Тема 2.6 Архитектура многопроцессорных систем

В настоящее время тенденция в развитии микропроцессоров и систем, построенных на их основе, направлена на все большее повышение их производительности. Вычислительные возможности любой системы достигают своей наивысшей производительности благодаря двум факторам: использованию высокоскоростных элементов и параллельному выполнению большого числа операций.

Существует несколько вариантов классификации систем параллельной обработки данных. По-видимому, самой ранней и наиболее известной является классификация архитектур вычислительных систем, предложенная 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 выделяются четыре класса архитектур:

SISD, MISD, SIMD, MIMD.

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. 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. Не имеет значения тот факт, что для увеличения скорости обработки команд и скорости выполнения арифметических операций процессор может использовать конвейерную обработку. В таком понимании машины данного класса фактически не относятся к параллельным системам.

SIMD (sINgle INsTRuction sTReam / multIPle data sTReam) - одиночный поток команд и множественный поток данных. Применительно к одному микропроцессору этот подход реализован в MMX- и SSE- расширениях современных микропроцессоров. Микропроцессорные системытипа SIMD состоят из большого числа идентичных процессорных элементов, имеющих собственную память. Все процессорные элементы в такой машине выполняют одну и ту же программу. Это позволяет выполнять одну арифметическую операцию сразу над многими данными - элементамивектора. Очевидно, что такая система, составленная из большого числа процессоров, может обеспечить существенное повышение производительности только на тех задачах, при решении которых все процессоры могут делать одну и ту же работу.

MISD (multIPe INsTRuction sTReam / sINgle data sTReam) - множественный поток команд и одиночный поток данных. Определениеподразумевает наличие в архитектуре многих процессоров, обрабатывающих один и тот же поток данных. Ряд исследователей к данному классу относят конвейерные машины.

MIMD (multIPe INsTRuction sTReam / multIPle data sTReam) - множественный поток команд и множественный поток данных. Базовой моделью вычислений в этом случае является совокупность независимых процессов, эпизодически обращающихся к разделяемым данным. В такой системе каждый процессорный элемент выполняет свою программу достаточно независимо от других процессорных элементов. Архитектура MIMD дает большую гибкость: при наличии адекватной поддержки со стороны аппаратных средств и программного обеспечения MIMD может преимущества также архитектура MIMD может работать как однопользовательская система, обеспечивая высокопроизводительную обработку данных для одной прикладной задачи, как многопрограммная машина, выполняющая множество задач параллельно, и как некоторая комбинация этих возможностей.

В этом случае выделяют следующие 4 класса систем:

- системы с симметричной мультипроцессорной обработкой (symmeTRic multIProcessINg), или SMP-системы;
- системы, построенные по технологии неоднородного доступа к памяти (non-un IForm memory access), или NUMA-системы;
- кластеры;
- системы вычислений с массовым параллелизмом (massively parallel processor), или MPP-системы.

Самым высоким уровнем интеграции ресурсов обладает система с симметричной мультипроцессорной обработкой, или SMP-система (рис.1).

В этой архитектуре все процессоры имеют равноправный доступ ко всему пространству оперативной памяти и ввода/вывода. Поэтому SMP-архитектура называется симметричной. Ее интерфейсы доступа к пространству ввода/вывода и ОП, система управления кэш-памятью, системное ПО и т. п. построены таким образом, чтобы обеспечить согласованный доступ к разделяемым ресурсам. Соответствующие механизмы блокировки заложены и в шинном интерфейсе, и в компонентах операционной системы, и при построении кэша.

