7.2. АРХИТЕКТУРНЫЕ ОСОБЕННОСТИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С ПРОГРАММИРУЕМОЙ СТРУКТУРОЙ

В архитектурном плане вычислительные системы с программируемой структурой – это аппаратурно-программные объекты, основанные на модели коллектива вычислителей. Первоначальное представление о функциональной организации ВС читатель уже получил (см. гл. 3 и 7.1). В этом параграфе будет дана детализация структуры ВС, точнее будут изложены элементы теории структур. Затем будут описаны режимы функционирования ВС и способы обработки информации в них и, наконец, будут отмечены аспекты, связанные с организацией операционных систем. Материал этого параграфа вместе с содержанием гл. 3 позволит достаточно полно судить об архитектуре ВС с программируемой структурой.

7.2.1. Структура ВС

Как известно (см. 3.1.2.), (макро)структура ВС описывается графом G, множеству вершин которого сопоставлены элементарные машины (или системные устройства, или локальные коммутаторы), а множеству рёбер – 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 учётом ненадёжности компонентов ВС). Здесь внимательнее рассмотрим структурные аспекты, прежде всего связанные с производительностью и живучестью ВС.

I. Требования, предъявляемые к структуре ВС

Остановимся на требованиях, которые предъявляются к структурам современных вычислительных систем.

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. 3.3.5) не должны быть связаны со значительными накладными расходами (например, с большим временем работы операционной системы по вложению параллельного алгоритма).

Удобство адресации элементарных машин и “переноса” подсистем в пределах вычислительной системы

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). Итак, при выполнении данных требований будет достигнута эффективность ВС как при работе в моно-, так и мультипрограммных режимах. Кроме того следует отметить, что данные требования – это необходимые условия для создания отказоустойчивых параллельных программ.

Осуществимость принципа близкодействия и минимума задержек при межмашинных передачах информации в ВС

Принцип близкодействия предопределяет реализацию обменов информацией между "удалёнными" друг от друга ЭМ через промежуточные машины системы (см. 3.2.1). Следовательно, в условиях ограниченности числа связей у каждой ЭМ структура должна обеспечивать минимум задержек при "транзитных" передачах информации.

Масштабируемость и большемасштабность структуры ВС

Для формирования конфигураций ВС с заданной эффективностью требуется, 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 число 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).

Коммутируемость структуры ВС

Как уже было показано (см. 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 взаимодействий между подмножествами ЭМ.

Живучесть структуры ВС

Важным требованием к ВС в целом является обеспечение работоспособности при отказе её компонентов или даже подсистем. Основой функциональной целостности ВС как коллектива элементарных машин является живучесть структуры. Под последним понимается способность структуры ВС обеспечить связность требуемого числа работоспособных ЭМ в системе при ненадёжных линиях межмашинных связей.

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 вычислительных систем. Заметим, что аналогичный вывод сделан в гл. 3 на основе опыта применения методики крупноблочного распараллеливания трудоёмких задач.

II. Структурные характеристики ВС

Структура вычислительной системы – это 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 вычислительных системах используются диаметр d и средний диаметр $\overline{d}$ структуры. *Диаметр* есть максимальное расстояние, определённое на множестве кратчайших путей между парами вершин структуры ВС:

$$d = \max_{i,j}\{d_{ij}\}, \tag{7.1}$$

а *средний диаметр* –

$$\overline{d} = (N-1)^{-1}\sum_{l=1}^{d} l \cdot n_l, \tag{7.2}$$

где d_{ij} – расстояние, т.е. минимальное число рёбер, образующих путь из вершины i в вершину j; $i, j \in \{0,1,...,N-1\}$; n_l – число вершин, находящихся на расстоянии l от любой выделенной вершины (однородного) графа G.

Показателем оценивающим *структурную коммутируемость* ВС, является вектор-функция

$$\boldsymbol{K}(G,s,s') = \{K_h(G,s,s')\}, \quad h \in \{1,2,...,[N/2]\}, \tag{7.3}$$

в которой координата $K_h(G,s,s')$ есть вероятность реализации в системе при заданных структуре G и коэффициентах готовности s и s' соответственно одной ЭМ и линии связи (см. 2.8.4) h одновременных межмашинных взаимодействий (обменов информацией) между непересекающимися подмножествами ЭМ; $[N/2]$ – целая часть числа $N/2$.

Структурная живучесть ВС оценивается вектор-функцией

$$\mathrm{L}(G,s,s') = \{\mathrm{L}_r(G,s,s')\}, \quad r \in E_2^N = \{2,3,...,N\}, \tag{7.4}$$

$\{L_r(G,s,s')\}$ является вероятностью 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' элементарной машины и линии связи соответственно.

Введённые показатели позволяют осуществить с достаточной полнотой анализ структурных возможностей ВС, анализ структурной живучести ВС, в частности. Отметим прикладное значение показателей (7.1) – (7.4).

Прежде всего подчеркнём, что диаметр (7.1) и средний диаметр (7.2) – это структурные характеристики, связанные с производительностью вычислительной системы. 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 может выступать в качестве показателя, оценивающего средние задержки при выполнении межмашинных взаимодействий.

