## 3.1. МОДЕЛЬ КОЛЛЕКТИВА ВЫЧИСЛИТЕЛЕЙ

Логика развития средств обработки информации 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 суть вычислительная система (ВС). Архитектура ВС основывается на структурной и функциональной имитации коллектива (ансамбля) людей-вычислителей*. Степень адекватности такой имитации определяет потенциальные архитектурные возможности вычислительной системы.

Ниже приведенная модель может рассматриваться как модель функциональной организации ВС или просто как модель вычислительной системы [1].

### *3.1.1. Принципы построения вычислительных систем*

Каноническую основу конструкции вычислительной системы и ее функционирования составляет модель коллектива вычислителей. Эта модель представляется парой:

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

где $H$ и $A$ – описание конструкции (или просто: 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* при обработке информации (параллельного выполнения операций на множестве $C$ вычислителей, взаимодействующих через связи структуры $G$);

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

3) *однородности 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 конструктивной однородности (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.

### *3.1.2. Структура вычислительных систем*

Структура коллектива вычислителей представляется графом $G$, вершинам которого сопоставлены вычислители $c_i \in C$, а ребрам – линии связи между ними. Проблема выбора (синтеза) структур вычислительных систем не является тривиальной. В самом деле, универсальное решение – структура в виде полного графа, однако такие структуры практически реализуемы при небольшом числе вычислителей. Для достижения производительности ВС в диапазоне $10^9 - 10^{15}$ опер./с при существующих интегральных технологиях требуется число вычислителей порядка $10^0 - 10^6$. Следовательно, структуры последних систем не могут быть организованы по типу полного графа хотя бы из-за ограничений на число выводов с корпусов интегральных схем. Какие же структуры сети связей между вычислителями используются при формировании ВС?

*Простейшие структуры ВС* – нульмерные, одномерные и двумерные (рис. 3.1). В первом случае структура сети межвычислительных связей “ вырождена ”, взаимодействие между вычислителями ВС осуществляется через общую шину. В случае одномерных структур (“линейки” или “кольца”) обеспечивается связь каждого вычислителя с двумя другими (соседними) вычислителями. В нульмерных структурах имеется общий ресурс – шина, в одномерных же структурах этот ресурс трансформируется в распределённый, т.е. в локальные связи между вычислителями. Следовательно, архитектурные возможности (в частности, надёжность) последних структур существенно выше, чем у нульмерных.

Увеличение размерности структур повышает структурную надёжность ВС. В самом деле, двумерные структуры (например, “ решётки” или “двумерные торы”) предоставляют каждому вычислителю непосредственную связь с четырьмя соседними (рис. 3.1). Следовательно, в системах с двумерной структурой при отказах некоторых вычислителей и (или) связей между ними сохраняется возможность организации связных подмножеств исправных вычислителей.

В $n$-мерных структурах каждый вычислитель связан с $2n$ соседними вычислителями. Ясно, что имеются технико-экономические и технологические ограничения в наращивании размерности структуры ВС.

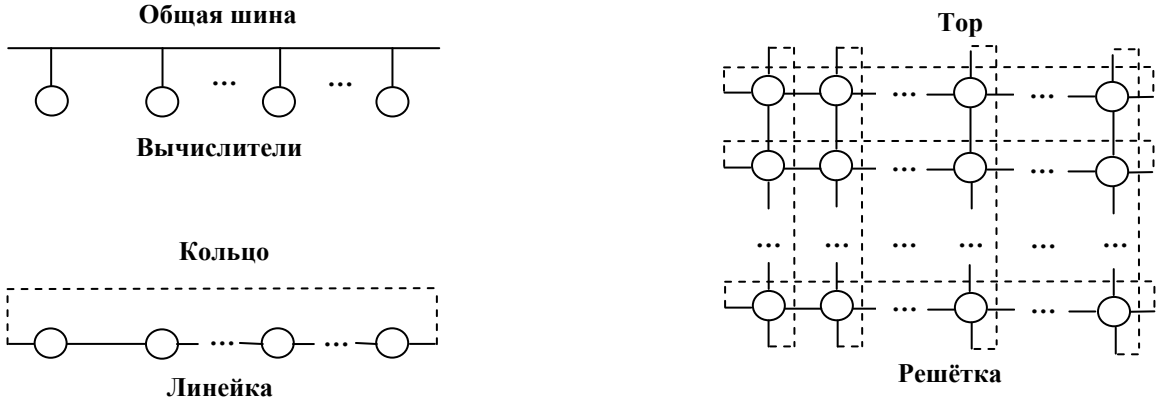


Рис.3.1. Фрагменты простейших структур ВС

*Гиперкубы* или *структуры в виде булевских* $n$*-мерных кубов* нашли широкое применение при построении современных высокопроизводительных ВС с массовым параллелизмом. Гиперкуб, по определению, это однородный граф, для которого справедливо

$$n = \log_2 N,$$

где $N$ – количество вершин, а $n$ – число ребер, выходящих из каждой вершины; $n$ – называют также *размерностью* куба. Итак, каждый вычислитель в гиперкубической ВС имеет связь ровно с $n$ другими вычислителями (рис.3.2).

