УДК 004.254

Т.Е. Сорока, аспирант
ГВУЗ «Донецкий национальный технический университет», Украина
ts.master7@hotmail.com

# Исследование масштабируемости алгоритмов поддержки когерентности кэш-памяти многоядерных процессоров

*Рассмотрены способы исследования и критерии масштабируемости алгоритмов поддержки когерентности кэш-памяти многоядерных процессоров. Проведено исследование масштабируемости протоколов когерентности MSI и MESI. По результатам исследования сделаны выводы о целесообразности расширения протоколов когерентности за счет введения дополнительных состояний для различных моделей рабочей нагрузки на кэш-память.*

***Ключевые слова: кэш-память, поддержка когерентности, многоядерные процессоры, протокол когерентности кэш-памяти, масштабируемость алгоритма.***

## *Введение*

В настоящее время многоядерные процессоры с общей памятью являются доминирующим классом вычислительных устройств общего назначения и находят широкое применение в аппаратных платформах мобильных, настольных и серверных вычислительных систем, а также являются основой (иногда совместно с GPU) вычислительных узлов суперкомпьютерных систем. Число ядер в широко распространенных моделях процессоров для настольных компьютеров уже достигает 6-8, а в серверных версиях – 12-16, при этом дальнейшее увеличение числа ядер признается одним из приоритетных направлений увеличения производительности.

Разработка эффективных архитектур для многоядерных процессоров порождает ряд сложных задач, одной из которых является создание эффективного аппаратного механизма поддержки когерентности кэш-памяти [1, 2, 3]. За последние годы ведущими производителями процессоров (Intel, AMD, IBM) было спроектировано и реализовано множество архитектур процессоров, использующих различные механизмы поддержки когерентности [3, 4]. Значительной проблемой на пути дальнейшего развития большинства моделей современных процессоров является недостаточная масштабируемость используемых алгоритмов поддержки когерентности кэш-памяти (и их аппаратных реализаций), ввиду чего увеличение числа процессорных ядер уже не всегда приводит к росту производительности системы. При работе с некоторыми приложениями эффект от добавления новых ядер может нивелироваться порождением слишком большого объема служебного трафика для поддержания когерентности через системную шину, что будет только замедлять работу; ввиду наличия данного эффекта исследование масштабируемости алгоритмов поддержки когерентности кэш-памяти является актуальной задачей как оптимизации архитектур кэш-памяти современных процессоров, так и для создания эффективных многопоточных приложений для выполнения на таких процессорах.

Можно отметить ряд работ, посвященных тематике исследования: в [1] рассматривается новая аппаратная реализация уже существующих алгоритмов поддержки когерентности и анализируется ее масштабируемость, общему анализу проблемы когерентности кэш-памяти в мультипроцессорных системах посвящены работы [2, 5], в [6] предлагается новый тип протокола когерентности с высокой масштабируемостью, в [7] рассматривается кэш-когерентная архитектура мультипроцессора для мобильных систем, оптимизации алгоритмов поддержки когерентности кэш-памяти посвящены работы [8, 9], в [10, 11] приводится сравнительный анализ производительности систем памяти (в том числе систем поддержки когерентности кэш-памяти) для нескольких классов мультипроцессорных систем.

Целью работы является разработка методики оценки масштабируемости алгоритмов поддержки когерентности кэш памяти для системы оценки эффективности архитектур системы памяти мультипроцессоров с общей памятью.

Задачей исследования является экспериментальная оценка и анализ масштабируемости алгоритмов поддержки когерентности кэш-памяти, на примере протоколов MSI и MESI.

## *Описание исследуемых алгоритмов*

В данной работе будут рассматриваться только алгоритмы поддержки когерентности на основе наблюдения, как наиболее характерные

для мультипроцессорных систем на базе шины. Такие алгоритмы, как правило, реализуются аппаратно, и именуются протоколами когерентности кэш-памяти [12].

В протоколах наблюдения процессоры должны широковещательно передавать на шину любые запросы на доступ к памяти, потенциально способные изменить состояние когерентности совместно используемых блоков данных. Кэш-контроллер каждого процессора затем определяет, присутствует ли в его кэш-памяти копия модифицируемого блока, и если это так, то такой блок аннулируется или обновляется (в зависимости от протокола) [12].

В протоколах когерентности кэш-памяти стратегия обеспечения когерентности рассматривается как смена состояний в конечном автомате: любой блок в локальной кэш-памяти может находиться в одном из фиксированных состояний [12]. Протокол когерентности определяет набор состояний и условий перехода между ними, задавая кэш-контроллеру набор правил, т.е. "указания", что он должен делать с блоком памяти, находящимся в том или ином состоянии в случае обнаружения шинной транзакции заданного типа.

