Архитектура и топологии вычислительных систем

В настоящее время сфера применения *многопроцессорных вычислительных систем (МВС)* непрерывно расширяется, охватывая все новые области в различных отраслях науки, бизнеса и производства. Стремительное развитие кластерных систем создает условия для использования многопроцессорной вычислительной техники в реальном секторе экономики.

Если традиционно *МВС* применялись в основном в научной сфере для решения вычислительных задач, требующих мощных вычислительных ресурсов, то сейчас из-за бурного развития бизнеса резко возросло количество компаний, отводящих использованию компьютерных технологий и электронного документооборота главную роль. В связи с этим непрерывно растет потребность в построении централизованных вычислительных систем для критически важных приложений, связанных с обработкой *транзакций*, управлением *базами данных* и обслуживанием телекоммуникаций. Можно выделить две основные сферы применения описываемых систем: обработка *транзакций* в режиме реального времени (OLTP, on-line transaction processing) и создание хранилищ данных для организации *систем поддержки принятия решений* (Data Mining, Data Warehousing, Decision Support System). Система для глобальных корпоративных вычислений — это, прежде всего, централизованная система, с которой работают практически все пользователи в корпорации, и, соответственно, она должна все время находиться в рабочем состоянии. Как правило, решения подобного уровня устанавливают в компаниях и корпорациях, где даже кратковременные простои сети могут привести к громадным убыткам. Поэтому для организации такой системы не подойдет обыкновенный сервер со стандартной архитектурой, вполне пригодный там, где нет жестких требований к производительности и времени простоя. *Высокопроизводительные системы* для глобальных корпоративных вычислений должны отличаться такими характеристиками как повышенная производительность, масштабируемость, минимально допустимое время простоя.

Наряду с расширением области применения по мере совершенствования *МВС* происходит усложнение и увеличение количества задач в областях, традиционно использующих *высокопроизводительную вычислительную технику*. В настоящее время выделен круг фундаментальных и прикладных проблем, эффективное решение которых возможно только с использованием сверхмощных вычислительных ресурсов. Этот круг, обозначаемый понятием «Grand challenges», включает следующие задачи:

- предсказания погоды, климата и глобальных изменений в атмосфере;
- науки о материалах;
- построение полупроводниковых приборов;
- сверхпроводимость;
- структурная биология;
- разработка фармацевтических препаратов;
- генетика;
- квантовая хромодинамика;
- астрономия;
- транспортные задачи;
- гидро- и газодинамика;
- управляемый термоядерный синтез;
- эффективность систем сгорания топлива;
- геоинформационные системы;
- разведка недр;
- наука о мировом океане;
- распознавание и синтез речи;
- распознавание изображений.

Многопроцессорные вычислительные системы могут существовать в различных конфигурациях. Наиболее распространенными типами *МВС* являются:

- *системы высокой надежности*;
- *системы для высокопроизводительных вычислений*;
- *многопоточные системы*.

Отметим, что границы между этими типами *МВС* до некоторой степени размыты, и часто система может иметь такие свойства или функции, которые выходят за рамки перечисленных типов. Более того, при конфигурировании большой системы, используемой как система общего назначения, приходится выделять блоки, выполняющие все перечисленные функции.

МВС являются идеальной схемой для повышения надежности информационно-вычислительной системы. Благодаря единому представлению, отдельные узлы или компоненты *МВС* могут незаметно для пользователя заменять неисправные элементы, обеспечивая непрерывность и безотказную работу даже таких сложных приложений как *базы данных*.

Катастрофоустойчивые решения создаются на основе разнесения узлов *многопроцессорной системы* на сотни километров и обеспечения механизмов глобальной синхронизации данных между такими узлами.

МВС для высокопроизводительных вычислений предназначены для параллельных расчетов. Имеется много примеров научных расчетов, выполненных на основе параллельной работы нескольких недорогих процессоров, обеспечивающих одновременное проведение большого числа операций.

МВС для высокопроизводительных вычислений обычно собраны из многих компьютеров. Разработка таких систем – процесс сложный, требующий постоянного согласования таких вопросов как инсталляция, эксплуатация и одновременное управление большим числом компьютеров, технических требований параллельного и высокопроизводительного доступа к одному и тому же системному файлу (или файлам), межпроцессорной связи между узлами и координации работы в параллельном режиме. Эти проблемы проще всего решаются при обеспечении единого образа операционной системы для всего кластера. Однако реализовать подобную схему удается далеко не всегда, и обычно она применяется лишь для небольших систем.

