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

ОТЧЁТ

по лабораторной работе № 3

по дисциплине «Параллельные алгоритмы»

Тема: «Реализация потокобезопасных структур данных без блокировок»

Студент гр.1304 _____ Поршнев Р.А.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург

2024

Задание

1. Реализовать очередь, удовлетворяющую lock-free гарантии прогресса (очередь Майкла-Скотта). Протестировать доступ к реализованной структуре данных для случаев разного количества потоков производителей и потребителей (аналогично работе 2).

2. Дополнительно (выполнение работы на отл.): реализовать корректное освобождение памяти.

3. Сравнить производительность с реализациями структур данных из работы 2.

Выполнение работы.

В ходе выполнения работы реализован класс lock_free_queue, который содержит следующие приватные поля:

• std::atomic<Node*> head – указатель на "голову" односвязного списка, объявленный как атомарный тип;

• std::atomic<Node*> tail – указатель на "хвост" односвязного списка, объявленный как атомарный тип.

Также данный класс имеет следующие методы:

• void enqueue(matrices someMatrcies) – данный метод принимает на вход структуру, которая содержит в себе левую матрицу, правую, а также матрицу, которая является их произведением, и предназначен для добавления данной структуры в односвязный список;

• void dequeue(matrices &value) – данный метод принимает на вход ссылку на переменную, в которую будет записан результат извлечения матриц из односвязного списка;

Класс lock_free_queue имеет конструктор, в котором инициализируется "голова" и "хвост" односвязного списка, а также деструктор, в котором последовательно удаляются узлы списка.

Для освобождения памяти при инициализации класса очереди сохраняется указатель на начальную "голову" очереди, что позволяет при вызове деструктора последовательно и корректно освободить память односвязного списка.

Остальная часть кода аналогична предыдущей лабораторной работе.

Также было проведено сравнение очереди Майкла-Скотта, "тонкой" и "грубой" блокировок при следующей конфигурации:

- размеры матриц: 50 на 50;
- количество пар матриц для каждого потока: 100;
- размер буфера очереди: 10;
- матрицы состоят только из -1;
- умножение матриц производится с помощью четырёх потоков;

Зависимость времени умножения матриц от количества потребителей/производителей (их количество равно) представлена на Рисунке 1.

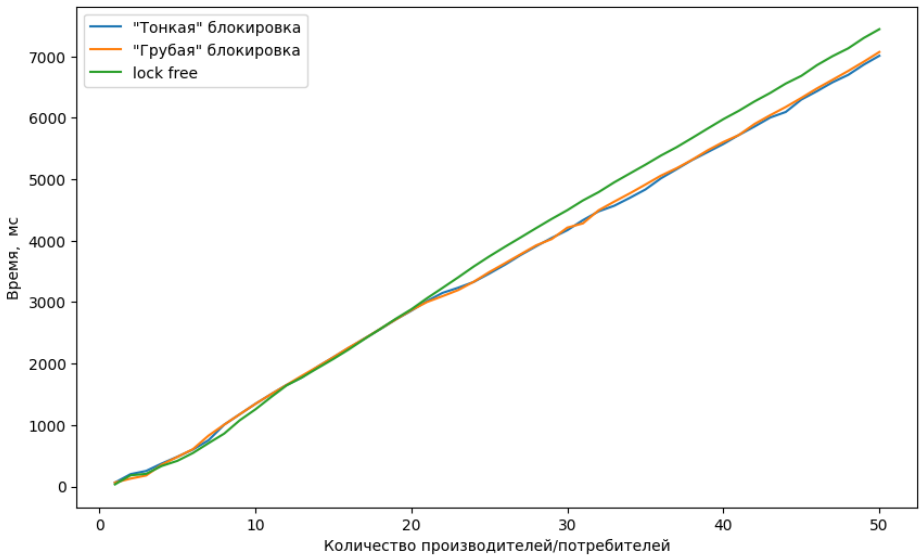

Рисунок 1 – Зависимость времени умножения матриц от количества потребителей/производителей

Исходя из полученных данных на Рисунке 1, при данной конфигурации очередь Майкла-Скотта лучше очередей с "грубой" и "тонкой" блокировками при количестве производителей и потребителей до 23, а далее она начинает показывать наихудшие результаты среди всех алгоритмов.

Зависимость времени умножения матриц от количества потребителей (количество производителей всегда равно 10) представлена на Рисунке 2.

