МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе № 2

по дисциплине «Параллельные алгоритмы»

Тема: Реализация потокобезопасных структур данных с блокировками

Студент гр. 0304 _____ Шквиря Е.В.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2023

Цели работы.

Изучить принцип построения потокобезопасных структур данных с блокировками.

Постановка задачи.

Реализовать итерационное (потенциально бесконечное) выполнение подготовки, обработки и вывода данных по шаблону “производитель-потребитель” (на основе лаб. 1 (части 1.2.1 и 1.2.2)). Обеспечить параллельное выполнение потоков обработки готовой порции данных, подготовки следующей порции данных и вывода предыдущих полученных результатов.

Использовать механизм “условных переменных”.

2.1

Использовать очередь с “грубой” блокировкой.

2.2

Использовать очередь с “тонкой” блокировкой.

Выполнение работы.

Были реализованы два класса - `ThreadSafeHardQueue` и `ThreadSafeFineQueue`. Первый - очередь с “грубой” блокировкой, второй - очередь с “тонкой” (мелкогранулярной) блокировкой. Каждый из них содержит в себе минимальный интерфейс для работы с очередью, а именно: метод `push` (положить элемент в конец очереди) и метод `popWithWaiting` (взять элемент из начала очереди). Рассмотрим каждую из вариаций очереди более подробно.

Очередь с “грубой” блокировкой представляет из себя обёртку над стандартной очередью, операции которой будут дополнительно синхронизироваться с помощью условной переменной и мьютекса. При добавлении элемента в очередь мьютекс будет блокироваться, а условная

переменная получать информацию, что очередь в данный момент не пустая. При получении и удалении элемента из очереди мьютекс также будет блокироваться, но при этом также будет происходить проверка, пуста ли очередь. Если очередь действительно пустая, то выполнение блокируется до момента, когда условная переменная получит сигнал, что в очереди появился элемент. Если очередь в момент получения элемента уже не пуста, то элемент извлекается сразу.

Очередь с “тонкой” блокировкой представляет собой однонаправленный список, который также поддерживает операции добавления элемента в конец очереди и извлечение элемента из начала очереди. Отличием от прошлой вариации очереди в том, что теперь у нас будут задействованы два мьютекса (на начало и на конец очереди) и условная переменная. При добавлении элемента блокируется мьютекс хвоста, после чего к хвосту добавляется новый элемент. Когда хвост изменился и элемент добавился - условная переменная получает информацию об этом событии. Извлечении элемента из очереди влечёт за собой блокировку мьютекса головы, после чего проверяется, пуста ли очередь в данный момент. Когда очередь не пуста - данные извлекаются.

Анализ

Перед переходом к вычислениям стоит упомянуть, что работа выполнялась на машине с количеством ядер процессора равным 6 и максимальным числом потоков равным 12. Результат одного замера получается как среднее между 10-ю выполнениями полного цикла эксперимента (от создания производителей/потребителей до их окончания работы), генерация матриц в расчёт не бралась.

За основу бралось перемножение двух квадратных матриц размером 300x300, рассматривались перемножения при различных значениях количества производителей и потребителей. За результат бралось время работы в микросекундах. Результаты приведены в таблице 1 и 2.

Таблица 1 - Очередь с “грубой” блокировкой

C \ P	5	10	15	20
5	226678	195817	180526	178807
10	249802	203297	179979	167131
15	250428	211113	197065	165655
20	259571	224624	212432	186514

Таблица 2 - Очередь с “тонкой” блокировкой

C \ P	5	10	15	20
5	227580	206442	205720	209875
10	210515	240066	257401	264066
15	215869	259096	282113	290876
20	220439	278875	308858	329071

Как можно заметить, по результатам очередь с “грубой” блокировкой выигрывает, если количество производителей превышает или равно количеству потребителей. С другой стороны, очередь с “тонкой” блокировкой выигрывает, когда количество потребителей превышает количество производителей.

Выводы.

В ходе выполнения лабораторной работы были изучены потокобезопасные очереди с различными видами блокировок при их работе по шаблону производитель/потребитель. Для этого были разработаны и проанализированы две очереди с блокировками - “грубая” и “тонкая” (мелкогранулярная).

Было выявлено, что при равном количестве производителей и потребителей или при большем количестве первых - очередь с “грубой”

блокировки выполняет работу быстрее, чем очередь с “тонкими” блокировками. В случае, когда количество потребителей больше, чем производителей - очередь с “тонкими” блокировками работает быстрее.

Объясняется это тем, что при “тонкой” блокировки выделение памяти под очередной элемент происходит дольше, чем если бы мы его просто клали в обычную очередь. С другой стороны, в очереди с “грубой” блокировкой когда у нас происходят постоянные блокировки при работе производителей - потребители тоже не могут выполнять свою работу. Из-за этого долгое выделение памяти при “тонких” блокировках оказывается не настолько значительным на фоне постоянных блокировок всей очереди.