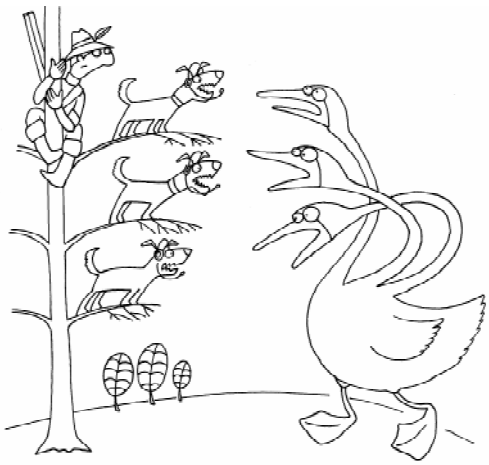
Демьянович Юрий Казимирович

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ УРОК 5. ОБ АРХИТЕКТУРЕ ПАРАЛЛЕЛЬНЫХ СИСТЕМ. КОММУНИКАЦИОННЫЕ СРЕДЫ

В предыдущих статьях (см. [1–4]) рассматривались вопросы распараллеливания: мы выяснили, что такое параллельная форма, какие имеются средства распараллеливания, как написать элементарные параллельные программы, какие имеются проблемы при распараллеливании, а также ознакомились с интерфейсом MPI, который позволяет распараллелить последовательную программу, написанную на алгоритмических языках Fortran, C и C++.

Напомним, что проблема распараллеливания вычислений появилась потому, что потребовалось решать суперсложные задачи, а их решение возможно лишь на вычислительных системах (ВС) с огромным быстродействием. При современных технологиях существенное увеличение скорости процессора невозможно, поэтому единственный способ достичь большого быстродействия – создать ВС с параллельно работающими процессорами (или ядрами) в надежде на то, что решаемую исходную суперсложную задачу удастся разбить на группы независимых подзадач, которые можно решать одновременно. Поскольку подзадачи являются частью исходной задачи, то они должны временами обмениваться информацией. Такой обмен возлагается на комплекс устройств, состоящий из линий связи (каналов), переключателей (коммутаторов), преобразователей (адаптеров) и других устройств, работающих по специальным алгоритмам (программам), часто внедренным («запаянным») в эти устройства. Этот комплекс (вместе с упомянутыми программами) называется коммуникационной средой.



...решаемую исходную суперсложную задачу удастся разбить на группы независимых подзадач, которые можно решать одновременно.

Класс решаемых задач предъявляет определенные требования к общей архитектуре ВС: к способу соединения вычислительных модулей (ВМ), к их относительному расположению, к их разрядности и т. п. Например, климатическая задача решается в относительно тонком приземном сферическом слое воздуха (тропосфере). Для эффективного ее решения хорошо было бы разместить вычислительные модули на поверхности сферы и соединить соседние (так называемая сферическая схема): ввиду локальных свойств решаемой задачи, в этом случае затраты на передачу информации, по-видимому, были бы минимальными; похожий

эффект получится, если имитировать сферическую схему лишь соответствующими соединениями этих ВМ (без реального расположения их на сфере).

В различных случаях под архитектурой ВС могут подразумеваться различные вещи: это и разрядность, с которой работает система (говорят о 32-х или о 64-х разрядной архитектуре), это и способ соединения отдельных ее частей, то есть ее топология (говорят о топологии кольца, о топологии звезды, о топологии решетки и т. д.). Несомненно, все эти аспекты связаны друг с другом: важнейшим связующим звеном является коммуникационная среда. Коммуникационная среда работает на несколько порядков медленнее процессоров. Процессоры миниатюрны, передачи информации внутри процессоров почти мгновенны, а каналы связи протяженные, они связывают различные процессоры (или ВМ), и передачи по каналам связи резко замедляют работу вычислительной системы. Основная задача при организации вычислений состоит в том, чтобы уменьшить количество передач по каналам связи и не задерживать работу процессоров во время таких передач. Важнейшую роль в решении этой задачи играет выбор коммуникационной среды.

Коммуникационные среды могут быть устроены по-разному. Для эффективного использования вычислительных систем при решении различных классов задач разрабатываются различные типы коммуникационных сред, удовлетворяющих определенным стандартам. В этой статье читатель сможет получить общее представление о коммуникационных средах и об их свойствах; здесь упомянуты лишь некоторые из таких сред. Подробное описание коммуникационных сред можно найти в монографии [5].

1. НЕКОТОРЫЕ ВСПОМОГАТЕЛЬНЫЕ ПОНЯТИЯ

При реализации мощной параллельной ВС используется большое число вычислительных модулей, представляющих собой процессоры (иногда многоядерные) с подключенной к ним памятью того или иного объема; эти ВМ соединяются каналами связи, то есть связующими устройствами (линками), содержащими соответствующее программное обеспечение. Иногда эти ВМ называют *узлами* и *точками* рассматриваемой ВС, а средства связи между ними – *коммуникационной средой*; все вместе часто называют *сетью* (таким образом, рассматриваемую параллельную ВС также можно считать сетью, хотя чаще под сетью подразумевают соединение слабо взаимодействующих между собой компьютеров).

Для создания коммуникационных сред используются стандартные элементы. Перечислим некоторые из них.

Контроллер – специализированный процессор, автоматически управляющий работой подключенных к нему устройств. Он является частью каналов связи, форматирует данные для передачи по каналу связи или для записи в память.

Адаптер представляет собой электронную схему, преобразующую один способ представления данных в другой. Он используется для согласования различных устройств и часто включается в состав устройств сопряжения различных компонент (иногда его называют сетевым интерфейсным контроллером).

