УДК 004.254

ОБЗОР ПРОТОКОЛОВ КОГЕРЕНТНОСТИ КЭШ-ПАМЯТИ

Сорока Т.Е., Фельдман Л.П., Михайлова Т.В.
Донецкий национальный технический университет
кафедра прикладной математики и информатики
E-mail: ts.master7@gmail.com

Аннотация

Сорока Т.Е., Фельдман Л.П., Михайлова Т.В. Обзор протоколов когерентности кэш-памяти. *Приведена постановка проблемы когерентности кэш-памяти для мультипроцессорных систем с общей памятью, рассмотрена возможность ее решения на аппаратном уровне с помощью реализации протоколов когерентности. Дана классификация протоколов когерентности и краткий обзор их исследований. Рассмотрены схемы протокола MESI и протокола на основе "полного справочника", приведена их сравнительная характеристика. Сделаны выводы об эффективности рассмотренных протоколов.*

Введение

Наиболее перспективным способом увеличения производительности вычислительных систем является дальнейшее совершенствование параллельных вычислительных комплексов. Для многопроцессорных вычислительных систем задача правильной организации памяти является одной из важнейших. Одним из наиболее перспективных направлений в развитии параллельных архитектур являются NUMA-системы, которые в последнее время рассматривается как самостоятельный вид вычислительных систем класса MIMD [1]. Системы этого класса используют физически распределенную память, которая логически является общей, позволяя всем процессорам работать в едином адресном пространстве. Системы этого класса относятся к системам с общей памятью и достаточно легки в программировании, в то же время, предоставляя возможность наращивания вычислительных ресурсов, аналогично системам с распределенной памятью. Использование общих данных в любой мультипроцессорной системе с разделяемой памятью, где каждый процессор имеет собственный кэш, приводит к ситуации, связанной с наличием в нескольких кэшах копий одних и тех же данных. После изменения каким-либо процессором значения общей переменной, ее значения в кэшах других процессоров оказываются недействительными, что приводит к проблеме согласованности (когерентности) кэш-памяти. Эта проблема является актуальной не только для многопроцессорных систем, но и для многоядерных процессоров, где каждое ядро имеет отдельный кэш первого уровня.

В данной статье приведены аппаратные средства разрешения проблемы когерентности кэш-памяти, как позволяющие достичь максимальной производительности мультипроцессорных систем и многоядерных процессоров; кроме того, аппаратная реализация прозрачна для программиста и позволяет создавать наиболее эффективные параллельные программы [2].

Постановка проблемы и обзор методов ее разрешения

На рис. 1 в качестве примера показано состояние двухпроцессорной системы с общей памятью и кэшами первого уровня для каждого процессора до и после изменения разделяемой переменной. При изменении значения переменной X процессором C1, значение ее либо сразу записывается только в кэш процессора C1, а в общую память при вытеснении из кэша C1 (такая стратегия называется обратной записью), либо сразу же и в кэш процессора C1, и в общую память (такая стратегия называется сквозной записью). В обоих случаях значение разделяемой переменной в кэше процессора C2 оказывается неправильным, что и ведет к несогласованности содержимого кэшей.

