УДК 004.4

РЕАЛИЗАЦИЯ ПРИМИТИВОВ СИНХРОНИЗАЦИИ ДЛЯ ЛЕГКИХ ПОТОКОВ В JAVA НА NUMA-АРХИТЕКТУРЕ

Коробейников Н.А. (Университет ИТМО)
Научный руководитель – доцент, PhD, Аксенов В. Е.
(Университет ИТМО)

Введение. С появлением многоядерных систем, перед программистами встала задача создания эффективных и масштабируемых многопоточных алгоритмов. Стандартным способом распараллеливания программ являются потоки, которые в свою очередь делятся на два типа: потоки ОС и легкие потоки. Как известно, некорректное использование потоков может привести к проблемам, не возникающим в однопоточных реализациях. Одной из таких проблем является состояние гонки, когда потоки могут производить параллельные записи и чтение в общей памяти. Состояние гонки плохо тем, что может привести к недетерминированному результату программы, нарушению консистентности данных, и даже к уязвимостям. Поэтому важно использовать примитивы синхронизации, например, блокировки.

Эффективный алгоритм блокировки крайне платформо зависим. Немалую роль в выборе блокировки играет архитектура памяти. Архитектуры памяти делятся на два типа: NUMA и UMA.

UMA или однородный доступ к памяти - это архитектура памяти, в которой все CPU имеют одинаковую скорость доступа к памяти. UMA имеет простое устройство, и для эффективного использования не требует специально написанных программ.

NUMA-архитектура (Non-Uniform memory access) или архитектура неоднородного доступа к памяти – это архитектура памяти компьютерной системы, при которой процессоры имеют разное время доступа к разным частям памяти. У каждого процессора или у группы процессоров есть локальная память, обращение к которой, в отличие от удаленной памяти, производится быстро.

Существует большое число научных статей, предлагающих различные алгоритмы эффективных блокировок для архитектуры неоднородной памяти [1][2][3]. Однако существующие реализации примитивов синхронизации на языке Java не учитывают особенности NUMA-архитектуры.

Реализация эффективной блокировки также зависит от потоков, которые будут ее использовать. В Java до недавнего времени существовали только тяжелые потоки. Однако в Java 19 в режиме preview появились легкие потоки. Они работают поверх платформенных потоков, и эффективно их переиспользуют. Ядро операционной системы ничего не знает о существовании легких потоков. Так как виртуальный поток не привязан к какому-то определенному потоку ОС, то система может поддерживать их миллионами. Данное нововведение в язык имеет большой потенциал, поэтому было принято решение реализовать блокировки для легких потоков в Java, учитывающие особенности NUMA архитектуры.

Основная часть. Целевым процессором для оптимизации блокировок выбраны процессоры линейки Kunpeng-920 на базе архитектуры Arm, обладающие двух и трехуровневой иерархической архитектурой доступа к памяти.

Для оценки эффективности примитивов синхронизации в первую очередь необходимо реализовать инструмент, который будет измерять пропускную способность блокировок и соответствующие накладные расходы. Инструмент поддерживает измерения как для платформенных потоков (тяжелых), так и для виртуальных потоков (легких). Инструмент был реализован при помощи библиотеки JMH (Java microbenchmark harness). Для измерения нужно заранее определить суммарное количество взятия критических секций и работу. Потоки ждут друг друга на барьере, а затем в цикле заходят в критическую секцию, где выполняют фиксированную работу. Количество работы делится на время исполнения, и таким образом мы получаем пропускную способность. Чтобы оценить накладные расходы,

нужно из времени работы с блокировками вычесть время выполнения на одном потоке без блокировок.
Затем нужно реализовать эффективные примитивы синхронизации для платформенных потоков, так как виртуальные работают поверх них. Было реализовано несколько блокировок на языке Java, такие как CNA [1], MCS [2], HMCS [3]. Некоторые блокировки были модифицированы под особенности целевого процессора. Основной идеей для реализации таких оптимизированных алгоритмов блокировки является снижение числа обращений ядер в удаленную память, воспользовавшись информацией об иерархии доступа памяти в процессоре.
Были произведены замеры пропускной способности и накладных расходов блокировок на платформенных потоках.
Затем блокировки, показавшие хороший результат на платформенных потоках, были модифицированы для работы с виртуальными потоками. Виртуальный поток исполняется на платформенном, поэтому мы можем воспользоваться информацией о текущем NUMA-узле и снизить количество обращений в удалённую память. Также особое внимание нужно уделить тому, что в системе может существовать несколько миллионов виртуальных потоков. Были произведены замеры пропускной способности и накладных расходов для реализованных блокировок и для блокировок из стандартной библиотеки Java на легких потоках.
Результаты были получены на процессорах Kunpeng с различным количеством ядер и некоторые реализации (например модификации HMCS) оказались быстрее стандартных.

Выводы. Были реализованы NUMA-aware примитивы синхронизации на языке Java для легких потоков. Для оценивания эффективности блокировок, были созданы инструменты для измерения пропускной способности и накладных расходов. Были произведены замеры, и некоторые реализации оказались более эффективными, чем стандартные блокировки в Java. Реализованные блокировки могут использоваться в Java приложениях, которые будут работать на NUMA процессорах.

Список использованных источников:

1. Dave Dice and Alex Kogan. Compact NUMA-Aware Locks. In Proceedings of the Fourteenth EuroSys Conference 2019, EuroSys ’19, New York, NY, USA, 2019. Association for Computing Machinery.
2. J. M. Mellor-Crummey and M. L. Scott. Algorithms for Scalable Synchronization on Shared-memory Multiprocessors. ACM Trans. Comput. Syst., Feb. 1991.
3. Milind Chabbi, Michael Fagan, and John Mellor-Crummey. High performance locks for multi-level NUMA systems. In Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), 2015.

Коробейников Н.А. (автор) Подпись

Аксенов В.Е. (научный руководитель) Подпись