Координаты *вектор-функции структурной коммутируемости* $\mathrm{K}(G,s,s')$ дают характеристику ВС относительно её возможностей по реализации обменов информацией между её 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. 3.3.5). Тогда от структуры ВС требуется, чтобы:

1) при дифференцированном обмене имелась возможность реализации одного взаимодействия между 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) даёт полную характеристику структуры ВС по реализации межмашинных взаимодействий при работе систем в монопрограммном режиме (когда все ЭМ используются для решения одной задачи).

В мультипрограммных режимах функционирования вычислительная система программным способом разбивается на подсистемы. Максимальное число подсистем в вычислительной системе определяется, очевидно, величиной $[N/2]$. В случае мультипрограммирования обмены совершаются лишь в пределах подсистем. Тогда, например, координата $K_h(G,s,s')$ вектор-функции (7.3) будет информировать о приспособленности структуры к генерации в пределах ВС h подсистем или к одновременному решению на ВС h задач, $1 \le h \le [N/2]$.

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, число которых может быть произвольным, в частности, превышающим $[N/2]$. В этой же ситуации вводят и виртуальные линии межмашинных связей ВС. Применительно к такой мультипрограммной работе ВС координата $K_h(G,s,s')$ 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 рёбер графа G) не должна обслуживать более чем l взаимодействий (разделяться более чем на l виртуальных линий); $l \in \{1, 2,..., h\}$; $h \in \{1, 2,..., [N/2]\}$.

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 $L_r(G,s,s')$ 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 $r \in \{2,3,..., N\}$. Далее, подмножество координат

$$\{L_{r^o}(G,s,s'), L_{r^o+1}(G,s,s'),..., L_{r^*}(G,s,s')\}$$

вектор-функции (7.4) характеризует уже приспособленность ВС по выполнению на ней *адаптирующихся параллельных программ*, допускающих автоматическое изменение своих рангов от r^o до r^*, где $1 < r^o$, $r^* \le N$. При $r^o = n$ и $r^* = N$ вектор-функция (7.4) будет характеризовать способность ВС с программируемой структурой к организации в ней виртуальных образований, называемых живучими ВС (см. [3,5], а также главу 10 данной книги).

III. Перспективные структуры ВС

В главе 3 (см. 3.1.2) мы уже сталкивались со структурами, перспективными для формирования вычислительных систем. Главы 4-6 дают хорошее представление о

структурах промышленных ВС с массовым параллелизмом. Здесь мы 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 вычислительных систем, известные сейчас как циркулянтные (circulant structures). Впервые они были определены и исследованы в Отделе вычислительных систем ИМ СО АН СССР в начале 70-х годов и первоначально назывались *D_n-графами* [3]. По определению *D_n-граф* или *циркулянтная структура* есть граф G вида: $\{N; \omega_1, \omega_2, ..., \omega_n\}$, в котором:

- N – число вершин или порядок графа;
- вершины помечены целыми числами i по модулю N, следовательно, $i \in \{0,1,..., N-1\}$;
- вершина i соединена ребром (или является смежной) с вершинами $i \pm \omega_1, i \pm \omega_2, ..., i \pm \omega_n (\text{mod } N)$;
- $\{\omega_1, \omega_2, ..., \omega_n\}$ – целые числа, $0 < \omega_1 < \omega_2 < ... < \omega_n < (N+1)/2,$ называемые образующими (для чисел $N; \omega_1, \omega_2, ..., \omega_n$ наибольший общий делитель есть 1);
- n – размерность графа;
- $2n$ – степень вершины в графе.

В качестве примера рассмотрим D_2-граф или двумерный циркулянт вида: $\{12;3,4\}$ см. рис. 7.1.

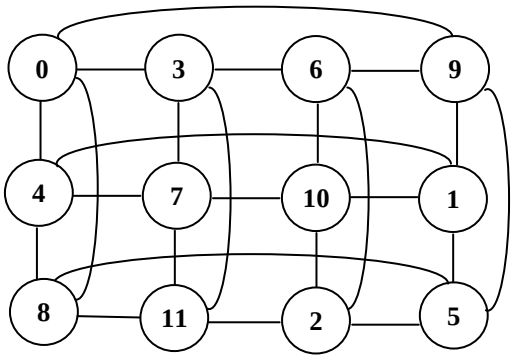
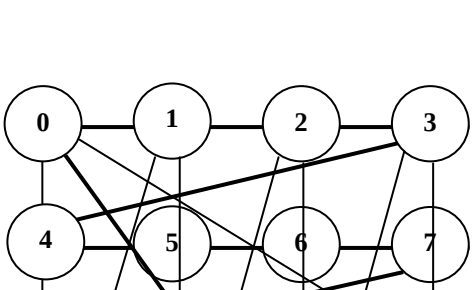


Рис. 7.1. D_2-граф: $\{12;3,4\}$

Графы G вида $\{N;1,\omega_2,...,\omega_n\}$, т.е. D_2-графы или циркулянты с единичной образующей (loop networks) интенсивно изучаются в последнее время. Циркулянтные структуры $\{N;1,\omega_2\}$ широко внедрены в практику вычислительных систем, и читатели уже имели с ними дело при изучении матричных ВС (см. гл.5). Так, например, на рис. 5.3 изображён циркулянт $\{64;1,8\}$ отражающий структуру квадранта вычислительной системы ILLIAC-IV.

