Моделирование выполнения атомарных операций в системах управления на основе многоядерных вычислительных систем

Е. А. Гончаренко[1], А. А. Пазников[2], А. В. Табаков[3]
Санкт-Петербургский государственный электротехнический университет
«ЛЭТИ» им. В. И. Ульянова (Ленина)
[1]kesvesna@rambler.ru, [2]apaznikov@gmail.com, [3]komdosh@yandex.ru

***Аннотация*. Современные сложные системы управления строятся на базе многоядерных вычислительных систем (ВС). Программный инструментарий управления сложными объектами для таких систем разрабатывается в парадигме многопоточного программирования. Одной из наиболее значимых задач, возникающих при организации функционирования таких систем, является синхронизация параллельных потоков. В существующих методах синхронизации потоков используются атомарные операции. Атомарные операции реализованы в виде инструкций на аппаратном уровне и применяются при разработке параллельных программ (построение средств блокировки потоков и неблокируемых структур данных) для использования в сложных системах управления. В работе исследуется зависимость влияния механизма когерентности кэш-памяти (cache coherence) в системах управления на основе многоядерных ВС на время выполнения атомарных операций. Разработана тестовая программа, позволяющая анализировать зависимость пропускной способности и латентности выполнения операций от состояния и расположения кэш-линии. Полученные результаты могут быть использованы при создании новых и оптимизации существующих систем управления на базе распределенных вычислительных систем.**

Ключевые слова: системы управления; моделирование систем управления; алгоритмы управления; атомарные операции; многопоточные программы

I. Введение

Современные сложные системы управления нередко строятся на базе многоядерных вычислительных системам (ВС). Они могут быть представлены как автономные десктопные и серверные системы, так и вычислительные узлы в составе распределенных систем (кластерные ВС, системы с массовым параллелизмом). Такие системы могут включать десятки и сотни процессорных ядер. Например, занимающий первое место в рейтинге TOP500 суперкомпьютер Summit (17 млн. процессорных ядер) включает в себя 4608 вычислительных узлов, каждый из которых укомплектован двумя 24-ядерными универсальными процессорами семейства IBM Power9 и шестью графическими ускорителями NVIDIA Tesla V100 (640 ядер). Суперкомпьютер Sunway Taihulight (более 10 млн. процессорных ядер, второе место в рейтинге TOP500) укомплектован 40 960 процессорами Sunway SW26010, включающими 260 вычислительных ядер. При реализации кэш-памяти применяются различные политики включения (инклюзивный и эксклюзивный кэш), протоколы когерентности (MESI, MOESI, MESIF, MOWESI, MERSI, Dragon) и топологии межпроцессорных шин в NUMA-системах (Intel QPI, AMD HyperTransport).

Одной из наиболее значимых задач при разработке параллельных программ, как для систем с общей памятью, так и для систем с распределенной памятью, является создание эффективных средств синхронизации потоков. Основными методами синхронизации являются средства блокировки потоков (locks) и неблокируемые (non-blocking) потокобезопасные (concurrent, thread-safe) структуры данных. Атомарные операции (atomic operations, atomics) применяются при программной реализации любых видов синхронизации [1-8], в том числе для распределенных вычислительных систем [9]. Операция называется атомарной, если она завершается в один неделимый шаг относительно других потоков. Ни один из потоков не может наблюдать изменение частично завершенным. Говоря иначе, если два или более потоков выполняют операции над разделяемой переменной и хотя бы один из них выполняет операцию записи, то оба потока должны использовать атомарные операции.

При разработке параллельных программ с использованием атомарных операций важно учитывать алгоритмы функционирования кэш-памяти. В частности, необходимо исследовать влияние механизма когерентности кэш-памяти на эффективность выполнения процессорных инструкций, влияние размеров данных, их выравнивание относительно строки кэш-памяти, количества потоков, удаленности данных от ядра (уровни кэш-памяти). В данной работе анализируется эффективность выполнения атомарных операций CAS, FAA, SWP, Load и Store.