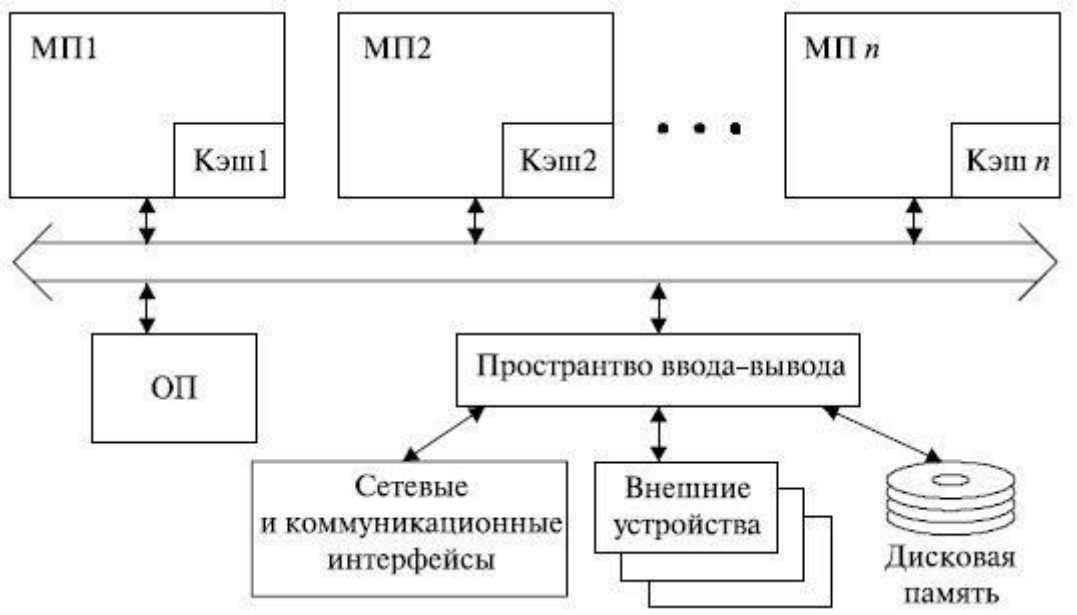

Рис. 1. Система с симметричной мультипроцессорной обработкой

С точки зрения прикладной задачи, SMP-система представляет собой единый вычислительный комплекс с вычислительными ресурсами, пропорциональными количеству процессоров. Распараллеливание вычислений обеспечивается операционной системой, установленной на одном из процессоров. Вся система работает под управлением единой ОС.

ОС автоматически в процессе работы распределяет процессы по процессорным ядрам, оптимизируя использование ресурсов. Ядра задействуются равномерно, и прикладные программы могут выполняться параллельно на всем множестве ядер. При этом достигается максимальное быстродействие системы. Важно, что для синхронизации приложений вместо сложных механизмов и протоколов межпроцессорной коммуникации применяются стандартные функции ОС. Таким образом, проще реализовать проекты с распараллеливанием программных потоков. Общая для совокупности ядер ОС позволяет с помощью служебных инструментов собирать статистику, единую для всей архитектуры. Соответственно, можно облегчить отладку и оптимизацию приложений на этапе разработки или масштабирования для других форм многопроцессорной обработки.

В общем случае приложение, написанное для однопроцессорной системы, не требует модификации при его переносе в мультипроцессорную среду. Однако для оптимальной работы программы или частей ОС они переписываются специально для работы в мультипроцессорной

среде.

Сравнительно небольшое количество процессоров в таких машинах позволяет иметь одну централизованную общую память и объединить процессоры и память с помощью одной шины.

Сдерживающим фактором в подобных системах является пропускная способность магистрали, что приводит к их плохой масштабируемости. Причиной этого является то, что в каждый момент времени 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 мешать друг другу. Когда произойдет такой конфликт, зависит от скорости связи и от количества вычислительных элементов. Кроме того, системная шина имеет ограниченное число слотов. Все это очевидно препятствует увеличению производительности при увеличении числа процессоров. В реальных системах можно задействовать не более 32 процессоров.

В современных микропроцессорах поддержка построения мультипроцессорной системы закладывается на уровне аппаратной реализации МП, что делает многопроцессорные системы сравнительно недорогими.

Сегодня SMP широко применяют в многопроцессорных суперкомпьютерах и серверных приложениях. Однако если необходимо детерминированное исполнение программ в реальном масштабе времени, например, при визуализации мультимедийных данных, возможности сугубосимметричной обработки весьма ограничены. Может возникнуть ситуация, когда приложения, выполняемые на различных ядрах, обращаются к одному ресурсу ОС. В этом случае доступ получит только одно из ядер.

Остальные будут простаивать до высвобождения критической области.

Естественно, при этом резко снижается производительность приложений реального времени.

Исчерпание производительности системной шины в SMP-системах при доступе большого числа процессоров к общему пространству оперативной памяти и принципиальные ограничения шинной технологии стали причиной сдерживания роста производительности SMP-систем. На данный момент эта проблема получила два решения. Первое - замена системной шины на высокопроизводительный коммутатор, обеспечивающий одновременный неблокирующий доступ к различным участкам памяти. Второе решение предлагает технология NUMA.