Структуры, отражённые на рис 5.3 и 7.1, представлены в виде двумерных матриц. На практике часто используется изображение структур как хордовых колец (см. рис. 7.2).



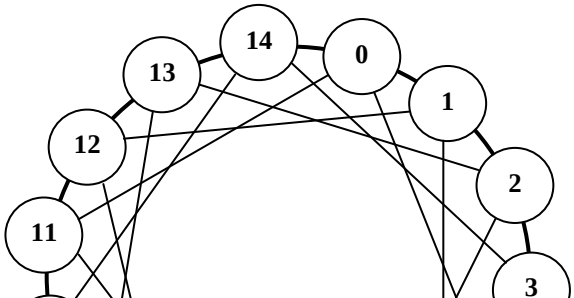


Рис. 7.2. D_2-граф: $\{15;1,4\}$: а) – двумерная матрица, б) – хордовое кольцо

Целые числа $i \in \{0,1,2,..., N-1\}$, отмечающие вершины D_n -графа, называют *адресами. Адресация* вершин в таких структурах *называется диофантовой* (в честь древнегреческого математика из Александрии Диофанта, Diophantos, 3 век). В циркулянтных структурах при полном переносе какой-либо подструктуры (всех вершин подструктуры на одно и то же расстояние в одном из направлений) сохраняются все её свойства и адресация вершин. Следовательно, при диофантовой адресации элементарных машин ВС можно простыми средствами реконфигурации осуществить виртуальную адресацию вершин-машин и, следовательно,

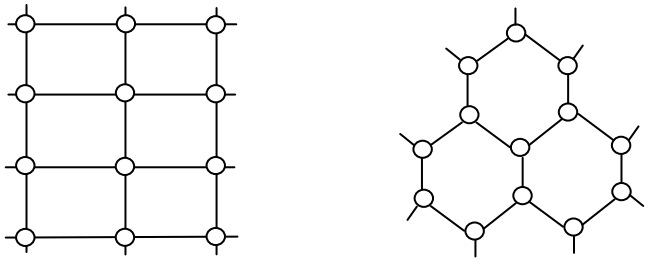
1) создавать отказоустойчивые параллельные программы,
2) реализовывать мультипрограммные режимы обработки информации,
3) исключать отказавшие вершины-машины из подсистем, а значит обеспечить живучесть ВС.

При этом алгоритм работы реконфигуратора структуры ВС сводится к изменению адресов α у всех машин подсистемы по формуле:

$$\alpha := [\alpha + (j - i)] \bmod N, \quad \alpha \in \{0,1,..., N-1\},$$

где i – номер ЭМ, исключаемой из подсистемы, а j – номер машины, включаемый в подсистему, $i, j \in \{0,1,..., N-1\}$.

В качестве структур, допускающих изменение числа машин в ВС без коренной перекоммутации уже имеющихся межмашинных связей, используются $Л(N, v, g)$*-графы* (введённые также в Отделе вычислительных систем ИМ СО АН СССР). В такие графы 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 $v \geq 3$ входит в не менее v кратчайших простых циклов длиной g (длина кратчайшего цикл в графе называется *обхватом*). При $v = 2$ $Л(N, v, g)$-граф является простым циклом с N вершинами.



а) б)

Рис. 7.3. Фрагменты $L(N,v,g)$-графов:
а) – $v=4, g=4$; б) – $v=3, g=6$

IV. Анализ и синтез структур ВС

Введённые показатели (7.1) – (7.4) в равной степени пригодны и для анализа и для синтеза структур вычислительных систем. Как в том, так и в другом случае требуется осуществлять расчёт 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. Рабочий метод расчёта этих показателей – статистическое моделирование.

Проблема синтеза структур заключается в поиске таких графов G^*, которые бы делали реальные (физические) конфигурации ВС максимально приспособленными для программирования (виртуальных) конфигураций. Если при проектировании ВС преследуется цель её 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 элементарных машин, то отыскивается структура G^*, дающая минимум задержек при транзитных передачах информации между ЭМ.

На определённом этапе работ по вычислительным системам заметную роль при выборе их структур играли интуитивные соображения. Так, например, было очевидно, что кольцевые и тороидальные структуры ВС обладают большей живучестью, чем “линейка” и “решётка” соответственно.

Проблема синтеза структур ВС может быть сформулирована с ориентацией на любой из структурных показателей (7.1) – (7.4). Здесь мы дадим следующую постановку задачи синтеза структур: найти структуру G^*, которая обеспечивала бы максимум координаты вектор-функции структурной живучести (7.4), т.е.

$$\max_{G} L_r(G,s,s') = \mathsf{L}_r(G^*), \tag{7.5}$$

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 $\text{Л}(N,v,g)$-графов.

