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. 9303 _____ Камакин Д.В.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург

2022

Цель работы.

Реализовать структуру данных с как минимум тонкой блокировкой.

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.

На основе лаб. работы 2 был реализован lock-free вариант структуры данных при помощи std::atomic.

Для реализации очередь была перестроена в виде связного списка. Каждый элемент хранит std::shared_ptr для данных и std::atomic<Node*> для указателя на следующий элемент. Сам список хранит два std::atomic<Node *> для начала и конца списка.

Теперь рассмотрим вставку элемента:

Для начала формируем новый узел из входных данных. Чтобы вставить элемент в конец списка поток должен быть уверен в том, что последний элемент, который он видит, актуален. Для этого пытаемся вставить в std::atomic<Node *>->next наш новый узел, если следующий элемент на момент проверки - null (т.е конец списка актуален). Далее какой-либо поток успешно поменяет указатель на следующий элемент, остальные будут ждать до тех пор, пока “счастливчик” не обновит указатель на конец списка.

Теперь рассмотрим удаление элемента:

Удаление элемента достаточно простое: потоку надо быть уверенным в том, что начало списка актуально. Делается это с помощью попытки CAS замены “головы” на следующий за ней элементом.

Сравним производительность со структурой с грубой синхронизацией при 5 итерациях.

Количество читателей/сумматоров(рабочих)/писателей	Lock-free, миллисекунды	Грубая синхронизация, миллисекунды
5/1(2)/2	100	90
4/2(2)/2	120	100
2/1(2)/2	80	75
1/1(1)/2	70	70
5/1(5)/5	200	120

Таблица 1 - Сравнение производительности lock-free структуру с грубой блокировкой на матрице 100x100

Как видно, в реальной жизни lock-free блокировка далеко не во всех случаях может быть полезна. Дело в том, что несмотря на возможные накладные ресурсы во время получения замка, в результате поток может уснуть, высвобождая вычислительные ресурсы, помогая всей системе. Lock-free же такого сделать не может и в результате практически всегда потоки работают, нагружая систему. Поэтому зачастую лучше выбрать работу с блокировками.

Выводы.

На языке программирования C++ была реализована lock-free структура данных.