К наиболее распространенным атомарным операциям относятся: «сравнение с обменом» (compare-and-swap, CAS), «выборка и сложение» (fetch-and-add, FAA),

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-07-00784 и при поддержке Совета по грантам Президента РФ для государственной поддержки молодых российских ученых (проект СП-4971.2018.5)

«обмен» (swap, SWP), «чтение» (load) и «запись» (store). Аргументами операции «сравнение и замена» (Compare-and-swap, CAS) являются: m – адрес ячейки памяти, r_1, r_2 – регистры процессора. Операция происходит следующим образом: значение m загружается из памяти в регистр r_1, если значение регистра r_1 равно значению в памяти m, то выполняется запись из регистра r_2 в память m. Эта операция используется в структурах данных без блокировок и без ожиданий. Аргументами операции «выборка и добавление» (Fetch-and-add, FAA) являются: m – адрес ячейки памяти, r_1 – регистр процессора. Операция происходит следующим образом: значение m загружается из памяти в регистр r_1, к этому значению в регистре r_1 добавляется еще некоторое значение и результат из регистра r_1 записывается обратно в память m. Данная операция используется при реализации разделяемых счетчиков и примитивов синхронизации, таких как мьютексы и семафоры.

Несмотря на высокую значимость и широкое применение, эффективность атомарных операций не проанализирована в достаточной степени. Например, считается, что операция CAS медленнее, чем FAA [10] и ее семантика позволяет ввести понятие «бесполезная работа» (wasted work) [11], поскольку неудачные попытки сравнения данных в памяти и в регистре приводят к дополнительной нагрузке на ядро при выполнении CAS, а это может быть причиной более низкой производительности в некоторых случаях. В работе [12] рассматривается эффективность выполнения атомарных операций CAS, FAA, SWP с целью анализа влияния архитектурных свойств системы и динамических параметров программы (размеры данных, состояние линии кэш-памяти, число потоков) на скорость выполнения атомарных операций и пропускную способность. Результаты исследований демонстрируют недокументируемые свойства тестируемой системы и скорость производительности атомарных операций. Однако тесты проводились при фиксированной частоте процессорных ядер, использовании больших страниц памяти (huge pages), с выключенной предварительной выборкой данных (prefetching).

В данной работе для получения результатов, максимально приближенных к реальным условиям выполнения параллельных программ, не проводились настройки для фиксирования рабочей частоты ядра, увеличения размера страниц памяти, выключения предварительной выборки данных. Кроме того, рассматриваются разные варианты результаты выполнения CAS (удачный и не удачный), а также выполнение этой операции при использовании данных локальной и удаленной кэш-памяти (по отношению к ядру, выполняющему операции).

Эксперименты проводились на процессоре AMD Athlon 2 X4 640, в котором применяется протокол когерентности кэш-памяти MOESI [13]. Исследуется влияние состояния кэш-линии (M, E, S, O, I) на эффективность выполнения атомарных операций.

Состояние M (Modified): кэш-линия модифицирована процессорным ядром и содержит корректные данные. Копия данных в основной памяти устарела и некорректна, ни один другой процессор не имеет копии данных.

Состояние E (Exclusive): кэш-линия содержит корректные данные. Значение в кэш-памяти совпадает со значением в основной памяти. Ни один другой процессор не имеет копии данных в своем кэше.

Состояние S (Shared): кэш-линия содержит наиболее свежие, корректные данные. Другие процессоры в системе могут иметь копии данных в разделяемом состоянии. Копия в основной памяти также содержит корректную копию данных, если ни один другой процессор не имеет данной кэш-линии в состоянии Owned.

Состояние O (Owned): кэш-линия содержит корректные данные. Только одно ядро может иметь данную кэш-линию в состоянии Owned, все остальные процессоры могут иметь эти данные в состоянии Shared.

Состояние I (Invalid): кэш-линия в состоянии Invalid содержит не корректные данные.

II. Методика проведения экспериментов

В качестве показателей эффективности в данной работе используются латентность выполнения атомарной операции и пропускная способность при этом.

Для измерения латентности l выполнения операции определяется среднее время выполнения отдельной атомарной операции. Данный показатель позволяет оценить время выполнения инструкций при реализации разделяемых счетчиков или флагов для синхронизации, которые используются в параллельных потоках.