Коммутатор является сложным переключателем и служит для соединения подведенных к нему входов и выходов (называемых иногда *полюсами*). Это обычно весьма сложное устройство, содержащее электрические и механические схемы. Коммутаторы с большим числом входов и выходов в одном каскаде технически сложно реализовать, поэтому вводят коммутаторы с несколькими каскадами (называемые *составными коммутаторами*). Каждый каскад составлен из *простых коммутаторов*, например, из коммутаторов с двумя входами и двумя выходами. Составной коммутатор технически проще реализовать, чем однокаскадный с тем же количеством входов и выходов. Недостатком такого коммутатора является задержка соединения, пропорциональная числу каскадов.

Маршрутизатор проще коммутатора, он служит для определения маршрута пе-

редачи данных между узлами сети. Маршрутизатор реализуется программно, а иногда и с использованием дополнительной аппаратуры.

Часто используется термин *интегральная схема* или эквивалентный ему термин *чип*; это – миниатюрная электронная схема, выполненная на поверхности или внутри полупроводникового кристалла. Часто чип содержит миллионы электрических элементов и позволяет хранить или передавать информацию большого объема.

Иногда применяют термин *мост* по отношению к совокупности технических и программных средств, позволяющих осуществить передачу информации между двумя сетями или двумя вычислительными системами без существенного преобразования информации. В ходу также термины *мост-адаптер* или *мост-коммутатор* по отношению к более сложным устройствам, реализующим обе функции: моста и адаптера или моста и коммутатора соответственно.

Совокупность правил, по которым структурируется и кодируется информация, и по которым производится ее передача, называется *протоколом*.

Память ВМ иногда имеет постраничную организацию: она распадается на фрагменты фиксированной длины, передаваемые как единое целое при обменах информации между устройствами; такие фрагменты называются *страницами*, а адресация часто называется *виртуальной*. Иногда выделяются страницы памяти, к которым имеют доступ все ВМ; такие страницы называются *общими* или *разделяемыми*.

Для эффективной работы ВС важно, чтобы быстродействие процессора согласовывалось со скоростью работы памяти...

Заметим еще, что сеть компьютеров часто называется *кластером* или *кластерной системой*.

Шинная структура, или просто *шина*, – часто используемая коммуникационная среда; в каждый момент времени на шине может передавать *только одно устройство*, а «слушать» – *все подключенные* к шине устройства.

Приведенной информации достаточно для краткого описания коммуникационных сред, которое дается в следующих пунктах.

2. ОДНОПРОЦЕССОРНАЯ СИСТЕМА. ПОНЯТИЕ КОГЕРЕНТНОСТИ

Сначала рассмотрим работу однопроцессорной ВС.

Для эффективной работы ВС важно, чтобы быстродействие процессора согласовывалось со скоростью работы памяти (то есть со скоростью извлечения информации из памяти и со скоростью ее погружения в память). Идеальная память должна обеспечивать процессор командами и данными непрерывно с тем, чтобы не было простоев процессора. Кроме того, память должна иметь большую емкость, для того чтобы обеспечивать обработку необходимых заданий.

Это достигается использованием многоуровневой памяти с соответствующей иерархией; иногда говорят об иерархии памятей (во множественном числе).

Время доступа к данным зависит от объема и типа используемой памяти. Малое время доступа характерно для памяти малого объема; поэтому пересылают данные из основной памяти в малую быструю буферную память, обрабатывают их с использованием этой памяти и результат отправляют обратно в основную память. Малую быструю буферную память называют *кэш-память*, или просто *кэш (cache memory)*.

Создание иерархической многоуровневой памяти, пересылающей блоки программ и данных между уровнями памяти за то время, пока другие блоки обрабатываются процессором, позволяет существенно снизить простои процессора в ожидании данных (см. схему на рис. 1).

Такая структура памяти эффективна для решения задач *локальными алгоритмами* (иногда в этом случае говорят об *алгоритмах с локальными данными*).

Что такое локальный алгоритм? Локальным алгоритмом называется такой алгоритм решения задачи, для которого выполнены два признака:

1) (наиболее важный признак) процессор *многократно* использует выбранные из основной памяти данные для получения результата до того момента, когда результат будет отправлен в основную память;

2) положение выбираемых данных *локализовано* в основной памяти (они выбираются «подряд», см. рис. 2).

Описанная ситуация часто складывается в научных и инженерных расчетах, например при решении уравнений математической физики, ибо эти уравнения обычно выводятся из физических процессов, подчиненных локальным законам (примененным к малому элементу объема, массы и т. п.).

В этом случае, переходя от одного элемента данных к другому, приходится обрабатывать небольшие порции данных с большим количеством вложенных циклов, так что каждое данное из обрабатываемой порции используется по многу раз.

Подводя некоторый итог сказанному, видим, что в погоне за скоростью решения задачи идут на создание многих экземпляров одной и той же информации: исходный экземпляр информации находится в основной памяти, а остальные – в кэшах нашей вычислительной системы. В начальный момент все эти экземпляры тождественны, то есть являются копиями. В динамике вычислений один из экземпляров информации может быть изменен процессором, что, вообще говоря, требует соответствующего изменения остальных «копий» (кавычки показывают, что одинаковость экземпляров уже не гарантируется). Однако сразу вносить эти изменения бывает невыгодно, ибо внесение изменений задерживает вычислительный процесс. Поэтому часто в этот момент лишь посылается сигнал (это не требует много времени), который придает этим «копиям» признак *несостоятельности* (этот признак является защитой от случайного использования этих копий).

Конечно, в определенный момент (автоматически или по указанию пользователя) устанавливается тождественность всех «копий» заменой несостоятельных «копий» на измененный экземпляр информации, упомянутый выше. Тождественные «копии» называют *когерентными*, а процесс установления их тождественности – *процессом установления когерентности копий*.

