**Текст выступления**

**2 Слайд**

Многопроцессорную ВС можно рассматривать как совокупность процессоров, подсоединенных к многоуровневой иерархической памяти. При таком представлении ВС －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 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. Используя механизм реализации когерентности в качестве основания, архитектуры ВС можно классифицировать по способу размещения данных в иерархической памяти и способу доступа к этим данным.

**3 Слайд**

На данном слайде представлена классификация архитектур вычислительных систем по признаку реализации когерентности, которая предполагает разбиение на 4 группы.

**4 Слайд**

В этой архитектуре, называемой также архитектурой с распределенной памятью, программист явно задает действия по поддержке когерентности памяти посредством передачи данных, программируемой с использованием специальных команд «послать» и «принять». Каждый процессор имеет свое собственное адресное пространство, 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. Архитектура ВС с распределенной памятью требует от программиста синхронизации обращений к разным блокам памяти и явного указания на необходимые пересылки данных. Распределение данных по блокам и уровням иерархической памяти также реализуется программистом.

**5 Слайд**

В этой архитектуре, называемой также архитектурой с разделяемой памятью, механизм реализации когерентности прозрачен для прикладного программиста, и в программах отсутствуют какие-либо другие команды обращения к памяти, кроме команд чтения и записи. Используется единое физическое или виртуальное адресное пространство. Архитектура ВС с разделяемой памятью имеет несколько преимуществ, среди которых:

* 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.

**6 Слайд**

На данном слайде представлены варианты механизмов неявной реализации когерентности, а также наиболее часто используемые способы организации кэш-памяти.

**7 Слайд**

Современные микропроцессоры имеют один или несколько уровней внутри кристальной 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.

Идеальная основная память должна иметь «неограниченно» наращиваемый объем и «малое» время доступа к данным. Время обращения к памяти по произвольным адресам должно быть одинаковым. Прямолинейное построение из микросхем памяти блока сколь угодно большого объема со свойствами идеальной основной памяти невозможно. Однако, в связи с тем, что характеристики памяти определяются не только физическими характеристиками микросхем памяти, но и режимом ее использования, прибегают к архитектурно-структурным приемам, учитывающим специфику обращений к памяти пользовательских программ и позволяющим создать при работе этих программ эффект малого времени доступа к большой основной памяти.

Однако в рамках однопроцессорных систем достижимый объем основной памяти с временем доступа, характерным для микросхем памяти, ограничен. Для создания памяти больших объемов прибегают к построению многопроцессорных систем.

В многоуровневой иерархической памяти эффект уменьшения времени доступа в память достигается за счет динамической локализуемости обрабатываемых данных в кэш-памяти и их многократного использования. Однако это дает эффект только на задачах, допускающих такую локализуемость. Если не удается упорядочить и локализовать адреса доступа в память, производительность падает до скорости доступа в оперативную память и даже ниже.

Наиболее часто используются три способа организации кэш-памяти, отличающиеся объемом аппаратуры, требуемой для их реализации. Это так называемые кэш-память с прямым отображением, наборно ассоциативная кэш-память, и ассоциативная кэш-память.

Соответствие между данными в основной памяти и кэш-памяти обеспечивается внесением изменений в те области основной памяти, для которых данные в кэш-памяти подверглись модификации. Эти внесения изменений данных выполняются параллельно с вычислениями.

При выводе данных на внешнее устройство также необходимо отслеживать на шине выдачи данных с соблюдением когерентности данных основной памяти и кэш-памяти.

Основная проблема при создании разделяемой памяти в многопроцессорных системах состоит в том, как извещать другие процессоры об изменениях в разделяемой памяти, вызванных выполнением команд записи в каждом из процессоров. Характерной тенденцией, обусловленной стремлением к повышению производительности микропроцессоров, служит переход от внутрикристальных кэш-памятей со сквозной записью к кэш-памятям с буферизированной записью и к кэш-памятям с обратной записью.

В современных микропроцессорах, используемых для построения мультипроцессорных систем, когерентность кэш-памяти 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.

**8 Слайд**

При аппаратном уровне реализации разделяемой памяти объем разделяемых единиц данных, как правило, составляет кэш-строку. Это обусловлено привязкой к механизму когерентности внутрикристальной многоуровневой кэш-памяти микропроцессоров. Вследствие этого аппаратные реализации универсальны в том смысле, что они не требуют введения в программы дополнительного кода для организации разделяемой памяти. Аппаратные реализации, вследствие применения специально разрабатываемых средств, имеют малую задержку при передаче данных и обеспечивают высокую пропускную способность коммуникационной среды. Однако применение специальных аппаратных средств существенно повышает стоимость таких реализаций по сравнению с программной реализацией разделяемой памяти. При аппаратурном уровне реализации разделяемая память доступна всем программам, как операционной системы, так и прикладным. Все ячейки памяти одинаково доступны, что создает единый системный образ.

