Тема 4. Архитектуры современных компьютеров

Основные вопросы:

1.1. Введение
1.2. Классификации ЭВМ и ВС по М. Флинну и Р. Хокни
1.3. Особенности организации и функционирования архитектур с общей, распределенной и смешанной памятью
1.3.1. Массивно-параллельные системы (MPP)
1.3.2. Симметричные мультипроцессорные системы (SMP)
1.3.3. Системы с неоднородным доступом к памяти (NUMA)
1.3.4. Параллельные векторные системы (PVP)
1.3.5. Кластерные системы
1.4. Организация схем коммутаций
1.4.1. Организация схем коммутации в МВС с общей памятью
1.4.2. Организация средств коммутации в архитектуре “Butterfly”
1.4.3. Организация схем коммутации в МВС с распределенной памятью
1.4.4. Архитектура систем со смешанной организацией памяти

1.1. Введение

Стремительное развитие науки и проникновение человеческой мысли во все новые области вместе с решением поставленных прежде проблем постоянно порождает новые, как правило, более сложные задачи. Во времена первых компьютеров казалось, что увеличение их быстродействия в 100 раз позволит решить большинство проблем, однако гигафлопная производительность современных суперЭВМ сегодня является явно недостаточной для многих ученых.

Электро- и гидродинамика,
сейсморазведка,
прогноз погоды,
моделирование химических соединений,
исследование виртуальной реальности – вот далеко не полный список областей науки, исследователи которых используют каждую возможность ускорить выполнение своих программ.

Наиболее перспективным и динамичным направлением увеличения скорости решения прикладных задач ***является широкое внедрение идей параллелизма в работу вычислительных систем***.

В научной литературе и технической документации можно найти более десятка различных названий, характеризующих лишь общие принципы функционирования параллельных машин: векторно-конвейерные, массивно-параллельные, компьютеры с широким командным словом, систолические массивы, гиперкубы, спецпроцессоры и мультипроцессоры, иерархические и кластерные компьютеры, dataflow, матричные ЭВМ и многие другие. Если же к подобным названиям для полноты описания добавить еще и данные о таких важных параметрах, как, например, организация памяти, топология связи между процессорами, синхронность работы отдельных устройств или способ исполнения

арифметических операций, то число различных архитектур станет и вовсе необозримым.

Попытки систематизировать все множество архитектур начались после опубликования в конце 60-х годов **М. Флинном** первого варианта классификации ВС, и непрерывно продолжаются по сей день. Ясно, что навести порядок в хаосе очень важно для лучшего понимания исследуемой предметной области, однако нахождение удачной классификации может иметь целый ряд существенных следствий.

Вспомним открытый в 1869 году **Д.И. Менделеевым** периодический закон, позволивший предсказать существование и свойства неизвестных до тех пор элементов, например, галлия.

Существующая классификация растительного и животного мира, в отличие от периодического закона, носит скорее описательный характер. С ее помощью намного сложнее предсказывать существование нового вида, однако знание того, что исследуемый экземпляр принадлежит к тому или иному роду/семейству/отряду/классу позволяет оправданно предположить наличие у него вполне определенных свойств.

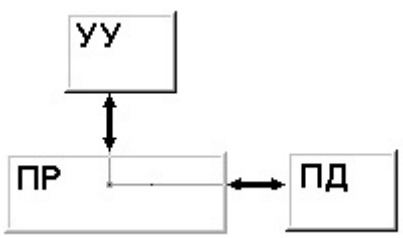
Подобную классификацию хотелось бы найти и для архитектур параллельных вычислительных систем. Основной вопрос – что заложить в основу классификации – может решаться по-разному, в зависимости от того, для кого данная классификация создается и на решение какой задачи направлена.

Удачная классификация могла бы подсказать возможные пути совершенствования компьютеров и в этом смысле она должна быть достаточно содержательной. Трудно рассчитывать на нахождение нетривиальных «белых пятен», но размышления о возможной систематике с точки зрения простоты и технологичности программирования могут оказаться чрезвычайно полезными для определения направлений поиска новых архитектур.

Рассмотрим наиболее известные на сегодня классификации.

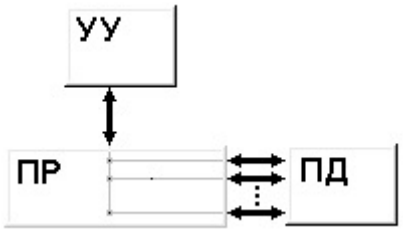
1.2. Классификации ЭВМ и ВС по М. Флинну

По-видимому, самой ранней и наиболее известной является классификация архитектур вычислительных систем, предложенная в 1966 году **М. Флинном**. Классификация базируется на понятии ***потока,*** под которым понимается **последовательность элементов, команд или данных, обрабатываемая процессором**. На основе числа потоков команд и потоков данных М. Флинн выделяет четыре класса архитектур: **SISD (ОКОД), MISD (МКОД), SIMD (ОКМД) и MIMD (МКМД)**.

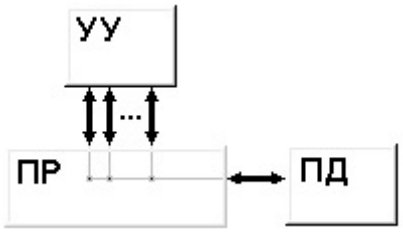


SISD (single instruction stream/single data stream) – одиночный поток команд и одиночный поток данных. К этому классу относятся, прежде всего, классические последовательные машины, или иначе, машины фон-неймановского типа, например, PDP-11 или VAX 11/780. В таких машинах есть только один поток команд, все команды обрабатываются последовательно друг за

другом и каждая команда инициирует одну операцию с одним потоком данных. Для увеличения скорости обработки команд и скорости выполнения арифметических операций может применяться конвейерная обработка. Поэтому, как машина CDC 6600 со скалярными функциональными устройствами, так и CDC 7600 с конвейерными попадают в этот класс.

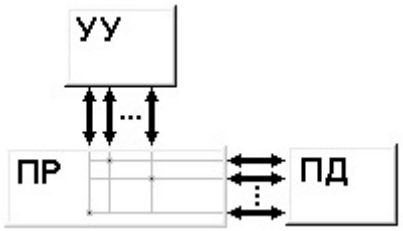


SIMD (single instruction stream/multiple data stream) – одиночный поток команд и множественный поток данных. В архитектурах подобного рода сохраняется один поток команд, включающий, в отличие от предыдущего класса, векторные команды. Это позволяет выполнять одну арифметическую операцию сразу над многими данными – элементами вектора. Способ выполнения векторных операций не оговаривается, поэтому обработка элементов вектора может производиться либо процессорной матрицей, как в ILLIAC IV, либо с помощью конвейера, как, например, в машине CRAY-1.



MISD (multiple instruction stream/single data stream) – множественный поток команд и одиночный поток данных. Определение подразумевает наличие в архитектуре многих процессоров, обрабатывающих один и тот же поток данных. Однако ни М. Флинн, ни другие специалисты в области архитектуры компьютеров до некоторого времени не могли представить убедительный пример реально существующей вычислительной системы, построенной на данном принципе. Ряд исследователей относят конвейерные машины к данному классу, однако это не нашло окончательного признания в научном сообществе.

Появившиеся многоядерные компьютеры можно отнести к данному классу.



MIMD (multiple instruction stream/multiple data stream) – множественный поток команд и множественный поток данных. Этот класс предполагает, что в вычислительной системе есть несколько устройств обработки команд, объединенных в единый комплекс и работающих каждое со своим потоком команд и данных.

Можно отметить два недостатка в классификации **М. Флинна**. Во-первых, некоторые заслуживающие внимания архитектуры, например, dataflow и векторно-конвейерные машины, четко не вписываются в данную классификацию. Во-вторых, класс MIMD чрезвычайно заполнен. Поэтому необходимо средство, более избирательно систематизирующее архитектуры, которые по М. Флинну попадают в один класс, но совершенно различны по числу процессоров, природе и топологии связи между ними, по способу организации памяти и, конечно же, по технологии программирования.