К сложным относится проблема синтеза оптимальных структур большемасштабных ВС, она практически решается при помощи стохастического моделирования (методом Монте-Карло) и, следовательно, с использованием мощных вычислительных средств. Трудоёмкость поиска G^* можно заметно снизить, если воспользоваться двумя нижеприведенными гипотезами.

Гипотеза 1. Структура G^*, при которой достигается $L_N(G^*)$-максимум живучести ВС, обеспечивает и $L_r(G^*)$-максимум живучести подсистем ранга $r < N$.

Гипотеза 2. Структура с минимальным (средним) диаметром относится к G^*, т.е. обладает максимальной структурной живучестью.

Справедливость гипотез подтверждена результатами статистического моделирования структур ВС. Эти гипотезы были высказаны автором в 70-х годах 20 столетия.

Ниже *оптимальными* будем называть структуры G^*, имеющие при заданных порядке N и степени v вершин минимальный (средний) диаметр. Создание общего алгоритма синтеза оптимальных структур является сложной задачей. Существуют алгоритмы синтеза для конкретных классов графов. Для целей практики созданы и развиваются каталоги оптимальных структур. Пользование каталогами так же просто, как таблицами элементарных функций. Фрагмент каталога оптимальных D_n-графов отражён в табл. 7.1.

Таблица 7.1.

D_n-граф	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	3 90	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

Структуры в виде оптимальных $Л(N,v,g)$-графов показаны на рис. 7.4. В этих структурах достигнуты минимумы диаметра d (7.1) и среднего диаметра $\overline{d}$ (7.2) и, следовательно, минимумы задержек при передаче информации между ЭМ в вычислительных системах.

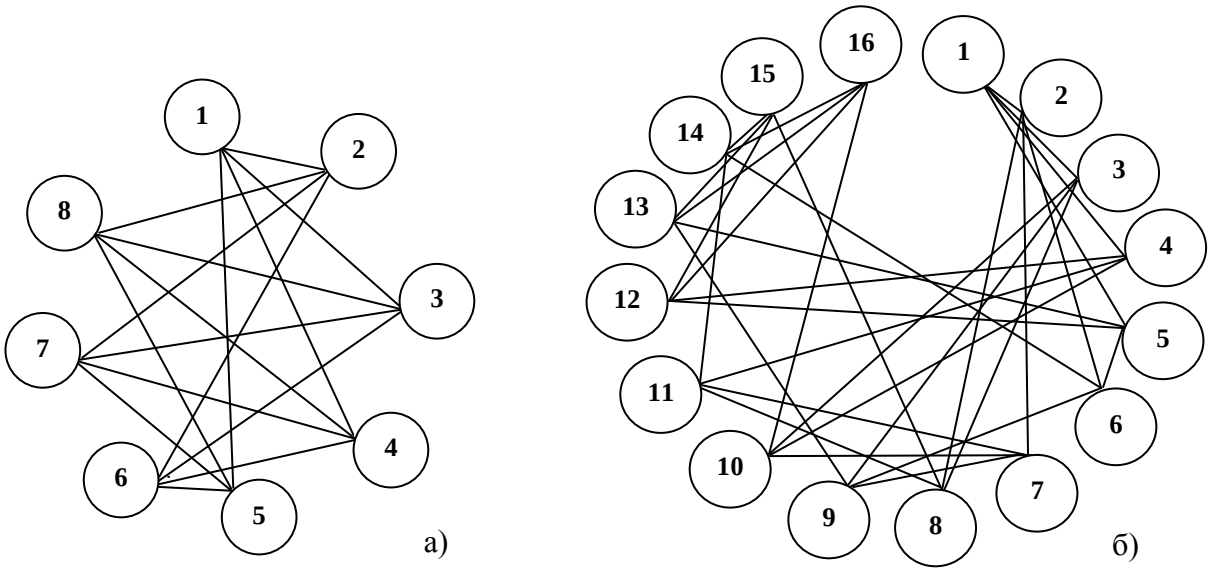


Рис. 7.4. Оптимальные $Л(N,v,g)$-графы:
а) $Л(8,4,4)$-граф, $d=2,\ \overline{d}=1{,}43$; б) $Л(16,4,4,)$-граф, $d=3,\ \overline{d}=1{,}91$

Графы $Л(N,v,g)$ можно описывать в виде матриц смежности. Пусть запись $i.j,k,...,l$ представляет элементы i-той строки матрицы смежности $Л(N,v,g)$-графа, которые равны 1. В качестве примера оптимальных $Л(N,v,g)$-графов ниже приводятся описания матриц смежности для $N=32$:

$Л(32,3,7)$-граф, $d=5,\ \overline{d}=2{,}94$

1. 2,3,4
2. 1,5,6
3. 1,7,8
4. 1,9,10
5. 2,11,12
6. 2,13,14
7. 3,15,16
8. 3,17,18
9. 4,19,20
10. 4,21,22
11. 5,22,29
12. 5,17,23
13. 6,24,25
14. 6,15,27
15. 7,14,30
16. 7,25,32
17. 8,12,26
18. 8,19,29
19. 9,18,31
20. 9,26,27
21. 10,23,28
22. 10,11,30
23. 12,21,24
24. 13,23,31
25. 13,16,26
26. 17,20,25
27. 14,20,28
28. 21, 27,32
29. 11,18,32
30. 15,22,31
31. 19,24,30
32. 16,28,29

