УДК 004.254
DOI: 10.17586/0021-3454- 2017-60-10-961-966

ПРОБЛЕМА ЛОЖНОГО РАЗДЕЛЕНИЯ СТРОК КЭШ-ПАМЯТИ ПРОЦЕССОРОВ В МНОГОПРОЦЕССОРНЫХ СИСТЕМАХ

А. О. Кузичкина, М. С. Косяков

Университет ИТМО, 197101, Санкт-Петербург, Россия
E-mail: mkosyakov@gmail.com

Исследована проблема ложного разделения строк кэш-памяти процессоров, обусловленная спецификой организации памяти в современных многоядерных многопроцессорных системах. Ложное разделение строк приводит к тому, что задачи, которые должны выполняться параллельно на нескольких ядрах, выполняются последовательно. Проблему трудно обнаружить, а ее наличие может стать причиной резкой деградации производительности системы в целом. В ходе проведенного исследования выявлены шаблоны программирования, использование которых для решения типовых задач в многопоточных приложениях приводит к ложному разделению. Экспериментальным путем измерено время выполнения шаблонов при наличии ложного разделения и после применения предложенных в работе методов для решения этой проблемы. Проанализированы зависимости ускорения выполнения шаблонов программирования и эффективности использования вычислительных ресурсов системы от числа параллельно работающих потоков. По результатам исследования установлено, что ложное разделение существенно влияет на временные характеристики работы многопоточных приложений, предложены методы его устранения.

Ключевые слова: *многопроцессорные системы, кэш-память, ложное разделение, многопоточное приложение*

В течение последних десяти лет основным направлением развития архитектуры вычислительных систем является проектирование многоядерных многопроцессорных систем [1]. Причиной этому стала необходимость увеличения производительности вычислительных машин, в то время как возможности повышения тактовой частоты практически исчерпаны.

Применение многопроцессорных систем позволяет разработчикам программного обеспечения организовывать параллельные вычисления в рамках одной системы, чтобы ускорить выполнение программ. При проектировании многопоточного приложения необходимо учитывать особенности аппаратного обеспечения, на котором оно будет развернуто, чтобы эффективно распределить функциональность приложения между доступными вычислительными ресурсами.

Важно учитывать факторы, которые могут повлиять на производительность параллельных программ: число и конфигурация процессоров, конкуренция за данные, перебрасывание кэша, ложное разделение строк кэш-памяти процессоров.

Организация кэш-памяти. Для уменьшения задержек доступа к совместно используемой памяти каждый процессор оснащен локальной кэш-памятью, которая значительно быстрее оперативной. Использование кэш-памяти предотвращает необходимость обращения к разделяемой основной памяти в многопроцессорных системах [2].

Кэш-память многоядерных процессоров имеет несколько уровней [3]:

— кэш первого уровня (L1) работает напрямую с ядром процессора, обладает наименьшим временем доступа. Разделен на кэш инструкций (L1i) и кэш данных (L1d);

— кэш второго уровня (L2) также принадлежит конкретному ядру процессора. Этот кэш больше и медленнее, чем кэш первого уровня;

— кэш третьего уровня (L3) является самым большим и медленным, но все же он намного быстрее, чем оперативная память. L3 разделяется между всеми ядрами процессора (в архитектуре процессоров Intel).

Строка кэш-памяти — это минимальный блок данных, который может быть передан между основной памятью и кэшем, ее размер для разных процессоров может различаться, но у большинства x86 он составляет 64 байт. Наличие кэшируемой памяти увеличивает скорость обработки данных, но усложняет работу системы при многопоточной обработке.

Кэш-память каждого ядра процессора заполняется данными, необходимыми для работы потока, выполняющегося на этом ядре. Если потоки, одновременно выполняющиеся на разных ядрах, используют общие данные и модифицируют их, возникает необходимость в механизмах синхронизации, которые гарантируют поочередный доступ к разделяемым данным и целостность содержимого локальных кэшей всех ядер [4]. При операции чтения проблем не возникает, данные просто копируются в кэш-память каждого ядра. Но при модификации данных потоком исполнения на одном ядре изменения должны попасть в кэши других ядер, потоки которых запрашивают доступ к этим данным.

Проблема поддержки целостности данных в кэш-памяти. Кэширование совместно используемых данных требует поддержки их целостности в кэш-памяти. Система памяти сохраняет целостность, если при любой операции чтения элемента данных, выполненной любым из процессоров, возвращается последнее записанное любым процессором значение этого элемента [5]. Таким образом, возникает необходимость синхронизации строк кэш-памяти [2]. Изменения в кэше одного процессора должны быть переданы в кэш другого процессора. Может случиться так, что значение общей переменной будет многократно передаваться из одного кэша в другой. Это явление называют перебрасыванием кэша, оно может серьезно сказаться на производительности приложения [6].

