**Министерство образования Российской Федерации**

**Марийский государственный технический университет**

Факультет ФИВТ

Кафедра ИВС

**Механизм когерентности обобщенного кольцевого гиперкуба с непосредственными связями**

###### ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ.

Выполнил:

студент группы ВМ-42 Трохимец Г.М.

(дата) (подпись)

Проверил: к.т.н., доцент Власов А.А.

(дата) (подпись)

Оценка:

г. Йошкар-Ола

2002г.

**Аннотация**

В данной работе были рассмотрены механизмы поддержания когерентности в многопроцессорной ВС. Также рассмотрена коммутационная структура типа обобщенного кольцевого гиперкуба, к которой был подобран свой механизм когерентности.

**Оглавление**

**Введение 4**

**Техническое задание 5**

**1. Общая часть 6**

1.1. Механизмы поддержания когерентности 6

1.2. Механизмы неявной реализации когерентности 7

1.2.1. Однопроцессорный подход 8

1.2.2. Многопроцессорный подход 10

1.2.2.1. Сосредоточенная память 10

1.2.2.2. Физически распределенная память 12

1.3. КС типа обобщенного кольцевого гиперкуба 15

1.3.1. Расчет основных параметров 16

**2. Алгоритмы механизма когерентности для обобщенного кольцевого гиперкуба 17**

2.1 Операция чтения 17

2.2 Операция записи 19

**Заключение 20**

**Список литературы 21**

Введение

Многопроцессорную ВС можно рассматривать как совокупность про­цессоров, подсоединенных к многоуровневой иерархической памяти. При таком представлении коммуникационная среда, объединяющая процес­соры и блоки памяти, составляет неотъемлемую часть иерархической памяти. Структурно-технические параметры коммуникационной среды определяют характеристики многоуровневой памяти.

В многопроцессорной ВС для каждого элемента данных должна быть обеспечена когерентность (согласованность, одинаковость) его копий, обрабатываемых разными процессорами и размещенных в разных блоках иерархической памяти. Механизмы реализации когерентности могут быть как явными, так и неявными для прикладного программиста.

Проблема о которой идет речь, возникает из-за того, что значение элемента данных в памяти, хранящееся в двух разных процессорах, доступно этим процессорам только через их индивидуальные кеши.

Современная технологическая база СБИС позволяет создавать вычислительные системы, содержащие в своем составе миллионы процессорных элементов (ПЭ). Препятствием на пути создания таких систем являются проблемы, связанные с организацией управления и обменов данными при решении задач широкого класса. При этом основная сложность заключается в организации коммутационной структуры с высокой степенью регулярности и высокой пропускной способностью при сравнительно небольших аппаратных затратах.

Известные коммутационные структуры не в полной мере отвечают этим требованиям. Все коммутационные структуры можно разделить на две большие группы: КС с непосредственными связями и КС с магистральными связями. Мы рассматриваем первую группу - КС с непосредственными связями. В частности КС обобщенного кольцевого гиперкуба.

Техническое задание

1. Изучить механизмы поддержания когерентности.
2. Рассмотреть КС типа обобщенный кольцевой гиперкуб.
3. Составить алгоритм механизма когерентности КС типа обобщенный кольцевой гиперкуб с непосредственными связями.

1. Общая часть

### Механизмы поддержания когерентности

Механизмы реализации когерентности могут быть как явными, так и неявными для прикладного программиста.

При таком рассмотрении архитектуры ВС можно классифицировать по способу размещения данных в иерархической памяти и способу дос­тупа к этим данным.

**Явное размещение данных**; **явное указание доступа к данным.** Програм­мист явно задает действия по поддержке когерентности памяти посред­ством передачи данных, программируемой с использованием специаль­ных команд "послать" (send) и "принять" (receive). Каждый процессор имеет свое собственное адресное пространство (память ВС распределе­на), а согласованность элементов данных выполняется путем установле­ния соответствия между областью памяти, предназначенной для пере­дачи командой send, и областью памяти, предназначенной для приема данных командой receive, в другом блоке памяти.

**Неявное размещение** **данных; неявное указание** доступа к данным. В ВС с разделяемой памятью механизм реализации когерентности прозрачен для прикладного программиста, и в программах отсутствуют какие-либо дру­гие команды обращения к памяти, кроме команд "чтение" (load) и "за­пись" (store). Используется единое физическое пространство или виртуальный адрес­. Архитектура ВС с разделяемой памятью имеет много привлекательных черт:

• однородность адресного пространства памяти, позволяющая при создании приложений не учитывать временные соотношения между об­ращениями к разным блокам иерархической памяти;

• создание приложений в привычных программных средах;

• легкое масштабирование приложений для исполнения на разном числе процессоров и разных ресурсах памяти.

