УДК 004.451.26

Т.С. НИКИТИНА

Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Украина

МЕТОД ОЦЕНКИ ТРЕБОВАНИЙ К КЭШ-ПАМЯТИ ДЛЯ ПЕРИОДИЧЕСКИХ ЗАДАЧ В СИСТЕМАХ НА ОСНОВЕ МНОГОЯДЕРНЫХ ПРОЦЕССОРОВ

Рассматриваются проблема влияния неравномерного распределения общей кэш-памяти на производительность в системах на основе многоядерных процессоров. Рассматриваются средства измерения производительности многоядерных процессоров, средства моделирования микропроцессоров с различной архитектурой. Рассматриваются методы равномерного распределения общей кэш-памяти, позволяющие повысить производительность системы на основе многоядерных процессоров. Также, предлагается метод оценки требования к кэш-памяти для периодических задач в системе на основе многоядерных процессоров.

Ключевые слова: *кэш-память, многоядерный процессор, алгоритм планирования, периодические задачи, реальное время*

Введение

На современном этапе переход к многоядерным процессорам (МП) становится основным направлением повышения производительности. Такая тенденция развития процессоров привела к тому, что МП являются частью почти всех вычислительных систем – не только серверов, высокопроизводительных систем, но и применяются в персональных и мини-компьютерах. Такой быстрый переход на параллельные вычисления создал ряд проблемных вопросов в разных областях вычислительной техники. Архитектура МП (рис. 1) в значительной степени отличается от однопроцессорных систем и для эффективного использования нового аппаратного обеспечения возникла необходимость в модификации, как операционных систем, так и программного обеспечения (ПО) [1]. Например, на двуядерном процессоре ПО с отсутствием или неправильной организацией поточной структуры может использовать не более 50 процентов расчетной производительности, в то время как производительность многопоточной программы может увеличиться практически в два раза [2]. Производительность современных систем в достаточной степени стала зависеть от степени оптимизации ПО под МП. Производительность процессора вычисляется следующим образом:

$$P = IPC \times F, \qquad (1)$$

где F – тактовая частота процессора;

IPC – количество инструкций, полученных за такт процессора.

Однако на производительность системы могут оказывают непосредственное влияние как его микроархитектура, так и отдельные характеристики системы.

Как доказывают в работе [3], производительность МП стала в значительной степени зависеть не только от тактовой частоты, но и от объема общей кэш-памяти.

Производительность МП с одинаковой тактовой частотой, но с различным объемом общей кэш-памяти L2 или L3 значительно отличается.

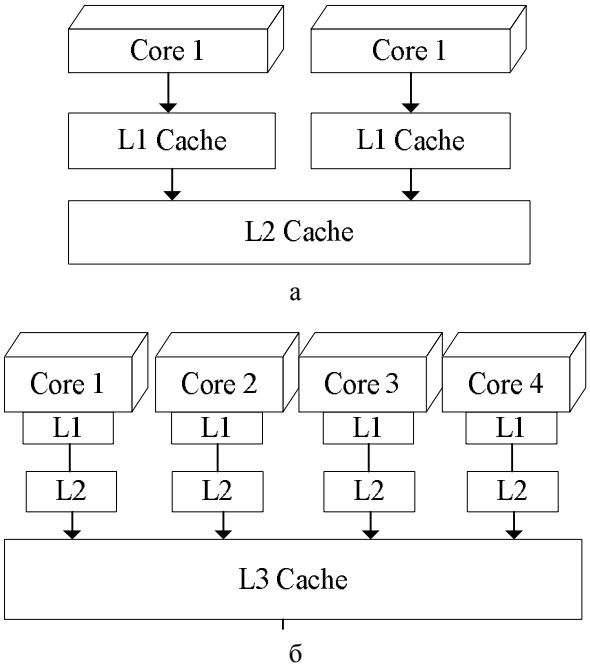

Рис. 1. Архитектура МП
а – Intel Core 2 Duo;
б – AMD Phenom X4, Intel Core i5, i7

© Т.С. Никитина

В ряде случаев большой объем кэш-памяти весьма полезен для ресурсоемких вычислений. Однако с другой стороны, при увеличении кэш-памяти увеличивается время поиска и извлечения данных из него. Вследствие ряда причин кэш-память является весьма ограниченным ресурсом, который в значительной степени влияет на производительность МП.

