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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Параллельные алгоритмы»
Тема: Реализация параллельной структуры данных с тонкой блокировкой

Студент гр. 9304 Ламбин А.В.

Преподаватель Сергеева Е.И.

Санкт-Петербург
2022

Цель работы.

Изучить основы реализации параллельных структур данных с тонкой блокировкой в языке программирования C++.

Задание.

Выполнить поэлементное сложение двух матриц $M \times N$. Входные матрицы вводятся из файла или генерируются, результат записывается в файл.

Обеспечить структуру данных из лабораторной работы №2 тонкой блокировкой или сделать её lock-free. Протестировать доступ в случае нескольких потоков производителей и потребителей. Сравнить производительность со структурой с грубой синхронизацией.

В отчёте сформулировать инвариант структуры данных.

Выполнение работы.

Для создания lock-free структуру данных был изменён класс *Buffer* из лабораторной работы №2. Класс реализует lock-free стек, содержит единственное приватное поле *top*, являющееся указателем на вершину стека. Каждый элемент стека представляет объект шаблонного класса *Node*, содержащий указатели на хранимые данные и на следующий узел структуры.

Для добавления узлов в стек реализован метод *push()*. Метод создаёт новый элемент с переданными ему данными, добавляет его в список. Для инварианта добавления используется CAS-операция: новый узел добавляется в вершину в том случае, если другой поток не успел её изменить. Если другой поток успел изменить вершину, то указатель *curNode*, хранящий указатель на вершину стека на момент попытки добавления узла, будет не совпадать с вершиной, что говорит о том, что вершина сместилась.

Для удаления вершины из стека реализован метод *pop()*. Метод ожидает ситуации, когда вершина стека не является *nullptr* и не один поток не изменил вершину на момент попытки забрать вершину, что достигается также с помощью CAS-операции. В конце метод возвращает данные, хранящиеся в удаляемом узле.

Был написан скрипт на языке программирования Python, вычисляющий зависимости между количеством матриц и временем выполнения программы. Результаты вычислений представлены на рисунке 1.

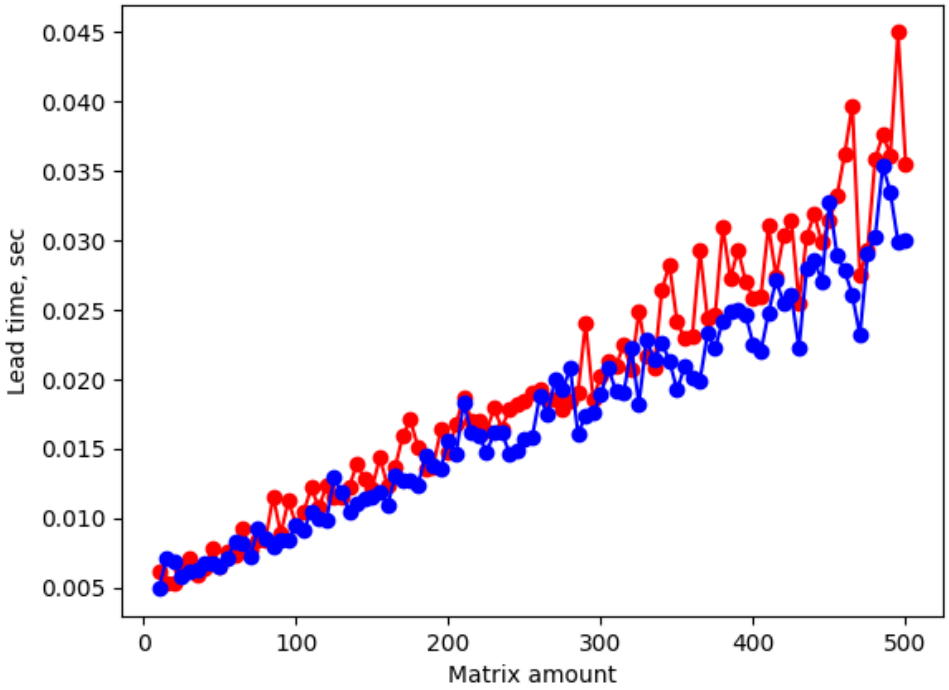

Рисунок 1 – Зависимость времени выполнения работы от количества матриц (красный цвет – тонкая синхронизация, синий цвет – грубая синхронизация)

Из полученных результатов видно, что на больших объёмах данных тонкая синхронизация работает несколько дольше, чем грубая.

Выводы.

В ходе лабораторной работы были изучены основы реализации параллельных структур данных с тонкой блокировкой в языке программирования C++. Была реализована lock-free структура данных, сформулирован её инвариант. Было проведено сравнение структур с тонкой и грубой блокировкой, на основании чего построена зависимость времени выполнения от объёма данных.