Система, построенная по технологии NUMA, представляет собой набор узлов, каждый из которых, по сути, является функционально законченным однопроцессорным или SMP-компьютером. Каждый имеет свое локальное пространство оперативной памяти и ввода/вывода. Но с помощью специальной логики каждый имеет доступ к пространству оперативной памяти и ввода/вывода любого другого узла (рис. 2). Физически отдельные устройства памяти могут адресоваться как логически единое адресное пространство - это означает, что любой процессор может выполнять обращения к любым ячейкам памяти, в предположении, что он имеет соответствующие права доступа. Поэтому иногда такие системы называются системами с распределенной разделяемой памятью (DSM - disTRibuted shared memory).

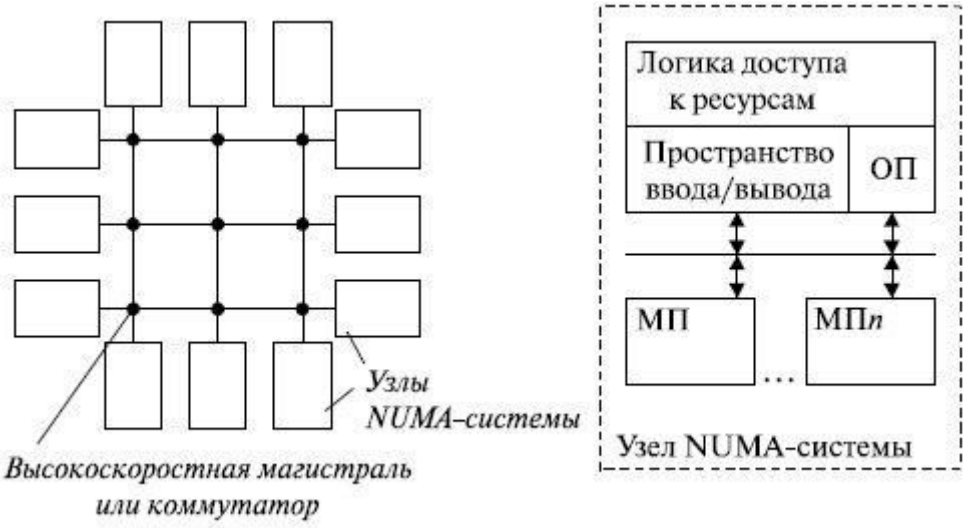

Рис. 2. Система, построенная по технологии неоднородного доступа к памяти

При такой организации память каждого узла системы имеет свою адресацию в адресном пространстве всей системы. Логика доступа к ресурсам определяет, к памяти какого узла относится выработанный процессором адрес. Если он не принадлежит памяти данного узла, организуется обращение к другому узлу согласно заложенной в логике доступа карте адресов. При этом доступ к локальной памяти осуществляется в несколько раз быстрее, чем к удаленной.

При использовании наиболее распространенного сейчас варианта cc-NUMA (cache-coherent NUMA - неоднородный доступ к памяти с согласованием содержимого кэш-памяти) обеспечивается кэширование данных оперативной памяти других узлов.

Обычно вся система работает под управлением единой ОС, как в SMP. Возможны также варианты динамического разделения системы, когда отдельные разделы системы работают под управлением разных ОС.

Довольно большое время доступа к оперативной памяти соседних узлов по сравнению с доступом к ОП своего узла в NUMA-системах на настоящий момент делает такое использование не вполне оптимальным.

Так что полной функциональностью SMP-систем NUMA-компьютеры на сегодняшний день не обладают. Однако среди систем общего назначения NUMA-системы имеют один из наиболее высоких показателей по масштабируемости и, соответственно, по производительности. На сегодня максимальное число процессоров в cc-NUMA-системах может превышать 1000 (серия OrigIN3000). Один из наиболее производительных суперкомпьютеров - Tera 10 - имеет производительностью 60 Тфлопс и состоит из 544 SMP-узлов, в каждом из которых находится от 8 до 16 процессоров Itanium 2.

Следующим уровнем в иерархии параллельных систем являются комплексы, также состоящие из отдельных машин, но лишь частично разделяющие некоторые ресурсы. Речь идет о кластерах.

Кластер представляет собой систему из нескольких компьютеров (в большинстве случаев серийно выпускаемых), имеющих общий разделяемый ресурс для хранения совместно обрабатываемых данных (обычно набор дисков или дисковых массивов) и объединенных высокоскоростной магистралью (рис. 3).