В связи с тем, что локально обрабатываемые данные могут возникать в динамике вычислений и не быть сконцентрированными в одной области основной памяти при статическом размещении, буферную память

$$\left\{\begin{array}{l}\text{эффект}\\ \text{уменьшения}\\ \text{времени доступа}\end{array}\right\} = \frac{\left\{\begin{array}{l}\text{время обработки}\\ \text{в кэш – памяти}\end{array}\right\}}{\left\{\begin{array}{l}\text{время пересылки}\\ \text{из кэш – памяти}\\ \text{в основную память}\end{array}\right\}} \rightarrow \left\{\begin{array}{l}\text{требуется}\\ \text{"локальность" данных}\end{array}\right\}$$

Рис. 1. Уменьшение простоев процессора

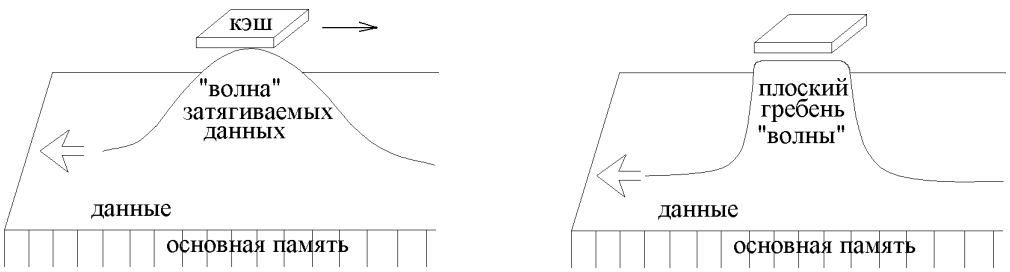


Рис. 2. Эффект локализации данных

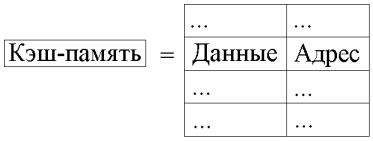


Рис. 3. Ассоциативный кэш

организуют как *ассоциативную* (в ассоциативной памяти данные содержатся в совокупности с их адресами, которые эти данные имеют в основной памяти (рис. 3)).

Типовая современная иерархия памятей для однопроцессорных ВС имеет следующую структуру (рис. 4):

1) регистры 64–256 слов со временем доступа 1 такт,

2) кэш I уровня, 8 Килослов, время доступа 1–2 такта,

3) кэш II уровня, 256 Килослов, время доступа 3–5 тактов,

4) основная память (до 4 Гигаслов), время доступа 12–55 тактов.

Приводимые данные не следует абсолютизировать, ибо непрерывный прогресс технологий может их постоянно изменять.

Кэш обычно содержит совокупность строк, которые состоят из фиксированного числа единиц памяти (битов, слов); типичный размер строки 16, 32, 64, 128 или 256 битов.

Имеется три типа кэш-памяти:

– с прямым обращением (direct-mapped cache);

– частично ассоциативная (self-associative cache);

– ассоциативная (fully associative cache).

Работа с кэш-памятью состоит в том, что если нет нужной строки, то одна из старых строк заменяется на требуемую.

При замене может использоваться один из принципов:

1) строки заменяются циклически;

2) заменяется наиболее редко используемая строка;

3) заменяется та строка, к которой давно не было обращений (рис. 5).

Поскольку в динамике вычислений данные могут быть изменены процессором, то требуются изменения и в основной памяти. Имеются следующие способы внесения изменений:

1) запись в основную память происходит сразу после возникновения изменений в кэше (write-through), в это время процессор простаивает;

2) запись в основную память происходит только в момент вытеснения строки из кэша (write-back); если адрес, в который необходимо произвести запись, находится в кэше, то запись идет только в кэш, иначе – сразу в основную память;

3) промежуточные варианты (buffed write through): запросы на изменение в основной памяти буферизуются и не задерживают процессор на время проведения записи в память.

Контроллер кэша отслеживает адреса памяти, выдаваемые процессором; если при чтении адрес соответствует данным, содержащимся в одной из строк кэша, то отмечается «попадание в кэш», и данные из кэша направляются в процессор. Если данных в кэше не оказывается, то отмечается «промах», и требуемая строка доставляется в кэш (рис. 6).

Заметим, что при вводе данные попадают сразу в основную память, и потому постоянно необходимо отслеживать обновление данных через устройства ввода и поддерживать когерентность имеющихся «копий» (рис. 7)

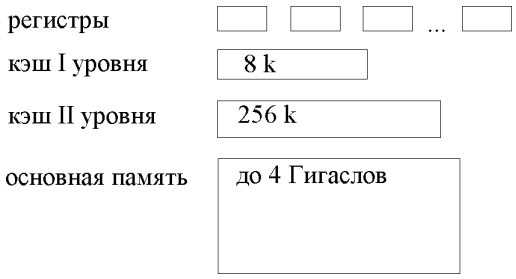


Рис. 4. Структура памяти

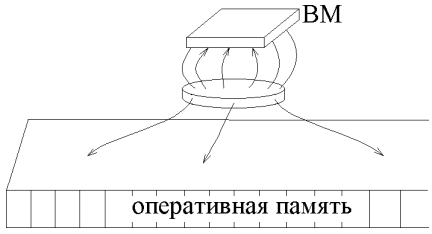


Рис. 5. Взаимодействие с кэшем

Рис. 6. Попадание в кэш

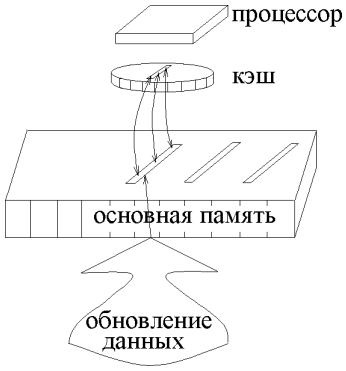


Рис. 7. Обновление данных

3. МНОГОПРОЦЕССОРНАЯ СИСТЕМА. ОБЩАЯ ПАМЯТЬ

В многопроцессорных системах идентичность данных в кэшах вычислительных модулей (то есть когерентность кэшей) поддерживается с помощью *межмодульных пересылок*. Одним из алгоритмов подобного рода является алгоритм поддержки когерентности кэшей MESI (Modified, Exclusive, Shared, Invalid).