Цель данной работы – исследовать проблему влияния неравномерного распределения общей кэш-памяти на производительность МП. Также будет предложен метод оценки требования к кэш-памяти для периодических задач в системе на основе многоядерных процессоров.

1. Анализ состояния проблемы

Промах к кэш-памяти возникает, когда процессор не находит необходимую информацию в кэш-памяти и обращается к памяти на уровень ниже. Процессор может простаивать значительное количество циклов, пока необходимые данные не появятся в кэш-памяти. Кэш-память второго уровня L2 считается критическим ресурсом [4] так как, непопадание в кэш L2 вызывает задержки в тысячи циклов процессора. Для сравнения кэш-память первого уровня L1 при непопадании в кэш вызывает задержку лишь несколько десятков циклов процессора. Данная проблема особенно актуальна, так как исполнительные ядра разделяют общую кэш-память неравномерно [5].

Проблема общей кэш-памяти является центральной в задаче организации эффективных параллельных вычислений на основе МП, поскольку существует возможность конфликтов во время доступа к ней. Если планировщик ОС ставит на параллельное исполнение ресурсоемкие процессы, то такая ситуация может приводить к перезаписи одних данных другими в силу ограничения объема общей кэш-памяти. А значит, каждое обращение к памяти на уровень ниже приводит к простою процессора в ожидании необходимых данных.

Рассмотрим пример, пусть ОС функционирует на базе процессора с двумя ядрами и общей кэш-памятью L2 1 Мб. В очереди на выполнение стоят процессы из табл. 1. Пусть процесс bs осуществляет кодирование файла размером 1024 Кб, аналогично crc осуществляет поиск по массиву данных объемом 908 Кб. Процесс stats и matmut работают с небольшим объемом данных. Объемом исполняемых инструкций мы будем пренебрегать, так как он не велик по сравнению с объемом данных. И так, первыми на исполнение стоят bs и crc, их общее требование к памяти практически в два раза превышает общий объем кэш-памяти. В силу того что объем общей кэш-памяти ограничен, такая ситуация будет приводить к перезаписи одних данных другими (которые необходимы для каждого отдельного ядра в большом объеме), что негативно повлияет на скорость исполнения ПО и понизит производительность системы.

Таблица 1

Счетчики МП семейства AMD

Приоритет	Процесс	Объем данных
1	bs	924 Кб
2	crc	908Кб
3	stats	1 Кб
4	matmult	24 Кб

Более удачная комбинация процесса 1 и 3, что исключает переполнение общей кэш-памяти. Однако возникает вопрос, каким образом можно просчитать требуемый объем кэш-памяти для каждого процесса, чтобы эффективно управлять процессами в системе на основе МП.

Уже известны исследования направленные на разработку методов планирования задач, позволяющих равномерно распределять общую кэш-память [6-8]. Данные методы планирования с помощью счетчиков МП измеряют требование к кэш-памяти для каждого процесса. В процессе планирования исключают параллельное исполнение ресурсоемких задач либо добавляют дополнительные кванты времени тем задачам, у которых во время исполнения возникло большое количество промахов к кэш-памяти. Такой подход позволяет работать системе с максимальной производительностью и обеспечивает справедливость планирования процессов для ОС общего назначения. Данные методы планирования уже реализованы в ОС Solaris.

Также вторым и не менее важным классом систем являются ОС реального времени (РВ), где проблема повышения производительности и равномерного распределения общей кэш-памяти также остается актуальной. Однако известные исследовательские работы [9-11] не решают данную проблему для класса систем РВ. Рассмотрим метод, который позволяет получить информацию от требовании к кэш-памяти для задач РВ.