Многопоточные системы используются для обеспечения единого интерфейса к ряду ресурсов, которые могут со временем произвольно наращиваться (или сокращаться). Типичным примером может служить группа web-серверов.

SMP-архитектура

SMP (symmetric multiprocessing) – ***симметричная многопроцессорная архитектура***. Главной особенностью систем с архитектурой SMP является наличие *общей физической памяти*, разделяемой всеми процессорами.

Рис. 3.1. Схематический вид SMP-архитектуры

Память служит, в частности, для передачи сообщений между процессорами, при этом все вычислительные устройства при обращении к ней имеют равные права и одну и ту же адресацию для всех ячеек памяти. Поэтому SMP-архитектура называется симметричной. Последнее обстоятельство позволяет очень эффективно обмениваться данными с другими вычислительными устройствами. SMP-система строится на основе высокоскоростной системной шины (SGI PowerPath, Sun Gigaplane, DEC TurboLaser), к слотам которой подключаются функциональные блоки типов: процессоры (ЦП), подсистема ввода/вывода (I/O) и т. п. Для подсоединения к модулям I/O используются уже более медленные шины (PCI, VME64). Наиболее известными SMP-системами являются SMP-серверы и рабочие станции на базе процессоров Intel (IBM, HP, Compaq, Dell, ALR, Unisys, DG, Fujitsu и др.) Вся система работает под управлением единой ОС (обычно UNIX-подобной, но для Intel-платформ поддерживается Windows NT). ОС автоматически (в процессе работы) распределяет процессы по процессорам, но иногда возможна и явная привязка.

Основные преимущества SMP-систем:

- простота и универсальность для программирования. Архитектура SMP не накладывает ограничений на модель программирования, используемую при создании приложения: обычно используется модель параллельных ветвей, когда все процессоры работают независимо друг от друга. Однако можно реализовать и модели, использующие межпроцессорный обмен.

Использование *общей памяти* увеличивает *скорость* такого обмена, пользователь также имеет доступ сразу ко всему объему памяти. Для SMP-систем существуют довольно эффективные средства автоматического распараллеливания;

- простота эксплуатации. Как правило, SMP-системы используют систему кондиционирования, основанную на воздушном охлаждении, что облегчает их техническое обслуживание;
- относительно невысокая цена.

Недостатки:

- системы с *общей памятью* плохо масштабируются.

Этот существенный недостаток SMP-систем не позволяет считать их по-настоящему перспективными. Причиной плохой *масштабируемости* является то, что в данный момент шина способна обрабатывать только одну транзакцию, вследствие чего возникают проблемы разрешения конфликтов при одновременном обращении нескольких процессоров к одним и тем же областям *общей физической памяти*. Вычислительные элементы начинают друг другу мешать. Когда произойдет такой конфликт, зависит от скорости связи и от количества вычислительных элементов. В настоящее время конфликты могут происходить при наличии 8-24 процессоров. Кроме того, системная шина имеет ограниченную (хоть и высокую) пропускную способность (ПС) и ограниченное число слотов. Все это очевидно препятствует увеличению производительности при увеличении числа процессоров и числа подключаемых пользователей. В реальных системах можно задействовать не более 32 процессоров. Для построения масштабируемых систем на базе SMP используются кластерные или NUMA-архитектуры. При работе с SMP-системами используют так называемую *парадигму программирования* с разделяемой памятью (shared memory paradigm).

MPP-архитектура

MPP (massive parallel processing) – ***массивно-параллельная архитектура***. Главная особенность такой архитектуры состоит в том, что память физически разделена. В этом случае система строится из отдельных модулей, содержащих процессор, локальный банк операционной памяти (ОП), *коммуникационные процессоры* (рутеры) или *сетевые адаптеры*, иногда – жесткие диски и/или другие устройства ввода/вывода. По сути, такие модули представляют собой полнофункциональные компьютеры (см. рис.3.2). Доступ к банку ОП из данного модуля имеют только процессоры (ЦП) из этого же модуля. Модули соединяются специальными коммуникационными каналами. Пользователь может определить логический номер процессора, к которому он подключен, и организовать обмен сообщениями с другими процессорами. Используются два варианта работы операционной системы (ОС) на машинах MPP-архитектуры. В одном полноценная операционная система (ОС) работает только на управляющей машине (front-end), на каждом отдельном модуле функционирует сильно урезанный вариант ОС, обеспечивающий работу только расположенной в нем ветви параллельного приложения. Во втором варианте на каждом модуле работает полноценная UNIX-подобная ОС, устанавливаемая отдельно.