Рассмотрим протокол когерентности MSI, являющийся базовым для большинства современных протоколов на основе наблюдения. Он предполагает нахождение каждого блока кэш-памяти в одном из трех состояний: I (invalid – недостоверный), M (modified – модифицированный, несогласованный с основной памятью и другими кэшами) и S (shared – разделяемый, согласованный с основной памятью и другими кэшами). Диаграмма переходов протокола MSI представлена на рис. 1 (сплошными стрелками отмечены транзакции инициированные локальным процессором, а курсивом - удаленным).

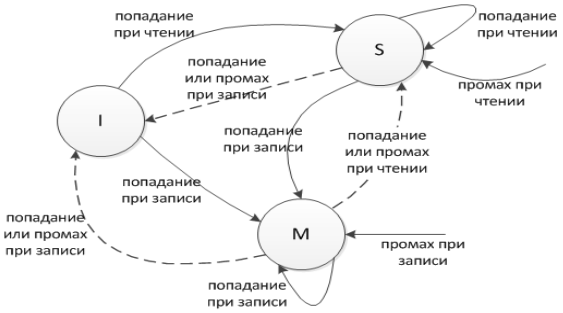


Рисунок 1 – Диаграмма переходов протокола MSI

Протокол работает следующим образом:

– в случае кэш-попадания при чтении блок немедленно загружается из кэша, если он не находится в состоянии I, иначе он загружается из кэша, в котором есть соответствующая копия в состоянии M (одновременно эта копия переписывается в основную память с изменением состояния на S), если же копии в состоянии M нет ни в одном кэше, блок загружается непосредственно из основной памяти в вызывающий кэш с изменением состояния на S;

– в случае промаха при чтении, если у любого другого кэша есть копия блока, он выставляет ее на шину; если блок в состоянии M, он также одновременно записывается в основную память; если блок в состоянии S, то кэш с наивысшим приоритетом выставляет его на шину; все кэши, имеющие копию блока, заметят промах и изменят состояния своих копий на S;

– в случае попадания при записи, если блок находится в состоянии M, запись может быть произведена без задержки; если блок в состоянии S, запись откладывается до тех пор, пока сигнал аннулирования не будет послан на шину, что вызовет смену всех состояний копии блока в других кэшах на I; затем записывающий кэш сможет продолжить запись и установит локальное состояние блока в M;

– в случае промаха при записи, если в другом кэше есть копия блока, то блок поставляется этим кэшем; остальные кэши аннулируют свои копии, блок загружается в вызывающий кэш в состояние M.

Современные протоколы когерентности представляют собой расширения протокола MSI за счет введения 1-2 дополнительных состояний. Второй протокол, масштабируемость которого мы будем рассматривать, – MESI, расширение протокола MSI за счет введения дополнительного состояния E; это протокол когерентности, являющийся базовым для современных процессоров Intel (в последних моделях процессоров Intel используется его расширение MESIF, которое добавляет к протоколу еще одно состояние [3]).

Состояние E (exclusive) означает, что блок действительный, согласован с основной памятью и отсутствует в любом другом кэше. Состояние S теперь указывает, что блок действительный, согласован с основной памятью и присутствует в нескольких кэшах одновременно. Состояния M и I полностью совпадают с соответствующими состояниями протокола MSI. Диаграмма переходов для протокола MESI представлена на рис. 2.

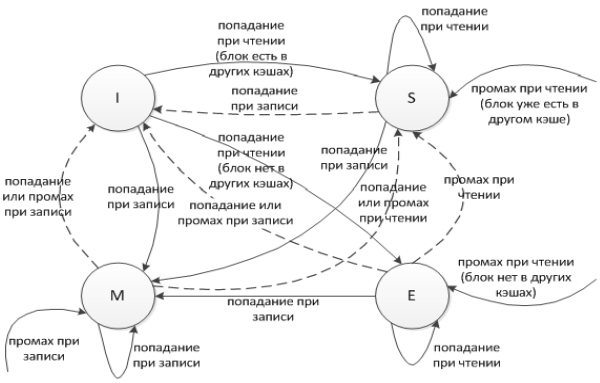


Рисунок 2 – Протокол MESI

Оптимизация по сравнению с протоколом MSI здесь заключается в том, что при промахе записи на блоке в состоянии E не требуется посылка на шину (и обработка контроллерами) сигнала аннулирования, так как заведомо известно, что копий этого блока в других кэшах нет; при многих моделях рабочей нагрузки эта особенность уменьшает число шинных транзакций и, таким образом, увеличивает производительность мультипроцессора.

Далее будем исследовать, насколько эффективна такая оптимизация при различных моделей рабочей нагрузки для процессоров с различным числом ядер.

### *Способы исследования и критерии оценки масштабируемости протоколов когерентности кэш-памяти*

Для исследования эффективности системы памяти мультипроцессоров (в том числе протоколов когерентности кэш-памяти) обычно используется имитационное моделирование с применением различных симуляторов мультипроцессора либо отдельных его узлов [1, 6, 7, 9, 11, 13], но встречаются и аналитические модели [8, 14, 15, 16], в большинстве случаев основанные на цепях Маркова или СМО.