Но даже если к некоторой ячейке памяти в каждый момент времени обращается только один поток, перебрасывание кэша возможно из-за ложного разделения строк кэш-памяти, возникающего, когда множество процессоров одновременно обращаются к разным элементам, находящимся в одной кэш-строке [3]. Эту проблему трудно обнаружить, а ее наличие существенно ухудшает временные характеристики работы многопоточных программ.

В ходе исследования были выявлены шаблоны программирования, использование которых в многопоточных приложениях приводит к ложному разделению. Экспериментально проанализировано влияние ложного разделения кэш-строк на временные характеристики работы параллельных приложений. Эксперименты проводились на сервере, который оснащен двумя восьмиядерными процессорами Xeon E5-2690 2.9GHz [7]; кэш-память процессоров: L1i — 32 кБ, L1d — 32 кБ, L2 — 256 кБ, L3 — 20 МБ. Оперативная память составляет 128 ГБ. Операционная система Oracle Linux 3.8.13.

Для анализа используются следующие метрики. Ускорение — это отношение времени последовательного выполнения программы к времени параллельного выполнения программы [8]:

$$S_N = T_1 / T_N . \qquad (1)$$

Эффективность — показывает, насколько хорошо программа использует вычислительные ресурсы системы. Для расчета эффективности параллельного приложения нужно наблюдаемое ускорение разделить на число используемых ядер N [8]:

$$E_N = \frac{S_N}{N} \times 100\,\% . \qquad (2)$$

Потоки выполняются на разных ядрах, значит, количество используемых ядер совпадает с числом запущенных потоков.

Для измерения времени выполнения программы используется ассемблерная команда rdtsc (Read Time Stamp Counter) для платформы x86, считывающая показания счетчика TSC (Time Stamp Counter) и возвращающая 64-битное количество тактов с момента последнего включения процессора. Полученное число тактов для выполнения программы переводится в секунды.

Разделяемый массив. Типичным шаблоном параллельной программы, в котором случается ложное разделение строк кэш-памяти, является создание разделяемого между потоками массива. Каждый поток заносит результат своей работы в один выделенный ему элемент массива. Параллельное выполнение задачи должно приводить к тому, что с увеличением количества потоков ускорение работы программы будет пропорционально увеличиваться. Было измерено время параллельного выполнения программы с разделяемым массивом: T_1=27, T_2=37, T_5=39, T_6= T_8=38, T_{11}=51, T_{16}=53 мс. Из полученных результатов измерений можно сделать вывод, что параллельная программа работает медленнее однопоточной реализации.

Элементы массива располагаются рядом в памяти и, вероятнее всего, они попадают в одну строку кэш-памяти. И хотя потоки одновременно изменяют разные элементы массива, будет возникать ложное разделение кэш-строк.

Эксперимент выполнялся с использованием шестнадцати параллельно выполняющихся потоков, которые доступны на используемом сервере: от двух до восьми запускалось на разных ядрах одного процессора; от девяти до шестнадцати — на разных ядрах разных процессоров. Время параллельного выполнения программы при распределении потоков по ядрам одного процессора T_8 = 380 мс, в то время как при распределении потоков между двумя процессорами время T_8 = 530 мс. Время выполнения однопоточного варианта рассматриваемой программы T_1 = 270 мс. Время выполнения программы при использовании потоков, запущенных на разных ядрах разных процессоров, значительно выше, чем при использовании ядер внутри одного процессора.

Это можно объяснить тем, что скорость взаимодействия ядер, расположенных на одном кристалле (в непосредственной близости друг от друга), значительно выше скорости передачи, когда используемые ядра расположены на разных кристаллах.

Распределение общей памяти для потоков родительским потоком. Другим шаблоном программирования является распределение родительским потоком общей памяти для его дочерних потоков, работающих независимо друг от друга. Главный поток перед созданием дочерних выделяет один блок памяти для всех структур этих потоков. Каждый дочерний поток обращается к элементам своей структуры. Структуры всех потоков располагаются рядом в памяти, и вероятнее всего, элементы этих структур попадают в одну строку кэш-памяти. И хотя потоки обращаются к несвязанным элементам разных структур, наблюдается ложное разделение кэш-строк.

По результатам данных рис. 1 можно сделать вывод, что временные характеристики работы программы существенно ухудшились с добавлением большего количества потоков (*1* — фактическое, *2* — ожидаемое время выполнения).