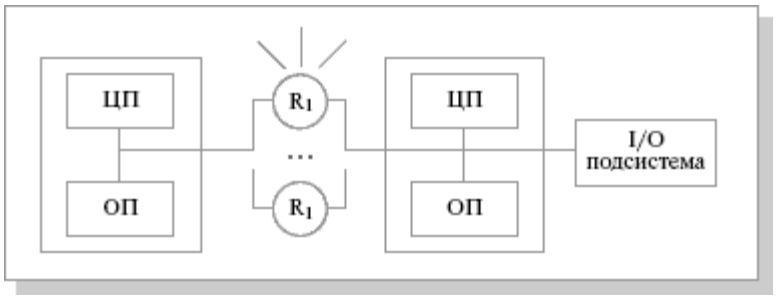

Рис. 3.2. Схематический вид архитектуры с раздельной памятью

Главным преимуществом систем с раздельной памятью является хорошая *масштабируемость*: в отличие от SMP-систем, в машинах с раздельной памятью каждый процессор имеет доступ только к своей *локальной памяти*, в связи с чем не возникает необходимости в потактовой синхронизации процессоров. Практически все рекорды по производительности на сегодня устанавливаются на машинах именно такой архитектуры, состоящих из нескольких тысяч процессоров (ASCI Red, ASCI Blue Pacific).

Недостатки:

- отсутствие *общей памяти* заметно снижает *скорость межпроцессорного обмена*, поскольку нет общей среды для хранения данных, предназначенных для обмена между процессорами. Требуется специальная техника программирования для реализации обмена сообщениями между процессорами;
- каждый процессор может использовать только ограниченный объем локального банка памяти;
- вследствие указанных архитектурных недостатков требуются значительные усилия для того, чтобы максимально использовать системные ресурсы. Именно этим определяется высокая цена программного обеспечения для *массивно-параллельных* систем с раздельной памятью.

Системами с раздельной памятью являются суперкомпьютеры МВС-1000, IBM RS/6000 SP, SGI/CRAY T3E, системы ASCI, Hitachi SR8000, системы Parsytec.

Машины последней серии CRAY T3E от SGI, основанные на базе процессоров Dec Alpha 21164 с пиковой производительностью 1200 Мфлопс/с (CRAY T3E-1200), способны масштабироваться до 2048 процессоров.

При работе с MPP-системами используют так называемую Massive Passing Programming Paradigm – *парадигму программирования* с передачей данных (MPI, PVM, BSPlib).

Гибридная архитектура NUMA

Главная особенность ***гибридной архитектуры*** NUMA (nonuniform memory access) – *неоднородный доступ к памяти*.

Гибридная архитектура совмещает достоинства систем с *общей памятью* и относительную дешевизну систем с раздельной памятью. Суть этой архитектуры – в особой организации памяти, а именно: память физически распределена по различным частям системы, но логически она является *общей*, так что пользователь видит *единое адресное пространство*. Система построена из однородных базовых модулей (плат), состоящих из небольшого числа процессоров и блока памяти. Модули объединены с помощью *высокоскоростного коммутатора*. Поддерживается *единое адресное пространство*, аппаратно поддерживается доступ к удаленной памяти, т.е. к памяти других модулей. При этом доступ к *локальной памяти* осуществляется в несколько раз быстрее, чем к удаленной. По существу, архитектура NUMA является MPP (*массивно-параллельной*) архитектурой, где в качестве отдельных вычислительных элементов берутся SMP (симметричная многопроцессорная архитектура) узлы. Доступ к памяти и обмен данными внутри одного SMP-узла осуществляется через *локальную память* узла и происходит очень быстро, а к процессорам другого SMP-узла тоже есть доступ, но более медленный и через более сложную систему адресации.