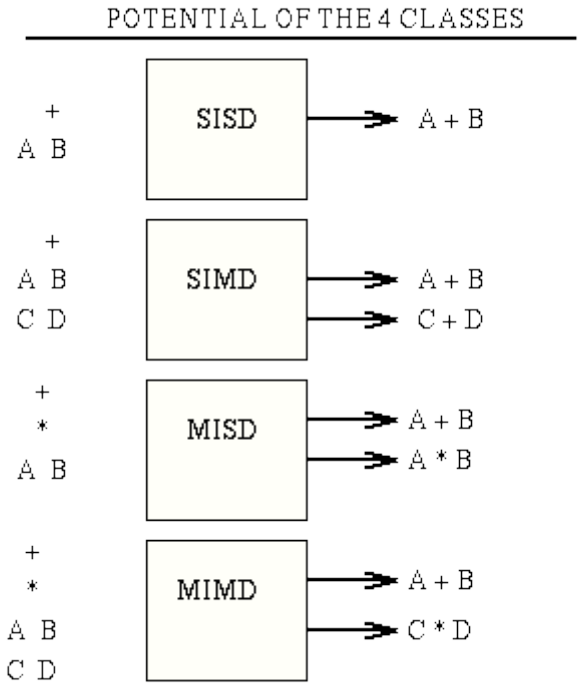


Рис.1.1. Пояснения к классификации М.Флинна на примерах

1.3. Особенности организации и функционирования архитектур с общей, распределенной и смешанной памятью

Классифицируя современные компьютеры, которые практически все относятся к классу MIMD (см. рис. 1.2), будем основываться на анализе используемых в системах ***способах организации оперативной памяти.***

Данный поход позволяет различать два важных типа многопроцессорных систем – **мультипроцессоры** (multiprocessors или системы с общей разделяемой

памятью) и **мультикомпьютеры** (multicomputers или системы с распределенной памятью).

Для **мультипроцессоров** учитывается способ построения общей памяти. Возможный подход – использование единой (централизованной) общей памяти. Такой подход обеспечивает однородный доступ к памяти (**uniform memory access** or **UMA**) и служит основой для построения векторных суперкомпьютеров (**parallel vector processor**, **PVP**) и симметричных мультипроцессоров (**symmetric multiprocessor** or **SMP**). Среди примеров первой группы суперкомпьютер Cray T90, ко второй группе относятся IBM eServer p690, Sun Fire E15K, HP Superdome, SGI Origin 300 и др.

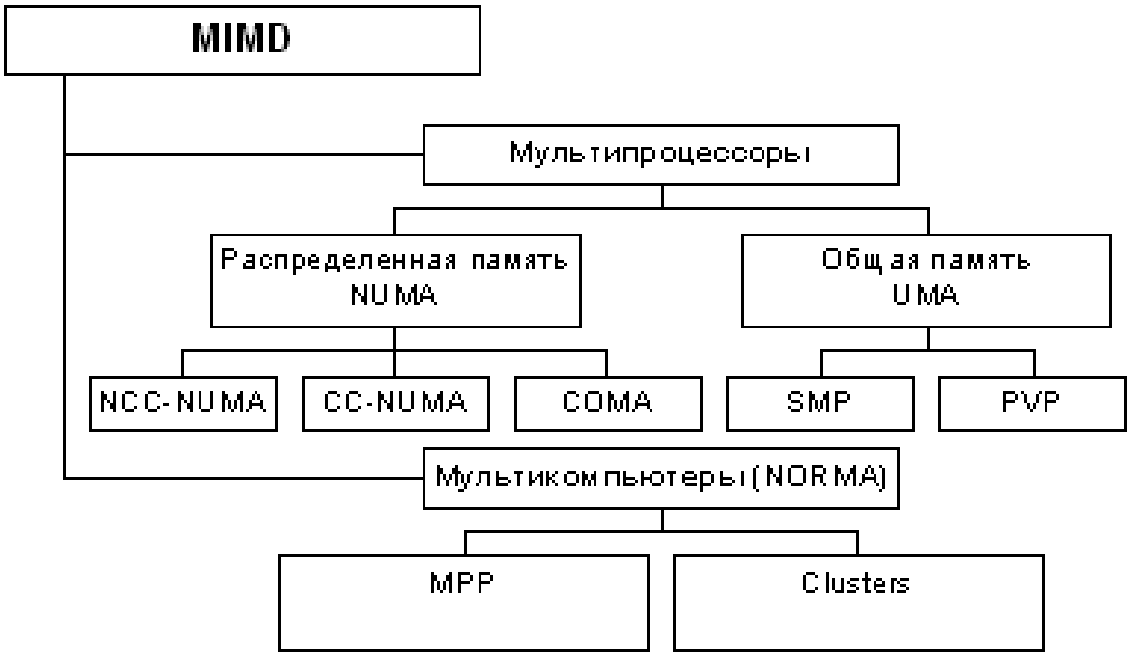


Рис. 1.2. Классификация систем MIMD

Общий доступ к данным может быть обеспечен и ***при физически распределенной памяти*** (при этом, естественно, длительность доступа уже не будет одинаковой для всех элементов памяти). Такой подход именуется как неоднородный доступ к памяти (**non-uniform memory access** or **NUMA**).

Среди систем с таким типом памяти выделяют:

- Системы, в которых для представления данных используется только локальная кэш память имеющихся процессоров (с**ache-only memory architecture** or **COMA**); примерами таких систем являются, например, KSR-1 и DDM;
- Системы, в которых обеспечивается однозначность (когерентность) локальных кэш памяти разных процессоров (**cache-coherent NUMA** or **cc-NUMA**); среди систем данного типа SGI Origin2000, Sun HPC 10000, IBM/Sequent NUMA-Q 2000;
- Системы, в которых обеспечивается общий доступ к локальной памяти разных процессоров без поддержки на аппаратном уровне когерентности кэша (**non-cache coherent NUMA** or **ncc-NUMA**); к данному типу относится, например, система Cray T3E.

Мультикомпьютеры уже не обеспечивают общий доступ ко всей имеющейся в системах памяти (**no-remote memory access** or **NORMA**). Данный подход используется при построении двух важных типов многопроцессорных вычислительных систем – массивно-параллельных систем (**massively parallel processor** or **MPP**) и кластеров (**clusters**). Среди представителей первого типа систем – IBM RS/6000 SP2, Intel PARAGON/ASCI Red, транспьютерные системы Parsytec и др.; примерами кластеров являются, например, системы AC3 Velocity, NCSA/NT Supercluster и др.

Следует отметить чрезвычайно быстрое развитие кластерного типа многопроцессорных вычислительных систем.

1.3.1. Массивно-параллельные системы (MPP)

На рис. 1.3 представлена типовая архитектура вычислительных систем с распределенной памятью.

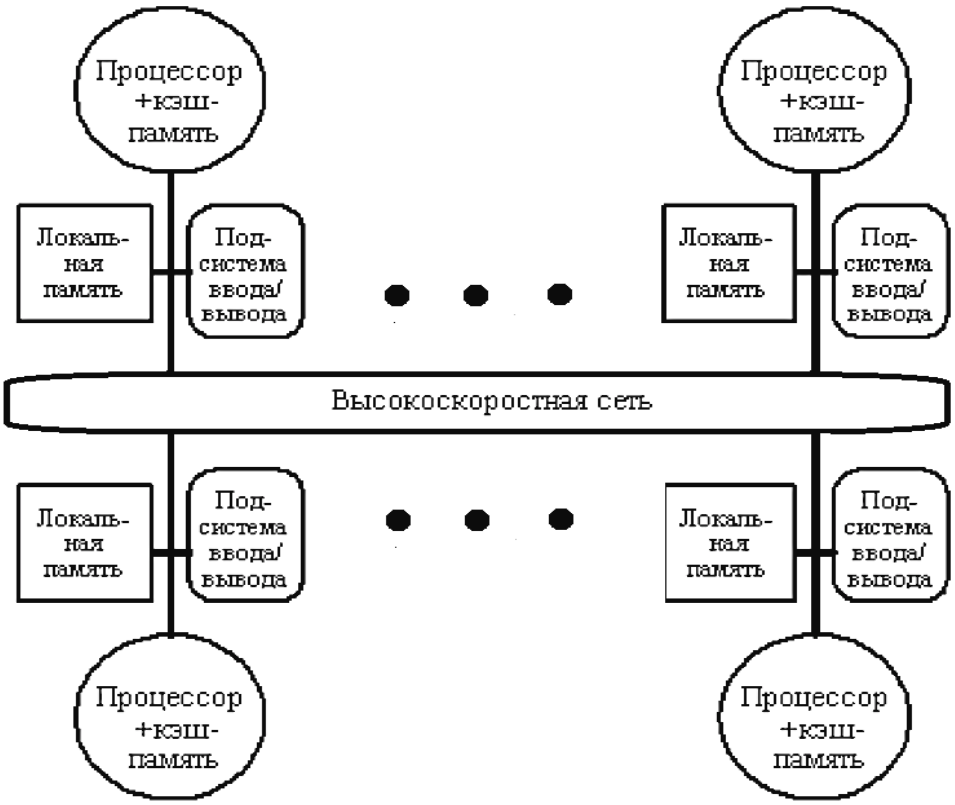


Рис. 1.3. Типовая архитектура машины с распределенной памятью

Архитектура: Система состоит из однородных вычислительных узлов, включающих:

- один или несколько центральных процессоров (обычно RISC),
- локальную память (прямой доступ к памяти других узлов невозможен),
- коммуникационный процессор или сетевой адаптер,
- иногда – жесткие диски (как в SP) и/или другие устройства Вв/Выв.