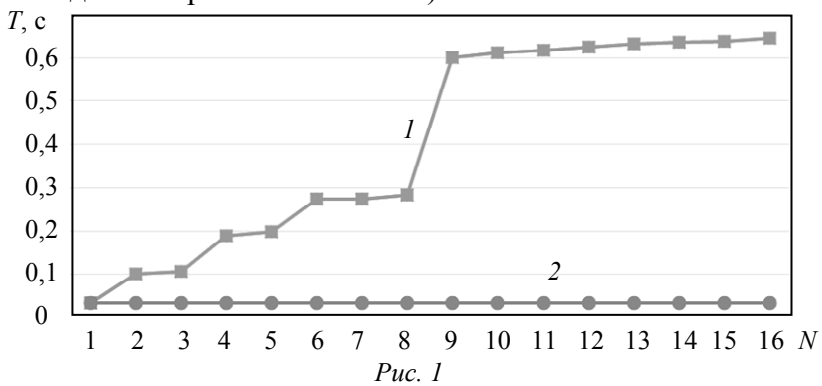

Рис. 1

Очередь. Очередь, реализованная на основе односвязного списка, является еще одним шаблоном параллельной программы, в котором возникает ложное разделение строк кэш-памяти. В данном шаблоне программирования реализована очередь на основе односвязного списка. Указатели head и tail указывают на начало и конец очереди соответственно. Запущены два параллельных потока. Один поток является „читателем“ и удаляет данные из начала очереди, другой — „писателем“ и добавляет новые элементы в конец очереди.

Для однопоточного варианта рассматриваемой программы $T_1 = 50$ мс, при параллельном выполнении программы $T_2 = 170$ мс. Таким образом, ускорение выполнения программы $S_2 = 0{,}3$, а эффективность использования вычислительных ресурсов $E_2 = 14{,}7$ %.

По результатам измерений временных характеристик кода видно, что параллельный код медленнее однопоточной реализации. Ложное разделение строк возникает между находящимися в одной кэш-строке указателями на начало и конец очереди.

Методы решения проблемы ложного разделения строк кэш-памяти. В случае разделяемого массива можно избежать ложного разделения строк кэш-памяти, используя локальные переменные в функции потоков. Из рис. 2 видно, что после преодоления проблемы ложного разделения (*1*) фактическое ускорение работы приложения (*2*) приблизилось к ожидаемому (*3*).

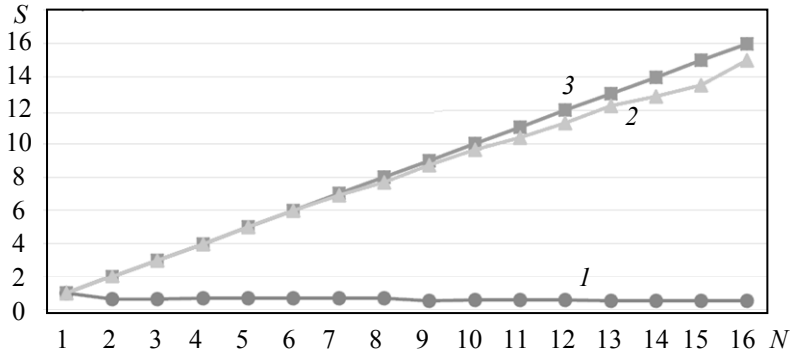

Рис. 2

На рис. 3 приведена зависимость эффективности использования вычислительных ресурсов для выполнения программы с разделяемым массивом от количества потоков (*1* — ложное разделение, *2* — без ложного разделения). Из рис. 3 следует, что после решения проблемы эффективность использования вычислительных ресурсов близка к 100 %.

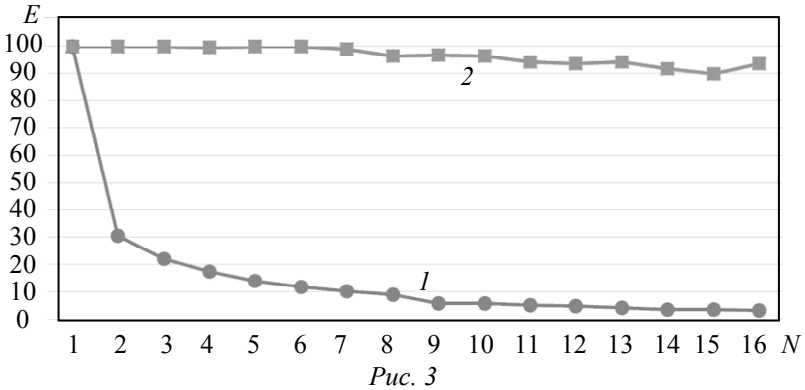

Рис. 3

Другим методом преодоления проблемы является использование выравнивания: между элементами добавляется элемент, размер которого равен размеру кэш-строки. Это гарантирует, что часто изменяемые данные не попадут в одну кэш-строку, а значит, проблема ложного разделения будет устранена.