Структурная схема компьютера с гибридной сетью: четыре процессора связываются между собой при помощи кроссбара в рамках одного SMP-узла. Узлы связаны сетью типа "бабочка" (Butterfly):

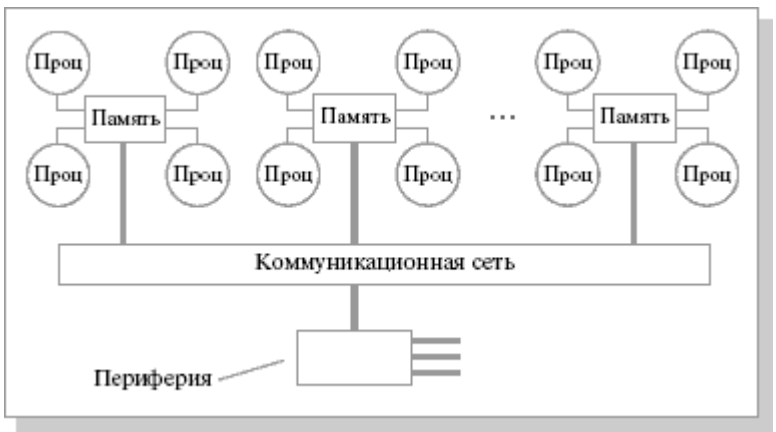

Рис. 3.3. Структурная схема компьютера с гибридной сетью

Впервые идею *гибридной архитектуры* предложил Стив Воллох, он воплотил ее в системах серии Exemplar. Вариант Воллоха – система, состоящая из восьми SMP-узлов. Фирма HP купила идею и реализовала на суперкомпьютерах серии SPP. Идею подхватил Сеймур Крей (Seymour

R.Cray) и добавил новый элемент – когерентный кэш, создав так называемую архитектуру cc-NUMA (Cache Coherent Non-Uniform Memory Access), которая расшифровывается как "*неоднородный доступ к памяти* с обеспечением *когерентности* кэшей". Он ее реализовал на системах типа Origin.

Организация когерентности многоуровневой иерархической памяти

Понятие *когерентности* кэшей описывает тот факт, что все центральные процессоры получают одинаковые значения одних и тех же переменных в любой момент времени. Действительно, поскольку кэш-память принадлежит отдельному компьютеру, а не всей многопроцессорной системе в целом, данные, попадающие в кэш одного компьютера, могут быть недоступны другому. Чтобы этого избежать, следует провести синхронизацию информации, хранящейся в кэш-памяти процессоров.

Для обеспечения *когерентности* кэшей существует несколько возможностей:

- использовать механизм отслеживания шинных запросов (snoopy bus protocol), в котором кэши отслеживают переменные, передаваемые к любому из центральных процессоров и при необходимости модифицируют собственные копии таких переменных;
- выделять специальную часть памяти, отвечающую за отслеживание достоверности всех используемых копий переменных.

Наиболее известными системами архитектуры cc-NUMA являются: HP 9000 V-class в SCA-конфигурациях, SGI Origin3000, Sun HPC 15000, IBM/Sequent NUMA-Q 2000. На сегодня максимальное число процессоров в cc-NUMA-системах может превышать 1000 (серия Origin3000). Обычно вся система работает под управлением единой ОС, как в SMP. Возможны также варианты динамического "подразделения" системы, когда отдельные "разделы" системы работают под управлением разных ОС. При работе с NUMA-системами, так же, как с SMP, используют так называемую *парадигму программирования* с *общей памятью* (shared memory paradigm).

PVP (Parallel Vector Process) – параллельная архитектура с векторными процессорами

Основным признаком PVP-систем является наличие специальных векторно-конвейерных процессоров, в которых предусмотрены команды однотипной обработки векторов независимых данных, эффективно выполняющиеся на *конвейерных функциональных устройствах*. Как правило, несколько таких процессоров (1-16) работают одновременно с общей памятью (аналогично SMP) в рамках многопроцессорных конфигураций. Несколько узлов могут быть объединены с помощью коммутатора (аналогично MPP). Поскольку передача данных в векторном формате осуществляется намного быстрее, чем в скалярном (максимальная скорость может составлять 64 Гбайт/с, что на 2 порядка быстрее, чем в скалярных машинах), то проблема взаимодействия между потоками данных при *распараллеливании* становится несущественной. И то, что плохо *распараллеливается* на скалярных машинах, хорошо *распараллеливается* на векторных. Таким образом, системы PVP-архитектуры могут являться *машинами общего назначения* (general purpose systems). Однако, поскольку *векторные процессоры* весьма дорого стоят, эти машины не могут быть общедоступными.