К системе могут быть добавлены специальные узлы ввода-вывода и управляющие узлы. Узлы связаны через некоторую коммуникационную среду (высокоскоростная сеть, коммутатор и т.п.)

Примеры: IBM RS/6000 SP2, Intel PARAGON/ASCI Red, SGI/CRAY T3E, Hitachi SR8000, транспьютерные системы Parsytec и др.

Масштабируемость: Общее число процессоров в реальных системах достигает нескольких тысяч (ASCI Red, Blue Mountain, Jaguar,…).

Операционная система: Существуют два основных варианта реализации:

1. полноценная ОС работает только на управляющей машине (front-end), на каждом узле работает сильно урезанный вариант ОС, обеспечивающий только работу расположенной в нем ветви параллельного приложения. Пример: Cray T3E.
2. на каждом узле работает полноценная UNIX-подобная ОС (вариант, близкий к кластерному подходу).

Пример: IBM RS/6000 SP + операционная система AIX, которая устанавливается отдельно на каждом узле.

Модель программирования: Программирование в рамках модели передачи сообщений (MPI, PVM, BSPlib)

Распределенность памяти означает, что каждый процессор имеет непосредственный доступ только к своей локальной памяти, а доступ к данным, расположенным в памяти других процессоров, выполняется другими способами.

Чтобы переслать информацию от процессора к процессору, необходим механизм передачи сообщений по сети, связывающей вычислительные узлы. Для абстрагирования от подробностей функционирования коммуникационной аппаратуры и программирования на высоком уровне, используются библиотеки передачи сообщений. Несмотря на существенные различия средств межпроцессорного взаимодействия в разных системах по скоростным параметрам и по способу аппаратной реализации, библиотеки обмена сообщениями выполняют приблизительно одни и те же функции.

Выбор топологии машины часто определяет способ решения прикладной задачи. Надо заметить, что оптимизация алгоритмов для параллельных архитектур существенно отличается от той же работы для последовательных систем. Если переход с одного скалярного процессора на другой практически никогда не требует пересмотра алгоритма, то алгоритм, идеально приспособленный для одной параллельной архитектуры, на другой машине (с тем же числом процессоров того же типа) может работать неприемлемо медленно. Для оценки производительности распределенной системы, кроме топологии связей, необходимо знать скорость выполнения арифметических операций, время инициализации канала связи и время передачи единицы объема информации. Если топология системы не тривиальна, то в состав операционной системы или пакета передачи сообщений приходится включать процедуры маршрутизации сообщений, работающие на каждом узле и обеспечивающие пересылку транзитных сообщений. Они также вызывают задержку при передаче информации между узлами, не имеющими прямого канала связи.

Таким образом, применение дешевых процессоров позволяет сделать относительно недорогой суперкомпьютер. Широкому распространению подобных архитектур препятствует в основном отсутствие эффективных параллельных программ, полностью использующих их возможности.

1.3.2. Симметричные мультипроцессорные системы (SMP)

На рис. 1.4 приведена типовая архитектура мультипроцессорных вычислительных систем с общей памятью.

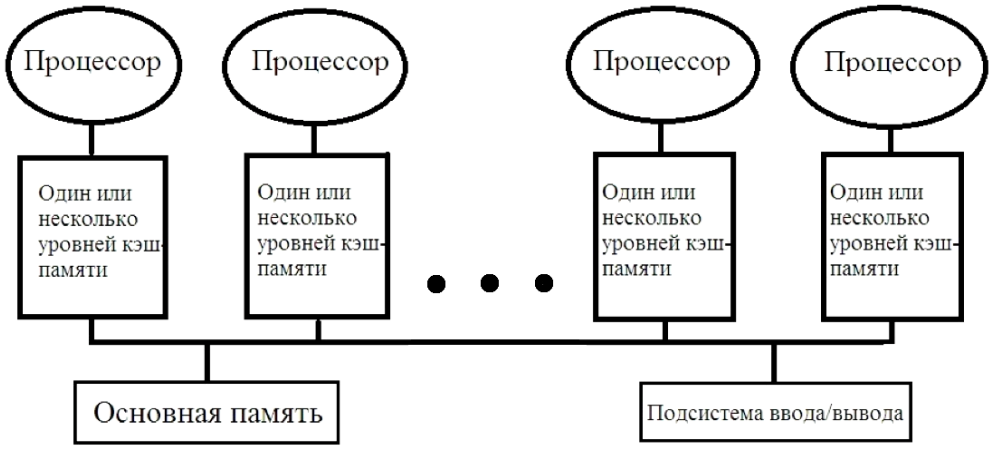


Рис. 1.4. Типовая архитектура мультипроцессорной системы с общей памятью

Архитектура: Система состоит из нескольких однородных процессоров и массива общей памяти (обычно из нескольких независимых блоков). Все процессоры имеют доступ к любой точке памяти с одинаковой скоростью. Процессоры подключены к памяти либо с помощью общей шины (базовые 2–4 процессорные SMP-сервера), либо с помощью crossbar-коммутатора (HP 9000). Аппаратно поддерживается когерентность кэшей.

Примеры: HP 9000 V-class, N-class; SMP-сервера и рабочие станции на базе процессоров Intel (IBM, HP, Compaq, Dell, ALR, Unisys, DG, Fujitsu и др.).

Масштабируемость: Наличие общей памяти сильно упрощает взаимодействие процессоров между собой, однако накладывает сильные ограничения на их число – не более 32 в реальных системах. Для построения масштабируемых систем на базе SMP используются кластерные или NUMA-архитектуры.

Операционная система: Вся система работает под управлением единой ОС (обычно UNIX-подобной, но для Intel-платформ поддерживается Windows NT). ОС автоматически (в процессе работы) распределяет процессы/нити по процессорам (scheduling), но иногда возможна и явная привязка.

Модель программирования: Программирование в модели общей памяти. (POSIX threads, OpenMP). Для SMP-систем существуют сравнительно эффективные средства автоматического распараллеливания.

SMP - это один компьютер с несколькими равноправными процессорами, но с одной памятью, подсистемой ввода/вывода и одной ОС. Каждый процессор имеет доступ ко всей памяти, может выполнять любую операцию ввода/вывода, прерывать другие процессоры и т.д., но это представление справедливо только на уровне программного обеспечения. На самом же деле в SMP имеется несколько устройств памяти.

Каждый процессор имеет, по крайней мере, одну собственную кэш-память, что необходимо для достижения хорошей производительности, поскольку основная память работает слишком медленно по сравнению со скоростью процессоров (и это соотношение все больше ухудшается), а кэш работает со скоростью процессора, но дорог, и поэтому устройства кэш-памяти обладают относительно небольшой емкостью. Из-за этого в кэш помещается лишь оперативная информация, остальное же хранится в основной памяти. Отсюда возникает проблема когерентности кэшей – получение процессором значения, находящегося в кэш-память другого процессора. Это решается при помощи отправки широковещательного запроса всем устройствам кэш-памяти, основной памяти и даже подсистеме ввода/вывода, если она работает с основной памятью напрямую, с целью получения актуальной информации.