Предположим, что:

1) каждый ВМ имеет *собственную* (то есть локальную) кэш-память,

2) имеется *общая* (то есть разделяемая) область основной памяти,

3) все ВМ подключены к основной памяти посредством шины,

4) к шине подключены внешние устройства (рис. 8).

В соответствии с алгоритмом MESI, каждая строка кэш-памяти может находиться в одном из состояний:

M – строка *модифицирована* (то есть доступна для чтения и записи только этим ВМ);

E – строка *монопольно кэширована* (то есть доступна для чтения и записи в этом ВМ и в основной памяти);

S – строка, *множественно копированная и разделяемая* (то есть доступна для чтения и записи в этом ВМ, а также в основной памяти и в кэшах других ВМ);

I – строка *несостоятельная* (то есть эту строку использовать нельзя).

Состояние строки применяется:

1) для определения процессором ВМ возможности локального (без выхода на шину) доступа к данным в кэше;

2) для управления механизмом когерентности.

Промах чтения в кэш приводит к необходимости:

– вызвать строку из основной памяти;

– придать ей состояние E или S.

Промах записи в кэш приводит к тому, что строка

– помещается в кэш;

– передается в основную память при предоставлении шины.

Если при чтении окажется, что строка в кэше не состоятельна (то есть находится в состоянии I), то произойдет копирование из основной памяти и размещение в кэше с состоянием E или S.

Если окажется, что строка не состоятельна при записи, то будет изменено содержа-

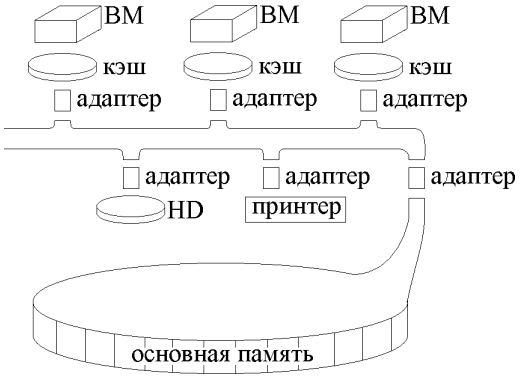


Рис. 8. Структура ВС с общей памятью

При записи данных в кэш-память процессор приостанавливается...

ние в основной памяти, но не в кэше (в кэше сохраняется состояние I).

Для *поддержания когерентности* строк в кэш-памяти при операциях ввода/вывода и при обращениях в основную память других процессоров на шине генерируются *специальные циклы опроса* состояния кэш-памятей. Эти циклы по указанному в операции адресу находят требуемую строку в кэшах других ВМ.

4. МНОГОПРОЦЕССОРНАЯ СИСТЕМА. РАСПРЕДЕЛЕННАЯ ПАМЯТЬ

Имеется два основных подхода к поддержанию когерентности в случае распределенной памяти.

Первый подход, который называют *прямым подходом*, заключается в том, что при *каждом промахе* в кэш в *любом процессоре* инициализируется запрос требуемой строки из того блока памяти, в котором она размещена (рис. 9).

Блок памяти, в котором расположена строка, называется *резидентным блоком* для этой строки.

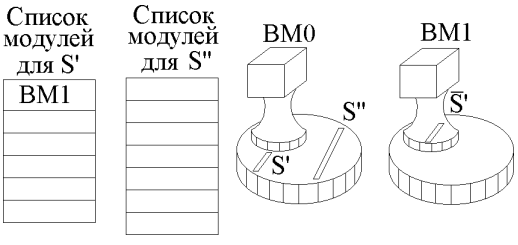


Рис. 10. Строки в блоках памяти

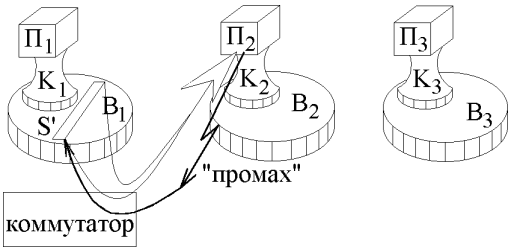


Рис. 9. Поддержание когерентности при прямом подходе

Запрос передается через коммутатор в ВМ с резидентным для данной строки блоком, строка через коммутатор пересылается в ВМ, где произошел промах.

Эта же схема используется при начальном заполнении кэшей.

В модуле, резидентном для данной строки, ведется список модулей, в кэшах которых эта строка размещена (рис. 10, 11).

Когерентность кэшей обеспечивается следующим образом.

При записи данных в кэш-память процессор приостанавливается и выполняет следующие действия:

1) измененная строка кэша копируется в память резидентного модуля;

2) если строка была разделяемой, то она копируется из резидентной памяти во все кэши всех модулей, указанных в списке для этой строки;

3) после получения подтверждения о том, что все кэши изменены, резидентный модуль разрешает приостановленному процессору продолжать вычисления (рис. 12).

Изложенный подход редко применяется из-за больших простоев процессоров. На практике применяют более сложные алгоритмы, обеспечивающие мéньшие простои процессоров.