Наиболее популярны три машины PVP-архитектуры:

1. **CRAY X1**, **SMP**-архитектура. Пиковая производительность системы в стандартной конфигурации может составлять десятки терафлопс.
2. **NEC SX-6**, **NUMA**-архитектура. Пиковая производительность системы может достигать 8 Тфлопс, производительность одного процессора составляет 9,6 Гфлопс. Система масштабируется с единым образом операционной системы до 512 процессоров.
3. **Fujitsu-VPP5000** (**vector parallel processing**), **MPP**-архитектура. Производительность одного процессора составляет 9.6 Гфлопс, пиковая производительность системы может достигать 1249 Гфлопс, максимальная емкость памяти – 8 Тбайт. Система масштабируется до 512

Парадигма программирования на PVP-системах предусматривает векторизацию циклов (для достижения разумной производительности одного процессора) и их *распараллеливание* (для одновременной загрузки нескольких процессоров одним приложением).

Рис. 4.1. CRAY SV-2

Рис. 4.2. Fujitsu-VPP5000

На практике рекомендуется выполнять следующие процедуры:

- производить векторизацию вручную, чтобы перевести задачу в матричную форму. При этом, в соответствии с длиной вектора, размеры матрицы должны быть кратны 128 или 256;
- работать с векторами в виртуальном пространстве, разлагая искомую функцию в ряд и оставляя число членов ряда, кратное 128 или 256.

За счет большой физической памяти (доли терабайта) даже плохо векторизуемые задачи на PVP-системах решаются быстрее на машинах со скалярными процессорами.

Кластер представляет собой два или более компьютеров (часто называемых узлами), объединяемые при помощи *сетевых технологий* на базе шинной архитектуры или коммутатора и предстающие перед пользователями в качестве единого информационно-вычислительного ресурса. В качестве узлов *кластера* могут быть выбраны серверы, рабочие станции и даже обычные персональные компьютеры. Узел характеризуется тем, что на нем работает единственная копия операционной системы. Преимущество кластеризации для повышения работоспособности становится очевидным в случае сбоя какого-либо узла: при этом другой узел *кластера* может взять на себя нагрузку неисправного узла, и пользователи не заметят прерывания в доступе. Возможности *масштабируемости кластеров* позволяют многократно увеличивать производительность приложений для большего числа пользователей технологий (Fast/Gigabit Ethernet, Myrinet) на базе шинной архитектуры или коммутатора. Такие суперкомпьютерные системы являются самыми дешевыми, поскольку собираются на базе *стандартных комплектующих элементов* ("off the shelf"), процессоров, коммутаторов, дисков и внешних устройств.

Кластеризация может осуществляться на разных уровнях компьютерной системы, включая аппаратное обеспечение, операционные системы, программы-утилиты, системы управления и приложения. Чем больше уровней системы объединены кластерной технологией, тем выше надежность, *масштабируемость* и управляемость *кластера*.

Типы кластеров

Условное деление на классы предложено Язеком Радаевским и Дугласом Эдлайном:

- **Класс I**. Класс машин строится целиком из стандартных деталей, которые продают многие поставщики компьютерных компонентов (низкие цены, простое обслуживание, аппаратные компоненты доступны из различных источников).
- **Класс II**. Система имеет эксклюзивные или не слишком широко распространенные детали. Таким образом можно достичь очень хорошей производительности, но при более высокой стоимости.

Как уже отмечалось, *кластеры* могут существовать в различных конфигурациях. Наиболее распространенными типами *кластеров* являются:

- системы высокой надежности;
- системы для высокопроизводительных вычислений;
- *многопоточные системы*.

Отметим, что границы между этими типами *кластеров* до некоторой степени размыты, и *кластер* может иметь такие свойства или функции, которые выходят за рамки перечисленных типов. Более того, при конфигурировании большого *кластера*, используемого как *система общего назначения*, приходится выделять блоки, выполняющие все перечисленные функции.