Имеется еще одно следствие, связанное с параллелизмом. ***Неявно производимая аппаратурой SMP пересылка данных между кэшами является наиболее быстрым и самым дешевым средством коммуникации в любой параллельной архитектуре общего назначения.*** Поэтому при наличии большого числа коротких транзакций (свойственных, например, банковским приложениям), когда приходится часто синхронизовать доступ к общим данным, архитектура SMP является наилучшим выбором; любая другая архитектура работает хуже.

Тем не менее, архитектуры с разделяемой общей памятью не считаются перспективными. Основная причина довольно проста. Рост производительности в параллельных системах обеспечивается наращиванием числа процессоров, что приводит к тому, что узким местом становится доступ к памяти. Увеличение локальной кэш-памяти не способно полностью решить проблему: задача поддержания согласованного состояния нескольких банков кэш-памяти столь же трудна. Как правило, на основе общей памяти не создают систем с числом процессоров более 32, при необходимости объединяя их в кластерные или NUMA-архитектуры.

1.3.3. Системы с неоднородным доступом к памяти (NUMA)

На рис. 1.5 изображена типовая архитектура мультипроцессорных вычислительных систем с неоднородным доступом к памяти.

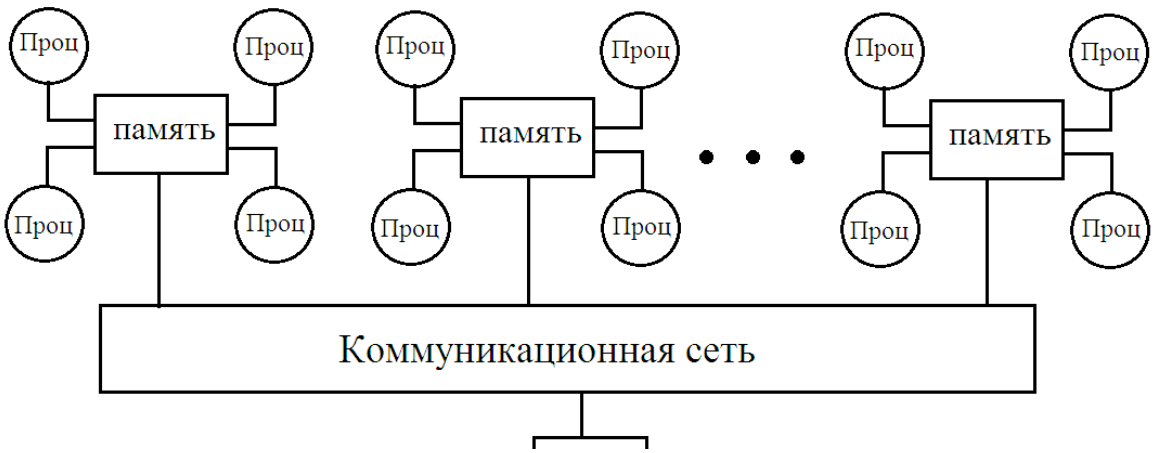


Рис. 1.5. Типовая архитектура мультипроцессорной системы с неоднородным доступом к памяти

Архитектура: Система состоит 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. Доступ к локальной памяти узла осуществляется в несколько раз быстрее, чем к удаленной.

В случае, если аппаратно поддерживается когерентность кэшей во всей системе (обычно это так), говорят об архитектуре cc-NUMA (cache-coherent NUMA)

Примеры: HP 9000 V-class в SCA-конфигурациях, SGI Origin2000, Sun HPC 10000, IBM/Sequent NUMA-Q 2000, SNI RM600 и др.

Масштабируемость: Масштабируемость NUMA-систем ограничивается объемом адресного пространства, возможностями аппаратуры поддержки когерентности кэшей и возможностями операционной системы по управлению большим числом процессоров. На настоящий момент, максимальное число процессоров в NUMA-системах составляет 256 (Origin2000).

Операционная система: Обычно вся система работает под управлением единой ОС, как в SMP. Но возможны также варианты динамического "подразделения" системы, когда отдельные "разделы" системы работают под управлением разных ОС (например, Windows NT и UNIX в NUMA-Q 2000).

Модель программирования: Аналогично SMP.

По сути своей NUMA представляет собой большую SMP-систему, разбитую на набор более мелких и простых SMP. Аппаратура позволяет работать со всеми отдельными устройствами основной памяти составных частей системы (называемых обычно узлами) как с единой гигантской памятью. Этот подход порождает ряд следствий. Во-первых, в системе имеется одно адресное пространство, распространяемое на все узлы. Реальный (не виртуальный) адрес 0 для каждого процессора в любом узле соответствует адресу 0 в частной памяти узла 0; реальный адрес 1 для всей машины – это адрес 1 в узле 0 и т.д., пока не будет использована вся память узла 0. Затем происходит переход к памяти узла 1, затем узла 2 и т.д. Для реализации этого единого адресного пространства каждый узел NUMA включает специальную аппаратуру (Dir), которая решает проблему когерентности кэшей, обеспечивая получение актуальной информации от других узлов.

Понятно, что этот процесс длится несколько дольше, чем если бы требуемое значение находилось в частной памяти того же узла. Отсюда и происходит словосочетание "неоднородный доступ к памяти". В отличие от SMP, время выборки значения зависит от адреса и от того, от какого процессора исходит запрос (если, конечно, требуемое значение не содержится в кэше).

Поэтому ключевым вопросом является ***степень "неоднородности"*** NUMA. Например, если для взятия значения из памяти другого узла требуется только на 10% большее время, то в этом случае NUMA- система может рассматриваться как SMP-машина, и разработанные для SMP программы будут выполняться достаточно эффективно.

Однако в текущем поколении NUMA-систем для соединения узлов используется сеть. Это позволяет включать в систему большее число узлов, до 64 узлов с общим числом процессоров 128 в некоторых системах. В результате, современные NUMA-системы не выдерживают правила 10% – лучшие образцы замедление 200–300% и даже более. При такой разнице в скорости доступа к памяти

для обеспечения должной эффективности следует позаботиться о правильном расположении требуемых данных. Чтобы этого добиться, можно соответствующим образом модифицировать операционную систему (и это сделали поставщики систем в архитектуре NUMA). Например, такая операционная система при запросе из программы блока памяти выделяет память в узле, в котором выполняется эта программа, так что когда процессор ищет соответствующие данные, то находит их в своем собственном узле. Аналогичным образом должны быть изменены подсистемы (включая СУБД), осуществляющие собственное планирование и распределение памяти (что и сделали Oracle и Informix). Как утверждает компания Silicon Graphics, такие изменения позволяют эффективно выполнять в системах с архитектурой NUMA приложения, разработанные для SMP, без потребности изменения кода.

1.3.4. Параллельные векторные системы (PVP)

Архитектура: 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 над общей памятью (аналогично SMP) в рамках многопроцессорных конфигураций. Несколько таких узлов могут быть объединены с помощью коммутатора (аналогично MPP).

Примеры: NEC SX-4/SX-5, линия векторно-конвейерных компьютеров CRAY: от CRAY-1, CRAY J90/T90, CRAY SV1, серия Fujitsu VPP и др.

Модель программирования: Эффективное программирование подразумевает векторизацию циклов (для достижения разумной производительности одного процессора) и их распараллеливание (для одновременной загрузки нескольких процессоров одним приложением).

1.3.5. Кластерные системы

Архитектура: Набор рабочих станций (или даже ПК) общего назначения, используется в качестве дешевого варианта массивно-параллельного компьютера. Для связи узлов используется одна из стандартных сетевых технологий (Fast/Gigabit Ethernet, Myrinet и др.) на базе шинной архитектуры или коммутатора.

При объединении в кластер компьютеров разной мощности или разной архитектуры, говорят о гетерогенных (неоднородных) кластерах.

Узлы кластера могут одновременно использоваться в качестве пользовательских рабочих станций. В случае, когда это не нужно, узлы могут быть существенно облегчены и/или установлены в стойку.

Примеры: NT-кластер в NCSA, Beowulf-кластеры, кластеры МГУ и СПбГУ, кластер МЭИ (см. рис. 1.6 и 1.7) и др.

Операционная система: Используются стандартные для рабочих станций ОС, чаще всего, свободно распространяемые – Linux/FreeBSD, вместе со специальными средствами поддержки параллельного программирования и распределения нагрузки.

Модель программирования: Программирование, как правило, в рамках модели передачи сообщений (чаще всего - MPI).

