освоила производство высокопроизводительных микропроцессоров и микросхем памяти большой емкостью. Особенностью рынка БИС в те годы было огромное разнообразие универсальных микропроцессоров, а также сигнальных и медийных микропроцессоров.

Заметной вехой в вычислительной индустрии стала одна из разработок фирмы Inmos (Великобритания), именно — транспьютер (Transputer). Под транспьютером понимается микропроцессор с собственной внутренней памятью и коммуникационными каналами (линками) для соединения с другими транспьютерами. Первый транспьютер — Т414 — был разработан фирмой Inmos в 1983 г.; он выпускался серийно с 1983 г. и имел следующие технические характеристики:

- разрядность — 32 двоичных разряда;
- тактовая частота — 20 МГц;
- быстродействие — 10 MIPS;
- объем внутренней памяти — 2 К байт;
- число линков — 4;
- скорость передачи информации по линку — 5, 10, 20 Мбит/с.

Как уже отмечалось, транспьютер являлся простейшим вариантом ЭМ (см. § 3.4. и разд. 7.6.1); он служил для реализации не только вычислительных, но и коммуникационных функций. В 1980-х годах транспьютер был эффективным функционально-конструктивным элементом для построения ВС с массовым параллелизмом. Однако в 1990-х годах транспьютер использовался лишь только в качестве коммуникационного элемента для построения ВС.

Среди универсальных высокопроизводительных микропроцессоров перспективными для построения ВС были семейства:

- i 860 компании Intel;
- PowerPC альянса компаний IBM, Apple и Motorola;
- Alpha корпораций DEC и Compaq.

Для микропроцессоров названных семейств диапазон производительности (с точностью до порядков) составляет от 100 MFLOPS до 10 GFLOPS.

В середине 1990-х годов промышленность стала выпускать сигнальные и медийные микропроцессоры, которые наряду с большими вычислительными возможностями обладали развитыми коммуникационными средствами. Примером служит семейство TMS 320C4x компании Texas Instruments, микропроцессоры которого обеспечивают от четырех до шести коммуникационных портов с пропускной способностью каждого в несколько десятков Мбайт в секунду.

Таким образом, в начале 1990-х годов элементная база ВТ (универсальные и коммуникационные микропроцессоры) была достаточно развита и позволяла приступить к созданию MIMD-систем с производительностью $10^9 \ldots 10^{12}$ FLOPS.

Научно-исследовательский институт «Квант» (Москва) Министерства радиопромышленности Советского Союза был одним из наиболее мощных отечественных коллективов по проектированию ВС с массовым параллелизмом. Институт, пройдя путь от специализированных ЭВМ и ВС с архитектурой SIMD, в те же годы приступил к созданию ВС с MIMD-архитектурой. В указанные годы к средствам обработки информации стали предъявляться требования по производительности, надежности и живучести, которые никак не могли быть удовлетворены концепцией ЭВМ Дж. фон Неймана и SIMD-архитектурой. Именно к этому времени активизировалось сотрудничество НИИ «Квант» с несколькими научными группами Сибирского отделения АН СССР; особенно плодотворным и многолетним оно было с Отделом вычислительных систем СО РАН.

Семейство МВС (многопроцессорных вычислительных систем) — это, по сути, промышленное расширение ряда: МИКРОС-1, МИКРОС-2, МИКРОС-Т. Данное семейство имеет несколько генераций, среди которых: МВС-100 (1992–1996), МВС-1000 (1997–2000) и МВС-15000ВМ (2003–2005).

Архитектура систем семейства МВС:

- MIMD-архитектура;
- распределенность средств управления, обработки и памяти;
- массовый параллелизм при обработке информации;
- программируемость структуры сети межмашинных связей;
- масштабируемость, модульность и однородность.

7.7.1. Функциональная структура систем семейства МВС

Рассмотрим архитектурные и функциональные особенности вычислительных систем семейства МВС (в сравнении с МИКРОС).

Элементарные машины. Генерации МВС-100 и МВС-1000 имеют одну и ту же функциональную структуру ЭМ — структуру ЭМ системы МИКРОС-Т (см. рис. 7.22). Однако технические реализации ЭМ для МВС-100 и МВС-1000 различны — каждая из ЭМ комплектуется из микропроцессоров, соответствующих времени разработки систем.

Для формирования ЭМ вычислительных систем МВС-100 использовались в качестве:

• коммуникационных процессоров — транспьютеры Inmos T425 или T805;

• вычислительных процессоров — i860 (i80360XR и i80860XP) или микропроцессор PowerPC (PowerPC 601 и PowerPC 603).

Элементарная машина МВС-100 имела следующие технические характеристики:

• быстродействие — порядка 10^2 MFLOPS;

• емкость главной памяти (ГОП) — 8...32 Мбайт;

• емкость коммуникационной памяти (КОП) — 2...8 Мбайт;

• пропускную способность канала межмашинной связи (одного из четырех линков) — 2 Мбайт/с.

Для конфигурирования ЭМ систем МВС-1000 используются мощные микропроцессоры:

• коммуникационный процессор — либо транспьютероподобный микропроцессор TMS 320C44, имеющий четыре линка с пропускной способностью каждого 20 Мбайт/с, либо связной микропроцессор SHARC ADSP 21060 фирмы Analog Devices, имеющий шесть линков с пропускной способностью каждого 40 Мбайт/с;

• вычислительный процессор — микропроцессор Alpha 21164.

Элементарная машина МВС-1000 характеризуется следующими параметрами:

• быстродействие — 1...2 GFLOPS;

• емкость ГОП — 0,1...2 Гбайт;

• пропускная способность канала межмашинной связи (одного из четырех или шести линков) — 20 Мбайт/с или 40 Мбайт/с.

Структурный модуль. Структура сети межмашинных связей в ВС семейства МВС подобна двумерному тору (для 4-линковых ЭМ). Структурный модуль ВС представляет собой «матрицу» из 4×4 связных ЭМ (рис. 7.25). Граничные линки модуля используются следующим образом:

1) четыре линка угловых ЭМ матрицы — для организации двух диагональных связей;

2) оставшиеся четыре линка угловых машин — для подсоединения хост-компьютеров (Host — ведущая машина) и внешних устройств (ВУ) и для связи с ЭМ других модулей;

3) восемь линков — для соединений с подобными структурными модулями.

Максимальная длина пути между ЭМ (диаметр) в структурном модуле равна 3. Следует отметить, что этот параметр в гиперкубической структуре из 16 ЭМ равен 4. Следовательно, рассматриваемый 16-машинный структурный модуль характеризуется меньшими задержками при передаче информации между ЭМ в сравнении с 4-мерным гиперкубом (см. рис. 3.2).

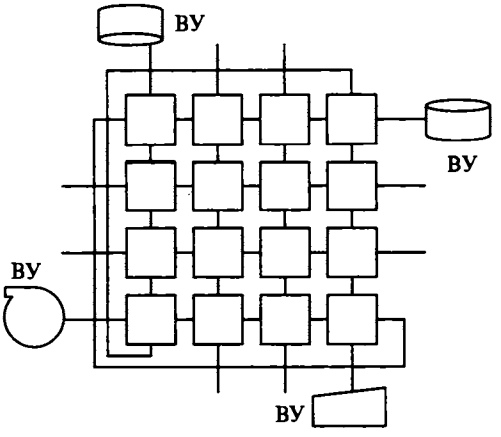


Рис. 7.25. Структурный модуль ВС семейства МВС:
ВУ — внешнее устройство

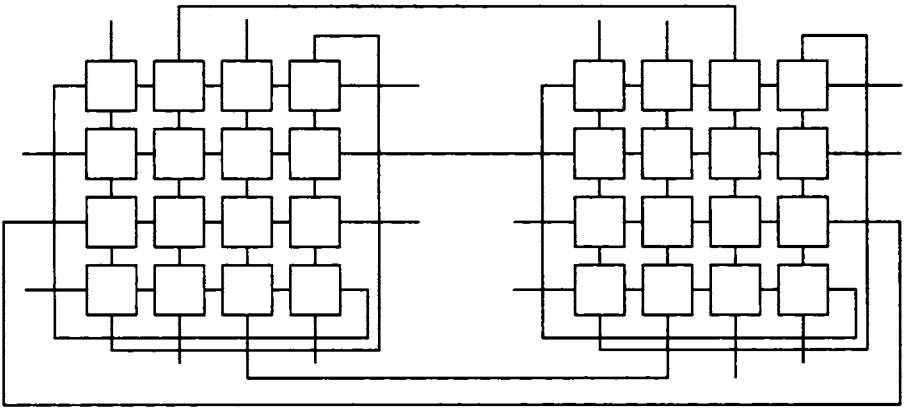


Рис. 7.26. Базовый вычислительный блок ВС семейства МВС

Конструктивным модулем ВС семейства МВС является базовый вычислительный блок (БВБ), содержащий два структурных модуля или 32 ЭМ (рис. 7.26). Диаметр структуры вычислительного блока равен 5, как в 32-вершинном гиперкубе. Свободные линки (максимально 16) вычислительных блоков используются для организации конфигураций ВС с числом ЭМ не менее 64.

На рис. 7.27 представлена структура 64-машинной системы семейства МВС, конфигурации ВС из двух БВБ.

Для формирования многомашинных конфигураций ВС могут быть использованы дополнительные коммуникационные процессоры (транспьютеры).

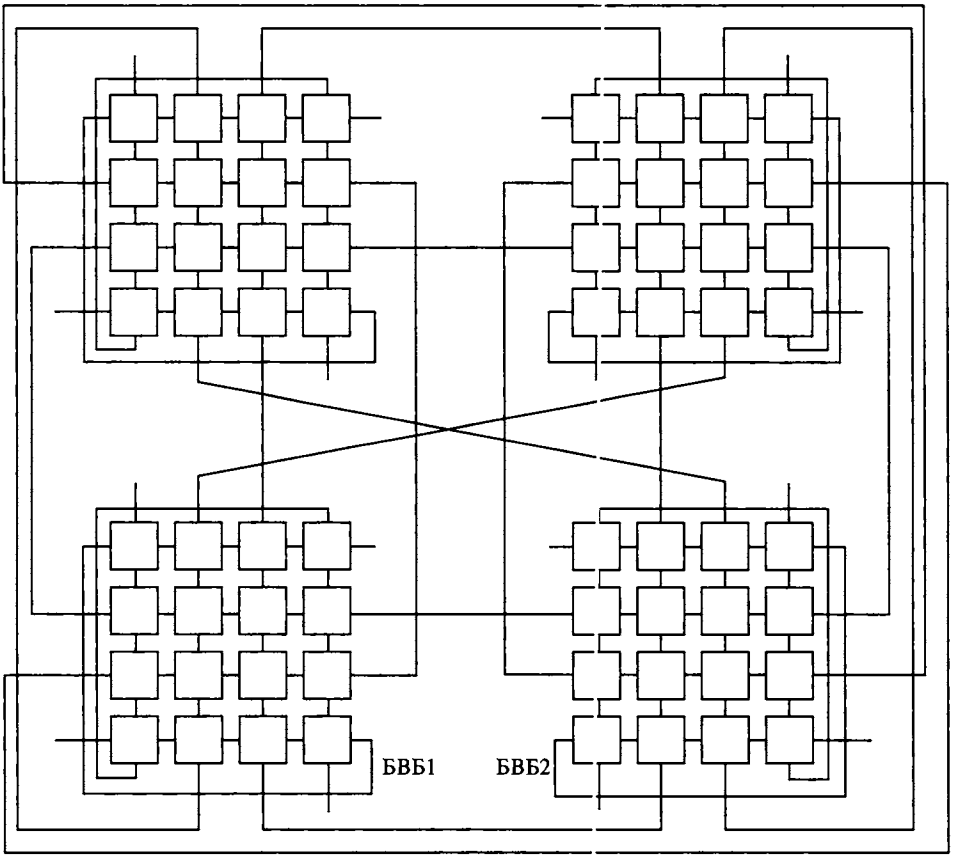


Рис. 7.27. Структура 64-машинной ВС семейства МВС:
ББВ — базовый вычислительный блок

При этом в композиции из ББВ и коммуникационного процессора (КП) один из внешних линков ББВ отождествляется с одним линком КП. Следовательно, в композиции ББВ & КП появляется два дополнительных внешних линка (по сравнению с ББВ). На рис. 7.28 приведена одна из возможных структур ВС из 128 ЭМ.

7.7.2. Конструкция и управление вычислительной системой семейства МВС

Для размещения элементарных машин ВС используются стандартные стойки размером $0{,}6 \times 0{,}8 \times 2{,}2$ м^3 Каждая стойка имеет блоки вторичного электропитания и вентиляции. Стойка рассчитана на 64 ЭМ и два ББВ. Каждый

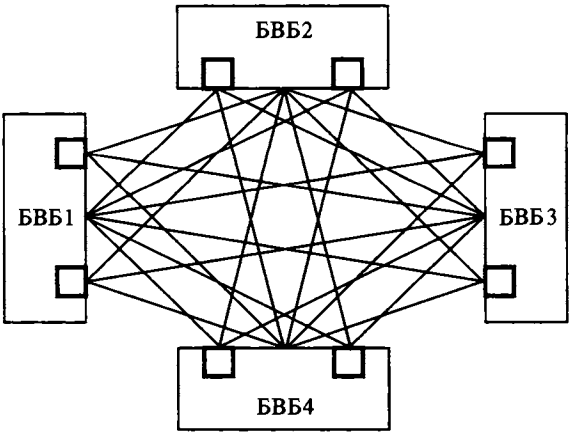


Рис. 7.28. Структура 128-машинной ВС семейства МВС:
ББВ — базовый вычислительный блок; □ — коммуникационный процессор

ББВ смонтирован на типовой многослойной плате. Вес стойки — 200 кг, потребляемая мощность — 4 кВт.

В многомашинных конфигурациях ВС используется несколько стоек. Так, для достижения производительности ВС порядка 1 TFLOPS требуется 512 ЭМ, следовательно, система должна быть размещена в восьми стойках.

Для обеспечения доступа к ВС извне, для управления множеством процессоров и внешними устройствами используется хост-компьютер. В качестве хост-компьютера могут служить рабочая станция AlphaStation с процессором Alpha или персональный компьютер с архитектурой IBM PC.

7.7.3. Программное обеспечение вычислительных систем семейства МВС

Операционная система выполняет функции по оптимальному использованию ресурсов ВС (коллектива ЭМ, средств ввода-вывода информации) и обеспечивает доступ пользователей. Операционные системы Digital Unix (Tru64 Unix) для AlphaStation и Linux для IBM PC устанавливаются в хост-компьютерах.

Средства программирования представлены языками и компиляторами FORTRAN 77, C и C++.

Коммуникационное ПО в ВС семейства МВС (генерации МВС-1000) строится на основе специализированной «транспортной службы» Router$^+$. В частности, на базе Router$^+$ реализована адаптированная к МВС-1000 библио-

тека интерфейса параллельного программирования MPI (Message Passing Interface). Аналогичные реализации допустимы для интерфейсов PVM, GNS, DVM, HPF и др. Функции Router$^+$ доступны из пользовательских программ.

Для задач визуализации разработана специализированная библиотека GraphLib.

В вычислительных системах генерации МВС-1000 реализован многопользовательский режим и удаленный доступ через специальный промежуточный компьютер (Gateway). Для пользователей обеспечивается Unix-совместимая среда компиляции и запуска программ.

7.7.4. Области применения вычислительной системы семейства МВС

Вычислительные системы семейства МВС — масштабируемые, следовательно, спектр областей их применения достаточно широк. Для каждой прикладной области может быть выбрана адекватная по составу и техническим параметрам конфигурация ВС. Системы предназначены прежде всего для решения сложных задач.

Приведем список задач, параллельные алгоритмы решения которых эффективно реализуются на ВС семейства МВС:

1) задачи расчета аэродинамики летательных аппаратов, в том числе интерференции при групповом движении;

2) расчет трехмерных нестационарных течений вязкосжимаемого газа;

3) расчет течений с локальными тепловыми неоднородностями в потоке;

4) квантовая статистика поведения вещества при экстремальных условиях;

5) структурообразование биологических макромолекул;

6) моделирование динамики молекулярных и биомолекулярных систем;

7) дифференциальные игры, динамические задачи конфликтов управления;

8) механика деформируемых твердых тел (с учетом процессов разрушения).

Данный список охватывает области фактического применения ВС генераций МВС-100 и МВС-1000.

Вычислительные системы генерации МВС-1000 используются для оснащения суперкомпьютерных центров России. Одна из первых 96-машинных систем МВС-1000 с производительностью 200 GFLOPS была установлена в Москве в Межведомственном суперкомпьютерном центре (МСЦ). Центр был открыт 5 ноября 1999 г.

В соответствии с планами развития МСЦ в 2001 г. была установлена система МВС-1000М с 768 процессорами, имеющая производительность 1 TFLOPS.

Вычислительная система МВС-1000М по своей архитектуре и функциональной структуре принципиально отличается от МВС-1000. Система МВС-1000М — это кластер, компонуемый из 2-процессорных вычислительных узлов. Каждый узел представляет собой композицию из двух микропроцессоров Alpha 21264 с тактовой частотой 667 МГц, оперативной памяти емкостью 2 Гбайт и внешней памяти на жестком диске. Узлы связаны между собой коммуникационной сетью Myrinet и работают под управлением ОС RedHat Linux 6.2. В качестве управляющей сети в МВС-1000М используется Fast Ethernet. Имеется реализация интерфейса MPI—MPICH-GM.

Вычислительная система МВС-1000М собрана в 18 стойках; потребляемая мощность — 120 кВт; стоимость системы — 10 млн долл.

С 2006 г. в МСЦ эксплуатируется кластерная вычислительная система МВС-15000 ВМ, состоящая из 1148 процессоров и имеющая пиковую производительность 10,1 TFLOPS (производительность на тесте LINPACK составляет 6,646 TFLOPS).

В основу архитектуры и функциональной структуры системы МВС-15000 ВМ заложена платформа IBM PowerPC. Для формирования ВС использованы микропроцессоры PowerPC 970 с тактовой частотой 2,2 ГГц и межпроцессорная коммуникационная сеть Myrinet.

В списке самых мощных компьютеров мира (Top500) вычислительная система МВС-15000 ВМ в 2006 г. (в 28-й редакции) занимала 99 позицию, а в 2007 г. (в 29-й редакции) переместилась на 187 место.

7.8. Анализ вычислительных систем с программируемой структурой

1. Вычислительные системы с программируемой структурой — гибкий класс средств обработки информации с архитектурой MIMD. Архитектура таких систем не имеет принципиальных ограничений на пути ко все более полному воплощению в реализациях принципов модели коллектива вычислителей как на макроуровне (на уровне системы в целом), так и на микроуровне (на уровне одной ЭМ). Концепция ВС с программируемой структурой свидетельствует о революционном отходе от классической архитектуры ЭВМ Дж. фон Неймана.

2. Накоплен опыт создания ВС с программируемой структурой («Минск-222», МИНИМАКС, СУММА, МИКРОС-1, МИКРОС-2, МИК-

РОС-Т, МВС-100, МВС-1000 и др.). Реализованы мультитранспьютерные живучие ВС с программируемой структурой МИКРОС-Т и МВС-100 и высокопроизводительные системы МВС-1000; распределенные децентрализованные ОС, инвариантные к структуре сети межмашинных связей и количеству ЭМ; языки параллельного программирования; комплексы отказоустойчивых прикладных параллельных программ.

Результаты многолетней эксплуатации созданных систем показывают высокую эффективность архитектурных решений, присущих концепции ВС с программируемой структурой. Эта концепция позволяет в условиях современных ограничений в производстве средств микропроцессорной техники (в технологии БИС) строить промышленные ВС, множество конфигураций которых составляют семейства (ряды) совместимых экономичных моделей для широкого диапазона по производительности, надежности и живучести. Примером служит ВС МИКРОС, подход в реализации которой допускает формирование требуемых сосредоточенных и распределенных моделей из серийных средств микропроцессорной техники и обеспечивает на многие годы возможность эволюционного совершенствования системы в соответствии с развитием технологии БИС.

3. Вычислительные системы с программируемой структурой — это коллектив ЭМ, количество которых и структура сети связей между которыми допускают варьирования в широких пределах. Рост производительности таких ВС обеспечивается увеличением количества ЭМ и расширением конфигураций каждой из них.

Концепция ВС с программируемой структурой позволяет создавать технико-экономически эффективные средства обработки информации, обладающие сверхвысокой производительностью, надежностью и живучестью.

8. ТРАНСПЬЮТЕРНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Развитие средств обработки информации всегда основывалось на перспективных технико-экономически обоснованных архитектурных и структурных решениях и на технологических достижениях в области элементной базы. Ярким примером гармонического сочетания этих факторов могут служить высокопроизводительные средства (1980-е годы), получившие название «Транспьютерные вычислительные системы». Такие ВС нашли применение во многих областях науки, техники и экономики, они внедрили в технику обработки данных архитектурные принципы модели коллектива вычислителей.

Транспьютерные ВС сыграли свою выдающуюся роль в развитии средств обработки информации с массовым параллелизмом. Они, безусловно, послужат основой будущих разработок суперВС как ансамблей микропроцессоров, размещаемых на крупномасштабных (неразрезных) полупроводниковых пластинах.

В данной главе будут даны понятие о транспьютерных ВС и описания архитектур транспьютеров семейств T200, T400, T800 и T9000 фирмы Inmos Ltd. (Великобритания).

8.1. Понятие о транспьютерных вычислительных системах

8.1.1. Предпосылки создания транспьютерных вычислительных систем

Совершенствование параллельных средств обработки информации — ВС — базируется главным образом на достижениях в области архитектуры. При этом развивается и макроархитектура ВС (архитектура коллектива вычислителей-микропроцессоров), и микроархитектура (архитектура микропроцессора).

Современная элементная база ВТ — это многоразрядные микропроцессоры с тактовой частотой до 100 ГГц. Эмпирический закон Мура (Gordon Moore, одного из основателей фирмы Intel) свидетельствует о том, что каж-

дые 18 месяцев плотность упаковки транзисторов на кристалле удваивалась и соответственно возрастала эффективность микропроцессора. Увеличение возможностей микропроцессора обеспечивалось не только за счет достижений технологии, а также и совершенствованием его архитектуры. Современные высокопроизводительные микропроцессоры имеют развитую архитектуру (микроархитектуру), которая существенно отличается от последовательной фон-неймановской; они основываются на конвейеризации вычислений.

Совершенно естественно возникла идея создания параллельных ВС как коллективов микропроцессоров, размещенных на одной пластине. Эта идея могла появиться на вполне определенном этапе развития ВТ, который характеризовался достаточно совершенной кремниевой технологией в производстве микропроцессорных БИС и созданием промышленных мультипроцессорных ВС. Этот этап — конец 1970-х — 1980-е годы. В плане реализации этой идеи и возникли транспьютерные ВС.

Работы по транспьютерным ВС зародились и первоначально велись только в фирме Inmos Ltd. (Великобритания). Фирма Inmos ставила перед собой цель выйти на микроэлектронные рынки США и Европы. В соответствии с поставленной целью в фирме Inmos и были инициированы работы по построению транспьютерных ВС с быстродействием 2,5 млрд опер./с.

8.1.2. Общие сведения о транспьютерных вычислительных системах

Транспьютерная ВС — это ансамбль элементарных процессоров — транспьютеров, взаимодействующих через регулярную (как правило, двумерную) *сеть связи.* Все транспьютеры имеют идентичную функциональную структуру. *Транспьютер — специализированный микропроцессор,* включающий в себя собственно микропроцессор (МП) с архитектурой, поддерживающей параллельные процессы, локальную память (ЛП) и линки (Л); линк — канал ввода-вывода информации, предназначенный для подключения соседнего транспьютера (рис. 8 1). Работы по транспьютерным ВС первоначально были ориентированы на создание высокоэффективного средства обработки информации в виде полупроводниковой пластины, содержащей несколько сотен элементарных процессоров (см. рис. 8.1 и 6.6).

Замысел, отраженный на рис. 8.1, не был воплощен на практике. Развитие ВС пошло по пути построения параллельных конфигураций из «дискретных» транспьютерных модулей.

Термин «транспьютер» является синонимом понятия «элементарная машина», если следовать отечественной терминологии (см. § 3.4 и гл. 7). Транспьютер — микроэлектронный элемент для построения распределен-

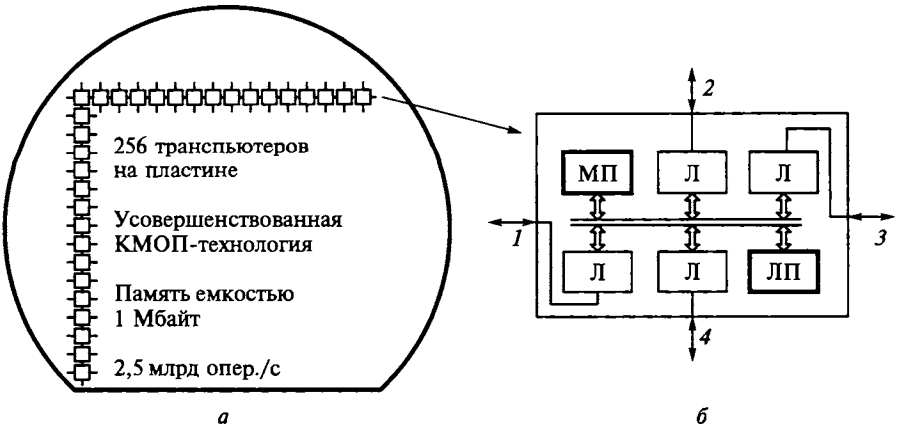


Рис. 8.1. Транспьютерная система фирмы Inmos:

а — система; *б* — транспьютер; МП — микропроцессор; Л — линк; ЛП — локальная память; *1–4* —линки

ных ВС с массовым параллелизмом. Название этого элемента служит ключом к пониманию как его архитектурных возможностей, так и технологии изготовления. В самом деле, ТРАНСПЬЮТЕР — это аббревиатура, образованная из слов «транзисторный компьютер» (TRANSPUTER — TRANSistor comPUTER). Следовательно, название «транспьютер» указывает на то, что этот элемент (по мнению разработчиков) должен был стать основой для построения нового поколения (не-фон-неймановских) ВС, точно так же, как транзистор послужил базой для создания первых полупроводниковых ЭВМ второго поколения. Первый транспьютер был создан в середине 1983 г. специалистами фирмы Inmos.

1980–1990-е годы характеризуются достаточно бурным подъемом транспьютерной индустрии. Термин «транспьютерные технологии» использовали чуть ли не чаще, чем «параллельные вычислительные технологии».

Создание транспьютера потребовало заметных трудозатрат и финансовых вложений. На разработку первого транспьютера было затрачено 100 человеко-лет (при этом широко использовались средства автоматизированного проектирования).

Транспьютерная вычислительная система — наиболее полный вариант технического воплощения на Западе модели коллектива вычислителей (см. § 3.1 и 3.2). Говоря другими словами, транспьютерная ВС является микроэлектронной реализацией концепции ВС с программируемой структурой (см. гл. 7). В самом деле:

• транспьютер — это микроэлектронная элементарная машина (простой конфигурации, без внешних устройств);

• транспьютерная ВС является множеством транспьютеров, взаимодействующих друг с другом через сеть связей (линков);

• линки — программно управляемые каналы для обмена информацией между соседними транспьютерами.

Следовательно, *транспьютерные ВС — это разновидность вычислительных систем с программируемой структурой* (см. гл. 7 и [5, 6]). Очевидно, что ВС на кристалле (Systems-on-Chip) является ближайшей перспективой в высокопроизводительной индустрии обработки информации.

8.2. Архитектура транспьютеров семейств T200, T400 и T800

8.2.1. Общие сведения об архитектуре транспьютера и языке программирования Оккам

Транспьютер, если его рассматривать в архитектурном плане, является функционально-структурным элементом для построения распределенных ВС с массовым параллелизмом. Транспьютер в технологическом плане — это программируемая микропроцессорная БИС. Транспьютер включает в себя собственно МП, реализующий операции с фиксированной запятой, локальную память для хранения данных и программ, параллельный интерфейс внешней памяти, линки (Links) — последовательные двунаправленные каналы передачи данных для непосредственных связей с другими (соседними) транспьютерами. Транспьютеры выпускаются семействами, в архитектурно развитых семействах в состав транспьютера входит и процессор с плавающей запятой.

Архитектура и конструкция транспьютера ориентированы на реализацию специального языка параллельного программирования Оккам (Occam). Этот язык был создан фирмой Inmos Ltd. совместно с Оксфордским университетом (Oxford University, Великобритания). Свое название он получил в честь английского философа и логика XIV в. Уильяма Оккама (W. Ockham, W. Occam). Согласно принципу «бритвы Оккама» («Entia non sunt multiplicanda praeter necessitatem») или «сущности не следует умножать без необходимости», или «сущность не должна превышать необходимость», всегда следует стремиться к простоте, т. е. из двух одинаково эффективных решений нужно выбирать более простое. Этим названием разработчики языка, по-видимому, стремились подчеркнуть то, что он не содержит никаких лишних конструкций.

Язык Оккам основан на теории CSP (Communicating Sequential Processes) — «взаимодействующих последовательных процессов», предложенной в 1978 г. профессором Оксфордского университета Ч.А.Р. Хором

(C.A.R. Hoare). Достоинство языка Оккам заключается в объединении двух взаимодействий — синхронизации и обмена информацией. Последнее обеспечивает синхронизацию процессов перед любыми обменами информацией между ними. При этом автоматически исключаются конфликты при обращении процессов к общим ресурсам (к файловой системе, дисплею, клавиатуре). В самом деле, обращение процесса к ресурсу — это суть обмен информацией с соответствующим процессом, а в любой момент времени каждый процесс может взаимодействовать только с одним другим процессом.

В 1982 г. разработчиками был создан простой и элегантный язык Оккам 1, содержащий небольшое число конструкций, он был универсальным и имел эффективные средства для программирования параллельных процессов. В 1986 г. была завершена разработка расширенной версии языка, получившей название Оккам 2. Элегантность языка Оккам, безусловно, делает его привлекательным для изучения основ параллельного программирования.

Отметим концептуальные особенности языка Оккам. В этом языке *процесс (Process)* является основным рабочим элементом, а *канал (Channel)* — основным элементом связи, позволяющим параллельным процессам взаимодействовать друг с другом. Каждый процесс порождается, выполняет некоторые действия и завершается. Действиями могут быть операции присваивания, ввода и вывода. При выполнении присваивания происходит изменение значения переменной, при вводе процесс получает значение из канала, а при выводе он посылает значение в канал.

Канал можно рассматривать как средство синхронизации, позволяющее двум процессам обмениваться информацией. Если два процесса связаны каналом, то никакие другие процессы не могут его использовать. Канал позволяет передавать данные только в одном направлении. Таким образом, для принимающего процесса он является только элементом чтения, а для передающего — только элементом записи.

Канал является единственным средством коммуникации параллельных процессов. Общие внешние переменные для этих процессов нельзя использовать для взаимодействия между ними. Это ограничение, однако, не запрещает использовать одни и те же переменные в параллельных процессах — должно выполняться требование, что ни один из процессов не меняет их значения (при присваивании или вводе). Компиляторы с Оккама проводят соответствующий контроль.

Обмен данными между процессами через каналы в Оккаме подобен обмену информацией между синхронно работающими электронными устройствами через линии связи: процесс, готовый к передаче (приему), приостанавливается и ждет готовности процесса получателя (отправителя); после осуществления обмена каждый процесс продолжает работать самостоятельно. Процесс может получать и передавать данные одновременно по не-

скольким различным каналам, причем это может происходить одновременно с выполнением операций присваивания.

Механизм реализации взаимодействий между процессами основывается на принципе близкодействия (см. разд. 3.2 1). Суть любого взаимодействия — обмен сообщениями по двухточечным каналам связи (Point-to-Point Communication Channel). Под каналом (Channel) здесь понимается не физический канал (Link), а языковая конструкция, обеспечивающая организацию и выполнение связи между процессами. Следует заметить, что при реализации взаимодействий между процессами, протекающими в различных транспьютерах, будут использованы в конечном итоге физические линки. Каждый такой линк, связывающий два транспьютера, используется для реализации двух каналов (Channels) между процессами языка Оккам (по одному каналу в каждом направлении).

Использование каналов для организации обменов между параллельными процессами подчеркивает *распределенный характер параллелизма в языке Оккам*. Если программа выполняется на множестве транспьютеров, каждый из которых имеет свою локальную память, то обмены путем межтранспьютерных посылок сообщений являются естественными. С другой стороны, если множество параллельных процессов выполняется на одном транспьютере (в режиме разделения времени), то естественны были бы обмены через общую память. Однако последнее не разрешено в Оккаме: одна и та же модель параллелизма используется как внутри транспьютера, так и между транспьютерами. Это позволяет, во-первых, устранить ошибки программирования, связанные с использованием общих данных; во-вторых, облегчить переносимость программ на ВС с произвольными конфигурациями; в-третьих, параллельные программы, предназначенные для мультитранспьютерной ВС, можно оттранслировать, протестировать и выполнить на однотранспьютерной системе.

Программа на Оккаме имеет иерархическую структуру; допустимо объединение множества взаимодействующих процессов в один «больший» процесс. Для этого в языке имеется набор специальных конструкторов.

Применение двухточечных языковых и физических каналов (Channels and Links), а также синхронных связей между процессами существенно упрощают построение мультитранспьютерных систем. Такой подход в сравнении с шинными мультипроцессорными ВС обладает рядом преимуществ, среди которых:

- простота аппаратурной реализации двухточечного канала (здесь имеется только один источник сообщения и один адресат, а в шинных структурах — множества указанных объектов);
- рост пропускной способности сети межтранспьютерных связей ВС при увеличении числа транспьютеров (в шинных мультипроцессорных системах имеет место обратная зависимость);

• возможность параллельного обращения транспьютеров (точнее, их микропроцессоров) к памяти (каждого к своей локальной памяти); следовательно, быстродействие памяти (композиции ЛП всех транспьютеров) в мультитранспьютерных ВС выше, чем у общей памяти большой емкости в мультипроцессорных ВС с шинной структурой.

Транспьютеры могли использоваться и как обычные микропроцессоры, они по своим техническим характеристикам не уступали однокристальным микропроцессорам. Сравним первый 32-разрядный транспьютер IMS T414 фирмы Inmos (1985) с 32-разрядным микропроцессором 80386 фирмы Intel (1986). Тактовые частоты названных схем сравнимы: транспьютер T414 мог иметь частоту 15 или 20 МГц, а процессор 80386 — 16 МГц. Архитектура T414 позволяла выполнять команды в среднем за 2 такта, следовательно, она обеспечивала быстродействие около 10 MIPS (млн операций над целыми числами в секунду, см. § 2.6). В микропроцессоре 80386 на выполнение команды в среднем требовалось 4 такта, что позволяло достичь быстродействия, равного только 4 MIPS. По размерам кристалла (8,7 × 8,9 мм^2) и числу транзисторов (200 тыс.) T414 несколько уступал 80386 (270 тыс. транзисторов), однако в транспьютере по сравнению с микропроцессором было достигнуто четырехкратное снижение энергопотребления (500 мВт вместо 2 Вт). При этом следует отметить, что на кристалле T414 кроме микропроцессора были размещены оперативная память емкостью 2 К байт, интерфейс внешней памяти (обеспечивающий прямой доступ к памяти с пропускной способностью 25 Мбайт/с) и четыре линка. Микропроцессор 80386 имел традиционную архитектуру: встроенная память в нем отсутствовала и имелся лишь один канал для обмена с внешними памятью и устройствами (пропускная способность — 32 Мбайт/с). Для формирования параллельных ВС на базе Intel 80386 требовалась разработка элементарного процессора как платы с множеством БИС. Габариты формируемой ВС (даже при небольшом количестве ЭП) были значительными, возможности межпроцессорных обменов информацией — весьма ограниченными, стоимость и энергопотребление — высокими, а надежность — низкой.

Следует еще раз подчеркнуть, что важной архитектурной особенностью транспьютеров является то, что они рассчитаны на реализацию языка параллельного программирования высокого уровня Оккам (Occam). Работа языка Оккам сводится к представлению сложной задачи в виде совокупности ветвей-процессов и организации обменов данными между ними. Транспьютер имеет аппаратурную поддержку основных конструкций Оккам, что позволяет выполнять программы, написанные на этом языке, с эффективностью Ассемблера.

Итак, комплексный проект, включающий разработку транспьютера и языка Оккам, имел следующие цели:

1) создание микроэлектронного элементарного процессора (транспьютера), ориентированного на формирование ВС с массовым параллелизмом;

2) обеспечение поддержки параллельной обработки информации как на аппаратурном, так и на программном уровнях;

3) достижение «масштабируемости» создаваемых ВС; увеличение производительности ВС при наращивании числа параллельно работающих процессоров;

4) использование параллельного языка высокого уровня для программирования элементарного процессора.

Работы по созданию транспьютеров и языка Оккам велись в рамках Западно-Европейской стратегической программы исследований и разработок в области информационной технологии Esprit (European Strategic Program of Research and Development in Information Technology).

8.2.2. Семейства транспьютеров

По первоначальному проекту предполагалось создать простой транспьютер как процессор со стековой архитектурой, имеющий небольшой набор однобайтовых и однотактных команд (использование микропрограмм не планировалось). Следовательно, при воплощении замысла транспьютер имел бы архитектуру RISC (Reduced Instruction Set Computer).

Архитектуры RISC в 1980-х годах приобрели большую популярность. Вычислительные средства с такой архитектурой характеризуются:

- сокращенной системой команд;
- простотой команд (имеющих фиксированный формат и однотактное исполнение);
- простой системой адресации;
- наличием емкой сверхоперативной памяти (кэш и большого числа регистров);
- аппаратурной реализацией команд (отсутствием микропрограмм);
- ориентацией на язык высокого уровня;
- технико-экономически оптимальным соотношением между аппаратурными и программными средствами.

Степень воплощения отмеченных архитектурных свойств зависит от реализаций микропроцессоров.

Транспьютер имеет заметные отклонения от канонической RISC-архитектуры. В самом деле, система команд транспьютера является далеко не сокращенной. Она насчитывает более ста команд, среди которых превалируют команды, выполняющиеся более чем за один такт (например, в первом транспьютере Т414 из 110 команд только 15 являются однотактными).

Тем не менее такие принципы, как простота команд, ориентация на применение языков высокого уровня и обеспечение оптимального соответствия между аппаратными и программными средствами в транспьютере прослеживаются явно. Можно подчеркнуть, что транспьютер соединил в себе основные достоинства RISC-архитектуры с достоинствами микропрограммирования и использования быстрой внутренней памяти.

Фирма Inmos производила следующие семейства транспьютеров: T200, T400, T800, T9000. Транспьютеры первых трех семейств выполнялись, как правило, в керамических корпусах площадью 30 × 30 мм^2 и с 84-мя выводами-штырями. Модели семейств совместимы (по системе команд) снизу вверх. Характеристики транспьютеров семейств T200, T400, T800 приведены в табл. 8.1. Семейство T9000 представлено вариантами транспьютера T9000, отличающимися друг от друга по тактовой частоте, его архитектура достаточно развита, она сочетает в себе достоинства высокопроизводительных микропроцессоров и транспьютеров предшествующих семейств. Далее детально рассмотрим архитектуру первых трех семейств транспьютеров.

Таблица 8.1

Семейство транспьютеров	Тип транспьютера	Разрядность	Емкость ЛП[1], К байт	Плавающая арифметика	Команды отладки[2]	Число линков	Буфер[3] в линке
T200	T212	16	2	–	–	4	–
	T222	16	4	–	–	4	+
	T225	16	4	–	+	4	+
	M212	16	2	–	–	2	–
T400	T400	32	2	–	–	2	–
	T414	32	2	–	–	4	+
	T425	32	4	–	+	4	+
T800	T800	32	4	+	–	4	+
	T801	32	4	+	+	4	+
	T805	32	4	+	+	4	+

[1] ЛП — локальная память, т. е. внутрикристальная оперативная память.

[2] Команды отладки позволяют реализовать возможность более эффективной отладки (Debugging) программ в транспьютерной системе.

[3] Буфер в линке — 8-разрядный буферный регистр для хранения 1 байт данных, принимаемого из канала; отсутствие этого буфера в ранних моделях транспьютеров приводило к задержкам при обменах информацией между транспьютерами.

Семейство T200. Производство транспьютеров семейства T200 было открыто в середине 1986 г. Транспьютеры этого семейства 16-разрядные и

обычно используются как встроенные контроллеры (например, для накопителей на дисках) или как средства управления электронными коммутаторами при построении транспьютерных плат с программируемой структурой. Безусловно, их можно применять и как элементарные процессоры ВС (для параллельной обработки 16-разрядных данных). Транспьютер IMS M212 занимает особое место, он представляет собой контроллер гибкого и жесткого дисков и имеет два двунаправленных канала.

Семейство T400. В семейство входит первый транспьютер IMS T414, который начал серийно выпускаться фирмой Inmos в октябре 1985 г. Это семейство 32-разрядных транспьютеров, которые рассчитаны на реализацию операций арифметики с фиксированной запятой. Модель IMS T400 представляет собой кристалл в пластмассовом корпусе, имеет два линка, его цена (при партии в 50 000 шт.) в 1990 г. составляла всего 20 долл. (более чем на порядок ниже, чем IMS T414). Следует отметить, что транспьютер IMS T212 является 16-разрядным вариантом IMS T414 и полностью совместим с ним.

В состав транспьютера IMS T414 (рис. 8.2) входят процессор, ориентированный на обработку информации с фиксированной запятой, локальная (или внутренняя) статическая память емкостью 2 К байт, интерфейс внешней памяти с пропускной способностью 25 Мбайт/с, четыре двунаправленных линка и системный блок. Транспьютер реализован в виде БИС на кристалле 8,7 × 8,9 мм^2, число транзисторов — 200 тыс., тактовая частота — 20 МГц, быстродействие — около 10 MIPS, потребляемая мощность — 500 мВт.

Процессор аппаратурно реализует модель параллельных вычислений и связей между процессами, предлагаемую языком Оккам. В процессоре имеется планировщик процессов, который позволяет выполнять любое число параллельных процессов в режиме разделения времени. Связь между про-

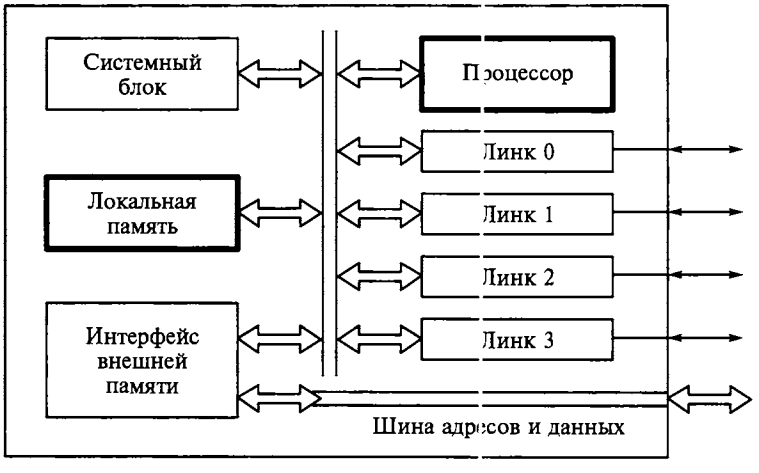


Рис. 8.2. Функциональная структура транспьютера IMS T414

цессами реализуется при помощи команд пересылок данных между блоками локальной памяти.

В процессоре обеспечивается два уровня приоритетов. Процессы высшего приоритета могут использоваться для маршрутизации при передаче информации и для быстрой реакции на внешние события. Реализована возможность программировать на языке Оккам аппаратурные и программные прерывания. Процессор содержит таймер, что позволяет выполняемому процессу либо осуществлять слежение за временем, либо ждать достижения определенного момента времени. Процессор способен работать с памятью емкостью 2^{32} байта (4 Гбит), при этом *локальная* (внутренняя) и *внешняя памяти* рассматриваются как единое пространство адресов. Под внешней памятью понимается оперативное запоминающее устройство, расположенное вне транспьютера. Связь процессора с ней осуществляется через интерфейс внешней памяти.

Процессор имеет средства обнаружения ошибок. Сигнал (флаг) об ошибке может быть использован для останова процессора с сохранением его состояния, что дает возможность выявить причину ошибки.

Линки 0–3 свидетельствуют о том, что транспьютер является четырехполюсником. Следовательно, из транспьютеров могут быть построены ВС с двумерной структурой. Линки обеспечивают скорость передачи данных 10 Мбит/с. Допускается параллельная работа процессора и линков. Каждый линк имеет последовательные вход и выход, которые используются для передачи как данных, так и управляющей информации (необходимой для реализации протоколов связи). Данные передаются в последовательном коде в виде последовательностей пакетов, каждый из которых содержит 1 байт информации (рис. 8.3). После передачи пакета данных транспьютер-отправитель ожидает получения пакета подтверждения от транспьютера-получателя о том, что он готов принимать следующий пакет данных.

Для синхронизации работы устройств транспьютера используются внутренние тактовые импульсы, вырабатываемые в системном блоке. При обмене данными (в синхронном режиме) обеспечивается правильное функционирование, если разность частот синхронизации в двух транспьютерах, связанных каналом, не превышает 400 имп./мин. Интенсивность отказов, обусловленных средствами синхронизации, оценивается значением 10^{-10} 1/ч.

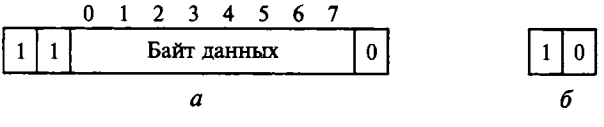


Рис. 8.3. Форматы транспьютерных пакетов:
а — пакет данных; *б* — пакет подтверждения

Интерфейс внешней памяти с помощью 32-разрядной шины адресов и данных обеспечивает для процессора быстрый доступ и к статическим, и к динамическим запоминающим устройствам с различными временными режимами работы. Более того, в интерфейсе заложена возможность программной настройки с учетом возможностей применяемой внешней памяти. Настройка проводится по специальной программе, которая способна генерировать требуемый режим работы памяти (в частности, формы сигналов).

В состав системного блока транспьютера входят устройство питания, тактовые генераторы, устройства инициализации и сброса.

Все устройства транспьютера связаны между собой общей двунаправленной 32-разрядной шиной адресов и данных. Эта шина позволяет процессору обращаться к локальной и внешней памяти и обмениваться информацией с внешней средой (например, другими транспьютерами) через линки.

Семейство T800. Это семейство 32-разрядных транспьютеров, способных выполнять арифметические операции с плавающей запятой. Модель IMS T800 производилась серийно с конца 1987 г. Транспьютер IMS T800 имел в своем составе 64-разрядный сопроцессор с плавающей запятой (FPU — Floating Point Unit) и локальную оперативную память емкостью 4 К байт. Число транзисторов в кристалле IMS T800 составляло 300 тыс.; при этом его площадь была увеличена (по сравнению с IMS T414) всего на 25 %. Это удалось достичь за счет оптимизации компоновки кристалла и применения 1,4-мкм КМОП-технологии.

