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.

Студентка гр. 0381 Ионина К.С.

Преподаватель Сергеева Е.И.

Санкт-Петербург
2022

Цель работы.

Реализация параллельной структуры данных с тонкой блокировкой для сложения матриц.

Задание.

Реализация параллельной структуры данных с тонкой блокировкой. 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.

Был реализован стек Трайбера, а для реализации свойства неблокируемости, которое гарантирует прогресс в системе, используется операция CAS. CAS — атомарная инструкция, сравнивающая значение в памяти с первым аргументом и, в случае успеха, записывающая второй аргумент в память.

Добавление (produce()) элементов в буфер(стек Трайбера) происходит следующим образом:

Запоминается, куда указывает голова стека (node). Создается новый элемент, который хотим добавить в начало стека newHead. Указатель на следующее значение для него — node. Пробуем переместить голову стека на новый элемент, при помощи CAS. Если удалось — добавление прошло успешно. Если нет, то кто-то другой изменил стек, пока происходило добавление элемента. Придется начинать сначала.

Удаление (consume()) элементов из буфера (стека Трайбера) происходит следующим образом:

Запоминается, куда указывает голова стека (node). Значение, которое хранит в себе node, — то, что необходимо будет вернуть. Пробуем переместить голову стеком CASом. Если удалось — вернем node.matrix. Если нет, то это

означает, что с момента начала операции стек был изменен. Поэтому операция выполняется заново.

Так же для реализации стека Трайбера был создан дополнительный класс Node. Он хранит в себе матрицу и указатель на следующий узел.

Далее было проведено сравнительное исследование данной структуры со структурой из предыдущей лабораторной работы. Исследование проводилось на пятидесяти итерациях на матрицах размера 500*500. Результаты представлены в табл.1.

Таблица 1 — результаты исследования

Число потоков	Время выполнения с lock-free	Время выполнения с грубой синхронизацией
2	3810 ms	1818 ms
5	8990 ms	1901 ms
7	13582 ms	1736 ms
9	17021 ms	1825 ms

Затем было произведено исследование зависимости времени выполнения от размера матрицы на десяти итерациях при 7 потоках. Результаты представлены в табл.2.

Таблица 2 — результаты исследования

Размер матрицы	Время выполнения с lock-free	Время выполнения с грубой синхронизацией
100*100	156 ms	29 ms
500*500	2730 ms	368 ms
1000*1000	9372 ms	1443 ms
3000*3000	96885 ms	12157 ms

Опираясь на полученные результаты, можно сделать вывод, что для операции сложения матриц производительность структуры с грубой синхронизацией лучше, чем с lock-free. Так же можно заметить, что чем больше матрицы, тем менее эффективно работает lock-free алгоритм.

Инвариант структуры данных:

Гарантируется, что в один момент времени общее состояние стека может быть изменено только одним потоком, независимо от других потоков, которые также пытаются внести изменения.

Потребитель не может забрать что-то из пустого списка.

Выводы.

В ходе выполнения лабораторной работы была реализована lock-free структура. Было проведено сравнительное исследование между lock-free структурой и структурой с грубой синхронизацией. По результатам исследования было выявлено, что для наилучшей производительности программы необходимо оптимально выбирать параметры и алгоритмы синхронизации.