**Неявное размещение данных как страниц памяти; явное** указание досту­пак данным**.** В этой архитектуре используется разделяемое множество страниц памяти, которые размещаются на внешних устройствах. При явном запросе страницы автоматически обеспечивается когерентность путем пересылки уже запрошенных ранее страниц не из внешней памя­ти, а из памяти модулей, имеющих эти страницы.

**Явное размещение данных с указанием разделяемых модулями страниц; неявное указание доступа к данным посредством команд load, store.**

Существует технология MEMORY CHANNEL эффективной организации кластерных систем на базе модели разделяемой памяти. Суть технологии заключается в следую­щем. В каждом компьютере кластера предполагается организация памяти на основе механизма виртуальной адресации. Адрес при этом состоит из двух частей: группы битов, служащих для определения номера страни­цы, и собственно адреса внутри страницы. В каждом компьютере в ходе инициализации выделяется предписанное, возможно разное, вплоть до полного отсутствия, количество физических страниц памяти, разделяе­мых этим компьютером с другими компьютерами кластера.

После установления во всех компьютерах отображения страниц па­мяти, доступ к удаленным памятям выполняется посредством обычных команд чтения (load) и записи (store) как к обычным страницам вирту­альной памяти без обращений к операционной системе или библиоте­кам времени исполнения.

### 1.2. Механизмы неявной реализации когерентности

Современные микропроцессоры имеют один или несколько уровней внутрикристальной кэш-памяти. Поэтому интерфейс микропроцессоров с необходимостью включает механизм организации когерентности внут­рикристальной кэш-памяти и внекристальной памяти. Внекристальная память может также быть многоуровневой: состоять из кэш-памяти и основной памяти.

Реализация механизма когерентности в ВС с разделяемой памятью требует аппаратурно-временных затрат. Причем уменьшить временную составляющую затрат можно за счет увеличения аппаратурной состав­ляющей и наоборот. Уменьшение временной составляющей требует соз­дания специализированной аппаратуры реализации когерентности. Умень­шение аппаратурной составляющей предусматривает некоторый мини­мум аппаратных средств, на которых осуществляется программная реа­лизация механизма когерентности.

### 1.2.1. Однопроцессорный подход

Создание иерархической многоуровневой памяти, пересылающей блоки программ и данных между уровнями памяти за время, пока пред­шествующие блоки обрабатываются процессором, позволяет существенно сократить простои процессора в ожидании данных. При этом эффект уменьшения времени доступа в память будет тем больше, чем больше время обработки данных в буферной памяти по сравнению с временем пересылки между буферной и основной памятями. Это достигается при локальности обрабатываемых данных, когда процессор многократно ис­пользует одни и те же данные для выработки некоторого результата.

В связи с тем, что локально обрабатываемые данные могут возникать в динамике вычислений и не быть сконцентрированными в одной об­ласти при статическом размещении в основной памяти, буферную па­мять организуют как ассоциативную, в которой данные содержатся в совокупности с их адресом в основной памяти. Такая буферная память получила название **кэш-памяти.** Кэш-память позволяет гибко согласо­вывать структуры данных, требуемые в динамике вычислений, со стати­ческими структурами данных основной памяти.

Типовая современная иерархия памятей для однопроцессорных ВС имеет следующую структуру:

• регистры 64 - 256 слов со временем доступа 1 такт процессора;

• кэш 1 уровня — 8к слов с временем доступа 1—2 такта;

• кэш 2 уровня — 256к слов с временем доступа 3—5 тактов;

• основная память - до 4 Гигаслов с временем доступа 12-55 тактов. Кэш имеет совокупность строк (cache-lines), каждая из которых со­стоит из фиксированного количества адресуемых единиц памяти (бай­тов, слов) с последовательными адресами. Типичный размер строки:

16**,** 32, 64, 128, 256 байтов.

Наиболее часто используются три способа организации кэш-памяти, отличающиеся объемом аппаратуры, требуемой дляих реализации:

Это, так называемые, кэш-память с прямым отображением (direct-mapped ,cache), частично ассоциативная кэш-память (set-associative cache) и ас­социативная кэш-память (fully associative cache).