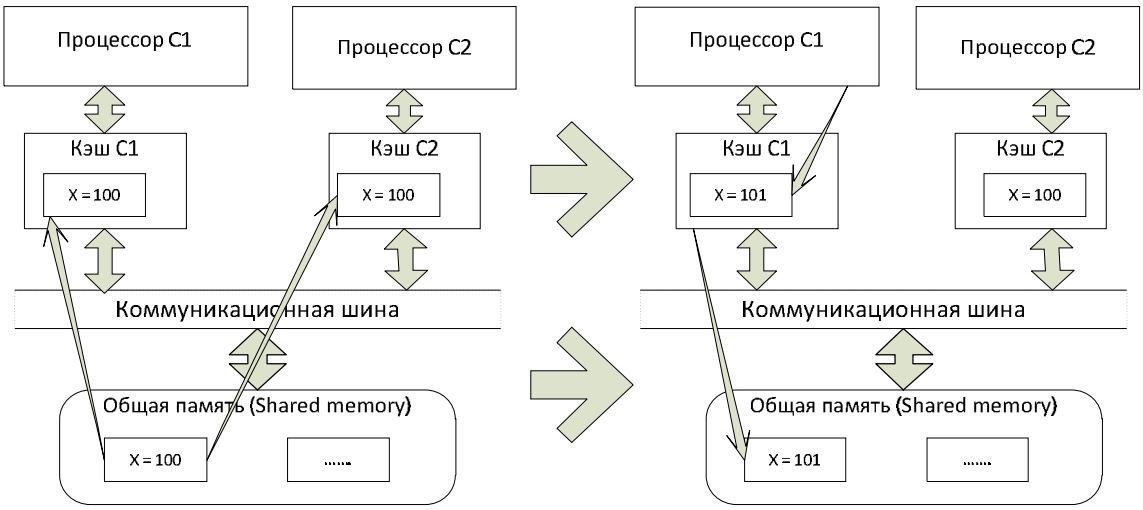

Рисунок 1 – Проблема когерентности кэш-памяти для двухпроцессорной системы

Согласно [2] система является когерентной, если каждая операция чтения по какому-либо адресу, выполняемая любым из процессоров, возвращает значение, занесенное в ходе последней операции записи по этому адресу, вне зависимости от того, какой из процессоров производил запись последним.

Существует несколько способов разрешения проблемы когерентности на аппаратном уровне:

– отказ от кэширования: принципиально решает проблему, но сильно снижает производительность системы за счет значительной разницы между скоростью работы процессоров и скоростью доступа к памяти;

– широковещательная рассылка: при изменении значения переменной одним из процессоров применяется сквозная запись переменной в кэш этого процессора и в общую память, сразу же после этого всем процессорам рассылается запрос на обновление этой переменной в их кэшах, – самый простой и абсолютно надежный способ, но порождает избыточное количество сообщений через коммуникационную среду, что также значительно снижает производительность всей системы;

– реализация специализированных протоколов когерентности: состоит в аппаратной реализации специфического алгоритма, – самый сложный способ, требующий предварительной верификации и анализа на уровне моделей, но позволяющий добиться максимальной производительности параллельной системы.

Под протоколом когерентности кэш-памяти понимают какой-либо аппаратный механизм преодоления проблемы когерентности. Первый протокол когерентности кэш-памяти (“write-once”) описан Гудманом в 1983 г. в [3]. Первой системной работой по анализу и сравнению нескольких протоколов когерентности кэшей является [4]. Среди более поздних работ по верификации, моделированию и анализу протоколов когерентности можно отметить [5,6].

Классификация протоколов когерентности кэш-памяти

Существует несколько классов протоколов когерентности:

– протоколы на основе справочника (directory based). Информация о состоянии блока физической памяти содержится только в одном месте, называемом справочником (физически справочник может быть распределён по узлам системы);

– протоколы наблюдения (snooping). Каждый кэш, который содержит копию данных некоторой части физической памяти, имеет также соответствующую копию служебной информации (биты состояния). Контроллеры кэш-памятей обмениваются между собой служебной информацией о состоянии блока данных.

В протоколах наблюдения процессоры должны широковещательно передавать на шину любые запросы на доступ к памяти, потенциально способные изменить состояние

когерентности совместно используемых блоков данных. Локальный контроллер кэш-памяти каждого процессора затем определяет, присутствует ли в его кэш-памяти копия модифицируемого блока, и если это так, то такой блок аннулируется или обновляется (по этому признаку протоколы разделяются на протоколы с аннулированием и протоколы с обновлением соответственно). Протоколы наблюдения характерны для мультипроцессорных систем на базе шины, в частности многоядерных процессоров.

Протоколы обеспечения когерентности на основе справочника характерны для сложных мультипроцессорных систем с совместно используемой памятью, где процессоры объединены многоступенчатой иерархической сетью. Такие протоколы предполагают сбор и отслеживание информации о содержимом всех локальных кэшей и обычно реализуются с помощью централизованного контроллера. Собственно справочник хранится в основной памяти. Когда контроллер локальной кэш-памяти делает запрос, контроллер справочника обнаруживает такой запрос и формирует команды, необходимые для пересылки данных из основной памяти либо из другой локальной кэш-памяти, содержащей последнюю версию запрошенных данных. Центральный контроллер отвечает за обновление информации о состоянии локальных кэшей, поэтому он должен быть извещен о любом локальном действии, способном повлиять на состояние блока данных.

Наиболее распространенным протоколом среди протоколов наблюдения является протокол MESI, а среди протоколов на основе директорий – протокол на основе полного справочника.

Протокол MESI

Протокол MESI реализован в современных многоядерных процессорах Intel и архитектуре PowerPC. Каждая строка кэша, согласно протоколу MESI, может находиться в одном из четырех состояний:

– Modified – строка в кэше была изменена (отлична от основной памяти) и новое значение доступно только в данном кэше;

– Exclusive – значение строки совпадает со значением в основной памяти, но его нет ни в одном другом кэше;

– Shared: значение строки совпадает со значением в основной памяти и может присутствовать в других кэшх;

– Invalid**:** строка кэша содержит недостоверную информацию.