$Л(32,4,5)$-граф, $d=4,\ \overline{d}=2{,}38$

1. 2,3,4,5
2. 1,6,7,8
3. 1,9,10,11
4. 1,12,13,14
5. 1,15,16,17
6. 2,17,18,32
7. 2,10,19,20
8. 2,9,12,21
9. 3,8,13,22
10. 3,7,23,24
12. 4,8,11,26
13. 4,9,27,28
14. 4,15,18,29
15. 5,11,14,30
16. 5,18,20,31
17. 5,6,21,22
18. 6,14,16,19
19. 7,18,22,25
20. 7,16,21,26
21. 8,17,20,28
23. 10,22,26,30
24. 10,25,27,31
25. 11,19,24,29
26. 12,20,23,27
27. 13,24,26,32
28. 13,21,29,31
29. 14,25,28,32
30. 15,23,31,32
31. 16,24,28,30
32. 6,27,29,30

11. 3,12,15,25 22. 9,17,19,23

$Л(32,4,6)$ -граф, $d = 4,\quad \overline{d} = 2{,}36$

1. 2,3,4,5
2. 1,6,7,8
3. 1,9,10,11
4. 1,12,13,14
5. 1,15,16,17
6. 2,16,18,22
7. 2,21,26,29
8. 2,19,30,32
9. 3,19,23,25
10. 3,26,27,30
11. 3,20,21,31
12. 4,20,24,29
13. 4,23,27,28
14. 4,21,22,32
15. 5,21,25,27
16. 5,6,28,29
17. 5,18,19,26
18. 6,17,31,32
19. 8,9,17,20
20. 11,12,19,22
21. 7,11,14,15
22. 6,14,20,27
23. 9,13,29,32
24. 12,25,26,31
25. 9,15,24,30
26. 7,10,17,24
27. 10,13,15,22
28. 13,16,30,31
29. 7,12,16,23
30. 8,10,25,28
31. 11,18,24,28
32. 8,14,18,23

Для наглядности на рис. 7.5 представлен граф для последней матрицы смежностей.