В кластерной системе некоторое распределенное приложение параллельно на нескольких узлах обрабатывает общий набор данных, как правило, таким образом, чтобы у пользователя возникла иллюзия работы на одной машине.

Обычно в кластерных системах не обеспечивается единая операционная среда для работы общего набора приложений на всех узлах кластера. То есть каждый компьютер кластера - это автономная система с отдельным экземпляром ОС и своими, принадлежащими только ей системными ресурсами: набором заведенных пользователей, системными буферами,

областью свопинга и т. п. Приложение, запущенное на нем, может видеть только общие диски или отдельные участки памяти. На узлах кластера работают специально написанные для такой конфигурации приложения, параллельно обрабатывающие общий набор данных. На каждой из машин они представлены рядом процессов, программ, взаимодействующих с помощью кластерного программного обеспечения. Таким образом, кластерное ПО - это лишь средство для взаимодействия узлов и синхронизации доступа к общим данным. Кластер как параллельная система формируется на прикладном уровне, а не на уровне операционной системы.

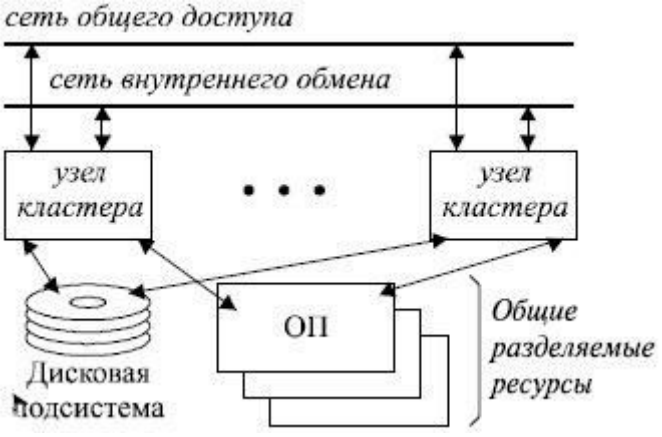

Рис. 3. Структура кластера

В настоящее время такие системы имеют две основные области применения: параллельные серверы баз данных и высоконадежные вычислительные комплексы. Рынок параллельных СУБД и есть фактически рынок кластеров приложений. Высоконадежные комплексы представляют собой группу узлов, на которых независимо друг от друга выполняются некоторые важные приложения, требующие постоянной, непрерывной работы. То есть в такой системе на аппаратном уровне фактически поддерживается основной механизм повышения надежности - резервирование. Причем узлы находятся в так называемом "горячем" резерве, и каждый из них в любой момент готов продолжить вычисления при выходе из строя какого-либо узла. При этом все приложения с отказавшего узла автоматически переносятся на другие машины комплекса. Такая система также формально является кластером, хотя в ней отсутствует параллельная обработка общих данных. Эти данные обычно монопольно используются выполняемыми в рамках кластера приложениями и должны быть доступны для всех узлов.

Если в кластере его узлы разделяют некоторые ресурсы, то параллельные системы другого класса - системы вычислений с массовым параллелизмом (MPP) - строятся из отдельных полностью независимых компьютеров, соединенных только высокоскоростной магистралью или коммуникационными каналами (рис.4). Это могут быть либо просто несколько серийно выпускаемых UNIX-машин, соединенных с помощью высокопроизводительной сетевой среды, либо специально сконструированная система из отдельных функциональных блоков, объединенных коммутатором.

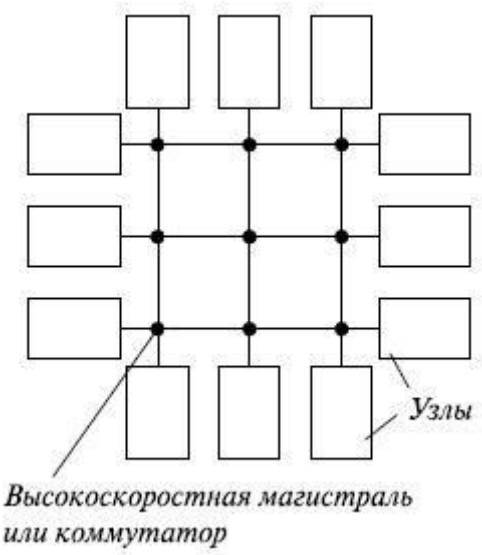

Рис. 4. Структура MPP-системы

В такой системе адресное пространство состоит из отдельных адресных пространств, которые логически не связаны между собой и доступ, к которым не может быть осуществлен аппаратно другим процессором.