Автоматная модель протокола MESI представлена на рис. 2 [7].

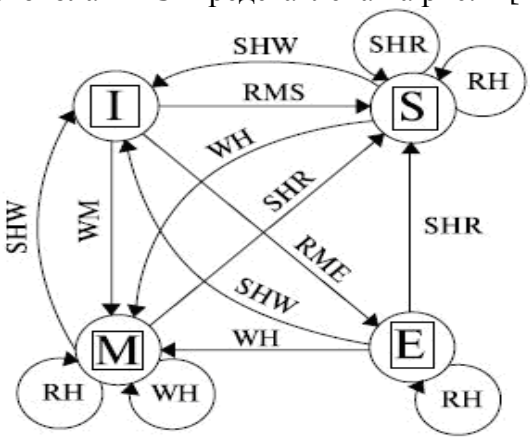

Рисунок 2 – Аавтоматная модель протокола MESI

Для каждого процессора возможны следующие ситуации:

– RH (Read Hit) - кэш-попадание при чтении;

– WH (Write Hit) - кэш-попадание при записи;

– RME (Read Miss Exclusive) - кэш-промах при чтении;

– RMS (Read Miss Shared) - кэш-промах при чтении, но соответствующий блок есть в

кэш-памяти другого микропроцессора;

– WM (Write Miss) - кэш-промах при записи;

– SHR (Snoop Hit Read) - обнаружение копии блока при прослушивании операции чтения другого кэша.

Например, если строка кэша является недостоверной (I), и отсутствует в других кэшах, а процессору требуется соответствующий блок на чтение, то происходит переход RME, и правильное значение переписывается из основной памяти в данный кэш, затем строка переходит в состояние Exclusive (I-RME-E).

Аналогичные автоматные модели лежат в основе многих работ по анализу протоколов наблюдения [5, 6].

Протокол на основе полного справочника

Протокол на основе “полного справочника” является наиболее простым из протоколов на основе директорий [1]. Здесь присутствует единый централизованный справочник, хранящийся в основной памяти, который поддерживает информацию обо всех кэшах. Справочник содержит множество записей, описывающих каждую кэшируемую ячейку ОП. Обращение к справочнику производится каждый раз, когда один из процессоров изменяет копию такой ячейки в своей локальной памяти. В этом случае информация из справочника нужна для того, чтобы аннулировать или обновить копии измененной ячейки (или всей строки, содержащей эту ячейку) в других локальных кэшах, где присутствует эта же строка.

Дли каждого блока ОП в справочнике выделяется одна запись, хранящая указатели на копии данной строки. Кроме того, в каждой записи выделен один бит модификации (D), показывающий, является ли копия “грязной”- (D = 1 - dirty) или “чистой”-(D = 0 - clean), то есть изменялось ли содержимое строки в кэш-памяти после того, как она была туда загружена. Этот бит указывает, имеет ли право процессор производить запись в данную строку. Ниже приведен простейший пример реализации протокола на основе полного справочника [1].

В системе из N процессоров каждая запись справочника будет содержать N однобитовых указателей. Если в соответствующей локальной кэш-памяти есть копия данных, бит-указатель устанавливается в 1, иначе - в 0. На рис. 3 показана схема N-процессорной системы с полным справочником. Здесь предполагается, что копия строки имеется в каждом кэше. Каждой строке придаются два индикатора состояния: бит достоверности (V) и бит владения (P). Если информация в строке корректна, ее V-бит устанавливается в 1. Единичное значение P-бита указывает, что данному процессору предоставлено право на запись в соответствующую строку своей локальной кэш-памяти.