Реализация механизма когерентности чаще всего осуществляется с использованием отслеживания (snooping) запросов на шине, связывающей процессор, память и интерфейс ввода/вывода. Контроллер кэша отслеживает адреса памяти, выдаваемые процессором, и если 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 фиксируется "промах" и ини­циируются действия по доставке в кэш из памяти требуемой строки. В ряде процессоров, выполняющих одновременно совокупность команд, допускается несколько промахов, прежде чем будет запущен механизм замены строк.

### 1.2.2. Многопроцессорный подход

В современных микропроцессорах, используемых для построения муль­типроцессорных систем, идентичность данных в кэшах ВМ (когерент­ность кэшей) поддерживается с помощью межмодульных пересылок. Существует несколько способов реализации когерентности, применяе­мых в зависимости от типа используемой коммуникационной среды и сосредоточенности или физической распределенности памяти между процессорными модулями.

### 1.2.2.1. Сосредоточенная память

Рассмотрим реализацию одного из алгоритмов поддержки когерент­ности кэшей, известного как MESI (Modified, Exclusive, Shared, Invalid) [б]. Алгоритм MES1 представляет собой организацию когерентности кэш­памяти с обратной записью. Этот алгоритм предотвращает лишние пере­дачи данных между кэш-памятью и основной памятью. Так, если данные в кэш-памяти не изменялись, то незачем их пересылать. Зададим некоторые начальные условия и введем определения. Итак, каждый ВМ имеет собственную локальную кэш-память, имеется общая разделяемая основная память, все ВМ подсоединены к основной памя­ти посредством шины. К шине подключены также внешние устройства. Важно понимать, что все действия с использованием транзакций шины, производимые ВМ и внешними устройствами, с копиями строк, как в каждой кэш-памяти, так и в основной памяти, доступны для отслежи­вания всем ВМ. Это является следствием того, что в каждый момент на шине передает только один, а воспринимают все, подключенные к шине абоненты. Поэтому, если для объединения ВМ используется не шина, а другой тип коммутационной среды, то для работоспособности алгорит­ма MES1 необходимо обеспечение вышеуказанного порядка выполне­ния транзакций. Каждая строка кэш-памяти ВМ может находиться в одном из сле­дующих состояний:

М - строка модифицирована (доступна по чтению и записи только в этом ВМ, потому что модифицирована командой записи по сравнению **со** строкой основной памяти);

Е - строка монопольно копированная (доступна по чтению и записи в этом ВМ и в основной памяти);

S - строка множественно копированная или разделяемая (доступна **по** чтению и записи в этом ВМ, в основной памяти и в кэш-памятях других ВМ, в которых содержится ее копия);

1 - строка, невозможная к использованию (строка не доступна ни по чтению, ни по записи).

Состояние строки используется, во-первых, для определения про­цессором ВМ возможности локального, без выхода на шину, доступа к данным в кэш-памяти, а, во-вторых, - для управления механизмом когерентности.

Для управления режимом работы механизма поддержки когерентно­сти используется бит WT, состояние 1 которого задает режим сквозной (write-through) записи, а состояние 0 - режим обратной (write-back) записи в кэш-память.

Промах чтения в кэш-памяти заставляет вызвать строку из основной памяти и сопоставить ей состояние Е или S. Кэш-память заполняется только при промахах чтения. При промахе записи транзакция записи помещается в буфер и посылается в основную память при предоставле­нии шины.

Для поддержки когерентности строк кэш-памяти при операциях вво­да/вывода и обращениях в основную память других процессоров на шине генерируются специальные циклы опроса состояния кэш-памятей. Эти циклы опрашивают кэш-памяти на предмет хранения в них строки, ко­торой принадлежит адрес, используемый в операции, инициировавшей циклы опроса состояния. Возможен режим принудительного перевода строки в состояние I, который задается сигналом INV.

### 1.2.2.2. Физически распределенная память

Прямолинейный подход к поддержанию когерентности кэшей в муль­типроцессорной системе, основная память которой распределена по ВМ, заключается в том, что при каждом промахе в кэш в любом процессоре инициируется запрос требуемой строки из того блока памяти, в кото­ром эта строка размещена. В дальнейшем этот блок памяти будет по от­ношению к этой строке называться резидентным. Запрос передается че­рез коммутатор в модуль с резидентным для строки блоком памяти, из которого затем необходимая строка через коммутатор пересылается в модуль, в котором произошел промах. Таким образом, в частности, обес­печивается начальное заполнение кэшей. Приэтом в каждом модуле для каждой резидентной строки ведется список модулей, в кэшах которых эта строка размещается, либо организуется распределенный по ВМ спи­сок этих строк. Строка, размещенная в кэше более чем одного модуля, в дальнейшем будет называться разделяемой.