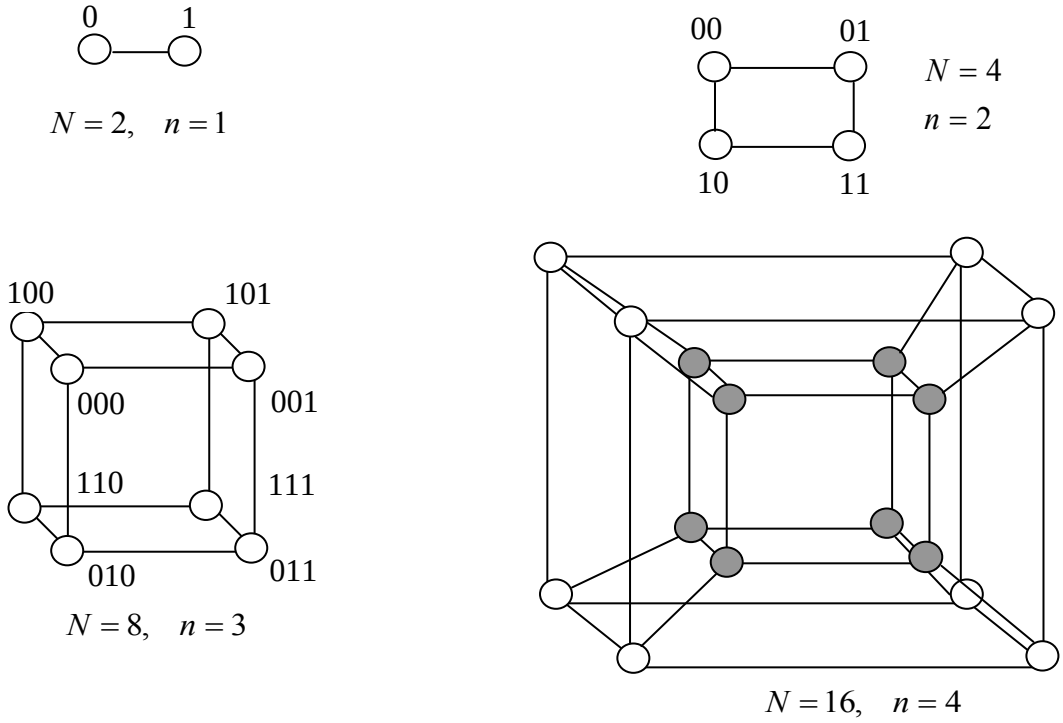


Рис.3.2. Гиперкубические структуры ВС

Если вершины гиперкуба пронумеровать от 0 до $n-1$ в двоичной системе счисления так, что каждый разряд соответствует одному из $n$-направлений, то будет булевский $n$-мерный куб (на рис. 3.2 такая нумерация представлена для $n = 1, 2, 3$).

**Замечания**

1. Тор (от лат. Torus – выпуклость) – это 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 вычислительных систем *тор* – это решетка, в которой имеют место отождествления граничных связей в каждой “строке” и в каждом “столбце” (рис. 3.1). Легко представить размещение такой структуры–графа на поверхности геометрического тора.

3. Гиперкуб $(N < 16, n = 4)$, представленный на рис. 3.2, является тоже 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 (3.5) есть *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, и именно он кладется в основу 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.

Эквивалентное представление алгоритма (3.3) работы коллектива вычислителей есть композиция

$$(A_0 * A_0^{'}) * \ldots * (A_i * A_i^{'}) * \ldots * (A_{N-1} * A_{N-1}^{'}),$$

где $(A_i * A_i^{'})$ 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 с вычислителями $c_j \in C \setminus c_i$. Последний алгоритм является суперпозицией

$$A_i^{'}(P_i^{'}(G)),$$

в которой $G$ – описание структуры коллектива $C$ вычислителей, $P_i^{'}$ – программа для установления связей и выполнения взаимодействий между вычислителем $c_i$ 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 вычислительных систем) 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* $\{A_o^{'}, A_1^{'}, ..., A_i^{'}, ..., A_{N-1}\}$ *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 ни каких 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 произвести следующую запись 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 коллектива вычислителей (3.7) позволяет создавать средства обработки информации разнообразных конфигураций. Представление коллектива вычислителей в качестве макровычислителя делает возможным формирование сверхколлективов или систем коллективов, или *макровычислительных систем*. Модель (3.7) применима и на микроуровне; она позволяет 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* [1, 2]. 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 20 столетия.