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. 0303 _____ Пичугин М.В.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург

2023

Цель работы.

Изучение и практическая реализация потокобезопасной очереди с грубой и тонкой блокировкой с использованием условных переменных.

Задание.

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.

Сравнить производительность 2.1. и 2.2 в зависимости от количества производителей и потребителей.

Выполнение работы.

На основе первой лабораторной работы был реализован класс matrix, который содержит все основные операции с матрицами. Этот класс был создан для того, чтобы упростить работу с матрицами и сделать ее более эффективной.

Класс matrix принимает размерность матрицы и на ее основе создает матрицу, заполненную случайными значениями от -10 до 10. Кроме того, в классе были перегружены операторы присваивания, умножения и вывода в потоке.

Умножение матрицы с использованием “грубой” блокировки.

Для выполнения данного пункта был реализован класс RoughBlockQueue, который представляет собой очередь с грубой блокировкой. Этот класс основан на использовании стандартного класса std::queue. Синхронизация в очереди осуществляется с помощью мьютекса и условной переменной.

Умножение матрицы с использованием “тонкой” блокировки.

В данном пункте был создан класс ThinBlockQueue, который представляет собой очередь с тонкой блокировкой. Эта очередь отличается от предыдущей тем, что для блокировки потоков используется два мьютекса. Один мьютекс используется для блокировки головы очереди, а другой - для блокировки хвоста очереди.

При вставке элемента в очередь блокируется хвост очереди. При удалении элемента из очереди блокируются голова и хвост очереди.

В программе поток producer генерирует матрицы и помещает их в очередь. Поток consumer извлекает из очереди две матрицы и умножает их.

Исследуем скорость работы в зависимости от количества потребителей и производителей для каждой подзадачи. Результат измерения скорости работы представлены в табл. 1 и в табл. 2.

Таблица 1 — Результат работы программы при использовании очереди с грубой блокировкой

Количество производителей	Количество потребителей	Произведённые пары матриц	Умноженные пары матриц
1	1	10339	1047
1	5	4231	2921
5	1	5660	437
5	5	5910	1558

Таблица 2 — Результат работы программы при использовании очереди с тонкой блокировкой

Количество производителей	Количество потребителей	Произведённые пары матриц	Умноженные пары матриц
1	1	11262	1103
1	5	4333	2638
5	1	5721	494
5	5	5933	1602

Исходя из результатов таблицы можно сделать вывод, что очередь с «тонкой» блокировкой работает быстрее, чем очередь с грубой блокировкой. Это связано с тем, что очередь с грубой блокировкой останавливает всю структуру данных.

Вывод.

В процессе выполнения лабораторной работы были реализованы потокобезопасные очереди с «грубой» и «тонкой» блокировкой для решения проблемы производитель-потребитель. Таким образом была повышена эффективность параллелизма.