С начала 1988 г. серийно выпускался транспьютер IMS T800 с тактовой частотой 20 МГц; среднее время выполнения одной команды в данном транспьютере — 2 такта, следовательно, его быстродействие при выполнении операций с фиксированной запятой было равно 10 MIPS. При реализации операций с плавающей запятой транспьютер IMS T800 обеспечивал быстродействие 1,5 MFLOPS и 1,1 MFLOPS соответственно для 32-разрядных и 64-разрядных чисел (у транспьютера IMS T414 эти значения значительно ниже: 0,1 и 0,025 MFLOPS).

Локальная (внутренняя) память позволяла осуществлять выборку слова за время одного такта, что при тактовой частоте 20 МГц составляет 50 нс. Следовательно, локальная память, по сути, была сверхоперативной.

Каждый из четырех линков IMS T800 обеспечивал скорость передачи данных, равную 10 Мбит/с. Реальная пропускная способность каналов связи между транспьютерами IMS T800 (при непосредственном соединении линков) оценивалась величинами 0,9 Мбайт/с и 1,2 Мбайт/с соответственно в полудуплексном и дуплексном режимах.

Интерфейс внешней памяти (до 4 Гбайт) позволял подключать к процессору БИС с различными видами запоминающих устройств (статических и динамических). Это достигалось возможностью реализации нескольких

стандартных режимов, обеспечивающих время обращения к внешней памяти от 3 до 12 тактов. В интерфейс была заложена возможность программирования специфичных режимов обращения к внешней памяти на случай, если ни один из стандартных режимов оказывался не подходящим по каким-либо причинам.

Важной архитектурной особенностью транспьютера IMS T800 было то, что его процессор (центральный), сопроцессор с плавающей запятой и четыре линка могли работать параллельно (до тех пор, пока не происходило обращение к одному и тому же ресурсу, например к внешней памяти).

Совмещения операций осуществлялись всегда, когда это было принципиально возможно. Так, при запросе процессора на запись слова во внешнюю память осуществлялась передача этого слова в интерфейс и собственно запись совмещалась с выполнением очередных команд процессора. Одновременно с этим могли выполняться команды приема-передачи данных по линкам и операции с плавающей запятой в сопроцессоре.

Традиционная система прерываний в транспьютере отсутствовала, а для обеспечения работы в режиме реального времени в состав процессора была включена система работы с приоритетами.

В транспьютере имелся таймер, который применялся как для отсчета текущего времени, так и для организации задержки выполнения процесса до наступления заранее определенного момента времени.

В транспьютер IMS T800 (кроме единой шины адреса и данных) были введены две дополнительные шины. Одна из них использовалась процессором для передачи управляющих команд в сопроцессор для плавающей арифметики и получения в ответ информации о состоянии. Другая шина служила для отдельного доступа процессора к линкам. Такая архитектура позволяла процессору и сопроцессору работать параллельно: во время вычисления процессором адресов сопроцессор проводил операции над числами с плавающей запятой.

Транспьютер IMS T801 в отличие от IMS T800 имел раздельные шины адресов и данных в интерфейсе внешней памяти. Это позволило использовать в качестве внешней памяти быстродействующие БИС статической памяти с временем обращения, равным двум тактам процессора. Разделение шин адресов и данных привело к изменению конструкции кристалла, в частности число выводов было увеличено до 100.

Транспьютер IMS T805 — наиболее распространенная модель семейства T800 (рис. 8.4). Состав IMS T805: 32-разрядный процессор, сопроцессор для выполнения операций над 64-разрядными числами с плавающей запятой, быстрая локальная память емкостью 4 К байт, интерфейс внешней памяти (расширяющий адресное пространство до 4 Гбайт), четыре линка, схемы обслуживания линков, таймер, схема обработки внешних запросов,

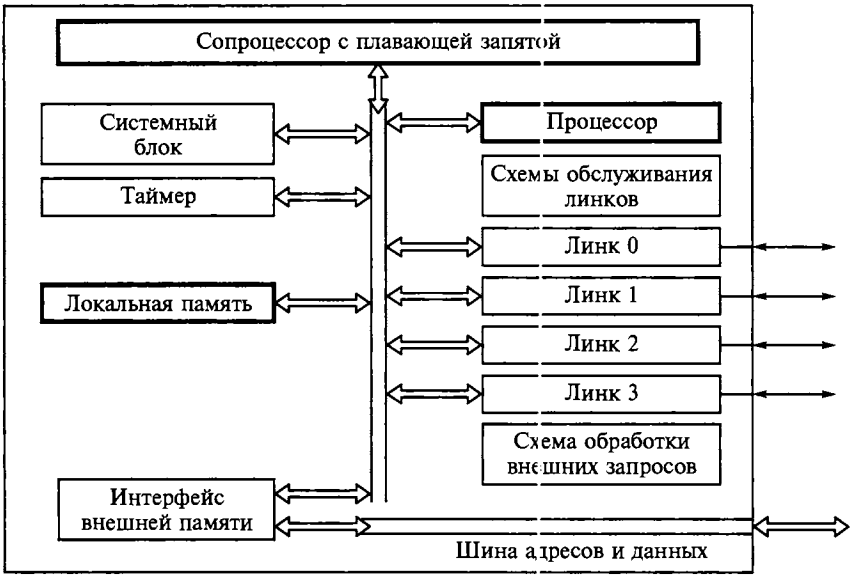


Рис. 8.4. Функциональная структура транспьютера IMS T805

системный блок. Имеются модификации транспьютера с тактовой частотой 20 и 30 МГц. В транспьютере с частотой 30 МГц пиковое быстродействие оценивается величинами: 30 MIPS и 4,3 MFLOPS. Среднее быстродействие при обращении к локальной памяти составляет 120 Мбайт/с, а к внешней — 40 Мбайт/с. Скорость передачи данных по линкам может программно настраиваться в диапазоне от 5 до 20 Мбит/с. Загрузка программ в транспьютер может выполняться из специального постоянного запоминающего устройства или по линку.

Транспьютер заключен в керамический корпус размером 30 × 30 мм^2 с 84-мя штырьковыми выводами. По назначению выводов он совместим с транспьютерами семейства T400 и с транспьютером IMS T800.

Координатный коммутатор IMS COO4 предназначается для построения ВС со сложной структурой (напомним, что транспьютер позволяет формировать ВС, структура которых не сложнее двумерных сетей). Координатный коммутатор IMS COO4 обеспечивает возможность в процессе решения задачи изменять структуру сети межтранспьютерных связей. Это полнодоступный коммутатор, позволяющий реализовать любые соединения своих 32 входов и 32 выходов. В нем любые коммутации реализуются без ослабления сигналов; поддерживаются две скорости передачи данных: 10 и 20 Мбит/с. Коммутатор реализован в виде отдельной интегральной схемы. Управление коммутатором в ВС осуществляется специально выделенным

транспьютером. При построении ВС с особо сложной структурой можно осуществлять каскадное соединение коммутаторов IMS COO4.

8.2.3. Организация памяти транспьютера

В процессе функционирования транспьютера осуществляются взаимодействия его процессора (и сопроцессора) с локальной внутренней и внешней памятью и с собственными регистрами [19, 20].

Адресация памяти. Для адресации памяти транспьютера используется одно слово, которое называется указателем. *Указатель (Pointer)* состоит из двух полей: *байтового сектора и адреса слова*. Байтовый сектор содержит столько битов, сколько необходимо для указания положения байта в слове. В 32-разрядных транспьютерах байтовый сектор занимает младшие 2 бит в указателе, а адрес слова — старшие 30 бит. Такая система адресации позволяет использовать одинаковые адресные команды вне зависимости от длины слова, что имеет большое значение для совместимости транспьютеров с различной разрядностью.

В системе команд транспьютера предусмотрены специальные инструкции (такие, как load local pointer, load non-local pointer, word subscript) для создания указателей и для манипуляций с ними при обращении к элементам массивов.

Значение указателя рассматривается как целое число со знаком (в диапазоне от минимального отрицательного до максимального положительного значений). Это позволяет использовать для указателей стандартные арифметические операции и функции сравнения.

Доступы к внутренней памяти (на кристалле) и к внешней памяти осуществляются идентично. Все адресное пространство линейно и однородно. Размер прямоадресуемой памяти составлял 4 Гбайт, или 2^{30} слов. Транспьютеры IMS T414 и IMS T800 имеют статическую внутреннюю память емкостью соответственно 2 К байт и 4 К байт.

Наибольшая производительность транспьютера достигается, если данные и программа размещены во внутренней памяти. Усредненные результаты выполнения некоторых типичных программ при различных вариантах расположения данных и программ во внутренней и внешней памяти транспьютера приведены в табл. 8.2. За единицу принято время выполнения программы для случая, когда данные и программа находятся во внутренней памяти транспьютера.

За один цикл внешней памяти можно или обратиться к одному слову данных, или провести выборку четырех команд. Поэтому лучшие результаты достигаются в случае размещения во внешней памяти не данных, а программ.

Таблица 8.2

Размещение		Время выполнения	
Программа	Данные	Программа с интенсивным обращением к данным	Вычислительная программа
Внутренняя память		1	1
Внешняя память	Внутренняя память	1,3	1,1
Внутренняя память	Внешняя память	1,5	1,2
Внешняя память		1,8	1,3

Регистры. Наличие быстродействующей внутренней памяти позволило разработчикам транспьютера ограничиться очень небольшим числом регистров. Это, в свою очередь, ускорило доступ к регистрам, упростило систему команд транспьютера, а также управляющую логику. При выполнении последовательных программ в процессоре используется шесть 32-разрядных регистров (рис. 8.5): регистры A, B и C составляют классический вычислительный стек; регистры W (Wordspace), I (next Instruction) и O (Operand) — специальные, причем:

W — регистр-указатель рабочей области; определяет место хранения локальных переменных выполняемого в текущий момент времени процесса;

I — регистр-указатель инструкций (или адреса следующей команды);

O — регистр операндов; используется при формировании операндов команд.

Вычислительный стек (из регистров A, B, C) используется практически всеми командами транспьютера. В нем размещаются операнды и результаты большинства арифметических и логических операций, а также команд планирования параллельных процессов и коммуникаций, в него записываются параметры при вызове процедур и т. п.

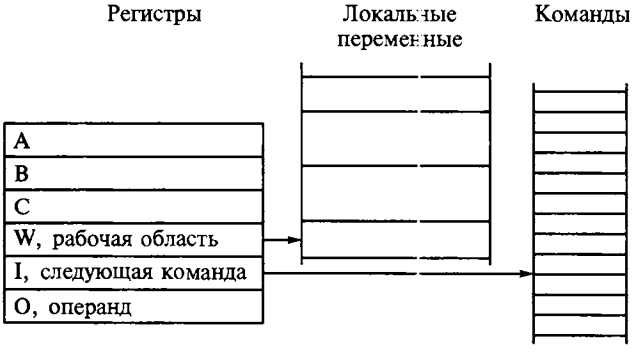


Рис. 8.5. Регистры общего назначения процессора транспьютера

Наличие вычислительного стека позволяет не вводить в структуру команд полей для явного задания регистров. Например, команда add («сложить») складывает значения, хранящиеся в регистрах A и B, помещает результат в A и копирует содержимое C в B. Поэтому большинство из выполняемых команд являются однобайтовыми (обычно 70...80 %). Статистически установлено, что при трех регистрах в стеке достигается оптимальный баланс между компактностью кодов и сложностью аппаратурной реализации.

В транспьютерах семейства T800 кроме вычислительного стека в процессоре имеется также два аналогичных стека в сопроцессоре с плавающей запятой. Каждый из них состоит из трех 64-разрядных регистров: AF, BF и CF. В любой момент времени для команд доступен только один стек. Второй стек используется для сохранения состояния сопроцессора (без записи в память) при прерывании процесса с низким приоритетом процессом с высоким приоритетом.

При выполнении транспьютерами параллельных процессов дополнительно используются еще четыре регистра, указывающих на начало и конец связных списков процессов с высокими и низкими приоритетами, ожидающих исполнения.

8.3. Система команд транспьютера

Как уже отмечалось в разд. 8.2.2, система команд транспьютера достаточно проста, а ее возможности намного шире, чем у набора команд RISC. Это достигнуто за счет комплекса архитектурных решений, заложенных в процессор [19, 20].

8.3.1. Формат команд

Все команды транспьютера имеют одинаковый формат. Длина каждой команды — 1 байт. Четыре старших разряда составляют код операции, четыре младших — поле данных (рис. 8.6). Такой формат команды позволяет определить 16 различных операций, которые кодируются значениями от 0 до F (в 16-ричном виде). Эти операции приведены в табл. 8.3.

Использование байтовых инструкций делает систему команд независимой от длины слова процессора, и поэтому оттранслированная программа

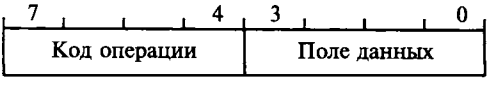


Рис. 8.6. Формат команды транспьютера

одинаково выполняется на транспьютерах с различными длинами слов (за исключением случаев, когда ее выполнение зависит от длины слова, например при переполнении). Простота команд позволила построить несложный механизм декодирования в процессоре, что привело к повышению быстродействия транспьютера и к уменьшению площади кристалла.

Таблица 8.3

Код	Команда	Число циклов	Назначение команды
0	**jump**	3	Переход
1	**load local pointer**	1	Загрузка указателя локальная
2	**prefix**	1	Префикс
3	**load non-local**	2	Загрузка нелокальная
4	**load constant**	1	Загрузка константы
5	**load non-local pointer**	1	Загрузка указателя нелокальная
6	**negative prefix**	1	Отрицательный префикс
7	**load local**	2	Загрузка локальная
8	**add constant**	1	Прибавление константы
9	**call**	7	Вызов
A	**conditional jump**	4	Условный переход (произошел)
		2	Условный переход (не произошел)
B	**adjust workspace**	1	Настройка рабочей области
C	**equals constant**	2	Приравнивание константы
D	**store local**	1	Запоминание локальное
E	**store non-local**	2	Запоминание нелокальное
F	**operate**	–	Выполнение

Короткие команды повышают также эффективность механизма выборки команд. За одно обращение к памяти происходит выборка сразу четырех команд. В процессоре имеется буфер команд длиной в два 32-разрядных слова (следовательно, рассчитанный на восемь команд). Поэтому редко возникают ситуации, когда процессору приходится ожидать выборки очередной команды из памяти. В случае размещения программы во внутренней памяти транспьютера это может происходить только после выполнения команды передачи управления. Время заполнения буфера команд незначительное — оно равно времени выборки из памяти только двух слов.

8.3.2. Команды с прямой адресацией

Из 16 команд, приведенных в табл. 8.3, 13 команд (кроме **prefix, negative prefix** и **operate**) используются наиболее часто. Содержимое поля данных является операндом команды и имеет значение от 0 до 15. Это со-

держимое может использоваться в качестве константы или смещения при адресации.

Команда **load constant** загружает значение константы в вычислительный стек процессора; **add constant** прибавляет константу к содержимому А-регистра.

Команды **load local**, **store local**, **load local pointer** позволяют пересылать данные из памяти в стек и обратно и загружать указатель в стек, используя указатель рабочей области в качестве базы, а значение поля данных команды в качестве смещения при адресации. К первым 16 ячейкам памяти можно обратиться при помощи одной байтовой команды.

Команды **load non-local**, **store non-local**, **load non-local pointer** выполняют функции, аналогичные предшествующим трем командам, за исключением того, что они используют в качестве базы при обращении к ячейкам памяти содержимое А-регистра стека.

Команды **jump** и **conditional jump** осуществляют безусловную и условную передачу управления команде, адрес которой определяется значением регистра операндов. Условная передача управления проводится, если содержимое А-регистра равно 0.

Команда **call** — вызов процедуры — приводит к нижеследующим действиям в транспьютере IMS T414. Содержимое вычислительного стека процессора и значение регистра-указателя инструкций копируются в рабочую область памяти. Обычно в вычислительном стеке процессора содержатся два первых параметра вызываемой процедуры и адрес ее рабочей области в памяти; остальные параметры должны быть вычислены и записаны в рабочую область заранее. После окончания выполнения процедуры команда **return** устанавливает первоначальное состояние стека и регистра-указателя инструкций. В транспьютере IMS T800 вызов процедуры происходит аналогично, необходимо только дополнительно сохранять состояние стека сопроцессора с плавающей запятой.

Рассмотрим два примера. Пусть x — локальная переменная, размещенная в памяти со смещением x (значение x ограничено от 0 до 15) относительно указателя рабочей области. Тогда строка Оккам-программы $x := 4$ будет иметь следующее представление в командах (сначала записывается операция, затем поле данных):

```
load constant 4
store local   x
```

Пусть в примере $x := y + 4$ переменная y является внешней по отношению к процессу, содержащему данный оператор. Адрес области памяти, где хранится значение переменной y, записан в локальной области памяти со смещением ***staticlink*** относительно указателя рабочей области. Поэтому

сначала проводится загрузка локальная в стек указателя на внешнюю область памяти, а затем загрузка нелокальная переменной *y*, расположенной со смещением *y* в первых 16 ячейках этой области памяти. Полная последовательность команд имеет следующий вид:

load local	***staticlink***
load non-local	*y*
add constant	4
store local	*x*

8.3.3. Префиксные команды

Известно, что поле данных команды (см. рис. 8.6) занимает четыре бита, следовательно, операнд ограничен значениями от 0 до 15. Для получения бо́льших значений операнда используются префиксные команды: **prefix** и **negative prefix** (см. табл. 8.3). При выполнении любой команды транспьютера 4 бит поля данных копируются в четыре младших разряда регистра операндов, содержимое которого используется в качестве операнда команды (рис. 8.7). Кроме префиксных все команды завершаются очисткой этого регистра.

Рис. 8.7. Формирование операндов

Команда **prefix** загружает свои 4 бит поля данных в регистр операндов, а затем сдвигает его на четыре разряда. Команда **negative prefix** действует аналогично с той лишь разницей, что перед сдвигом она инвертирует содержимое регистра операндов. Это позволяет легко задать отрицательные значения операнда (отрицательные числа в транспьютере представляются в дополнительном коде). Выполняя последовательность префиксных команд, можно получить операнд любой длины вплоть до разрядности регистра операндов.

В следующем примере записана последовательность команд для загрузки в А-регистр вычислительного стека 16-ричной константы # 754 и приведено содержимое регистра операндов и А-регистра после выполнения каждой команды:

		О-регистр	А-регистр
prefix	# 7	# 7	?
prefix	# 5	# 75	?
load constant	# 4	0	# 754

8.3.4. Команды с косвенной адресацией

При выполнении команды **operate** ее операнд интерпретируется как код операции над данными в вычислительном стеке. Например, если в результате такой интерпретации выявляется операция **add,** то выполняется суммирование данных из регистров A и B, размещение результата в A-регистре и копирование содержимого C в регистре B.

Команда транспьютера **operate** не содержит явно заданных адресов операндов и поэтому называется командой с косвенной адресацией. Префиксные команды можно использовать для расширения системы команд транспьютера, точнее: границ возможных значений операндов для команды **operate** (это делается так же, как для других команд).

Система команд транспьютера построена так, что наиболее часто встречающиеся операции с косвенной адресацией кодируются в 1 байт. Это — некоторые арифметические и логические операции, а также операции сравнения, ввода-вывода, управления и планирования параллельных процессов. Менее часто встречающиеся операции кодируются в 2 байта с использованием одной префиксной команды (число команд транспьютера значительно меньше 512).

Статистика, полученная из большого числа программ, показывает, что 70...80 % выполняемых инструкций кодируется в 1 байт, причем больше половины из них — одноцикловые.

Рассмотрим, например, вычисление выражения

$$x := (v - w) * (y + \#\,24)$$

Пусть локальные переменные x, y, v, w хранятся в первых 16 ячейках относительно указателя рабочей области памяти со смещениями x, y, v, w. Последовательность команд для вычисления этого выражения имеет следующий вид:

		Число байтов	Число циклов
load local	*v*	1	2
load local	*w*	1	2
operate	**subtract**	1	1
load local	*y*	1	2
prefix	# 2	1	1
add constant	# 4	1	1
operate	**multiply**	2	38
store local	*x*	1	1

Операция **operate multiply** кодируется в 2 байта (# 25 и # F3), так как код **multiply** (операнд для **operate**) равен # 53. Поэтому фактически происходит выполнение двух команд — **prefix** # 5 и **operate** # 3.

Если размер вычислительного стека является недостаточным для вычисления выражения, компилятор создает промежуточные переменные в локальной области памяти. Однако на практике такие случаи встречаются редко.

В дальнейшем слово **operate** в командах будет опускаться. Любая команда транспьютера, кроме перечисленных в табл. 8.3, использует в качестве своего кода операции операнд инструкции # F — **operate**. Из всех команд транспьютера только 16 кодируется в 1 байт, остальные — в 2 байта (используя для формирования операнда одну префиксную команду). Для изучения полной системы команд транспьютера читателю рекомендуется обратиться к специальным руководствам [19, 20].

8.3.5. Команды для операций с плавающей запятой

Система команд транспьютера IMS T800 включает в себя все команды IMS T414, а также имеет дополнительные инструкции для загрузки операндов из памяти в вычислительный стек сопроцессора и записи из этого стека в память, арифметические операции и операции сравнения над операндами с плавающей запятой. В набор команд транспьютера IMS T800 входит также и ряд сложных инструкций, являющихся комбинациями уже отмеченных команд. Кроме того, в IMS T800 реализованы и команды для работы с цветной графикой, распознавания образов, обработки кодов исправления ошибок.

Необходимо отметить, что транспьютер IMS T414 имеет микропрограммную поддержку для выполнения операций с плавающей запятой. Дальнейшее изложение будет сконцентрировано только на системе команд.

В IMS T800 передача операндов между стеком сопроцессора и памятью осуществляется командами загрузки и записи операндов с плавающей запятой. Существует две группы таких команд — одна для 32-разрядных операндов, другая для 64-разрядных. Ниже будут рассмотрены команды для операндов двойной длины; команды для операндов с одинарной длиной идентичные, необходимо только заменить слова **double** на слово **single**.

Адрес операнда с плавающей запятой предварительно вычисляется в стеке процессора, а затем операнд, расположенный по этому адресу в памяти, загружается в стек сопроцессора. Добавлены две команды для более эффективного доступа к словам с двойной длиной. Одна из них — **word subscript double** — используется при выборке элементов массивов, другая — **duplicate** — необходима, когда процессор работает с адресами старшего и младшего слов объектов двойной длины.

Операнды в вычислительном стеке тегируются. Тег определяет длину операнда; он устанавливается во время загрузки операнда или его вычисления. Наличие тега позволяет уменьшить число команд для операций с плавающей запятой. Поэтому, например, не требуется иметь одновременно команды **floating add single** и **floating add double** — достаточно одной команды **floating add**.

Две команды загружают операнды двойной длины в вычислительный стек сопроцессора — **floating load non-local double** и **floating load indexed double**. Первая из них загружает в AF-регистр операнд, адрес которого в памяти задается содержимым A-регистра стека процессора. Вторая команда действует также как последовательность двух инструкций — **word subscript double** и **floating load non-local double**. Операнд загружается в AF-регистр, используя содержимое B-регистра как смещение (в словах двойной длины), а содержимое A-регистра как базу при адресации. На рис. 8.8 показано действие команды **floating load indexed double**.

Введение команды загрузки с индексацией, заменяющей последовательность двух более простых команд, позволяет во многих случаях существенно уменьшить размер откомпилированной программы. Инструкция **floating load indexed double** кодируется в 2 байта, в то время как последовательность из двух команд требует 4 байта. Эта последовательность встречалась бы в оттранслированном коде для каждого обращения к элементам массива.

В системе команд IMS T800 присутствует инструкция **floating store double**, которая используется для записи операндов двойной длины из

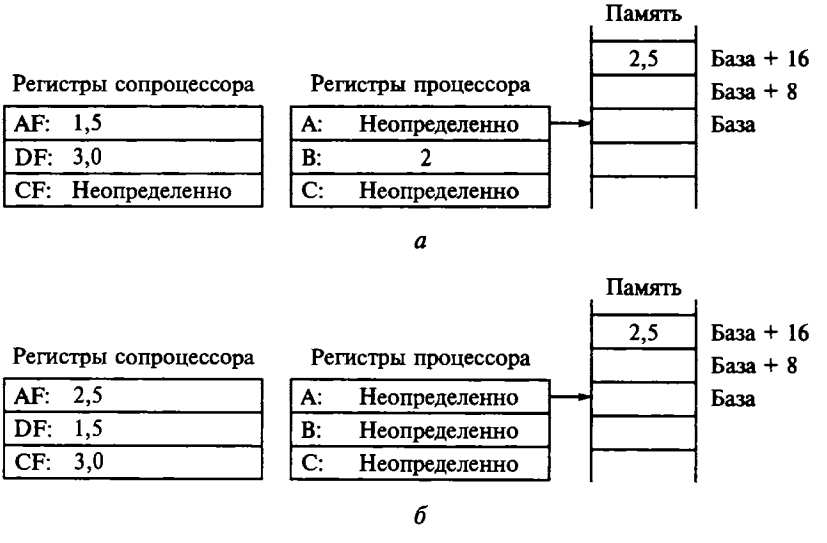


Рис. 8.8. Действие команды **floating load indexed double:**

а — перед выполнением; *б* — после выполнения

AF-регистра в память. Команда записи с индексацией отсутствует, так как в любой программе число операций записи в память меньше числа операций загрузки.

Команды сложения, вычитания, умножения и деления с плавающей запятой используют в качестве операндов содержимое AF- и BF-регистров, размещают результат в AF и копируют содержимое CF в BF. Команды **floating greater then** и **floating equality** сравнивают содержимое AF- и BF-регистров и загружают результат (логическое значение) в A-регистр стека процессора.

В следующем примере приведен фрагмент Оккам-программы, в котором булева переменная convergated указывает, больше ли значение переменной abs. error величины epsilon или нет:

```
BOOL convergated:
REAL32 abs. error, epsilon:
SEQ

        convergated: = abs. error < epsilon
```

Откомпилированный код для этого фрагмента имеет следующий вид:

load local pointer *epsilon*	— адрес epsilon
floating load non-local single	— загрузка AF сопроцессора
load local pointer *abs. error*	— адрес abs. error
floating load non-local single	— загрузка BF сопроцессора
floating greater then	— результат в стеке процессора
store local *convergated*	— запись convergated в A процессора

Существует четыре команды, заменяющие последовательности из двух инструкций: загрузки операнда и операции сложения или умножения. Это команды **floating load non-local and add double** и **floating load non-local and muitiply double** и две аналогичные инструкции для операндов одинарной (**single**) длины.

8.3.6. Параллельное функционирование процессора и сопроцессора

В транспьютере IMS T800 процессор может одновременно работать с сопроцессором. Это позволяет производить вычисления адресов в процессоре во время выполнения сопроцессором операций с плавающей запятой, следовательно, приводит к увеличению производительности транспьютера.

Особенно большой эффект от параллельной работы процессора и сопроцессора достигается при выборе элементов многомерных массивов. При выполнении адресных вычислений обычно используется команда быстрого умножения без контроля над переполнением **product**.

Например, рассмотрим выполнение следующего фрагмента Оккам-программы:

```
[20] [20] REAL64 a:
SEQ
        . . .
        b: = a[i][ j] + c * d
```

В этом фрагменте выборка a[i][j] включает в себя вычисление смещения этого фрагмента относительно базы массива A — *A.link*. Перед загрузкой a[i][j] выполняются следующие команды:

load local	*i*	— загрузка *i* в стек процессора
prefix	# 1	
load constant	# 4	— загрузка 20 в стек процессора
product		— умножение *i* на 20
load local	*j*	— загрузка *j* в стек процессора
add		— сложение *j* и *i* * 20
load local	*A.link*	— загрузка *A.link* в стек процессора
word subscript double		— формирование адреса a[i][j]

В данном случае команда **product** выполняется за восемь циклов. Приведенный фрагмент программы выполняется за 18 циклов. Следующая команда — **floating load non-local double** — загружает a[i][j] в стек сопроцессора. Поскольку при выполнении умножения 64-разрядных слов с плавающей запятой требуется 21 цикл, никакой задержки из-за вычисления адреса элемента массива не происходит.

8.4. Параллельная обработка и коммуникации транспьютеров

8.4.1. Обработка параллельных процессов

В языке Оккам параллельные процессы создаются динамически и взаимодействуют между собой через каналы. Параллельные процессы могут выполняться как на одном транспьютере, так и на различных процессорах в мультитранспьютерной системе. Архитектура транспьютера предусматривает эффективные средства для поддержки обоих видов параллельной обработки. Рассмотрим обработку параллельных процессов на одном транспьютере.

Транспьютер может одновременно обрабатывать любое число параллельных процессов. Он имеет специальный *планировщик*, который выполняет распределение времени между процессами. В любой момент времени параллельные процессы делятся на два класса — *активные процессы* (выпол-

няются или готовы к выполнению) и *пассивные процессы* (ожидают ввода-вывода или определенного времени).

Активные процессы, ожидающие выполнения, помещаются в планировочный список. Последний является списком рабочих областей этих активных процессов в памяти и задается значениями двух регистров, один из которых указывает на первый процесс в списке, а другой — на последний. Состояние процесса, готового к выполнению, сохраняется в его рабочей области. Состояние определяется двумя словами — текущим значением указателя инструкций и указателем рабочей области следующего процесса в планировочном списке. На рис. 8.9 изображена обработка параллельных активных процессов, причем процесс S выполняется, а процессы P, Q и R ожидают выполнения в планировочном списке.

Команда транспьютера **start process** создает новый активный процесс, добавляя его в конец планировочного списка. Перед выполнением этой команды в А-регистр вычислительного стека должен быть загружен указатель инструкций этого процесса, а в В-регистр — указатель его рабочей области. Команда **start process** позволяет новому параллельному процессу выполняться вместе с другими процессами, которые транспьютер обрабатывает в данное время.

Команда **end process** завершает текущий процесс, убирая его из планировочного списка. В языке Оккам конструкция **PAR** может закончиться только тогда, когда завершатся все ее компоненты — параллельные процессы. Каждая команда **start process** увеличивает их число, а **end process** уменьшает. В транспьютере предусмотрен специальный механизм учета числа незавершившихся компонентов данной параллельной конструкции (необходимо учитывать как активные, так и неактивные процессы).

При обработке параллельных процессов на самом деле присутствует

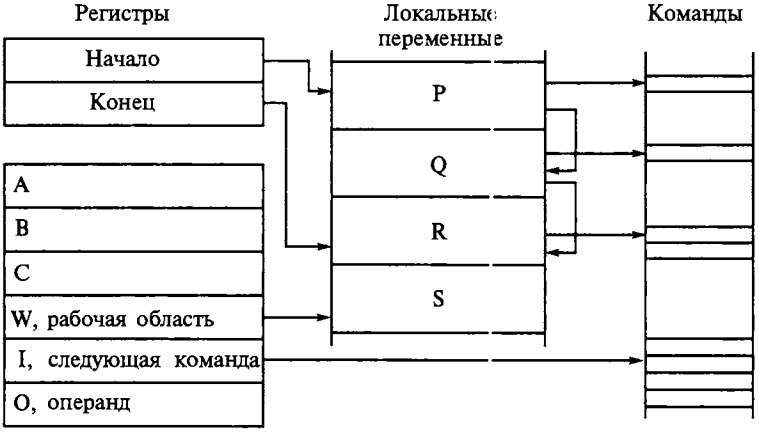


Рис. 8.9. Обработка параллельных активных процессов

не один, а два планировочных списка — список высокого приоритета и список низкого приоритета. Процессы с низким приоритетом могут выполняться только тогда, когда список высокого приоритета пуст. Процессы с высоким приоритетом обычно вводятся для обеспечения отклика системы на них в реальном времени. Если список высокого приоритета пуст и процесс с высоким приоритетом становится активным, он прерывает выполнение текущего процесса (с низким приоритетом) в конце текущей команды или в ее промежуточной точке. Состояние прерываемого процесса сохраняется в фиксированной области памяти. Для осуществления подобного прерывания требуется в среднем 1 мкс, в наихудшем случае — 3,7 мкс. Напомним, что сопроцессор транспьютера IMS T800 имеет два вычислительных стека, которые используются для процессов с разными приоритетами. Наличие двух стеков устраняет необходимость копирования состояния сопроцессора в памяти, что существенно уменьшает время прерывания, а следовательно, и время отклика системы. Если готовы к выполнению несколько процессов с высоким приоритетом, они помещаются в очередь в списке высокого приоритета. Для ускорения отклика системы необходимо, чтобы процессы с высоким приоритетом были как можно короче.

Текущий активный процесс низкого приоритета выполняется до тех пор, пока не произойдет одно из следующих событий: 1) данный процесс завершится (командой **end process**); 2) он станет неактивным; 3) появится готовый к выполнению процесс с высоким приоритетом; 4) время выполнения процесса превысит определенную величину — период синхронизации. В последнем случае начинает выполняться следующий процесс из планировочного списка, а текущий процесс помещается в конец этого списка. Использование такого механизма гарантирует, что ни один активный процесс не будет ожидать выполнения неограниченно долго. Один процесс низкого приоритета может непрерывно занимать процессор в среднем не более 750 мкс. Описанное переключение процессов может произойти только после выполнения команд **jump** (переход) и **end loop** (конец цикла), когда вычислительные стеки процессора и сопроцессора не содержат полезной информации и количество сохраняемой информации минимально.

По отношению к процессам с высоким приоритетом подобный механизм переключения с использованием периодов синхронизации не применяется, поэтому они не должны выполняться неограниченно долго.

Весь механизм выполнения параллельных процессов на одном транспьютере спроектирован так, чтобы минимизировать накладные расходы. В параллельных программах большинство процессов находится в состоянии ожидания коммуникаций, и поэтому они не включены в планировочный список. Количество регистров очень небольшое, что позволяет минимизировать объем сохраняемой информации при прекращении выполнения процесса.

8.4.2. Коммуникации

Коммуникации между параллельными процессами осуществляются через каналы. Канал между двумя процессами, выполняющимися на одном транспьютере, задается одним словом в памяти, а обмен информацией осуществляется посредством пересылок между рабочими областями этих процессов в памяти транспьютера. Канал между процессами, которые выполняются на разных транспьютерах, реализуется аппаратно в виде двухточечной линии связи между этими транспьютерами.

Система команд транспьютера имеет ряд инструкций, которые поддерживают пересылку сообщений между параллельными процессами. Наиболее важными являются **input message** (ввод сообщения) и **output message** (вывод сообщения). Есть также команды **output word** (вывод слова) и **output byte** (вывод байта), которые можно считать частными случаями команды **output message**.

Команды **input message** и **output message** используют адрес канала для определения, является ли он внутренним или внешним. Несмотря на принципиальные различия в реализации внутренних и внешних каналов, все команды для коммуникаций одинаковые. Это позволяет осуществлять компиляцию процедур безотносительно к способу реализации каналов, а следовательно, и к конфигурации ВС.

В языке Оккам коммуникации происходят только тогда, когда и принимающий, и передающий процессы готовы к обмену данными. Поэтому процесс, который первым достиг готовности, должен ожидать, когда и второй процесс также будет готов к обмену информацией.

Команды ввода или вывода сообщений реализуются следующим образом. В вычислительный стек процессора загружаются указатель начала сообщения, адрес канала и количество байтов в сообщении, а затем выполняется команда **input message** или **output message**.

8.4.3. Пересылка данных по внутреннему каналу

Внутренний канал реализуется одним словом в памяти транспьютера. Во время инициализации в канал записывается специальное значение **empty** (не занято). Значение **empty** выбрано так, что оно не может отождествляться ни с одним процессом (с указателем его рабочей области). Перед началом передачи данных между процессами через внутренний канал команды пересылки проверяют содержимое канала.

Проиллюстрируем порядок работы внутреннего канала на следующем примере. Пусть процесс P готов к выводу данных в незанятый канал C (рис. 8.10, *а*). Вычислительный стек процессора содержит указатель на пересылаемые данные, адрес канала C и количество пересылаемых байтов.

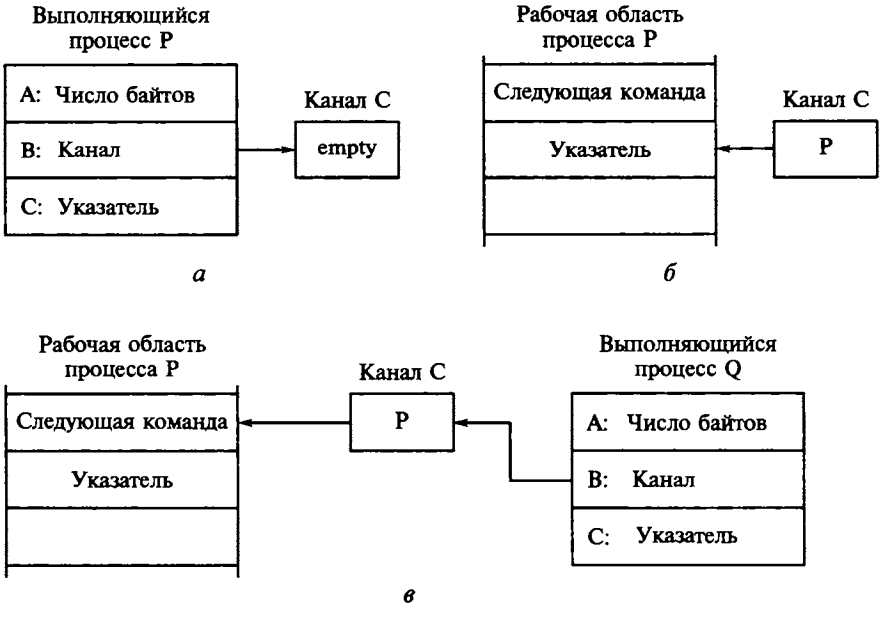


Рис. 8.10. Передача данных по внутреннему каналу:

а — процесс P готов к выводу данных; *б* — процесс P ожидает ввода данных другим процессом; *в* — обмен данными между процессами P и Q

Во время выполнения команды **output message** в канал C записывается указатель рабочей области процесса P, а указатель на пересылаемые данные сохраняется в рабочей области P. Процесс P прерывает свое выполнение и становится неактивным, ожидая ввода данных другим процессом (рис. 8.10, *б*). Процессор начинает выполнение следующего процесса из планировочного списка.

Канал C и процесс P остаются в этом состоянии до того момента, когда второй процесс Q выполнит команду **input message** для этого канала (рис. 8.10, *в*). После чего пересылаемые данные копируются в памяти, процесс P становится активным и добавляется в конец планировочного списка, а в канал C снова записывается значение **empty**. Механизм организации обменов не зависит от того, какой процесс (передающий или принимающий) первым достиг готовности.

8.4.4. Пересылка данных по внешнему каналу

Команда пересылки данных по внешнему каналу направляет автономному канальному интерфейсу задание на передачу сообщения и приостанавливает выполнение процесса. После окончания передачи сообщения каналь-

ный интерфейс помещает этот процесс в планировочный список. При передаче данных по внутреннему каналу процесс, который последним достиг готовности к обмену, остается активным. Во время обменов по внешнему каналу оба процесса становятся неактивными. Это позволяет транспьютеру продолжать обработку других процессов во время пересылки данных через более медленные внешние автономные каналы.

Каждый канальный интерфейс использует три своих регистра, в которые загружаются указатель на рабочую область процесса, указатель на пересылаемые данные и количество пересылаемых байтов.

8.4.5. Каналы межтранспьютерной связи

Физический канал связи (Link) между двумя транспьютерами реализует два Оккам-канала (Channels), по одному в каждом направлении. Канал связи между транспьютерами создается путем соединения их канальных интерфейсов двумя однонаправленными сигнальными проводниками. Каждый из проводников предназначен для передачи данных и управляющих сигналов.

Для пересылки данных используется простой протокол, который обеспечивает передачу произвольной последовательности байтов. Протокол связи универсальный, он не зависит от разрядности транспьютера. Последнее позволяет соединять друг с другом транспьютеры разных типов.

Сообщения передаются в виде отдельных пакетов (см. рис. 8.3), каждый из которых содержит 1 байт данных. Следовательно, наличие буфера в 1 байт в транспьютере является достаточным для исключения временных потерь при межтранспьютерных пересылках. После отправления пакета данных транспьютер ожидает получения пакета подтверждения от принимающего транспьютера. Пакет подтверждения показывает, что процесс-получатель готов принять этот байт и что канал принимающего транспьютера может начать прием следующего байта.

Протокол передачи пакета данных позволяет отсылать пакет подтверждения, как только транспьютер идентифицировал начало пакета данных. Подтверждение может быть получено передающим транспьютером еще до завершения передачи всего пакета данных, и поэтому пересылка данных может быть непрерывной, т. е. без пауз между пакетами.

Каналы могут передавать данные с частотой 5, 10 и 20 Мбит/с, причем частоту можно задавать для каждого канала в отдельности. Передача данных производится синхронно; для синхронизации используются внутренние тактовые импульсы, которые вырабатываются на основе внешних тактовых сигналов (использующих частоту 5 МГц).

В транспьютере IMS T414 не реализовано описанное временное перекрытие при передаче пакетов данных и пакетов подтверждения.

8.5. Архитектура транспьютера IMS T9000

Модель транспьютера IMS T9000 характеризуется развитой архитектурой, сочетающей в себе достоинства архитектур транспьютеров семейств T200, T400, T800 и высокопроизводительных микропроцессоров. Транспьютер IMS T9000 ориентирован на реализацию операций и с фиксированной, и с плавающей запятой соответственно над 32- и 64-разрядными числами. При тактовой частоте 50 МГц пиковое быстродействие транспьютера IMS T9000 оценивается значениями: 200 MIPS и 25 MFLOPS; скорость передачи информации по линку составляет 100 Мбит/с. Количество транзисторов в кристалле IMS T9000 равно 3,2 млн, а число выводов с корпуса — 208. Кристалл транспьютера IMS T9000 основывается на усовершенствованной КМОП-технологии (CMOS technology, комплементарная металл-окисел-полупроводник технология). Программное обеспечение данного транспьютера совместимо с предшествующими генерациями для транспьютеров семейств T200, T400 и T800. В 1999 г. был налажен выпуск транспьютеров IMS T9000 с тактовой частотой 25 МГц.

Системы команд IMS T9000 и транспьютеров семейств T200, T400 и T800 полностью совместимы. Увеличение производительности в транспьютере IMS T9000 по сравнению с IMS T805 достигнуто и за счет совершенствования технологии, и главное — новых архитектурных решений.

8.5.1. Транспьютер IMS T9000

Транспьютер IMS T9000 имеет суперскалярную RISC-архитектуру, в нем достигнут баланс между вычислительными и коммуникационными возможностями. Суперскалярность архитектуры транспьютера как микропроцессора означает, что его аппаратура обеспечивает загрузку параллельно работающих функциональных устройств при использовании традиционных последовательных программ (команды не содержат никаких указаний на параллельную обработку внутри процессора).

В состав транспьютера IMS T9000 входят (рис. 8.11): процессорный конвейер (Processor Pipeline), кэш-память команд и данных (Instruction and Data Cache), программируемый интерфейс внешней памяти (Programmable Memory Interface), четыре рабочих линка (Links 0-3), схемы обработки внешних запросов (Events 0-3), процессор виртуальных каналов (Virtual Channel Processor), системный блок (System Services), таймеры (Timers), два управляющих линка (Control Links, Clink 0 и Clink 1) и 32-разрядная внутренняя шина (Common Bus).

Процессорный конвейер включает в себя 32-разрядный процессор (ALU — Arithmetic-Logical Unit) для выполнения операций над целыми чис-

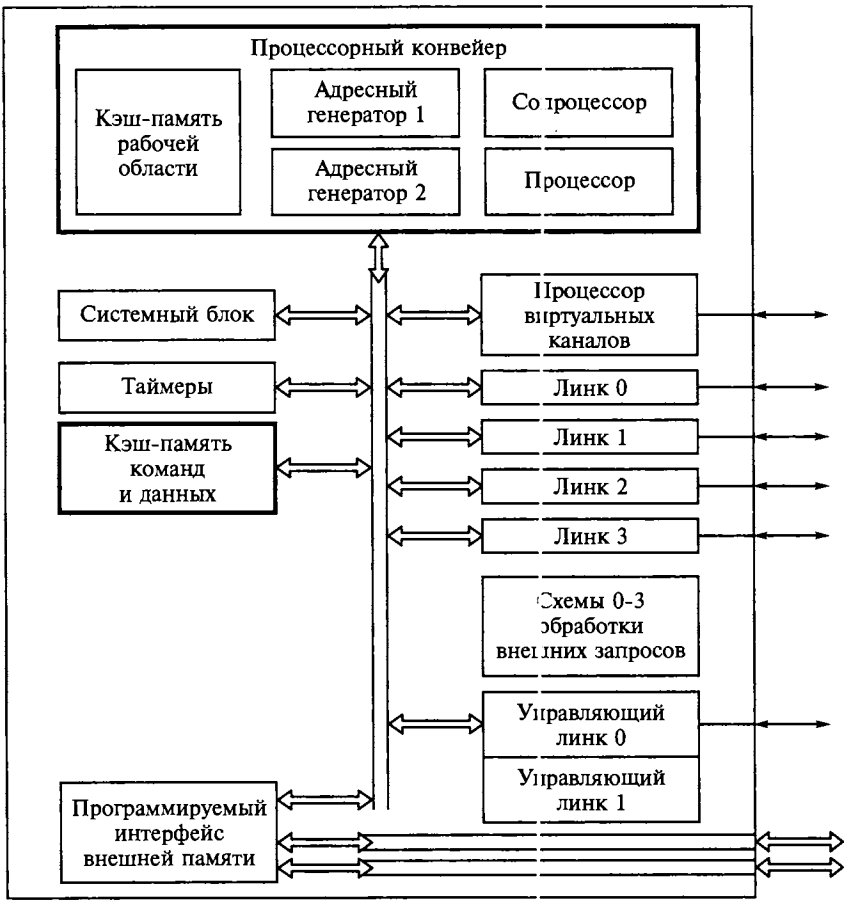


Рис. 8.11. Функциональная структура транспьютера IMS T9000

лами, 64-разрядный сопроцессор (FPU — Floating Point Unit) для реализации операций над числами с плавающей запятой, два адресных генератора (Address Generators), кэш-память рабочей области (Workspace Cache), управляющее устройство выборки и группировки команд и буфер команд. Кэш-память рабочей области обеспечивает быстрый доступ к локальным переменным, она, по сути, выполняет роль блока регистров общего назначения. Допускается параллельная работа процессора и сопроцессора.

