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

ОТЧЕТ

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

Студент гр. 9303 _____ Махаличев Н.А.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2022

Цель работы.

Реализация параллельной структуры данных с тонкой блокировкой для сложения матриц.

Задание.

Обеспечить структуру данных из лаб.2 как минимум тонкой блокировкой (*сделать lock-free).

Протестировать доступ в случае нескольких потоков-производителей и потребителей. Сравнить производительность со структурой с грубой синхронизацией (т.е. с лаб.2).

В отчёте сформулировать инвариант структуры данных.

Выполнение работы.

Для реализации lock-free был переписан класс Buffer, в основе которого теперь связный список. За основу буфера был взят стек Трайбера, использующий lock-free алгоритм и CAS, который был несколько оптимизирован. CAS (Compare And Swap, сравнение с обменом) – атомарная операция, сравнивающая значение в памяти с первым аргументом и, в случае успеха, записывающая второй аргумент в память.

Принцип работы produce():

1. Запоминается указатель на текущую голову стека;
2. Создается новый элемент списка, в котором сохраняется указатель на матрицу;
3. Указатель на следующий элемент нового элемента устанавливается на сохранённый в п.1. указатель;
4. С помощью CAS в бесконечном цикле проверяется, был изменен список другим потоком или нет:
    a. Если список изменён, повторяются пункты 1 и 3;

b. Если список не был изменён, устанавливается указатель на голову как указатель на созданный в пункте 2 элемент (т.е. новый элемент списка помещается в его голову).

Принцип работы consume():

1. Запоминается указатель на текущую голову стека;
2. С помощью бесконечного цикла проверяется пустой ли сохранённый указатель и был изменен список другим потоком или нет:
   a. Если сохранённый указатель пустой или список был изменён, выполняется пункт 1;
   b. Иначе указатель на голову буфера считается как указатель на следующий элемент текущей головы и возвращается матрица, хранящаяся в текущем сохраненном элементе (т.е. получение матрицы происходит из головы списка).

Инвариант:

1. Для lock-free гарантируется прогресс хотя бы одним потоком. Если какой-то из потоков находится в бесконечном цикле, это означает, что другой поток выполняет операцию над структурой данных (или в случае потребителя нет данных);
2. Два указателя списка не могут ссылаться на один и тот же элемент;
3. В связном списке не может быть циклов;
4. Потребитель не может забрать что-то из пустого списка.

Сравним производительность в случае нескольких потоков-производителей и потребителей (количество итераций – 40). Сравнение представлено в табл. 1.

Таблица 1 – Производительность в случае нескольких потоков-производителей и потребителей

Размер матрицы	Количество потоков	Время выполнения, мс
500x500	2	1159
1000x1000		4167
2000x2000		16761
500x500	4	1139
1000x1000		4234
2000x2000		18091
500x500	8	1122
1000x1000		4261
2000x2000		17663

Как можно заметить, не всегда увеличение количества потоков дает прирост производительности. На время выполнения большое влияние оказывают сложность выполняемой операции, количество входных данных, время на создание потоков, а также характеристики вычислительной системы. Поэтому следует выбирать оптимальное количество потоков для требуемой задачи.

Установим количество потоков равным 2 и сравним производительность со структурой с грубой синхронизацией (количество итераций – 40). Сравнение производительности представлены в табл. 2.

Таблица 2 – Сравнение производительности со структурой с грубой синхронизацией

Размер матрицы	Время выполнения с lock-free, мс	Время выполнения с грубой синхронизацией, мс
500x500	1159	926
1000x1000	4167	3929
2000x2000	16761	15671
5000x5000	109639	100979

Как можно заметить для операции сложения матриц производительность структуры с грубой синхронизацией оказалась лучше, чем с lock-free.

Выводы.

В процессе выполнения лабораторной работы было реализовано сложение двух матриц с использованием lock-free структуры. Произведено сравнение производительности с различными параметрами и с структурой с грубой синхронизацией, откуда был сделан вывод, что для получения наилучшей производительности необходимо оптимально выбирать параметры и алгоритмы синхронизации для поставленной задачи.