Собственно когерентность кэшей обеспечивается следующим. При обращении к кэш-памяти в ходе операции записи данных, после самой записи, процессор приостанавливается до тех пор пока не выполнится последовательность действий: измененная строка кэша пересылается в резидентную память модуля, затем, если строка была разделяемой, она пересылается из резидентной памяти во все модули, указанные в списке разделяющих эту строку. После получения подтверждений, что все ко­пии изменены, резидентный модуль пересылает в процессор, приоста­новленный после записи, разрешение продолжать вычисления.

Изложенный алгоритм обеспечения когерентности хотя и является логически работоспособным, однако практически редко применяется из-за больших простоев процессоров при операциях записи в кэш стро­ки. На практике применяют более сложные алгоритмы, обеспечиваю­щие меньшие простои процессоров, например, DASH, который заключается следующем. Каждый модуль памяти имеет для каждой строки, резидентной в мо­дуле, список модулей, в кэшах которых размещены копии строк.

С каждой строкой в резидентном для нее модуле связаны три ее воз­можных глобальных состояния:

1) "некэшированная", если копия строки не находится в кэше ка­кого-либо другого модуля, кроме, возможно, резидентного для этой строки;

2) "удаленно-разделенная", если копии строки размещены в кэшах других модулей;

3) "удаленно-измененная", если строка изменена операцией записи

в каком-либо модуле.

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

1) "невозможная к использованию";

2) "разделяемая", если.есть неизмененная **копия,** которая, возмож­но, размешается также в других кэшах;

3) "измененная", если копия изменена операцией записи. Каждый процессор может читать из своего кэша, если состояние чи­таемой строки "разделяемая" или "измененная". Если строка отсутству­ет в кэше или находится в состоянии "невозможная к использованию", то посылается запрос "промах чтения", который направляется в мо­дуль, резидентный для требуемой строки.

Если глобальное состояние строки в резидентном модуле "некэши­рованная" или "удаленно-разделенная", то копия строки посылается в запросивший модуль и в список модулей, содержащих копии рассмат­риваемой строки, вносится модуль, запросивший копию.

Если состояние строки "удаленно-измененная", то запрос "промах чтения" перенаправляется в модуль, содержащий измененную строку. Этот модуль пересылает требуемую строку в запросивший модуль и в модуль, резидентный для этой строки, и устанавливает в резидентном модуле для этой строки состояние "удаленно-распределенная".

Если процессор выполняет операцию записи и состояние строки, в которую производится запись "измененная", то запись выполняется и вычисления продолжаются. Если состояние строки "невозможная к ис­пользованию" или "разделяемая", то модуль посылает в резидентный для строки модуль запрос на захват в исключительное использование этой строки и приостанавливает выполнение записи до получения под­тверждений, что все остальные модули, разделяющие с ним рассматри­ваемую строку, перевели ее копии в состояние "невозможная к исполь­зованию".

Если глобальное состояние строки в резидентном модуле "некэши­рованная", то строка отсылается запросившему модулю, и этот модуль продолжает приостановленные вычисления.

Если глобальное состояние строки "удаленно-разделенная", то рези­дентный модуль рассылает по списку всем модулям, имеющим копию строки, запрос на переход этих строк в состояние "невозможная к ис­пользованию". По получении этого запроса каждый из модулей изменя­ет состояние своей копии строки на "невозможная к использованию" и посылает подтверждение исполнения в модуль, инициировавший опе­рацию записи. При этом в приостановленном модуле строка после ис­полнения записи переходит в состояние "удаленно-измененная".

Предпринимаются попытки повысить эффективность реализации ал­горитма когерентности, в частности, за счет учета специфики парал­лельных программ, в которых используются асинхронно одни и те же данные на каждом временном интервале исключительно одним процес­сором с последующим переходом обработки к другому процессору. Та­кого рода ситуации случаются, например, при определении условий окончания итераций. В этом случае возможна более эффективная схема передачи строки из кэша одного процессора в кэш другого процессора.

### 1.3. КС типа обобщенного кольцевого гиперкуба

Описываемая в данной работе среда обеспечивает построение легко наращиваемой вычислительной системы, которая может содержать большое число процессоров. Поэтому при проектировании она изначально предназначалась для создания систем распределенных вычислений. Однако универсальность коммуникационных процессоров узлов позволяет использовать ее также при создании сетей рабочих станций. Простота наращивания количественных параметров среды обусловлена, прежде всего, регулярностью коммутационной структуры.