Кластеры для высокопроизводительных вычислений предназначены для параллельных расчетов. Эти *кластеры* обычно собраны из большого числа компьютеров. Разработка таких *кластеров* является сложным процессом, требующим на каждом шаге согласования таких вопросов как инсталляция, эксплуатация и одновременное управление большим числом компьютеров, технические требования параллельного и высокопроизводительного доступа к одному и тому же системному файлу (или файлам) и межпроцессорная связь между узлами, и координация работы в параллельном режиме. Эти проблемы проще всего решаются при обеспечении единого образа операционной системы для всего *кластера*. Однако реализовать подобную схему удается далеко не всегда и обычно она применяется лишь для не слишком больших систем.

Многопоточные системы используются для обеспечения единого интерфейса к ряду ресурсов, которые могут со временем произвольно наращиваться (или сокращаться). Типичным примером может служить группа web-серверов.

В 1994 году Томас Стерлинг (Sterling) и Дон Беккер (Becker) создали 16-узловой *кластер* из процессоров Intel DX4, соединенных сетью 10 Мбит/с Ethernet с дублированием каналов. Они назвали его "Beowulf" по названию старинной эпической поэмы. *Кластер* возник в центре NASA Goddard Space Flight Center для поддержки необходимыми вычислительными ресурсами проекта Earth and Space Sciences. Проектно-конструкторские работы быстро превратились в то, что известно сейчас как проект Beowulf. Проект стал основой общего подхода к построению параллельных кластерных компьютеров, он описывает многопроцессорную архитектуру, которая может с успехом использоваться для параллельных вычислений. Beowulf-*кластер*, как правило, является системой, состоящей из одного серверного узла (который обычно называется головным), а также одного или нескольких подчиненных (вычислительных) узлов, соединенных посредством стандартной компьютерной сети. Система строится с использованием стандартных аппаратных компонентов, таких как ПК, запускаемые под Linux, стандартные сетевые адаптеры (например, Ethernet) и коммутаторы. Нет особого программного пакета, называемого "Beowulf". Вместо этого имеется несколько кусков программного обеспечения, которые многие пользователи нашли пригодными для построения *кластеров* Beowulf. Beowulf использует такие программные продукты как операционная система Linux, системы передачи сообщений PVM, MPI, системы управления очередями заданий и другие стандартные продукты. Серверный узел контролирует весь *кластер* и обслуживает файлы, направляемые к клиентским узлам.

Проблемы выполнения сети связи процессоров в кластерной системе

Архитектура кластерной системы (способ соединения процессоров друг с другом) в большей степени определяет ее производительность, чем тип используемых в ней процессоров. Критическим параметром, влияющим на величину производительности такой системы, является расстояние между процессорами. Так, соединив вместе 10 персональных компьютеров, мы получим систему для проведения высокопроизводительных вычислений. Проблема, однако, будет состоять в поиске наиболее эффективного способа соединения стандартных средств друг с другом,

поскольку при увеличении производительности каждого процессора в 10 раз производительность системы в целом в 10 раз не увеличится.

Рассмотрим для примера задачу построения симметричной 16-процессорной системы, в которой все процессоры были бы равноправны. Наиболее естественным представляется соединение в виде плоской решетки, где внешние концы используются для подсоединения внешних устройств.