Кластер МЭИ (ТУ)

В 2007 г. в Московском Энергетическом Институте был установлен кластер с пиковой производительностью 281 T*Flops*. Производительность кластера МЭИ при тестировании на *LinPack* составляет 230 T*flops*.

Все узлы кластера реализованы на базе серверов SUN Fire X4100. В его состав входят 16 вычислительных узлов (computing node – CN) (см. рис. 1.6.) и один управляющий узел (control node – Con N) (см. рис.1.7). Каждый из CN выполнен на двух двухядерных процессорах AMD Opteron 275, а Con N на одноядерных процессорах AMD Opteron 254 .

Каждый CN имеет в своем составе общую память емкостью 4ГБ (MEM 1 и MEM 2) и дисковую память HDD емкостью 2*73 ГБ. Con N имеет общую память емкостью 8ГБ и дисковую – 73ГБ. Связь между ядрами CN осуществляется с помощью локальной сети AMD Hyper Transport, связь между узлами с помощью системной сети InfiniBand, связь с внешним миром осуществляется с помощью вспомогательной сети Gigabit Ethernet. В качестве операционной системы выступает ОС SUSE LINUX Enterprise Server 10

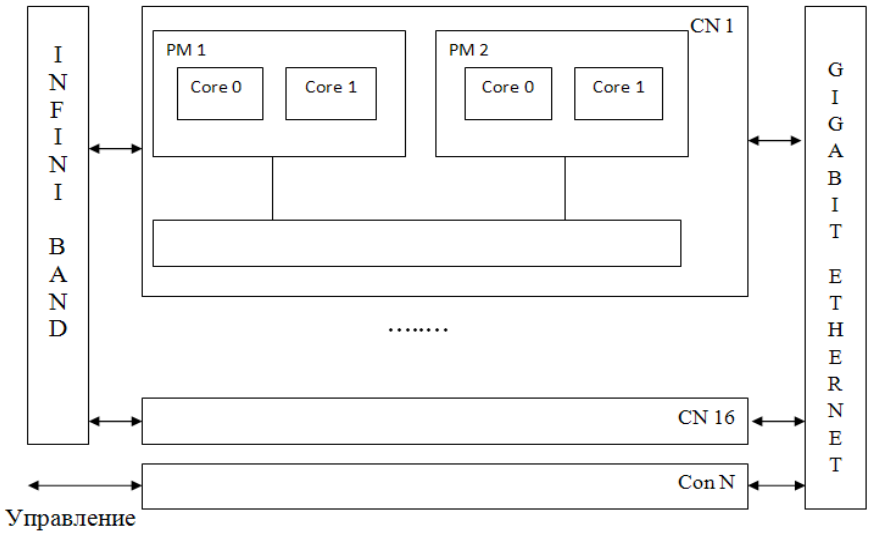
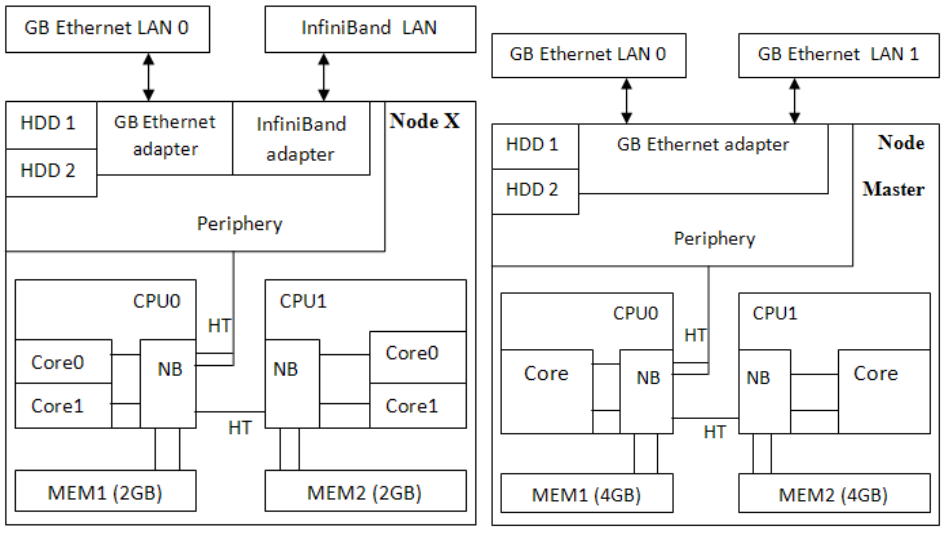


Рис. 1.6. Вычислительный кластер МЭИ



а) б)

Рис. 1.7. Структурные схемы вычислительного (а) и управляющего (б) узлов кластера

Каждый процессор имеет доступ только к одному своему блоку оперативной памяти. Доступ к другому блоку памяти возможен только через контроллер памяти другого процессора.

При построении кластера МЭИ использовались 3 технологии передачи данных: InfiniBand, HyperTransport, Gigabit Ethernet.

1.4. Организация схем коммутации

Важнейшим аспектом создания высокопроизводительных архитектур является построение средств коммутации.

Структура линий коммутации между процессорами вычислительной системы (топология сети передачи данных) определяется, как правило, с учетом возможностей технической реализации. Немаловажную роль при выборе структуры сети играет и анализ интенсивности информационных потоков при параллельном решении наиболее распространенных вычислительных задач. Рассмотрим основные схемы коммутации в МВС разной системы организации памяти.

1.4.1. Организация схем коммутации в МВС с общей памятью

На рис. 1.8 представлена типовая архитектура МВС с общей памятью.

Замечание. В реальных вычислительных системах (ВС) с общей памятью количество процессоров n не превосходит 32. Число модулей памяти m, осуществляющих хранение данных, не превосходит n, то есть $m \le n \le 32$.

Возможно несколько вариантов построения ВС с общей памятью:

1. ВС с единственным модулем памяти ($m = 1$), или многовходовой памятью;
2. ВС с несколькими модулями памяти ($1 < m < n$), или несколькими многовходовыми памятями;
3. ВС с количеством модулей памяти равным количеству процессоров ($m = n$) и коммутацией в виде коммутационной сети.

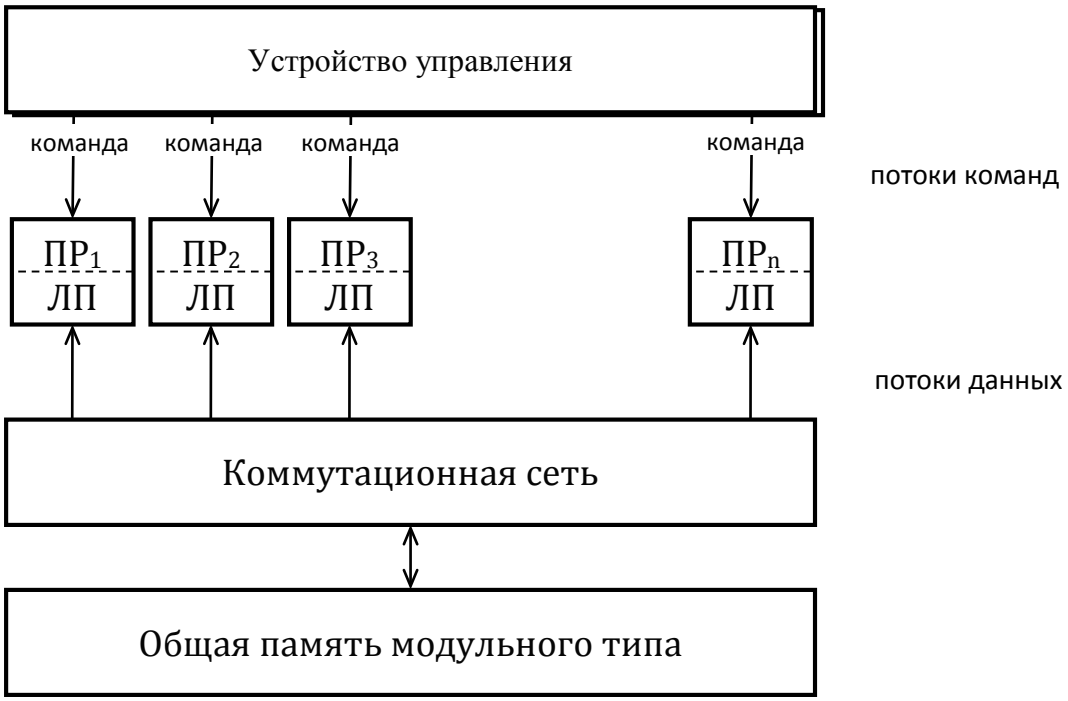


Рис. 1.8. Типовая архитектура МВС с общей памятью

На рис. 1.9, 1.10 и 1.11 представлены варианты реализации.