Пропускная способность b рассчитывается по формуле $b = n / t$. Для этого оценивается число n выполненных атомарных операций за время t при реализации последовательного доступа к ячейкам буфера. Данный показатель позволяет оценить скорость выполнения инструкций за единицу времени.

Разработана тестовая программа. Перед началом выполнения организуется буфер q в виде целочисленного массива размера s = 128 Мб. Данные помещаются в кэш-память, при этом она переводится в заданное состояние протокола когерентности.

Перевод кэш-линий в состояние M происходит при записи произвольных значений в ячейки буфера. Для перевода в состояние E сначала выполняется запись произвольных значений в ячейки буфера, затем выполняется команда clflush для перевода кэш-линий в состояние I, после этого производится чтение ячеек буфера и кэш-линии переключаются в состояние E. Переход кэш-линий в состояние S происходит после чтения ячеек буфера из кэш-памяти в состоянии E одного ядра в кэш-память другого ядра. Перевод кэш-линий в состояние O достигается путем чтения из кэш-памяти в состоянии M одного ядра в кэш-память другого ядра, при этом кэш-линии ядра из состояния M переключаются в состояние O.

Для CAS выполняется два эксперимента: для успешной и неудачной операции. Неудачным считается CAS, при выполнении которого данные в r_1 не совпадают с данными в m. То есть с момента копирования из m в r_1 эти данные были изменены. Заведомо неудачное выполнение CAS

достигается путем сравнения адреса указателя с данными по этому указателю. В успешном CAS сравнивается значение в r_1 с данными в m.

III. Результаты экспериментов

Эксперименты выполнялись на многоядерной ВС, укомплектованной четырёхъядерным процессором AMD Athlon 2 X4 640 (архитектура Propus). Структура кэш-памяти включает два уровня: первый уровень – расположенная на ядре кэш-память данных L1d (64 Кб) и кэш-память инструкций L1i (64 Кб), второй уровень – эксклюзивная кэш-память L2 (512 Кб) (рис. 1).

Рис. 1. Архитектура Propus

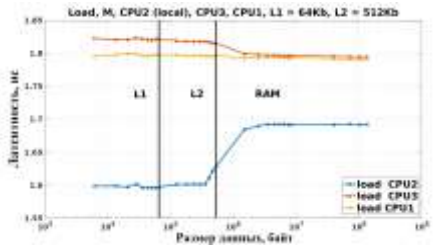

Рис. 2. Операция Load в состоянии кэш-линии M

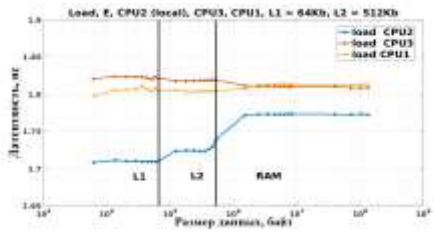

Рис. 3. Операция Load в состоянии кэш-линии E

CPU2 – ядро, которое использует локальные данные своей кэш-памяти (L1i/L1d). CPU3 – ядро, которое находится рядом с CPU2 и использует данные из кэш-памяти CPU2. CPU1– удаленное по отношению к кэш-памяти CPU2. Выполнялись измерения латентности выполнения операций Load, Store, SWP, FAA, «неудачный CAS», «успешный CAS». Операции выполнялись для кэш-линий в состояниях M, O, E, S, I при использовании данных локального кэша ядра CPU2, как самим CPU2, так и ядром CPU3.

Для данных, имеющих размер L1 64Кб, тест выполнялся с числом итераций 10000. Для данных, имеющих размер L2 512Кб, тест выполнялся с числом итераций 1000. Для данных, имеющих размер, превышающий L2, количество итераций в тесте – 100.