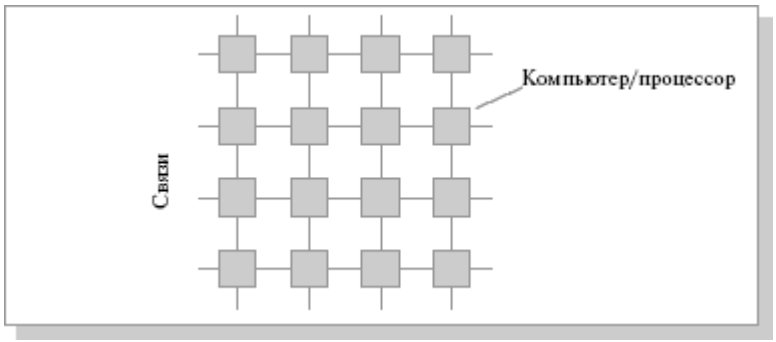

Рис. 4.3. Схема соединения процессоров в виде плоской решетки

При таком типе соединения максимальное расстояние между процессорами окажется равным 6 (количество связей между процессорами, отделяющих самый ближний процессор от самого дальнего). Теория же показывает, что если в системе максимальное расстояние между процессорами больше 4, то такая система не может работать эффективно. Поэтому при соединении 16 процессоров друг с другом плоская схема является нецелесообразной. Для получения более компактной конфигурации необходимо решить задачу о нахождении фигуры, имеющей максимальный объем при минимальной площади поверхности. В трехмерном пространстве таким свойством обладает шар. Но поскольку нам необходимо построить узловую систему, вместо шара приходится использовать куб (если число процессоров равно 8) или *гиперкуб*, если число процессоров больше 8. Размерность *гиперкуба* будет определяться в зависимости от числа процессоров, которые необходимо соединить. Так, для соединения 16 процессоров потребуется четырехмерный *гиперкуб*. Для его построения следует взять обычный трехмерный куб, сдвинуть в нужном направлении и, соединив вершины, получить *гиперкуб* размером 4.

Архитектура *гиперкуба* является второй по эффективности, но самой наглядной. Используются и другие *топологии сетей связи*: трехмерный тор, "кольцо", "звезда" и другие.

Наиболее эффективной является архитектура с топологией "толстого дерева" (fat-tree). Архитектура "fat-tree" (hypertree) была предложена Лейзерсоном (Charles E. Leiserson) в 1985 году. Процессоры локализованы в листьях дерева, в то время как внутренние узлы дерева скомпонованы во внутреннюю сеть. Поддеревья могут общаться между собой, не затрагивая более высоких уровней сети.