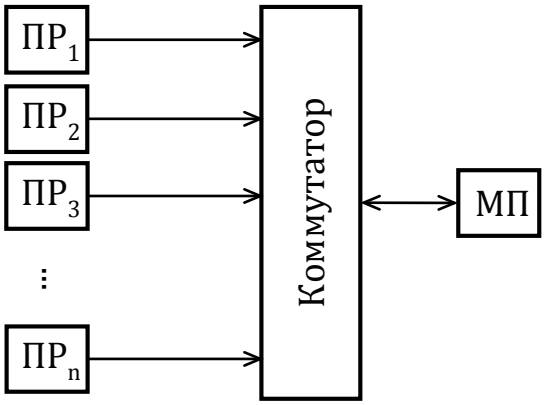


Рис. 1.9. Единый модуль памяти $m=1$

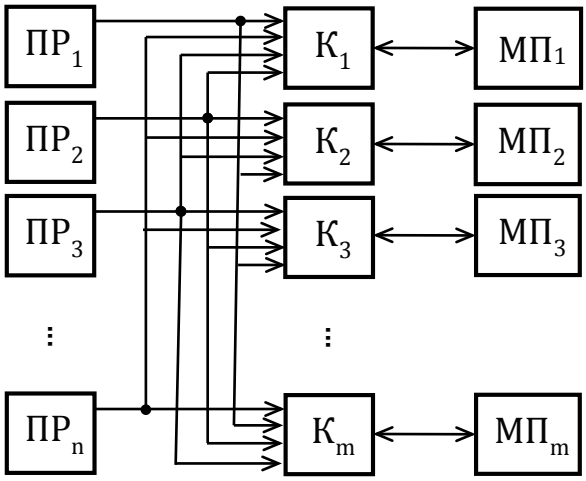


Рис. 1.10. Несколько модулей памяти $1<m<n$

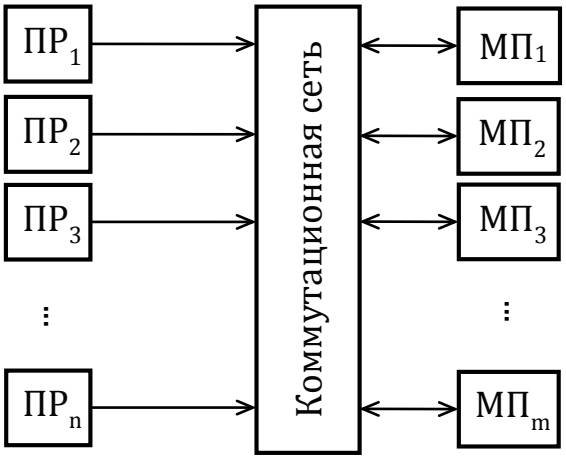


Рис. 1.11. Число модулей памяти равно числу процессоров $m=n$

1.4.2. Организация средств коммутации в архитектуре “Butterfly”

Для архитектуры “Butterfly” - $n=m=256$. Коммутационный узел (КУ), как изображено на рис. 1.12, имеет 4 входа и 4 выхода, причем каждый вход соединён с каждым выходом.

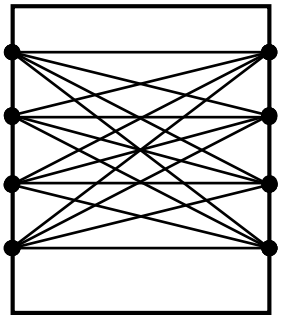


Рис. 1.12. Коммутационный узел «4 х 4»

Коммутационный блок (КБ) представляет собой аналогичную КУ структуру, «атомарными» в которой являются КУ (рис. 1.13).

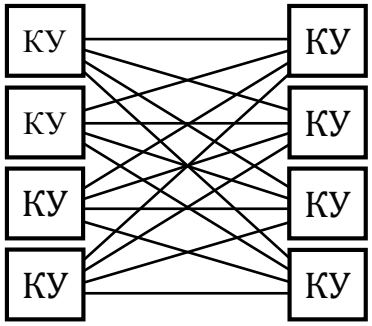


Рис. 1.13. Коммутационный блок «16 х 16»

Коммутационная сеть по типу Butterfly изображена на рис. 1.14.

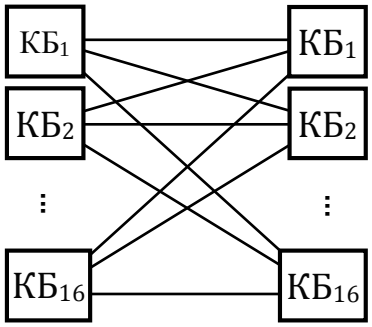


Рис. 1.14. Коммутационная сеть по типу Butterfly «256 х 256»

В МВС с общей памятью в качестве средства передачи информации часто используют шины. Каждый модуль памяти имеет свою шину, к которой подключаются все процессорные узлы. Схематически это показано на рис. 1.15.

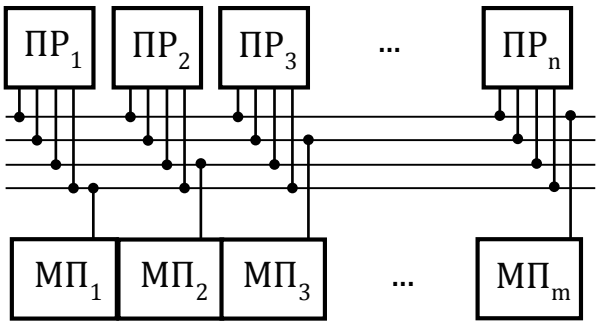


Рис. 1.15. МВС с общей памятью

1.4.3. Организация схем коммутации в МВС с распределенной памятью

На рис. 1.16 представлена типовая архитектура МВС с распределенной памятью.

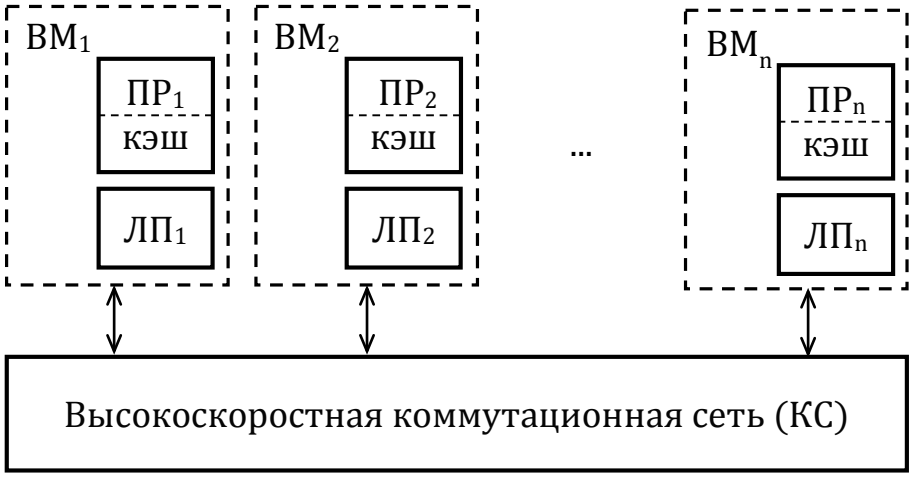


Рис. 1.16. Типовая архитектура схем коммутации в МВС с распределенной памятью

При обсуждении этого типа архитектур будем использовать следующие обозначения: число шин – m, число процессоров – n. Также будем учитывать ограничение: $m < \frac{n}{2}$. Средства коммутации для этого типа архитектуры представляют важнейший структурный элемент ВС.

Наиболее распространенные решения при построении КС для МВС с распределенной памятью можно разбить на три группы.

1. ВС со связями через общую шину (см. рис. 1.17).