При этом для обмена данными используется механизм передачи сообщений между процессорами. Поэтому эти машины часто называют машинами с передачей сообщений. Пользователь может определить логический номер процессора, к которому он подключен, и организовать обмен сообщениями с другими процессорами.

Программирование в такой системе - достаточно сложная задача.

Она требует специального инструментария и особого системного программного обеспечения для работы параллельных приложений, которые ориентированы на функционирование параллельных процессов, распределенных по узлам MPP-системы, с обменом сообщениями между ними.

Повышение производительности машин с массовым параллелизмом путем увеличения в них числа процессоров имеет определенные ограничения. Чем большее число процессоров входит в состав MPP-системы, тем длиннее каналы передачи управления и данных, а значит, и тем меньше тактовая частота. Происшедшее возрастание нормы массивности для больших машин до 512 и даже 64К процессоров обусловлено не ростом размеров машины, а увеличением степени интеграции схем, позволившей за последние годы резко повысить плотность размещения элементов в устройствах. Топология сети межпроцессорного обмена в такого рода системах может быть различной.

Главным преимуществом MPP-систем является их хорошая масштабируемость: в отличие от SMP-систем, здесь 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.

Основными недостатками систем данного типа являются следующие:

- 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 задействовать системные ресурсы, следствием чего является высокая цена программного обеспечения для MPP-систем с раздельной памятью.

Подведем некоторые итоги, касающиеся областей применимости систем параллельной обработки данных различных типов.

SMP-системы потенциально обладают достаточными возможностями для обеспечения необходимой для большинства применений производительности: вполне естественно увеличивать число процессоров, а не ставить рядом еще один компьютер. Добавление одного процессора гарантированно увеличивает производительность, а добавление, например, узла в кластер адекватного ускорения не даст. Более того, в некоторых случаях общая производительность системы может даже упасть, когда узлы кластера начинают активно конкурировать за доступ к общим ресурсам, и взаимные блокировки сводят на нет преимущества параллельной обработки.

NUMA-системы создаются для вполне определенных целей - обеспечения масштабных расчетов. Системы, использующие эту архитектуру, прежде всего применяются для уникальных высококачественных и высокопроизводительных прикладных программ, требующих более восьми процессоров. Однако они имеют высокую стоимость и требуют уникального ПО (прикладные программы и ОС).

Для современных систем помимо вполне традиционных требований по производительности, масштабируемости, цене дополнительные высокие требования предъявляются к надежности их работы. Именно по этим соображениям вычислительные комплексы на основе кластеров или МРР-машин завоевывают все большую популярность.

МРР-системы обладают рядом преимуществ, главным из которых является лучшая среди всех рассмотренных архитектур масштабируемость. Именно поэтому МРР-компьютеры обычно используются при больших ресурсоемких вычислениях. Конечно, они применяются и при построении больших баз данных, и в отказоустойчивых вычислительных комплексах. Но здесь их использование довольно ограничено. Это отчасти связано с тем, что они все-таки дороже кластеров и имеют достаточно большую начальную цену. Кластер же можно построить из относительно дешевых машин произвольной конфигурации.

Приведенная классификация систем параллельной обработки данных достаточно условна. Разработчики вычислительных систем не проектируют машину какого-то специального класса, а стараются создать более производительную архитектуру. Кроме этого, сам пользователь может с использованием стандартных компонентов спроектировать комплекс, архитектурно и функционально наиболее подходящий для решения конкретной задачи.

Транспьютеры. Для построения многопроцессорных систем могут быть использованы специально разработанные процессоры, называемые транспьютерами. Они были созданы в середине 1980-х годов фирмой INMOS Ltd (ныне - подразделение STMicroelecTRoniCS).

Транспьютер – это микропроцессор со встроенными средствами межпроцессорной коммуникации, предназначенной для построения многопроцессорных систем. Его название происходит от слов TRansfer (передатчик) и computer (вычислитель).

Транспьютер включает в себя средства для выполнения вычислений (ЦП, АЛУ с плавающей точкой, внутрикристальную память) и 4 канала для связи (линка) с другими транспьютерами и/или другими устройствами. Каждый линк представляет собой 2 однонаправленных последовательных канала передачи информации. Встроенный интерфейс позволяет подключать внешнюю память емкостью до 4 Гбайт (рис. 5).