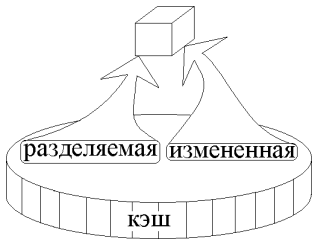


Рис. 11. Обработка строк в кэше

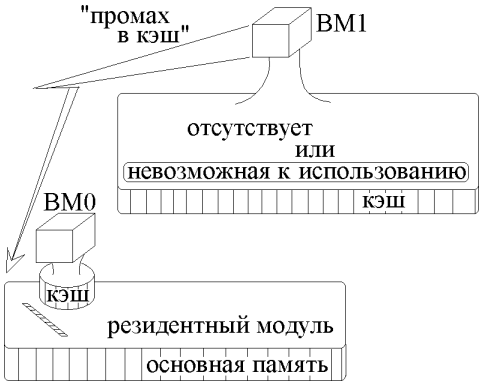


Рис. 12. Промах в кэш

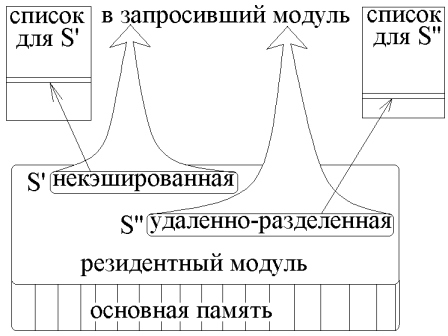


Рис. 13. Доставка строк

Второй – более эффективный подход – проиллюстрируем на примере алгоритма DASH.

При использовании этого алгоритма для каждой строки в резидентном ВМ ведется список модулей, в кэшах которых размещены копии строки.

С каждой строкой в резидентном для нее ВМ связаны три глобальных состояния:

1) «некэшированная» – копия строки не находится в кэше другого ВМ, кроме резидентного;

2) «удаленно-разделенная» – копии строки помещены в других ВМ;

3) «удаленно-измененная» – строка изменена где-либо операцией записи.

Кроме этого, каждая строка кэша любого нерезидентного ВМ находится в одном из трех локальных состояний:

1) «несостоятельная» – если ее использование запрещено;

2) «разделяемая» – если есть неизменная копия, которая, возможно, размещается в других кэшах;

3) «измененная» – если копия изменена операцией записи в данном ВМ (рис. 13).

Опишем действия алгоритма при чтении информации:

1) если состояние читаемой строки «разделяемая» или «измененная», то каждый процессор может читать ее из своего кэша;

2) если строка отсутствует в кэше или находится в состоянии «несостоятельная», то посылается запрос «промах чтения», который направляется в ВМ, резидентный для данной строки;

3) если глобальное состояние строки в резидентном модуле «некэшированная» или «удаленно-разделенная», то копия строки посылается в запросивший модуль и заносится в список этого модуля;

4) если глобальное состояние строки «удаленно-измененная», то запрос «промах чтения» перенаправляется в ВМ, содержащий измененную строку; этот ВМ пересылает требуемую строку в запросивший модуль и в резидентный для нее модуль и устанавливает в последнем состояние «удаленно-разделенная» (рис. 14, 15)

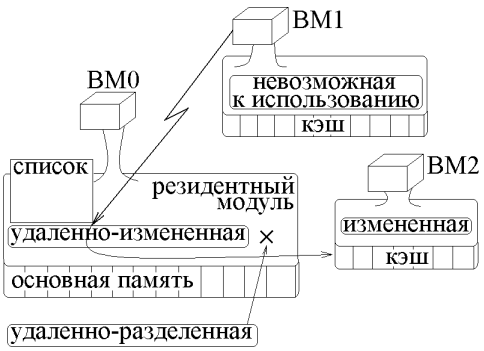


Рис. 14. Чтение: преобразование глобальных состояний

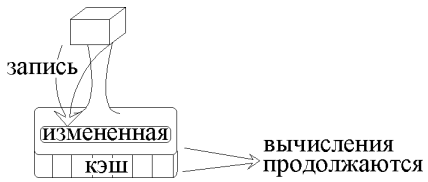


Рис. 15. Запись в свой кэш

...ВМ посылает запрос на захват...

При записи предпринимаются следующие действия:

1) если состояние строки «измененная», то запись выполняется и вычисления продолжаются;

2) если состояние строки «несостоятельная», то ВМ посылает запрос на *захват в исключительное использование* этой строки и приостанавливает операцию записи до получения подтверждения, что все остальные разделяюшие эту строку вычислительные модули перевели ее копии в состояние «несостоятельная»;

3) если имеется «промах в кэш» и глобальное состояние строки в резидентном ВМ «некэшированная», то строка отсылается запросившему модулю, и модуль продолжает приостановленные вычисления, а в резидентном ВМ она становится «удаленно-разделенной»;

4) если глобальное состояние строки «удаленно-разделенная», то резидентный ВМ рассылает (по списку) всем ВМ с копией этой строки запрос на перевод их в состояние «несостоятельная» и посылает подтверждение исполнения в модуль, инициирующий операцию записи; в резидентном ВМ после исполнения записи строка переходит в состояние «удаленно-измененная» (рис. 16, 17).

5. СТРУКТУРА КОММУНИКАЦИОННОЙ СРЕДЫ НА БАЗЕ SCI

Идея коммуникационной среды SCI (Scalable Coherent Interface) – ввести у каждого вычислительного модуля дополнительную стандартную кэш-память, через которую происходит передача информации в сеть и извлечение информации из сети; упомянутая кэш-память вместе с подключенным к ней ВМ называется узлом SCI. Каждый узел имеет входной и выходной каналы.