Такая схема является, с одной стороны, дешевой и просто реализуемой, а также соответствует структуре передачи данных при решении многих вычислительных задач. Легко наращивается число подключаемых вычислительных модулей (ВМ). С другой стороны, для обеспечения эффективной работы шины требуется тщательное планирование использования шины (арбитр шины), работающей в режиме разделения времени, от которой зависит производительность всей системы.

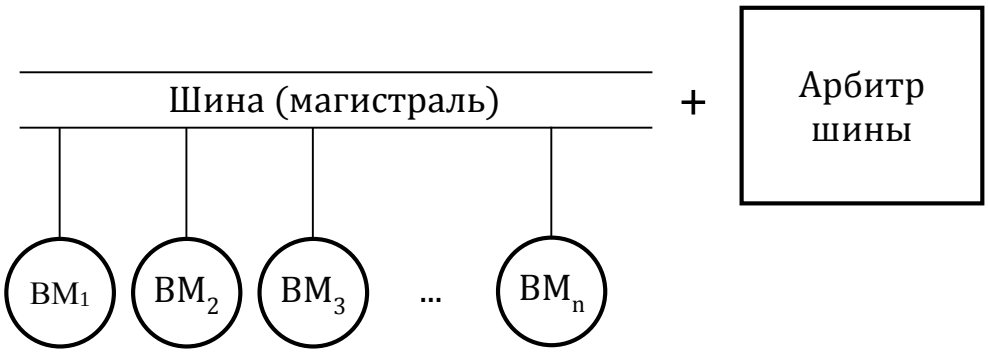


Рис. 1.17. Архитектура схем коммутации в МВС с распределенной памятью со связями через общую шину

2. Более дорогой вариант – архитектура со связями через несколько шин – представлена на рис. 1.18. При таком способе связи поддерживается режим прямого доступа к памяти, причем передача производится обычно блоками из фиксированного набора слов.

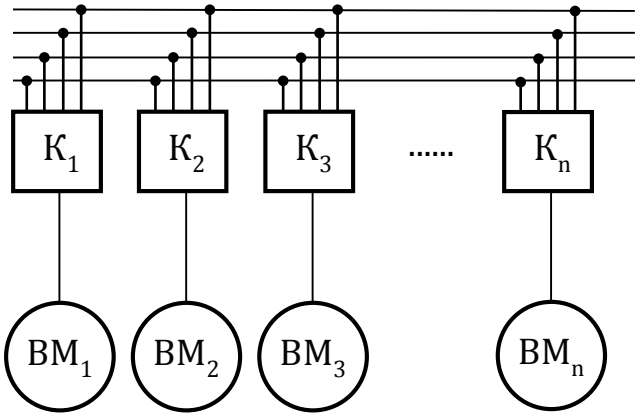


Рис. 1.18. Архитектура схем коммутации в МВС с распределенной памятью со связями через несколько шин

Достоинством архитектур МВС, использующих при коммутации несколько шин, является бо'льшая производительность и надежность, чем у аналогов с одной шиной. Однако для организации эффективных вычислений необходимы *n* коммутаторов шин, а это высокие аппаратные затраты.

3. На рис. 1.19 представлена архитектура со связями через многоступенчатый переключатель.

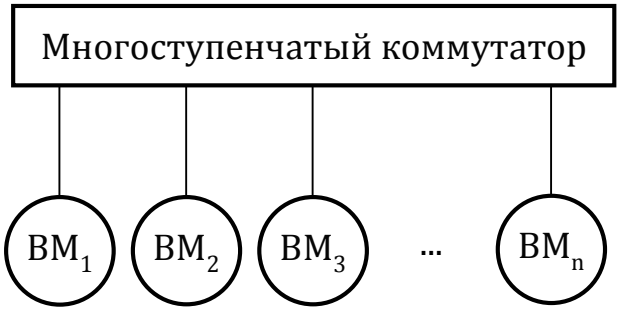


Рис. 1.19. МВС с распределенной памятью со связями через многоступенчатый переключатель

Существует большое разнообразие в организации коммутации такого вида (**многоступенчатого коммутатора**).

3.1. При соединении первого и последнего процессоров линейки через коммутатор получается топология, называемая **кольцом** (рис. 1.20).

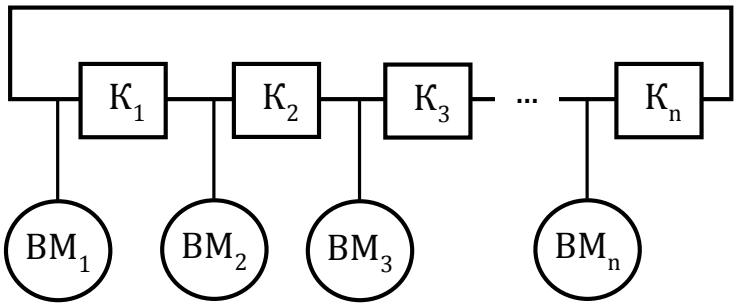


Рис. 1.20. Многоступенчатый коммутатор кольцевого типа

3.2. Система, в которой каждый ВМ связан с каждым другим ВМ прямой линией связи, построена на основе КС с **полным набором связей** (рис. 1.21). Такая топология обеспечивает высокую надежность и минимальные затраты при передаче данных, однако является сложно реализуемой при большом количестве процессов. Каждый узел – это ВМ со своим многовходовым коммутатором.

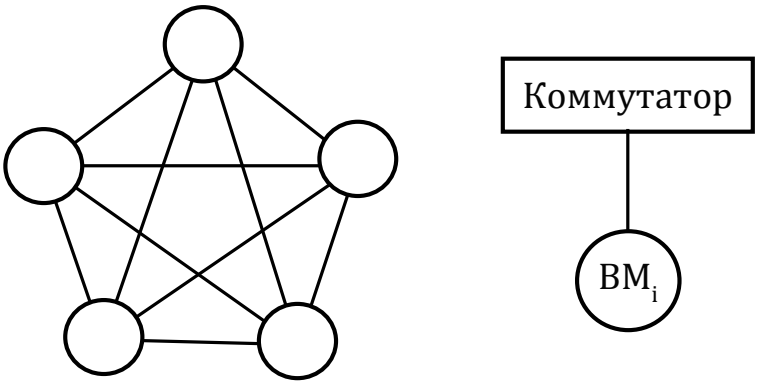


Рис. 1.21. Схема КС с полным набором связей

3.3. Система, в которой все ВМ с помощью своих коммутаторов имеют линии связи с некоторым центральным коммутатором или центральным коммутационным узлом (ЦКУ) или управляющим процессором, называется ВС с топологией типа «**звезда**».

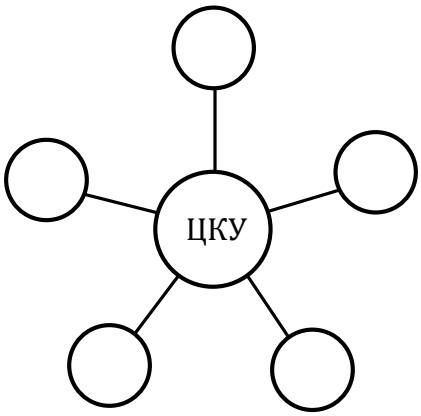


Рис. 1.22. Коммутатор по типу «звезды»

3.4. ВС, в которой топология линий связи образует прямоугольную сетку (обычно двух- или трехмерную), называется системой с **топологией решетки**. Подобная топология может быть достаточно просто реализована и, кроме того, эффективно использована при параллельном выполнении многих численных алгоритмов (например, при реализации методов анализа математических моделей, описываемых дифференциальными уравнениями в частных производных). На рис. 1.23 представлена топология связей по типу «двумерной решетки». Каждый ВМ связан с 4-мя соседями и через них с любыми другими ВМ.