Отдельно следует остановиться на модели рабочей нагрузке для системы памяти в случае имитационного моделирования, здесь можно выделить 3 основных подхода:

— использование реальных трасс обращений к памяти, полученных в результате выполнения реальных приложений на мультипроцессорах различной архитектуры (trace driven simulation);

— использование искусственных трасс обращений к памяти, сгенерированных с помощью аналитических моделей различных классов задач;

— использование искусственных трасс обращений к памяти, сгенерированных на основе векторов случайных величин, задающих характеристика распределения запросов к памяти.

В данном исследовании применен последний подход; используемая модель рабочей нагрузки была описана в [17] и реализована в имитационной модели, рассмотренной в [18]; первичные результаты исследования масштабируемости нескольких протоколов когерентности кэш-памяти для некоторых моделей нагрузки были представлены в [18, 19].

В качестве входного параметра для генерации трасс обращений к памяти, на основе прогона которых будет анализироваться масштабируемость рассматривается вектор Y из 6 элементов:

$$Y = \{P_{запр}, P_{чт}, P_{разд}, n_{разд}, \delta, t\} \qquad (1)$$

где запроса $P_{запр}$ – вероятность генерации запроса к памяти в очередной такт процессорного времени (доля обращений к памяти в общем числе операций процессора), $P_{чт}$ – вероятность того, что очередной запрос к памяти будет запросом на чтение (доля операций чтения среди всех запросов к памяти), $P_{разд}$ – вероятность того, что очередной запрос к памяти будет адресован к области разделяемых блоков (уровень разделения), $n_{разд}$ – число разделяемых блоков памяти (таких блоков, к которым в ходе выполнения программы могут обращаться все процессоры системы), $\delta$ – среднее отклонение адреса запроса для области частных блоков (параметр для регулировки коэффициента кэш-попаданий), $t$ – число процессорных тактов, которые должны быть промоделированы (время моделирования в дискретных единицах) [18].

На выходе модели получаем вектор показателей эффективности G:

$$G = \{h, \rho, w, \overline{P_{сост}}\} \qquad (2)$$

где $h$ – коэффициент кэш-попаданий, $\rho$ – загрузка системы (рассчитывается как отношение полезного времени работы, когда процессор не простаивает в ожидании ответа на запрос к памяти, к общему времени работы), $w$ – суммарная мощность работы системы в расчете на 1000 тактов процессорного времени (сумма всех полезных тактов дискретного времени, выполненных всеми процессорами системы за время моделирования в 1000 единиц), $\overline{P_{сост}}$ – вектор с усредненными долями пребывания блока кэш-памяти в том или ином состоянии протокола когерентности, для протоколов MSI и MESI будет состоять из 3 и 4 элементов соответственно: $\{P_M, P_S, P_I\}$ и $\{P_M, P_E, P_S, P_I\}$.

Под масштабируемостью протокола когерентности понимается свойство алгоритма не терять эффективность (либо терять в настолько незначительной мере, что это не сказывается отрицательно на общей эффективности системы) при увеличении нагрузки на подсистему когерентности кэш-памяти, т.е. при увеличении числа процессорных ядер и/или росте интенсивности взаимодействия процессоров с разделяемой областью памяти. В данном исследовании будем моделировать системы с числом ядер не более 16.

Масштабируемость протокола когерентности будем оценивать с помощью 4 параметров:

— максимальное эффективное число процессоров ($N_{ПРОЦmax}$) ;

— максимальная мощность, которая была достигнута при моделировании с использованием данного протокола ($w_{max}$);

— относительный рост мощности при увеличении числа ядер с 2 единиц до $N_{ПРОЦmax}$ (w↑);

– относительное падение мощности при увеличении числа ядер с максимально эффективного числа ($N_{ПРОЦmax}$) до 16 (w↓).

Кроме того, с целью всестороннего исследования масштабируемости сделаем сравнительный анализ зависимостей загрузки системы от уровня разделения и числа разделяемых блоков для протоколов MSI и MESI.

Чтобы исследовать эффект от введения дополнительного состояния в протоколе MESI, также проанализируем изменение вероятности нахождения блоков памяти в состоянии E в зависимости от уровня разделения блоков для систем с различным числом процессорных ядер.

### *Результаты моделирования и их анализ*

Моделирование проводилось для системы с кэш-памятью размером 2Мб, размер блока – 64 байта, отношение времени цикла основной памяти к циклу кэша – 10 к 1. В ходе моделирования во всех случаях каждый процессор выполнил 50000 операций, минимальный коэффициент попаданий кэш-памяти 95%, доля запросов к памяти в потоке операций процессора – 40%, доля запросов на чтение среди всех запросов к памяти – 60%, размер области разделяемых блоков – 16Кб.

На рис. 3 представлены результаты исследования зависимости производительности системы от уровня разделения данных и числа процессорных ядер при использовании протокола MSI.