Коммуникационная среда SCI представляет собой однонаправленное кольцо, по которому циркулируют пакеты, генерируемые подключенными к нему узлами SCI. После генерации пакета узел SCI может вставить его на свободное место в поток циркулирующих пакетов. При получении пакета адресат (узел SCI) генерирует эхо-пакет, направляемый узлу-источнику, с тем чтобы подтвердить получение. Узел-источник сохраняет копию посланного пакета до получения эхо-пакета, а в случае отсутствия такового через определенное время повторяет передачу.

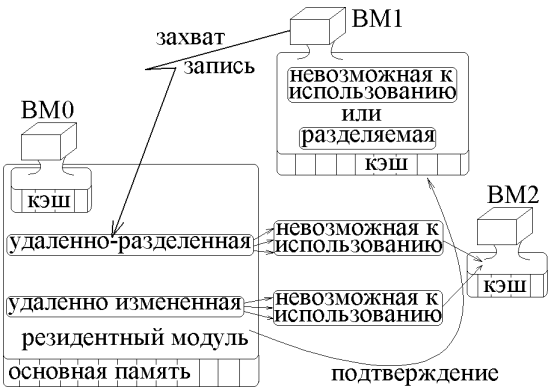


Рис. 16. Преобразование состояний строк при записи

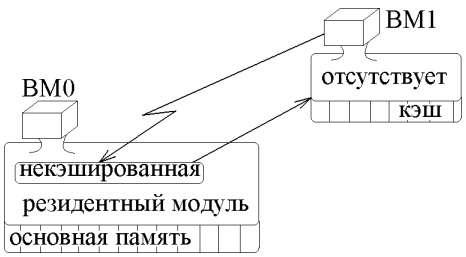


Рис. 17. Преобразование локальных и глобальных состояний

Среди узлов SCI выделен один (и только один) узел, называемый scrubber; ему свойственны следующие функции:

1) при инициализации системы он инициализирует узлы SCI и устанавливает адреса;

2) он управляет таймерами (измерителями времени);

3) метит проходящие (непомеченные) пакеты;

4) уничтожает помеченные пакеты (то есть пакеты, не нашедшие адресата (рис. 18)).

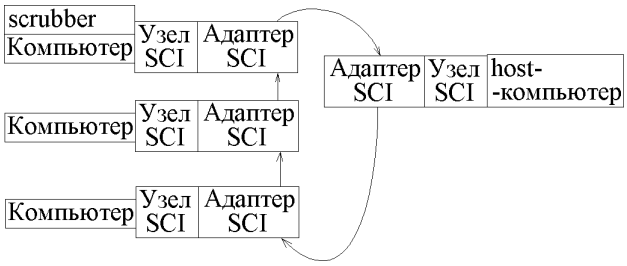


Рис. 18. Маркерное кольцо SCI

Коммуникационная среда SCI допускает построение сложных систем, связанных между собой «агентами». Например, используются *агент-коммутатор* и *агент-мост*, которые связывают различные кольца или обеспечивают связь с коммуникационными средами другого типа (с другим протоколом), например с шинной структурой.

Однонаправленность передач в коммуникационной среде SCI не требует переключения микросхем с приема на передачу и обратно, что занимает время и создает электрические помехи. Заметим также, в кольцевой структуре преодолевается фундаментальное ограничение шин, состоящее в том, что шина единовременно может быть предоставлена только одному устройству.

Коммуникационная среда SCI достаточно надежна; она широко используется для проведения ответственных вычислений.

6. О КОММУНИКАЦИОННОЙ СРЕДЕ MYRINET

Логический протокол коммуникационной среды MYRINET разрабатывался при создании экспериментальной высокоскоростной локальной сети и финансировался министерством обороны США.

Рассматриваемая среда стандартизирует формат пакета, способ адресации вычислительных модулей, набор управляющих символов протокола передачи пакетов.

Коммуникационная среда MYRINET образуется адаптерами «шина компьютера – линк сети» и коммутаторами линков сети (рис. 19). Каждый линк содержит пару однонаправленных каналов. Передаваемые пакеты устроены просто: они состоят из заголовка, данных и концевика (рис. 20).

Поток передачи данных может быть на определенное время остановлен получателем. Если передающий адаптер заблокирован на время бóльшее, чем пересылка 2^{22} байтов (4 Мбайта или 50 миллисекунд при пропускной способности 80 Мбайт/с), то адаптер посылает получателю сигнал сброса.

Если линк подключен к неработоспособному адаптеру, то передача не блокируется, а передаваемые пакеты теряются.

Ввиду сказанного, можно сделать вывод о том, что коммуникационная среда MYRINET обладает мéньшей надежностью по сравнению с SCI; она применяется в проектах, для которых вычисления не являются критичными.

7. ШИНА PCI И ЕЕ СВОЙСТВА

Рассматривая коммуникационные среды нельзя не остановиться на таком замечательном продукте компании Intel, как шина PCI (Peripheral Component Interconnect). Эта шина была создана в 90-е годы прошлого столетия и с тех пор применяется практи-

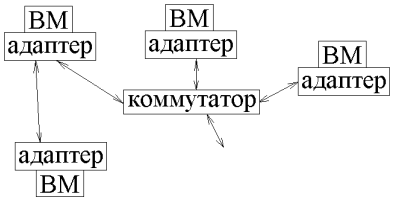


Рис. 19. Структура среды MYRINET

Рис. 20. Пакет в среде MYRINET

Каждый знаком с функцией Plug and Play (включай и играй)...

чески в каждом компьютере. Каждый знаком с функцией Plug and Play (включай и играй), которая состоит в том, что без выключения вычислительной системы (настольного компьютера, ноутбука и т. п.) можно подключить дополнительную память (например «флэшку», дополнительный внешний диск), подключить принтер и т. п.; эта функция поддерживается шиной PCI и совместимым с ней оборудованием.