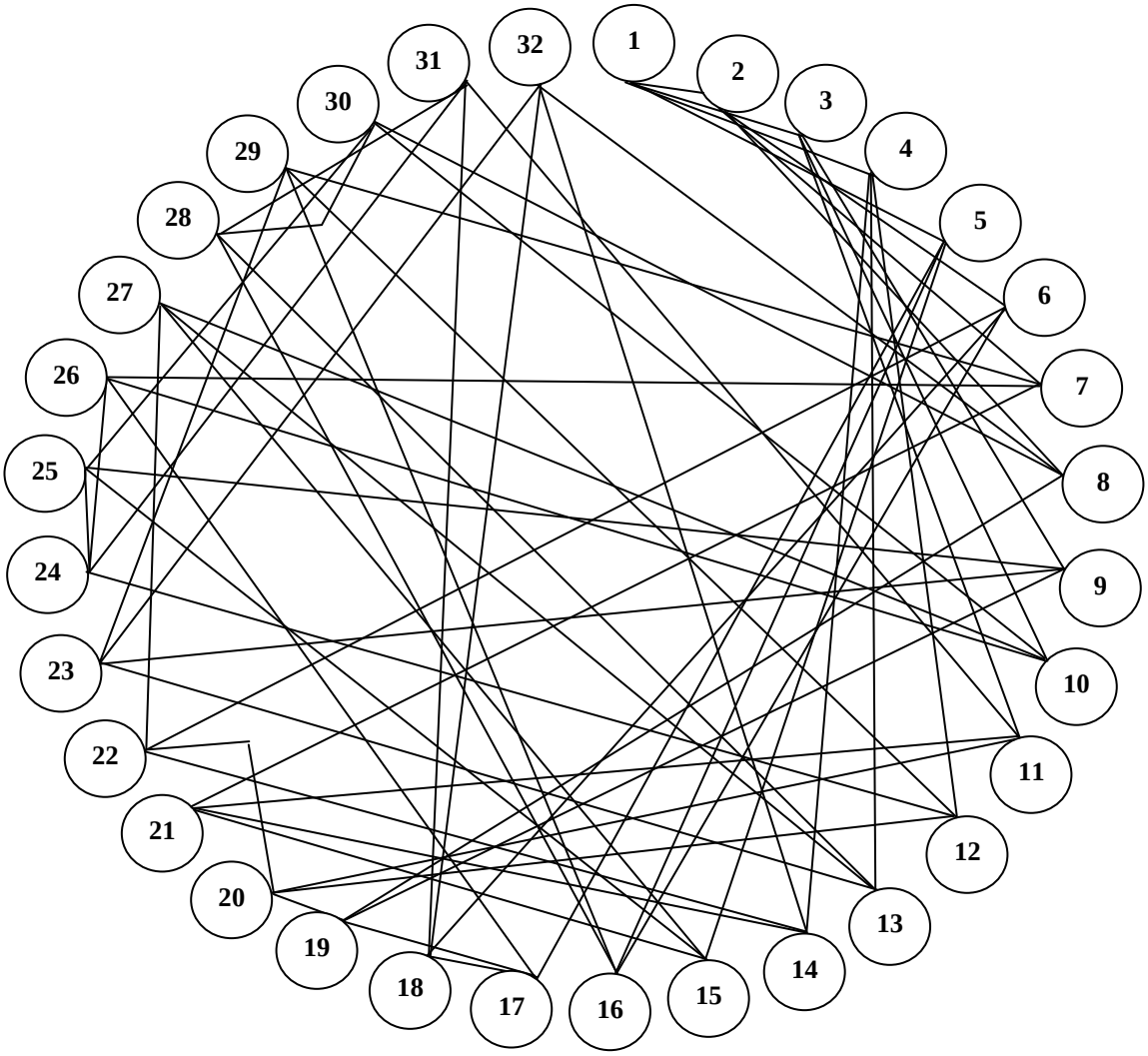


Рис. 7.5. Оптимальный $Л(32,4,6)$-граф; $d = 4, \quad \bar{d} = 2{,}36$

Легко заметить, 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 $\bar{d}$ – от 2,94 до 2,38. Далее, при фиксации степени вершины $v = 4$ увеличение обхвата в графе приводит к некоторому уменьшению среднего диаметра, от $\bar{d}$ =2,38 до $\bar{d}$ =2,36.

Как уже отмечалось (см. 3.1.2), гиперкубы весьма популярны при формировании сетей межмашинных связей в большемасштабных вычислительных системах (см. 5.4). Представляет интерес сравнить структурные показатели гиперкубов, 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 $\bar{d}$ (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 как функций от N.

Т а б л и ц а 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
131072	17	8.5	16	8	6.73	17	6	5	4.39
262144	18	9.0	18	8	6.75	18	6	5	4.62
1048576	20	10.0	20	8	7.41	20	6	5	4.85
16777216	24	12.0	24	10	8.76	24	6	6	5.56
268435456	28	14.0	28	11	10.15	28	6	6	5.94

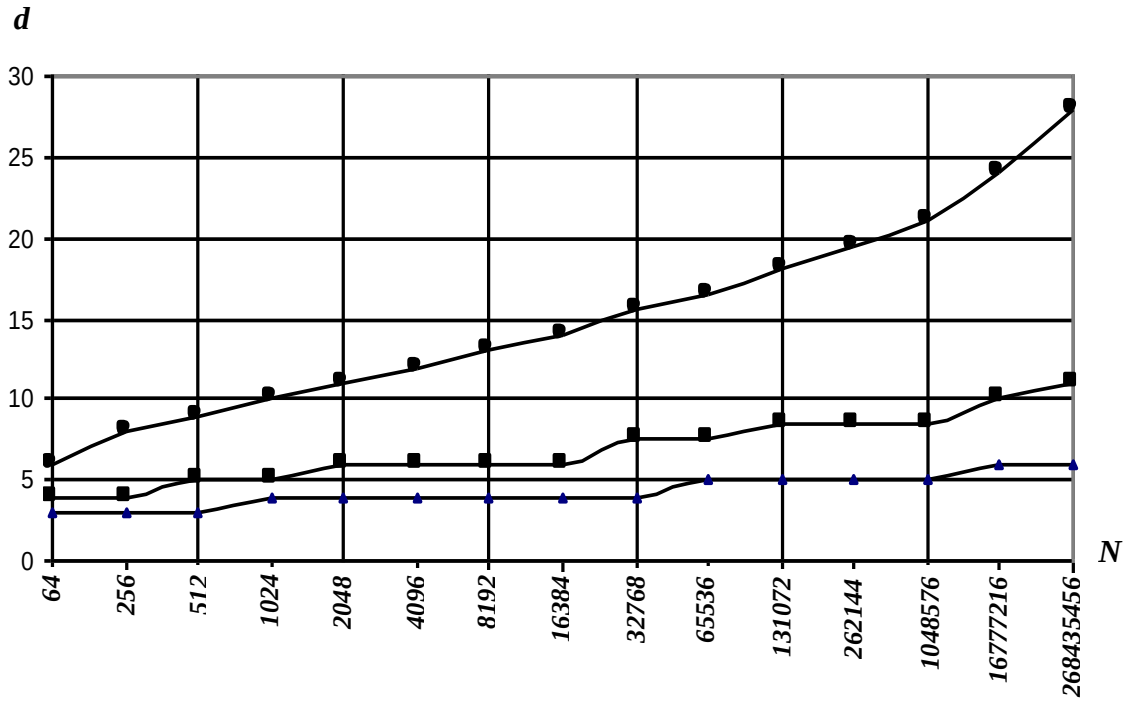


Рис. 7.6 Зависимость диаметра структуры ВС от числа вершин;
– гипер куб, _ D_n-граф, _ $Л(N,v.g)$-граф

Легко заметить, что D_n-графы при одинаковых (или даже меньших на единицу) степенях имеют диаметры, которые меньше чем в 3 раза по сравнению с диаметром гиперкубов. Более того, D_n-графы обладают и меньшими средними диаметрами по сравнению с гиперкубами. Рассматриваемые показатели для $Л(N,v,g)$-графов при $g > 4$ самые лучшие: так диаметры для оптимальных $Л(N,v,g)$-графов оцениваются вели-