Для простоты понимания будем рассматривать частный случай – трехмерную среду с кольцевыми связями. Каждый узел представляет собой совокупность двух процессоров – обрабатывающего и коммуникационного. Обрабатывающий процессор – это общее понятие, под которым понимается любое устройство обработки информации, но которое не может осуществить самостоятельную передачу данных в среде. А коммуникационный процессор является инструментом для обрабатывающего процессора, предоставляющим ему возможность осуществить обмен информацией с другими узлами среды.

Рис. 1

Каждый узел среды имеет 6 двунаправленных каналов ввода/вывода, которые используются в качестве непосредственных связей с соседними узлами (Рис.1). Возможно, применение двух встречных однонаправленных каналов вместо одного двунаправленного, что позволяет увеличить пропускную способность при незначительном увеличении аппаратных затрат. Каждый узел также имеет 6 магистральных каналов ввода/вывода – по два в каждом измерении. Коммуникационный процессор производит прием/передачу информации по каналам ввода/вывода, причем может использоваться как коммутация пакетов, так и коммутация каналов.

### 1.3.1. Расчет параметров

Рассчитаем оценки параметров кольцевого гиперкуба при n=3, m=3. Где n – количество измерений, а m – размерность.

1. Диаметр (D)

;



1. Степень (S)

S=2n = 6;

1. Количество узлов (N)



1. Общее число связей



1. Отказоустойчивость



2. Алгоритмы механизма когерентности для обобщенного кольцевого гиперкуба

Для КС типа обобщенный кольцевой гиперкуба можно реализовать алгоритм механизма поддержания когерентности - DASH, описанный выше в п.1.2.2.2. Так как узел гиперкуба можно подразумевать как ЦП разделенный на:

Операционный процессор

Коммутационный процессор

#### КЕШ

Выполняет все вычисления но не может связываться с другим процессором

Осуществляет связь с другими процессорами

#### Хранит копии строк

### 2.1 Операция чтения

НАЧАЛО

##### Запрос на чтение

Строка измененная или разделяемая

**нет**

**да**

##### Невозможна к использованию

##### Запрос в резидентный модуль,"промах чтения”

**1**

**2**

##### Чтение

**1**

Некешированная или удаленно-разделенная строка

##### Пересылка копии строки в запросивший модуль

**нет**

**да**

##### Строка «удаленно-измененная»

##### Запрос «промах чтения» направляется в модуль содерж. изм. строку

##### Пересылка данной строки в запросив. модуль и в резидентн.

##### Установка состояния строки в «удаленно-рспределенная»

**2**

КОНЕЦ

### 2.2 Операция записи

**нет**

**да**

НАЧАЛО

##### Запрос на запись

**нет**

**да**

измененная

##### Запись

некешированная

##### Пересылка строки запросившему модулю

##### Невозможна к использованию

##### Запрос в резидентный модуль на захват строки

**да**

**нет**

##### Рассылка всем модулям, имеющим копию строки, запроса на перевод строки в состояние «невозможного к использованию»

КОНЕЦ

Заключение

В данной работе мы изучили механизмы поддержания когерентности. Алгоритмы их работы. Рассмотрели КС типа обобщенный кольцевой гиперкуб, рассчитали основные оценки параметров данного гиперкуба (Рис. 1). Более подробно мы остановились на алгоритме DASH, который в наибольшей степени подходит к КС типа обобщенный кольцевой гиперкуб. При построении алгоритма мы видим, что для данной КС с непосредственными связями, чем больше структура, тем дольше ожидание ЦП на запросы запрещения строки. Т.к. с увеличением структуры будет увеличиваться диаметр и до последнего узла сообщение будет доходить с большим опозданием, что вызовет простой запросившего процессора.

В тоже время, если бы мы использовали структуру с магистральными связями, данный алгоритм поддержания когерентности будет работать гораздо эффективнее, т.к. диаметр будет постоянен.

Список литературы

1. Артамонов Г.Т. «Топология регулярных вычислительных сетей и сред.№» М.: Радио и связь, 1985.
2. Власов А.А. «Коммутационные структуры и коммуникационные среды: Лабораторный практикум.» – Йошкар-Ола: МарГТУ, 2002.
3. Горяшко А.П. «Специализированные вычислительные структуры.» /Искусственный интеллект, Кн.3. Программные и аппаратные средства. М.: Радио и связь, 1987.
4. Корнеев В.В. «Параллельные ВС», М., Нолидж, 1999.
5. www.narod.nov.ru/par.html «Мультипроцессорная когерентность кеш-памяти»