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 _____ Тиняков С.А.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2022

Цель работы.

Изучить реализацию тонкой блокировки и lock-free.

Задание.

Обеспечить структуру данных из лабораторной работы №2 как минимум тонкой блокировкой (или сделать lock-free).

Протестировать доступ в случае нескольких потоков-производителей и потребителей. Сравнить производительность со структурой с грубой синхронизацией (т.е. с лабораторной работы №2).

Выполнение работы.

Было реализовано два вида неблокирующей (lock-free) очереди: обычная и циклическая. В обычной очереди для каждого нового значения выделяется новый узел, который неблокирующим способом добавляется в конец очереди. После получения значения из очереди память, связанная с узлом, очищается. Была реализована структура *NodePointer*, которая считает количество обращений к данному указателю. После этого последний из считавших очищает память.

В циклической очереди выделение и очищение памяти происходит только в моменте создания и уничтожения очереди. Каждый из узлов имеет пометку на то, что он готов к чтению и готов к хранению нового значения. Поток, который хочет добавить или удалить элемент, начинает идти с головы или хвоста пока не найдет «свободный» узел: помеченный к чтению или к записи.

Программа *thread_sum* была модифицирована таким образом, чтобы минимизировать процесс вычисления. Были замерены результаты для блокирующей очереди и для неблокирующей обычной очереди. Результаты представлены в табл. 1.

Таблица 1 – Результаты для модифицированной программы *thread_sum*

Тип очереди	**Время, нс**
Блокирующая	20627728
Обычная неблокирующая	25103971

Была реализована новая программа *test_time*, которая создаёт задаваемое количество писателей и читателей, которое одновременно записывают и читают из очереди данные. Было замерено выполнении времени для блокирующей очереди, неблокирующей обычной и циклической. Результаты представлены в табл. 2.

Таблица 2 – Результаты для программы *test_time*

Тип очереди	**Время, нс**
Блокирующая	1973066
Обычная неблокирующая	3825620
Циклическая неблокирующая	3023307

Как видно из результатов, блокирующая очередь выигрывает в скорости, потому что в неблокирующей очереди потоки находятся в активном ожидании. Также видно, что неблокирующая циклическая очередь быстрее, обычной неблокирующей, потому что вместо активного ожидания идёт поиск нужного узла по заранее выделенному списку узлов.

Выводы.

Были изучены реализации тонкой блокировки и lock-free для очереди. Была реализована неблокирующая очередь в двух вариациях: обычная и циклическая. Было проведено два сравнения реализованных очередей, из которых было выявлено, что самой быстрой является блокирующая очередь, а самой медленной – неблокирующая обычная.