2. Метод оценки требований к кэш-памяти для периодических задач в системе на основе МП

Рассмотрим вопрос о том, каким образом можно идентифицировать ресурсоемкие задачи РВ, которые имеют высокое требование к кэш-памяти. Задачи РВ исполняются с определенным периодом, фактически это идентичные вычисления, повторяемые через некоторый промежуток времени. Для по-

лучения информации о требовании к кэш-памяти задач РВ необходимо сделать следующие последовательные действия:

– обнулить счетчик для подсчета промахов к общей кэш-памяти перед началом исполнения процесса;

– выполнить одну итерацию процесса на одном из ядер, отключив при этом остальные;

– получить информацию о количестве промахов к кэш-памяти для текущего процесса.

Далее, по следующей формуле мы можем получить требуемый объем кэш-памяти для задачи РВ:

$$\mathrm{Task}_{\mathrm{cache\_size}} = \mathrm{cache\_misses} * \mathrm{cache\_line}, \quad (2)$$

где $\mathrm{cache\_misses}$ – количество промахов к общей кэш-памяти;

$\mathrm{cache\_line}$ – размер строк кэш-памяти (от 8 до 512 байт в зависимости от архитектуры процессора).

Если же задача является многопоточной, то необходимо распределить потоки задачи по ядрам в зависимости от архитектуры МП и оценить среднее требование к кэш-памяти для исследуемой задачи:

$$\mathrm{Task}_{\mathrm{cache\_size}} = \frac{\mathrm{cache\_misses}}{\mathrm{thread\_count}} * \mathrm{cache\_line} \quad (3)$$

где $\mathrm{thread\_count}$ – количество потоков в многопоточной задаче.

Такие измерения можно проводить повторно, получая при этом среднее значение, которое может быть более точным. Данная информация о процессах РВ позволит разрабатывать более эффективные методы управления процессами на уровне ОС для достижения максимальной производительности системы. Для исследования данного метода могут использоваться счетчики МП, которые позволяют получать статистические данные о работе системы.

Рассмотрим практические аспекты использования счетчиков МП семейства AMD для измерения производительности. В табл. 2 представлены основные события МП семейства AMD [12]. Основные показатели производительности IPC и CPI (количество циклов на инструкцию), измеряются следующим образом:

$$\mathrm{IPC} = \frac{\mathrm{Ret\_instructions}}{\mathrm{CPU\_clocks}}, \quad (4)$$

$$\mathrm{CPI} = \frac{\mathrm{CPU\_clocks}}{\mathrm{Ret\_instructions}}. \quad (5)$$

Количество промахов к кэш-памяти L3 измеряется следующим образом:

$$\mathrm{L3\_miss\_rate} = \frac{\mathrm{L3\_misses}}{\mathrm{Ret\_Instructions}}. \quad (6)$$

Таким же образом измеряется количество промахов к кэш-памяти L1, L2. Аналогичными счетчиками производительности обладают процессора компании Intel, Sun Microsystems и другие.

Таблица 2

Счетчики МП семейства AMD

Код события	Обозначение события	Описание
0x76	CPU_clocks	Для вычисления IPC и CPI достаточно двух базовых событий
0xC0	Ret_instructions	
0x7F	L2_fill_write	Для вычисления количества промахов к кэш-памяти L2
0x7E	L2_misses	
0x0C0	Ret_instructions	Для вычисления количества промахов кэш-памяти L3
0x4E0	L3_requests	
0x4E1	L3_misses	

Заключение

Проведенный анализ позволяет сделать вывод, что ограниченный ресурс в виде общей кэш-памяти оказывает значительное влияние на производительность современных вычислительных систем. Также возникает необходимость учитывать данную особенность МП на уровне ОС, что позволит максимально эффективно использовать ресурсы процессора. Данная проблематика частично решена для ОС общего назначения, однако для ОС РВ она является открытой, чему и будет посвящена дальнейшая исследовательская работа.

Литература

1. Imbs D. Software transactional memories: an approach for multicore programming / D. Imbs, T. Sampson // SpringerLink. – 2009. – Vol. 8542. – P. 173-183.