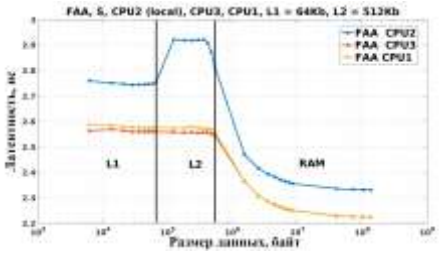

Рис. 4. Операция FAA в состоянии кэш-линии S

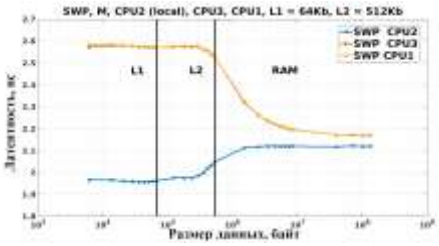

Рис. 5. Операция SWP в состоянии кэш-линии M

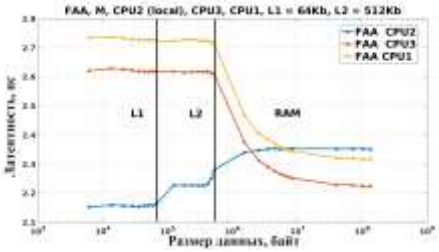

Рис. 6. Операция FAA в состоянии кэш-линии M

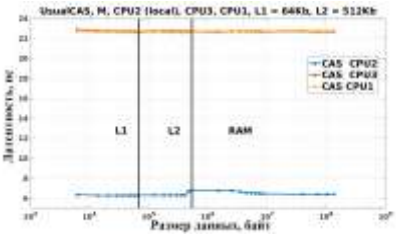

Рис. 7. Операция «успешный CAS» в состоянии кэш-линии M

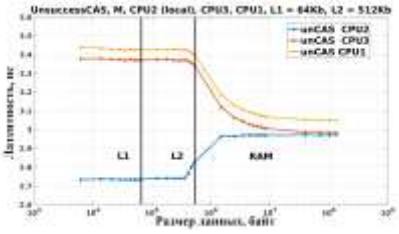

Рис. 8. Операция «неудачный CAS» в состоянии кэш-линии M

Из результатов экспериментов (рис. 2, рис. 3, рис. 5–8) видно, что все атомарные операции выполняются с меньшей латентностью в состояниях кэш-линии M и E на CPU2, кроме операции Store в состоянии E (рис. 9): операция имеет наименьшую латентность выполнения на CPU2 пока данные не превышают размеры L1 и L2.

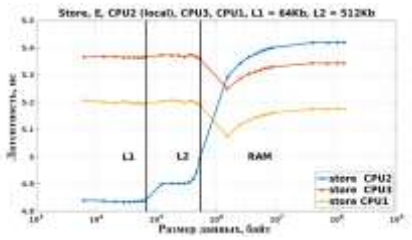

Рис. 9. Операция Store в состоянии кэш-линии E

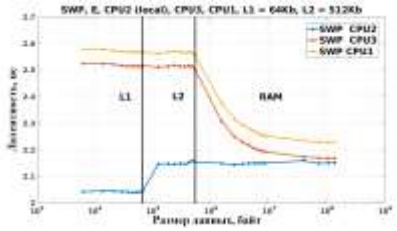

Рис. 10.Операция SWP в состоянии кэш-линии E

Операция Load вне зависимости от состояния кэш-линии имеет латентность меньше других операций, абсолютная величина ее латентности составляет от 1,6 до 1,75 нс. На ядрах CPU3 и CPU1 наименьшую латентность имеют операции FAA (2,2 – 2,6 нс), Store (5,0 – 5,25 нс) в состоянии кэш-линии S и O. С максимальной латентностью выполняется операция «успешный CAS» на CPU3 и CPU1 с максимальной задержкой 35 наносекунд в состояниях кэш-линии S и O. При выполнении операции FAA и состоянии кэш-линии S (рис. 4), задержка выполнения на CPU2 больше, чем на CPU3 и CPU1 в худшем случае на 0,3 наносекунды. Операция SWP в состоянии кэш-линий E (рис. 10) при выполнении на CPU1 имеет большую латентность по сравнению с CPU3, при этом если размер данных больше 100Мб, то CPU2 и CPU3 имеют сопоставимую латентность выполнения операции с абсолютным значением около 2,16 наносекунд.