Для используемой имитационной модели в ходе данного эксперимента максимальная мощность при всех уровнях разделения была достигнута при 8 процессорных ядрах (при дальнейшем увеличении числа ядер производительность системы падает, так как чрезмерный рост трафика через системную шину нивелирует эффект от добавления новых ядер).

Для большинства случаев максимальная мощность достигается при невысоком уровне разделения данных (10%-30%), минимальная мощность почти всегда достигается при среднем и высоком уровне разделения (40%-50%). Это можно объяснить тем, что при более высоком уровне разделения данных механизм поддержки когерентности задействуется чаще, что порождает дополнительные объемы служебного трафика через системную шину, из-за чего снижается общая производительность системы.

Для систем с 10-16 ядрами производительность меняется довольно незначительно с изменением как числа ядер, так и уровня разделения.

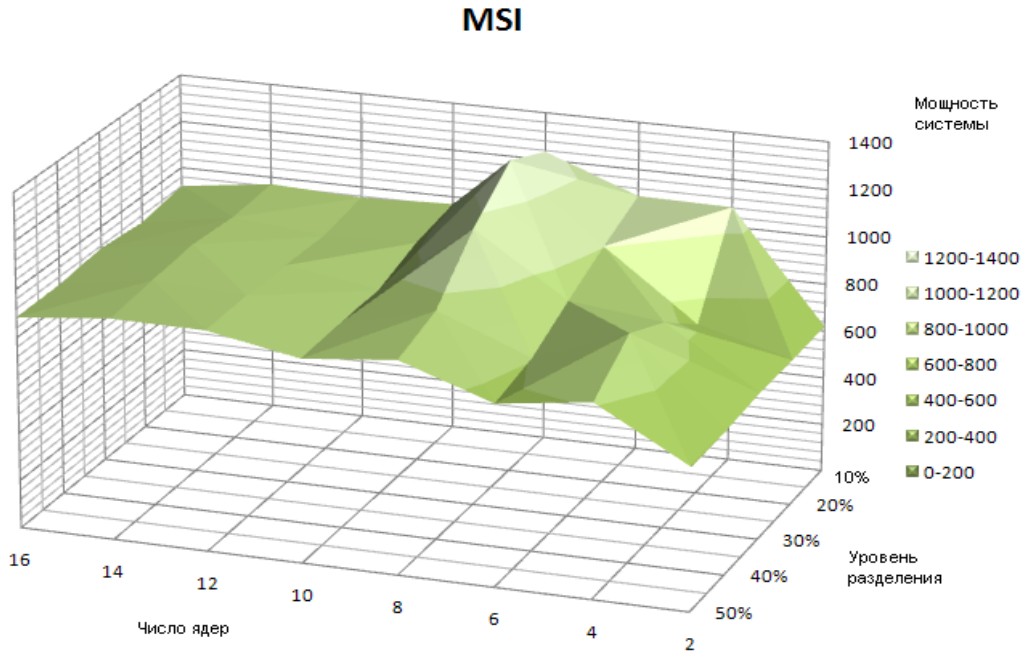


Рисунок 3 – Изменение мощности системы в зависимости от уровня разделения и числа ядер процессора для протокола MSI

На рис. 4 представлены результаты соответствующего эксперимента для протокола MESI. Общий вид характера изменения производительности системы остается прежним: при фиксированном числе ядер производительность падает с увеличением уровня разделения данных, при фиксированном уровне разделения производительность сначала растет до достижения числа ядер $N_{ПРОЦmax}$, а затем плавно падает, достигая минимума при 16 ядрах. В табл. 1 приведены параметры масштабируемости по результатам экспериментов для обоих протоколов.