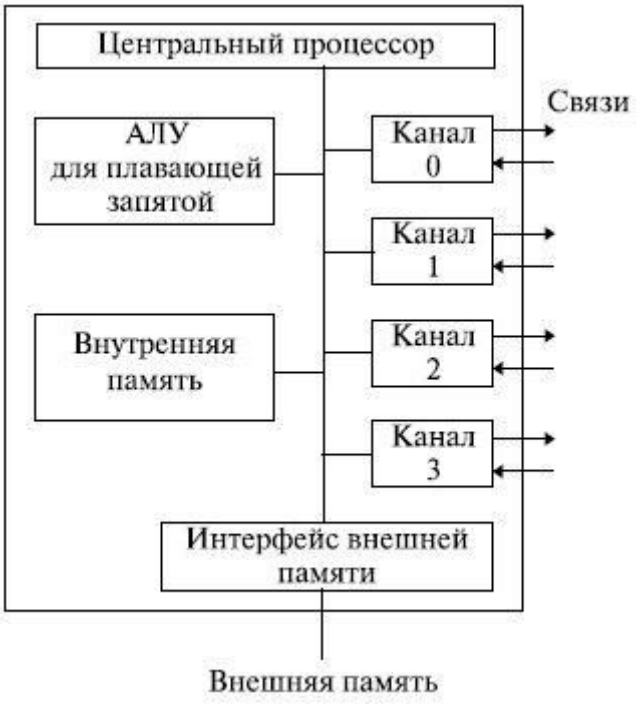

Рис. 5. Структура транспьютера

Многопроцессорная система может создаваться из набора транспьютеров, которые функционируют независимо и взаимодействуют через последовательные каналы связи (рис. 6).

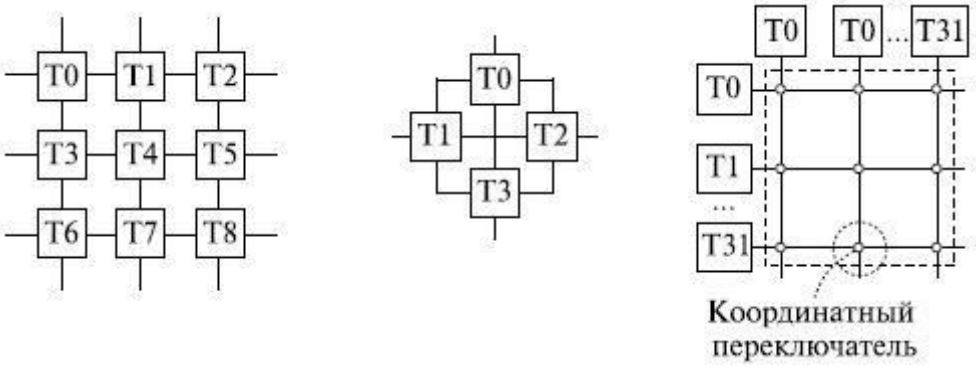

Рис. 6. Структуры многопроцессорных систем на базе транспьютеров

При передаче данных в *линк* процесс должен исполнить команду вывода. Процесс, исполнивший такую команду, задерживается до тех пор, пока все данные не будут переданы. Аналогично, при приеме данных из линка процесс должен исполнить команду ввода. При исполнении такой команды процесс блокируется до тех пор, пока *буфер* не будет заполнен данными. Взаимодействие с внешним устройством через *линк* позволяет транспьютеру синхронизовать свою *деятельность* с этими устройствами без использования механизма прерываний.

Использование такого подхода позволило организовать виртуальные каналы связи между процессами, которые могли размещаться как на единственном транспьютере, так и на нескольких транспьютерах, и виртуальные линки между процессами. Любой *транспьютер* может одновременно образовывать любое число *параллельных процессов*. Он имеет специальный *планировщик*, который производит распределение процессорного времени между этими процессами. Тем самым появляется возможность, имея всего лишь один *транспьютер*, написать параллельную программу, которая полностью выполняется на нем. Задача разбивается на ряд процессов, и все эти процессы параллельно протекают внутри одного транспьютера, периодически останавливаясь для получения данных друг от друга. Систему можно расширить другими транспьютерами и перенести на них ряд процессов. При этом нужно просто переопределить таблицу связей процессов,

вычислительная *мощность* системы, естественно, увеличивается.

Помимо интересных возможностей, связанных с построением мультипроцессорных систем