УДК 004.318

Проблемы масштабирования общего кэша многоядерного процессора и методы их решения

Ю.А. Недбайло
«Институт электронных управляющих машин им. И.С. Брука»
«МЦСТ»

Кэши в процессорах играют важную роль, снижая время доступа в память и уменьшая частоту обращений к ней. Большинство многоядерных процессоров имеют, помимо небольших *приватных* кэшей, каждый из которых используется только одним ядром, также и большой *общий* кэш, доступный всем ядрам и разделённый на банки, распределённые по кристаллу. Такое сочетание кэшей гораздо эффективнее, чем использование только приватных, но с ростом количества ядер обостряются ряд проблем его реализации.

Одной из проблем является увеличение времени доступа в общий кэш из-за увеличения среднего расстояния (в тактах) между его банками и ядрами. Значительно уменьшить проблему можно, стараясь хранить данные в банке кэша наиболее близком к ядру, их использующему, а не случайно выбранном на основе физического адреса данных.

Можно выделить два подхода к такой оптимизации хранения, первый из них — привязка страниц памяти к банкам кэша силами ОС, что почти не требует изменений в оборудовании, но имеет ограничения. Второй подход обычно называют *кооперативными кэшами* (cooperative caching), его суть заключается в а) миграции строки в ближайший к ядру банк при обращении к ней и б) перемещении строки в другой банк при вытеснении. [1]

Другой проблемой, возникающей при увеличении количества абонентов кэша, является его *ассоциативность*, особенно при использовании механизмов *разбиения* (partitioning). Когда каждому ядру выделена фиксированная часть объёма кэша, алгоритм замещения кэша часто становится очень ограничен в выборе вытесняемой строки, и его эффективность может снизиться почти до уровня случайного алгоритма.

Повышать ассоциативность увеличением размера сета затратно с точки зрения площади и энергопотребления; аналогичного эффекта без этих затрат можно добиться переходом на *скошенную* (skewed) ассоциативность и оптимизацией механизма замещения известной как ZCache [2]. Механизм разбиения также можно значительно улучшить, например Futility Scaling [3] почти избавлен от указанной проблемы.

Третьей проблемой является поддержка *когерентности* кэшей процессора, то есть хранение в них только актуальных копий. Применение широковещательных сообщений для каждой операции масштабируется плохо, актуальным решением является использование

распределённого *справочника*, хранящего информацию о содержимом всех кэшей процессора. Общий кэш, как правило, берёт на себя его функции.

Кодирование информации справочника является одной из проблем его реализации. Простейший вариант — хранение битовой маски, соответствующей всем кэшам — требует всё больше площади с увеличением числа ядер. Загрубление маски и заведение нескольких записей по одному адресу — известные альтернативы. Сочетая все три варианта можно добиться высокой эффективности и хорошего масштабирования справочника.

Другая трудность в реализации справочника — поддержание его актуальным. Распространённым подходом является вытеснение строки из кэшей при вытеснении соответствующей записи из справочника (back invalidation). При этом не учитывается состояние строк в кэшах, что снижает их эффективность; оптимизации, компенсирующие это, требуют лишних обменов между справочником и кэшами. Хорошо масштабируемым подходом считается [4] оповещение кэшами справочника о вытеснении строк из них.

Ещё одной проблемой масштабирования справочника является сериализация обращений по одному адресу, замедляющая работу, например, барьеров. Если несколько ядер одновременно модифицируют одну кэш-строку, все ядра ждут, пока одно ядро полностью выполнит свою запись, которая занимает десятки тактов. С увеличением числа ядер такие простои наносят всё больший урон производительности параллельных задач. Доработка протокола когерентности позволяет избавиться от этой проблемы [5]

В результате проведённого исследования выяснен набор методов, позволяющих спроектировать хорошо масштабируемый общий кэш.

Литература

1. *R. Balasubramonian, N.P. Jouppi, N. Muralimanohar.* Multi-Core Cache Hierarchies. Synthesis Lectures on Computer Architecture. – Morgan & Claypool, 2011. – 154 c.
2. *Daniel Sanchez, Christos Kozyrakis*. The ZCache: Decoupling ways and associativity // Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO ’43, Washington, DC, USA. – 2010. – C. 187–198
3. *Ruisheng Wang, Lizhong Chen*. Futility scaling: High-associativity cache partitioning // Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-47, Washington, DC, USA. – 2014. – C. 356–367
4. *Milo M. K. Martin, Mark D. Hill, Daniel J. Sorin.* Why on-chip cache coherence is here to stay. – Commun. ACM. – 2012. – 55(7):78–89.
5. *S. Subramaniam, S. C. Steely, W. Hasenplaugh, A. Jaleel, C. Beckmann, T. Fossum, J. Emer.* Using in-flight chains to build a scalable cache coherence protocol. – ACM Trans. Archit. Code Optim. – 2013. – 10(4).