Достаточно условно системы с разделяемой (общей) памятью можно разбить на два класса:

* системы с физически одной сосредоточенной общей памятью;
* системы с физически распределенной по вычислительным модулям (ВМ) разделяемой памятью.

Системы, имеющие единую разделяемую память, в основном представлены симметричными мультипроцессорами (ЅМР).

Системы с распределенной разделяемой памятью подразделяются на:

* системы с архитектурой NUMA с неоднородной архитектурой памяти;
* системы с архитектурой СОМА;
* системы с рефлексивной памятью (RM).

B архитектурах NUMA, COMA используется объявление данных несостоятельными, а в системах с рефлективной памятью применяется модификация разделяемых данных. В системах SMP возможно как объявление модифицируемых данных несостоятельными, так и модификация разделяемых данных.

**9 Слайд**

Симметричные мультипроцессоры обычно состоят из небольшого количества процессоров, объединенных с памятью коммуникационной средой (шиной или коммутатором типа «кроссбар»), пропускная способность которой достаточна для поддержания быстрого доступа в память.

Как правило, в ЅMP применяется протокол когерентности, основанный на «прослушивании» всеми процессорами коммуникационной среды и проведении согласованных изменений кэш-памятей и основной памяти. Модель состоятельности памяти строгая, либо последовательная, либо процессорная.

Один из алгоритмов поддержки когерентности кэш-памятей, известный как MESI - этот алгоритм МЕЅІ представляет собой организацию когерентности кэш-памяти 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. Кроме того, возможны еще усовершенствования, связанные с уменьшением количества сквозных записей.

**10 Слайд**

Основная идея архитектуры ccNUMA использование для образования мультипроцессорных систем интерфейса ВМ между кэш-памятью и основной памятью. Протокол когерентности типа MESI обеспечивает механизм поддержки когерентности кэш-памяти и удаленной основной памяти. Время доступа процессора в свою локальную память в несколько раз меньше, чем время доступа через коммуникационную сеть в память другого ВМ.

ВС NUМА － построена из модулей SHV Quad, содержащих ЅМР. Каждый ВМ помимо 4 процессоров имеет память объемом до 4 Гбайт, 2 моста «шина Р6-шина РС» и адаптер, реализующий протокол когерентности, а также входной и выходной каналы по стандарту ЅСІ.

Для построения сcNUMA в адаптер IQ-link вводится дополнительный блок кэш-памяти уровня L3, называемый удаленной кэш-памятью (Remote Cache), который содержит копии строк данных, загружаемых из блоков памяти других ВМ.

**11 Слайд**

В системах с архитектурой СОМА модули памяти ВМ строятся как кэш-память, которую называют притягивающей памятью (AM). При запросе фрагмента памяти из другого ВМ поступивший фрагмент размещается как в кэш-памяти запрашивающего ВМ, так и в его АМ. Ранее размещенный в АМ фрагмент может быть удален из АМ, если нет места для вновь поступившего другого фрагмента.

АМ содержит для каждого фрагмента данных тег, включающий адрес и состояние фрагмента. При промахе в кэш-память контроллер памяти ВМ просматривает теги АМ для определения возможности извлечения требуемого фрагмента из локальной АМ. При отсутствии фрагмента в локальной АМ вырабатывается запрос на его доставку.

В системах с так называемой плоской СОМА архитектурой в каждом ВМ вводится каталог для указания местоположения резидентных в этом ВМ фрагментов. В этих системах может применяться обычная высокоскоростная сеть в отличие от специализированной, используемой в иерархической СОМА. При промахе в АМ по адресу требуемого фрагмента определяется резидентный ВМ, в каталоге которого содержится информация о местонахождении требуемого фрагмента. В резидентный ВМ направляется запрос фрагмента. Запрос или вызывает передачу фрагмента, или, при отсутствии в резидентной АМ фрагмента, перенаправляется согласно каталогу к месту нахождения фрагмента для удовлетворения запроса.

**12 Слайд**

Основное свойство рефлексивной памяти, обусловившее ее название, состоит в том, что каждая копия разделяемого данного в локальной памяти любого ВМ имеет одно и то же значение.

Функционирование системы с рефлексивной памятью (RM) показано на рисунке. Итак, система состоит из коммуникационной среды и совокупности ВМ. Каждый ВМ имеет процессор, блок памяти, шину процессора, по которой он обращается в память, и адаптер коммуникационной среды (сетевой интерфейс). Адаптер состоит из приемо-передающей части коммуникационной среды (1) и приемо-передающей части процессора (2).