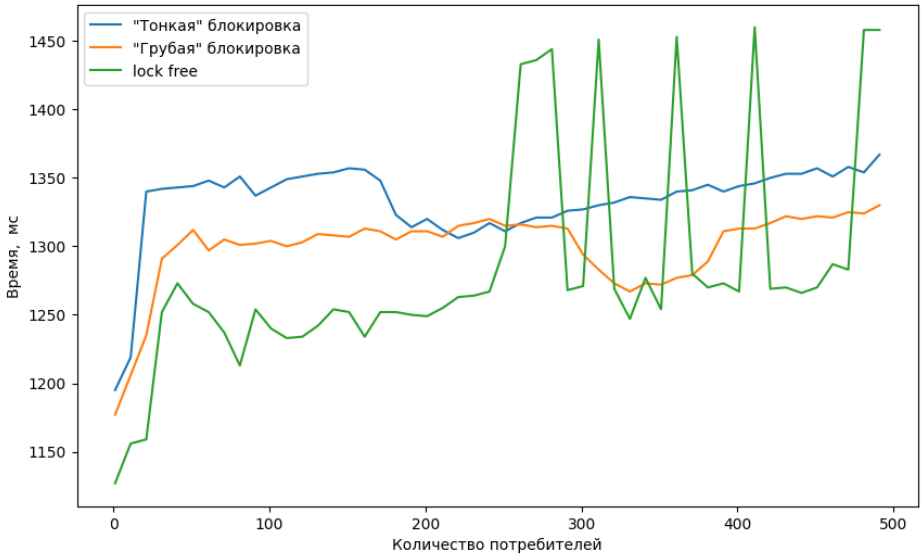

Рисунок 2 – Зависимость времени умножения матриц от количества потребителей и при постоянном количестве производителей

Исходя из полученных данных на Рисунке 2, при данной конфигурации очередь Майкла-Скотта лучше очередей с "грубой" и "тонкой" блокировками при количестве потребителей до 250 и при постоянном количестве производителей, равном 10. Далее при увеличении количества потребителей зависимость скорости перемножения матриц от количества потребителей принимает пилообразную форму, постоянно показывая то наилучшие результаты среди других алгоритмов, то наихудшие.

Зависимость времени умножения матриц от количества производителей (количество потребителей всегда равно 10) представлена на Рисунке 3.

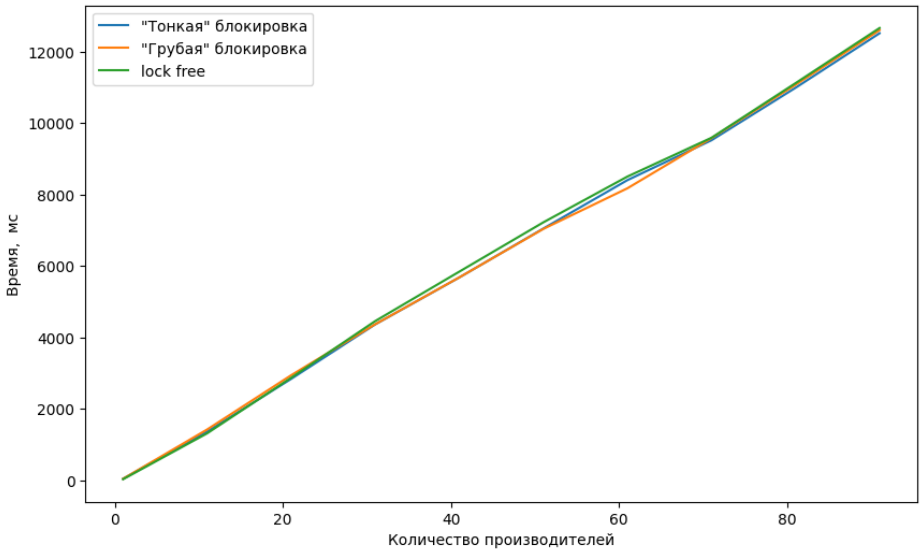

Рисунок 3 – Зависимость времени умножения матриц от количества производителей и при постоянном количестве потребителей

Исходя из полученных данных на Рисунке 3, при данной конфигурации очередь Майкла-Скотта сопоставима с очередями из предыдущей лабораторной работы.

Вывод

В ходе выполнения работы изучено применение потоков в UNIX-подобных системах для решения практической задачи: перемножение матриц. Также изучена зависимость времени перемножения матриц от числа потребителей/производителей, от числа потребителей при постоянном количестве производителей, от числа производителей при постоянном количестве потребителей. Проведено сравнение очереди Майкла-Скотта и очередей с блокировками при разном количестве потребителей и производителей. Установлено, что очередь без блокировок работает лучше при небольшом количестве потребителей относительно числа производителей, а в остальных случаях она как максимум не лучше очередей с блокировками, что может быть связано с тем, что для упорядочивания событий использовался механизм полного упорядочивания (memory_order_seq_cst).

Реализована программа, которая генерирует матрицы разными потоками, а другие потоки принимают данные матрицы и перемножают их, распараллеливая процесс умножения каждой пары матриц. Данная программа обеспечивает добав-

ление сгенерированных матриц в lock-free очередь на базе односвязного списка, а также извлечение элементов из неё.