Результаты измерений после решения проблемы для примера с распределением общей памяти родительским потоком и примера с очередью аналогичны результатам для примера с разделяемым массивом.

Таким образом, установлено, что временные характеристики работы многопоточных программ в случае ложного разделения значительно ухудшаются с добавлением большего количества потоков. На примере запуска шаблонных программ на сервере, оснащенном двумя восьмиядерными процессорами, и использования шестнадцати параллельно работающих потоков продемонстрировано, что при наличии ложного разделения строк кэш-памяти ускорение работы многопоточного кода может быть в 30 раз меньше ожидаемого. Применение предложенных методов позволит существенно улучшить производительность многопоточных приложений.

СПИСОК ЛИТЕРАТУРЫ

1. *Цилькер Б. Я., Орлов С. А*. Организация ЭВМ и систем. СПб: Питер, 2011. 688 с.
2. *Таненбаум Э.* Архитектура компьютера. СПб: Питер, 2007. 843 с.
3. *Drepper U*. What every programmer should know about memory // Red Hat, Inc. 2007. Vol. 11. 114 p.
4. *Venkataramani G., Hughes C. J., Kumar S., Prvulovic M*. Coherence Miss Classification for Performance Debugging in Multi-Core Processors // Proc. of the 13th Workshop on Interaction between Compilers and Computer Architecture. 2009. 10 p.
5. *Hennessy J. L., Patterson D. A*. Computer architecture a quantitative approach. Waltham (Massachusetts): Elsevier, 2012. 848 p.
6. *Уильямс Э*. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ. М.: ДМК Пресс, 2012. 672 с.
7. Спецификации процессора Intel® Xeon® Processor E5-2690 [Электронный ресурс]: <http://ark.intel.com/ru/products/64596/Intel-Xeon-Processor-E5-2690-20M-Cache-2_90-Ghz-8_00-Gts-Intel-QPI>.
8. *Brendan G.* Systems Performance: Enterprise and the Cloud. Upper Saddle River (New Jersey): Prentice Hall, 2013. 729 p.

Сведения об авторах

Анастасия Олеговна Кузичкина — студентка; Университет ИТМО; кафедра вычислительной техники; E-mail: nastya.kuzichkina@gmail.com

Михаил Сергеевич Косяков — канд. техн. наук; Университет ИТМО; кафедра вычислительной техники; E-mail: mkosyakov@gmail.com

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 03.07.17 г.

Ссылка для цитирования: *Кузичкина А. О., Косяков М. С.* Проблема ложного разделения строк кэш-памяти процессоров в многопроцессорных системах // Изв. вузов. Приборостроение. 2017. Т. 60, № 10. С. 961—966.

THE PROBLEM OF FALSE SPLITTING OF CPU CACHE MEMORY STRINGS IN MULTIPROCESSOR SYSTEMS

A. O. Kuzichkina, M. S. Kosyakov

ITMO University, 197101, St. Petersburg, Russia
E-mail: mkosyakov@gmail.com

The organization of memory in modern multi-core multiprocessor systems causes the problem of false split of CPU cash string addressed in the paper. The false CPUs cache lines split makes tasks scheduled to be run in parallel by several cores, are executed sequentially. The problem is difficult to detect, and its presence can cause a sharp degradation of the system performance. Programming patterns which give rise to the problem are identified. The templates execution time is measured in the presence of false splitting and after applying the methods proposed in the work to solve the problem. The effect of the number of threads running in parallel on the speed-up of programming templates execution and the efficiency of utilization of the system computing resources are analyzed. It is shown experimentally that the false split significantly affects temporal characteristics of multi-threaded applications. Several methods to eliminate the difficulties are proposed.

Keywords: multiprocessor systems, cache memory, false splitting, multi-threaded application

Data on authors

Anastasia O. Kuzichkina — Student; ITMO University, Department of Computation Technologies; E-mail: nastya.kuzichkina@gmail.com

Mikhail S. Kosyakov — PhD; ITMO University, Department of Computation Technologies; E-mail: mkosyakov@gmail.com

For citation: Kuzichkina A. O., Kosyakov M. S. The problem of false splitting of CPU cache memory strings in multiprocessor systems. *Journal of Instrument Engineering*. 2017. Vol. 60, N 10. P. 961—966 (in Russian).

DOI: 10.17586/0021-3454- 2017-60-10-961-966