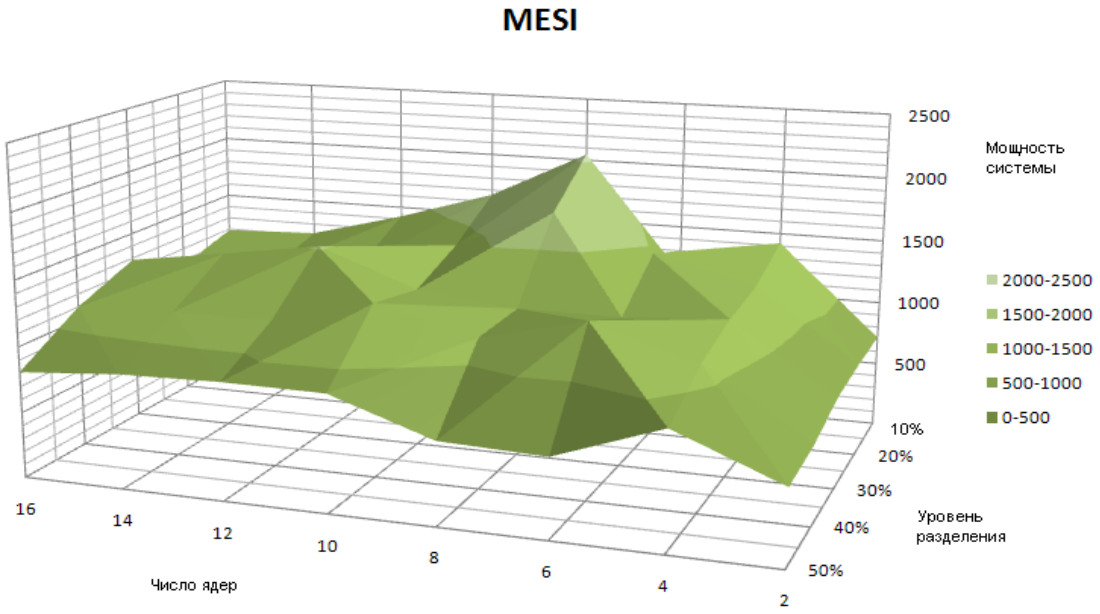


Рисунок 4 – Изменение мощности системы в зависимости от уровня разделения и числа ядер для протокола MESI

Протокол MESI, в большинстве случаев, показывает лучшую производительность в среднем на 20-40%.

Таблица 1. Сравнение показателей масштабируемости протоколов MSI и MESI

| Протокол / уровень разделения | $N_{ПРОЦmax}$ | w↑ (2→max) | w↓ (max→16) | $w_{max}$ |
|---|---|---|---|---|
| MSI (10%) | 8 | +96,9% | -27,4% | 1234 |
| MESI (10%) | 8 | +28% | -9,1% | 2047 |
| MSI (20%) | 8 | +94,4% | -25,4% | 1322 |
| MESI (20%) | 8 | +110% | -45% | 1727 |
| MSI (30%) | 8 | +85,4% | -17,4% | 1163 |
| MESI (30%) | 12 | +125% | -20% | 1515 |
| MSI (40%) | 8 | +66,9% | -10,4% | 957 |
| MESI (40%) | 6 | +115% | -12% | 1337 |
| MSI (60%) | 8 | +64,2% | -7,4% | 890 |
| MESI (60%) | 12 | +32% | -7% | 890 |

Кроме того, для протокола MESI при уровнях разделения 10% и 20% наиболее производительной (как и для MSI) оказалась 8-ядерная конфигурация, но при уровнях 30% и 50% – 12-ядерная, а для уровня разделения 40% – 6-ядерная (единственный эксперимент, где протокол MESI оказался менее масштабируемым, чем MSI, хотя и показал примерно одинаковый с MSI результат).

В целом протокол MESI является более масштабируемым MSI, и позволяет достичь большей производительности в большинстве случаев.

В следующем эксперименте проанализируем масштабируемость с несколько другой стороны, для чего рассмотрим зависимость загрузки системы от уровня разделения и размера области разделяемых блоков при фиксированном числе ядер. Диаграмма изменения загрузки системы в зависимости от уровня разделения и числа разделяемых блоков для протоколов MSI и MESI по результатам моделирования 4-ядерного процессора представлена на рис. 5.

Как видно из рисунка, протоколы MSI и MESI почти одинаково чувствительны к изменению уровня разделения и числу разделяемых блоков. Для обоих протоколов при фиксированном уровне разделения наблюдается незначительное падение уровня загрузки при увеличении числа разделяемых блоков (3.5%-5%); при фиксированном числе разделяемых блоков с увеличением уровня разделения с 5% до 45% загрузка системы падает на 12-13% для протокола MSI и на 12-14% для протокола MESI.

В табл. 2 представлены результаты сравнения суммарной загрузки системы по результатам исследования для трех уровней разделения: 5% (низкий уровень), 25% (средний уровень) и 45% (высокий уровень). В табл. 1 представлены результаты сравнения суммарной загрузки системы по результатам последнего исследования для трех размеров разделяемой области: 16 (1Кб), 128 (8Кб) и 1024 (64Кб) разделяемых блоков.

Для нескольких фиксированных размеров разделяемой области не была выявлена четкая зависимость, при малом и большом размере области несколько лучший результат показал протокол MSI, в то же время для среднего размера области (8Кб – 128 разделяемых блоков) более весомый перевес показал протокол MESI. В целом применение протокола MESI является более целесообразным.

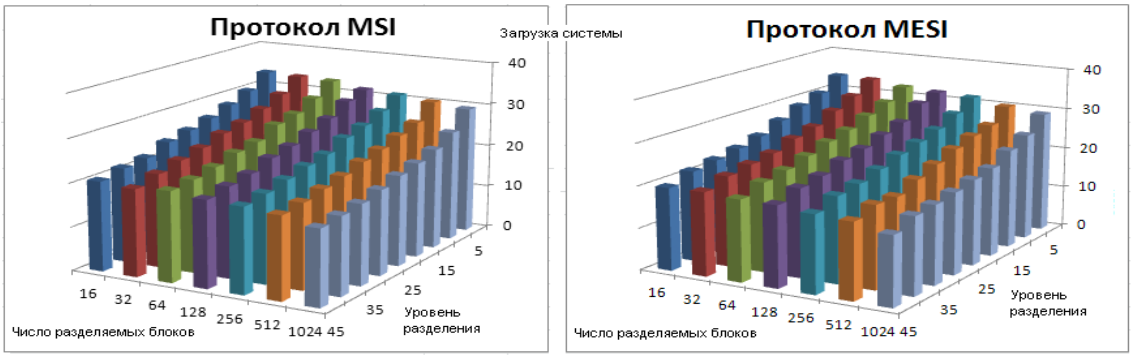


Рисунок 5 – Изменение загрузки системы в зависимости от уровня разделения и числа разделяемых блоков для протоколов когерентности MSI и MESI