Одной из архитектурных особенностей процессорного конвейера является аппаратурная группировка команд, повышающая загрузку параллельно функционирующих устройств. Максимальное число одновременно выполняемых команд в группе равно 8. Группировка команд осуществляется управляющим устройством выборки и группировки команд и буфером

команд. Увеличение быстродействия в IMS T9000 достигнуто также и за счет сокращения числа тактов, необходимых для выполнения ряда арифметических и логических операций (по некоторым операциям получено 10-кратное сокращение по сравнению с числом тактов IMS T805).

Кэш-память команд и данных имеет емкость 16 К байт. Эта память рассчитана на работу в одном из трех режимов:

- как быстродействующее внутреннее ОЗУ при решении небольших задач (в этом случае внешняя память может быть не использована);
- как единая кэш-память для обеспечения быстрого доступа к наиболее часто используемым данным или фрагментам программы;
- как два блока емкостью по 8 К байт: ОЗУ и кэш-память.

Программируемый интерфейс внешней памяти обеспечивает возможность подключения к транспьютеру внешней памяти различных видов и емкости. Данный интерфейс позволяет работать транспьютеру с 8-, 16-, 32- и 64-разрядной статической и 32- и 64-разрядной динамической памятью. При этом вся внешняя память может быть разделена на четыре области с различными параметрами.

Процессор виртуальных каналов позволяет мультиплексировать линки, т. е. организовать так называемые виртуальные каналы. Это дает возможность использовать один и тот же линк одновременно несколькими процессами. Следовательно, данный аппаратурный механизм дает возможность ввести столько программных каналов, сколько их требуется для решения задачи.

Таким образом, процессор виртуальных каналов позволяет выполнять по одному физическому линку обмен сообщениями между произвольным числом пар процессов, протекающих в различных транспьютерах. Каждое сообщение, передаваемое от процесса-отправителя к процессу-получателю, делится процессором на пакеты. Пакет содержит 32 байта данных (последний пакет может быть размером от 1 до 32 байт), заголовок и признак конца пакета (а в последнем — признак конца сообщения). При получении пакета процессор виртуальных каналов принимающего транспьютера передает подтверждение в виде «пустого» пакета (содержащего только заголовок и признак конца пакета). Маршрутизация пакетов и их сборка в сообщение осуществляются процессорами виртуальных каналов на основе информации, содержащейся в заголовках пакетов. Следовательно, обмен данными между процессами выглядит так же, как и для транспьютеров предшествующих поколений, что способствует преемственности программного обеспечения.

Механизм маршрутизации сообщений в межтранспьютерной сети делает неразличимыми обмены информацией в пределах как одного транспьютера, так и мультитранспьютерной системы. Последнее существенно упрощает разработку параллельных программ для мультитранспьютерных систем и повышает их эффективность (так как удается избежать усложне-

ний в текстах программ, т. е. дополнительных расходов на описание процессов маршрутизации сообщений).

Линки транспьютера IMS T9000 рассчитаны на передачу данных по каждому из них со скоростью 100 Мбит/с. Следовательно, пропускная способность линков при обмене информацией между транспьютерами в дуплексном режиме оценивается величиной 8 × 100 Мбит/с.

Управляющие линки позволяют создать единую систему управления транспьютерами. Эти линки предназначены для инициализации и управления работой транспьютеров. Они обеспечивают двунаправленный обмен информацией между транспьютерами.

8.5.2. Коммутатор IMS C104

Программируемый коммутатор или маршрутизатор IMS C104 — элемент, позволяющий формировать мультитранспьютерные системы со сложной структурой. Он дает возможность реализовать механизм пакетной коммутации сообщений и организовать информационное взаимодействие между транспьютерами. Композиция из коммутатора IMS C104 и транспьютера IMS T9000 увеличивает число физических линков у последнего элемента.

Элемент IMS C104 — полнодоступный коммутатор размером 32 × 32, допускающий соединение любого из своих линков с любым другим. Задержка при любом переключении коммутатора не превышает 700 нс. Следовательно, коммутатор IMS C104 способен осуществить передачу пакетов сообщения с любого из своих 32 входов на любой из 32 выходов (в соответствии с их заголовками). При передаче пакетов коммутатор обеспечивает пропускную способность 640 Мбайт/с.

Пример двумерной мультитранспьютерной системы на базе транспьютеров IMS T9000 и коммутаторов IMS C104 представлен на рис. 8.12.

8.5.3. Язык Оккам 3

Язык Оккам 3 (Occam 3) — версия языка Оккам; она ориентирована на применение в системах, использующих транспьютеры IMS T9000 и коммутаторы C104.

При использовании этого языка нет необходимости устанавливать соответствие между программными и физическими каналами в системе. Отпадает также необходимость в предварительном размещении процессов по отдельным транспьютерам, эту работу выполняет компилятор.

Язык Оккам 3 обеспечивает независимость параллельной программы от структуры (топологии) мультитранспьютерной вычислительной системы (программу можно переносить с одной системы на другую без существенных изменений).

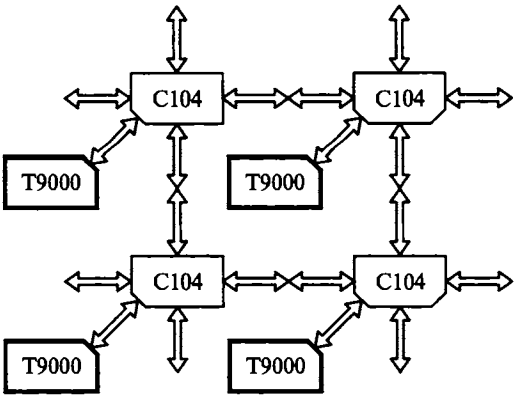


Рис. 8.12. Фрагмент двумерной транспьютерной системы

Для транспьютера IMS T9000 имеется набор средств для разработки и отладки программ (который почти аналогичен соответствующему набору для транспьютеров семейства T800).

8.6. Анализ транспьютерных технологий

Понятие «транспьютерные технологии» охватывает не только микроэлектронные технологии в производстве ЭМ — транспьютеров, но и архитектурные и функциональные решения по формированию систем как коллективов транспьютеров, а также методы организации параллельных вычислений и программирования. Транспьютерные технологии широко внедрялись в 1980-х и 1990-х годах. Популярность таких технологий объяснялась тем, что транспьютер в то время был единственным системным элементом, который был способен реализовать как коммуникационные, так и вычислительные функции.

Транспьютерные технологии в конце XX в. были восприняты многими западными и отечественными организациями, занятыми созданием высокопроизводительных средств обработки информации. Они позволяли создавать технико-экономически эффективные масштабируемые суперВС того времени.

Достижением организаций, занимавшихся созданием и развитием средств транспьютерных технологий, следует признать то, что они уже в 1990-х годах вплотную подошли к рубежу, начиная с которого суперВС могли бы быть реализованы на большемасштабных полупроводниковых пластинах (в виде System-on-Chip).

Опыт создания транспьютерных ВС с массовым параллелизмом безусловно будет востребован в XXI столетии.

9. НАДЕЖНОСТЬ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Понять архитектуру ВС, оценить их функциональный потенциал невозможно без анализа эффективности. Специфика современных ВС не позволяет решить проблемы их эффективности путем прямой трансформации методов существующей теории эффективности систем и ЭВМ. В ходе исследований по проблемам эффективности ВС потребовалось:

а) ввести показатели качества функционирования ВС, которые устанавливали бы взаимосвязь между производительностью, надежностью, живучестью и технико-экономической эффективностью;

б) создать нетрудоемкий и адекватный математический аппарат для расчета этих показателей; провести численный анализ качества функционирования ВС;

в) разработать технологию экспресс-анализа эффективности функционирования ВС.

В данной главе изучается надежность (Reliability) распределенных ВС. Под надежностью ВС будем понимать свойство системы сохранять заданный уровень производительности путем программной настройки ее структуры и программной организации функционального взаимодействия между ее ресурсами.

9.1. Производительность вычислительных систем

Содержание гл. 4–8 (а также [5, 6]) убеждает в том, что современными высокопроизводительными средствами обработки информации являются распределенные ВС (Distributed Computer Systems), системы с массовым параллелизмом (Massively Parallel Processing Systems). Число функционально-конструктивных элементов обработки информации (ЭМ или процессоров) в таких ВС уже сейчас составляет порядка 10^6. Это обстоятельство дает основание специалистам в области анализа эффективности (производительности, надежности, живучести и технико-экономической эффективности) средств обработки информации называть распределенные ВС большемасштабными (Large-Scalable Computer Systems).

Полнота воплощения принципов модели коллектива вычислителей определяет архитектурную гибкость ВС. Отмечалось, что современные рас-

пределенные ВС обладают свойством масштабируемости (Scalability), а архитектурно гибкие их представители характеризуются и программируемостью структуры (Structure Programmability).

В гл. 3 и 7 отмечалось, что ВС рассчитаны на работу в моно- и мультипрограммных режимах. В первом случае для решения любой задачи отводятся все ресурсы ВС, а во втором — лишь часть из них. Очевидно, что монопрограммный режим целесообразен при решении на ВС сложных или трудоемких задач (см. разд. 3.3.4). К последним относят задачи, для решения которых требуется выполнить достаточно большое количество операций. Например, для ВС, состоящей из N ЭМ, сложной можно считать задачу с числом операций порядка $N \cdot 10^l$, где $l \in \{1, 2, 3, ...\}$.

Пусть N — число ЭМ в распределенной ВС (с программируемой структурой). Будем говорить, что система находится в состоянии $k \in E_0^N$, $E_0^N = \{0, 1, 2, \ldots, N\}$, если в ней имеется k исправных ЭМ. Теоретически (на основе анализа параллельных алгоритмов) и экспериментально (путем реализации параллельных программ сложных задач на действующих ВС) показано, что производительность (Performance) системы, находящейся в состоянии $k \in E_0^N$, равна

$$\Omega(k) = A_k k \omega, \tag{9.1}$$

где ω — показатель производительности ЭМ; A_k — коэффициент. В качестве ω может быть использовано быстродействие по Гибсону, номинальное и среднее быстродействия машины (см. разд. 2.6.2).

Линейная зависимость производительности ВС от количества исправных ЭМ — следствие применения методики крупноблочного распараллеливания сложных задач (см. § 3.3). Суть методики состоит в однородном разделении задачи на крупные блоки — подзадачи, между которыми существует слабая связность. Говоря иначе, методика предписывает организацию таких однородных параллельных ветвей алгоритма решения сложной задачи, общее число операций в которых много больше числа операций обмена информацией между ветвями. Ясно, что это обеспечивает высокую эффективность реализации параллельных программ сложных задач и, следовательно, значение коэффициента A_k, близкое к единице. Практически (на основе обработки результатов решения сложных задач различных классов на ВС) установлено, что коэффициент A_k не менее единицы ($A_k \geqslant 1$). Это неравенство объясняет «парадокс параллелизма» (см. разд. 3.3.3) и является следствием «эффекта памяти» (в ВС сокращается количество обращений к внешним запоминающим устройствам, быстродействие которых меньше скорости передачи информации по кана-

лам связи между ЭМ) и применения новых методов решения сложных задач, не реализуемых на ЭВМ.

Формула (9.1) остается верной и при работе распределенных ВС в мультипрограммных режимах (при обработке наборов задач или обслуживании потоков задач, см. разд. 7.2.2). В самом деле, в таких системах в отличие от матричных ВС каждая ЭМ обладает своим устройством управления. Следовательно, в распределенных ВС имеется возможность одновременной реализации нескольких программ, причем каждой на своей части системы, т. е. на своей подсистеме. В пределе каждую ЭМ можно заставить решать свою задачу. Накладные расходы, связанные с мультипрограммной работой распределенных ВС, зависят от методов и алгоритмов организации функционирования. Созданные методы и алгоритмы (см., например [5]) эффективно реализуются на ЭВМ и системах (не требуют большой памяти и значительного компьютерного времени) и обеспечивают высокую загрузку технических ресурсов. Можно ожидать, что при мультипрограммной работе распределенной системы будет иметь место $A_k > 0{,}85$, $k \in E_0^N$

Таким образом, реальная производительность ВС будет не менее 85 % суммарной производительности ЭМ как для моно-, так и для мультипрограммных режимов. Заметим, что такой уровень достигнут в матричных системах (в частности, в ILLIAC-IV, см. § 5.2) только в режиме реализации одной параллельной программы на всех N элементарных процессорах. При выполнении программ, имеющих число ветвей $l < N$, в матричных ВС будет простаивать $(N - l)$ элементарных процессоров.

Раскрытию взаимосвязи между надежностью (безотказностью) и производительностью ВС и посвящена данная глава.

9.2. Вычислительные системы со структурной избыточностью

Вычислительные системы компонуются в общем случае из неабсолютно надежных ЭМ (см. § 2.8). Пусть λ — интенсивность потока отказов в любой из N машин (см. (2.12), (2.14)). Следовательно, величина λ^{-1} — среднее время безотказной работы одной ЭМ (или средняя наработка до отказа ЭМ).

Отказы, возникающие в ВС, устраняются при помощи процедуры восстановления. Последняя предусматривает и контроль функционирования ВС, и локализацию неисправных ЭМ, и их ремонт или замену на исправные машины, например из резерва. Будем считать, что эта процедура реализуется при помощи восстанавливающей системы, состоящей из m устройств, $1 \leqslant m \leqslant N$.

После отказа ЭМ либо поступает на обслуживание в любое свободное восстанавливающее устройство (ВУ), либо (если таковых нет) ставится в очередь на восстановление. Считается, что в каждый момент времени любое из m ВУ может быть либо свободным, либо занятым восстановлением не более одной ЭМ. Пусть μ — интенсивность восстановления (2.18) отказавших ЭМ одним ВУ. Из сказанного следует, что в системе допустимы переходы из состояния $k \in E_0^N$ как в состояние $k-1$ $(k \neq 0)$, так и в состояние $k+1$ $(k \neq N)$.

В теории потенциальной надежности ВС введены системы со структурной избыточностью, которые являются обобщением систем с резервом [5, 6]. Вычислительные системы со структурной избыточностью с позиций архитектуры и способов обработки информации не являются специальным классом систем в ряду: конвейерные, матричные, мультипроцессорные ВС, ВС с программируемой структурой и т. д. *Вычислительная система со структурной избыточностью* по существу представляет собой виртуальную ВС или, точнее, программно-настроенную конфигурацию, в которой:

1) выделены основная подсистема (вычислительное ядро) из n ЭМ и подсистемы, подчиненные основной и составляющие избыточность из $(N-n)$ машин $(n \neq 0,\ n \in E_0^N)$;

2) основная подсистема предназначена для решения сложных задач, представленных параллельными программами из n ветвей, а любая подчиненная подсистема — для решения фоновых задач;

3) функции отказавшей ЭМ основной подсистемы может взять на себя любая исправная ЭМ любой подчиненной подсистемы;

4) производительность (при изменении состояния $k = 0, 1, 2, \ldots, N$) подчиняется следующему закону:

$$\Omega(k) = A_n \Delta(k-n)\varphi(n,\ \omega), \tag{9.2}$$

где A_n — коэффициент;

$$\Delta(k-n) = \begin{cases} 1, & \text{если } k \geqslant n; \\ 0, & \text{если } k < n; \end{cases} \tag{9.3}$$

ω — производительность (точнее, один из показателей производительности) ЭМ; $\varphi(n, \omega)$ — неубывающая функция от n и ω (как правило, $\varphi(n, \omega) = n\omega$).

Очевидно, что для решения сложных задач требуются ВС с массовым параллелизмом и, следовательно, параллельные программы с большим числом ветвей. При этом можно считать, что $n \gg (N-n)$, и производительность основной подсистемы будет близка к суммарной производительности ВС.

Таким образом, ВС со структурной избыточностью выглядит для пользователей как (виртуальная) система из n ЭМ. Эта же ВС с позиций проектировщиков и эксплуатационников выглядит как система с высоким уровнем надежности, который достигается за счет $(N-n)$ избыточных машин. Величина $(N-n)$ структурной избыточности в таких системах изменяется программно, но для заданной сложной задачи (для выбранной области применения ВС) она постоянна. Значение n (и, следовательно, $N-n$) выбирается из требований обеспечения производительности и надежности ВС.

Параллельная программа решения задачи имеет фиксированный объем информационной избыточности, которая используется для того, чтобы исправная ЭМ структурной избыточности могла взять на себя функции отказавшей ЭМ основной подсистемы. Информационная и структурная избыточности не уменьшают времени решения задачи, а увеличивают надежность работы ВС в целом (как аппаратурно-программного комплекса).

9.3. Показатели надежности вычислительных систем

Для количественного анализа работы ВС используется набор показателей надежности. Главное требование, которое предъявляется к набору, это обеспечение полноты характеристики качества функционирования ВС. Следовательно, должны быть показатели, характеризующие производительность ВС и в текущий момент времени, и на промежутке времени, а также показатели, позволяющие оценить способность системы к восстановлению заданного уровня производительности после отказа ее отдельных машин. Среди показателей надежности ВС, безусловно, должны быть такие, которые характеризуют поведение систем и на начальном этапе функционирования (в переходном режиме), и при длительной эксплуатации (в стационарном режиме).

Как было показано ранее (9.1), производительность ВС определяется числом исправных машин. Ясно, что в условиях, когда имеются отказы и восстановления машин, число исправных ЭМ в системе в момент времени $t \geqslant 0$ есть случайная функция, обозначим ее через $\xi(t)$. Пусть i — начальное состояние ВС, т. е. число исправных машин в системе при $t = 0$, где $i \in E_0^N$ Функция $\xi(t)$ определяется начальным состоянием i ВС (следовательно, моментами освобождения восстанавливающих устройств, которые были заняты ремонтом при $t = 0$), моментами появления новых отказов в машинах (или поступления новых отказавших машин на обслуживание ВУ), моментами устранения новых отказов.

Пусть в некоторый момент времени t^* известно число исправных машин в системе, т. е. $\xi(t^*)$. Очевидно, что значения $\xi(t)$ после t^* (дальнейшее течение процесса обслуживания отказавших машин восстанавливающими устройствами) не зависят от того, что было до t^* С одной стороны, моменты освобождения занятых ВУ не зависят от того, что было до t^*, так как закон (2.18) распределения времени восстановления машины — экспоненциальный. С другой стороны, моменты появления новых отказов не зависят от того, что было до t^*, так как поток отказов — простейший (2.15) и, следовательно, в нем отсутствует последействие. Независимость окончания устранения новых отказов в ЭМ от t^* также следует из закона (2.18). Таким образом, все величины, определяющие $\xi(t)$, не зависят от того, что было до момента времени t^* Такие случайные процессы, течение которых не зависит от прошлого, называются марковскими процессами. Следовательно, $\xi(t)$ *является марковским случайным процессом.*

Обозначим через $\{P_j(i, t)\}$ распределение вероятностей состояний системы в момент $t \geqslant 0$ при условии, что в начальный момент времени было исправно $i \in E_0^N$ машин; $\sum_{j=0}^{N} P_j(i, t) = 1$ — условие нормировки. Функция $P_j(i, t)$ — вероятность того, что в системе, начавшей функционировать в состоянии $i \in E_0^N$, в момент $t \geqslant 0$ будет $j \in E_0^N$ исправных машин:

$$P_j(i, t) = P\{\xi(t) = j \mid i = E_0^N\}, \quad j \in E_0^N \tag{9.4}$$

Тогда вероятность $P_j(i, t)$ $(i, j \in E_0^N)$ и будет показателем, характеризующим поведение ВС в переходном режиме функционирования. Очевидно, что при $i \neq j$, $i, j \in E_0^N$, имеют место равенства: $P_j(i, 0) = 0$, $P_i(i, 0) = 1$.

Используя результаты теории массового обслуживания, нетрудно показать, что распределение $\{P_j\}$ $(j \in E_0^N)$, где

$$P_j = \lim_{t \to \infty} P_j(i, t); \quad \sum_{j=0}^{N} P_j = 1 \tag{9.5}$$

не зависит от начального состояния $i \in E_0^N$ ВС. Следовательно, P_j $(j \in E_0^N)$ есть показатель надежности для стационарного (или установившегося) режима работы ВС.

Переходный режим функционирования ВС характеризуется детерминированно заданным начальным состоянием, а стационарный — распределением вероятностей состояний $\{P_j\}$, таким, что для всякого $j \in E_0^N$ (т. е. $\forall j \in E_0^N$) имеет место $P_j \neq 0$.

Для характеристики качества функционирования ВС (со структурной избыточностью) в переходном режиме используются функции от времени t: $R(t)$, $U(t)$ и $S(t)$ — соответственно функция надежности (вероятность безотказной работы), функция восстановимости (вероятность восстановления) и функция готовности ВС.

Функцию надежности определим как вероятность того, что производительность ВС, начавшей функционировать в состоянии i $(n \leqslant i \leqslant N)$ *на промежутке времени* $[0, t)$, *равна производительности основной подсистемы.* Приведем формальную запись определения функции надежности ВС:

$$R(t) = P\{\forall \tau \in [0, t) \to \Omega(\tau) = A_n n\omega |\ n \leqslant i \leqslant N\},$$

где $\Omega(\tau)$ — производительность системы в момент времени τ.

Говоря иначе, функция $R(t)$ есть вероятность того, что в системе, начавшей функционировать с i, $n \leqslant i \leqslant N$, исправными машинами, на промежутке времени $[0, t)$ будет не менее n исправных машин:

$$R(t) = P\{\forall \tau \in [0, t) \to \xi(\tau) \geqslant n |\ n \leqslant i \leqslant N\}. \qquad (9.6)$$

Очевидно, что $R(0) = 1$, $R(+\infty) = 0$.

Под функцией восстановимости будем понимать вероятность того, что в ВС, имеющей начальное состояние i $(0 \leqslant i < n)$, *будет восстановлен на промежутке времени* $[0, t)$ *уровень производительности, равный производительности основной подсистемы.* Функцию восстановимости ВС можно определить с помощью двух эквивалентных выражений:

$$U(t) = 1 - P\{\forall \tau \in [0, t) \to \Omega(\tau) = 0 | 0 \leqslant i < n\};$$
$$U(t) = 1 - P\{\forall \tau \in [0, t) \to \xi(\tau) < n | 0 \leqslant i < n\}. \qquad (9.7)$$

Очевидно, что $U(0) = 0$, $U(+\infty) = 1$.

В инженерной практике наиболее употребительны не $R(t)$ и $U(t)$, а математическое ожидание времени безотказной работы (средняя наработка до отказа) и среднее время восстановления ВС, которые по определению соответственно равны:

$$\theta = \int_0^\infty R(t)dt; \tag{9.8}$$

$$T = \int_0^\infty tdU(t) \tag{9.9}$$

(см. формулы (2.11) и (2.18)).

Функцией готовности назовем *вероятность того, что производительность системы, начавшей функционировать в состоянии* $i \in E_0^N$, *равна в момент времени* $t \geqslant 0$ *производительности основной подсистемы:*

$$S(t) = P\{\Omega(t) = A_n n\omega \,|\, i \in E_0^N\}.$$

Говоря иначе, функция $S(t)$ есть вероятность того, что при $t \geqslant 0$ число исправных ЭМ в ВС, имевшей начальное состояние $i \in E_0^N$, не менее числа машин основной подсистемы:

$$S(t) = P\{\xi(t) \geqslant n \,|\, i \in E_0^N\}. \tag{9.10}$$

Из определения следует, что $0 < S(+\infty) < 1$,

$$S(0) = \begin{cases} 1, & \text{если } n \leqslant i \leqslant N; \\ 0, & \text{если } 0 \leqslant i < n \end{cases}$$

для невосстанавливаемых ВС $R(t) = S(t)$. Если учесть (9.4) и (9.10), то функцию готовности системы можно выразить через вероятности ее состояний:

$$S(t) = \sum_{j=n}^{N} P_j(i, t), \quad i \in E_0^N \tag{9.11}$$

Таким образом, *функции надежности и готовности характеризуют способности ВС обеспечить требуемое быстродействие на промежутке времени* $[0, t)$ *и в момент* $t \geqslant 0$ *соответственно. Функция восстановимости раскрывает возможности системы к восстановлению, т. е. характеризует способность системы приобретать требуемый уровень производительности после отказа всех избыточных машин и части машин основной подсистемы.*

Предельные значения показателей (9.6), (9.7), (9.10) при $t \to \infty$ будут характеризовать надежность ВС в стационарном режиме работы. Однако для данного режима такие показатели, как $R(t)$ и $U(t)$, не информативны:

$$\lim_{t\to\infty} R(t) = 0, \quad \lim_{t\to\infty} U(t) = 1. \tag{9.12}$$

Поскольку имеет место (9.12) и есть практическая потребность в оценке на промежутке времени производительности ВС, находящихся длительное время в эксплуатации, то целесообразно рассматривать две функции: $R^*(t)$ и $U^*(t)$ (которые могли бы быть названы как *функции оперативной надежности и восстановимости ВС*).

Функция $R^*(t)$ определяется как вероятность того, что производительность системы, которая в начальный момент находится в состоянии i, $n \leqslant i \leqslant N$, с вероятностью P_i (9.5), равна на промежутке времени $[0, t)$ производительности основной подсистемы. Приведем формальную запись данного определения:

$$R^*(t) = P\{\forall \tau \in [0, t) \to \Omega(\tau) = A_n n\omega | P_i,\ i \in E_n^N\},$$

или же, что эквивалентно,

$$R^*(t) = P\{\forall \tau \in [0, t) \to \xi(\tau) \geqslant n | P_i, i \in E_n^N\}, \qquad (9.13)$$

где $E_n^N = \{n, n+1, \ldots, N\}$. Из (9.13) следует, что $R^*(0) = \sum_{i=n}^{N} P_i$.

Под *функцией* $U^*(t)$ понимается вероятность того, что в ВС, находящейся в начальном состоянии i, $0 \leqslant i < n$, с вероятностью P_i (9.5), на промежутке времени $[0, t)$ будет восстановлен уровень производительности основной подсистемы. Приведем два эквивалентных выражения для данной функции:

$$U^*(t) = 1 - P\{\forall \tau \in [0, t) \to \Omega(\tau) = 0 | P_i,\ 0 \leqslant i < n\};$$
$$U^*(t) = 1 - P\{\forall \tau \in [0, t) \to \xi(\tau) < n | P_i,\ 0 \leqslant i < n\}. \qquad (9.14)$$

Из определения $U^*(t)$ следует, что $U^*(0) = 1 - \sum_{i=0}^{n-1} P_i$.

В отличие от функций надежности (9.6) и восстановимости (9.7) функция готовности (9.10) может быть использована в качестве количественной характеристики стационарного режима работы ВС. В самом деле, из (9.10), (9.11) и (9.5) следует, что

$$\lim_{t \to \infty} S(t) = \sum_{j=n}^{N} \lim_{t \to \infty} P_j(i, t) = \sum_{j=n}^{N} P_j = S, \qquad (9.15)$$

причем предел S не зависит от начального состояния системы $i \in E_0^N$. Величину S называют *коэффициентом готовности* ВС. Он является самым рас-

пространенным показателем для стационарного режима функционирования ВС.

Отметим прикладное значение введенных показателей надежности ВС. Показатели надежности устанавливают взаимосвязь между производительностью и собственно надежностью ВС. Следовательно, показатели надежности позволяют, во-первых, подобрать такой состав вновь компонуемой ВС, при котором обеспечиваются заданные уровни и производительности, и надежности; во-вторых, проанализировать качество работы существующей ВС и оценить ее возможности по решению задач. Последнее важно знать и при организации контрольно-профилактических и диагностических работ в ВС и при организации прохождения задач пользователей. Так, показатели надежности для переходного режима работы ВС позволяют получить следующую информацию:

1. С какой вероятностью задача пользователя будет решена, если в момент ее поступления производительность ВС не менее требуемой (9.6); говоря другими словами, сможет ли пользователь успеть решить свою задачу до отказа системы (9.8)?

2. Как быстро можно ожидать восстановления требуемого для пользователя уровня производительности, если в момент поступления задачи производительность ВС низка (9.7), (9.9)?

3. Будет ли ВС иметь необходимую производительность (точнее, с какой вероятностью она будет ее иметь) в момент поступления задачи в систему (9.10)?

Показатели надежности для стационарного режима функционирования ВС, в частности, информируют о следующем:

1. Могут ли быть решены поступающие задачи, если система длительно эксплуатируется? Могут ли быть решены задачи, если в момент их поступления достоверно неизвестно, в каком состоянии находится система (9.13)?

2. Сколь быстро можно ожидать восстановления требуемого уровня производительности в условиях, когда ВС уже длительно эксплуатируется (9.14)?

3. Будет ли система иметь необходимую производительность в любой момент поступления задачи, если она уже достаточно долго находится в эксплуатации (9.15)?

9.4. О методике расчета показателей надежности вычислительных систем

К методике расчета показателей качества работы ВС предъявляются следующие требования:

1) приемлемость методики к большемасштабным и масштабируемым ВС, или, иначе говоря, к ансамблям с любым количеством одинаковых ЭМ или процессоров;

2) адекватность стохастических моделей функционирования ВС реальному процессу их работы или реализации принципа квазианалогии, который применительно к рассматриваемой проблеме гарантирует не подобие между стохастическими моделями и функционированием ВС, а удовлетворительную для практики точность расчетов;

3) единообразие методов и приемов исследования функционирования ВС как в переходном, так и в стационарном режимах;

4) простота численного анализа функционирования ВС при произвольном количестве машин (определение числовых значений показателей функционирования ВС не должно быть связано с трудоемкими вычислениями, т. е. с решением сложных задач, доступных мощным средствам ВТ);

5) возможность выявления общих количественных закономерностей по производительности и надежности функционирования ВС, которые отражают достигнутый и перспективный уровни технологии ВТ.

При расчете показателей надежности за основу берется стохастическая модель функционирования ВС, представленная на рис. 9.1. В случае отказа одной или нескольких ЭМ основной подсистемы и после их локализации (диагностики ВС) требуется реконфигурация ВС в целом. С помощью реконфигуратора в пределах ВС порождается новая конфигурация основной подсистемы из n исправных ЭМ. В качестве таковой конфигурации, в частности, может выступать модернизированный вариант основной подсистемы, который включает все исправные ЭМ исходной подсистемы и необходимое число исправных ЭМ структурной избыточности. В рамках сказанного

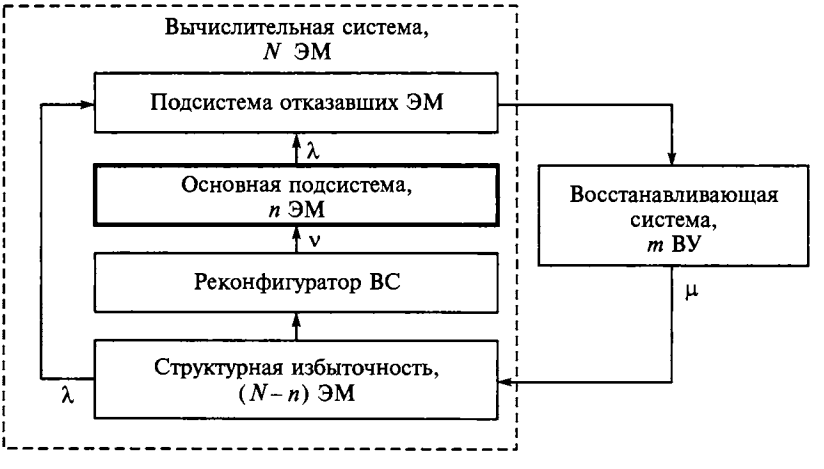


Рис. 9.1. Модель функционирования ВС со структурной избыточностью

можно говорить о виртуальном переключении исправных ЭМ из подсистем структурной избыточности в основную подсистему и об интенсивности переключения ν. Результаты изучения возможностей аппаратурно-программных средств диагностики и реконфигурации ВС и степени влияния параметра ν на значения показателей надежности ВС позволяют здесь отказаться от учета интенсивности переключения при анализе функционирования ВС (см. [5], § 7.3 и 7.4). Итак, при расчете показателей надежности ВС можно считать, что реконфигурация системы осуществляется «мгновенно», т. е. что $\nu^{-1} = 0$.

Далее, для ВС могут быть построены достаточно эффективные аппаратурно-программные средства (само)контроля и (само)диагностики, при которых можно считать, что отказавшие машины «мгновенно» поступают на обслуживание в восстанавливающую систему. В противном случае потребуется лишь корректировка значений параметра μ, т. е. соответствующее увеличение среднего времени восстановления отказавшей ЭМ одним ВУ.

При изучении надежности большемасштабных распределенных ВС (с программируемой структурой) должна быть учтена следующая особенность: процесс восстановления обнаруженных отказавших ЭМ предусматривает не ремонт машин, а обязательно реконфигурацию систем. При этом проверка работоспособности ВС и поиск отказавших машин выполняются соответственно средствами (само)контроля и (само)диагностики. Последние средства для краткости будем называть *контролёром* и *диагностом*. Реконфигурация системы заключается в программной настройке новой конфигурации с заданным числом n исправных ЭМ; она осуществляется *реконфигуратором*. Для создания новой конфигурации основной подсистемы могут быть использованы в общем случае машины из избыточности и/или резерва. Контролёр, диагност и реконфигуратор являются компонентами распределенной ОС. Эта композиция, по сути, является *виртуальным восстанавливающим устройством* (ВУ) для распределенной ВС; следуя традиции, ее будем называть просто ВУ.

В распределенных ВС допустима генерация нескольких виртуальных трехкомпонентных восстанавливающих устройств. Тогда на каждое ВУ могут быть возложены функции по обслуживанию только одной элементарной машины ВС. При этом функциями компонентов каждого ВУ будут следующие:

- для диагноста — выбор (локализация) обслуживаемой ЭМ;
- для контролера — проверка работоспособности выбранной машины;
- для реконфигуратора — альтернативное выполнение одной из двух функций: сохранение проверяемой ЭМ в составе основной подсистемы, если она исправна, или включение машины из резерва в состав основной подсистемы в противном случае.

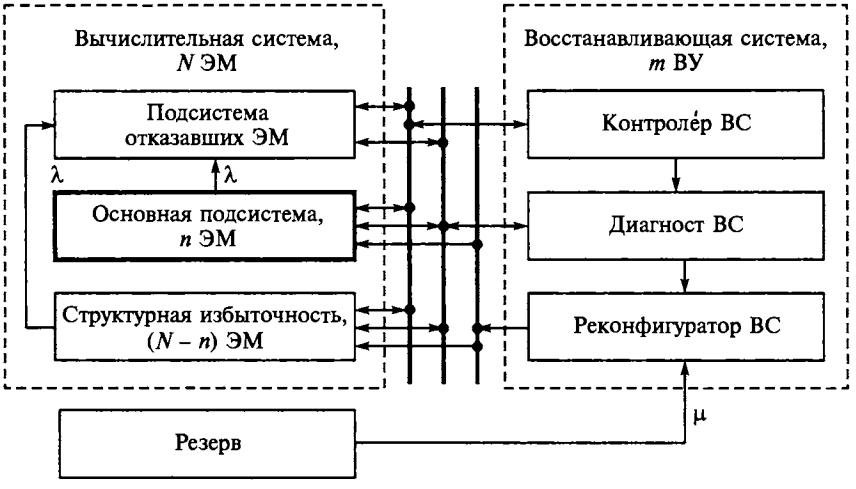


Рис. 9.2. Модель функционирования большемасштабных ВС со структурной избыточностью

В дальнейшем будем считать, что m виртуальных ВУ составляют восстанавливающую систему, $1 \leqslant m \leqslant N$. Интенсивность μ в случае большемасштабности ВС интерпретируется как среднее число машин резерва, включаемых в единицу времени одним ВУ (точнее, реконфигуратором) в состав ВС вместо отказавших ЭМ. При этом среднее время восстановления ЭМ

$$\tau = \mu^{-1} = \tau_к + \tau_д + \tau_р,$$

где $\tau_к$, $\tau_д$ и $\tau_р$ — математические ожидания времени соответственно контроля, диагностики и реконфигурации ВС.

Таким образом, учет специфики большемасштабных распределенных ВС требует модернизации модели, изображенной на рис. 9.1. Результаты соответствующей модернизации представлены на рис. 9.2.

В условиях изложенных требований практически приемлемым для вычисления показателей (9.6), (9.7), (9.10) будет подход, основанный на классическом аппарате теории массового обслуживания и методах приближенных вычислений. Данный подход достаточно хорошо развит [5, 6] и позволяет исследовать ВС с большим числом ЭМ. Изложим схему этого подхода:

- составляются дифференциальные уравнения для вероятностей состояний системы с учетом подмножества поглощающих состояний;
- задаются начальные условия;
- система дифференциальных уравнений с помощью преобразования Лапласа сводится к алгебраической;

• по правилу Крамера определяется решение алгебраической системы уравнений, причем решение выражается через полиномы, вычисляемые рекуррентно;

• доказываются свойства корней полиномов, позволяющие приближенно вычислять их значения;

• после обращения преобразования Лапласа выписываются формулы для показателей качества функционирования ВС;

• для получения числовых значений показателей составляются программы на одном из алгоритмических языков.

Учитывая объем учебного пособия и степень распространения тех или иных показателей надежности ВС, ограничимся лишь асимптотическими оценками вероятностей безотказной работы (9.6) и восстановления (9.7). В Приложении 2 описан метод расчета функции $R(t)$ надежности ВС.

Далее рассмотрим способы расчета математического ожидания времени безотказной работы (9.8), среднего времени восстановления (9.9) и функции готовности (9.10), которые существенно проще вычислений по описанной выше схеме и которые позволяют провести анализ функционирования ВС в переходном режиме. Для анализа поведения ВС в стационарном режиме выполним расчет показателей (9.13)–(9.15).

9.5. Расчет показателей надежности для переходного режима функционирования вычислительных систем

9.5.1. Надежность большемасштабных вычислительных систем

Прежде всего изучим поведение невосстанавливаемой ВС со структурной избыточностью при неограниченном наращивании во времени общего количества N ЭМ, точнее говоря, получим для фиксированного числа n машин основной подсистемы оценки функции надежности ВС (9.6) при $N \to \infty$.

Пусть параметры ВС изменяются в дискретные моменты времени: $1, 2, \ldots, \tau, \ldots, t, \ldots$; $r(\tau)$ — вероятность безотказной работы одной ЭМ (2.10) или вероятность того, что ЭМ в момент τ исправна; $N(\tau)$ — общее число ЭМ в момент τ.

Можно доказать [6] справедливость следующих формул для функции надежности ВС:

$$R(t)\begin{cases}\geqslant 1-B^{-1}(1-t^{-B});\\ \leqslant \dfrac{2}{1+t};\\ \approx e^{-\Lambda t};\end{cases}\quad \text{при}\quad N(\tau)\begin{cases}\geqslant \dfrac{1+B}{K}\ln(\tau+1);\\ \leqslant \dfrac{1}{K}\ln(\tau+1);\\ = N = \text{const}\end{cases} \tag{9.16}$$

соответственно, где B — произвольное положительное число;

$$K = \max_{1\leqslant\tau\leqslant t}\kappa(\tau),\quad k = \min_{1\leqslant\tau\leqslant t}\kappa(\tau); \tag{9.17}$$

$$\kappa(\tau) = \nu(\tau)\ln\frac{\nu(\tau)}{r(\tau)} + [1-\nu(\tau)]\ln\frac{1-\nu(\tau)}{1-r(\tau)},\quad \nu(\tau) = \frac{n-1}{N(\tau)}; \tag{9.18}$$

для константы Λ справедливо неравенство

$$\ln[1-e^{-KN}] \geqslant \Lambda \geqslant \ln[1-e^{-kN}]. \tag{9.19}$$

Таким образом, чтобы невосстанавливаемая ВС имела достаточно высокий уровень надежности (т. е. чтобы $R(t)\to 1$, см. (9.16)–(9.19)) сколь угодно продолжительное время ($t\to\infty$), необходим, по крайней мере, логарифмический рост во времени числа ее ЭМ.

Из (9.16) следует, что вероятность безотказной работы системы, в которой $N = \text{const}$, экспоненциально с ростом t стремится к нулю. Скорость уменьшения $R(t)$ зависит от параметра Λ, т. е. от интенсивности отказов ВС. Очевидно, что надежность системы может быть повышена за счет уменьшения Λ. Последнее можно достичь также, если ввести в систему средства поиска неисправностей и восстановления.

9.5.2. Математические ожидания времени безотказной работы и восстановления вычислительной системы

Применение классического способа расчета математических ожиданий времени безотказной работы θ (9.8) и времени восстановления T (9.9) для большемасштабных ВС наталкивается на серьезные препятствия, связанные с трудоемкими и сложными вычислениями функций надежности $R(t)$ и восстановимости $U(t)$. Вычисления функций $R(t)$ и $U(t)$ основываются, коротко говоря, на традиционных стохастических моделях теории массового обслуживания и методах приближенных вычислений. Трудоемкость такого расчета повышается с ростом количества машин в системе, и, кроме того, на этом пути не удается получить аналитические формулы для отыскания числовых значений θ и T.

Для распределенных ВС θ и T удобно рассчитывать «частотным» методом*, который достаточно прост и дает результаты, хорошо согласующиеся с более точными вычислениями. Легко установить, что среднее время безотказной работы ВС при $n \neq N$ и при $n = N$ соответственно равно:

$$\theta = \sum_{j=n+1}^{N} \frac{1}{j\lambda} \prod_{l=n}^{j-1} \frac{\mu_l}{l\lambda} + \frac{1}{n\lambda}; \quad \theta = \frac{1}{N\lambda}. \tag{9.20}$$

Среднее время восстановления ВС при $n \neq 1$ и при $n = 1$ определяется соответственно выражениями:

$$T = \frac{1}{\mu_0} \prod_{l=1}^{n-1} \frac{l\lambda}{\mu_l} + \sum_{j=1}^{n-1} \frac{1}{j\lambda} \prod_{l=j}^{n-1} \frac{l\lambda}{\mu_l}; \quad T = \frac{1}{\mu_0}. \tag{9.21}$$

В формулах (9.20), (9.21)

$$\mu_l = \begin{cases} (N-l)\mu, & \text{если } (N-m) \leqslant l \leqslant N; \\ m\mu, & \text{если } 0 \leqslant l < (N-m); \end{cases} \tag{9.22}$$

λ^{-1} — среднее время безотказной работы одной ЭМ (2.14); m — число восстанавливающих устройств; μ^{-1} — среднее время восстановления отказавшей ЭМ одним восстанавливающим устройством (2.18).

9.5.3. Функция готовности вычислительных систем

Рассмотрим методы расчета функции готовности ВС. Для простоты будем вместо (9.4) употреблять обозначение

$$P_j(t) = P\{\xi(t) = j \,|\, i \in E_0^N\}, \quad j \in E_0^N. \tag{9.23}$$

Для вывода дифференциальных уравнений воспользуемся формулой полных вероятностей

$$P_j(t + \Delta t) = \sum_{l=0}^{N} P_l(t) P_{lj}(\Delta t), \tag{9.24}$$

где $P_{lj}(\Delta t)$ — условная вероятность того, что ВС, находящаяся в некоторый момент $t \geqslant 0$ в состоянии l, т. е. $\xi(t) = l$, перейдет по истечении времени

* *Безносов Г.П., Зеленцов Б.П.* Частотный метод анализа надежности систем с восстановлением, состоящих из однотипных элементов // Известия СО АН СССР, Сер. техн. — 1966. Т. 1. № 2. С. 106–111.

Δt в состояние j, т. е. $\xi(t+\Delta t)=j$. Определим асимптотические оценки для вероятностей $P_{ij}(\Delta t)$ при $\Delta t \to 0$, $l, j \in E_0^N$

Прежде всего из (2.15) и (2.21) следует, что вероятность появления за время Δt одного отказа в машине есть величина порядка $\lambda\Delta t$, а вероятность появления более одного отказа — величина вида $o(\Delta t)$, где $o(\Delta t)$ — бесконечно малая порядка выше Δt. Вероятность появления за время Δt одного отказа в ВС, находящейся в состоянии $l \in E_0^N$, есть величина $l\lambda\Delta t + o(\Delta t)$.

Из (2.18) ясно, что вероятность восстановления за время Δt отказавшей машины одним восстанавливающим устройством равна $\mu\Delta t + o(\Delta t)$. Если ВС находится в состоянии $l \in E_0^N$ и восстановлением машин занято k устройств, то вероятность восстановления за время Δt одной отказавшей машины будет равна $k\mu\Delta t + o(\Delta t)$, где $k=(N-l)$ при $(N-m) \leqslant l$ и $k=m$ при $(N-m) > l$.

Переход системы из состояния l в состояние j при $|l-j|>1$ требует, очевидно, наступления по меньшей мере двух событий (или двух отказов, или двух восстановлений). Следовательно,

$$P_{ij}(\Delta t) = o(\Delta t), \quad |l-j| \geqslant 2, \quad l, j \in E_0^N \tag{9.25}$$

Далее, для перехода ВС из состояния $0 \leqslant l < N$ в состояние $l+1$ требуется, чтобы произошло одно восстановление либо наступило несколько событий, поэтому

$$P_{l,l+1}(\Delta t) = \begin{cases} m\mu\Delta t + o(\Delta t), & 0 \leqslant l \leqslant (N-m); \\ (N-l)\mu\Delta t + o(\Delta t), & (N-m) < l < N. \end{cases} \tag{9.26}$$

Очевидно, что

$$P_{l,l-1}(\Delta t) = l\lambda\Delta t + o(\Delta t), \quad 0 < l \leqslant N. \tag{9.27}$$

Наконец, учитывая (9.25)–(9.27), имеем

$$P_{ll}(\Delta t) = 1 - P_{l,l+1}(\Delta t) - P_{l,l-1}(\Delta t) + o(\Delta t), \quad 0 \leqslant l \leqslant N,$$

где при $l=N$ второй, а при $l=0$ третий член правой части надо заменить на 0. Следовательно,

$$\left.\begin{aligned} &P_{00}(\Delta t) = 1 - m\mu\Delta t + o(\Delta t); \\ &P_{ll}(\Delta t) = \begin{cases} 1 - l\lambda\Delta t - m\mu\Delta t + o(\Delta t), & 0 < l \leqslant (N-m); \\ 1 - l\lambda\Delta t - (N-l)\mu\Delta t + o(\Delta t), & (N-m) < l < N; \end{cases} \\ &P_{NN}(\Delta t) = 1 - N\lambda\Delta t + o(\Delta t). \end{aligned}\right\} \tag{9.28}$$

Подставляя в (9.24) асимптотические оценки (9.25)–(9.28), получаем:

$$P_0(t+\Delta t) = P_0(t)P_{00}(\Delta t) + P_1(t)P_{10}(\Delta t) + o(\Delta t) =$$
$$= (1 - m\mu\Delta t)P_0(t) + \lambda\Delta t P_1(t) + o(\Delta t);$$
$$P_j(t+\Delta t) = P_{j-1}(t)P_{j-1,\,j}(\Delta t) + P_j(t)P_{jj}(\Delta t) + P_{j+1}(t)P_{j+1,\,j}(\Delta t) + o(\Delta t) =$$
$$= \begin{cases} m\mu\Delta t P_{j-1}(t) + [1 - (j\lambda + m\mu)\cdot\Delta t]P_j(t) + \\ + (j+1)\lambda\Delta t P_{j+1}(t) + o(\Delta t), & 0 < j \leqslant (N-m); \\ [N-(j-1)]\mu\Delta t P_{j-1}(t) + \{1 - [j\lambda + (N-j)\mu]\Delta t\}P_j(t) + \\ + (j+1)\lambda\Delta t P_{j+1}(t) + o(\Delta t), & (N-m) < j < N; \end{cases}$$
$$P_N(t+\Delta t) = P_{N-1}(t)P_{N-1,\,N}(\Delta t) + P_N(t)P_{NN}(\Delta t) + o(\Delta t) =$$
$$= \mu\Delta t P_{N-1}(t) + (1 - N\lambda\Delta t)P_N(t) + o(\Delta t).$$

Перенеся $P_j(t)$, $j \in E_0^N$, в левую часть последних равенств, разделив на Δt и перейдя к пределу при $\Delta t \to 0$, получим:

$$\left.\begin{array}{l} P_0'(t) = -m\mu P_0(t) + \lambda P_1(t); \\ P_j'(t) = \begin{cases} m\mu P_{j-1}(t) - (j\lambda + m\mu)P_j(t) + (j+1)\lambda P_{j+1}(t), \\ \qquad\qquad 0 < j \leqslant (N-m); \\ [N-(j-1)]\mu P_{j-1}(t) - [j\lambda + (N-j)\mu]P_j(t) + (j+1)\lambda P_{j+1}(t), \\ \qquad\qquad (N-m) < j < N; \end{cases} \\ P_N'(t) = \mu P_{N-1}(t) - N\lambda P_N(t). \end{array}\right\} \quad (9.29)$$

Начальные условия могут быть заданы в следующем виде:

$$P_j(0) = 0, \quad j \neq i, \quad P_i(0) = 1, \quad i, j \in E_0^N.$$

Задача интегрирования (9.29) системы $(n+1)$ линейных однородных дифференциальных уравнений относительно неизвестных функций $P_j(t)$, $j \in E_0^N$, принципиально разрешима. Практически же отыскание решения системы (9.29) при заданных начальных условиях осуществляется численными методами и по схеме, изложенной в § 9.4. Процесс расчета вероятностей $P_j(t)$, $j \in E_0^N$, для большемасштабных ВС достаточно трудоемок, сложность расчета повышается с увеличением N.