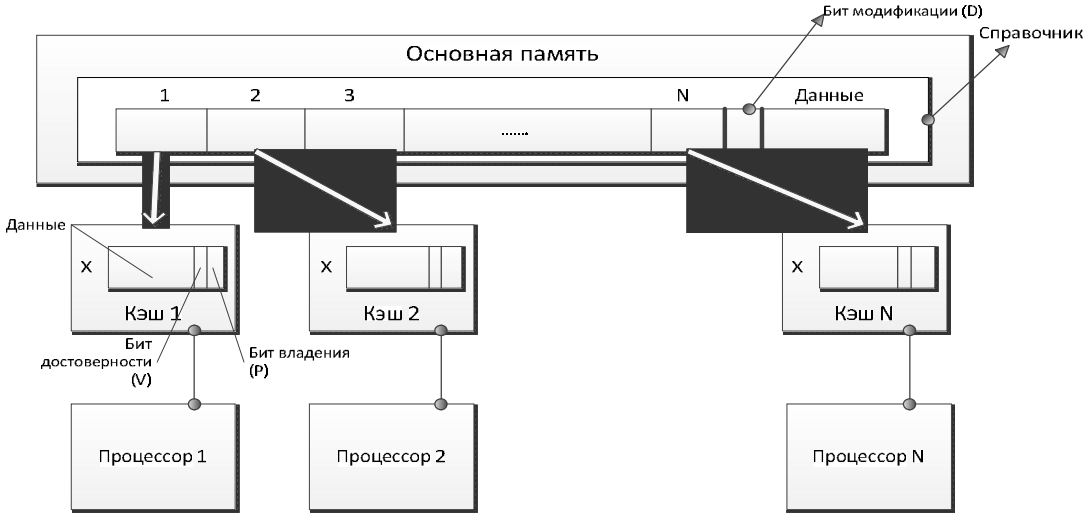

Рисунок 3 – N-процессорная схема с общим справочником

Предположим, что процессор 2 производит запись в ячейку X. В исходный момент процессор еще не получил разрешения на такую запись. Он формирует запрос, в ответ на который во все кэши, где есть копии строки, содержащей ячейку X, выдается сигнал аннулирования имеющихся копий. Каждый кэш, получивший этот сигнал, сбрасывает бит достоверности аннулируемой строки (V) в 0 и возвращает контроллеру справочника сигнал подтверждения. После приема всех сигналов подтверждения контроллер справочника устанавливает в единицу бит модификации (D) соответствующей записи справочника и посылает процессору 2 сигнал, разрешающий запись в ячейку x. С этого момента процессор 2 может продолжить запись в собственную копию ячейки X, а также в основную память, если в кэше реализована схема сквозной записи.

Выводы

Приведены возможные способы решения проблемы когерентности кэшей на аппаратном уровне, дана классификация протоколов когерентности и описаны основные особенности каждого класса протоколов. Для каждого класса рассмотрены наиболее актуальные протоколы. Выявлены типы параллельных систем, для которых целесообразно применения каждого класса протоколов:

– для протоколов наблюдения – параллельные системы с общей памятью, имеющие относительно простую структуру связей (многоядерные процессоры) и небольшое число процессоров (до 32-х); применение протоколов наблюдения для больших мультипроцессорных систем не рекомендуется ввиду их плохой масштабируемости (значительное увеличение числа процессоров резко снижает производительность системы);

– для протоколов на основе справочника – большие мультипроцессорные системы со сложной структурой и большим числом процессоров (32-64 и более); применение таких протоколов для малых систем нецелесообразно ввиду дороговизны их реализации и больших накладных расходов на размещение справочника в основной памяти.

В качестве направления дальнейших исследований для протоколов на основе наблюдения можно выделить построение автоматных моделей для различных модификаций протокола MESI, и оценку производительности протоколов и их верификацию с помощью этих моделей. Для протоколов на основе справочника наиболее интересным представляется исследование возможности более рационального хранения справочника, оптимизацию его структуры, а также возможность распределения справочника по узлам системы.

Литература

1. Цилькер Б. Я., Орлов С. А. Организация ЭВМ и систем - СПб: Питер, 2006. - 668 с.

2. Tomasevic M., Milutinovic V. “The cache coherence problem in shared-memory multiprocessors”, IEEE Computer Society Press, LosAlamos, CA. 1993.

3. Goodman J. R., “Using cache memory to reduce processor-memory traffic”, Proceeding of the 10th International Symposium on Computer Architecture, 1983.

4. Archibald J., Baer J., “Cache coherence protocols: evaluation using a multiprocessor simulation model”. ACM Trans. Comput. Syst., 1986.

5. Emerson E. A., Kahlon V. “Rapid parameterized model checking of snoopy cache coherence protocol”, Proceedings of 9th International Workshop on frontiers of combining systems, 2002.

6. Рудометов В. В., Семенов В. С., Анализ когерентности кэш-памятей для повышения эффективности тестирования подсистемы памяти. //Сборник научно-технических трудов “Высокопроизводительные вычислительные системы и микропроцессоры”, - М.: ИМВС РАН №4, 2003.

7. Гуров В. В. Архитектура микропроцессоров. – Режим доступа: http://www.intuit.ru/department/hardware/microarch