Таблица 2. Сравнение протоколов MSI и MESI по суммарному уровню загрузки системы при фиксированном уровне разделения

| Уровень разделения | Лучший протокол | Перевес в загрузке |
|---|---|---|
| 5% | MESI | 0.7% |
| 25% | MESI | 0.6% |
| 45% | MSI | 5% |

Таблица 3. Сравнение протоколов MSI и MESI по суммарному уровню загрузки системы при фиксированном размере разделяемой области

| Число разделяемых блоков | Лучший протокол | Перевес в загрузке |
|---|---|---|
| 16 блоков | MSI | 0.4% |
| 128 блоков | MESI | 1.6% |
| 1024 блока | MSI | 0.1% |

В заключительном эксперименте проведем анализ зависимости вероятности нахождения блоков кэш-памяти в каждом состоянии, предусмотренном протоколом, от числа ядер для уровней разделения данных 20% и 40%.

На рис. 6 и рис. 7 представлены результаты распределения вероятностей для протокола MSI при уровне разделения 20% и 40% соответственно. На рис. 8 и рис. 9 представлены результаты распределения вероятностей для протокола MESI при уровне разделения 20% и 40% соответственно.

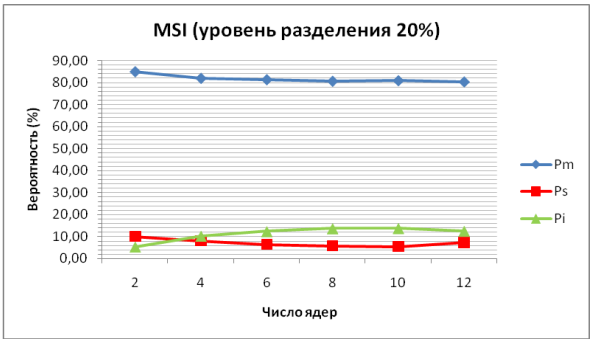


Рисунок 6 – Зависимость вероятности нахождения блоков в различных состояниях протокола MSI от числа ядер при уровне разделения 20%

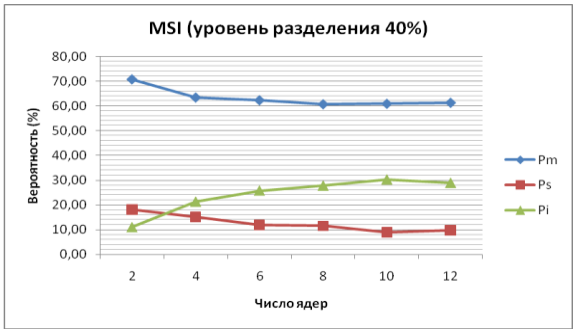


Рисунок 7 – Зависимость вероятности нахождения блоков в различных состояниях протокола MSI от числа ядер при уровне разделения 40%

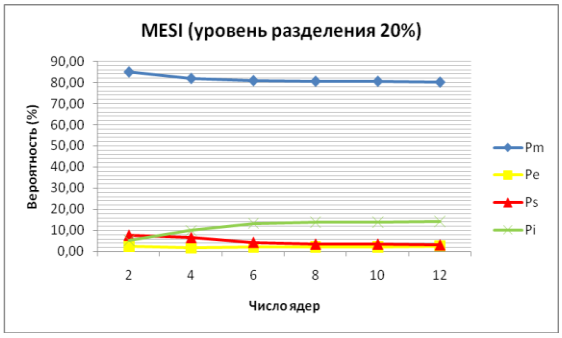


Рисунок 8 – Зависимость вероятности нахождения блоков в различных состояниях протокола MESI от числа ядер при уровне разделения 20%

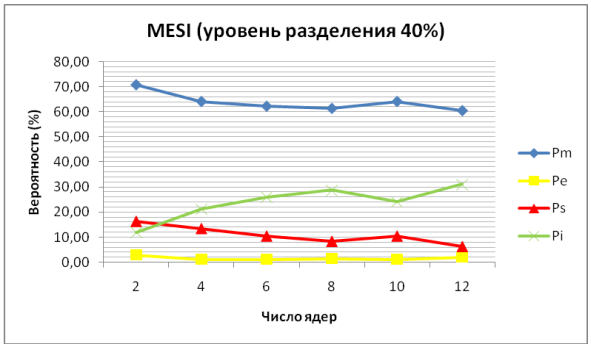


Рисунок 9 – Зависимость вероятности нахождения блоков в различных состояниях протокола MESI от числа ядер при уровне разделения 40%

Как видно из рисунков, для обоих протоколов с увеличением числа ядер вероятности нахождения блоков в состояниях M и S незначительно уменьшаются, в то же время доля блоков в состоянии I заметно увеличивается, причем для более высокого уровня разделения данных эти закономерности прослеживаются четче.