Как уже отмечалось, распределенные ВС в своем составе могут иметь большое количество элементарных машин или процессоров. Так, существуют и создаются ВС, число элементарных процессоров в которых имеет порядок 10^2–10^6 Следовательно, при изучении поведения большемасштабных

ВС можно строить математические модели с числом ЭМ, равным бесконечности (точнее, при $N \to \infty$). Последнее допущение существенно упрощает расчет функции готовности ВС (9.10).

Модернизируем обозначения, введенные для случая, когда N конечно. Пусть $E_0^\infty = \{0, 1, 2, \ldots\}$ — пространство состояния системы; $P_j(t)$ — вероятность того, что ВС в момент времени $t \geqslant 0$ имеет $j \in E_0^N$ исправных машин. Тогда для расчета функции готовности ВС следует вместо (9.11) использовать формулу

$$S(t) = \sum_{j=n}^{\infty} P_j(t) = 1 - \sum_{j=0}^{n-1} P_j(t). \tag{9.30}$$

Очевидно, что для любого состояния $j \in E_0^\infty$ число отказавших ЭМ системы будет больше числа m восстанавливающих устройств. Итак, при любом состоянии ВС и для любого момента времени восстановлением отказавших ЭМ будет занято m ВУ и, следовательно,

$$P_{l,l+1}(\Delta t) = m\mu\Delta t + o(\Delta t), \quad l \in E_0^\infty. \tag{9.31}$$

Учет разницы между асимптотическими оценками (9.26) и (9.31) позволяет преобразовать систему уравнений (9.29) к более простому виду:

$$\left.\begin{aligned} P_0'(t) &= -m\mu P_0(t) + \lambda P_1(t), \\ P_j'(t) &= m\mu P_{j-1}(t) - (j\lambda + m\mu)P_j(t) + (j+1)\lambda P_{j+1}(t), \quad j \geqslant 1. \end{aligned}\right\} \tag{9.32}$$

При этом нормировочное условие принимает вид $\sum_{j=0}^{\infty} P_j(t) = 1$.

Решение однородной системы (9.32) обыкновенных линейных дифференциальных уравнений первого порядка может быть найдено методом производящих функций [21]:

$$P_j(t) = e^{-\frac{m\mu}{\lambda}(1-e^{-\lambda t})} \sum_{l=0}^{j} b_l(t) \frac{(m\mu/\lambda)^{j-l}(1-e^{-\lambda t})^{j-l}}{(j-l)!}, \tag{9.33}$$

где $b_l(t)$ определяется из равенства

$$\sum_{l=0}^{\infty} b_l(t)x^l = \sum_{l=0}^{\infty} P_l(0)(xe^{-\lambda t} + 1 - e^{-\lambda t})^l, \tag{9.34}$$

$$\sum_{l=0}^{\infty} P_l(0) = 1, \quad |x| \leqslant 1, \quad j \in E_0^\infty.$$

Формулы (9.33) и (9.34) позволяют анализировать готовность ВС с массовым параллелизмом.

9.6. Расчет показателей надежности для стационарного режима работы вычислительных систем

Вычислительные системы общего назначения, как правило, рассчитаны на длительную эксплуатацию. Целью настоящего параграфа является изучение с позиций надежности поведения ВС при $t \to \infty$.

9.6.1. Функции оперативной надежности и восстановимости вычислительных систем

Выведем расчетные формулы для функций $R^*(t)$ и $U^*(t)$, характеризующих работу ВС в стационарном режиме, (9.13), (9.14). Пусть

$$Q_i(t) = P\{\forall\tau \in [0, t) \to \xi(\tau) \geqslant n \,|\, i \in E_0^N\} \tag{9.35}$$

является условной вероятностью события $\xi(\tau) \geqslant n$ в предположении, что в системе в момент времени $t = 0$ имеется $i \in E_n^N$ исправных ЭМ; τ — любой момент времени, принадлежащий $[0, t)$. Тогда, учитывая (9.13), по формуле полных вероятностей имеем

$$R^*(t) = \sum_{i=n}^{N} P_i Q_i(t), \tag{9.36}$$

где P_i определяется формулами (9.4) и (9.5).

Рассчитаем функцию (9.35). Для этого обозначим через $\pi_r(t)$ и $u_l(t)$ соответственно вероятность того, что произойдет ровно r отказов, и вероятность того, что произойдет ровно l восстановлений за время t, если в системе имеется $i \in E_n^N$ исправных машин. Тогда

$$Q_i(t) = \sum_{l=0}^{\infty} u_l(t) \sum_{r=o}^{i-n+l} \pi_r(t). \tag{9.37}$$

Если учесть (2.15), (2.18), то можно показать, что

$$\pi_r(t) = \frac{(i\lambda t)^r}{r!} e^{-i\lambda t}; \tag{9.38}$$

$$u_l(t) = \frac{(\mu t)^l}{l!}[\Delta(N - i - m)m^l e^{-m\mu t} + \Delta(m - N + i)(N - i)^l e^{-(N-i)\mu t}], \tag{9.39}$$

где $r, l \in E_0^{\infty}$,

$$\Delta(x) = \begin{cases} 1, & \text{если } x \geqslant 0; \\ 0, & \text{если } x < 0, \end{cases}$$

и считается, что $0^0 = 1$.

Таким образом, формулы (9.37)–(9.39) позволяют рассчитать условную вероятность $Q_i(t)$, $i \in E_n^N$, с заданной степенью точности.

В некоторых случаях достаточно найти оценку $\tilde{R}^*(t)$ снизу для функции $R^*(t)$:

$$\tilde{R}^*(t) = \sum_{i=n}^{N} P_i \sum_{r=0}^{i-n} \pi_r(t). \tag{9.40}$$

Начальное значение функции $R^*(t)$, как это следует из (9.15), (9.36)–(9.39), равно коэффициенту готовности ВС, т. е.

$$R^*(0) = S.$$

Можно показать, что

$$R^*(+\infty) = \lim_{t\to\infty} R^*(t) \approx \begin{cases} \sum_{i=n}^{N-1} P_i & \text{при } n < N; \\ 0 & \text{при } n = N. \end{cases} \tag{9.41}$$

В самом деле,

$$\lim_{t\to\infty} Q_N(t) = \lim_{t\to\infty} \sum_{r=0}^{N-1} \pi_r(t) = 1 - \lim_{t\to\infty} \sum_{r=N-n+1}^{\infty} \pi_r(t) = 0, \tag{9.42}$$

так как при $i = N$ из (9.39) следует, что $u_l(t) = 0$, $l > 0$, $u_0(t) = 1$. Поскольку $N - n = \text{const}$, то при $t \to \infty$ в системе наиболее вероятными становятся количества отказов $r \in \{N - n + 1, N - n + 2, \ldots\}$. Поэтому при $t \to \infty$ с достаточной для практики точностью можно считать, что $\sum_{r=N-n+1}^{\infty} \pi_r(t) \to 1$ и, следовательно, $Q_N(t) \to 0$.

Далее из (9.39) видно, что при $i \in \{n, n+1, \ldots, N-1\}$ и при большом t вероятности $u_l(t)$ будут тем больше отличаться от нуля, чем больше будет l. Иначе говоря, если $i \in \{n, n+1, \ldots, N-1\}$, то при большом t число восста-

новлений l в системе будет также большим, т. е. если $t \to \infty$, то и $l \to \infty$. Если это так, то при $t \to \infty$ имеет место $\sum_{r=0}^{i-n+1} \pi_r(t) \to 1$ и, следовательно (9.37), $Q_i(t) \to 1$, $i \in E_n^{N-1}$. Переходя к пределу при $t \to \infty$ в (9.36) и учитывая поведение $Q_i(t)$ при $t \to \infty$, получаем (9.41).

Действуя аналогично, находим, что функция (9.14)

$$U^*(t) = 1 - \sum_{i=0}^{n-1} P_i \sum_{r=0}^{\infty} \pi_r(t) \sum_{l=0}^{n-i-1+r} u_l(t). \qquad (9.43)$$

Очевидно, что $U^*(0) = S$, $U^*(+\infty) = \lim_{t\to\infty} U^*(t) = 1$.

Оценкой сверху для $U^*(t)$ будет функция

$$\tilde{U}^*(t) = 1 - \sum_{i=0}^{n-1} P_i \sum_{l=0}^{n-i-1} u_l(t). \qquad (9.44)$$

Выведенные в данном разделе формулы позволяют оценить функционирование ВС в стационарном режиме.

9.6.2. Распределение вероятностей состояний вычислительных систем

Рассчитаем *распределение вероятностей (9.5) состояний ВС для стационарного режима функционирования*: $\{P_0, P, \ldots, P_j, \ldots, P_N\}$; это позволит вычислить функции (9.36) и (9.43) и коэффициент готовности (9.15) ВС.

Для расчета P_j, $j \in E_0^N$, потребуется осуществить предельный переход при $t \to \infty$ в системе уравнений (9.29). Правые части всех уравнений (9.29) при $t \to \infty$ имеют пределы. Следовательно, каждая из производных P_i', $i \in E_0^N$, при $t \to \infty$ также имеет предел и такой предел может быть только нулем. Если бы какая-нибудь производная P_i', $i \in E_0^N$, стремилась к числу, отличному от нуля, то соответствующее $|P_i(t)|$ при $t \to \infty$ неограниченно бы возрастало. Последнее противоречило бы физическому смыслу величин $P_i(t)$ (так как $0 \leqslant P_i(t) \leqslant 1$, $i \in E_0^N$) и формуле (9.5). Таким образом, при $t \to \infty$ имеет место $P_i'(t) \to 0$, $i \in E_0^N$. Следовательно, система дифференциальных уравнений (9.29) преобразуется в систему алгебраических уравнений:

$$\left.\begin{aligned}
&0 = -m\mu P_0 + \lambda P_1;\\
&0 = m\mu P_{j-1} - (j\lambda + m\mu)P_j + (j+1)\lambda P_{j+1}, \qquad 0 < j \leqslant (N-m);\\
&0 = (N-j+1)\mu P_{j-1} - [j\lambda + (N-j)\mu]P_j + (j+1)\lambda P_{j+1}, \quad (N-m) < j < N;\\
&0 = \mu P_{N-1} - N\lambda P_N.
\end{aligned}\right\} \tag{9.45}$$

Если положить

$$\left.\begin{aligned}
&m\mu P_{j-1} - j\lambda P_j = z_j, \qquad 0 < j \leqslant (N-m+1);\\
&[N-(j-1)]\mu P_{j-1} - j\lambda P_j = z_j^*, \quad (N-m+1) \leqslant j \leqslant N
\end{aligned}\right\}, \tag{9.46}$$

то систему (9.45) можно записать в следующем виде:

$$\left.\begin{aligned}
&z_1 = 0;\\
&z_j - z_{j+1} = 0, \qquad 1 < j \leqslant (N-m);\\
&z_{N-m+1} - z^*_{N-m+2} = 0;\\
&z_j^* - z_{j+1}^* = 0, \qquad (N-m+2) \leqslant j < N;\\
&z_N^* = 0.
\end{aligned}\right\} \tag{9.47}$$

Из (9.47) следует, что $z_j = 0$, $0 < j \leqslant (N-m+1)$; тогда, учитывая (9.46) и принимая $\alpha = \mu/\lambda$, получаем

$$P_j = \frac{m\mu}{j\lambda} P_{j-1} = \frac{\alpha m}{j} P_{j-1}; \qquad P_j = \frac{m^j \alpha^j}{j!} P_0, \quad 1 \leqslant j \leqslant (N-m+1). \tag{9.48}$$

Кроме того, из (9.47) следует, что $z_j^* = 0$, $(N-m+1) \leqslant j \leqslant N$;

$$P_j = \frac{[N-(j-1)]\alpha}{j} P_{j-1} = \frac{[N-(j-1)]\ldots[N-(N-m)]\alpha^{i-N+m}}{j(j-1)\ldots(N-m+1)} P_{N-m} =$$

$$= \frac{\alpha^{j-N+m} m!\ (N-m)!}{(N-j)!\, j!} \frac{m^{N-m}\alpha^{N-m}}{(N-m)!} P_0 = \frac{m! m^{N-m} \alpha^j}{(N-j)!\, j!} P_0. \tag{9.49}$$

Используя условие нормировки и (9.48), (9.49), находим

$$P_0 = \left[\sum_{l=0}^{N-m} \frac{m^l \alpha^l}{l!} + m! m^{N-m} \sum_{l=N-m+1}^{N} \frac{\alpha^l}{(N-l)!\, l!} \right]^{-1} \tag{9.50}$$

$$P_j = \frac{\alpha^j}{j!} \left[\Delta(N-m-j) m^j + \Delta(j-N+m) \frac{m! m^{N-m}}{(N-j)!} \right] P_0, \tag{9.51}$$

где $j \in \{1, 2, \ldots, N\}$,

$$\Delta(x) = \begin{cases} 1, & \text{если } x \geqslant 0; \\ 0, & \text{если } x < 0. \end{cases}$$

Заметим, что в (9.48)–(9.51) принято $0^0 = 1$.

Рассмотрим два крайних случая: $m = 1$, $m = N$. В первом случае, когда в ВС имеется одно восстанавливающее устройство, решение (9.51) принимает вид

$$P_j = \left(\frac{\mu}{\lambda}\right)^j \frac{1}{j!}\left[\sum_{l=0}^{N}\left(\frac{\mu}{\lambda}\right)^l \frac{1}{l!}\right]^{-1}, \quad j \in E_0^N \tag{9.52}$$

Если в (9.52) перейти к пределу при $N \to \infty$, получим вероятность того, что при $m = 1$ в стационарном режиме работы большемасштабной ВС исправно j машин, равна:

$$P_j = \left(\frac{\mu}{\lambda}\right)^j \frac{1}{j!} e^{-\mu/\lambda}, \quad j \in E_0^\infty. \tag{9.53}$$

Распределение вероятностей (9.53) является распределением Пуассона.

В случае, когда количество восстанавливающих устройств равно количеству машин в системе, решение (9.51) преобразуется к виду

$$P_j = \frac{N!}{(N-j)!\,j!} \frac{\mu^j \lambda^{N-j}}{(\lambda+\mu)^N}, \quad j \in E_0^N, \tag{9.54}$$

а коэффициент готовности ВС

$$S = 1 - \lambda^{N-n+1}(\lambda+\mu)^{-(N-n+1)}. \tag{9.55}$$

9.6.3. Готовность масштабируемых вычислительных систем

Изучим влияние числа N при его неограниченном увеличении на потенциальную готовность ВС к выполнению основных функций по переработке информации. В дальнейшем будем говорить, что ВС имеет высокую готовность, если $\lim\limits_{N\to\infty} S = 1$. Ранее было показано, что вероятность того, что машина в стационарном режиме исправна, равна ее коэффициенту s готовности (2.26). Поскольку элементарные машины ВС независимо друг от друга могут находиться в исправном состоянии или в состоянии отказа, то, используя при условии $v = (n-1)N^{-1} < s$ известные оценки биноминальных сумм [22], можно показать, что коэффициент готовности ВС

$$S > 1 - e^{-N\kappa}, \quad S = 1 - e^{-N\kappa + o(\ln N)}, \tag{9.56}$$

где $\kappa = v\ln(vs^{-1}) + (1-v)\ln[(1-v)(1-s)^{-1}]$. Из (9.56) следует, что ВС имеет высокую готовность при выполнении условия $v < s$.

Наряду с монопрограммным режимом, когда на системе решается одна сложная задача, представленная в параллельной форме, в ВС широко используются и мультипрограммные режимы (см. разд. 7.2.2). При организации мультипрограммной работы система программным способом разбивается на подсистемы. Количество подсистем соответствует количеству одновременно решаемых задач на ВС, а количество ЭМ в каждой из подсистем не менее количества ветвей в реализуемой параллельной программе. В связи с этим представляет интерес изучить, каким образом параметры подсистем влияют на надежность (точнее, на коэффициент готовности) ВС в целом.

Пусть ВС состоит из h подсистем, причем подсистема с номером $j \in E_1^h$ имеет следующие параметры: N_j — число ЭМ; n_j — минимально допустимое число исправных ЭМ, при котором подсистемой обеспечивается требуемый уровень производительности

$$\sum_{j=1}^{h} N_j = N; \quad v_j = (n_j - 1)N_j^{-1}; \quad \kappa_j = v_j \ln\frac{v_j}{s} + (1 - v_j)\ln\frac{1 - v_j}{1 - s}.$$

Рассмотрим случай, когда все подсистемы важны для существования ВС. Тогда коэффициент готовности ВС

$$S = \prod_{j=1}^{h} S_j \geqslant \prod_{j=1}^{h} (1 - e^{-N_j\kappa_j}), \tag{9.57}$$

где S_j — коэффициент готовности подсистемы с номером $j \in E_1^h$, а неравенство получено на основе (9.56).

Пусть числа n_j и N_j $(j \in E_1^h)$ — переменны, т. е. допускается их варьирование и выполняются условия:

1) для каждой подсистемы с номером j параметр $v_j = \text{const}$;

2) общее число машин постоянно: $N = \text{const}$.

Требуется найти такое распределение $\{N_j^*\}$, $j \in E_1^h$, чисел ЭМ в подсистемах, при котором нижняя оценка (9.57) достигает максимума.

Точное решение сформулированной задачи методом множителей Лагранжа при больших значениях N_j, $j \in E_1^h$, асимптотически совпадает с решением, получающимся при значениях N_j, обеспечивающих постоянство сомножителей в оценке (9.57). Следовательно,

$$N_j^* = N\left[\kappa_j \sum_{l=1}^{h} \kappa_l^{-1}\right]^{-1} \tag{9.58}$$

при этом

$$S \geqslant \left[1 - e^{-\frac{N\bar{\kappa}}{h}}\right]^h \qquad \bar{\kappa} = \left[\frac{1}{h}\sum_{l=1}^{h} \kappa_l^{-1}\right]^{-1} \tag{9.59}$$

$\bar{\kappa}$ — средний гармонический коэффициент κ_j, $j \in E_1^h$ [22, 23].

Рассмотрим более общую ситуацию, при которой оснащение (модулями памяти, внешними запоминающими устройствами, средствами ввода-вывода информации) ЭМ различно. Тогда каждая l-я ЭМ будет иметь свой коэффициент готовности s_l, $l \in E_1^N$ Можно показать, что в общем случае сохраняются все вышеприведенные соотношения, если вместо s рассматривать величину

$$\bar{s} = N^{-1} \sum_{l=1}^{N} s_l.$$

Исследуем теперь поведение ВС в зависимости от степени дробления на подсистемы. Пусть количество N элементарных машин ВС увеличивается вместе с количеством h ее подсистем так, что

$$N = \frac{1+\varepsilon}{\bar{\kappa}} h \ln h - \frac{1}{\bar{\kappa}} h \ln d, \tag{9.60}$$

где d, ε — произвольные действительные числа, $d > 0$. Тогда из (9.59) следует, что

$$S \approx e^{-dh^{-\varepsilon}} \begin{cases} = e^{-dh^{|\varepsilon|}} & \text{при } \varepsilon < 0; \\ > 1 - dh^{-\varepsilon} & \text{при } \varepsilon > 0. \end{cases} \tag{9.61}$$

Следовательно, ВС имеет высокую готовность, если с увеличением h порядок роста N больше величины $h \ln h / \bar{\kappa}$. Из (9.58)–(9.61) видно, что для достижения высокой готовности ВС необходимо выполнение условия $N_j > \kappa_j^{-1} \ln h$, $j \in E_1^h$.

9.6.4. Коэффициент готовности большемасштабных вычислительных систем

Выведем расчетные формулы для коэффициента готовности ВС с большим количеством машин. Для таких систем в стохастических моделях считается, что $N \to \infty$, а функция готовности рассчитывается по формуле (9.30).

Полагая в (9.34) $x = 0$, находим

$$b_0(t) = \sum_{l=0}^{\infty} P_l(0)(1 - e^{-\lambda t})^l, \quad \lim_{t\to\infty} b_0(t) = \sum_{l=0}^{\infty} P_l(0) = 1. \tag{9.62}$$

Если положить в (9.34) $x = 1$, то

$$\sum_{l=0}^{\infty} b_l(t) = \sum_{l=0}^{\infty} P_l(0) = 1. \tag{9.63}$$

Из формул (9.62) и (9.63) при $l \geqslant 1$ следует, что $\lim\limits_{t\to\infty} b_l(t) = 0$. Учитывая (9.33) и последнюю формулу, получаем, что стационарными вероятностями состояний ВС будут

$$P_j = \left(\frac{m\mu}{\lambda}\right)\frac{1}{j!}e^{-m\mu/\lambda}, \quad j \in E_0^{\infty}. \tag{9.64}$$

Заметим, что при $m = 1$ формула (9.64) преобразуется в известный результат (9.53).

Предельный переход при $t \to \infty$ в (9.30) и результат (9.64) приводят к расчетной формуле для коэффициента готовности большемасштабной ВС:

$$S = 1 - \sum_{j=0}^{n-1}\left(\frac{m\mu}{\lambda}\right)^j \frac{1}{j!}e^{-m\mu/\lambda} \tag{9.65}$$

При анализе функционирования большемасштабных ВС (или ВС с массовым параллелизмом) можно использовать статистику не о потоке отказов в одной ЭМ, а об отказах ВС в целом. Тогда стохастическую модель функционирования ВС можно изменить следующим образом. Пусть задана не интенсивность λ отказов одной ЭМ системы, а интенсивность Λ (простейшего) потока отказавших машин ВС. Тогда Λ представляет собой среднее количество отказавших ЭМ, поступающих на m восстанавливающих устройств за единицу времени. Пусть $\overline{P}_j$ — вероятность того, что в стационарном режиме в ВС имеется $j \in E_0^{\infty}$ отказавших машин. Действуя традиционно, получаем:

$$\left.\begin{aligned} &-\Lambda\overline{P}_0 + \mu\overline{P}_1 = 0; \\ &\Lambda\overline{P}_{j-1} - (\Lambda + j\mu)\overline{P}_j + (j+1)\mu P_{j+1} = 0, \quad 1 \leqslant j < m; \\ &\Lambda\overline{P}_{j-1} - (\Lambda + m\mu)\overline{P}_j + m\mu\overline{P}_{j+1} = 0, \qquad m \leqslant j. \end{aligned}\right\} \tag{9.66}$$

Как было показано в (2.18), μ — среднее число восстановлений ЭМ, которое может произвести одно восстанавливающее устройство в единицу времени. Тогда производительность восстанавливающей системы будет характеризоваться величиной $m\mu$. Ясно, что очередь на восстановление отказавших ЭМ не будет расти безгранично и, следовательно, ресурсы ВС будут эксплуатироваться с потенциально возможной эффективностью при условии $\Lambda \leqslant m\mu$.

Легко показать, что решениями системы уравнений (9.66) при выполнении условия нормировки и $\Lambda \leqslant m\mu$ будут:

$$\bar{P}_0 = \left[\sum_{l=0}^{m-1} \frac{1}{l!}\left(\frac{\Lambda}{\mu}\right)^l + \frac{\mu}{(m-1)!(m\mu-\Lambda)}\left(\frac{\Lambda}{\mu}\right)^m\right]^{-1} \tag{9.67}$$

$$\bar{P}_j = \left(\frac{\Lambda}{\mu}\right)^j \left[\Delta(m-j)\frac{1}{j!} + \Delta(j-m)\frac{1}{m!m^{j-m}}\right]\bar{P}_0, \quad j \in E_0^{\infty}. \tag{9.68}$$

Заметим, что по определению (9.2) n — количество ЭМ основной подсистемы ВС со структурной избыточностью, следовательно, n является предельно допустимым числом исправных машин, при котором производительность системы остается не ниже требуемого уровня. По аналогии с n в рамках данной стохастической модели целесообразно ввести $\bar{n}$ — максимальное количество отказавших ЭМ, при котором производительность ВС не ниже предельно допустимой. Следовательно, если ВС находится в состоянии $j \in E_0^{\bar{n}}$, где $E_0^{\bar{n}} = \{0, 1, 2, \ldots, \bar{n}\}$, $E_0^{\bar{n}} \subset E_0^{\infty}$, то будем считать, что она готова к выполнению основных функций по обработке информации. Тогда коэффициент готовности ВС рассчитывают по формуле

$$S = \sum_{j=0}^{\bar{n}} \bar{P}_j,$$

где величины $\bar{P}_j$ определяются из (9.67) и (9.68).

Зная вероятности $\bar{P}_j$, $j \in E_0^{\infty}$, легко вычислить не только коэффициент S готовности ВС, но и ряд других количественных характеристик надежности ВС для стационарного режима работы. Среднее количество отказавших машин ВС, ожидающих начала восстановления,

$$M_1 = \frac{\Lambda}{m\mu(1-\Lambda/m\mu)^2}\bar{P}_m$$

при условии $\Lambda \leqslant m\mu$. Математическое ожидание количества отказавших машин системы

$$M_2 = M_1 + \overline{P}_0 \sum_{j=1}^{m-1} \frac{1}{(j-1)!}\left(\frac{\Lambda}{\mu}\right)^j + \frac{m\overline{P}_m}{1-\Lambda/m\mu}.$$

Среднее число свободных восстанавливающих устройств

$$M_3 = \overline{P}_0 \sum_{j=0}^{m-1} \frac{m-j}{j!}\left(\frac{\Lambda}{\mu}\right)^j$$

Вероятность того, что все восстанавливающие устройства заняты ремонтом отказавших машин ВС,

$$\Pi = \overline{P}_m (1-\Lambda/m\mu)^{-1}$$

Закон распределения времени η ожидания начала восстановления отказавшей ЭМ (в стационарном режиме) или вероятность того, что время пребывания отказавшей ЭМ в очереди на ремонт больше t,

$$P\{\eta > t\} = \Pi e^{-(m\mu-\Lambda)t}$$

Математическое ожидание и дисперсия времени ожидания начала восстановления отказавшей ЭМ вычислительной системы соответственно равны

$$M\eta = \Pi/(m\mu-\Lambda); \quad D\eta = M\eta[2/(m\mu-\Lambda) - M\eta].$$

В заключение следует подчеркнуть, что $N \to \infty$ является парадигмой для построения стохастической модели функционирования большемасштабных ВС. В реальных системах N безусловно конечно, но при массовом параллелизме как N, так и n являются большими числами. Более того, производительность любой такой ВС оценивается (9.2) величиной $n\omega$, которая близка к суммарной пиковой производительности $N\omega$ (см. разд. 3.3.3), ω — производительность одной ЭМ (см. § 2.6). Основываясь на этом факте, при анализе готовности большемасштабных ВС можно считать, что $\overline{n} = N - n$, $\Lambda = N\lambda$.

9.7. Потенциальный контроль вычислительных систем

Работоспособные ресурсы в ВС выявляются с помощью средств контроля и диагностики. При этом *контроль* системы (или ее части — подсистемы) позволяет установить факт работоспособности или неработоспособности проверяемых ресурсов. *Диагностика* же (после установления неработоспособности ВС или ее подсистемы) позволяет определить, какой из

ресурсов системы неработоспособен. В качестве ресурсных элементов в ВС выступают, как правило, ЭМ или элементарный процессор.

Контроль и диагностику отдельной элементарной машины ВС можно произвести так же, как традиционной ЭВМ. Это следует из того, что каждая ЭМ систем есть композиция ЭВМ и локального коммутатора или системного устройства (см. гл. 7). Для контроля и диагностики в пределах ЭМ можно выделить контрольно-диагностическое ядро. При этом необходимым условием осуществления контроля и диагностики ЭМ будет работоспособность ядра. Проверка работоспособности ядра ЭМ выполняется специальными средствами. Остальная часть ЭМ проверяется с помощью контрольно-диагностических тестов (программ), выполняемых ядром. Ясно, что эффективность средств контроля и диагностики будет тем выше, чем будет меньше объем ядра по отношению к проверяемой части машины.

В вычислительной системе как совокупности из N ЭМ, взаимодействующих между собой через программно-настраиваемую сеть связей, может быть применен нетрадиционный подход к контролю и диагностике. Программируемость структуры позволяет в пределах ВС организовывать такие топологически различные схемы из ЭМ, которые обеспечат полноту контроля работоспособности всех ресурсов системы и полноту диагностики неисправностей (т. е. отыскания всех неработоспособных ЭМ с помощью взаимопроверок). Следовательно, по отношению к ВС вполне допустимо применение терминов «самоконтроль» и «самодиагностика».

Для самоконтроля ВС могут быть применены как универсальные, так и проблемно-ориентированные контрольные тесты. Проблемно-ориентированные тесты учитывают специфику сферы применения ВС, структуру решаемых задач, частоту использования тех или иных команд и т. п. Контрольным тестом, адекватным структуре решаемой задачи (т. е. обеспечивающим полноту проверки работоспособности системы, достаточную для решения задачи), будет, например, тест, являющийся наиболее характерной частью параллельной программы. Проблемно-ориентированные тесты существенно проще универсальных. Применение частей параллельной программы в качестве контрольных тестов не требует дополнительной емкости памяти для их хранения. Состояние работоспособности или неработоспособности ресурсов конкретной подсистемы (или в пределе машины) устанавливается после сравнения результатов выполнения контрольного теста в нескольких однородных подсистемах (в нескольких ЭМ).

Возникает вопрос: в какие моменты времени должен производиться (само)контроль ВС? Ясно, что выбор моментов (само)контроля находится в прямой зависимости от надежности ВС. Надежность ВС определяется общим числом N машин, числом ЭМ в основной подсистеме, параметрами λ, Λ, m, μ (см. § 9.6) и может достигать сколь угодно высокого уровня. Но

из последнего не должно следовать, что контроль правильности работы машин ВС должен производиться через очень большие промежутки времени. В самом деле, число работоспособных ЭМ может быть долгое время не меньше числа $n \leqslant N$, но среди отказавших могут быть машины, участвующие в решении сложной задачи (представленной параллельной программой из n ветвей). С другой стороны, при выборе моментов контроля машин системы нельзя ориентироваться лишь только на надежность одной ЭМ. Действительно, в условиях отсутствия структурной избыточности надежность одной ЭМ всегда выше надежности подсистемы из n машин. Таким образом, при выборе моментов (само)контроля ЭМ вычислительных систем необходимо ориентироваться на надежность подсистемы из n ЭМ, $n \leqslant N$.

При оценке надежности подсистемы воспользуемся средним временем безотказной работы θ (9.8), (9.20). Поскольку допускается, что отказ даже одной ЭМ приводит к отказу подсистемы, то $\theta = (n\lambda)^{-1}$. Будем считать, что контроль правильности работы элементарных машин ВС будет достаточно эффективным, если он будет производиться через время θ.

Заметим, что техническая сложность элементарной машины ВС имеет тот же порядок, что и для (электронной части) ЭВМ. Следовательно, уровни надежности элементарных машин ВС и ЭВМ имеют один и тот же порядок:

$$\lambda^{-1} = 10^{-5} \ldots 10^{-8} \text{ ч.}$$

Если число ЭМ в подсистеме $n = 10 \ldots 10^{6}$, то

$$\theta = 10^{-1} \ldots 10^{7} \text{ ч.}$$

Таким образом, через время θ в каждой машине ВС должен пропускаться контрольный тест (общий для ЭМ одной подсистемы).

Оценим время, которое допустимо для реализации контрольных и диагностических тестов в ВС. Вполне естественно, что необходимо стремиться к тому, чтобы время $\Delta\theta$, расходуемое в ВС на выполнение контрольного и диагностического тестов, было как можно меньше (но достаточное для полноты контроля и локализации неисправностей). По-видимому, удовлетворительным для практики будет время $\Delta\theta \leqslant 0{,}001\theta$. Ориентируясь на современный уровень ВТ, легко заметить, что

$$\Delta\theta \leqslant 10^{-4} \ldots 10^{4} \text{ ч.}$$

При таком неравенстве построение и контрольных, и диагностических тестов для ЭМ не вызывает трудностей. Чем больше средняя скорость работы контрольных и диагностических тестов, тем выше не только интенсивность восстановления μ ЭМ, но и производительность ВС.

При выполнении (само)диагностики ВС, во-первых, в качестве ядра могут быть использованы любые работоспособные ЭМ и в пределе ядро любой произвольно выбранной машины и, во-вторых, выбор ядра ВС и определение его работоспособности могут быть проведены автоматически (с помощью средств системы).

Заключение о работоспособности или неработоспособности отдельных ЭМ системы принимается коллективно всеми машинами на основе сопоставления их индивидуальных заключений о работоспособности соседних с ними машин. Достоверность такого заключения хотя бы относительно одной выбранной ЭМ достигается при условии, что $\varepsilon \leqslant [(N-1)/2]$, где ε — максимальное число неработоспособных ЭМ в системе; $[x]$ — целая часть числа x. При современном уровне развития ВТ вероятность того, что количество отказавших машин в восстанавливаемой ВС составляет более десятой части общего их количества, близка к нулю. Следовательно, последнее неравенство удовлетворяется практически всегда.

9.8. Численное исследование надежности вычислительных систем

В гл. 3–8 показано то, что процесс развития архитектуры ВС отражает этапы максимизации степени воплощения принципов модели коллектива вычислителей: параллелизма при обработке информации, программируемости структуры и однородности конструкции (см. разд. 3.1.1). Независимо от изначальной архитектурной парадигмы (конвейерной, матричной, мультипроцессорной и др.) группы разработчиков и фирмы-производители параллельных средств информатики перешли на платформу распределенных ВС. Такие ВС характеризуются не только массовым параллелизмом и программной реконфигурируемостью структуры, но и тем, что их основным функционально-конструктивным модулем является ЭМ (см. § 6.5, 6.6 и 7.1).

Распределенные ВС к началу XXI в. стали занимать доминирующее положение. При оценке потенциальных возможностей вычислительных систем будем ориентироваться на ВС с программируемой структурой (см. гл. 7), так как они обладают максимальной архитектурной гибкостью.

Целью численного исследования надежности ВС является выявление зависимости их показателей качества функционирования от следующих параметров: N — количества элементарных машин; $N-n$ — количества «избыточных» машин; λ — интенсивности потока отказов в одной ЭМ; m — количества (виртуальных) восстанавливающих устройств; μ — интенсивно-

сти восстановления машин одним ВУ; i — начального состояния (числа исправных ЭМ в момент времени $t = 0$).

При численном анализе надежности переходного режима функционирования ВС будем использовать функции $R(t)$, $U(t)$ и $S(t)$ соответственно надежности, восстановимости и готовности ВС (9.6), (9.7), (9.10). Методика вывода расчетных формул для этих показателей совсем не проста, и она находится за границами изучаемого здесь предмета. Именно поэтому в § 9.4 была изложена лишь схема для расчетов, основанная на методах теории массового обслуживания и вычислительной математики. Читателям, которые проявляют повышенный интерес к методике расчета показателей качества функционирования систем в переходном режиме, могут быть рекомендованы для изучения Приложение 2 и [5, 24].

В соответствии с принятой в данной книге концепцией представления материала изучение надежности систем начнем с простейших ВС и завершим его системами с массовым параллелизмом. Это позволит читателю ознакомиться с этапами совершенствования архитектуры ВС и оценить потенциальные пределы их эффективности.

9.8.1. Надежность вычислительных систем «Минск-222»

Вычислительная система «Минск-222» (см. § 7.3) конфигурировалась из ЭВМ второго поколения «Минск-2» (либо «Минск-22»). Ясно, что в современных условиях надежность таких ЭВМ оценивается как весьма низкая. Численный анализ функционирования ВС «Минск-222» представляет интерес и в настоящее время для специалистов — создателей параллельных средств обработки информации. В самом деле, результаты такого анализа позволят:

- установить потенциальные пределы снизу для показателей надежности вычислительных систем;
- ответить на вопрос: могут ли быть созданы высоконадежные системы из низконадежных ЭМ или какова цена достижения уровня надежности ВС, который не ниже надежности одной из составляющих их машин.

Прежде всего приведем числовые значения показателей надежности ЭВМ «Минск-2»: $\lambda = 0,024$ 1/ч, $\mu = 0,7$ 1/ч. Если же учитывались только отказы устройства управления, арифметического устройства и магнитного ОЗУ ЭВМ «Минск-2», то параметры принимают следующие значения: $\lambda = 0,0084$ 1/ч, а $\mu = 0,91$ 1/ч. Приведенные значения для интенсивностей λ и μ ЭВМ «Минск-2» использованы и для ЭМ ВС «Минск-222», обоснованием чему служит предельная простота системного устройства (см. разд. 7.3.1).

Функция надежности ВС. Функция $R(t)$ надежности ВС информирует пользователей о возможности решения тех или иных задач, точнее, она позволяет оценить сложность задач, доступных для решения на рассматриваемой ВС.

Степень влияния величины $(N-n)$ структурной избыточности и количества m восстанавливающих устройств на значения функции надежности ВС «Минск-222» представлена на рис. 9.3. На этом же рисунке изображена функция надежности ЭМ (2.14) $r(t) = \exp(-0,024t)$.

Видно (см. рис. 9.3), что ценой невысокой избыточности, т. е. при $(N-n) \leqslant 3$ и даже при $m = 1$ можно было достичь в ВС «Минск-222» значений вероятности безотказной работы, превышавших значения функции надежности базовой ЭВМ «Минск-2».

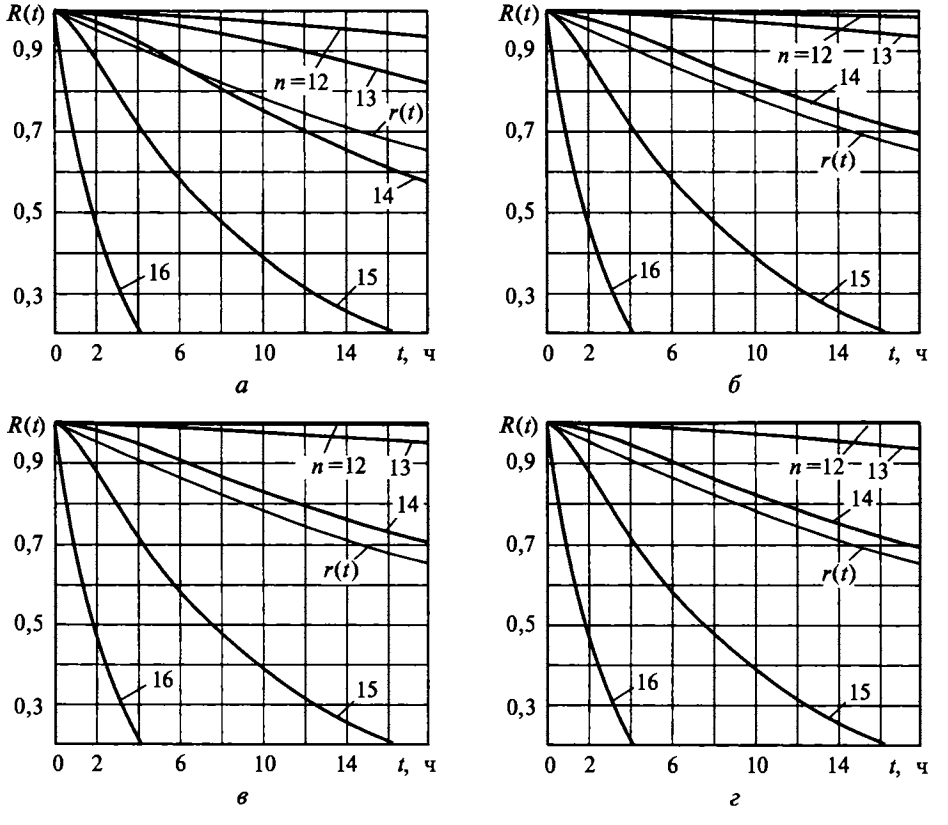


Рис. 9.3. Функция надежности ВС «Минск-222»:

а — $m = 1$; *б* — $m = 2$; *в* — $m = 3$; *г* — $m = 16$; $N = i = 16$; $n = \overline{12,\ 16}$; $\lambda = 0,024$ 1/ч; $\mu = 0,7$ 1/ч

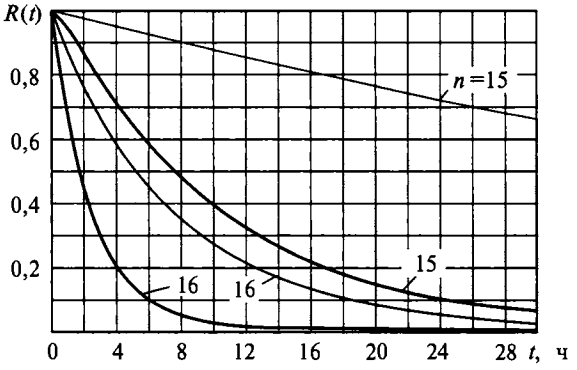


Рис. 9.4. Функция надежности ВС «Минск-222»: $N = i = 16$; $n = 15, 16$; $m = 1$; ——— — $\lambda = 0{,}0084$ 1/ч; $\mu = 0{,}91$ 1/ч; ——— — $\lambda = 0{,}024$ 1/ч; $\mu = 0{,}7$ 1/ч

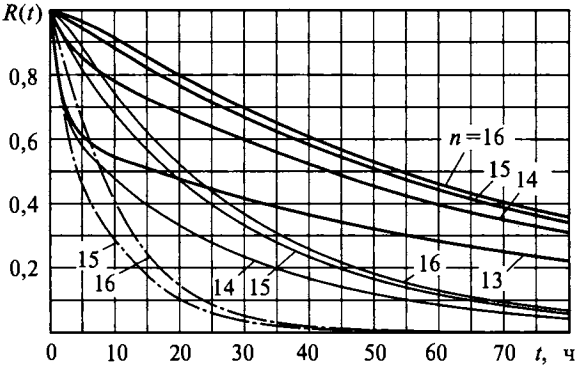


Рис. 9.5. Зависимость функции надежности ВС «Минск-222» от начального состояния:
——— — $N = 16$; $n = 13$, $m = 1$, $i = 13, 14, 15, 16$; $\lambda = 0{,}024$ 1/ч; $\mu = 0{,}7$ 1/ч;
——— — $N = 16$; $n = 14$, $m = 1$, $i = 14, 15, 16$; $\lambda = 0{,}024$ 1/ч; $\mu = 0{,}7$ 1/ч;
–·–·– — $N = 16$; $n = 15$, $m = 1$, $i = 15, 16$; $\lambda = 0{,}024$ 1/ч; $\mu = 0{,}7$ 1/ч

Далее, из анализа графиков рис. 9.3 следует, что даже при использовании низконадежных ЭМ существует предел в наращивании количества восстанавливающих устройств, после которого надежность ВС повышается практически несущественно.

На рис. 9.4 и 9.5 представлены графики, показывающие влияние соответственно параметров λ и μ и начального состояния i на надежность 16-машинной ВС. Из рисунков следует, что увеличение среднего времени λ^{-1} безотказной работы ЭМ и интенсивности μ восстановления отказавших машин приводит к заметному повышению надежности ВС (см. рис. 9.4). Видно также, что надежность ВС резко падает во времени, если в момент начала

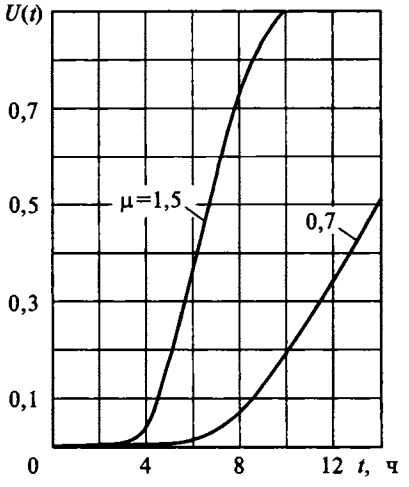


Рис. 9.6. Функция восстановимости ВС «Минск-222»:

$N = 10$; $n = 9$; $m = 1$; $i = 0$; $\lambda = 0{,}024$ 1/ч; $[\mu] = 1/ч$

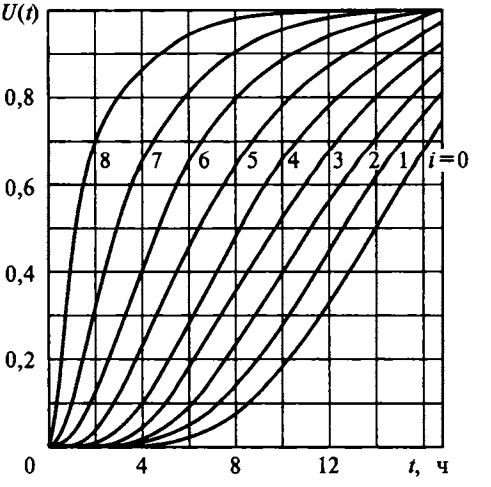


Рис. 9.7. Зависимость функции восстановимости ВС «Минск-222» от начального состояния:

$N = 10$; $n = 9$; $m = 1$; $i = \overline{0, 8}$; $\lambda = 0{,}024$ 1/ч; $\mu = 0{,}7$ 1/ч

функционирования количество отказавших ЭМ было близко к количеству машин, составляющих избыточность.