чиной $0{,}21\ \log_2 N$, в то время как в гиперкубах – $\log_2 N$. Кроме того, $Л(N,v,g)$-графы характеризуются логарифмической зависимостью диаметров от количества N вершин при фиксированной степени вершин. Из сказанного следует, что в вычислительных системах, использующих D_n- и $Л(N,v,g)$-графы, время межмашинных (межпроцессорных) 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. Учёт топологии межмашинных связей и надёжности 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 надёжна и обеспечивает связность любых подмножеств элементарных машин). Высокая живучесть вычислительных систем с программируемой структурой объясняется тем, что в них достигается близкая к единице вероятность существования связной подсистемы исправных машин, включающей 80 – 90% их общего числа, уже при четырёх межмашинных связях в каждой ЭМ.

Для более основательного изучения теории структур вычислительных систем можно рекомендовать работы [3,6-8].

7.2.2. Режимы функционирования ВС и способы обработки информации

В зависимости от сложности задач и характера их поступления можно выделить следующие *основные режимы работы ВС с программируемой структурой:*

1) решение одной сложной задачи;
2) обработку набора задач;
3) 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 число ветвей в параллельной программе менее общего числа ЭМ в системе, то “избыточные” машины используются для повышения надёжности функционирования ВС.

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 каком промежутке 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 путём 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 исключён. Для режима потока задач созданы методы и алгоритмы [3], обеспечивающие стохастически оптимальное функционирование вычислительных систем.

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 операционной системы) реконфигурируется так, чтобы обеспечить её адекватность текущей мультипрограммной ситуации. Любая подсистема обладает всеми архитектурными свойствами системы, поэтому её 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 вычислительной системе (или на её части – подсистеме) предусматривает следующие этапы:

1) выбор способа обработки данных;
2) разработку параллельного алгоритма (в общем случае отказоустойчивого, способного адаптироваться на число работоспособных ЭМ как на параметр), который эффективно реализуется на ВС при заданной её структуре и выбранном способе обработки данных;
3) запись (параллельного) алгоритма решения задачи на языке (высокого уровня);
4) получение объектной программы решения задачи на системе.

Выделяются *распределённый, матричный и конвейерный способы обработки информации*. Последние два способа обработки информации получили промышленное воплощение в виде высокопроизводительных (порядка $10^8 - 10^{12}$ опер./с) матричных и конвейерных вычислительных систем (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 *распределённой обработке* (рис. 7.7.а) программы и данные расчленяются и рассредоточиваются по элементарным машинам ВС. Допустимо построение адаптирующихся параллельных программ, число ветвей в которых в процессе их реализации соответствует числу (работоспособных) ЭМ в системе. Способ распределённой 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 распределённой обработки информации и позволил сделать выводы, что содержатся в 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, по каждой над своими операндами.

Матричный способ в сравнении с распределённым дает экономию в использовании (распределённой 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 лишён *обобщенный матричный способ обработки информации* (рис. 7.7.в). При последнем 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 распределённого в машину сегмента будет равен её номеру. И для распределённого, и для матричного способов обработки информации характерно то, что в процессе решения задачи имеют место обмены данными между ЭМ системы.

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 распределённым (по памятям всех машин конвейера). В процессе решения задачи данные “пропускаются” через последовательность машин, составляющих конвейер.