Для протокола MESI доля блоков в состоянии E колеблется от 1% до 3% и не зависит ни от числа ядер, ни от уровня разделения. Следовательно для более детального анализа масштабируемости протокола MESI требуются дополнительные исследования с использованием других моделей рабочей нагрузки.

### *Заключение*

В ходе работы была разработана методика для оценки масштабируемости протоколов когерентности кэш-памяти и проанализирована масштабируемость двух протоколов: MSI и NESI. Согласно разработанной методике для анализа масштабируемости проводятся 3 типа исследований с использование имитационной модели мультипроцессора:

– устанавливается зависимость производительности (иди загрузки) системы от уровня разделения данных и числа ядер процессора;

– устанавливается зависимость производительности (иди загрузки) системы от уровня разделения данных и числа разделяемых блоков памяти нескольких конфигураций процессоров с различным числом ядер;

– для каждого протокола когерентности определяется средняя вероятность нахождения блока кэш-памяти в каждом из состояний, предусмотренным протоколом, для процессоров с различным числом ядер при нескольких фиксированных уровнях разделения данных.

По результатам исследований протокол MESI показал лучшую масштабируемость чем MSI для большинства вариантов рабочей нагрузки, таким образом, введение дополнительного состояния в случае этого протокола оказалось целесообразным.

Практическая значимость работы заключается в том, что разработанная методика в дальнейшем может применяться для анализа масштабируемости оптимизированных протоколов когерентности кэш-памяти.

Программный модуль, реализующий указанную методику, планируется включить в состав имитационно-аналитической системы для анализа эффективности архитектур мультипроцессоров с общей памятью.

### *Список литературы*


1. XPoint cache: scaling existing bus-based coherence protocols for 2D and 3D many-core systems / [Ronald G. Dreslinski, Thomas Manville, Korey Sewall, Reetuparna Das et al] // Proceedings of the 21st international conference on Parallel architectures and compilation techniques, (September, 2012). – 2012. - PP. 75-86.
2. Milo M.K. Martin. Why on-chip cache coherence is here to stay / Milo M.K. Martin, Mark D. Hill, Daniel J. Sorin // Communications of the ACM. - 2012. – Vol. 55, Issue 7. – PP. 78-89.
3. Trent Rolf. Cache Organization and Memory Management of the Intel Nehalem Computer Architecture [Электронный ресурс] / Trent Rolf. - Режим доступа: http://rolfed.com/nehalem/nehalemPaper.pdf.
4. Blake G. A survey of multicore processors / Geoffrey Blake, Ronald G. Dreslinski and Trevor Mudge // IEEE Signal Processing Magazine: Special Issue on Signal Processing on Platforms with Multiple Cores: Part 1 - Overview and Methodology. - 2009. – PP. 27-36.
5. A survey on cache coherence for tiled many-core processor / [Limin Han, Jianfeng An, Deyuan Gao, Xiaoya Fan] // IEEE International Conference on Signal Processing, Communications and Computing (ISCPCC), (August, 2012). – 2012. - PP. 114-118.
6. A composite and scalable cache coherence protocol for large scale CMPs / [Yi Xu, Yu Du, Youtang Zhang, Jun Yang] // Proceedings of the 11th international conference on Supercomputing, (August, 2001). – 2001. – PP. 285-294.
7. Bourmoutian G. Dynamic, multi-core cache coherence architecture for power-sensitive mobile processors / G. Bourmoutian, A. Orailoglu // Proceedings of the 9th International Conference on Hardware / Software Codesign and System Synthesis (CODES+ISSS), (October, 2011). – 2011. - PP. 89-97.
8. State space reduction in modeling checking parameterized cache coherence protocol by two-dimensional abstraction / [Guo Yang, Qu Wanxia, Zhang Long, Xu Weixia] // The Journal of Supercomputing. – 2012. - Volume 62, Issue. – PP. 828 - 854.
9. Robert Slater. Neal Tiberlava – expert consulting & services [Электронный ресурс] / Robert Slater & Neal Tibrewala // Optimizing the MESI Cache Coherence Protocol for Multithreaded Applications on Small Symmetric Multiprocessor Systems. - Режим доступа: http: //tibrewala.net/papers/mesi98.
10. Hackenberg D. Comparing cache architectures and coherency protocols on x86-64 multicore SMP systems / Daniel Hackenberg, Daniel Molka, Wolfgang E. Nagel // Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, 2009. – PP. 413-422.