Функция восстановимости ВС. Функция $U(t)$ есть вероятность того, что при выполнении восстановительных работ в ВС за время t будет достигнут утраченный уровень производительности и на системе можно будет продолжать решение задач.

График зависимости функции восстановимости ВС «Минск-222» от интенсивности μ показан на рис. 9.6. Иллюстрация влияния начального состояния $i \in \{0, 1, \ldots, 8\}$ на значения $U(t)$ представлена на рис. 9.7.

На рис. 9.6 четко видна существенная зависимость значений $U(t)$ от интенсивности восстановления отказавших машин. Вместе с тем анализ кривых, приведенных на рис. 9.3, позволяет выявить наличие «границы» для наращивания числа восстанавливающих устройств ($m \to N$), начиная с которой значения функции надежности ВС увеличивались незначительно. Следовательно, для ВС целесообразно было иметь небольшое (по сравнению с N) количество высокопроизводительных ВУ (т. е. устройств, обладающих высокой интенсивностью восстановления).

Анализ функции $U(t)$ убеждает в том, что в распределенных ВС легко обеспечить практически приемлемые значения показателей восстановимости.

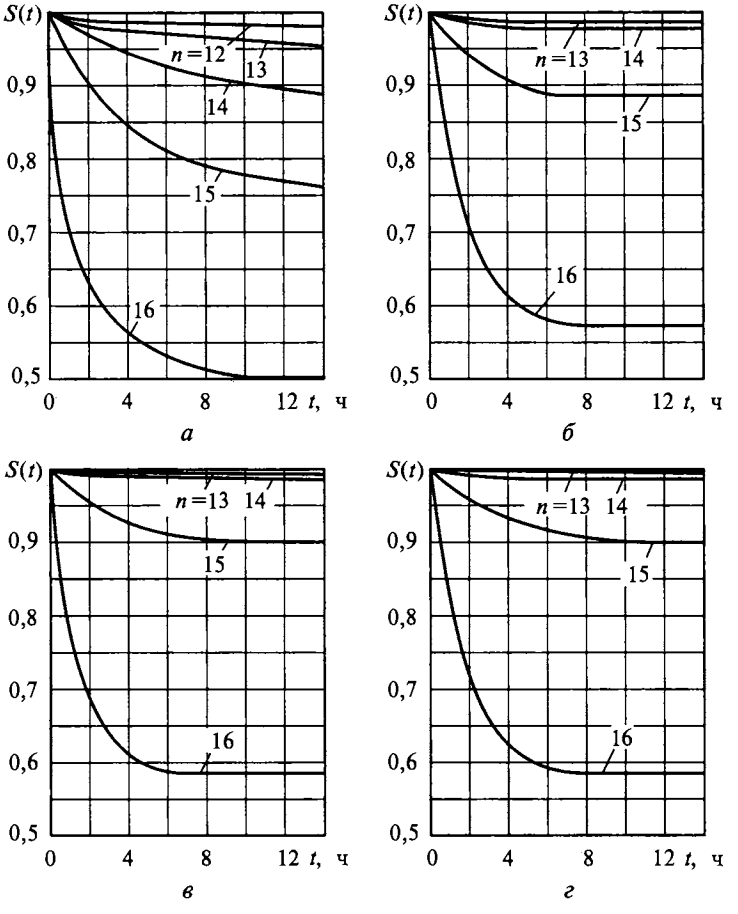


Рис. 9.8. Функция готовности ВС «Минск-222»:

а — $m = 1$; *б* — $m = 2$; *в* — $m = 3$; *г* — $m = 16$; $N = i = 16$; $n = \overline{12, 16}$; $\lambda = 0{,}024$ 1/ч; $\mu = 0{,}71$ 1/ч

Функция готовности ВС. Функция $S(t)$ готовности ВС позволяет судить об уровне производительности ВС в момент поступления задачи, будет ли производительность ВС достаточной для решения задачи.

Зависимость функции готовности ВС «Минск-222» от количества восстанавливающих устройств показана на рис. 9.8. На рисунке видно, что восстанавливаемые ВС могли иметь относительно высокий уровень готовности, даже если они сконфигурированы из низконадежных ЭВМ. Кроме того, системы сравнительно быстро входят в стационарный режим работы.

Уровни готовности ВС «Минск-222» ($N = 16$, $n = \overline{11, 16}$, $\lambda = 0{,}024$ 1/ч, $\mu = 0{,}7$ 1/ч) в стационарном режиме функционирования приведены в табл. 9.1.

Из таблицы видно, что значения коэффициента готовности S были достаточно высоки, если в системе имелись избыточные машины. Кроме того, при точности счета 0,1 % максимумы S для $n = 16$ и $n = 15$ соответствовали $m = 4$, для $n = 14$ и $n = 13$ достигались при $m = 3$, для $n = 12$ — при $m = 2$, наконец, для $n = 11$ — при $m = 1$.

Таблица 9.1

n	m					
	1	2	3	4		16
16	0,485	0,577	0,582	0,583	...	0,583
15	0,751	0,894	0,902	0,903	...	0,903
14	0,887	0,975	0,985	0,985	...	0,985
13	0,953	0,995	0,998	0,998	...	0,998
12	0,982	0,999	1,000	1,000	...	1,000
11	0,993	1,000	1,000	1,000	...	1,000

Эмпирические неравенства для выбора m и $(N-n)$ в распределенных вычислительных системах

В распределенных ВС, построенных из средств ЭВМ второго поколения, при выборе количества восстанавливающих устройств и величины избыточности достаточно было воспользоваться следующими эмпирическими соотношениями:

$$m \leqslant]0,1N[, \quad (N-n) \leqslant]0,1N[, \tag{9.69}$$

где $]x[$ — такое ближайшее к x целое число, что $]x[\geqslant x$. Это следует из того, что иметь числа m или $(N-n)$ более $]0,1N[$ было экономически нецелесообразно, так как их увеличение от $]0,1N[$ соответственно до N и $N-1$ не приводило к существенному повышению надежности ВС.

Таким образом, распределенные ВС могут обладать высокой надежностью, если они сконфигурированы даже из низконадежных ЭМ. При этом высокие уровни надежности, восстановимости и готовности ВС достигаются при незначительных ценах восстанавливающих систем и структурной избыточности.

Функции оперативной надежности и восстановимости ВС. Вероятность $R^*(t)$ позволяет оценить возможности решения задач на неабсолютно надежных восстанавливаемых ВС, находящихся длительное время в эксплуатации (9.13).

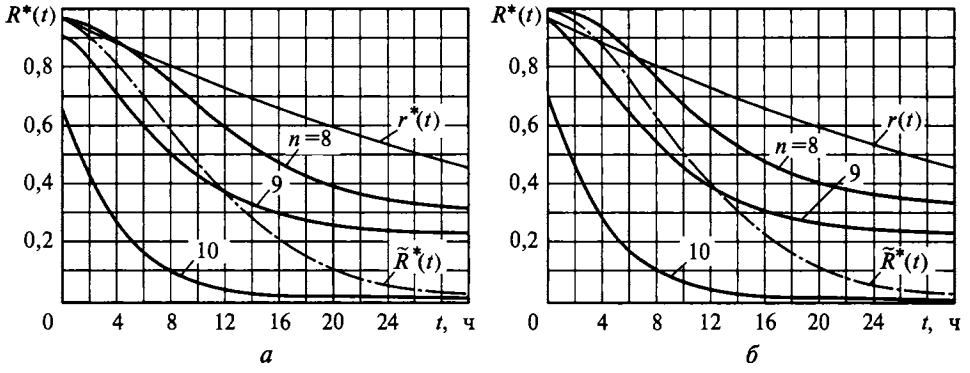


Рис. 9.9. Функция оперативной надежности ВС «Минск-222»:

$а$ — $m = 1$; $б$ — $m = 10$; $N = 10$; $n = \overline{8, 10}$

Проиллюстрируем зависимость функции оперативной надежности $R^*(t)$ ВС «Минск-222» от ее параметров. На рис. 9.9 представлены значения функции $R^*(t)$ системы при $N = 10$, $n = 8, 9, 10$, $m = 1$ и $m = 10$, оценка $\tilde{R}^*(t)$ при $n = 8$ (9.40) и функция оперативной надежности ЭМ для стационарного режима:

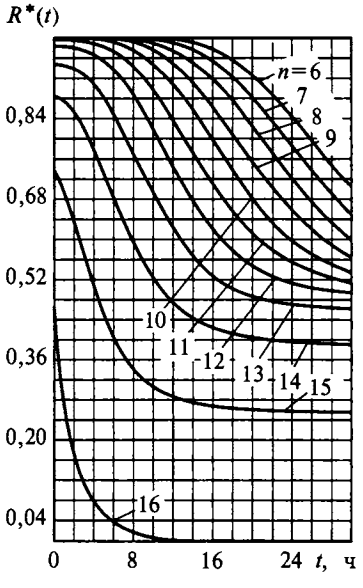


Рис. 9.10. Функция оперативной надежности ВС «Минск-222»:

$N = 16$; $n = \overline{6, 16}$; $m = 1$

$$r^*(t) = P_1 r(t) = \frac{\mu}{\lambda + \mu} e^{-\lambda t},$$

где P_1 — вероятность того, что в стационарном режиме ЭМ исправна (2.19), $r(t)$ — функция надежности ЭМ (2.14). Функция $R^*(t)$ оперативной надежности ВС для $N = 16$, $n = \overline{6, 16}$, $m = 1$ приведена на рис. 9.10. На рис. 9.9 и 9.10 видно, что в ВС могут быть созданы структуры, надежность которых в стационарном режиме даже выше надежности одной ЭМ.

Из рис. 9.9 следует, что число m восстанавливающих устройств влияет на $R^*(t)$ незначительно. Более наглядно это отражено на рис. 9.11, где приведены значения функции оперативной надежности ВС «Минск-222» при $N = 16$, $n = 13$ и $m = \overline{1, 16}$.

Практически в многомашинных ВС выбирать пределы сверху для числа m и величины избыточности $(N-n)$ при существующих параметрах λ и μ можно по формулам (9.69).

Таким образом, в ВС могут быть организованы конфигурации, обладающие высокой надежностью в стационарном режиме функционирования.

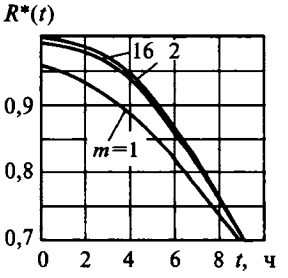


Рис. 9.11. Функция оперативной надежности ВС «Минск-222»:

$N = 16$; $n = 13$; $m = \overline{1, 16}$

Функция $U^*(t)$ информирует о вероятности восстановления требуемого уровня производительности неабсолютно надежной ВС, находящейся в длительной эксплуатации (9.14).

Воспользуемся формулами (9.38), (9.39), (9.43) и (9.44) для расчета функции $U^*(t)$ системы «Минск-222». На рис. 9.12 представлена функция $U^*(t)$ оперативной восстановимости ВС (9.43). На рисунке видно, что $\tilde{U}^*(t)$ несущественно отличается от $U^*(t)$. Поэтому, несмотря на то, что $\tilde{U}^*(t)$ является оценкой сверху для $U^*(t)$, в практических расчетах можно использовать формулу (9.44), которая проще (9.43). На рис. 9.12 изображена также стационарная функция восстановимости ЭМ:

$$u^*(t) = 1 - P_0[1 - u(t)] = 1 - \frac{\lambda}{\lambda + \mu} e^{-\mu t},$$

где P_0 — вероятность того, что в стационарном режиме работы ЭМ неисправна; $u(t)$ — функция восстановимости машины (2.18). На рисунке видно, что

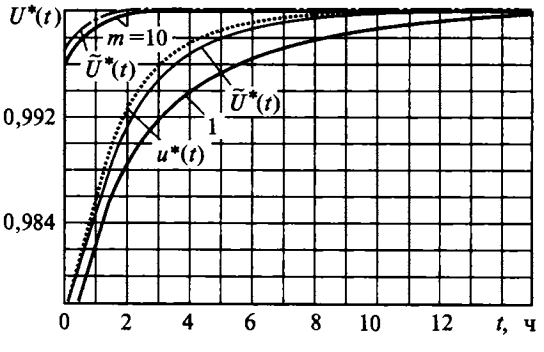


Рис. 9.12. Функция оперативной восстановимости ВС «Минск-222»:
$N = 10$; $n = 8$

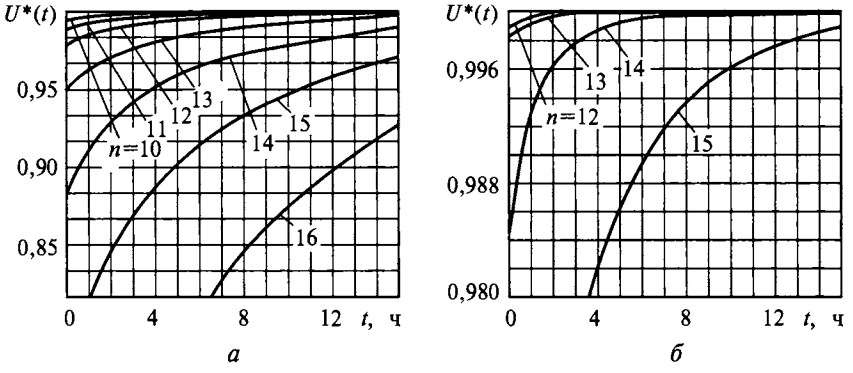


Рис. 9.13. Функция оперативной восстановимости ВС «Минск-222»:

а — $m = 1$; *б* — $m = 16$; $N = 16$; $n = \overline{10, 16}$

восстановимость элементарной машины выше, чем у систем при $m = 1$, однако меньше восстановимости системы, у которой количество восстанавливающих устройств равно количеству ЭМ ($m = N$) и у которой имеется избыточность.

Графики функции оперативной восстановимости 16-машинной ВС «Минск-222» для $n = \overline{10,\ 16}$, $m = 1$ и $m = 10$ приведены на рис. 9.13.

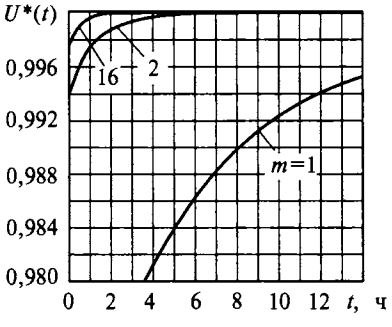


Рис. 9.14. Функция $U^*(t)$ системы «Минск-222»:

$N = 16$; $n = 13$; $m = \overline{1, 16}$

Зависимость функции $U^*(t)$ системы «Минск-222» ($N = 16$, $n = 13$) от числа m восстанавливающих устройств представлена на рис. 9.14.

По рис. 9.13 и 9.14 можно установить, что для выбора числа m, которое бы обеспечивало вполне достаточный уровень восстановимости ВС, необходимо воспользоваться формулой (9.69).

Результаты изучения графиков $U^*(t)$ убеждают в том, что ВС в стационарном режиме работы могут обладать высокой восстановимостью (которая не ниже вероятности восстановления одной ЭМ).

9.8.2. Надежность мини-ВС и микроВС

Функциональные структуры и архитектурные свойства мини-ВС описаны в § 6.2 (C.mmp), § 7.4 (МИНИМАКС) и § 7.5 (СУММА). Мини-ВС — это системы, конфигурируемые из средств мини-машинной техники. Отече-

ственные системы МИНИМАКС и СУММА относились к распределенным и имели программируемую структуру. Элементарные машины первой ВС формировались, в частности, из мини-ЭВМ М-6000, а второй — из машин «Электроника-100И». Несмотря на разницу в значениях таких характеристик, как быстродействие, разрядность слов и емкость памяти, ЭМ систем и МИНИМАКС, и СУММА имели одни и те же показатели надежности:

$$\lambda = 10^{-2}...10^{-3}\ \ 1/\text{ч};\ \ \mu = 0{,}1...1{,}0\ \ 1/\text{ч}.$$

Диапазоны значений для λ и μ установлены в результате анализа существовавших возможных вариантов компоновки элементарных машин.

Известно (см. § 6.6 и 7.6), что генерациями, следовавшими за мини-ВС, являлись микроВС. Базовыми средствами для формирования микроВС были микроЭВМ. Архитектуры мини-ВС и микроВС (и по свойствам, и по количественным характеристикам) были достаточно близки. В самом деле, границы для количества ЭМ и для их параметров эффективности остались неизменными. В микроЭВМ в отличие от мини-ЭВМ использовались интегральные схемы (ИС) с малой степенью интеграции, а не узлы из дискретных элементов. Последнее привело к миниатюризации самой машины, к ее удешевлению, но показатели производительности и надежности остались в тех же диапазонах, что и для мини-ЭВМ.

Отразим результаты численного анализа надежности микромашиной ВС МИКРОС-1. Параметры λ и μ для возможных конфигураций, составляющих основу ЭМ, представлены в табл. 9.2.

Таблица 9.2

Номер конфигурации l	Конфигурация микроЭВМ или комплекса	λ, 1/ч	μ, 1/ч
1	МикроЭВМ «Электроника 60М»	0,001	1,0
2	Комплекс (15 ВУМС-28-026) на базе микроЭВМ «Электроника 60М»	0,003	0,5
3	Расширенный комплекс (15 ВУМС-28-026) на базе микроЭВМ «Электроника 60М»	0,004	0,5
4	МикроЭВМ «Электроника НЦ 80-01 Д»	0,0002	1,0
5	Комплекс ДВК	0,002	0,5

Графики для функции $R(t)$ надежности ВС МИКРОС-1 представлены на рис. 9.15–9.18. Кривые рис. 9.15 характеризуют при $m = 1$ временную зависимость вероятности безотказной работы ВС от конфигураций ЭМ. Семейство кривых, изображенных пунктирными линиями, соответствует

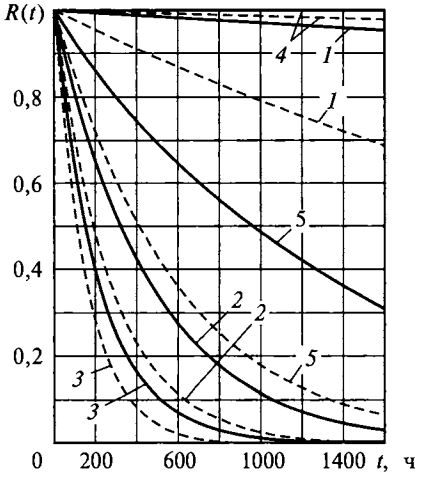


Рис. **9.15.** Функции надежности ВС МИКРОС-1:

------ — $N = i = 16$, $n = 15$, $m = 1$;

——— — $N = i = 32$, $n = 30$, $m = 1$

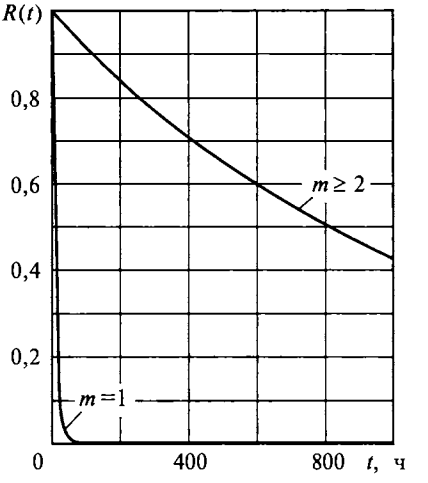


Рис. 9.16. Функция надежности ВС МИКРОС-1:

$N = i = 128$; $n = 126$; $m = \overline{1, 128}$;

$\lambda = 0{,}001$ 1/ч; $\mu = 1{,}0$ 1/ч

16-машинным ВС, в которых только одна ЭМ составляет избыточность ($N = i = 16$, $n = 15$). Второе семейство (сплошные линии) относится к ВС, каждая из которых сформирована из 32 ЭМ и имеет избыточность из двух машин ($N = i = 32$, $n = 30$). В качестве базовых средств при компоновке ЭМ при $l = 1$ использовались микроЭВМ «Электроника 60М», при $l = 2$ — вычислительные комплексы 15 ВУМС-28-026, при $l = 3$ — расширенные комплексы 15 ВУМС-28-026, при $l = 4$ — микроЭВМ «Электроника НЦ 80-01Д» и при $l = 5$ — комплексы ДВК-2.

Таким образом, и 16-машинные, и 32-машинные конфигурации ВС обладали достаточно высокой надежностью (для 70-х годов XX в.). Усложнение состава ЭМ, безусловно, снижало надежность ВС в целом, однако она оставалась достаточной для параллельного моделирования и решения задач, не доступных для отдельно взятой микроЭВМ. Анализ графиков рис. 9.15 убеждает в том, что система МИКРОС-1 не уступала по надежности ВС *C.mmp* и *Cm** (см. § 6.2, 6.6). Вместе с этим ВС МИКРОС-1 обладала бóльшей архитектурной гибкостью, способностью к наращиванию вычислительных ресурсов (в то время как мини-ВС *C.mmp* была рассчитана на фиксированное количество машин, которое не превышало 16).

На рис. 9.16 приведена зависимость функции надежности ВС МИКРОС-1 ($N = i = 128$, $n = 126$) от количество m восстанавливающих устройств ($m = \overline{1, 128}$). В качестве ЭМ взята конфигурация на базе микроЭВМ «Электроника 60М» ($\lambda = 0,001$ 1/ч, $\mu = 1,0$ 1/ч). Кривые рис. 9.17 отражают временну́ю зависимость вероятности безотказной работы ВС ($N = i = 128$, $m = 1$, $\lambda = 0,001$ 1/ч, $\mu = 1,0$ 1/ч) от числа основных машин $n \in \{123, 124, \ldots, 128\}$ или, что то же самое, от числа избыточных ЭМ $(N - n) \in \{0, 1, \ldots, 5\}$. Кроме того, на рис. 9.17 представлена и вероятность $r(t)$ безотказной работы ЭМ (2.14) пунктирной линией. Очевидно, что введение избыточности существенно повышает надежность ВС. Анализ графиков рис. 9.16 и 9.17 не только подтверждает справедливость эмпирических неравенств (9.69), но и усиливает их.

На рис. 9.18 приведена зависимость вероятности безотказной работы системы МИКРОС-1 от общего числа машин; $N = i \in \{64, 128, 256, 512\}$; $m = 1$; $\lambda = 0,001$ 1/ч; $\mu = 1,0$ 1/ч при фиксированной относительной избыточности. Пунктирные линии относятся к конфигурациям ВС, в которых избыточность оценивается максимальным числом ЭМ, не превышающим 1 % от общего числа машин. Сплошные линии соответствуют ВС при избыточности, близкой к 2 % от N. Из анализа кривых следует, что при фиксированной относительной избыточности надежность ВС повышается с увеличением общего числа ЭМ.

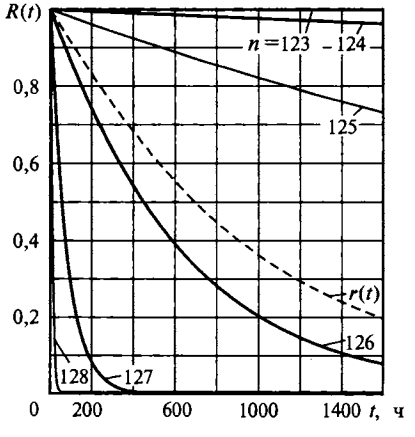


Рис. 9.17. Функция надежности ВС МИКРОС-1:

$N = i = 128$; $m = 1$; $n = \overline{123, 128}$;
$\lambda = 0,001$ 1/ч; $\mu = 1,0$ 1/ч

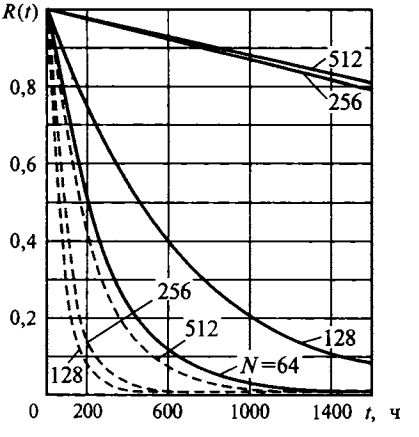


Рис. 9.18. Функция надежности ВС МИКРОС-1:

$N = i = 64, 128, 256, 512$; $m = 1$;
--------- — $(N - n) \leqslant 0,01N$;
——— — $(N - n) \leqslant 0,02N$

9.8.3. Надежность большемасштабных распределенных вычислительных систем

Современные большемасштабные распределенные ВС или ВС с массовым параллелизмом являются ансамблями из значительного числа N элементарных машин или процессоров, $N \leqslant 10^6$ Естественно, возникает вопрос: как в таких ансамблях достичь уровня надежности, характерного для отдельно взятой ЭМ? Ориентиром здесь может служить среднее время $\vartheta = \lambda^{-1}$ безотказной работы ЭВМ (2.11). В начале XXI столетия среднее время безотказной работы электронной части персонального компьютера (в частности, и с архитектурой IBM PC) достигло нескольких лет: $\vartheta = 50\,000$ ч.

Элементной базой для построения ВС служат микропроцессоры — интегральные схемы с большим количеством компонентов. Надежность микропроцессорных БИС оценивается средним временем ϑ безотказной работы в диапазоне $10^5 \ldots 10^8$ ч.

При численном изучении потенциальной надежности большемасштабных ВС будем использовать следующие параметры для ЭМ:

$$\lambda = 10^{-5} \ldots 10^{-8} \text{ 1/ч}; \quad \mu^{-1} = 0,001 \ldots 0,25 \text{ ч}.$$

Диапазон значений для среднего времени восстановления μ^{-1} ЭМ одним ВУ установлен в результате анализа временны́х возможностей виртуальных восстанавливающих устройств (точнее: контроле́ров, диагностов и реконфигураторов; см. рис. 9.2) для распределенных ВС.

При анализе надежности большемасштабных ВС будет применяться наиболее распространенный в инженерной практике показатель — среднее время θ безотказной работы (9.8).

На рис. 9.19–9.21 изображены графики зависимости значений математического ожидания времени безотказной работы большемасштабных ВС от параметров N, m и $N - n$. Так, на рис. 9.19 представлены значения θ для ВС со следующими параметрами: $N = 65536$; $N - n = \overline{0,9}$; $m \geqslant 1$; $\lambda = 10^{-5}$ 1/ч. Эти значения позволяют установить степень влияния интенсивности μ восстановления ЭМ и количества m ВУ на среднее время θ безотказной работы ВС. Так, семейство сплошных линий показывает, что варьирование интенсивности μ в промежутке от 4 1/ч до 1000 1/ч даже при $m = 1$ приводит к существенному улучшению качества функционирования ВС с большим количеством ЭМ.

Пунктирная линия соответствует значениям среднего времени ϑ безотказной работы ВС для количества восстанавливающих устройств $m \geqslant 2$ (каждое из которых обладает интенсивностью $\mu = 4$ 1/ч).

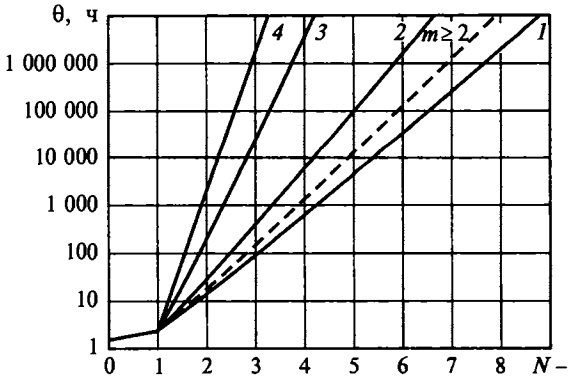


Рис. 9.19. Среднее время безотказной работы большемасштабных ВС:
$N = 65\,536$; $m \geqslant 1$; $\lambda = 10^{-5}$ 1/ч; *1* — $\mu = 4$ 1/ч; *2* — $\mu = 10$ 1/ч; *3* — $\mu = 100$ 1/ч; *4* — $\mu = 1000$ 1/ч

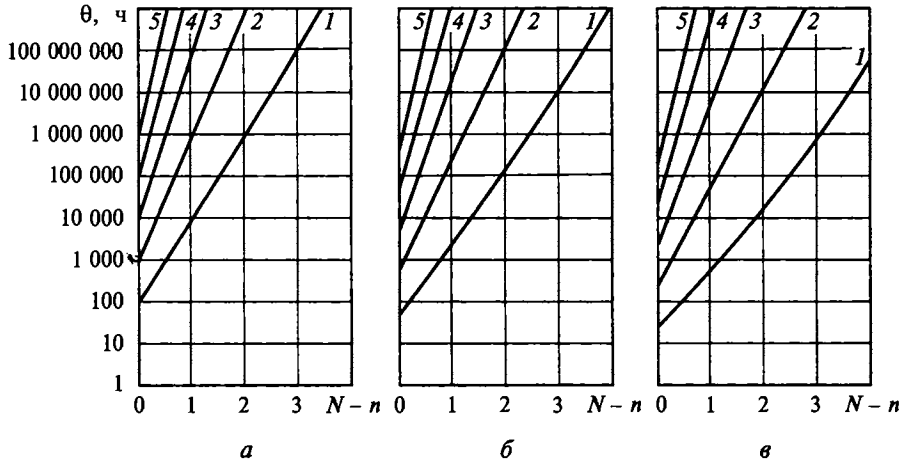


Рис. 9.20. Среднее время безотказной работы большемасштабных ВС:
$m = 1$; $\lambda = 10^{-5} \ldots 10^{-9}$ 1/ч; $\mu = 4$ 1/ч; *а* — $N = 1024$; *б* — $N = 2048$; *в* — $N = 4096$; *1* — $\lambda = 10^{-5}$ 1/ч; *2* — $\lambda = 10^{-6}$ 1/ч; *3* — $\lambda = 10^{-7}$ 1/ч; *4* — $\lambda = 10^{-8}$ 1/ч; *5* — $\lambda = 10^{-9}$ 1/ч

Видно, что при количестве избыточных ЭМ $(N - n) \geqslant 6$ (т. е. при избыточности в машинах, равной 0,009 %) достигается уровень надежности ВС, который не ниже, чем для одной ЭМ. Кроме того, имеется граница для количества ВУ, после которой увеличение m приводит к незначительному росту среднего времени безотказной работы ВС.

Зависимость показателя θ от общего числа N машин большемасштабных ВС и от их избыточности $(N - n)$ иллюстрируют рис. 9.20 и 9.21; $m = 1$, $\mu = 4$ 1/ч.

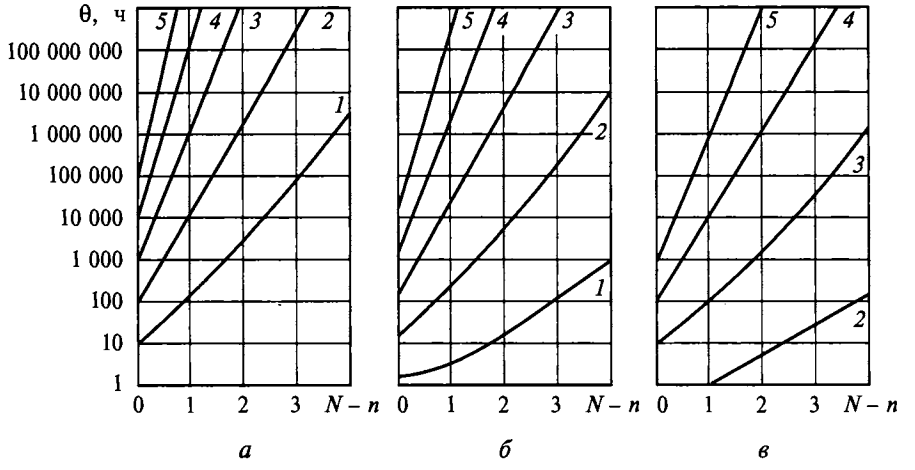


Рис. 9.21. Среднее время безотказной работы большемасштабных ВС:
$m = 1$; $\lambda = 10^{-5}...10^{-9}$ 1/ч; $\mu = 4$ 1/ч; *а* — $N = 8192$; *б* — $N = 65536$; *в* — $N = 1048576$; *1* — $\lambda = 10^{-5}$ 1/ч; *2* — $\lambda = 10^{-6}$ 1/ч; *3* — $\lambda = 10^{-7}$ 1/ч; *4* — $\lambda = 10^{-8}$ 1/ч; *5* — $\lambda = 10^{-9}$ 1/ч

Приведенные на рис. 9.19–9.21 графики позволяют определить не только достигнутый уровень надежности большемасштабных ВС, но и оценить потенциальные возможности перспективных разработок.

Эмпирические формулы для m и $(N-n)$ большемасштабных вычислительных систем

Анализ рис. 9.19–9.21 показывает, что для большемасштабных распределенных ВС выбор количества m восстанавливающих устройств и количества $(N-n)$ избыточных ЭМ следует осуществлять по формулам:

$$1 \leqslant m \leqslant [\lg N];\ 1 \leqslant (N-n) \leqslant [\lg N], \qquad (9.70)$$

где N — количество ЭМ в системе; $[\lg N]$ — число, округленное до ближайшего к $\lg N$ целого числа (снизу или сверху). Формулы (9.70) гарантируют уровень надежности большемасштабной ВС не ниже уровня надежности одной ее ЭМ. При этом с ростом масштабности системы и с увеличением среднего времени безотказной работы одной ЭМ требуемое количество восстанавливающих устройств и избыточных машин относительно общего количества ЭМ асимптотически уменьшается.

Таким образом, за счет улучшения параметров λ и μ относительные цены восстанавливающей системы и избыточности снижаются более чем на порядок по сравнению с ценами для предшествующих поколений ВС (см. (9.69), (9.70) и рис. 9.19–9.21).

Численный анализ надежности ВС позволяет сделать следующие выводы.

1. С позиций надежности распределенные ВС являются перспективными средствами индустрии обработки информации. При современном уровне развития производственно-технологической базы микропроцессорной техники практически возможно построение высоконадежных ВС с количеством ЭМ $10...10^6$, обеспечивающих производительность в диапазоне 10 GFLOPS...1 PFLOPS.

2. Для обеспечения уровня надежности большемасштабной распределенной ВС не менее уровня надежности одной ее ЭМ требуется избыточность в машинах, не превышающая десятичного логарифма числа составляющих ЭМ.

3. Для выполнения восстановительных работ в распределенных ВС, как правило, достаточно иметь одно (виртуальное) восстанавливающее устройство независимо от количества ЭМ. Среднее время восстановления системы с избыточностью при этом имеет тот же порядок, что и среднее время восстановления одной машины.

9.9. Анализ вычислительных систем со структурной избыточностью

1. Вычислительные системы со структурной избыточностью являются обобщением систем с резервом. Со стороны пользователя они выглядят как виртуальные системы, способные реализовать параллельные программы с фиксированным числом ветвей (равным числу основных элементарных машин). С позиций проектировщиков и эксплуатационников предложенные ВС представляют собой программно-настроенные конфигурации в пределах ВС, избыточность которых позволяет достичь любой априори заданной надежности.

2. Рассмотренные показатели надежности для переходного и стационарного режимов работы ВС и описанные инженерные методы их расчета вполне приемлемы для утилитарных целей (для анализа и синтеза ВС).

3. Численный анализ показывает, что при структурной избыточности, равной десятичному логарифму общего количества машин в большемасштабной распределенной ВС, достигается уровень надежности системы, который не ниже уровня надежности ее ЭМ.

Изученные архитектурные концепции (см. гл. 4–7) и описанный в данной главе подход к организации ВС с избыточностью позволяют создавать масштабируемые вычислительные суперсистемы с производительностью от GFLOPS до PFLOPS. При этом в любой суперсистеме достигается надежность, которая не ниже надежности любой из составляющих ЭМ (при цене относительно невысоких затрат).

10. ЖИВУЧЕСТЬ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

В главе изучается одно из замечательных свойств распределенных ВС с программируемой структурой — живучесть (Robustness). Данным свойством ЭВМ не обладают, живучесть — это отличительная особенность средств обработки информации, основанных на модели коллектива вычислителей (см. § 3.1).

Живучесть является более емким понятием, чем надежность ВС. Под живучестью понимается способность ВС (достигаемая программной организацией структуры и функционального воздействия между ее компонентами) в любой момент функционирования использовать суммарную производительность всех исправных ресурсов для решения задач.

Изучение проблемы живучести основывается на парадигме, получившей название живучей ВС и являющейся обобщением ВС со структурной избыточностью (см. § 9.2). В данной главе вводятся показатели потенциальной живучести ВС и предлагается оригинальная методика их расчета, базирующаяся на континуальной стохастической модели. Адекватность модели обосновывается большемасштабностью распределенных ВС или, говоря иначе, их массовым параллелизмом — большим количеством ЭМ (до 10^6). Такая методология имеет свой аналог в механике сплошных сред.

10.1. Живучие вычислительные системы

Живучесть ВС должна достигаться при решении задач, представленных программами с любым допустимым числом параллельных ветвей или, что то же самое, с любым рангом r, $1 \leqslant r \leqslant N$, где N — общее количество ЭМ в системе. Исключение не должны составлять задачи с переменным рангом $\tilde{r}$, допускающим варьирование от r° до r^*, $1 \leqslant r^\circ, r^* \leqslant N$, $r^\circ \leqslant \tilde{r} \leqslant r^*$ Живучесть должна обеспечиваться и в монопрограммном, и в мультипрограммных режимах работы ВС. При монопрограммном режиме для определенности будем полагать, что переменный ранг задачи характеризуется величинами $r^\circ = n$ и $r^* = N$. Величина n одновременно является и нижней границей количества работоспособных ЭМ (считается, что при количестве отказавших

ЭМ, равном $(N - n + 1)$, имеет место полный отказ ВС). При мультипрограммировании сумма рангов одновременно решаемых задач не превышает N.

С позиций анализа живучести ВС мультипрограммные режимы могут быть сведены к монопрограммному. При мультипрограммировании в ВС порождаются подсистемы, число машин в каждой из которых соответствует рангу решаемой задачи. Ясно, что в методическом плане ничто не мешает нам рассматривать каждую из таких подсистем как самостоятельную систему. Итак, не нарушая общности, при исследовании живучести ВС будем считать, что они функционируют в монопрограммном режиме.

Живучесть ВС рассматривается в двух аспектах: потенциальном и структурном. При анализе потенциальной живучести ВС особенности структуры или сети межмашинных связей в прямом виде не учитываются и считается, что в системе обеспечиваются возможности по достижению необходимой связности исправных ЭМ. При изучении структурной живучести ВС, как было показано в разд. 7.2.1, учитываются топологический вид сети межмашинных связей и надежностные характеристики компонентов этой сети.

В современных распределенных ВС единицей вычислительных ресурсов выступает ЭМ («многополюсник», например микропроцессор с локальным коммутатором и линками — связями, или транспьютер, или некое аппаратурно-программное образование, или виртуальная ЭМ). Пусть N — количество однородных ЭМ, составляющих ВС. Это число в современных высокопроизводительных ВС достаточно большое, $10 \leqslant N \leqslant 10^6$. Вычислительные системы, как правило, восстанавливаемые (или даже самовосстанавливаемые или реконфигурируемые). Будем считать, что ремонтные работы в ВС осуществляются некоторой (виртуальной) восстанавливающей системой, состоящей из m устройств (ВУ), $1 \leqslant m \leqslant N$. Каждое виртуальное ВУ в любой момент времени может производить ремонт только одной ЭМ.

Говорят, что ВС находится в состоянии $k \in E_0^N$, $E_0^N = \{0, 1, \ldots, N\}$, если в ней имеется k работоспособных ЭМ. Программы, при реализации которых автоматически устанавливается число параллельных ветвей, равное числу работоспособных машин в текущий момент времени, относятся к *адаптирующимся*. Теоретически и экспериментально установлено, что для достаточно широкого круга задач могут быть составлены эффективные параллельные программы, обладающие способностью адаптации к составу работоспособных вычислительных ресурсов.

Под *живучей ВС* понимается (виртуальная) конфигурация из N ЭМ, в которой:

1) указано минимально допустимое число n работоспособных ЭМ, обеспечивающее производительность системы не менее требуемой;

2) реализована возможность решения сложных задач, представленных адаптирующимися параллельными программами;

3) отказы любых ЭМ (вплоть до числа $N - n$) и восстановления отказавших машин приводят только к увеличению или уменьшению времени реализации параллельной программы;

4) при изменении состояния $k = 0, 1, 2, \ldots, N$ производительность подчиняется следующему закону:

$$\Omega(k) = A_k \Delta(k - n)\varphi(k, \omega), \qquad (10.1)$$

где A_k — коэффициент (в общем случае $A_k \neq A_n$, см. (9.2)); $\varphi(k, \omega)$ — неубывающая функция от k и ω (как правило, $\varphi(k, \omega) = k\omega$ при решении сложных задач, см. разд. 3.3.4).

Следует обратить внимание на то, что в живучей ВС вычислительное ядро составляют все $k \in \{n, n+1, \ldots, N\}$ исправных ЭМ и что число избыточных ЭМ в ней переменное и заключено между 0 и $(N - n)$. В живучей системе нет резервирования, нет простоев исправных машин. Все исправные ЭМ такой ВС включаются в вычислительное ядро и участвуют в реализации параллельных процессов, что приводит к сокращению времени решения задач.

Итак, в живучих ВС отказы ЭМ не приводят к отказу систем в целом. Более того, в таких ВС при выходе машин из строя сохраняется возможность продолжения счета на всех исправных ЭМ (при наличии отказавших вплоть до $(N - n)$). Реализация такого виртуального механизма переменной избыточности машин уменьшает время решения сложной задачи на ВС.

Качественные зависимости производительности $\Omega(k)$ от числа k работоспособных машин для систем со структурной избыточностью и живучих ВС приведены на рис. 10.1.

Будем полагать, что для формирования в системе живучих конфигураций имеются специальные (аппаратно-программные) средства, составляющие *реконфигуратор*. Он предназначается для выполнения следующих функций: исключения из вычислительного ядра отказавших машин и включения в него машин после их восстановления; формирования вычислительного ядра из оставшихся работоспособных ЭМ и вновь отремонтированных машин; преобразования адаптирующейся параллельной программы с целью достигнуть соответствия между количеством ее ветвей и количеством машин вычислительного ядра; вложения преобразованной программы в ядро с новой структурой и организации ее прохождения.

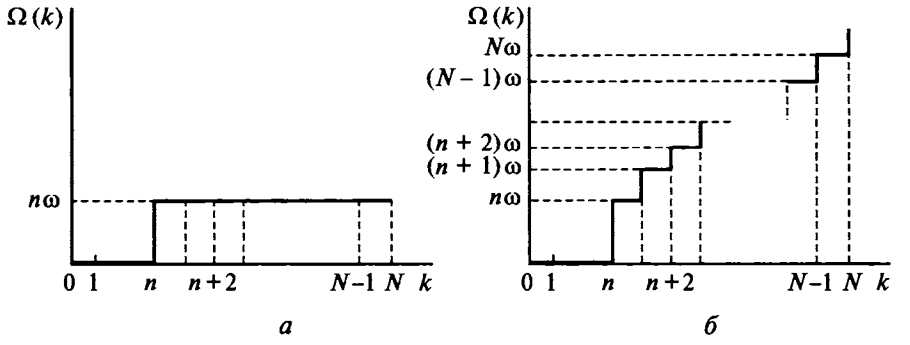


Рис. 10.1. Производительность вычислительных систем:
а — ВС со структурной избыточностью; *б* — живучие ВС

Вычислительные системы с программируемой структурой обладают наиболее гибкими структурными возможностями для такой виртуализации (см. гл. 7).

10.2. Показатели потенциальной живучести вычислительных систем

К набору показателей потенциальной живучести ВС (характеризующих предельные возможности систем) предъявляется комплекс требований, аналогичный набору для показателей надежности (см. § 9.3). Показатели живучести ВС должны учитывать то обстоятельство, что при решении задач используются все исправные ЭМ, число которых не постоянно. Говоря иначе, при определении показателей живучести следует учесть то, что параллельные программы сложных задач при их реализации на живучих ВС способны задействовать суммарную производительность всех работоспособных ЭМ системы.

Качество функционирования живучих ВС будем характеризовать их функциями потенциальной живучести $\mathcal{N}(i, t)$ и занятости восстанавливающей системы $\mathcal{M}(i, t)$, вектор-функциями $\mathbf{R}(t)$, $\mathbf{U}(t)$ и $\mathbf{S}(t)$ ВС. Функции $\mathcal{N}(i, t)$ и $\mathcal{M}(i, t)$ характеризуют в момент времени $t \geqslant 0$ среднюю производительность ВС и среднюю загруженность восстанавливающей системы, если ВС начала функционировать с $i \in E_0^N$ работоспособными ЭМ. Вектор-функции $\mathbf{R}(t)$, $\mathbf{U}(t)$, $\mathbf{S}(t)$ являются обобщениями функций надежности (9.6), восстановимости (9.7) и готовности (9.10) ВС.

Функцией потенциальной живучести ВС назовем отношение

$$\mathcal{N}(i, t) = \bar{\Omega}(i, t)/N\omega, \tag{10.2}$$

где $\bar{\Omega}(i, t)$ — математическое ожидание производительности ВС в момент $t \geqslant 0$ при условии, что в момент начала функционирования в системе было i работоспособных ЭМ, $i \in E_0^N$, $N\omega$ — суммарная производительность всех машин ВС; N — общее количество ЭМ системы; ω — показатель производительности одной ЭМ (см. § 2.6).

Очевидно, что для $\bar{\Omega}(i, t)$ допустимо представление в виде $\bar{\Omega}(i, t) = \mathcal{N}(i, t)\omega$, где $\mathcal{N}(i, t)$ — среднее число работоспособных машин в момент $t \geqslant 0$ при условии, что система начала функционировать в состоянии $i \in E_0^N$ (Заметим, что $\mathrm{N}(i, 0) = i$, $i \in E_0^N$.) Тогда функция потенциальной живучести ВС может быть выражена через $\mathcal{N}(i, t)$:

$$\mathcal{N}(i, t) = \mathcal{N}(i, t)/N. \tag{10.3}$$

Функцией занятости восстанавливающей системы назовем

$$\mathcal{M}(i, t) = \mathcal{M}(i, t)/m, \tag{10.4}$$

где $\mathcal{M}(i, t)$ — математическое ожидание числа занятых восстанавливающих устройств в момент времени $t \geqslant 0$ при условии, что ВС начала функционировать в состоянии $i \in E_0^N$; m — число устройств в восстанавливающей системе.