Память гранулирована на страницы. Совокупность страниц локальной памяти ВМ делится на две группы: локальные неразделяемые страницы и глобальные разделяемые страницы. Рефлексивная память образуется из всех этих распределенных по различным блокам физической памяти глобально разделяемых страниц памяти, отображенных в глобальное разделяемое адресное пространство.

**13 Слайд**

В большинстве программных реализаций в качестве наименьших элементов разделяемой памяти служат страницы. Однако сравнительно большой объем разделяемых страниц создает проблему ложного разделения, при котором конфликтными признаются модификации разными процессорами различных ячеек памяти одной страницы, на самом деле не конфликтующие между собой.

Программный уровень реализации разделяемой памяти может базироваться либо на компиляторе, либо на библиотеке процедур, статически или динамически связываемых с исполняемой прикладной программой.

Эффективность программных реализаций разделяемой памяти ограничивается большим размером разделяемых единиц, временем выполнения операций синхронизации, включая определение модифицированных данных, а также задержкой и пропускной способностью коммуникационной среды.

При программном уровне реализации разделяемая память доступна только ограниченному классу прикладных программ, написанных с использованием ограниченного круга языковых конструкций, ориентированных на поддержку выявления компилятором записей в разделяемые переменные или использование библиотек для доступа к этим переменным.

Не все ячейки памяти одинаково доступны.

**14 Слайд**

Для реализации модели свободной состоятельности могут быть использованы различные протоколы, в том числе неторопливой свободной состоятельности (LRC) и нетерпеливой (ERC) свободной состоятельности памяти.

В рамках этих моделей задержки, вызванные передачей модифицируемых данных, скрываются за счет буферизации записей или их конвейеризации в пределах критических интервалов. При этом в критическом интервале возможно объединение групп записей в одну, что повышает эффективность их передачи через коммуникационную среду.

Все ВМ выполняют обычные операции чтения/записи и операции синхронизации ***acquire*** и ***release***. По условиям модели должна быть обеспечена последовательная состоятельность выполнения операций синхронизации, то есть все ВМ должны видеть один и тот же порядок выполнения этих операций. Одним из механизмов реализации установления частичной упорядоченности операций синхронизации служит введение в каждом ВМ локальной отметки времени и вектора отметок времени. Вектор отметок времени служит для поддержки временных зависимостей между всеми ВМ. Этот вектор имеет компоненты, однозначно сопоставленные каждому BM.

Каждая модификация памяти описывается отметкой записи (write-notice). Каждый ВМ формирует список отметок записи для каждой страницы.

**15 Слайд**

При неторопливой свободной состоятельности (LRC) операция синхронизации ***acquire***, открывающая критический интервал, обеспечивает состоятельность всех разделяемых страниц для продолжения выполнения критического интервала. Критический интервал завершается операцией ***release***, после которой всем ВМ становятся доступны переменные, модифицированные в этом критическом интервале. Однако отметки записи для модифицированных разделяемых данных завершаемого критического интервала, необходимые для поддержания состоятельности страниц, могут быть получены другим ВМ только при выполнении им операции ***acquire.***

Когда ВМ выполняет операцию ***acquire*** (s), он посылает ВМ, выполнившему перед ним операцию ***release***, копию своего вектора отметок времени. Этот ВМ в ответ высылает отметки записей, отсутствующие в ВМ, выполняющем операцию ***acquire.*** В пересылаемом списке отметок записи имеются сведения не только о записях, произведенных самим ВМ, но и о записях, о которых ВМ стало известно в результате выполнения им операций синхронизации с другими ВМ.

Так, на рисунке для трех ВМ в квадратных скобках указаны векторы отметок времени в каждом ВМ. ВМ0 и ВМ1, выполняют запись в разделяемые страницы памяти в собственных интервалах 2. При операции синхронизации, выполняемой ВМ1, ВМ1, получает из ВМ0, отметку записи в страницу 1. ВМ2, при операции синхронизации получает отметки записи в страницы 1 и 2.

Представленный выше механизм обеспечивает поддержку в каждом ВМ списка модифицированных страниц.

**16 Слайд**

При этом протоколе записи буферизируются до тех пор, пока не потребуется сделать их доступными при очередном выполнении операции release путем трансляционной рассылки измененных данных. Таким образом, основное отличие протокола ERC от LRC состоит в том, что состоятельность страниц поддерживается операцией release, а не асquire. При этом протокол ERC может делать лишние передачи данных для поддержания состоятельности копий страниц, к которым в некоторых ВМ не будет обращений. Соответственно недостатком LRC служит то, что после обращения к несостоятельной странице требуется ожидание, пока будет вычислен и передан этому ВМ необходимый diff.