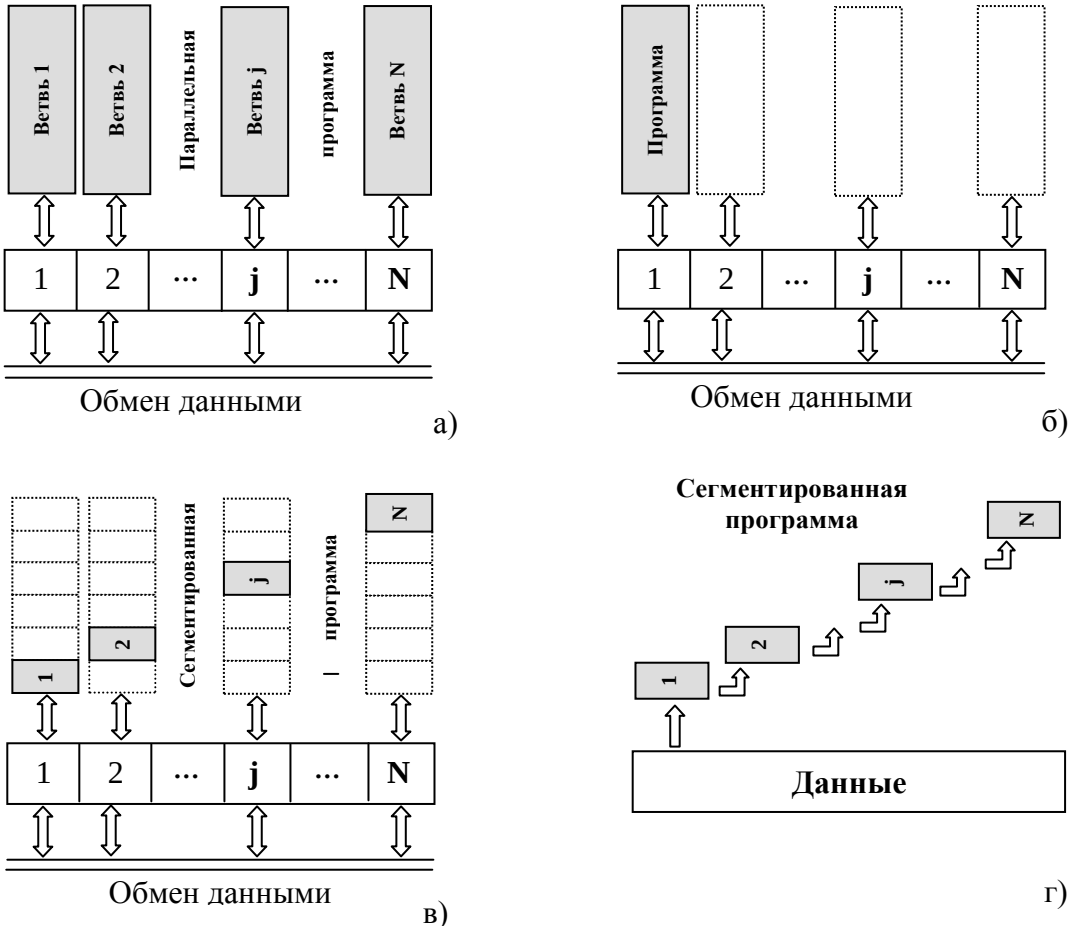


Рис. 7.7. Способы обработки информации:
а) – распределённый; б) – матричный;
в) – обобщённый матричный; г) – конвейерный

Рассмотренные способы обработки информации на ВС иллюстрирует рис. 7.7. Верхние части четырёх рисунков соответствуют программам, а нижние – данным. Стрелками показаны направления потоков данных. Номер $j \in \{1,2,...,N\}$, приписанный к блокам данных и программ (к ветви или сегменту), соответствует номеру ЭМ, в памяти которой они размещены. Несплошной линией изображены те недостающие части программы, которые будут получены машиной в процессе решения задачи от ЭМ, хранящих программы или её сегменты.

Итак, при распределённом способе обработки информации полностью используются возможности ВС с архитектурой MIMD. Матричный и магистральный

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 операционную систему – 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 остроту вопроса о влиянии 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 передаётся 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 их выполнению), а также предопределяют наличие в операционной системе 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 реальном времени некоторого алгоритма на части ресурсов системы. *Совместность процессоров* 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 на уровне задач, представленных в параллельной форме. Мультипрограммирование является следствием разделения ресурсов ВС между: а) процессами, относящимися к различным параллельным программам пользователей; б) процессами пользовательских программ и процессами операционной системы; в) отдельными процессами, относящимися к одной программе решения задачи. Управление работой ВС в мультипрограммном режиме возлагается на ядро (резидентную часть) операционной системы и непосредственно осуществляется с помощью операций управления процессами.

Базовый набор средств, обеспечивающий управление процессами (как процессами пользователей, так и процессами самой операционной системы) каждой ЭМ ВС, составляют средства для порождения и уничтожения процессов и для реализации взаимодействий между ними. На основе базового набора строятся все процессы операционной системы элементарной машины ВС: управление распределением ресурсов,

связь с внешними устройствами и с пользователями, связь с операционными системами других ЭМ. Последние обеспечивают слияние идентичных операционных систем ЭМ в операционную систему ВС. Ясно, что такой способ построения для ВС операционной системы делает её распределённой, 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.

Временное 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 трёх канонических принципов модели коллектива вычислителей. Так, программируемость структуры даёт возможность уже на этапе программирования сложной задачи не только указывать последовательность реализации процессоров (подчинённость процессов по данным и по управлению), но и задавать размещение этих процессов в пространстве ресурсов ВС. Это размещение осуществляется с учётом характера взаимодействия процессов, свойств самих процессов и структуры обрабатываемых данных. Оно определяет структуру подсистемы – области ВС, выделяемой для решения задачи. Далее, программируемость структуры ВС позволяет и в условиях мультипрограммности осуществлять временное 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 операционной системы. Последние обеспечивают управление структурой элементарных машин, 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 вычислительных системах с программируемой структурой с помощью специальных средств, которые в комплексе с традиционными средствами организации функционирования и составляют операционную систему. Функции операционной системы, как правило, реализуются с помощью программных средств. Однако уже в современных условиях технологии микропроцессорных БИС возможна и аппаратурная реализация основных функций операционной системы.

В общем случае операционная система имеет иерархическую структуру. За начальное условие для решения проблемы управления берётся 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 операционной системы. За ним закрепляются параметры, значения которых вырабатываются на основе анализа соответствующей части описания операционной обстановки и служат исходными данными для вышестоящего уровня. Каждый уровень операционной системы допускает представление в виде совокупности процессов, реализуемых в различных ЭМ ВС.