Вектор-функция

$$\mathbf{R}(t) = \{R_k(t)\},\ k \in E_n^N, \tag{10.5}$$

где координата $R_k(t)$ определяется как вероятность того, что производительность системы, начавшей функционировать в состоянии i, $k \leqslant i \leqslant N$, не менее производительности k машин на всем промежутке времени $[0, t)$, $E_n^N = \{n, n+1, \ldots, N\}$. Учитывая (9.6) и (10.5), запишем:

$$\begin{gathered} R_k(t) = P\{\forall\tau \in [0,\ t) \to \Omega(\tau) \geqslant A_k k\omega \mid k \leqslant i \leqslant N\}; \\ R_k(t) = P\{\forall\tau \in [0,\ t) \to \xi(\tau) \geqslant k \mid k \leqslant i \leqslant N\}; \\ R_k(0) = 1,\ R_k(+\infty) = 0,\ k \in E_n^N, \end{gathered} \tag{10.6}$$

здесь $\Omega(\tau)$ и $\xi(\tau)$ — соответственно производительность ВС и количество исправных машин в системе в момент времени $\tau \in [0, t)$; i — начальное состояние ВС.

Вектор-функция

$$\mathbf{U}(t) = \{U_k(t)\},\ k \in E_n^N, \tag{10.7}$$

где координата $U_k(t)$ является вероятностью того, что в ВС, имеющей начальное состояние i, $0 \leqslant i < k$, будет восстановлен на промежутке времени $[0, t)$ уровень производительности не менее k ЭМ. Основываясь на (9.7) и (10.7), сделаем формальную запись:

$$\begin{gathered} U_k(t) = 1 - P\{\forall\tau \in [0,\ t) \to \Omega(\tau) < A_k k\omega \mid 0 \leqslant i < k\}; \\ U_k(t) = 1 - P\{\forall\tau \in [0,\ t) \to \xi(\tau) < k \mid 0 \leqslant i < k\}; \\ U_k(0) = 0,\ U_k(+\infty) = 1,\ k \in E_n^N. \end{gathered} \tag{10.8}$$

По аналогии с (9.8) и (9.9) можно рассматривать *вектор среднего времени безотказной работы* (средней наработки до отказа)

$$\mathbf{и} = \{\theta_k\},\ \theta_k = \int_0^\infty R_k(t)dt \tag{10.9}$$

и *вектор среднего времени восстановления*

$$\mathbf{T} = \{T_k\},\ T_k = \int_0^\infty t dU_k(t) \tag{10.10}$$

вычислительной системы, $k \in E_n^N$.

Вектор-функция готовности ВС

$$\mathbf{S}(t) = \{S_k(t)\},\ k \in E_n^N, \tag{10.11}$$

где $S_k(t)$ — вероятность того, что в момент времени $t \geqslant 0$ производительность системы, начавшей работать в состоянии $i \in E_0^N$, не менее производительности k ЭМ:

$$\begin{gathered} S_k(t) = P\{\Omega(t) \geqslant A_k k\omega \mid i \in E_o^N\}; \\ S_k(t) = P\{\xi(t) \geqslant k \mid i \in E_0^N\}; \\ S_k(0) = \begin{cases} 1, & \text{если } k \leqslant i \leqslant N; \\ 0, & \text{если } 0 \leqslant i < k; \end{cases} \\ 0 < S(+\infty) < 1. \end{gathered} \tag{10.12}$$

Предельные значения показателей (10.3) и (10.4) при $t \to \infty$ будут характеризовать потенциальную живучесть ВС в стационарном режиме работы. Пределы

$$\mathcal{N} = \lim_{t\to\infty} \mathcal{N}(i, t); \tag{10.13}$$

$$\mathcal{M} = \lim_{t\to\infty} \mathcal{M}(i, t), \tag{10.14}$$

не зависящие от начального состояния $i \in E_0^N$ (10.3), (10.4), назовем *коэффициентом потенциальной живучести ВС* и *коэффициентом занятости восстанавливающей системы соответственно.*

Воспользовавшись опытом обобщения (10.5), (10.7) показателей (9.6), (9.7) на случай живучих ВС и формулами (9.13), (9.14), можно без особого труда определить *вектор-функции оперативной надежности и восстановимости*:

$$\mathbf{R}^*(t) = \{R_k^*(t)\},\ \ \mathbf{U}^*(t) = \{U_k^*(t)\},\ \ k \in E_n^N, \tag{10.15}$$

компоненты которых соответственно равны:

$$R_K^*(t) = P\{\forall\tau \in [0,\ t) \to \xi(\tau) \geqslant k \mid P_i,\ k \leqslant i \leqslant N\}; \tag{10.16}$$

$$U_k^*(t) = 1 - P\{\forall\tau \in [0,\ t) \to \xi(\tau) < k \mid P_i,\ 0 \leqslant i < k\}; \tag{10.17}$$

$$R_k^*(0) = U_k^*(0) = \sum_{i=k}^{N} P_i, \tag{10.18}$$

где величины P_i вычисляются по (9.50), (9.51).

Совокупность величин $S_k = \lim_{t\to\infty} S_k(t)$, не зависящих от начального состояния ВС и представленных в виде

$$\mathbf{S} = \{S_k\},\ \ k \in E_n^N, \tag{10.19}$$

называется *вектор-коэффициентом готовности.*

Введенные показатели достаточно полно характеризуют поведение живучих ВС в переходном (10.3)–(10.12) и стационарном (10.13)–(10.19) режимах функционирования. Уместно подчеркнуть значение для практики лишь показателей (10.3) и (10.4). По функции потенциальной живучести ВС судят о том:

1) как быстро система, начавшая функционировать в одном из возможных состояний, войдет в стационарный режим работы (10.13);

2) какую производительность в среднем может обеспечить система в любой момент времени (10.3) или при длительной эксплуатации (10.13), говоря иначе, на сколько машин в среднем можно рассчитывать в момент поступления задачи;

3) сколько машин в среднем может быть использовано при решении задачи (сколько в среднем ветвей будет иметь место при реализации адаптирующейся параллельной программы).

Функция занятости восстанавливающей системы дает следующую информацию:

1) за какое время после начала работы ВС наступит установившийся режим восстановления отказавших машин (10.14);

2) как загружены в среднем восстанавливающие устройства на начальном участке работы ВС (10.4) и после длительной ее эксплуатации (10.14) или насколько эффективен выбранный состав восстанавливающих устройств.

При численном экспериментировании введенные функции потенциальной живучести ВС и занятости восстанавливающей системы позволяют подобрать оптимальные параметры единого комплекса «вычислительная система — восстанавливающая система».

Таким образом, в живучих ВС обеспечивается максимум эффективности использования вычислительных ресурсов при решении сложных задач. В любой момент времени для решения сложных задач привлекаются все работоспособные ЭМ (если их число не менее n). Это сокращает время решения сложной задачи, но требует составления специальных программ с информационной избыточностью, т. е. адаптирующихся параллельных программ. Процесс составления таких программ несущественно сложнее, чем написание программ с фиксированным числом параллельных ветвей.

10.3. О методике расчета показателей живучести вычислительных систем

Живучие ВС представляют собой обобщение систем со структурной избыточностью, следовательно, методика расчета координат вектор-функций (10.5), (10.7) и (10.11) остается неизменной по сравнению с методикой расчета функций надежности (9.6), восстановимости (9.7) и готовности (9.10). Более того, если в расчетных формулах для показателей надежности ВС заменить n на k, то получим формулы для координат соответствующих количественных характеристик живучести. Так, например, заменив n на k, можно использовать формулы (9.20) и (9.21) для расчета компонентов векторов соответственно среднего времени безотказной работы (10.9) и среднего времени восстановления ВС (10.10). Аналогичная процедура превращает (9.36) и (9.43) в расчетные формулы для координат вектор-функций оперативной надежности и восстановимости ВС.

Функции потенциальной живучести ВС (10.3) и занятости восстанавливающей системы (10.4), безусловно, могут быть рассчитаны по известной схеме (см. § 9.4), теории массового обслуживания. Пусть известно распределение вероятностей состояний системы $\{P_j(i, t)\}$, $j, i \in E_0^N$, i — начальное состояние ВС, тогда математическое ожидание числа работоспособных ЭМ

$$\mathcal{N}(i, t) = \sum_{j=0}^{N} jP_j(i, t), \tag{10.20}$$

а среднее число занятых восстанавливающих устройств

$$\mathcal{M}(i, t) = m\sum_{j=0}^{N-m} P_j(i, t) + \sum_{j=N-m+1}^{N} (N - j)P_j(i, t). \tag{10.21}$$

Из формул (10.20) и (10.21) следует, что сложность вычисления функций потенциальной живучести ВС и занятости восстанавливающей системы не менее сложности расчета функции готовности ВС (9.10). Расчет существенно упрощается для стационарного режима работы ВС, однако и он не может быть выполнен без ЭВМ для практически необходимых значений N (от 10 до 10^6). Последнее следует из (9.50), (9.51), (10.20) и (10.21).

Как же преодолеть указанную трудность? Ясно, что кардинальный путь может быть основан лишь на новой стохастической модели функционирования ВС, которая:

1) была бы воплощением принципа квазианалогии (см. § 9.4);

2) приводила бы к простым формулам, допускающим счет без применения ЭВМ и для больших значений N.

При исследовании потенциальной живучести ВС вместо рассмотрения всего ее пространства состояний $E_0^N = \{0, 1, ..., N\}$, т. е. учета функционирования каждой ЭМ, можно изучать поведение системы в целом как ансамбля большого числа идентичных ЭМ. Такой подход основывается на допущении, что в любой момент времени производительности вычислительной и восстанавливающей систем пропорциональны соответственно не случайному числу исправных ЭМ и не случайному числу занятых ВУ, а математическим ожиданиям соответствующих чисел. Допущение естественно для большемасштабных ВС (для систем с большим числом ЭМ и ВУ), в которых случайности, связанные с выходом ЭМ из строя или их восстановлением, либо с включением ВУ в ремонт ЭМ или освобождением ВУ, мало сказываются на значениях суммарной производительности систем. Эти значения для каждого момента времени оказываются близкими к средним значениям производительности систем. Случайности, связанные с выходом машин из строя и освобождением ВУ, сказываются, если количество работоспособных ЭМ в ВС становится небольшим или если число занятых устройств в восстанавливающей системе становится близким к m. Однако вероятности таких событий при существующем уровне надежности микропроцессоров ($\lambda^{-1} \leqslant 10^8$ ч) чрезвычайно малы.

В работах по исследованию операций (см., например [25]) установлено, что при описании динамики боя вполне допустима замена случайного

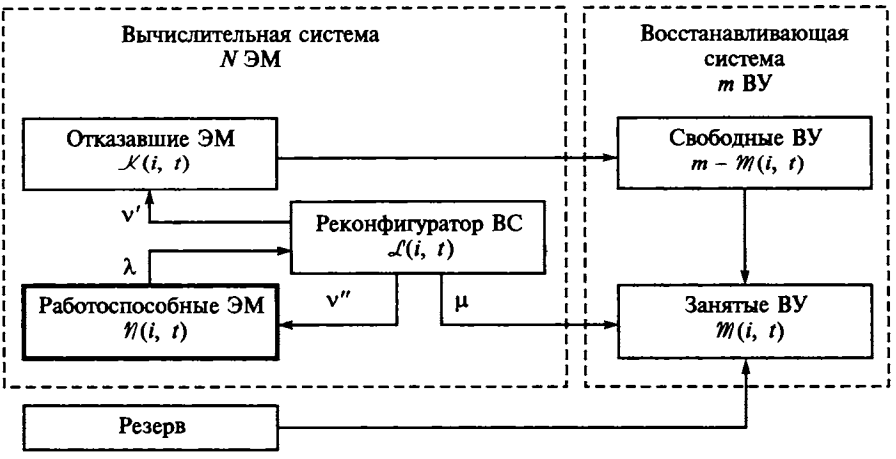


Рис. 10.2. Модель функционирования живучей ВС

числа его средним, если число элементов в каждой из систем противников не менее 50. В [5] показано, что такая замена применительно к ВС допустима, даже если число элементов (таких, как ЭМ, вычислительные модули, процессоры) на порядок меньше. Это объясняется тем, что в бою имеются два процесса взаимного уничтожения сторон, в то время как в ВС есть только один процесс уничтожения (поток отказов) и, кроме того, второй процесс приводит к регенерации ресурсов (процесс восстановления отказавших ЭМ).

На основании вышесказанного при изучении потенциальной живучести ВС за основу берется стохастическая модель функционирования, представленная на рис. 10.2. Производительность ВС в любой момент времени $t \geqslant 0$ определяют машины вычислительного ядра, т. е. те работоспособные ЭМ, которые непосредственно используются для реализации адаптирующейся параллельной программы. Пусть $\mathcal{N}(i,\ t)$ — математическое ожидание числа работоспособных ЭМ, на которых выполняется в момент $t \geqslant 0$ адаптирующаяся программа; i — число работоспособных машин при $t = 0$, т. е. $\mathcal{N}(i,\ 0) = i,\ \ i \in E_0^N$. Итак, считаем, что $\mathcal{N}(i,\ t)$ машин в момент $t \geqslant 0$ составляют вычислительное ядро ВС.

В случае отказа ЭМ «покидает» вычислительное ядро и берется на учет реконфигуратором ВС. Пусть $\mathcal{L}'(i,\ t)$ — среднее число отказавших ЭМ, учитываемых реконфигуратором ВС в момент $t \geqslant 0,\ \ i \in E_0^N$. Реконфигуратор исключает из вычислительного ядра отказавшие ЭМ, образует из оставшихся работоспособных ЭМ связную подсистему, сокращает число параллельных ветвей в адаптирующейся программе и организует ее прохождение на вычислительном ядре с новой структурой. В результате выполне-

ния таких функций реконфигуратор с интенсивностью ν' «переключает» отказавшие ЭМ из ядра в число машин, подлежащих восстановлению. Пусть $\mathcal{K}(i, t)$ — математическое ожидание числа отказавших машин, учитываемых восстанавливающей системой.

Как и ранее, $\mathcal{M}(i, t)$ — среднее число устройств, занятых восстановлением отказавших ЭМ; μ — интенсивность восстановления отказавших ЭМ одним ВУ. После восстановления ЭМ берутся на учет реконфигуратором ВС. Пусть $\mathcal{L}''(i, t)$ — среднее число восстановленных ЭМ, взятых на учет реконфигуратором ВС. Включение восстановленных ЭМ в состав ядра осуществляется с интенсивностью ν''. Среднее время $1/\nu''$ такого включения зависит от времени образования связного подмножества машин из существовавшего ядра и восстановленных ЭМ, времени перенастройки параллельной программы на большее число ветвей и времени запуска этой программы на вновь созданном ядре.

При изучении большемасштабных реконфигурируемых ВС, как уже отмечалось в § 9.4, используются виртуальные восстанавливающие устройства, и ремонт обнаруженных отказавших ЭМ сводится к их замене на машины из резерва. В этих случаях интенсивность μ воспринимается как среднее число машин резерва, включаемых в единицу времени одним виртуальным ВУ в состав вычислительной системы.

Очевидно следующее равенство:

$$\mathcal{L}(i, t) + \mathcal{K}(i, t) + \mathcal{N}(i, t) = N,$$

где $\mathcal{L}(i, t) = \mathcal{L}'(i, t) + \mathcal{L}''(i, t)$ — среднее число ЭМ, с которыми работает реконфигуратор ВС. Численные исследования показали, что учет интенсивностей переключения ν' и ν'' ЭМ в состав машин, подлежащих восстановлению, и в состав вычислительного ядра мало изменяет значения функций потенциальной живучести ВС и занятости восстанавливающей системы. Предположение о том, что такие переключения ЭМ совершаются мгновенно, при существующих параметрах λ и μ и аппаратурно-программных средствах реконфигурации ВС практически оправданно [5]. Поэтому далее будем считать, что $1/\nu' = 1/\nu'' = 0$; $\mathcal{L}(i, t) = 0$; $\mathcal{K}(i, t) = N - \mathcal{N}(i, t)$, $i \in E_o^N$

Таким образом, при анализе потенциальной живучести ВС достаточно знать математическое ожидание числа работоспособных ЭМ и числа занятых ВУ и не описывать дискретно каждое состояние системы (трудоемкими методами теории массового обслуживания).

Описанный подход к анализу потенциальной живучести большемасштабных ВС был предложен автором настоящей книги и назван *континуальным.*

10.4. Расчет функции потенциальной живучести вычислительных систем

Из формул (10.3) и (10.4) следует, что расчет функций потенциальной живучести ВС $\mathcal{N}(i, t)$ и занятости восстанавливающей системы $\mathcal{M}(i, t)$ сводится к вычислению математических ожиданий соответственно числа $\mathcal{N}(i, t)$ исправных ЭМ и числа $\mathcal{M}(i, t)$ занятых ВУ в момент времени $t \geqslant 0$ при условии, что в начальный момент $t = 0$ было $i \in E_0^N$ работоспособных машин.

Прежде чем вывести дифференциальные уравнения для $\mathcal{N}(i, t)$ и $\mathcal{M}(i, t)$, получим вспомогательные оценки. Учитывая (2.14), (2.15) и (2.21), найдем вероятность того, что в ЭМ произойдет не менее одного отказа за время Δt:

$$1 - r(\Delta t) = r_1(\Delta t) + \sum_{k=2}^{\infty} r_k(\Delta t) = \lambda \Delta t e^{-\lambda \cdot \Delta t} + o(\Delta t) = \lambda \Delta t[1 - \lambda \Delta t + \\ + o(\Delta t)] + o(\Delta t) = \lambda(\Delta t) + o(\Delta t) = r_1(\Delta t) + o(\Delta t).$$

Следовательно, в машине за время Δt может произойти только один отказ с вероятностью $\lambda \Delta t$; вероятность появления за Δt более одного отказа есть величина порядка $o(\Delta t)$.

Вспомним (2.14), что λ — среднее число отказов, появляющихся в машине за единицу времени, следовательно, $\mathcal{N}(i, t)\lambda \Delta t$ есть среднее число отказов, возникающих в системе на промежутке времени $[t, t + \Delta t)$. Поскольку за время Δt в машине может произойти не более одного отказа, то $\mathcal{N}(i, t)\lambda \Delta t$ и будет средним числом ЭМ, вышедших из строя на промежутке времени $[t, t + \Delta t)$. Аналогично рассуждая, убеждаемся, что $\mathcal{M}(i, t)\mu \Delta t$ есть среднее число восстановленных ЭМ на промежутке времени $[t, t + \Delta t)$, $i \in E_0^N$.

Легко заметить, что математическое ожидание числа исправных ЭМ в ВС в момент времени $(t + \Delta t)$ равно этому числу в момент t, уменьшенному на среднее число отказавших машин и увеличенному на среднее число восстановленных ЭМ в последующие Δt единиц времени:

$$\mathcal{N}(i, t + \Delta t) = \mathcal{N}(i, t) - \mathcal{N}(i, t)\lambda \Delta t + \mathcal{M}(i, t)\mu \Delta t + o(\Delta t). \qquad (10.22)$$

Перенеся $\mathcal{N}(i, t)$ в левую часть (10.22), разделив на Δt и перейдя к пределу при $\Delta t \to 0$, получим уравнение

$$\frac{d}{dt}\mathcal{N}(i,\ t) = -\lambda\,\mathcal{N}(i,\ t) + \mu\,\mathcal{M}(i,\ t), \tag{10.23}$$

причем

$$\mathcal{M}(i,\ t) = \begin{cases} m & \text{при } N - \mathcal{N}(i,\ t) > m; \\ N - \mathcal{N}(i,\ t) & \text{в противном случае.} \end{cases} \tag{10.24}$$

Найдем решение уравнения (10.23) при начальном условии $\mathcal{N}(i,\ 0) = i,\ i \in E_0^N$, для всех практически важных случаев.

Случай 1. Восстанавливающая система имеет высокую производительность, т. е. для любого $t \geqslant 0$ выполняется условие

$$N - \mathcal{N}(i,\ t) \leqslant m, \tag{10.25}$$

где $i \in E_{N-m}^N = \{N - m,\ N - m + 1,\ \ldots,\ N\}$.

Область определения устанавливается из следующего: неравенство (10.25) должно выполняться и при $t = 0$, а в начальный момент $\mathcal{N}(i,\ 0) = i$, значит, (10.25) превращается в неравенство $N - i \leqslant m$.

В рассматриваемом случае уравнение (10.23), как легко установить из (10.24), (10.25), трансформируется к виду

$$\frac{d}{dt}\mathcal{N}(i,\ t) = N\mu - (\lambda + \mu)\mathcal{N}(i,\ t),\ \ \mathcal{N}(i,\ 0) = i,\ \ i \in E_{N-m}^N. \tag{10.26}$$

Применив преобразование Лапласа—Карсона [9], вместо (10.26) можно записать алгебраическое уравнение

$$p[\overline{\mathcal{N}}(i,\ p) - \mathcal{N}(i,\ 0)] = N\mu - (\lambda + \mu)\overline{\mathcal{N}}(i,\ p),$$

где p — комплексный параметр; $\overline{\mathcal{N}}(i,\ p)$ — изображение функции $\mathcal{N}(i,\ t),\ i \in E_{N-m}^N$. Из последнего уравнения следует, что

$$\overline{\mathcal{N}}(i,\ p) = (ip + N\mu)/[p + (\lambda + \mu)]. \tag{10.27}$$

Формула обращения преобразования Лапласа—Карсона

$$\frac{\alpha p + \beta}{p + a} \sim \frac{\beta}{a} + \frac{\alpha a - \beta}{a} e^{-at}$$

позволяет вместо (10.27) записать решение уравнения (10.26)

$$\mathcal{N}(i,\ t) = \frac{N\mu}{\lambda + \mu} + \frac{i\lambda - (N - i)\mu}{\lambda + \mu} e^{-(\lambda+\mu)t},\ i \in \{N - m,\ N - m + 1,\ \ldots,\ N\}. \tag{10.28}$$

В результате подстановок легко убедиться, что решение (10.28) удовлетворяет начальному условию и уравнению (10.26). Учитывая (10.24), получаем

$$\mathcal{M}(i,\ t)=\frac{N\lambda}{\lambda+\mu}-\frac{i\lambda-(N-i)\mu}{\lambda+\mu}e^{-(\lambda+\mu)t},\quad i\in\{N-m,\ N-m+1,...,\ N\}.\quad (10.29)$$

В стационарном режиме математические ожидания числа работоспособных ЭМ и числа занятых восстанавливающих устройств не зависят от начального состояния ВС $i\in E_{N-m}^{N}$ и соответственно равны:

$$\mathcal{N}=\lim_{t\to\infty}\mathcal{N}(i,\ t)=\frac{N\mu}{\lambda+\mu};\qquad (10.30)$$

$$\mathcal{M}=\lim_{t\to\infty}\mathcal{M}(i,\ t)=\frac{N\lambda}{\lambda+\mu}.\qquad (10.31)$$

Выведем условие, при котором (10.25) выполняется на всем промежутке времени [0, ∞). Очевидно, что функция (10.28) монотонная. Если функция $\mathcal{N}(i,\ t)$ убывающая, то выполнение (10.25) на всем промежутке времени обеспечивается заданием начального состояния $i\in E_{N-m}^{N}$. Если же функция $\mathcal{N}(i,\ t)$, $i\in E_{N-m}^{N}$, возрастающая, то (10.25) должно выполняться и при $t\to\infty$. Тогда из (10.25) и (10.30) следует, что

$$N\lambda\leqslant m(\lambda+\mu).\qquad (10.32)$$

Таким образом, при заданных параметрах ВС N и λ *неравенство (10.32) является условием высокой производительности восстанавливающей системы.* Для средств ВТ практически всегда выполняется неравенство $\lambda\ll\mu$, поэтому вместо (6.32) можно записать

$$N\lambda\leqslant m\mu.\qquad (10.33)$$

Из (10.33) видно, что восстанавливающая система может быть отнесена к высопроизводительным, если среднее число отказов, появляющихся в единицу времени в ВС, не превышает среднего числа восстановлений, которые могут произвести все ВУ за ту же единицу времени. Величина $m\mu$ является количественной характеристикой производительности восстанавливающей системы.

Условия (10.32) или (10.33), как правило, удовлетворяются, поэтому для расчета функций потенциальной живучести ВС (10.3) и занятости восстанавливающей системы (10.4) можно пользоваться формулами:

$$\mathcal{N}(i,\ t) = \frac{\mu}{\lambda+\mu} + \frac{i\lambda - (N-i)\mu}{N(\lambda+\mu)} e^{-(\lambda+\mu)t}; \tag{10.34}$$

$$\mathcal{M}(i,\ t) = \frac{N\lambda}{m(\lambda+\mu)} - \frac{i\lambda - (N-i)\mu}{m(\lambda+\mu)} e^{-(\lambda+\mu)t}, \tag{10.35}$$

а для вычисления коэффициентов потенциальной живучести ВС (10.13) и занятости восстанавливающей системы (10.14) — формулами:

$$\mathcal{N} = \mu/(\lambda+\mu);\ \mathcal{M} = N\lambda/[m(\lambda+\mu)]. \tag{10.36}$$

Случай 2. Восстанавливающая система имеет невысокую производительность, т. е. при любом $t \geqslant 0$

$$N - \mathcal{N}(i,\ t) > m,\ \ i \in E_0^{N-m-1} = \{0, 1, \ldots, N-m-1\}. \tag{10.37}$$

Очевидно, что в этом случае $\mathcal{M}(i,\ t) = m$, а уравнение (10.23) принимает вид

$$\frac{d}{dt}\mathcal{N}(i,\ t) = m\mu - \lambda\,\mathcal{N}(i,\ t),\ \ \mathcal{N}(i,\ 0) = i,\ \ i \in E_0^{N-m-1}. \tag{10.38}$$

Решением (10.38) является

$$\mathcal{N}(i,\ t) = \frac{m\mu}{\lambda} + \frac{i\lambda - m\mu}{\lambda} e^{-\lambda t},\quad i \in E_0^{N-m-1}, \tag{10.39}$$

а *условиями малой производительности восстанавливающей системы (10.37) будут неравенства, обратные (10.32), (10.33).*

Функция и коэффициент потенциальной живучести ВС соответственно равны:

$$\mathcal{N}(i,\ t) = \frac{m\mu}{N\lambda} + \frac{i\lambda - m\mu}{N\lambda} e^{-\lambda t},\ \ i \in E_0^{N-m-1};\ \ \mathcal{N} = m\mu/(N\lambda). \tag{10.40}$$

Функция занятости восстанавливающей системы тождественно равна константе: $\mathcal{M}(t) = \mathcal{M} = 1$.

Случай 3. Восстанавливающая система имеет невысокую производительность, но $\mathcal{N}(i,\ 0) = i,\ \ i \in E_{N-m}^{N}$. В этом случае до момента времени t^*, когда впервые нарушится условие (10.25), будут справедливы формулы (10.34), (10.35). С момента t^* будет справедлива формула (10.40), в которой следует положить $i = N - m - 1$.

Случай 4. Восстанавливающая система имеет высокую производительность, однако $\mathcal{N}(i,\ 0) = i,\ \ i \in E_0^{N-m-1}$. В этом случае вначале будет спра-

ведлива формула (10.40); с момента t^*, когда впервые нарушится условие (10.37), справедливыми станут уже формулы (10.34), (10.35), в которых $i = N - m$.

Вероятность ситуации, соответствующей случаю 1, существенно выше вероятности случая 2. Случаи 3 и 4 практически маловероятны.

Результаты, полученные в данном параграфе, свидетельствуют о диалектическом единстве ЭВМ и ВС, позволяют глубже понять физический смысл функции $\mathcal{N}(i, t)$ потенциальной живучести ВС. В самом деле, несмотря на кажущееся на первый взгляд различие в приемах расчета функции (2.19) готовности ЭВМ и функции (10.2) $\mathcal{N}(i, t)$, при внимательном рассмотрении обнаруживается единообразие не только в схемах проведения расчетов, но и в окончательных результатах. Так, формулы (2.24) или (2.25) для функции готовности ЭВМ являются частными результатами по отношению к формуле (10.34) для функции потенциальной живучести ВС. В справедливости последнего легко убедиться при подстановке в (10.34) $i = N$ или $i = 0$. Следовательно, все семейство кривых $\mathcal{N}(i, t)$ (10.34) для $i = 0, 1, \ldots, N$ заключено между $s(0, t)$ (2.25) и $s(1, t)$ (2.24), т. е. имеет место неравенство $s(0, t) \leqslant \mathcal{N}(i, t) \leqslant s(1, t)$ для любого $i \in E_0^N$.

Далее, коэффициент (2.26) готовности ЭВМ является средней долей времени пребывания машины в работоспособном состоянии, т. е. в состоянии, когда она способна работать с потенциально возможной производительностью. Коэффициент готовности ЭВМ полностью совпадает с коэффициентом (10.13) потенциальной живучести ВС (см. формулы (2.26) и (10.36)). Значит, коэффициент потенциальной живучести ВС дает информацию о средней доле времени функционирования каждой ЭМ системы с потенциально возможной производительностью. Из вышесказанного и из формул (10.2), (10.28)–(10.31), (10.39) следует, что математические ожидания производительности (емкости памяти и т. п.) ВС и производительности восстанавливающей системы к моменту времени $t \geqslant 0$ равны

$$\overline{\Omega}(i, t) = \mathcal{N}(i, t)\omega, \quad \mathcal{M}(i, t)\mu, \quad i \in E_0^N,$$

а в стационарном режиме при выполнении (10.33) —

$$\overline{\Omega} = \mathcal{N}\omega = \frac{N\mu}{\lambda + \mu}\omega, \quad \frac{N\lambda\mu}{\lambda + \mu},$$

где ω — производительность (емкость памяти) одной ЭМ.

10.5. Анализ живучих вычислительных систем

1. Рассмотренные живучие вычислительные системы являются обобщением ВС со структурной избыточностью. Живучие ВС с позиций пользователя выглядят как виртуальные системы, способные реализовать адаптирующиеся параллельные программы с числом ветвей, заданном в некотором диапазоне. Для проектировщиков и эксплуатационников такие ВС являются системами, обладающими аппаратурно-программными средствами реконфигурации, которые позволяют использовать все работоспособные ресурсы для реализации адаптирующихся параллельных программ.

2. Предложенные показатели качества функционирования живучих ВС в переходном и стационарном режимах и методы их расчета, выведенные формулы для функции потенциальной живучести вполне приемлемы в инженерной практике.

3. Численный анализ живучести большемасштабных распределенных ВС показывает, что они входят в стационарный режим работы за время, не превышающее 10 ч.

4. Установлено, что организация работы вычислительных систем как живучих ВС позволяет достичь потенциальной живучести, близкой к готовности одной ЭМ. При экспресс-анализе потенциальной живучести ВС достаточно использовать предельно простую формулу для коэффициента потенциальной живучести (10.36) или, что то же самое, формулу для коэффициента готовности одной элементарной машины (2.26).

5. В условиях современной элементной базы, т. е. микропроцессорных БИС ($\lambda \leqslant 10^{-8}$ 1/ч), большемасштабные распределенные ВС с программируемой структурой являются высокопроизводительными, высоконадежными и живучими средствами обработки информации.

11. ОСУЩЕСТВИМОСТЬ РЕШЕНИЯ ЗАДАЧ НА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

В данной главе изучаются вероятностные закономерности, связанные с решением задач на распределенных ВС. Любая ВС является стохастическим объектом, никакая технология БИС, никакие архитектурные решения не гарантируют абсолютную надежность ее ресурсов. Процесс поступления задач на ВС является также случайным, и каждая задача характеризуется вектором вероятностных параметров в общем случае.

В главах 9 и 10 рассмотрены потенциальные возможности ВС по обеспечению необходимых уровней производительности при решении сложных задач (см. разд. 3.3.4), представленных параллельными программами соответственно с фиксированными и переменными количествами ветвей.

Показатели надежности и живучести оценивают качество работы ВС вне связи с процессами поступления и решения задач. Поэтому для оценки потенциальных возможностей ВС по достижению цели их функционирования (решения поступивших задач) используют показатели осуществимости решения задач. Эти показатели достаточно полно характеризуют качество работы систем с учетом их надежности и параметров поступающих задач. Говоря иначе, они характеризуют процесс решения задач на неабсолютно надежных ВС.

11.1. Режимы функционирования вычислительных систем

Вычислительные системы должны обеспечивать эффективное решение задач в моно- и мультипрограммных режимах. Монопрограммный режим функционирования ВС предопределяет использование всех исправных ресурсов —ЭМ для решения одной сложной задачи (см. разд. 3.3.4 и 7.2.2), представленной параллельной программой. Мультипрограммные режимы работы ВС предусматривают распределение всего множества (исправных) ЭМ между задачами (различной сложности, с различным числом параллельных ветвей в их программах). К последним режимам относятся обработка набора задач и обслуживание потока задач (см. разд. 7.2.2).

При организации функционирования ВС в случае набора задач учитывается не только количество задач, но их параметры: число ветвей в программе (точнее, число машин, на которых она будет выполняться), время решения или вероятностный закон распределения времени решения и др. Алгоритмы организации функционирования ВС [5] задают распределение задач по машинам и последовательность выполнения задач на каждой машине. В результате становится известным, в каком промежутке времени и на каких машинах (или на какой подсистеме) будет решаться любая задача набора. Этот режим, безусловно, является обобщением режима решения сложной задачи на ВС.

Вместе с тем с позиций теории осуществимости решения задач этот режим без нарушения общности сводится к первому. В самом деле, осуществимость решения любой задачи набора зависит от параметров заданной для нее подсистемы, а анализ эффективности работы последней ничем не отличается от анализа всей системы при решении одной сложной задачи. Поэтому в дальнейшем ограничимся рассмотрением только режима решения одной сложной задачи на ВС.

Режим обслуживания потока задач на ВС принципиально отличается от обработки задач набора: задачи поступают в случайные моменты времени, их параметры случайны, следовательно, детерминированный выбор подсистем для решения тех или иных задач исключен.

Для режимов решения сложной задачи и обслуживания потока задач на ВС ниже будут приведены показатели осуществимости и рассмотрены методы их расчета. При анализе осуществимости решения задач будем учитывать большемасштабность и масштабируемость ВС.

11.2. Анализ решения сложных задач на вычислительных системах

Возможность программирования структуры ВС позволяет организовать решение одной сложной задачи на виртуальных конфигурациях, названных системами со структурной избыточностью и живучими ВС (см. § 9.2 и 10.1). В первом случае число параллельных ветвей в программе фиксировано и равно n, а во втором — это число варьируемо от n до N, где N — общее число элементарных машин в ВС, $n \leqslant N$.

11.2.1. Функция осуществимости решения задач на вычислительных системах со структурной избыточностью

Осуществимость решения задачи на ВС со структурной избыточностью оценивается функцией

$$F(t) = R(t)\Phi(t), \qquad (11.1)$$

где $R(t)$ — функция надежности системы или вероятность безотказной работы ВС в течение времени t (9.6); $\Phi(t)$ — вероятность решения задачи на n работоспособных ЭМ за время t, т. е. $\Phi(t) = P\{0 \leqslant \zeta < t\}$, ζ — случайная величина, являющаяся моментом решения задачи на множестве из n исправных машин.

В момент начала решения задачи ВС может находиться в состоянии $i \in E_n^N = \{n, n+1, \ldots, N\}$, т. е. в ней может быть исправно i ЭМ. Если во множестве из i работоспособных ЭМ можно выделить подмножество из $n < i$, $i \in E_n^N$, связных машин, тогда это подмножество будет подсистемой, способной выполнять программу из n ветвей. В случае, когда совокупность $i \in E_n^N$ работоспособных ЭМ распадается на несвязанные между собой подсистемы (с числом ЭМ в каждой из них меньшим n), в системе исчезает возможность реализации программы из n ветвей. *Функция $\Phi(t)$ — это вероятностный закон решения сложной задачи на любой совокупности из n работоспособных машин при произвольном их распределении в пределах всей ВС.* Вид этого закона устанавливается на основе статистической обработки результатов решения задач на ВС.

При эксплуатации статистически установлено [22], что закон распределения времени решения простых задач на одной машине является экспоненциальным. Данный факт и опыт решения сложных задач на ВС позволяют считать, что

$$\Phi(t) = 1 - \exp(-\beta_n t), \qquad (11.2)$$

где β_n — интенсивность ($1/\beta_n$ — среднее время) решения задач на n машинах. Практически величина β_n близка к $n\beta_1$ (при любом из способов обработки информации: распределенном, матричном, конвейерном), что является следствием методики крупноблочного распараллеливания сложных задач (см. разд. 7.2.2).

Следовательно, функция (11.1) позволяет судить о том, с какой вероятностью за время $t \geqslant 0$ сложная задача, представленная параллельной программой с n ветвями, будет решена на неабсолютно надежной ВС, в которой из N машин $(N - n)$ ЭМ составляют структурную избыточность (в частности, резерв).

Поскольку $R(t)$ и $\Phi(t)$ являются соответственно невозрастающей и неубывающей функциями, то существует такое значение t_m времени, при котором $F(t)$ достигает максимума: $F(t_m) = \max_t F(t)$. Из последнего видно, что наиболее вероятно ожидать решения задачи в момент t_m, после прохождения этого времени вероятность решения задачи уменьшается и асимптотически стремится к нулю.

Функцию $F(t)$ (11.1) называют *функцией осуществимости решения задачи на ВС со структурной избыточностью* (или функцией осуществимости решения задачи на n машинах ВС). При этом говорят, что решение сложной задачи осуществимо на системе со структурной избыточностью (на n машинах ВС), если для некоторого t одновременно имеют место неравенства $F(t) \geqslant F^{\circ}$, $t \leqslant t^{\circ}$; F° и t° называют порогами осуществимости параллельного решения задачи и их значения выбирают из практических соображений.

Методика расчета функции осуществимости $F(t)$, как следует из (11.1), ничем не отличается от методики расчета функции надежности ВС, т. е. $R(t)$. В § 9.4 отмечены вычислительные трудности при выводе формул для $R(t)$. Численный расчет $F(t)$ не обходится без привлечения методов вычислительной математики, следовательно, он не может быть выполнен без применения средств ВТ. Учитывая ориентацию данной книги, мы вынуждены отказаться здесь от выкладок, приводящих к формулам для программирования, и сослаться на Приложение 2 или [5].

Если анализировать осуществимость решения задач на ВС, режим работы которой стационарен, то вместо (11.1) достаточно использовать

$$F^{*}(t) = R^{*}(t)\Phi(t),$$

где $R^{*}(t)$ рассчитывается по формулам (9.13) и (9.36). Функцию $F^{*}(t)$ назовем *функцией оперативной осуществимости решения задачи на ВС со структурной избыточностью.*

На практике при расчете $F^{*}(t)$ достаточно учесть лишь оценку $\tilde{R}^{*}(t)$ снизу для $R^{*}(t)$, т. е. (9.40). Однако даже и при такой точности расчета анализ осуществимости параллельного решения задач не может быть выполнен без применения ЭВМ или ВС, т. е. анализ остается трудоемким. Чтобы все-таки удовлетворить потребности инженеров (а не исследователей в области анализа функционирования ВС), ниже рассчитаем показатели, позволяющие легко оценить потенциальную осуществимость решения задачи на ВС. Последнего будет вполне достаточно для инженеров-проектировщиков, эксплуатационников и пользователей ВС.

11.2.2. Функция осуществимости решения задач на живучих вычислительных системах

Математическое ожидание $\eta(i, t)$ числа работоспособных машин при условии, что в начальный момент времени было исправно $i \in E_0^N = \{0, 1, 2, \ldots, N\}$ ЭМ, достаточно точно говорит об уровне потенциальной производительности ВС в любой момент $t > 0$. Это тем точнее, чем

больше общее число N машин в системе; сделанное допущение оправданно на практике при $N \geqslant 10$. Далее, для сложных задач допустимо составление адаптирующихся параллельных программ, в которых нижняя граница n числа параллельных ветвей не превышает ожидаемого числа работоспособных машин в любой момент времени, т. е. в которых $n \leqslant \mathcal{N}(i,\ t)$. Тогда осуществимость параллельного решения сложной задачи на живучей ВС целесообразно оценивать функцией

$$\mathcal{J}(i,\ t) = 1 - \exp\left[-\beta \int_0^t \mathcal{N}(i,\ \tau) d\tau\right], \tag{11.3}$$

где $\beta = \beta_1$ — интенсивность решения задач на одной ЭМ.

Из формулы (11.3) видно, что функция $\mathcal{J}(i,\ t)$ является вероятностью того, что сложная задача, представленная адаптирующейся параллельной программой, будет решена за время t на ВС, начавшей функционировать в состоянии $i \in E_0^N$ и использующей для решения задачи все работоспособные ЭМ. Если же ВС функционирует достаточно долго (находится в стационарном режиме), то вероятность решения задачи может быть выражена предельно просто:

$$\mathcal{J}(t) = 1 - \exp(-\beta\, \mathcal{N} t). \tag{11.4}$$

Здесь $\mathcal{N} = \lim\limits_{t \to \infty} \mathcal{N}(i,\ t)$ (см. формулу (10.30)).

Функции $\mathcal{J}(i,\ t)$ и $\mathcal{J}(t)$ позволяют проанализировать процесс параллельного решения задачи соответственно в переходном и стационарном режимах работы живучей ВС. Считают, что решение сложной задачи осуществимо на живучей ВС в промежутке времени $[0,\ t)$, если выполняются неравенства $\mathcal{J}(i,\ t) \geqslant \mathcal{J}^\circ,\ t \leqslant t^\circ$ для переходного режима и неравенства $\mathcal{J}(t) \geqslant \mathcal{J}^\circ,\ t \leq t^\circ$ для стационарного режима функционирования системы. Величины $\mathcal{J}^\circ$ и t° называются порогами осуществимости решения сложной задачи на живучей ВС.

Использовав результаты (10.4), в частности формулы (10.28) и (10.39), можно найти, что

$$\mathcal{J}(i,\ t) = 1 - \exp\begin{cases} -\beta\left\{\dfrac{N\mu}{\lambda+\mu}t + \dfrac{i\lambda-(N-i)\mu}{(\lambda+\mu)^2}\left[1-e^{-(\lambda+\mu)t}\right]\right\}, \\ \text{если} \quad i \in \{N-m,\ N-m+1, \quad N\},\ N\lambda \leqslant m\mu; \\ -\beta\left\{\dfrac{m\mu}{\lambda}t + \dfrac{i\lambda-m\mu}{\lambda^2}\left[1-e^{-\lambda t}\right]\right\}, \\ \text{если} \quad i \in \{0,\ 1,\ \ldots,\ N-m-1\},\ N\lambda > m\mu. \end{cases} \tag{11.5}$$

Расчет числовых значений функции $\mathcal{J}(i,\ t)$, безусловно, проще, чем $F(t)$ (11.1). Однако численный расчет осуществимости параллельного решения сложных задач допускает еще одно упрощение. В самом деле, для ВС общего назначения типичен режим длительной эксплуатации. Кроме того, ранее было показано (см. § 9.8), что ВС достаточно «быстро» входят в стационарный режим работы. Поэтому даже и для ВС специального назначения в ряде случаев можно ограничиться анализом их стационарного режима работы. Если это так, то очевидная последовательность выкладок с использованием формул (11.4), (10.30) и (10.39) приводит к простейшему результату:

$$\mathcal{J}(t) = 1 - \exp\begin{cases} -\beta N\mu(\lambda+\mu)^{-1}t, & \text{если } N\lambda \leqslant m\mu; \\ -\beta m\mu\lambda^{-1}t & \text{в противном случае.} \end{cases} \qquad (11.6)$$

11.3. Анализ обслуживания потока задач на вычислительных системах

Проанализируем работу ВС в режиме обслуживания потоков задач. В общем случае в потоке присутствуют задачи различных рангов r (т. е. с различным числом параллельных ветвей в их программах), где $1 \leqslant r \leqslant N$, N — количество ЭМ некоторой ВС, используемых для обслуживания потока задач (в частности, это может быть общее количество ЭМ в системе). Для решения задач каждого ранга в пределах ВС выделяются одна или несколько подсистем, количество связанных машин в каждой из которых равно соответствующему рангу. Если из-за физических ограничений это осуществить невозможно, то с помощью механизма мультипрограммирования операционной системы осуществляется выделение таких подсистем в виртуальном смысле.

При наличии потока задач различных рангов функционирование ВС организуется таким образом, что для задач любого ранга имеются свои подсистемы машин. При этом не представляет особого труда установить интенсивность потока задач любого ранга на соответствующие подсистемы и интенсивность обслуживания задач каждой из подсистем.

Из сказанного видно, что анализ функционирования ВС при наличии потока задач сводится к решению проблемы в упрощенной постановке. Пусть на ВС (состоящую из N неабсолютно надежных ЭМ) поступает пуассоновский поток простых задач с интенсивностью α. Каждая задача представлена последовательной программой и решается на исправной ЭМ в среднем за время $1/\beta$. Требуется рассчитать следующие показатели осуществимости решения задач: *математические ожидания* $\mathcal{A}(t)$ *и* $\mathcal{B}(t)$ *соот-*

ветственно количества задач, находящихся в системе, и количество ЭМ, занятых решением, в момент времени $t \geqslant 0$. Вместе с тем следует заметить, что такое трансформирование сложности исходной постановки задачи не всегда удовлетворяет требованиям практики.

Заметим, что формулы для $\mathcal{A}(t)$ и $\mathcal{B}(t)$ можно получить классическими методами теории массового обслуживания [21, 26] лишь для абсолютно надежных ВС. Поэтому здесь применим опробованный в § 10.4 прием, приводящий к простым расчетным формулам, для двух случаев.

Случай 1. Поток задач имеет слабую интенсивность и такую, что для любого $t \geq 0$ выполняется условие

$$\mathcal{A}(t) \leqslant \mathcal{N}(i,\ t), \tag{11.7}$$

т. е. всегда в системе есть работоспособные и свободные машины для решения поступающих задач. Из (11.7) видно, что $\mathcal{B}(t) = \mathcal{A}(t)$.

Математическое ожидание количества задач, находящихся в системе в момент времени $t + \Delta t$, равно этому количеству в момент t плюс среднее число задач, поступивших в ВС за время Δt, минус среднее число задач, покинувших ВС за время Δt вследствие их решения, т. е.

$$\mathcal{A}(t + \Delta t) = \mathcal{A}(t) + \alpha\Delta t - \mathcal{A}(t)\beta\Delta t. \tag{11.8}$$

Очевидные преобразования (11.8) приводят к следующему дифференциальному уравнению:

$$\frac{d}{dt}\mathcal{A}(t) = \alpha - \beta\mathcal{A}(t). \tag{11.9}$$

Неравенство (11.7) устанавливает область допустимых значений для $\mathcal{A}(t)$ при $t = 0$:

$$\mathcal{A}(0) = j;\ \ j \in \{0, 1, \ldots, i\} = E_0^i,\ \ i \in E_0^N \tag{11.10}$$

Применяя преобразование Лапласа—Карсона [9], вместо (11.9) получаем алгебраическое уравнение вида

$$p\left[\bar{\mathcal{A}}(p) - \mathcal{A}(0)\right] = \alpha - \beta\bar{\mathcal{A}}(p),$$

где p — комплексный параметр; $\bar{\mathcal{A}}(p)$ – изображение функции $\mathcal{A}(t)$. Из последнего с учетом (11.10) следует

$$\bar{\mathcal{A}}(p) = (jp + \alpha)/(p + \beta).$$

Используя известную формулу обращения преобразования Лапласа—Карсона

$$\frac{jp+\alpha}{p+\beta} \sim \frac{\alpha}{\beta} + \frac{j\beta-\alpha}{\beta} e^{-\beta t},$$

находим решение уравнения (11.9) при начальных условиях (11.10):

$$\mathcal{A}(t) = \frac{\alpha}{\beta} + \left(j - \frac{\alpha}{\beta} \right) e^{-\beta t} \tag{11.11}$$

Подстановка $t = 0$ в (11.11) и самой функции $\mathcal{A}(t)$ в (11.9) убеждает в том, что (11.11) удовлетворяет начальному условию (11.10) и уравнению (11.9).

