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 _____ Середенков А.А.

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 1 лабораторной работы был реализован класс matrix. В него были перенесены все основные операции с матрицей, находившиеся до этого в файле matrix_operations.h. Класс принимает размерность матрицы и на её основе создаёт матрицу, заполненную случайными значениями от [-10, 10].

Были перегружены операторы присваивания, умножения и вывода в потоке.

Умножение матрицы с использованием “грубой” блокировки.

Для данного пункта был реализован класс *Rough_block_Queue* — очередь с грубой блокировкой. Данный класс основан на использовании *std::queue.* Синхронизация происходит за счёт использования мьютекса и условной переменной.

Умножение матрицы с использованием “тонкой” блокировки.

Для данного пункта был реализован класс *Thin_block_Queue* — очередь с тонкой блокировкой. Данный класс отличается от предыдущего наличием двух мьютексов для блокировки потока (один для «головы», второй для «хвоста» очереди). При вставке элемента блокируется «хвост», а при удалении элемента из очереди - «голова» и «хвост».

В программе «producer» генерирует матрицы и кладёт их в очередь, «consumer» достаёт пару матриц 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	18076	1847
1	5	13549	6726
5	1	52918	1304
5	5	29462	4333

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

Количество производителей	Количество потребителей	Произведённые пары матриц	Умноженные пары матриц
1	1	21008	2050
1	5	13309	6849
5	1	63152	1306
5	5	41960	4521

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.

Выводы.

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.