IV. Заключение

В данной работе для систем управления сложными объектами на основе многоядерных вычислительных систем представлен анализ эффективности атомарных операций (CAS, FAA, SWP, Load, Store). Показано, что «неудачный CAS», FAA и SWP характеризуются сопоставимой задержкой. «Успешный CAS» при любых размерах данных и состояниях кэш-линий выполняется быстрее на CPU2 локальном ядре. Операция FAA имеет меньшую латентность выполнения на локальном ядре, в состояниях кэш-линий E и M, в среднем на 0,4 наносекунды по сравнению с латентностью выполнения этой операции на CPU3 ближнем и CPU1 удаленном ядре. На удаленном и ближнем операция имеет наименьшую латентность выполнения в O и S, причем на локальном при этих состояниях L2 всегда медленнее L1 и RAM.

Операция «неудачный CAS» имеет наименьшую латентность выполнения на локальном в любых состояниях, если размер данных не превышает размеры L1 и L2. Операция SWP имеет наименьшую латентность выполнения на локальном при любом размере данных в состояниях кэш-линий M и E. Операция Store при больших размерах данных, больше L1 и L2, имеет меньшую латентность при выполнении на удаленном и ближнем ядре. В состояниях S и O локальное выполняет эту операцию всегда с большей латентностью, чем удаленное и ближнее.

Представленные результаты моделирования могут быть использованы для разработки более эффективных параллельных программ для распределенных систем управления (выбор субоптимальных алгоритмов при реализации разделяемых структур данных и примитивов синхронизации) и при создании новых протоколов реализации кэш-памяти в системах управления.

Список литературы

[1] Herlihy M., Shavit N. The art of multiprocessor programming. Morgan Kaufmann. 2012. 537 p.

[2] Paznikov A., Shichkina Y. Algorithms for Optimization of Processor and Memory Affinity for Remote Core Locking Synchronization in Multithreaded Applications, Information. Vol. 9. I. 1. 2018. P. 21.

[3] Paznikov A.A., Pavsky K.V., Pavsky V.A., Kupriyanov M.S. Simulation of the algorithms for optimization of remote core locking method for multicore computer systems // in Proc. of the II International Conference on Control in Technical Systems (CTS). 2017. pp. 51-54.

[4] Пазников А.А. Оптимизация делегирования выполнения критических секций на выделенных процессорных ядрах // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2017. № 38. С. 52-58.

[5] Аненков А.Д., Пазников А.А. Алгоритмы оптимизации масштабируемого потокобезопасного пула на основе распределяющих деревьев для многоядерных вычислительных систем // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2017. № 39. С. 73-84.

[6] Кулагин И.И., Курносов М.Г. Оптимизация обнаружения конфликтов в параллельных программах с транзакционной памятью // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2016. Т. 5. №. 4. С. 46-60.

[7] Смирнов В.А., Омельниченко А.Р., Пазников А.А. Алгоритмы реализации потокобезопасных ассоциативных массивов на основе транзакционной памяти // Известия СПбГЭТУ «ЛЭТИ». 2018. № 1. С. 12-18.

[8] Tabakov A.V., Paznikov A.A. “Algorithms for Optimization of Relaxed Concurrent Priority Queues in Multicore Systems,” in Proc. of the 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2019, pp. 360-365.

[9] Paznikov A.A., Anenkov A.D. Implementation and Analysis of Distributed Relaxed Concurrent Queues in Remote Memory Access Model // in Proc. of the 13th International Symposium “Intelligent Systems – 2018” (INTELS’18), Procedia Computer Science. Vol. 150. 2019. P. 654-662.

[10] Meeker W.Q., Escobar L.A. Statistical methods for reliability data. John Wiley & Sons, 2014.

[11] Bordes L., Paroissin C., Salami A. Parametric inference in a perturbed gamma degradation process //arXiv preprint arXiv:1005.1214. 2010.

[12] Schweizer H., Besta M., Hoefler T. Evaluating the cost of atomic operations on modern architectures // PACT. 2015. P. 445-456.

[13] Molka D., Hackenberg D., Schöne R. Main memory and cache performance of Intel Sandy Bridge and AMD Bulldozer // MSPC. – ACM, 2014. С. 4.