В стационарном режиме среднее количество задач, находящихся в ВС, не зависит от начального условия:

$$\mathcal{A} = \lim_{t \to \infty} \mathcal{A}(t) = \alpha / \beta. \tag{11.12}$$

Вместо (11.7) выведем простое условие. Для этого учтем, что (11.7) должно выполняться на всем промежутке времени $[0, \infty)$, и используем (11.12), (10.30), (10.39):

$$\lim_{t \to \infty} \mathcal{A}(t) \leqslant \lim_{t \to \infty} \mathcal{N}(i,\ t),\ \ \mathcal{A} \leqslant \mathcal{N}.$$

Следовательно, поток поступающих на ВС задач считается слабоинтенсивным, если выполняются неравенства:

$$\frac{\alpha}{\beta} \leqslant \begin{cases} N\mu(\lambda+\mu)^{-1} & \text{при } N\lambda \leqslant m(\lambda+\mu); \\ m\mu\lambda^{-1} & \text{в противном случае.} \end{cases} \tag{11.13}$$

Неравенства (11.13) допускают дальнейшие упрощения. В самом деле, для современных ЭВМ имеет место $\lambda \ll \mu$. Тогда (11.13) принимает следующий вид:

$$\alpha \leqslant \begin{cases} N\beta & \text{при } N\lambda \leqslant m\mu; \\ m\mu\beta\lambda^{-1} & \text{при } N\lambda > m\mu. \end{cases} \tag{11.14}$$

Таким образом, (11.13), (11.14) указывают на условия, при которых справедлива формула (11.11) для расчета математического ожидания количества задач, находящихся в ВС в момент времени $t \geqslant 0$. В частности, если среднее количество отказов, появляющихся в единицу времени в ВС (т. е. $N\lambda$), не превышает производительности восстанавливающей системы (т. е. $m\mu$) и если среднее количество задач, поступающих в единицу времени (α), не более потенциальной производительности ВС ($N\beta$), то расчет среднего количества задач в системе можно производить по простым формулам (11.11) и (11.12).

Случай 2. Поток поступающих на ВС задач — сильноинтенсивный и такой, что имеет место неравенство

$$\mathcal{A}(t) > \mathcal{N}(i, t).$$

Следовательно, ВС эксплуатируется в условиях, когда имеется очередь задач, ожидающих обслуживания. Тогда $\mathcal{B}(t) = \mathcal{N}(i, t)$ и справедлива такая последовательность выкладок:

$$\left.\begin{aligned} &\mathcal{A}(t+\Delta t) = \mathcal{A}(t) + \alpha\Delta t + \mathcal{N}(i, t)\lambda\Delta t - \mathcal{N}(i, t)\beta\Delta t; \\ &\frac{d}{dt}\mathcal{A}(t) = \alpha + (\lambda - \beta)\mathcal{N}(i, t); \\ &\mathcal{A}(0) = j, \quad j \in \{i+1, i+2, \ldots\} = E_{i+1}^{\infty}. \end{aligned}\right\} \qquad (11.15)$$

После применения преобразования Лапласа—Карсона к (11.15) находим, что

$$p\left[\bar{\mathcal{A}}(p) - \mathcal{A}(0)\right] = \alpha + (\lambda - \beta)\bar{\mathcal{N}}(i, p),$$

следовательно,

$$\bar{\mathcal{A}}(p) = j + \frac{\alpha}{p} + \frac{\lambda - \beta}{p}\bar{\mathcal{N}}(i, p). \qquad (11.16)$$

Для изображения $\bar{\mathcal{N}}(i, p)$, как было установлено в § 10.4, справедлива формула

$$\bar{\mathcal{N}}(i, p) = \begin{cases} (ip + N\mu)/[p + (\lambda + \mu)] & \text{при } N\lambda \leqslant m(\lambda + \mu); \\ (ip + m\mu)/(p + \lambda) & \text{при } N\lambda > m(\lambda + \mu). \end{cases}$$

Подставив значение $\bar{\mathcal{N}}(i, p)$ для ситуации, когда имеет место неравенство $N\lambda \leqslant m(\lambda + \mu)$, в (11.16) получим

$$\bar{\mathcal{A}}(p) = j + \frac{\alpha}{p} + \frac{i(\lambda - \beta)}{p + (\lambda + \mu)} + \frac{N\mu(\lambda - \beta)}{p[p + (\lambda + \mu)]}.$$

Запишем известные формулы преобразования Лапласа—Карсона [9]:

$$\frac{\alpha}{p} \sim at; \quad \frac{i(\lambda - \beta)}{p + (\lambda + \mu)} \sim \frac{i(\lambda - \beta)}{\lambda + \mu}\left[1 - e^{-(\lambda+\mu)t}\right];$$

$$\frac{N\mu(\lambda - \beta)}{p[p + (\lambda + \mu)]} \sim \frac{N\mu(\lambda - \beta)}{\lambda + \mu}t - \frac{N\mu(\lambda - \beta)}{(\lambda + \mu)^2}\left[1 - e^{-(\lambda+\mu)t}\right].$$

Последние формулы переводят $\bar{\mathcal{A}}(p)$ в решение для уравнения (11.15) при заданных начальных условиях.

Далее, если подставить значение $\mathcal{N}(i, p)$ при $N\lambda > m(\lambda + \mu)$ в формулу (11.16), то можно найти, что

$$\bar{\mathcal{A}}(p) = j + \frac{\alpha}{p} + \frac{i(\lambda - \beta)}{p + \lambda} + \frac{m\mu(\lambda - \beta)}{p(p + \lambda)}.$$

Тривиален путь превращения этого изображения $\bar{\mathcal{A}}(p)$ в оригинал $\mathcal{A}(t)$.

Общая формула для математического ожидания количества задач, находящихся в ВС в момент времени $t \geqslant 0$ при невыполнении неравенства (11.7), имеет вид

$$\mathcal{A}(t) = \left[j + \frac{i(\lambda - \beta)}{x} - \frac{y\mu(\lambda - \beta)}{x^2} \right] + \left[\alpha + \frac{y\mu(\lambda - \beta)}{x} \right] t - \\ - \left[\frac{i(\lambda - \beta)}{x} - \frac{y\mu(\lambda - \beta)}{x^2} \right] e^{-xt}, \qquad (11.17)$$

где

$$x = \begin{cases} \lambda + \mu, & \text{если } N\lambda \leqslant m(\lambda + \mu), \\ \lambda, & \text{если } N\lambda > m(\lambda + \mu); \end{cases} \qquad (11.18)$$

$$y = \begin{cases} N, & \text{если } N\lambda \leqslant m(\lambda + \mu), \\ m, & \text{если } N\lambda > m(\lambda + \mu). \end{cases} \qquad (11.19)$$

Заметим, что выражения (11.17)—(11.19) характеризуют процесс обслуживания сильноинтенсивного потока задач ВС независимо от режима (переходного или стационарного) ее работы. Однако в стационарном режиме функционирования ВС третьим и первым слагаемыми в (11.17) можно пренебречь. В самом деле, при $t \to \infty$ функция $e^{-xt} \to 0$, а первое слагаемое становится пренебрежимо малым по сравнению со вторым. Таким образом, математическое ожидание количества задач, находящихся в системе в момент $t \geqslant 0$, для стационарного режима работы ВС не зависит от начального числа задач и начального состояния ВС и имеет вид

$$\mathcal{A}(t) = \left[\alpha + \frac{y\mu(\lambda - \beta)}{x} \right] t = \alpha^* t. \qquad (11.20)$$

Очевидно, что величина α^* является средним числом задач, на которое увеличивается в единицу времени их общее число в системе или очередь нерешенных задач. Условием, при котором очередь нерешенных задач будет неограниченно расти во времени, является неравенство $\alpha^* > 0$ или

$$\alpha > y\mu(\beta - \lambda)/x.$$

Учитывая последнее неравенство, формулы (11.18), (11.19), а также, что $\lambda \ll \mu$, получаем более простое условие роста очереди нерешенных задач:

$$\alpha > \begin{cases} N(\beta - \lambda), & \text{если } N\lambda \leqslant m\mu; \\ m\mu(\beta - \lambda)\lambda^{-1}, & \text{если } N\lambda > m\mu. \end{cases} \qquad (11.21)$$

Из (11.21) следует, что показатель $A(t)$ осуществимости решения задач потока следует рассчитывать по формулам (11.17)–(11.19), если интенсивность потока задач выше суммарной интенсивности их решения всеми машинами ВС при условиях неабсолютной надежности ЭМ и работы восстанавливающей системы.

Заметим, что во втором случае производительность и надежность ВС и производительность восстанавливающей системы недостаточны для того, чтобы все поступающие задачи решались без простаивания в очереди. Если анализ убеждает в справедливости (11.21), то, для того чтобы избежать образования очереди задач на обслуживание, необходимо ввести в эксплуатацию более мощную и надежную ВС (или, в частности, увеличить производительность восстанавливающей системы).

Таким образом, случай 1 практически наиболее важен.

11.4. Оценка потенциальных возможностей вычислительных систем по осуществимости решения задач

1. Для ВС, организованных как системы со структурной избыточностью или как живучие ВС, существуют, как показано выше, показатели осуществимости решения задач. Эти показатели устанавливают взаимосвязь между количественными характеристиками надежности или живучести ВС и вероятностными параметрами поступающих на решение задач. Они дают информацию о качестве работы ВС и в переходном, и в стационарном режимах. Полученные расчетные формулы практически приемлемы для целей экспресс-анализа осуществимости решения задач на ВС с произвольным количеством ЭМ.

2. Численное моделирование показало, что до 10 ч устанавливается стационарный режим решения задач на ВС. Данный факт приводит к предельной простоте в экспресс-анализе процесса решения задач на неабсолютно надежных ВС.

3. Континуальный подход адекватен и эффективен при анализе осуществимости параллельного решения задач на большемасштабных распределенных ВС.

12. ТЕХНИКО-ЭКОНОМИЧЕСКАЯ ЭФФЕКТИВНОСТЬ ФУНКЦИОНИРОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Экономические закономерности функционирования ЭВМ как автономных объектов изучены в § 2.9. В этой главе с технико-экономических позиций рассмотрено функционирование распределенных ВС, говоря иначе, коллективное поведение ЭМ как ансамблей; введены и рассчитаны показатели, устанавливающие взаимосвязь между производительностью, надежностью и стоимостью систем. При этом проанализирована работа ВС как в переходном, так и в стационарном режимах. Для такого анализа, естественно, необходим соответствующий математический инструментарий. Особое внимание уделено анализу большемасштабных ВС.

12.1. Цена быстродействия вычислительных систем

Эмпирический закон Гроша, приведенный в разд. 2.9.2 и устанавливающий взаимосвязь между производительностью и ценой ЭВМ, применим и к одной элементарной машине ВС. Действительно, каждая ЭМ есть композиция ЭВМ и системного устройства или локального коммутатора (см. § 7.3–7.6), следовательно, цена ЭМ

$$v = v_{\text{ЭВМ}} + v_{\text{СУ}},$$

где $v_{\text{ЭВМ}}$ и $v_{\text{СУ}}$ — цены ЭВМ и системного устройства соответственно. В условиях ЭВМ второго и третьего поколений, как показано в § 7.3, $v_{\text{ЭВМ}} \gg v_{\text{СУ}}$. Даже для мини- и микромашинной техники (см. § 7.4 и 7.6) последнее неравенство справедливо (оно усиливалось при применении расширенных конфигураций мини- и микроЭВМ). Следовательно, технические возможности производства и технологические достижения в элементной базе ЭВМ второго и третьего поколений позволяли при проектировании элементарной машины для ВС руководствоваться законом Гроша:

$$\omega' = hv^a,$$

где ω' — номинальное быстродействие ЭМ; v — цена ЭМ; h — коэффициент, имеющий размерность $a \geqslant 2$.

В современных большемасштабных ВС элементарная машина представляет собой микропроцессорную БИС (например, транспьютер, см. § 7.6, 8.3 и 8.6). Относительная цена локального коммутатора по сравнению с остальной частью такой машины достаточно мала. Следовательно, если закон Гроша выполняется для микропроцессорной ЭВМ, то он будет справедлив и для элементарной машины ВС; значит, его можно использовать при анализе технико-экономической эффективности проектируемых ЭМ.

С другой стороны, закон Гроша нельзя распространить на ВС как коллектив ЭМ. Если исходить из обратного и учитывать, что цена системы из N ЭМ равна $V = Nv$, то при расчете показателя производительности Ω ВС получим значение, прямо пропорциональное $(Nv)^a$, где $a \geqslant 2$. Такая зависимость находится в резком противоречии с закономерностью (9.1), установленной и теоретически, и практически [5, 6]:

$$\Omega = A_N N\omega,$$

где ω —показатель производительности одной ЭМ; $A_N \geqslant 1$.

Итак, для вычислительных систем справедлив следующий *закон*:

$$\Omega = \psi V,$$

выражающий взаимосвязь между показателем производительности Ω рассматриваемой ВС и ее ценой V; ψ — коэффициент, имеющий размерность.

При технико-экономической характеристике ВС (как и ЭВМ) широко используется и *цена одной операции в секунду*

$$\Sigma = V / \Omega. \qquad (12.1)$$

В § 2.5 приведены числовые значения цены одной операции в секунду для ЭВМ:

$$\sigma = v/\omega.$$

Характерно, что для ЭВМ первого поколения этот показатель составлял значительную величину — 10 долл./опер./с. С появлением ЭВМ второго и третьего поколений значения цены одной операции в секунду уменьшались на порядок.

У первых конвейерных и матричных ВС значения показателя Σ были того же порядка (табл. 12.1), что и значения σ для ЭВМ третьего поколения.

Однако ВС позволили достичь производительности на порядок выше, чем в ЭВМ третьего поколения.

Таблица 12.1

Технико-экономический показатель	Система		
	STAR-100	Cray-1	ILLIAC-IV
Цена операции, $\frac{\text{долл.}}{\text{опер.} \cdot \text{с}^{-1}}$	0,15	Не более 0,1	0,15

Архитектура и конвейерных ВС (см. гл. 4), и матричных ВС (см. гл. 5) характеризуется тем, что вычислительные процессы в них организуются единым устройством управления (точнее, хост-компьютером или сосредоточенным аппаратурно-программным комплексом). Дальнейшее уменьшение цены одной операции в секунду (12.1) достигается в модифицированных мультипроцессорных системах (см. гл. 6) и в системах с программируемой структурой (см. гл. 7). Названные системы обладают не только «распределенным» процессором и памятью (как это имеет место в матричных ВС), но и распределенным управлением, именно поэтому для них часто используют и обобщающее понятие *«распределенные ВС»*. Этот термин получил распространение за рубежом, по-видимому, в 80-х годах прошлого века. В отечественной литературе он использовался применительно к подклассу ВС с программируемой структурой (см. § 7.1). Ниже этот термин будет употребляться, когда речь пойдет о ВС с программируемой структурой, в которой ресурсы (ЭМ) рассредоточены в пространстве (так же, как в сетях).

Таким образом, дальнейшее уменьшение цены операции в средствах вычислительной техники может быть достигнуто в основном в результате применения не-фон-неймановских архитектурных решений.

12.2. Математическое ожидание бесполезных расходов при эксплуатации вычислительных систем

Пусть имеются вычислительная и восстанавливающая системы, состоящие соответственно из N элементарных машин и m устройств, $N \geqslant m \geqslant 0$; λ и μ — интенсивности соответственно отказов ЭМ (2.14) и восстановления отказавших ЭМ одним восстанавливающим устройством (ВУ) (2.18); c_1 и c_2 — стоимости соответственно эксплуатации одной ЭМ (2.27) и содержания одного ВУ (2.28) в единицу времени. Требуется оценить средние эксплуатационные расходы при работе ВС в переходном и стационарном режимах.

Ясно, что количество отказавших ЭМ и количество ВУ, занятых ремонтом машин, являются случайными числами. Отказавшие машины не могут быть использованы для коммерческих целей, следовательно, вычислительный центр, эксплуатирующий ВС, будет нести потери. Если количество отказавших ЭМ менее общего количества m восстанавливающих устройств, то будут иметь место и потери из-за простоя свободных ВУ. Однако если имеются ВУ, не занятые восстановлением, то отказ очередной ЭМ прекращает простой одного из свободных устройств.

Эксплуатационные расходы (потери), вызванные простоями машин ВС из-за отказов и простоями ВУ из-за недостатка отказавших ЭМ, будем называть *бесполезными*. Для оценки этих расходов используют функцию $\Gamma(i, t)$, являющуюся *математическим ожиданием бесполезных эксплуатационных расходов* к моменту времени $t \geqslant 0$ при условии, что система находилась в состоянии $i \in E_0^N = \{0, 1, 2, \ldots, N\}$ при $t = 0$ (т. е. в системе было i работоспособных ЭМ в момент начала функционирования ВС).

12.2.1. Расчет эксплуатационных расходов для переходного режима функционирования вычислительных систем

Рассмотрим начальный период работы ВС. Очевидно, что расчет функции бесполезных расходов $\Gamma(i, t)$, $i \in E_0^N$, включает вывод дифференциального уравнения, а последнее, в свою очередь, связано с оценкой ожидаемых расходов $\Gamma(i, t + \Delta t)$, $i \in E_0^N$, при эксплуатации ВС в течение времени $(t + \Delta t)$; Δt — малый промежуток времени. Эти расходы складываются из затрат за время t и издержек, которые происходят на промежутке времени $[t, t + \Delta t)$.

Как обычно (см. § 10.4), $\mathcal{N}(i, t)$ — математическое ожидание количества работоспособных машин в системе; $\mathcal{M}(i, t)$ — среднее количество занятых ВУ в момент $t \geqslant 0$ при условии, что при $t = 0$ были работоспособными $i \in E_0^N$ ЭМ, т. е. $\mathcal{N}(i, 0) = i$. Тогда $[N - \mathcal{N}(i, t)]$ и $[m - \mathcal{M}(i, t)]$ будут математическими ожиданиями количества отказавших ЭМ и количества свободных ВУ в момент $t \geqslant 0$ при $\mathcal{N}(i, 0) = i$, $i \in E_0^N$ Расходы из-за простоя машин (вследствие их отказа) и из-за простоя ВУ (вследствие отсутствия достаточного количества отказавших ЭМ) на промежутке времени $[t, t + \Delta t)$, очевидно, оцениваются величинами

$$[N - \mathcal{N}(i, t)]c_1\Delta t, \quad [m - \mathcal{M}(i, t]c_2\Delta t. \tag{12.2}$$

Тогда, учитывая оценки (12.2), можно записать

$$\Gamma(i,\ t+\Delta t)=\Gamma(i,\ t)+[N-\mathcal{N}(i,\ t)]c_1\Delta t+[m-\mathcal{M}(i,\ t]c_2\Delta t+o(\Delta t). \quad (12.3)$$

Если в формуле (12.3) член $\Gamma(i, t)$ перенести в левую часть, затем поделить на Δt обе части равенства и перейти к пределу при $\Delta t \to 0$, то получим дифференциальное уравнение

$$\frac{d}{dt}\Gamma(i,\ t)=[N-\mathcal{N}(i,\ t)]c_1+[m-\mathcal{M}(i,\ t]c_2. \quad (12.4)$$

При этом в качестве начальных условий естественно взять $\Gamma(i,\ 0)=0,\ i\in E_0^N$.

Для нахождения решения уравнения (12.4) необходимо знать вид функций $\mathcal{N}(i,\ t)$ и $\mathcal{M}(i,\ t)$. Классический подход к расчету (на основе методов теории массового обслуживания) здесь неприемлем. В этом легко убедиться, если подставить в (12.4) функции $\mathcal{N}(i,\ t)$ и $\mathcal{M}(i,\ t)$, выраженные через вероятности состояний системы (10.20), (10.21). Поэтому при расчете $\Gamma(i,\ t),\ i\in E_0^N$, воспользуемся методикой, изложенной в разд. 2.9.3 и § 10.4.

Случай 1. Пусть восстанавливающая система имеет высокую производительность, т. е. выполняется неравенство

$$N\lambda \leqslant m\mu.$$

Тогда подставив (10.28) и (10.29) в (12.4), получим

$$\frac{d}{dt}\Gamma(i,t)=\left[\frac{N\lambda}{\lambda+\mu}-\frac{i\lambda-(N-i)\mu}{\lambda+\mu}e^{-(\lambda+\mu)t}\right]c_1+\left[m-\frac{N\lambda}{\lambda+\mu}+\frac{i\lambda-(N-i)\mu}{\lambda+\mu}e^{-(\lambda+\mu)t}\right]c_2. \quad (12.5)$$

После интегрирования уравнения (12.5) находим:

$$\Gamma(i,t)=-\varepsilon_i+\gamma t+\varepsilon_i\delta(t),\ \ i\in E_{N-m}^N; \quad (12.6)$$

$$\delta(t)=\exp[-(\lambda+\mu)t];$$

$$\gamma=\frac{N\lambda}{\lambda+\mu}(c_1-c_2)+mc_2; \quad (12.7)$$

$$\varepsilon_i=\frac{i\lambda-(N-i)\mu}{(\lambda+\mu)^2}(c_1-c_2). \quad (12.8)$$

Очевидно, что решение (12.6) удовлетворяет начальному условию: $\Gamma(i,\ 0)=0,\ \ i\in E_0^N$

Следует заметить, что результаты (12.6) — (12.8), полученные для ВС с произвольным количеством N ЭМ, уже известны для предельного варианта системы — одной ЭМ, т. е. для $N=1$. В самом деле, если в (12.7) и (12.8)

подставить $N = 1$, $m = 1$, $i = 0$ или $i = 1$, то получим (2.36) и (2.38). Более того, здесь просматривается глубокая связь: в дифференциальном уравнении для $\Gamma(i, t)$ в случае систем (12.4) используется математическое ожидание $\mathcal{N}(i, t)$ (10.28) количества работоспособных машин ВС, а в случае ЭВМ (2.34) — функция $s(i, t)$ (2.24), (2.25) готовности машины. Но при $N = 1$ функция $\mathcal{N}(0, t)$ (10.28) совпадает с $s(0, t)$ (2.24), а $\mathcal{N}(1, t)$ (10.28) — с $s(1, t)$ (2.25).

Таким образом, подход, предложенный в разд. 2.9.3 для расчета математического ожидания бесполезных расходов при эксплуатации ЭВМ, легко обобщается на случай многомашинных ВС.

Случай 2. Восстанавливающая система имеет низкую производительность, т. е. неравенство (10.33) не выполняется. После подстановки (10.39) и $\mathcal{M}(i, t) = m$ в (12.4) получим дифференциальное уравнение

$$\frac{d}{dt}\Gamma(i, t) = \left(N - \frac{m\mu}{\lambda} - \frac{i\lambda - m\mu}{\lambda}e^{-\lambda t}\right)c_1. \tag{12.9}$$

Интегрирование уравнения (12.9) показывает, что выражение (12.6) при $i \in E_0^{N-m-1}$,

$$\delta(t) = \exp(-\lambda t);$$

$$\gamma = (N\lambda - m\mu)\lambda^{-1}c_1; \tag{12.10}$$

$$\varepsilon_i = (i\lambda - m\mu)\lambda^{-2}c_1 \tag{12.11}$$

является математическим ожиданием бесполезных расходов при эксплуатации ВС в условиях низкой производительности восстанавливающей системы.

12.2.2. Расчет эксплуатационных расходов для стационарного режима функционирования вычислительных систем

Рассмотрим работу ВС при длительной эксплуатации. В § 10.4 было показано (см. (10.30), (10.31), (10.40)), что предельные значения $\mathcal{N}(i, t)$ и $\mathcal{M}(i, t)$ при $t \to \infty$ не зависят от начального состояния системы $i \in E_0^N$. Если это так, то и средние бесполезные эксплуатационные расходы ВС не будут зависеть от ее начального состояния:

$$\Gamma(t) = \lim_{t\to\infty}\Gamma(i, t). \tag{12.12}$$

Тогда на основании (12.4), (10.30), (10.31) и (10.40) можно записать дифференциальные уравнения для математического ожидания бесполезных экс-

плуатационных расходов в условиях стационарного режима работы ВС (12.12):

$$\frac{d}{dt}\Gamma(t)=\begin{cases}\dfrac{N\lambda}{\lambda+\mu}c_1+\left(m-\dfrac{N\lambda}{\lambda+\mu}\right)c_2 & \text{при } N\lambda \leqslant m\mu;\\ (N-\dfrac{m\mu}{\lambda})c_1 & \text{в противном случае.}\end{cases} \tag{12.13}$$

Из уравнения (12.13) следует, что

$$\Gamma(t)=\gamma t, \tag{12.14}$$

где γ определяется по формулам (12.7) и (12.10). Таким образом, при оценке бесполезных эксплуатационных расходов в условиях длительной работы ВС можно использовать простейшую формулу (12.14).

Формулу (12.14) можно вывести иначе, т. е. непосредственно из решения (12.6) для переходного режима работы ВС. Действительно, при $t \to \infty$ в (12.6) функция $\delta(t) \to 0$, а величина $-\varepsilon_i$ не может оказать заметного влияния на произведение γt и в случае $N\lambda \leqslant m\mu$, и в случае $N\lambda > m\mu$. Следовательно, формулы можно трансформировать в (12.14). При этом величина γ, как следует из (12.14), выражает средние бесполезные расходы в единицу времени при длительной эксплуатации ВС, ее называют *коэффициентом эксплуатационных потерь ВС.*

Очевидно, что выполнить расчет γ можно и классически, т. е. через вероятности состояний системы. При таком способе расчета вместо обозначения γ будем употреблять γ^* Тогда

$$\gamma^* = (N-\mathcal{N})c_1+(m-\mathcal{M})c_2, \tag{12.15}$$

где величины $\mathcal{N}=\lim\limits_{t\to\infty}\mathcal{N}(i,\ t)$, $\mathcal{M}=\lim\limits_{t\to\infty}\mathcal{M}(i,\ t)$ вычисляются через вероятности состояний системы $P_j=\lim\limits_{t\to\infty}P_j(i,\ t)$, $i\in E_0^N$, $j=\overline{0,\ N}$. Подставив предельные значения (10.20) и (10.21) при $t \to \infty$ в (12.15), получим

$$\gamma^* = c^1\sum_{j=0}^{N-m-1}(N-j)P_j+c_2\sum_{j=N-m}^{N}(m+j-N)P_j. \tag{12.16}$$

Здесь $\{P_j\}$ $(j\in E_0^N)$ — стационарное распределение вероятностей состояний ВС, а сами вероятности рассчитываются по формулам (9.50), (9.51).

Очевидно, что расчет по (12.16) существенно сложнее, чем по (12.7) или (12.10). Однако (12.16) позволяет:

1) установить степень адекватности систематически излагаемого здесь подхода с классическим, принятым в теории массового обслуживания;

2) определить оптимальное количество m^* восстанавливающих устройств, т. е. такое количество устройств, при котором достигается минимум γ^* (12.16).

В качестве показателя, оценивающего степень близости γ (12.7) и γ^* (12.16), возьмем

$$\Delta\gamma = |\gamma^* - \gamma| / \gamma^* \qquad (12.17)$$

Численные эксперименты показывают, что значения γ и γ^* достаточно близки; при $N \geqslant 100$ справедливо $\Delta\gamma < 0{,}1$. Расчеты показывают, что ВС достаточно быстро входят в стационарный режим работы, а условию $N\lambda \leqslant m\mu$ легко удовлетворить. Следовательно, на практике для расчета математического ожидания бесполезных расходов ВС можно широко применять простейшие формулы (12.7), (12.14).

Оптимальное количество m^* восстанавливающих устройств при заданных N, c_1, c_2, λ, μ можно вычислить по (12.16). Такой процесс определения m^* достаточно трудоемок, и его применение в инженерной практике весьма ограничено. С другой стороны, при оптимальном количестве m^* выполняется условие (10.33) и, следовательно, допустимо использование формулы (12.7). Однако из (12.7) видно, что чем меньше m, тем меньше потери в единицу времени. Поэтому из (12.7) нельзя определить оптимальное количество восстанавливающих устройств. Численные расчеты показывают, что при $N \to \infty$ имеет место

$$N / m^* \to (\lambda + \mu)\lambda^{-1},$$

поэтому для многомашинных ВС при расчете значений m', близких к m^*, можно использовать следующую формулу:

$$m' =]N\lambda/(\lambda + \mu)[, \qquad (12.18)$$

где $]x[$ — ближайшее к x целое число и такое, что $]x[\geqslant x$.

Формула (12.18) следует и из интуитивных соображений. Так, количество восстанавливающих устройств будет близко к оптимальному, если оно равно ожидаемому количеству отказавших ЭМ при длительной эксплуатации ВС. Следовательно, можно записать, что $m' \approx N - \mathcal{N}$ Подставив в последнее выражение значение $\mathcal{N}$ (10.30), получим $m' \approx N\lambda(\lambda + \mu)^{-1}$, что подтверждает справедливость (12.18).

Таким образом, при определении для многомашинной ВС количества восстанавливающих устройств, близкого к оптимальному, достаточно воспользоваться простой формулой (12.18).

12.3. Математическое ожидание дохода вычислительных систем

Определим функцию $D(i,\ t)$, $i \in E_0^N$, — *математическое ожидание дохода*, который принесет ВС за время $t \geqslant 0$, если она начинает функционировать с i исправными машинами. При расчете $D(i,\ t)$, $i \in E_0^N$, потребуется как известные величины N, m, λ, μ, c_1 и c_2 (см. § 12.2), так и новые: c_2' — себестоимость содержания одного восстанавливающегося устройства в единицу времени (2.28); c_3 — стоимость запасных технических средств, расходуемых при однократном восстановлении отказавшей ЭМ (2.29).

Очевидно, что расчет $D(i,\ t)$ может быть выполнен с помощью аппарата марковских процессов с доходами [5, 8]. Однако он трудоемок и получение числовых значений для $D(i,\ t)$ немыслимо без использования средств вычислительной техники. В [5] показано, что континуальный подход (или метод динамики средних [25]) здесь также работает, а числовые результаты, получаемые этими методами, хорошо согласуются. Поэтому в данном параграфе ограничимся изложением метода, основанного на континуальной модели функционирования ВС (см. § 10.3, 10.4). Этот метод является обобщением способа, описанного в разд. 2.9.4 для ЭВМ, на случай вычислительных систем.

12.3.1. Расчет дохода для переходного режима работы вычислительных систем

Рассмотрим прежде всего работу ВС на начальном участке времени. Для этого оценим математическое ожидание дохода ВС к моменту $(t + \Delta t)$ с учетом малых порядка не выше Δt. Очевидно, что $D(i, t + \Delta t)$, $i \in E_0^N$, определяется как сумма доходов, ожидаемых на промежутках времени $[0, t)$ и $[t, t + \Delta t)$, за вычетом расходов по содержанию восстанавливающих устройств и средней стоимости технических средств, устанавливаемых вместо отказавших при ремонте ЭМ в течение времени Δt.

Математическое ожидание дохода, приносимого при эксплуатации ВС за промежуток времени $[0,\ t)$, по определению равно $D(i,\ t)$, $i \in E_0^N$ Доход при эксплуатации ВС в промежутке времени $[t, t + \Delta t)$ определяется

работой исправных ЭМ. Если в момент t в системе в среднем исправно $\mathcal{N}(i,\ t)$ ЭМ и стоимость единицы полезного времени работы одной ЭМ составляет c_1, то средний доход ВС в промежутке $[t,\ t+\Delta t)$ будет равен

$$\mathcal{N}(i,\ t)c_1\Delta t. \tag{12.19}$$

С другой стороны, c_2' — себестоимость содержания одного из m ВУ в единицу времени и, следовательно, величина

$$mc_2'\,\Delta t \tag{12.20}$$

оценивает расходы по содержанию восстанавливающих устройств в течение времени Δt. Наконец, если в момент времени t ремонтом занято в среднем $\mathcal{M}(i,\ t)$ ВУ, то математическое ожидание расходов на запасные технические средства, используемые на восстановление отказавших машин в промежутке $[t, t+\Delta t)$, будет равно

$$\mathcal{M}(i,\ t)c_3\mu\Delta t, \tag{12.21}$$

где c_3 — средняя стоимость запасных технических средств, используемых при однократном восстановлении одной ЭМ; μ — среднее число восстановлений, производимых одним ВУ.

Учитывая приведенные оценки (12.19)–(12.21), можно записать

$$D(i,\ t+\Delta t) = D(i,\ t) + \mathcal{N}(i,\ t)c_1\Delta t - mc_2'\,\Delta t - \mathcal{M}(i,\ t)c_3\mu\Delta t + o(\Delta t). \tag{12.22}$$

От формулы (12.22) очевиден переход к дифференциальному уравнению

$$\frac{d}{dt}D(i,t) = \mathcal{N}(i,\ t)c_1 - m_2' - \mathcal{M}(i,\ t)c_3\mu. \tag{12.23}$$

При этом в качестве начальных условий естественно принять $D(i,\ 0) = 0,\ i \in E_0^N$

При вычислении $D(i,\ t)$, $i \in E_0^N$, воспользуемся результатами, приведенными в § 10.4 и 12.2. Учитывая, что функции $\mathcal{N}(i,\ t)$ и $\mathcal{M}(i,\ t)$ имеют вид (10.28) и (10.29) при выполнении неравенства (10.33) и вид (10.39) и $\mathcal{M}(i,\ t) = m$ при невыполнении (10.33), можно найти решение уравнения (12.23) при заданных начальных условиях. Это решение имеет следующий вид:

$$D(i,\ t) = D_i + gt - D_i\delta(t), \tag{12.24}$$

где $i \in E_{N-m}^N$; $\delta(t) = \exp[-(\lambda+\mu)t]$;

$$g = \frac{N\mu}{\lambda + \mu}(c_1 - \lambda c_3) - mc_2'; \tag{12.25}$$

$$D_i = \frac{i\lambda - (N-i)\mu}{(\lambda + \mu)^2}(c_1 + \mu c_3) \tag{12.26}$$

при выполнении неравенства $N\lambda \leqslant m\mu$ или $i \in E_0^{N-m-1}$;

$$\delta(t) = \exp(-\lambda t);$$

$$g = \frac{m\mu}{\lambda}(c_1 - \lambda c_3) - mc_2'; \tag{12.27}$$

$$D_i = \varepsilon_i = (i\lambda - m\mu)\lambda^{-2}c_1$$

при выполнении $N\lambda > m\mu$.

Таким образом, сложность расчета математического ожидания дохода, приносимого ВС, остается такой же, что и при расчете средних эксплуатационных расходов.

12.3.2. Расчет дохода для стационарного режима работы ВС

Исследуем поведение ВС в условиях ее длительной эксплуатации. Для этого в дифференциальное уравнение (12.23) подставим предельные значения $\mathcal{N}(i, t)$ и $\mathcal{M}(i, t)$ при $t \to \infty$ и учтем, что доход при эксплуатации системы в стационарном режиме не зависит от ее начального состояния, т. е.

$$D(t) = \lim_{t\to\infty} D(i, t).$$

Тогда

$$\frac{d}{dt}D(t) = \mathcal{N}c_1 - mc_2' - \mathcal{M}c_3\mu. \tag{12.28}$$

После подстановки значений $\mathcal{N}$ и $\mathcal{M}$, вычисленных в § 10.4, в уравнение (12.28) получим

$$\frac{d}{dt}D(t) = \begin{cases} \dfrac{N\mu}{\lambda+\mu}c_1 - mc_2' - \dfrac{N\lambda\mu}{\lambda+\mu}c_3 & \text{при } N\lambda \leqslant m\mu; \\ \dfrac{m\mu}{\lambda}c_1 - mc_2' - mc_3\mu & \text{при } N\lambda > m\mu. \end{cases} \tag{12.29}$$

Решая уравнения (12.29), находим, что

$$D(t) = gt, \tag{12.30}$$

где g определяется по (12.25) и (12.27). Из (12.30) видно, что величина g является средним доходом, приносимым в единицу времени при длительной эксплуатации ВС; ее называют *прибылью* ВС.

Формула (12.30) следует и из общего решения (12.24). В самом деле, при длительной эксплуатации ВС, т. е. при $t \to \infty$ функция $\delta(t) \to 0$ и имеет место неравенство $D_i \ll gt$.

Рассчитать прибыль ВС можно и с помощью аппарата марковских процессов с доходами [5, 8]. Преодолевая известные вычислительные трудности такого расчета, можно установить степень точности расчета прибыли ВС, выполненного в данном параграфе. Пусть g и g^* — значения прибыли, которые вычислены соответственно по (12.25) и с помощью аппарата марковских процессов с доходами

$$\Delta g = |g^* - g| / g^*$$

Расчеты показали, что при числе машин в ВС не менее 100 обеспечивается выполнение неравенства $\Delta g \leqslant 0,01$. Таким образом, точность расчета прибыли g на порядок выше точности вычисления γ (см. § 12.2).

При определении для заданной ВС количества восстанавливающих устройств, близкого к оптимальному (т. е. к числу, дающему максимум g), достаточно воспользоваться известной формулой (12.18).

12.4. Технико-экономическое исследование структур вычислительных систем в условиях потока задач

До сих пор мы рассматривали экономические вопросы работы ВС как коллективов ЭМ, в которых структура взаимосвязей между ЭМ и процесс поступления задач на ВС не находили отражения в явном виде. Приведенные в § 12.1–12.3 результаты позволяют оценить лишь потенциальные возможности ВС по обеспечению их технико-экономической эффективности. В данном параграфе будет введен показатель эффективности ВС, который учитывает экономические характеристики каналов межмашинной связи, структуру ВС, распределения машин и терминалов ВС по вычислительным центрам и характеристики потоков задач, поступающих в систему через каждый терминал. Будет также указан путь организации стохастически оптимального функционирования ВС с использованием введенного показателя и аппарата стохастического программирования.

12.4.1. Оценка потерь при обслуживании вычислительных систем потока задач

Пусть имеется распределенная ВС с программируемой структурой (см. разд. 7.1.2 и § 12.1), состоящая из N элементарных машин и обслужи-

вающая L терминалов. Машины и терминалы размещены на H вычислительных центрах (ВЦ). Обозначим N_i — количество ЭМ, а L_i — количество терминалов, принадлежащих i-му ВЦ, $i \in E_1^H = \{1, 2, \ldots, H\}$. Очевидно, что

$$\sum_{i=1}^{H} N_i = N, \quad \sum_{i=1}^{H} L_i = L. \tag{12.31}$$

На каждый из терминалов поступает поток задач различной сложности (см. разд. 3.3.4). Считается, что задача, поступающая на терминал $j \in E_1^L = \{1, 2, \ldots, L\}$, представлена адаптирующейся параллельной программой, количество $\tilde{a}_j$ ветвей в которой допускает варьирование от a_j до a_j^*; или, иначе говоря, считается, что такая задача имеет переменный ранг $\tilde{a}_j$ и такой, что $a_j \leqslant \tilde{a}_j \leqslant a_j^*$. Минимальный и максимальный ранги a_j и a_j^* выбираются на этапе программирования с учетом сложности и структуры решаемой задачи, и других условий. По определению (см. § 10.1) адаптирующаяся программа допускает реализацию на системах (подсистемах) с переменным рангом (или количеством машин). Ранг подсистемы в данном случае соответствует рангу задачи $\tilde{a}_j$ и заключен в границы от a_j до a_j^*, $j \in E_1^L$ В пределах подсистемы обеспечивается возможность передачи информации по сети связей между любыми ЭМ. Считается, что все величины $\tilde{a}_j$, $j \in E_j^L$, являются независимыми случайными величинами.

Из сказанного следует, что спрос на подсистемы различных рангов носит случайный характер. Допускается, что всегда можно определить средний спрос на подсистемы различных рангов в зависимости от диапазонов допустимых рангов для задач (т. е. от заложенных в их параллельные программы возможностей по варьированию количества ветвей).

Пусть $p_j(a_j)$ и $p_j^*(a_j^*)$ — вероятности того, что для терминала $j \in E_1^L$ потребуются подсистемы рангов соответственно a_j и a_j^*, т. е. $p_j(a_j)$ и $p_j^*(a_j^*)$ — вероятности запроса с терминала j соответственно a_j и a_j^* связных ЭМ. Очевидно, что

$$\sum_{a_j=0}^{\infty} p_j(a_j) = 1, \quad \sum_{a_j^*=0}^{\infty} p_j^*(a_j^*) = 1. \tag{12.32}$$

Тогда средние спросы на терминале j при минимально и максимально требуемых рангах подсистем для решения задач потока соответственно равны:

$$\rho_j = \sum_{a_j=0}^{\infty} a_j p_j(a_j); \quad \rho_j^* = \sum_{a_j^*=0}^{\infty} a_j^* p_j^*(a_j^*). \tag{12.33}$$

Обозначим через x_j ранг подсистемы, которая назначается на терминал $j \in E_1^L$. Говоря иначе, a_j и a_j^* — соответственно минимальный и максимальный ранги подсистем, которые требуются, а x_j — ранг подсистемы, которая может быть использована на терминале $j \in E_1^L$

Найдем оценки для компонентов, составляющих суммарные потери из-за «неправильного» распределения машин ВС по терминалам. Пусть c_j — средняя стоимость пользования одной ЭМ распределенной ВС с терминала $j \in E_1^L$ Считается, что при выделении для каждого терминала машин сверх требуемого спроса имеют место простои избыточных ЭМ. Тогда потери для терминала $j \in E_1^L$ при решении задачи, ранг которой не может быть больше a_j^*, составят $c_j(x_j - a_j^*)$. При необеспечении терминала минимально необходимым количеством ЭМ считается, что все выделенные этому терминалу ЭМ простаивают, и имеют место потери из-за нерешения задачи. В последнем случае общие потери для терминала $j \in E_1^L$ оцениваются величиной $(c_j x_j + K_j \rho_j^*)$, где $K_j \rho_j^*$ — штраф за неудовлетворение спроса на терминале; K_j — коэффициент штрафа. Ясно, что, задавая K_j соответствующим образом, можно достичь реализации политики приоритетов терминалов, если, например, $K_j = f(j)$, или приоритетов ВЦ, если, например, $K_j = f(k_j)$, где k_j — номер ВЦ, содержащего терминал $j \in E_1^L$, $k_j \in E_1^H$

Среднее количество избыточных ЭМ для терминала $j \in E_1^L$ равно

$$\sum_{a_j^*=0}^{x_j} (x_j - a_j^*) p_j^*(a_j^*) = x_j - \rho_j^* + \sum_{a_j^*=x_j}^{\infty} (a_j^* - x_j) p_j^*(a_j^*),$$

что вытекает из (12.32) и (12.33). Следовательно, ожидаемые потери от избытка ЭМ для терминала $j \in E_1^L$ составят

$$c_j(x_j - \rho_j^*) + c_j \sum_{a_j^*=x_j}^{\infty} (a_j^* - x_j) p_j^*(a_j^*). \tag{12.34}$$

Среднее количество ЭМ, не достающих до максимального требуемого для терминала $j \in E_1^L$, равно

$$n_j(x_j) = \sum_{a_j^* = x_j}^{\infty} (a_j^* - x_j) p_j^*(a_j^*).$$

Введем функцию

$$\delta_j(x_j) = \begin{cases} 1, & \text{если } n_j(x_j) > \rho_j^* - \rho_j; \\ 0, & \text{если } n_j(x_j) \leqslant \rho_j^* - \rho_j, \end{cases} \tag{12.35}$$

тогда ожидаемые потери от недостатка ЭМ для терминала $j \in E_1^L$ могут быть представлены в следующем виде:

$$\delta_j(x_j)(c_j x_j + K_j \rho_j^*). \tag{12.36}$$

Оценим издержки, связанные с использованием каналов связи между ВЦ. Пусть c_{ik} — минимальная стоимость использования каналов, обеспечивающих связь между i-м и k-м ВЦ, $i, k \in E_1^H$ Матрица $\|c_{ik}\|$ характеризует издержки при передаче информации по сети связей между ВЦ. Допускается, что передача информации между ЭМ в пределах одного ВЦ требует пренебрежимо малых расходов по сравнению с пересылкой данных между машинами различных ВЦ. Обозначим через κ_{ij}, $i \in E_1^H$, $j \in E_1^L$, количество ЭМ i-го ВЦ, используемых j-м терминалом, и введем функцию $\delta_{ij}(\kappa_{ij})$ такую, что

$$\delta_{ij}(\kappa_{ij}) = \begin{cases} 1, & \text{если } \kappa_{ij} > 0; \\ 0, & \text{если } \kappa_{ij} = 0. \end{cases} \tag{12.37}$$

Тогда издержки при передаче информации между j-м терминалом и всеми элементарными машинами ВС и потери вследствие использования каналов связи между ВЦ будут соответственно равны:

$$\sum_{i=1}^{H} \delta_{ij}(\kappa_{ij}) c_{ik_j}; \quad \sum_{j=1}^{L} \sum_{i=1}^{H} \delta_{ij}(\kappa_{ij}) c_{ik_j}. \tag{12.38}$$

Выпишем *суммарные ожидаемые потери* Z при функционировании распределенной ВС коллективного пользования, обслуживающей поток задач:

$$Z = \sum_{j=1}^{L} \left[c_j(x_j - \rho_j^*) + c_j \sum_{a_j^* = x_j}^{\infty} (a_j^* - x_j) p_j^*(a_j^*) + \delta_j(x_j)(c_j x_j + K_j \rho_j^*) + \sum_{i=1}^{H} \delta_{ij}(\kappa_{ij}) c_{ik_j} \right], \tag{12.39}$$

что следует из (12.34), (12.36), (12.38).

Пусть спрос на подсистемы различных рангов подчинен пуассоновскому закону:

$$p_j(a_j)=\frac{\rho_j^{a_j}}{a_j!}e^{-\rho_j};\quad p_j^*(a_j^*)=\frac{(\rho_j^*)^{a_j^*}}{a_j^*!}e^{-\rho_j^*}, \tag{12.40}$$

$p_j(a_j)$ и $p_j^*(a_j^*)$ — плотности вероятностей случайных величин соответственно a_j и a_j^*.