11. Per Stenstrom. Comparative performance evaluation of cache-coherent NUMA and COMA architectures / Per Stenstrom, Truman Joe, Anoop Gupta // Proceedings of the 19th annual international symposium on Computer architecture, 1992. – PP. 80-91.
12. Орлов С.А. Организация ЭВМ и систем: учебник для вузов / С.А. Орлов, Б.Я. Цилькер. – [2-е изд.]. – СПб.: Питер, 2011. – 688 с.
13. Archibald J. Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model / J. Archibald, J.L. Baer // ACM Transactions on Computer Systems. – 1986. – Vol. 4. – PP. 273-298.
14. Nikolov A. Analytical model for a multiprocessor with private caches and shared memory / Angel Nikolov // Int. J. of Computers. – 2008. - No III. – PP. 172-182.
15. Nikolov A. Comparison of the Performance of Two Service Disciplines for a Shared Bus Multiprocessor with Private Caches / Angel Nikolov, Lerato Lerato // International Journal of Computer Science. - March 2010. - Volume 7, Issue 2.
16. David H. Albonesi. An Analytical Model of High Performance Superscalar-Based Multiprocessors / David H. Albonesi, Israel Koren // Proceedings of the IFIP WG10.3 working conference on Parallel architectures and compilation techniques, 1995. – PP.194-203.
17. Сорока Т.Е. Модель рабочей загрузки для системы моделирования иерархической памяти мультипроцессора и ее применение для оценки эффективности протоколов когерентности кэш-памяти / Т.Е. Сорока, Л.П. Фельдман // Матеріали VII Міжнародної науково-технічної конференції студентів, аспірантів та молодих вчених «Інформатика і комп'ютерні технології», (22-23 листопада 2011р., м. Донецьк). – Донецьк: ДонНТУ, 2011. – С. 244-248.
18. Фельдман Л.П. Имитационная модель для оценки эффективности протоколов когерентности кэш-памяти мультипроцессоров с общей памятью / Л.П. Фельдман, Т.В. Михайлова, Т.Е. Сорока // Сборник научных трудов Донецкого национального технического университета. Серия «Информатика, кибернетика и вычислительная техника». – 2012. – Вып. 15. – С. 64-71.
19. Фельдман Л.П. Оценка эффективности протоколов когерентности кэш-памяти мультипроцессоров с общей памятью с помощью имитационного моделирования / Л.П. Фельдман, Т.Е. Сорока // Материалы XVIII Международной конференции по вычислительной механике и современным прикладным программным системам, (22-31 мая 2013г., Алушта). – М.: Изд-во МАИ, 2013. – 888 с.

*Надійшла до редакції 30.06.2013*

**Т.Є. СОРОКА**
ДВНЗ «Донецький національний технічний університет»

**ДОСЛІДЖЕННЯ МАСШТАБОВАНОСТІ АЛГОРИТМІВ ПІДТРИМКИ КОГЕРЕНТНОСТІ КЄШ-ПАМ`ЯТІ БАГАТОЯДЕРНИХ ПРОЦЕСОРІВ**

Розглянуті способи дослідження і критерії масштабованості алгоритмів підтримки когерентності кеш-пам'яті багатоядерних процесорів. Проведено дослідження масштабованості протоколів когерентності MSI і MESI. За результатами дослідження зроблені висновки про доцільність розширення протоколів когерентності за рахунок введення додаткових станів для різних моделей робочого навантаження на кеш-пам'ять.

***Ключові слова: кеш-пам`ять, протокол когерентності, багатоядерні процесорі, масштабованість.***

**T.Ye. SOROKA**
Donetsk National Technical University

**RESEARCH OF CACHE COHERENCE PROTOCOLS SCALABILITY FOR MULTI-CORE PROCESSORS**

The work is devoted to the research of cache coherence protocols scalability. Designs of modern multi-core processors often include a private cache-memory system for each core, which gives rise to the cache coherence problem. We examined two distributed hardware based protocols (MSI and MESI) for shared bus multiprocessor and evaluated their relative performance on the basis of a simulation model.

The introduction deals with motivation of the research and related works description. The first section describes tested algorithms, formal description of MSI and MESI cache coherence protocols is given. The second section provides a brief overview of methods to research the effectiveness of multiprocessor memory system with reference to relevant sources, we also describe simulation model used for evaluation and workload model for cache-memory system of a multiprocessor with shared memory based on a system bus. The third section presents the results of the experiments and the scalability analysis. Three types of experiments were performed: research of the impact of the number of cores and the level of data sharing on the system performance, the research of the impact of the data sharing level and shared memory blocks number on the system loading for 4-core configuration and the research of the impact of the number of cores on the probability of cache-memory blocks being in states in accordance with the modeling protocol for 20% and 40% data sharing level. By the results of experiments protocol MSI was defined as slightly more scalable than MESI by all the criteria. The proposed methodology of scalability analysis is going to be implemented in the hybrid simulation/analytical model to evaluate the efficiency of multi-core processors memory system architecture.

***Keywords: cache-memory, coherence protocol, multi-core processors, scalability.***