2. Multicore Programming Challenges. / M. Perrone // SpringerLink. – 2009. – Vol. 5704. – P. 1-2.

3. Jie T. Performance Advantage of Reconfigurable Cache Design on Multicore Processor Systems. / T. Jie, M. Kunze, F. Nowak, R. Buchty, W. Karl // SpringerLink. – 2008. – Vol. 36. – P. 347-360.

4. L2-Cache Hierarchical Organizations for Multicore Architectures / M. Donato // SpringerLink. – 2006. Vol. 4331. – P. 74-83.

5. Fedorova A. Operating System Scheduling for Chip Multithreaded Processor : dis. Doctor of Philosophy. Computer Science/ . – Massachusetts, 2006. – 199 p. – Бібліогр. : p. 55–140.

6. Task Parallel Scheduling over Multi-core System. / B. Wang // SpringerLink. – 2009. – Vol. 5931. – P. 423-434.

7. Suh G. A New Memory Monitoring Scheme for Memory-Aware Scheduling and Partitioning / G. Suh, S. Devadas, L. Rudolph // High Performance Computer Architecture. – 2002. Vol. 1331. – P. 117-228.

8. Ritson G. Multicore Scheduling for Lightweight Communicating Processes. / G. Ritson, T. Sampson, M. Barnes // SpringerLink. – 2009. – Vol. 5521. – P. 163-183.

9. Seongbeom K. Fair Cache Sharing and Partitioning in a Chip Multiprocessor Architecture / K. Seongbeom, Ch. Dhruba, Y. Solihin // Parallel Architectures and Compilation Techniques. – 2004. Vol. 3311. – P. 94-103.

10. Chandra D. Predicting Inter-Thread Cache Contention on a Multi-Processor Architecture. / D. Chandra, F. Guo, S. Kim, and Y. Solihin. // High Performance Computer Architecture. – 2005. – Vol. 5704. – P. 340-351.

11. Matick R. Analytical Analysis of Finite Cache Penalty and Cycles per Instruction of Multiprocessor Memory Hierarchy Using Miss Rates and Queuing Theory / R. Matick, T. J Heller, and M. Ignatowski // SpringerLink. – 2006. Vol. 4331. – P. 74-83.

12. Basic Performance Measurements for AMD Athlon 64, AMD Opteron and AMD Phenom Processors / J. Drongowski [Електронний ресурс] – Режим доступа: http://developer.amd.com/Assets/Basic_Performance_Measurements.pdf.

Поступила в редакцию 14.01.2010

Рецензент: д-р техн. наук, проф. В.М. Вартанян, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.

МЕТОД ОЦІНКИ ВИМОГ ДО КЕШ-ПАМ'ЯТІ ДЛЯ ПЕРІОДИЧНИХ ЗАДАЧ В СИСТЕМАХ НА ОСНОВІ БАГАТОЯДЕРНИХ ПРОЦЕСОРІВ

Т.С. Нікітіна

Розглядається проблема впливу нерівномірного розподілу загальної кеш-пам'яті на продуктивність багатоядерних процесорів. Розглядаються засоби виміру продуктивності багатоядерних процесорів, засоби моделювання мікропроцесорів з різною архітектурою. Розглядаються методи рівномірного розподілу загальної кеш-пам'яті, що дозволяють підвищити продуктивність системи на основі багатоядерних процесорів. Також пропонується метод оцінки вимоги до кеш-пам'яті для періодичних задач в системах на основі багатоядерних процесорів.

Ключові слова: кеш-пам'ять, багатоядерний процесор, алгоритм планування, періодична задача, реальний час.

METHOD OF ESTIMATING THE REQUIREMENTS OF CACHE-MEMORY FOR PERIODIC TASKS IN MULTICORE SYSTEMS

T.S. Nikitina

Examined the problem of influence shared cache memory on multicore systems performance. Facilities for measuring the performance in multicore systems are examined. The methods of fair cache sharing among cores, allowing to effect on performance are examined. The method of estimating the requirement of cache memory for periodic tasks in multicore systems is considered.

Key words: multicore processor, scheduling algorithm, periodic task, real time, cache memory.

Никитина Татьяна Сергеевна – аспирантка каф. 603 Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: tanya-nt@mail.ru.