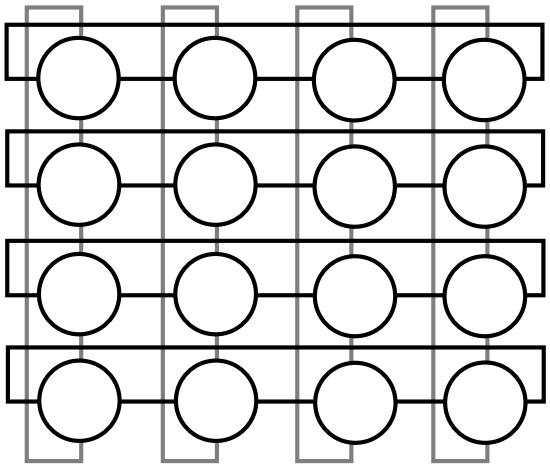
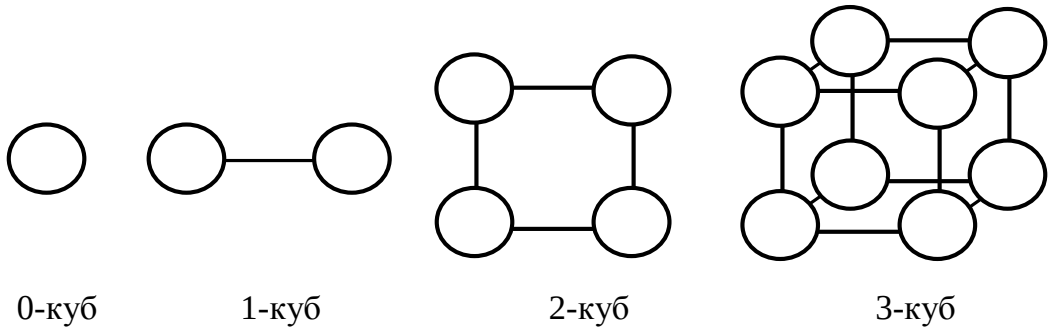


Рис. 1.23. Структура ВС по типу «решетки»

3.5. Топология ВС по типу **«гиперкуб»** – частный случай решетки, когда по каждой размерности сетки имеется только два процессора. Примеры гиперкуб-коммутаторов представлены на рис. 1.24. Гиперкуб содержит 2^N процессоров при размерности гиперкуба *N*.



0-куб 1-куб 2-куб 3-куб

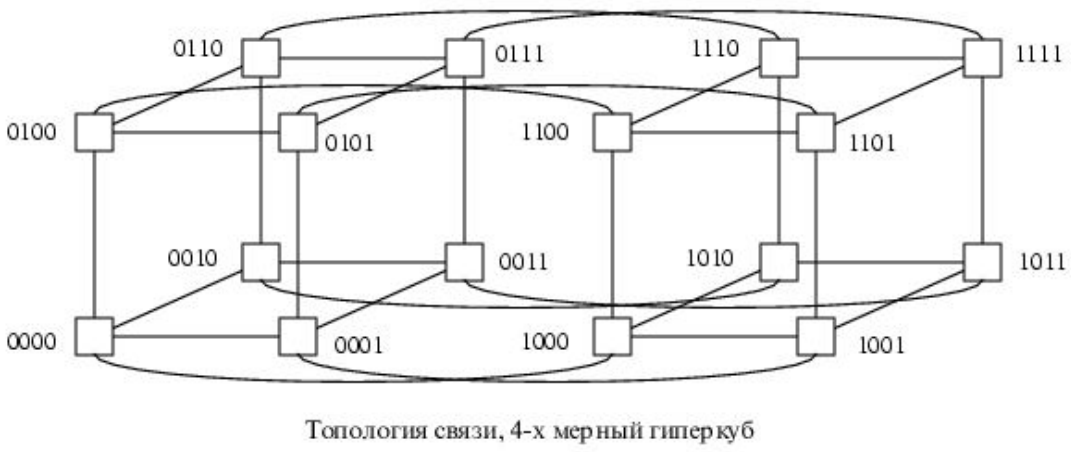


Рис. 1.24. Коммутатор по типу «гиперкуба»

Такой вариант организации сети передачи данных достаточно широко распространен на практике, достаточно прост в построении, и характеризуется следующими свойствами:

1. Два процессора имеют соединение, если двоичные представления их номеров имеют только одну различающуюся позицию.
2. В N-мерном гиперкубе каждый процессор связан ровно с N соседями.
3. Кратчайший путь между двумя любыми процессорами имеет длину, совпадающую с количество различающихся битовых значений в номерах процессов. Отсюда следует, что максимальная длина пути в N-мерном гиперкубе равна N.
4. N-мерный гиперкуб может быть разделен на два (N–1)-мерных гиперкуба (всего возможно N таких разбиений).

Очевидные преимущества такой топологии:

1. ВМ, располагаясь в вершине N куба, не отстоит более чем на N-ребер ни от какого другого ВМ, что значительно облегчает создание эффективных коммуникаций в системе (например, для N=12 допустимое число ВМ 2^{12} =4096!);
2. т.к. структура соединений в N-кубе хорошо согласуется с двоичной логикой, то достаточно легко реализуется алгоритм маршрутизации для передачи сообщений между узлами;
3. между любой парой ВМ существует несколько альтернативных путей коммуникаций, что позволяет в целом снизить задержки при передачах.

1.4.4. Архитектура систем со смешенной организацией памяти

Для решения практических задач часто используют вычислительные системы со смешанной организацией памяти. На рис. 1.25 схематично представлен пример такой архитектуры.

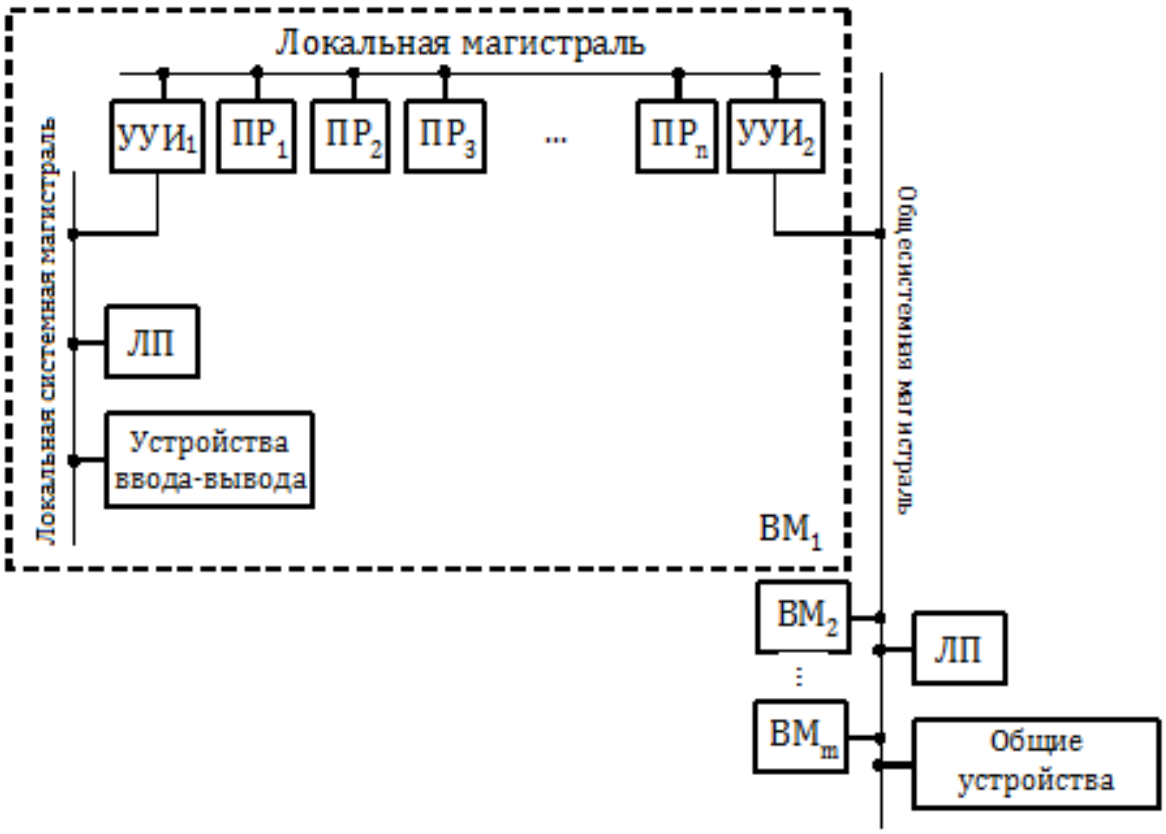


Рис. 1.25. Типовая архитектура систем со смешанной организацией памяти

Здесь:
$УУИ_1$, $УУИ_2$ – устройства управления интерфейсами;
ПР – процессор;
ЛП – локальная память;
ВМ – вычислительный модуль.

P.S. См. презентации по теме из папки «Дополнительные материалы»