Шина PCI имеет ряд особенностей, которые выгодно выделяют ее среди остальных шин; к таким особенностям относятся синхронный режим передачи данных с использованием обратной связи, поддержка тактовых частот 33, 66 или 132 МГц, 32-х, 64-х или 128-разрядная архитектура, три адресных пространства (памяти устройств ввода/вывода, конфигурации). Кроме того, шина PCI обладает свойством независимости от

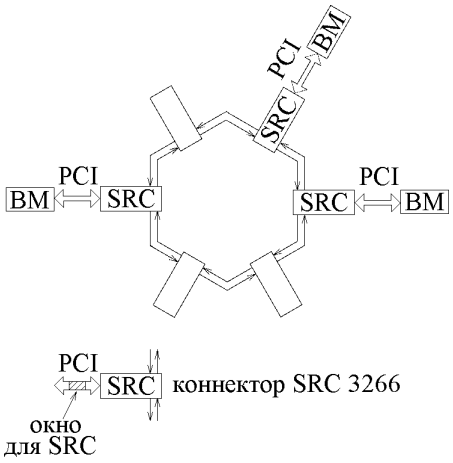


Рис. 21. Двунаправленное кольцо SRC

используемого процессора (от его адресного пространства, системы прерываний, тактовой частоты, формата данных), гарантирует малое время ожидания для устройств реального времени (имеется тот максимум времени ожидания, при котором такому устройству шина обязательно предоставляется), имеет широкие возможности энергосбережения (в том числе может использовать два уровня напряжений 3.3 В и 5 В) и др.

Шина PCI постоянно совершенствуется: повышается частота, увеличивается пропускная способность, улучшаются энергосберегающие свойства и т. д. Без этой шины трудно представить конструкцию современных компьютеров.

8. КОННЕКТОР ШИН PCI (КОННЕКТОР SRC 3266)

Коннектор шин SRC 3266 предназначен для работы с 32-разрядной шиной PCI на частотах не более 66 Мгц. Структура коммуникационной среды, образуемой объединением кристаллов SRC 3266, представляет собой двунаправленное кольцо (рис. 21).

Линки, связывающие SRC-кристаллы, имеют ширину 16 битов (то есть число проводников равно 16, и каждый из них находится в «состоянии» 0 или 1) и частоту 266 Мгц, так что пиковая производительность равна 532 Мбайт/с (из-за ширины: последняя равняется 2 байтам). Коннектор SRC допускает длину линка 1–3 метров. При взаимодействии двух шин PCI формируется виртуальный PCI-PCI мост. На частоте 33 МГц может функционировать до 50 таких мостов (одновременно) без снижения пропускной способности.

Подключенные к шинам PCI устройства (BM) прозрачным образом обмениваются пакетами, переносящими их между шинами PCI по двум транспортным кольцам.

Каждому SRC 3266 при инициализации выделяется часть адресного пространства шины PCI, которая называется окном для SRC. Окна могут иметь размер 1, 2, 4, 8, 16 Мбайт при числе шин PCI до 256. Если же размеры окон больше указанных, то это уже накладывает ограничения на число шин PCI.

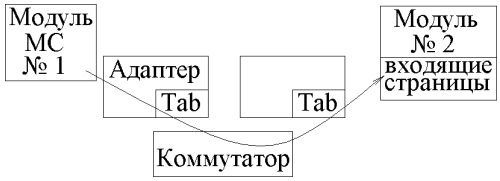


Рис. 22. Передачи в МС

Рис. 23. Двойное подтверждение в МС

В SRC 3266 реализована полностью технология Plug and Play и автоконфигурация средствами BIOS (BIOS содержит программы операционной системы, находящейся в памяти компьютера).

Дальнейшим развитием этой коммуникационной среды является коннектор SRC 6466, предназначенный для 64-разрядных шин PCI.

9. ТЕХНОЛОГИЯ MEMORY CHANNEL

Технология Memory Channel (MC) предназначена для эффективной организации кластерных систем. Она состоит в следующем.В каждом компьютере в ходе инициализации выделяется предписанное количество страниц физической памяти, доступных и другим компьютерам кластера (разделяемые страницы.

Каждый компьютер имеет встраиваемую в него плату-адаптер, соединяющую шину компьютера и входной/выходной каналы, а также имеется две таблицы управления страницами:

а) таблица *обращения* в удаленные разделяемые страницы;

б) таблица *приема обращений* других компьютеров к разделяемым страницам данного.

Каждый элемент таблицы содержит:

1) данные, необходимые для доставки сообщения в другой компьютер (например идентификатор компьютера);

2) данные для указания точного места в странице;

3) служебные данные, указывающие на состоятельность элемента, особенности маршрутизации и т. д.

На основе этих данных адаптер формирует сообщения (пакеты), которые передаются через выходной линк в сеть передачи данных (рис. 22).

Сообщение содержит

– либо *команду чтения* + адрес блока данных, которые необходимо прочитать, + адрес, куда передать;

– либо *команду записи* + адрес, куда записать, + записываемые данные;

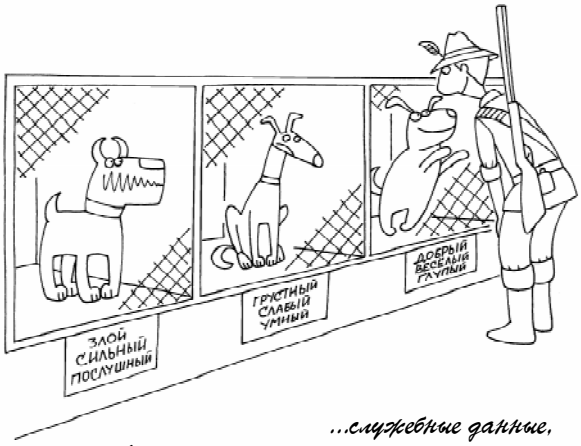
– либо ту или иную служебную информацию.