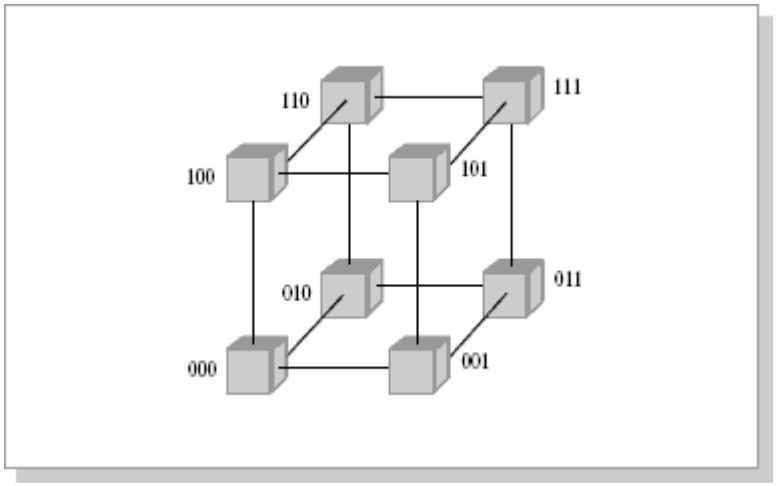

Рис. 4.4. Топология связи, 3-х мерный гиперкуб

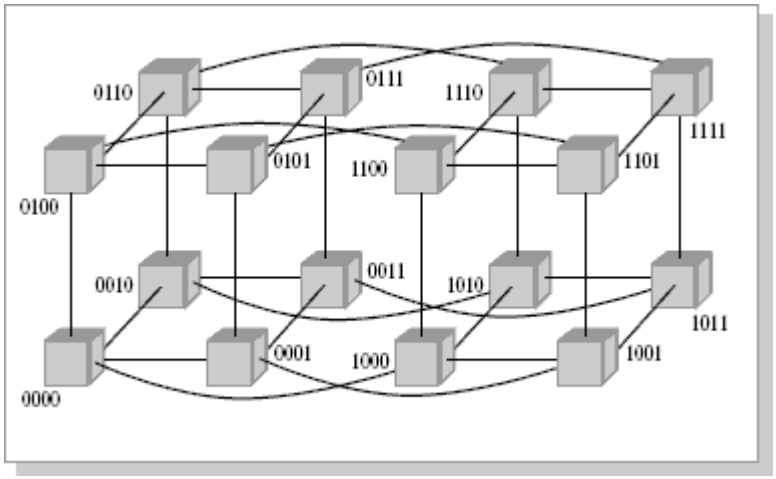

Рис. 4.5. Топология связи, 4-х мерный гиперкуб

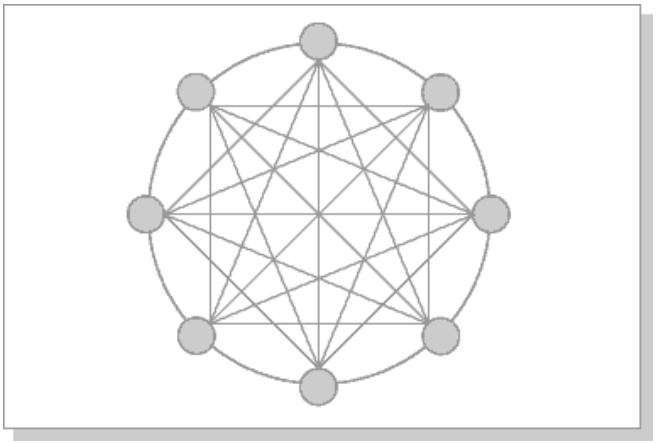

Рис. 4.6. Архитектура кольца с полной связью по хордам (Chordal Ring)

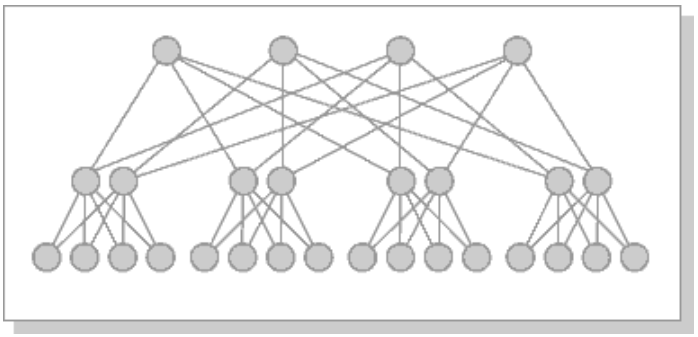

Рис. 4.7. Кластерная архитектура "Fat-tree"

Поскольку способ соединения процессоров друг с другом больше влияет на производительность *кластера*, чем тип используемых в ней процессоров, то может оказаться более целесообразным создать систему из большего числа дешевых компьютеров, чем из меньшего числа дорогих. В *кластерах*, как правило, используются операционные системы, стандартные для рабочих станций, чаще всего *свободно распространяемые* (Linux, FreeBSD), вместе со специальными средствами поддержки параллельного программирования и *балансировки нагрузки*. При работе с *кластерами*, так же, как и с MPP-системами, используют так называемую Massive Passing Programming Paradigm – парадигму программирования с передачей данных (чаще всего – MPI). Умеренная цена подобных систем оборачивается большими накладными расходами на взаимодействие параллельных процессов между собой, что сильно сужает потенциальный класс решаемых задач.

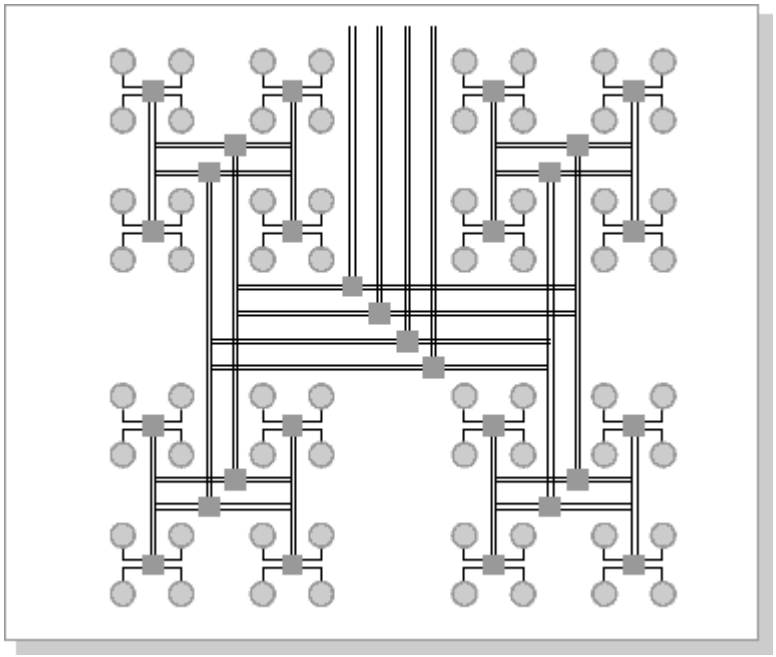

Рис. 4.8. Кластерная архитектура "Fat-tree" (вид сверху на предыдущую схему)