УДК 004.4

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ПРИМИТИВОВ СИНХРОНИЗАЦИИ В СИСТЕМАХ С РАЗЛИЧНОЙ РЕАЛИЗАЦИЕЙ ЛЕГКИХ ПОТОКОВ

Скаженик Т.М. (ИТМО)
Научный руководитель – PhD, науки Аксенов В.Е. (ИТМО)

Введение. Продолжительное развитие индустрии программного обеспечения и вычислительных устройств привело к тому, что на сегодняшний день для наиболее эффективного использования аппаратных ресурсов программы вынуждены выполнять многие из своих операций параллельно. Базовой абстракцией, применяемой для этой цели, служит поток. Многие годы в мире доминировали тяжелые потоки, эквивалентные или сопоставимые по эффективности с потоками операционной системы. Но в последнее время стали набирать популярность легкие потоки, исполняемые поверх тяжелых потоков в пространстве пользователя и обеспечивающие малые затраты на переключение контекста исполнения. В одних языках, таких как Kotlin, Java, Go, они уже стали частью языка, а в других, например, C++, существует множество библиотек, которые их реализуют.

Одной из классических проблем, с которой сталкиваются при написании многопоточного кода, является разграничение доступа к разделяемым ресурсам. Традиционным решением данной задачи служит примитив синхронизации – мьютекс. История его развития началась еще в конце прошлого века с появлением первых простых реализаций и продолжается до сих пор в виде создания более сложных конструкций, учитывающих различные аспекты систем исполнения. Но основной фокус таких исследований до сих пор сосредоточен преимущественно на примитивах для тяжелых потоков. Поэтому исследование алгоритмов синхронизации в контексте множества особенностей легких потоков остается в стороне, что оставляет определенный простор для данной работы.

Основная часть. В данном исследовании был проведен сравнительный анализ эффективности всех наиболее известных примитивов синхронизации, обеспечивающих взаимное исключение исполнения критических секций кода.

Для определения границ исследования прежде всего были зафиксированы системы, предоставляющие способы для работы с легкими потоками. На первом этапе они были разделены на две группы: с реализацией легких потоков на уровне языка и с помощью сторонних библиотек. Для первой группы характерна меньшая гибкость, поэтому был выбран один пример – Java Virtual Threads. Для второй группы, напротив, свойственна возможность настройки, вследствие чего был выбран язык C++ с тремя различными вариантами реализации легких потоков. Первая из них предоставляется экосистемой Argobots [1], отмеченной в ряде научных публикаций и используемой в рамках крупных проектов с открытым исходным кодом. Вторая – фреймворком Userver для создания высоконагруженных приложений от компании Yandex. А третья – библиотекой Boost Fiber, наработки из которой имеют шансы на включение в стандарт языка.

Вторым этапом стало определение набора тестируемых мьютексов. Для справедливого сравнения в него вошли как простые алгоритмы, например, TestTestAndSet или TicketLock, так и иерархические примитивы: MCS [2], HMCS [3], CNA [4]. Кроме того, были рассмотрены различные их комбинации. Стоит отметить, что для большинства приведенных алгоритмов нет эталонных реализаций в рассматриваемых языках программирования, поэтому для проведения анализа потребовалось, во-первых, написать реализацию самостоятельно, во-вторых, адаптировать алгоритмы к работе в условиях кооперативной многозадачности.

На третьем этапе был зафиксированы условия, в рамках которых предстояло сравнить выбранные примитивы. В качестве метрик для оценки эффективности работы примитива были выбраны: пропускная способность, задержка взятия блокировки и ее честность, определяемая числом взятых блокировок вне очереди. Для сравнения с показателями из научных работ были заимствованы классические сценарии работы с памятью и защитой стандартной структуры данных, такой как дерево, с помощью мьютекса. Кроме того, были разработаны специальные сценарии, подчеркивающие влияние методов синхронизации на прогресс других потоков в

рамках кооперативной многозадачности. К таким сценариям можно отнести запуск множества легких потоков внутри критической секции, а также симуляцию посылки асинхронных задач другим потокам под блокировкой.

На заключительном этапе был проведен поиск и выделение конструктивных особенностей мьютексов, наличие или отсутствие которых значительно влияет на эффективность примитива синхронизации при работе в системах с легкими потоками.

Выводы. В рамках работы было выбрано множество реализаций легких потоков, оказывающих влияние на крупные проекты в индустрии. После чего был сформирован и реализован набор из большинства известных алгоритмов синхронизации и их комбинаций. Этот набор был протестирован на различных сценариях во всех обозначенных системах под высокой нагрузкой. На основании полученных данных был сформулирован перечень конструктивных решений, оказывающий заметное влияние на производительность кооперативной системы. Выводы и реализованные алгоритмы могут быть использованы в дальнейших исследованиях по созданию универсальных мьютексов, эффективных в различных окружениях под высокими нагрузками.

Список использованных источников:

1. Argobots: A Lightweight Low-Level Threading and Tasking Framework / S. Seo [и др.] // IEEE Transactions on Parallel and Distributed Systems. — 2018. — Т. 29, No 3. — С. 512–526
2. Mellor-Crummey J. M., Scott M. L. Algorithms for Scalable Synchronization on Shared-Memory Multiprocessors // ACM Trans. Comput. Syst. — New York, NY, USA, 1991. — Февр. — С. 21–65.
3. Chabbi M., Fagan M., Mellor-Crummey J. High Performance Locks for Multi-Level NUMA Systems // SIGPLAN Not. — New York, NY, USA, 2015. — Янв. — Т. 50, № 8. — С. 215–226.
4. Dice D., Kogan A. Compact NUMA-Aware Locks // Proceedings of the Fourteenth EuroSys Conference 2019. — Dresden, Germany : Association for Computing Machinery, 2019. — (EuroSys ’19).