Каждый ВМ имеет адаптер, присоединяемый к коммутатору; последний производит либо соединение точка–точка, либо соединение точка–всем точкам. В Memory Channel передачи программируются как прямой доступ в память и происходят непосредственно от процессора к процессору в обход операционной системы, что позволяет избежать задержек, причиной которых является операционная система (рис. 23).

Заметим, что коммуникационная среда МС имеет в основном исторический интерес: она мало употребительна.

10. ТРАНСПЬЮТЕРЫ

Появление транспьютеров связано с определенным историческим этапом развития технологий (80-е годы прошлого столетия).



...служебные данные, указывающие на состоятельность элемента, особенности маршрутизации и т.д.

Первоначально они рассматривались как быстродействующие процессоры, но дальнейший технологический прогресс отвел им место в коммуникационных средах.

Транспьютеры на одном кристалле содержат процессор, блок памяти и коммуникационные каналы (линки) для объединения транспьютеров в параллельную систему.

В качестве коммуникационных каналов первоначально использовались OS-линки; они передавали данные в каждом направлении по одному проводу со скоростью 10 Мбит/сек. (иногда был возможен режим 20 Мбит/сек.). Недостатки OS-линков – малая пропускная способность и небольшая скорость передачи – привели к созданию DS-линков; последние имеют встроенный механизм одновременной передачи большого числа информационных потоков.

Физический DS-линк состоит из двух проводников, предназначенных для передачи в одном направлении, и из двух проводников для передачи в противоположном направлении.

В каждом направлении по одному проводнику передается информационный сигнал, а по второму – синхросигнал (строб). При передаче очередного бита данных, если его значение меняется по сравнению с предыдущим битом данных, то значение сигнала в линии стробирования не меняется, а если значение передаваемого и предыдущего битов одинаковы, то меняется значение бита в линии стробирования. Благодаря этому, в каждый момент времени переключение состояния происходит разве лишь в одном проводнике, что снижает помехи. Упомянутая синхронизация позволяет игнорировать фазовые сдвиги и повысить скорость передачи до 100 Мбит/с.

В заключение заметим, что транспьютерные коммуникационные среды все реже употребляются в вычислительных системах.

Для закрепления материала предлагаются вопросы, которые помогут эффективно освоить предложенный в данной статье материал.

1. Зачем нужны параллельные вычислительные системы?

2. Что такое архитектура вычислительной системы?

3. Что такое коммуникационная среда?

4. Как решается задача ускорения вычислений?

5. Какие встречаются топологии соединения вычислительных модулей?

6. Какова роль стандартов при создании коммуникационных сред?

7. В чем отличие контроллера от адаптера?

8. Опишите функции коммутатора. Чем отличаются простой и составной коммутаторы?

9. Что такое чип?

10. Разъясните понятие «шинная структура». Какие шинные структуры вам известны?

11. Что означает «постраничная организация памяти»? Знаете ли вы, какая коммуникационная среда использует такую организацию памяти?

12. Что значит «несостоятельные данные»?

13. Что такое «локальный алгоритм»?

14. Что такое «кэш-память»? В каких случаях она полезна?

15. В каких задачах можно рассчитывать на то, что для их решения удастся использовать локальный алгоритм?

16. Почему приходится хранить данные в нескольких экземплярах?

17. Что означает «копия» в динамике вычислений (почему слово «копия» приходится писать в кавычках)?

18. В чем состоит основная функция коммуникационных сред?

19. Какова иерархия памятей однопроцессорной вычислительной системы?

20. Почему бывает не выгодно сразу вносить изменения в основную память?

21. Какие типы кэш-памяти используются?

22. Какие способы замены строк кэш-памяти вы знаете?

23. Что значит фраза «поддержание когерентности кэшей»?

24. Назовите основные стратегии внесения изменений в основную память.

25. В каких состояниях может находиться строка кэша в алгоритме MESI?

26. Для чего используются упомянутые в предыдущем вопросе состояния?

27. В чем состоит прямой подход поддержания когерентности кэшей в многопроцессорной системе?

28. Каковы основные черты алгоритма DASH?

29. Какова структура коммуникационной среды на базе SCI?

30. Расскажите о коммуникационной среде MYRINET.

31. В чем состоят замечательные свойства шины PCI?

32. Опишите коммуникационную среду SRC 3266.

Литература

1. *Демьянович Ю.К.* Параллельные вычисления. Урок 1. Параллельная форма // Компьютерные инструменты в школе, 2011. № 1. С. 36–39.

2. *Демьянович Ю.К.* Параллельные вычисления. Урок 2. О средствах распараллеливания... Элементарная параллельная программа // Компьютерные инструменты в школе, 2011. № 2. С. 17–21.

3. *Демьянович Ю.К.* Параллельные вычисления. Урок 3. О проблемах распараллеливания // Компьютерные инструменты в школе, 2011. № 3. С. 35–41.

4. *Демьянович Ю.К.* Параллельные вычисления. Урок 4. Программируем на MPI // Компьютерные инструменты в школе, 2011. № 4. С. 30–39.

5. *Корнеев В.В.* Вычислительные системы. М.: Гелиос АРВ, 2004.

Демьянович Юрий Казимирович, доктор физико-математических наук, профессор, заведующий кафедрой параллельных алгоритмов математико-механического факультета СПбГУ.

Схематические рисунки к статье выполнила Т.О. Евдокимова.