Из (12.40) следует, что

$$a_j^* p_j^*(a_j^*)=\rho_j^* p_j^*(a_j^*-1),\quad a_j^*\geqslant 1. \tag{12.41}$$

Таким образом, на основании (12.41) можно записать

$$\sum_{a_j^*=x_j}^{\infty}(a_j^*-x_j)p_j^*(a_j^*)=\begin{cases}\rho_j^* P_j^*(x_j-1)-x_j P_j^*(x_j), & \text{если } x_j\gg 1;\\ \rho_j^*, & \text{если } x_j=0,\end{cases} \tag{12.42}$$

где

$$P_j^*(x_j)=\sum_{a_j^*=x_j}^{\infty}p_j^*(a_j^*) \tag{12.43}$$

— интегральная функция распределения случайной величины a_j^*.

Подставляя (12.42) в (12.39), получаем

$$Z=\sum_{j=1}^{L}\left\{\sum_{i=1}^{H}\delta_{ij}(\kappa_{ij})c_{ik_j}+\delta_j(x_j)(c_j x_j+K_j\rho_j^*)+c_j(x_j-\rho_j^*)+\right.$$
$$\left.+c_j\left[\rho_j^* P_j^*(x_j-1)-x_j P_j^*(x_j)\right]\right\}. \tag{12.44}$$

Последняя формула (12.44), а также формулы (12.33), (12.35), (12.37), (12.40), (12.43) позволяют осуществить анализ эксплуатационных потерь, имеющих место в распределенных ВС, обслуживающих потоки задач с терминалов.

12.4.2. Организация стохастически оптимального функционирования вычислительных систем

Построение высокопроизводительных ВС коллективного пользования немыслимо без эффективных методов организации функционирования, без методов, которые бы позволяли в процессе решения задач потока не только

достигать стохастически предельной эффективности в работе систем, но были бы сами эффективно реализуемы на средствах ВТ. Примером такого метода может служить стохастическое программирование [27], эффективность которого следует из того, что необходимые вероятностные задачи решаются только один раз для заданной ВС и заданной статистики спросов на ее ресурсы. Кроме того, вычислительные трудности приемов решения легко преодолеваются путем применения исследуемой ВС.

Очевидно, что для целей организации стохастически оптимального функционирования распределенной ВС коллективного пользования требуется решить следующую задачу. Найти x_j и κ_{ij}, $j \in E_1^L$, $i \in E_1^H$, которые обеспечивали бы *минимум целевой функции* (12.44), т. е.

$$\min\ Z \tag{12.45}$$

при условиях, что x_j и κ_{ij} — неотрицательные целые числа,

$$\sum_{j=1}^{L} \kappa_{ij} = N_i; \tag{12.46}$$

$$\sum_{i=1}^{H} \kappa_{ij} - x_j = 0 \tag{12.47}$$

имеют место формулы (12.31), (12.33), (12.37), (12.40), (12.43) и что

$$\delta_j(x_j) = \begin{cases} 1, & \text{если } \rho_j^* P_j^*(x_j - 1) - x_j P_j^*(x_j) > \rho_j^* - \rho_j; \\ 0 & \text{в противном случае.} \end{cases} \tag{12.48}$$

Задача стохастического программирования (12.45)–(12.48) с требованием целочисленности и большим числом ограничений не может быть точно решена известными методами математического программирования. Кроме того, вероятностный характер задачи не может оправдать затрат на поиск точного оптимума, тем более что для операционных систем ВС более ценным является быстрое и гарантированное получение хотя бы приближенного (или субоптимального) решения, чем получение точного решения через значительное время (без полной уверенности в успехе).

Задача эффективно может быть решена методом цепей Монте-Карло, который заключается в случайном отыскании лучшего распределения, находящегося на некотором расстоянии от базового. Найденное распределение становится новым базовым, относительно которого снова отыскивается случайным образом лучшее распределение. Этот процесс продолжается до тех пор, пока в течение некоторого времени не будут наблюдаться улучшения. Полученное таким образом распределение является наилучшим относитель-

но исходного базового, что позволяет ограничиться случайным поиском только в его окрестности. Метод цепей Монте-Карло применительно к проблеме организации оптимального функционирования вычислительных систем в мультипрограммных режимах изложен с достаточной полнотой в [5].

12.4.3. Численное технико-экономическое исследование типовых структур распределенных вычислительных систем

Проиллюстрируем описанный в разд. 12.4.1 и 12.4.2 метод организации стохастически оптимального функционирования распределенных конфигураций ВС. Пусть имеются восемь ВЦ, на каждом из которых находятся по одной ЭМ и одному терминалу, т. е. имеет место $H = N = L = 8$, $N_i = L_i = 1$, $i = \overline{1, 8}$. На терминалы поступают потоки задач различных рангов, которые устанавливают средние спросы (12.33) на подсистемы в соответствии с табл. 12.2. Считается, что и ЭМ, и терминал, находящиеся на одном ВЦ, имеют одинаковые номера, равные номеру ВЦ.

Таблица 12.2

	Номер терминала							
j	1	2	3	4	5	6	7	8
ρ_j	1	1	4	2	2	2	5	5
ρ_j^*	1	5	4	4	3	2	6	7

Далее, пусть стоимость пользования одной ЭМ не зависит ни от номера ЭМ, ни от номера j терминала; для определенности положим, что для любого $j \in \{1, 2, \ldots, 8\}$ имеет место $c_j = 2$. Допустим также, что стоимость использования каналов, обеспечивающих связь между машинами i и k $i, k \in \{1, 2, \ldots, 8\}$, равна $c_{ik} = cl_{ik}$, где l_{ik} — минимальное число каналов, образующих информационную цепь между ЭМ с номерами i и j (или l_{ik} — расстояние между вершинами — машинами графа, представляющего структуру ВС); c — средняя стоимость использования канала между соседними ЭМ. Для определенности примем $c = 3$. Кроме того, для простоты допустим, что $K_j = K$, $j = \overline{1, 8}$, и будем учитывать три значения K: 0, 2, 10.

Требуется найти такое распределение машин по терминалам, при котором достигается минимум суммарных потерь Z (12.44) для следующих распределенных конфигураций:

1) ВС со структурой в виде $Л(8, 4, 4)$-графа (см. разд. 7.2.1 и рис. 7.4, *а*);

2) ВС со структурой в виде $Л(8, 3, 4)$-графа;

3) ВС с кольцевой структурой (когда каждая ЭМ i связана непосредственно с машиной j, где $j = i+1 \pmod 8$; $i, j \in \{1, 2, \ldots, 8\}$;

4) линейной ВС («кольцевой» ВС, в которой отсутствует непосредственная связь между машинами 1 и 8);

5) ВС с радиальной структурой (при которой ЭМ с номером 1 имеет непосредственную связь с машинами 2, 3, 4, 5, 6, 7, 8).

Следует отметить, что реализация в пределах систем МИКРОС-1 и МИКРОС-2 (см. § 7.6) конфигураций 3 и 4 основывается на использовании в каждой ЭМ по одному системному устройству, реализация конфигураций 1 и 2 требует по два системных устройства для каждой ЭМ, наконец, реализация конфигурации 5 рассчитана на применение в машинах с номерами i, $2 \leqslant i \leqslant 8$, по одному системному устройству, а в ЭМ с номером 1 четырех таких устройств.

Таблица 12.3

Структура	Характеристика ВС		
	K	$Z_{\min}$	ΔZ, %
$Л(8, 4, 4)$-граф	0	17, 18	48, 8
	2	60, 14	37, 8
	10	250, 14	29,2
$Л(8, 3, 4)$-граф	0	20, 18	49,0
	2	69, 52	32,8
	10	238, 14	33,8
Кольцевая	0	22, 57	67,9
	2	68, 80	43,4
	10	237, 30	37,1
Линейная	0	19, 60	65,9
	2	73, 66	39,4
	10	247, 16	34,5
Радиальная	0	21, 93	54,9
	2	70, 48	37,3
	10	234, 28	36,4

Результаты решения задачи оптимального распределения машин по терминалам для распределенных конфигураций ВС приведены в табл. 12.3 и на рис. 12.1–12.5. Значения целевой функции (12.44), обозначенные как $Z_{\min}$ (см. табл. 12.3), получены методом цепей Монте-Карло. При отыскании $Z_{\min}$ проводилось около тысячи испытаний и просмотр распределений машин по терминалам осуществлялся на расстоянии $h = 15$ от начального базового распределения. В качестве расстояния между двумя последова-

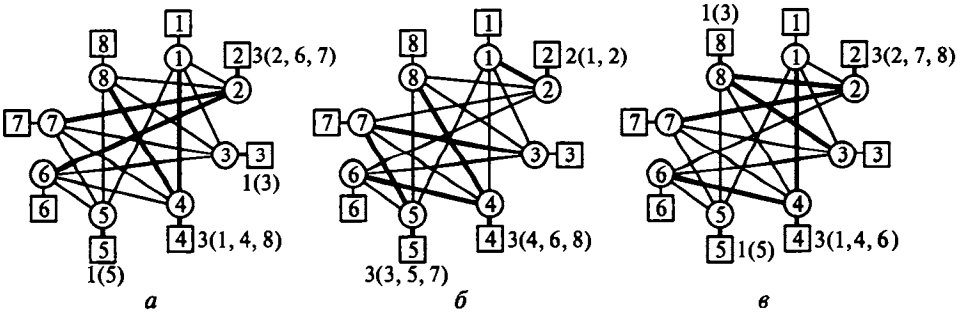


Рис. 12.1. Распределение машин по терминалам для ВС со структурой в виде $Л(8, 4, 4)$-графа:

а — $K = 0$; *б* — $K = 2$; *в* — $K = 10$; ○ — ЭМ; □ — терминал; —— — связь, выделенная для обменов между ЭМ или между ЭМ и терминалом

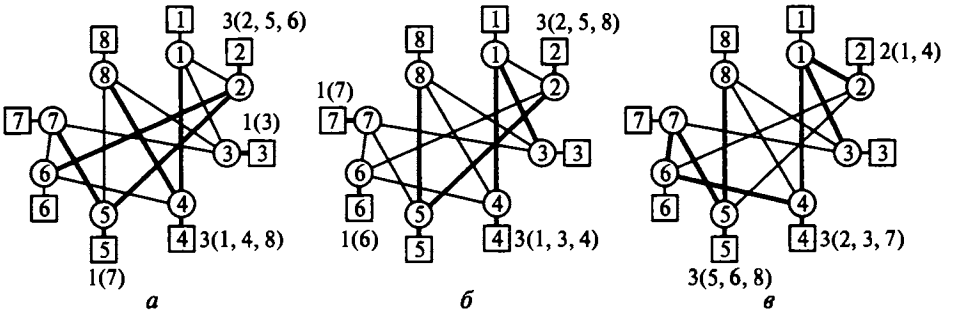


Рис. 12.2. Распределение машин по терминалам для ВС со структурой в виде $Л(8, 3, 4)$-графа:

а — $K = 0$; *б* — $K = 2$; *в* — $K = 10$; ○ — ЭМ; □ — терминал; —— — связь, выделенная для обменов между ЭМ или ЭМ и терминалом

тельностями номеров принималось число номеров во второй последовательности, которые не следовали за теми же номерами, как это имело место в первой последовательности. Метод получения последовательности с расстоянием, не большим h (относительно исходной), основан на псевдослучайных числах. В качестве начального базового распределения машин по терминалам может выступать произвольное из допустимых распределений. В данном случае взято распределение машин, которое обеспечивало минимальные потребности терминалов (см. табл. 12.2), причем потребности удовлетворялись в порядке возрастания номеров свободными ЭМ с близлежащих ВЦ (находящихся на наименьших расстояниях от рассматриваемого терминала).

В табл. 12.3 приведены также значения показателя

$$\Delta Z = \frac{(Z_{\text{баз}} - Z_{\min})}{Z_{\text{баз}}} \cdot 100\%, \qquad (12.49)$$

где $Z_{\text{баз}}$ — значение суммарных ожидаемых потерь (12.44) для исходного базового распределения машин по терминалам. Показатель (12.49) характеризует качество окончательного распределения, по нему можно судить об относительном уменьшении ожидаемых потерь при применении оптимизации.

На каждом из рис. 12.1 — 12.5 отражено три варианта распределения ЭМ по терминалам; запись $l(i, k, \ldots, f)$, соотнесенная с квадратом j, означает, что l машин с номерами $i, k, \ldots, f$ назначены для обслуживания терминала j.

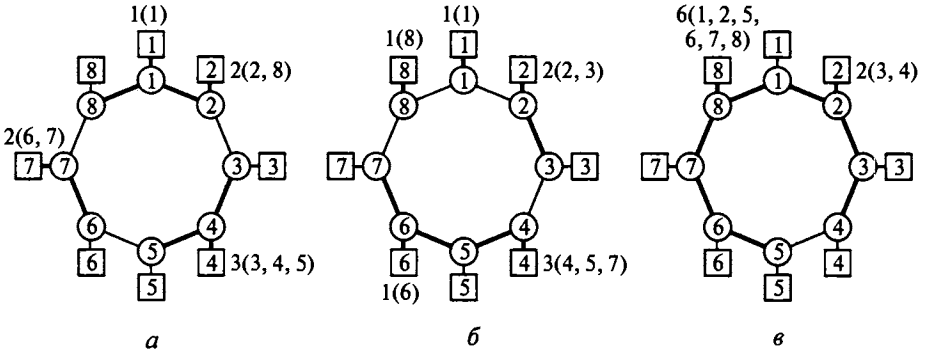


Рис. 12.3. Распределение машин по терминалам кольцевой ВС:

а — $K = 0$; *б* — $K = 2$; *в* — $K = 10$; ○ — ЭМ; □ — терминал; —— — связь, выделенная для обменов между ЭМ или между ЭМ и терминалом

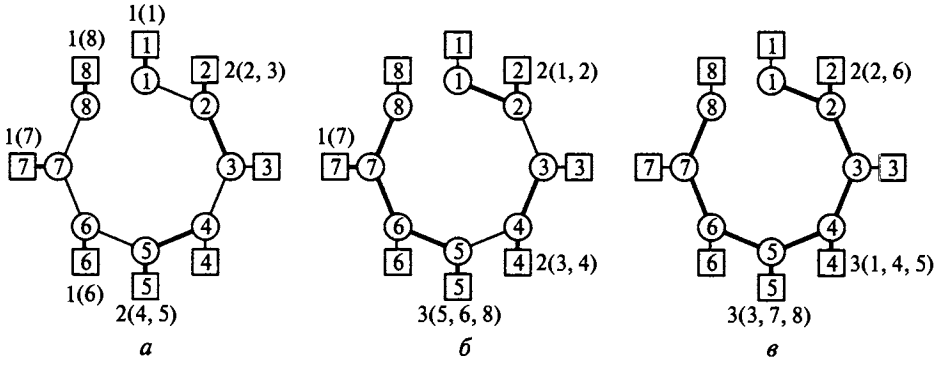


Рис. 12.4. Распределение машин по терминалам линейной ВС:

а — $K = 0$; *б* — $K = 2$; *в* — $K = 10$; ○ — ЭМ; □ — терминал; —— — связь, выделенная для обменов между ЭМ или между ЭМ и терминалом

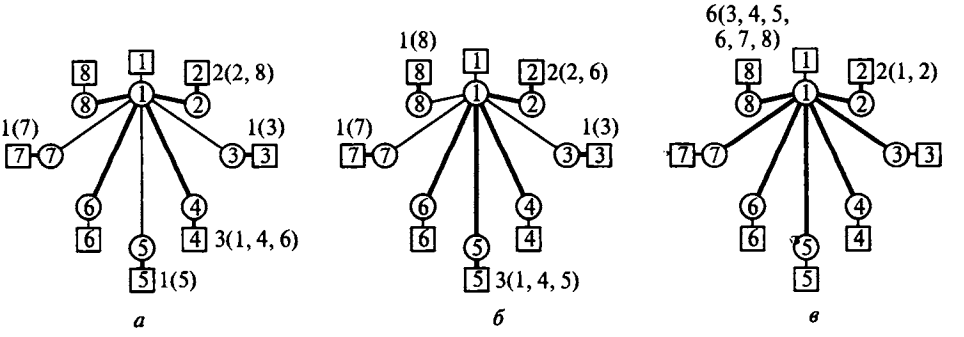


Рис. 12.5. Распределение машин по терминалам радиальной ВС:
а — $K = 0$; *б* — $K = 2$; *в* — $K = 10$; ○ — ЭМ; □ — терминал; —— — связь, выделенная для обменов между ЭМ или между ЭМ и терминалом

Если проанализировать средние спросы на терминалах (см. табл. 12.2) и полученные распределения ЭМ по терминалам, то можно заметить, что имеют место несоответствия, когда количество x_j ЭМ, назначенных на терминал j, меньше минимального спроса ρ_j, $j \in \{1, 2, \ldots, 8\}$ Это объясняется не столько несовершенством метода оптимизации, сколько вероятностной природой задачи об организации функционирования ВС. Действительно, в табл. 12.2 приведены не точные, а средние спросы ρ_j и ρ_j^* на подсистемы для каждого из терминалов j. Вероятности появления на терминале j задач с рангами вне промежутка $[\rho_j, \rho_j^*]$ не равны нулю. Следовательно, даже если для терминала j выделенное количество x_j машин окажется за пределами диапазона $[\rho_j, \rho_j^*]$, то это не будет противоречить реальным процессам поступления задач и функционирования распределенных ВС.

Численный анализ технико-экономических показателей каналов для передачи данных и средств ВТ, данные табл. 12.3 и рис. 12.1—12.5 позволяют сделать следующие выводы.

1. При формировании распределенных вычислительных систем нет необходимости использовать машины с большим количеством полюсов для межмашинных связей; при незначительном количестве ЭМ достаточно ограничиться линейными структурами.

2. Подход, описанный в разд. 12.5.1 и 12.5.2, приводит к стохастически оптимальной организации функционирования ВС, что адекватно характеру поступления задач в систему и их сложности. Оптимизация функционирования ВС с помощью метода цепей Монте-Карло гарантирует получение не минимума, а субминимума целевой функции — эксплуатационных потерь при обслуживании потоков задач, поступающих на терминалы.

3. Увеличение коэффициента штрафа K, регулирующего степень учета потерь из-за недостаточного выделения машин терминалам, приводит к повышению субминимального значения целевой функции Z (12.44) и к уменьшению ΔZ (12.49). Следовательно, с увеличением K ослабевает влияние процедур оптимизации при отыскании окончательного распределения (см. табл. 12.3). Значит, эти процедуры можно сделать менее трудоемкими; например, применительно к методу цепей Монте-Карло можно уменьшить и количество испытаний, и расстояние, на котором следует просматривать распределения относительно исходного.

4. Метод отыскания стохастически оптимального функционирования ВС на основе цепей Монте-Карло эффективно реализуем и на ЭВМ, и на ВС. Для заданной ВС и существующей статистики спросов на вычислительные ресурсы задача организации стохастически оптимального функционирования системы решается один раз. Предложенный метод может стать основой программных блоков операционных систем, обеспечивающих работу ВС в режиме коллективного пользования.

12.5. Анализ технико-экономических возможностей вычислительных систем

1. Введенные показатели с достаточной полнотой позволяют оценить технико-экономическую эффективность функционирования ВС как ансамблей ЭМ. Показатели устанавливают взаимосвязь между надежностью и стоимостью ВС. По ним можно анализировать работу систем в условиях и переходного, и стационарного режимов. Выведенные формулы вполне удовлетворяют требованиям практики, они просты и позволяют производить экспресс-анализ ВС с произвольным количеством ЭМ без применения средств ВТ.

2. Численно установлено, что ВС не более чем за 10 ч входят в стационарный режим работы. Данное обстоятельство приводит к предельной простоте экспресс-анализа технико-экономической эффективности функционирования ВС.

3. Предложен подход к изучению технико-экономической эффективности функционирования ВС, учитывающий топологию сети межмашинных связей и случайный характер спроса на ресурсы. Подход приемлем как при анализе работы многотерминальных ВС, так и при организации стохастически оптимальной работы систем. Он позволяет создавать эффективные операционные системы для ВС коллективного пользования.

4. Микропроцессорные ВС (в том числе и МИКРОС), основанные на принципах параллелизма, программируемости структуры, конструктивной однородности, являются перспективными средствами обработки информации и по технико-экономическим показателям.

ПРИЛОЖЕНИЯ

П.1. Расчет функции надежности вычислительных систем

При выводе расчетных формул для функции $R(t)$ надежности распределенных ВС или, что то же самое, для вероятности безотказной работы ВС в переходном режиме (9.6) будем считать, что задано:

N — число ЭМ, составляющих анализируемую систему;

n — число ЭМ, образующих основную подсистему (вычислительное ядро) (см. § 9.2);

$E_0^N = \{0, 1, 2, ..., n, ..., N\}$ — пространство состояний ВС;

$E_0^{N-n} = \{0, 1, ..., N-n\}$ — подмножество начальных состояний ВС;

m — число (виртуальных) ВУ;

λ — интенсивность потока отказов в одной ЭМ (2.11);

μ — интенсивность восстановления отказавших ЭМ одним ВУ (2.18).

Рассчитаем функцию $R(t)$ надежности ВС, используя технику теории массового обслуживания и методы приближенных вычислений.

Будем полагать, что если система находится в состоянии $j \in E_0^N$, то она имеет j отказавших машин. Тогда $P_j(t)$ будет вероятностью того, что в системе имеется $j \in E_0^N$ отказавших машин в момент времени $t \geqslant 0$.

Функция надежности $R(t)$ является вероятностью того, что на промежутке времени $[0, t)$ ВС, находящаяся в начальный момент времени в состоянии $j \in E_0^{N-n}$, ни разу не войдет ни в одно из состояний подмножества $E^* = \{N-n+1, N-n+2, ..., N\}$. Назовем E^* *множеством поглощающих состояний*, т. е. таких, которые нельзя покинуть после попадания в них [21].

В силу сделанных предположений функционирование ВС описывается марковским процессом с конечным числом состояний. Поэтому ввиду ординарности потоков отказов и восстановлений

$$R(t) = 1 - P_{N-n+1}(t), \qquad (П.1.1)$$

где $P_{N-n+1}(t)$ можно рассматривать как вероятность того, что за время t система войдет в состояние $(N-n+1)$.

Учитывая, что E^* — множество поглощающих состояний, легко составить систему дифференциальных уравнений для $P_j(t)$, $j \in E_0^N$, которая описывает функционирование ВС:

$$\left.\begin{aligned}
&P_0'(t) = -N\lambda P_0(t) + \mu P_1(t);\\
&P_h'(t) = \begin{cases}(N-h+1)\lambda P_{h-1}(t) - [(N-h)\lambda + h\mu]P_h(t) + (h+1)\mu P_{h+1}(t), h < m;\\ (N-h+1)\lambda P_{h-1}(t) - [(N-h)\lambda + m\mu]P_h(t) + m\mu P_{h+1}(t), \quad h \geqslant m;\\ h = 1, 2, ..., N-n-1;\end{cases}\\
&P_{N-n}'(t) = \begin{cases}(n+1)\lambda P_{N-n-1}(t) - [n\lambda + (N-n)\mu]P_{N-n}(t), \quad (N-n) < m;\\ (n+1)\lambda P_{N-n-1}(t) - [n\lambda + m\mu]P_{N-n}(t), \quad (N-n) \geqslant m;\end{cases}\\
&P_{N-n+1}'(t) = n\lambda P_{N-n}(t).
\end{aligned}\right\} \quad (\text{П}.1.2)$$

Задавая начальные условия

$$P_j(0) = 0, \ \ j \in \{0, 1, ..., i-1, i+1, ..., N-n+1\}, \ \ P_i(0) = 1, \ \ i \in \{0, 1, ..., N-n\},$$

решим (П.1.2) относительно $P_{N-n+1}(t)$, (П.1.1). Для этого умножим каждое уравнение (П.1.2) на e^{-st} и проинтегрируем его по t от 0 до ∞. Применяя преобразование Лапласа

$$a_h(s) = \int_0^\infty e^{-st} P_h(t)dt, \quad h = 0, 1, ..., N-n+1,$$

получим алгебраическую систему уравнений:

$$\left.\begin{aligned}
&\delta_0^i = (N\lambda + s)a_0(s) - \mu a_1(s);\\
&\delta_h^i = \begin{cases}-(N-h+1)\lambda a_{h-1}(s) + [(N-h)\lambda + h\mu + s]a_h(s) - (h+1)\mu a_{h+1}(s), h < m;\\ -(N-h+1)\lambda a_{h-1}(s) + [(N-h)\lambda + m\mu + s]a_h(s) - m\mu a_{h+1}(s), \quad h \geqslant m;\\ h = 1, 2, ..., N-n-1;\end{cases}\\
&\delta_{N-n}^i = \begin{cases}-(n+1)\lambda a_{N-n+1}(s) + [n\lambda + (N-n)\mu + s]a_{N-n}(s), \quad (N-n) < m;\\ -(n+1)\lambda a_{N-n+1}(s) + [n\lambda + m\mu + s]a_{N-n}(s), \quad (N-n) \geqslant m;\end{cases}\\
&0 = -n\lambda a_{N-n}(s) + s a_{N-n+1}(s).
\end{aligned}\right\} \quad (\text{П}.1.3)$$

Здесь $\delta_h^i = 1$ при $h = i$ и $\delta_h^i = 0$ при $h \neq i$.

Решая систему (П.1.3) по правилу Крамера, найдем

$$a_{N-n+1}(s) = \frac{n\lambda(n+1)\lambda \cdot ... \cdot (N-i)\lambda \Delta_i(s)}{s\Delta_{N-n+1}(s)} = \frac{(N-i)!\lambda^{N-i-}}{(n-1)!} \frac{\Delta_i(s)}{s\Delta_{N-n+1}(s)}, \quad (\text{П}.1.4)$$

где $\Delta_i(s)$ — определитель, образованный первыми i-ми строками и первыми i-ми столбцами определителя $\Delta_{N-n+1}(s)$; $s\Delta_{N-n+1}(s)$ — определитель системы (П.1.3).

Разлагая определители $\Delta_i(s)$ и $\Delta_{N-n+1}(s)$ по последнему столбцу, получим рекуррентные соотношения:

$$\left.\begin{aligned}\Delta_{h+1}(s) &= [s+(N-h)\lambda+h\mu]\Delta_h(s)-(N-h+1)\lambda h\mu\Delta_{h-1}(s), \quad h<m;\\ \Delta_{h+1}(s) &= [s+(N-h)\lambda+m\mu]\Delta_h(s)-(N-h+1)\lambda m\mu\Delta_{h-1}(s), \quad h\geqslant m;\end{aligned}\right\} \qquad (\text{П.1.5})$$

$h=0,1,...,i-1$ для $\Delta_i(s)$; $h=0,1,...,N-n$ для $\Delta_{N-n+1}(s)$; $\Delta_{-1}(s)=0$; $\Delta_0(s)=1$; $\Delta_1(s)=s+N\lambda$.

После обращения преобразования Лапласа с учетом (П.1.4) имеем

$$P_{N-n+1}(t)=\frac{(N-i)!\lambda^{N-n+1-i}}{(n-1)!2\pi\sqrt{-1}}\int_C\frac{\Delta_i(s)e^{st}}{s\Delta_{N-n+1}(s)}ds, \qquad (\text{П.1.6})$$

где C — контур, охватывающий все нули знаменателя, $\pi=3{,}14...$

Используя методы вычислительной математики, легко найти корни $\Delta_{N-n+1}(s)$, так как система многочленов $\Delta_h(s), h\in\{1,2,...,N-n+1\}$, удовлетворяющая соотношениям (П.1.5), обладает следующими свойствами (см. [24]):

1) все корни $\Delta_h(s)$ различны и отрицательны;

2) корни соседних многочленов $\Delta_{h-1}(s)$ и $\Delta_h(s)$ чередуются, т. е. между каждой парой корней полинома $\Delta_{h-1}(s)$ (в том числе между нулем и наименьшим по модулю корнем) лежит один корень многочлена $\Delta_h(s)$;

3) сумма корней многочлена $\Delta_h(s)$ равна

$$-B_h=\frac{2N-h+1}{2}h\lambda-\frac{h(h-1)}{2}\mu, \qquad h\leqslant m,$$

$$-B_h=\frac{2N-h+1}{2}h\lambda-\frac{2h-m-1}{2}m\mu, \quad h>m.$$

Эти свойства позволяют вычислить корни многочлена $\Delta_{N-n+1}(s)$, например, с помощью метода половинного деления.

Итак, пусть $-\alpha_1,-\alpha_2,...,-\alpha_l,...,-\alpha_{N-n+1}$ — корни полинома $\Delta_{N-n+1}(s)$, тогда рациональную дробь $\dfrac{\Delta_i(s)}{s\Delta_{N-n+1}(s)}$ можно разложить на простейшие дроби:

$$\frac{\Delta_i(s)}{s\Delta_{N-n+1}(s)}=\frac{A_{i0}}{s}+\sum_{l=1}^{N-n+1}\frac{A_{il}}{s+\alpha_l}. \qquad (\text{П.1.7})$$

Здесь

$$A_{il}=\frac{\Delta_i(-\alpha_l)}{[s\Delta_{N-n+1}(s)]'_{s=-\alpha_l}}=\frac{\Delta_i(-\alpha_l)}{\Delta_{N-n+1}(-\alpha_l)-\alpha_l\Delta'_{N-n+1}(-\alpha_l)}.$$

Очевидно, что

$$\left.\begin{aligned} A_{i0} &= \Delta_i(0)/\Delta_{N-n+1}(0), \\ A_{il} &= \frac{\Delta_i(-\alpha_l)}{-\alpha_l \Delta'_{N-n+1}(-\alpha_l)}, \quad l = 1, 2, ..., N-n+1. \end{aligned}\right\} \qquad (П.1.8)$$

Учитывая (П.1.6)–(П.1.8), получаем

$$P_{N-n+1}(t) = \frac{(N-i)!\lambda^{N-n+1-i}}{(n-1)!}\left[\frac{\Delta_i(0)}{\Delta_{N-n+1}(0)} + \sum_{l=1}^{N-n+1} \frac{\Delta_i(-\alpha_l)e^{-\alpha_l t}}{(-\alpha_l)\Delta'_{N-n+1}(-\alpha_l)}\right]. \qquad (П.1.9)$$

Методом математической индукции, используя (П.1.5), легко показать, что

$$\Delta_h(0) = \frac{N!}{(N-h)!}\lambda^h, \qquad (П.1.10)$$

где $\Delta_h(0)$ — определитель, образованный первыми h строками и первыми h столбцами определителя $\Delta_{N-n+1}(0)$. Действительно, если $h < m$, то при $h = 1$ определитель $\Delta_1(0) = N\lambda$, а при $h = 2$ и $h = 3$ соответствующие определители равны

$$\Delta_2(0) = [(N-1)\lambda + \mu]N\lambda - N\lambda\mu = \frac{N!}{(N-2)!}\lambda^2;$$

$$\Delta_3(0) = [(N-2)\lambda + 2\mu]\frac{N!}{(N-2)!}\lambda^2 - (N-1)\lambda 2\mu N\lambda = \frac{N!}{(N-3)!}\lambda^3$$

Допустим, что (П.1.10) справедливо для $h = r$; докажем справедливость (П.1.10) для $h = r+1$. Действительно,

$$\Delta_{r+1}(0) = [(N-r)\lambda + r\mu]\frac{N!\lambda^r}{(N-r)!} -$$

$$-(N-r+1)\lambda r\mu\frac{N!\lambda^{r-1}}{(N-r+1)!} = \frac{N!}{[N-(r+1)]!}\lambda^{r+1},$$

что и требовалось доказать. Аналогично доказывается справедливость (П.1.10) и при $h \geqslant m$.

Подставляя значения $\Delta_i(0)$, $\Delta_{N-n+1}(0)$, вычисляемые по (П.1.10), в (П.1.9) и учитывая формулу (П.1.1), находим

$$R(t) = \frac{(N-i)!\lambda^{N-n-i+1}}{(n-1)!}\sum_{l=1}^{N-n+1}\frac{\Delta_i(-\alpha_l)e^{-\alpha_l t}}{\alpha_l \Delta'_{N-n+1}(-\alpha_l)}.$$

Выведенные формулы для $R(t)$ позволяют осуществлять численный анализ надежности распределенных вычислительных систем.

П.2. Экспресс-анализ функционирования вычислительных систем

Заданные параметры вычислительной системы

N — количество ЭМ, составляющих ВС;

λ — интенсивность отказов ЭМ — среднее количество отказов, происходящих в одной ЭМ в течение 1 ч (λ^{-1}, ч — средняя наработка до отказа машины);

m — число (виртуальных) восстанавливающих устройств (ВУ); каждое ВУ способно ремонтировать в любой момент времени $t \geqslant 0$ только одну ЭМ;

μ — интенсивность восстановления ЭМ — среднее количество восстановлений машин, которое может произвести одно ВУ за 1 ч (μ^{-1}, ч — среднее время восстановления машины одним ВУ);

β — интенсивность решения задачи на одной ЭМ ВС (β^{-1}, ч — среднее время решения задачи на одной машине);

c_1, руб./ч — стоимость часа полезного времени работы ЭМ или арендная плата за 1 ч эксплуатации одной ЭМ;

c_2' и c_2, руб./ч — соответственно себестоимость и стоимость содержания восстанавливающего устройства в течение 1 ч;

c_3, руб. — средняя стоимость технических средств (интегральных схем, типовых элементов замены и т. п.), расходуемых при однократном восстановлении отказавшей ЭМ;

i — начальное состояние ВС или число работоспособных ЭМ в момент начала функционирования системы, т. е. при $t = 0$; $\{0, 1, \ldots, N\}$ — множество состояний ВС.

Рассчитываемые показатели эффективности вычислительной системы

1. Математическое ожидание числа работоспособных ЭМ в системе

1.1. Стационарный режим; η — среднее количество работоспособных ЭМ в системе при длительной ее эксплуатации. Расчет выполнить по формуле

$$\eta = \begin{cases} N\mu/(\lambda+\mu), & \text{если } N\lambda \leqslant m\mu; \\ m\mu/\lambda & \text{в противном случае.} \end{cases}$$

1.2. Переходный режим; $\eta(i, t)$ — среднее число работоспособных ЭМ в момент времени $t \geq 0$ в системе, начавшей функционировать в состоянии i, $0 \leq i \leq N$. Расчет осуществлять по формуле

$$\eta(i, t) = \begin{cases} \dfrac{N\mu}{\lambda+\mu} + \dfrac{i\lambda-(N-i)\mu}{\lambda+\mu} e^{-(\lambda+\mu)t}, & (N-m) \leqslant i \leqslant N, \text{ если } N\lambda \leqslant m\mu; \\ \dfrac{m\mu}{\lambda} + \dfrac{i\lambda-m\mu}{\lambda} e^{-\lambda t}, & 0 \leqslant i < (N-m), \text{ если } N\lambda > m\mu. \end{cases}$$

Информация для эксплуатационников и пользователей ВС

1. Расчет $\mathcal{N}(i,\ t)$ следует производить только при необходимости тонкого анализа производительности ВС, при оценке времени вхождения в стационарный режим. В условиях коммерческой эксплуатации режим работы ВС стационарен, поэтому достаточно ограничиться расчетом только $\mathcal{N}$.

2. Количество (виртуальных) ВУ и их параметры должны быть подобраны так, чтобы выполнялось неравенство $N\lambda \leqslant m\mu$. Тогда для одного и того же промежутка времени среднее количество отказов, появляющихся в ВС, не будет превышать среднего количества восстановлений, которые могут выполнить все ВУ. Тем самым будет достигнута гармоничность работы собственно ВС и совокупности ВУ как единого целого.

3. При определении среднего количества машин, которые могут быть использованы для решения задач, достаточно ограничиться расчетом по формуле

$$\mathcal{N} = N\mu/(\lambda+\mu).$$

Следовательно, параллельные программы должны разрабатываться с учетом того, что число ветвей в них не должно превышать $\mathcal{N}$.

4. Для оценки средней производительности ВС достаточно воспользоваться формулой

$$\Omega = \omega N\mu/(\lambda+\mu),$$

где ω — показатель производительности одной ЭМ (например, номинальное быстродействие машины, выраженное в операциях в секунду).

2. Функция осуществимости решения задачи на живучей ВС

2.1. Стационарный режим; $\mathcal{F}(t)$ — вероятность того, что на ВС, находящейся в длительной эксплуатации, будет решена за время $t \geqslant 0$ задача, представленная адаптирующейся параллельной программой, т. е. программой, использующей в любой момент времени все работоспособные ЭМ. Расчет производить по формуле

$$\mathcal{F}(t) = 1 - \exp\begin{cases} -\beta N\mu t/(\lambda+\mu), & \text{если } N\lambda \leqslant m\mu; \\ -\beta m\mu t/\lambda & \text{в противном случае.} \end{cases}$$

2.2. Переходный режим; $\mathcal{F}(i,\ t)$ — вероятность того, что на ВС, начавшей функционировать в состоянии $i, 0 \leqslant i \leqslant N$, будет за время $t \geqslant 0$ решена задача, представленная в виде адаптирующейся параллельной программы. Расчет выполнять по формуле

$$\mathcal{F}(i,\ t) = 1 - \exp\begin{cases} -\beta\left\{\dfrac{N\mu}{\lambda+\mu}t + \dfrac{i\lambda-(N-i)\mu}{(\lambda+\mu)^2}\left[1-e^{-(\lambda+\mu)t}\right]\right\}, \\ \qquad (N-m) \leqslant i \leqslant N, \text{ если } N\lambda \leqslant m\mu; \\ -\beta\left[\dfrac{m\mu}{\lambda}t + \dfrac{i\lambda-m\mu}{\lambda^2}(1-e^{-\lambda t})\right], \\ \qquad 0 \leqslant i < (N-m), \text{ если } N\lambda > m\mu. \end{cases}$$

Информация для проектировщиков и пользователей ВС

1. Расчет функции $\mathcal{F}(i, t)$ необходимо выполнять при проектировании ВС специального назначения, например систем управления подвижными объектами. При коммерческой эксплуатации ВС общего назначения такой расчет нужен в редких случаях.

2. Режим работы ВС общего назначения, как правило, стационарен. Кроме того, должны обеспечиваться условия, при которых выполняется неравенство $N\lambda \leqslant m\mu$. Следовательно, пользователям ВС для оценки осуществимости решения их задач достаточно воспользоваться простейшей формулой

$$\mathcal{F}(t) = 1 - \exp\left[-\beta N\mu(\lambda + \mu)^{-1} t\right].$$

3. Математическое ожидание бесполезных расходов при эксплуатации ВС

3.1. Стационарный режим; γ — средние бесполезные расходы за 1 ч при длительной эксплуатации ВС:

$$\gamma = \begin{cases} \dfrac{N\lambda}{\lambda + \mu}(c_1 - c_2) + mc_2, & \text{если } N\lambda \leqslant m\mu; \\ (N\lambda - m\mu)\lambda^{-1}c_1 & \text{в противном случае.} \end{cases}$$

3.2. Переходный режим; $\Gamma(i, t)$ — средние бесполезные эксплуатационные расходы к моменту времени $t \geqslant 0$ при условии, что в начальный момент система находилась в состоянии i, $0 \leqslant i \leqslant N$. Расчет выполнять по формуле

$$\Gamma(i, t) = -\varepsilon_i + \gamma t + \varepsilon_i \delta(t),$$

где

$$\varepsilon_i = \frac{i\lambda - (N - i)\mu}{(\lambda + \mu)^2}(c_1 - c_2);$$

$$\gamma = \frac{N\lambda}{\lambda + \mu}(c_1 - c_2) + mc_2;$$

$$\delta(t) = e^{-(\lambda+\mu)t}, \quad (N - m) \leqslant i \leqslant N$$

при выполнении условия $N\lambda \leqslant m\mu$, или

$$\varepsilon_i = \frac{i\lambda - m\mu}{\lambda^2}c_1; \quad \gamma = \frac{N\lambda - m\mu}{\lambda}c_1;$$

$$\delta(t) = e^{-\lambda t}, \quad 0 \leqslant i < N - m$$

при невыполнении условия $N\lambda \leqslant m\mu$.

Информация для эксплуатационников ВС

Оценку издержек, связанных с эксплуатацией ВС, проводят в коммерческих целях. При этом обеспечивается стационарный режим работы ВС и создаются усло-

вия, при которых выполняется неравенство $N\lambda \leqslant m\mu$. Эксплуатационнику, анализирующему такой технико-экономический показатель, как средние бесполезные издержки, достаточно применять лишь простейшую формулу

$$\gamma = \frac{N\lambda}{\lambda + \mu}(c_1 + c_2) + mc_2.$$

4. Математическое ожидание дохода ВС

4.1. Стационарный режим; g — прибыль вычислительной системы или средний доход, приносимый за 1 ч при длительной эксплуатации ВС:

$$g = \begin{cases} \dfrac{N\mu}{\lambda + \mu}(c_1 - \lambda c_3) - mc_2', & \text{если } N\lambda \leqslant m\mu; \\ \dfrac{m\mu}{\lambda}(c_1 - \lambda c_3) - mc_2', & \text{если } N\lambda > m\mu. \end{cases}$$

4.2. Переходный режим; $D(i,t)$ — средний доход, который приносит ВС за время $t \geqslant 0$, если она начинает функционировать в состоянии i, $0 \leqslant i \leqslant N$:

$$D(i,\ t) = D_i + gt - D_i\delta(t),$$

где

$$D_i = \frac{i\lambda - (N-i)\mu}{(\lambda + \mu)^2}(c_1 + \mu c_3);$$

$$g = \frac{N\mu}{\lambda + \mu}(c_1 - \lambda c_3) - mc_2;$$

$$\delta(t) = e^{-(\lambda+\mu)t}, \quad (N-m) \leqslant i \leqslant N$$

при выполнении условия $N\lambda \leqslant m\mu$, или

$$D_i = \frac{i\lambda - m\mu}{\lambda^2}c_1; \quad g = \frac{m\mu}{\lambda}(c_1 - \lambda c_3) - mc_2';$$

$$\delta(t) = e^{-\lambda t}, \ 0 \leqslant i \leqslant (N-m)$$

при невыполнении условия $N\lambda \leqslant m\mu$.

Информация для эксплуатационников ВС

Даже при самом широком толковании понятия «доход ВС» значения этого показателя представляют интерес в основном при коммерческой эксплуатации. При этом характерным является стационарный режим эксплуатации ВС. Работа ВС должна быть организована так, чтобы обеспечивалось выполнение неравенства $N\lambda \leqslant m\mu$. В этих условиях следует использовать формулу

$$g = \frac{N\mu}{\lambda + \mu}(c_1 - \lambda c_3) - mc_2,$$

которая позволяет рассчитывать прибыль, приносимую при работе ВС.

СПИСОК ЛИТЕРАТУРЫ

1. *Нейман Дж. фон.* Теория самовоспроизводящихся автоматов: Пер. с англ./ Под ред. В.И. Варшавского. М.: Мир, 1971.
2. *Лебедев С.А., Дашевский Л.Н., Шкабара Е.А.* Малая электронная счетная машина. М.: АН СССР, 1952.
3. *Лебедев Сергей Алексеевич.* К 100-летию со дня рождения основоположника отечественной электронной вычислительной техники. М.: Физматлит, 2002.
4. *Малиновский Б.Н.* История вычислительной техники в лицах. Киев: Фирма «КИТ», ПТОО «А.С.К.», 1995.
5. *Евреинов Э.В., Хорошевский В.Г.* Однородные вычислительные системы. Новосибирск: Наука, 1978.
6. *Хорошевский В.Г.* Инженерный анализ функционирования вычислительных машин и систем. М.: Радио и связь, 1987.
7. *Смирнов А.Д.* Архитектура вычислительных систем. М.: Наука, 1990.
8. *Ховард Р.А.* Динамическое программирование и марковские процессы: Пер. с англ. М.: Сов. радио, 1964.
9. *Диткин В.А., Прудников А.П.* Справочник по операционному исчислению. М.: Высшая школа, 1965.
10. *Ортега Дж.* Введение в параллельные и векторные методы решения линейных систем. М.: Мир, 1991.
11. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. С.-Петербург: БХВ-Петербург, 2002.
12. *Корнеев В.Д.* Параллельное программирование в MPI. Новосибирск: СО РАН, 2000.
13. *Головкин Б.А.* Параллельные вычислительные системы. М.: Наука, 1980.
14. *Евреинов Э.В., Косарев Ю.Г.* О возможности построения вычислительных систем высокой производительности. Новосибирск: Изд-во СО АН СССР, 1962.
15. *Каляев А.В.* Многопроцессорные системы с программируемой архитектурой. М.: Радио и связь, 1984.
16. *Каляев А.В., Левин И.И.* Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. М.: Янус-К, 2003.
17. *Евреинов Э.В., Косарев Ю.Г.* Однородные универсальные вычислительные системы высокой производительности. Новосибирск: Наука, 1966.
18. *Корнеев В.В.* Архитектура вычислительных систем с программируемой структурой. Новосибирск: Наука, 1985.
19. *Краснов С.А.* Транспьютеры, транспьютерные вычислительные системы и Оккам. В кн. «Вычислительные процессы и системы» / Под ред. Г.И. Марчука. Вып. 7. М.: Наука, 1990.
20. *Транспьютеры.* Архитектура и программное обеспечение/ Под редакцией Г. Харпа. М.: Радио и связь, 1993.
21. *Хинчин А.Я.* Работы по математической теории массового обслуживания. М.: ГИФМЛ, 1963.
22. *Флейшман Б.С.* Статистические пределы эффективности сложных систем // Прикладные задачи технической кибернетики. М.: Сов. радио, 1966.
23. *Пирс У.* Построение надежных вычислительных машин: Пер. с англ. М.: Мир, 1968.
24. *Гнеденко Б.В., Беляев Ю.К., Соловьев А.Д.* Математические методы в теории надежности. М.: Наука, 1965.
25. *Вентцель Е.С.* Исследование операций. М.: Сов. радио, 1972.
26. *Саати Т.Л.* Элементы теории массового обслуживания и ее приложения. М.: Сов. радио, 1971.
27. *Хедли Дж.* Нелинейное и динамическое программирование: Пер. с англ. М.: Мир, 1967.

Учебное издание

Информатика в техническом университете

Хорошевский Виктор Гаврилович

АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Редактор *Н.Е. Овчеренко*
Художник *Н.Г Столярова*
Компьютерная графика *О.В. Левашовой*
Корректор *Г.С. Беляева*
Компьютерная верстка *С.Ч. Соколовского*

Оригинал-макет подготовлен в Издательстве МГТУ им. Н.Э. Баумана

Подписано в печать 09.04.08. Формат 70×100/16.
Печать офсетная. Бумага офсетная. Гарнитура «Таймс».
Усл. печ. л. 42,9. Уч.-изд. л. 39,87.
Тираж 1500 экз. Заказ № 685

Издательство МГТУ им. Н.Э. Баумана
105005, Москва, 2-я Бауманская, 5

Отпечатано с готовых диапозитивов в ГУП ППП «Типография «Наука»
121099, Москва, Шубинский пер., 6