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 структур данных без блокировок

Студент гр. 0304 _____ Шквиря Е.В.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург

2023

Цели работы.

Изучить принцип построения потокобезопасных структур данных без блокировками.

Постановка задачи.

Выполняется на основе работы 2.

Реализовать очередь, удовлетворяющую lock-free гарантии прогресса.

Протестировать доступ к реализованной структуре данных в случае нескольких потоков производителей и потребителей.

В отчёте: Сравнить производительность с реализациями структур данных из работы. В отчёте сформулировать инвариант структуры данных.

Выполнение работы.

Была реализована очередь без блокировок на основе очереди Майкла-Скотта. Класс самой очереди называется `LockFreeQueue`.

Внутри себя данная очередь устроена как односвязный список, который поддерживает две операции - положить элемент в конец (`push`) и взять элемент из начала (`pop`). Каждый элемент очереди хранит в себе данные узла и атомарный указатель на следующий элемент списка. В обоих методах очереди реализовано использование бесконечного цикла, в котором происходит попытка атомарной замены указателя. При работе данной очереди важно соблюдать инвариант - по окончанию выполнения любого из методов не должно быть состояния, когда голова очереди находится правее хвоста очереди.

У lock-free очереди, используемой несколькими потоками, существует несколько проблем. В случае с методом `push` - нельзя выполнить добавление элемента в очередь и перемещение хвоста атомарно. В случае с методом `pop` - мы не можем гарантировать соблюдение инварианта очереди, так как в процессе выполнения `push` могут быть случаи, что голова будет указывать на элемент правее, чем тот, на который указывает хвост. Для

решения обеих этих проблем был добавлен механизм “помощи” другим потокам, который внутри методов будет обнаруживать промежуточные состояния изменения очереди.

Анализ

Перед переходом к вычислениям стоит упомянуть, что работа выполнялась на машине с количеством ядер процессора равным 6 и максимальным числом потоков равным 12. Результат одного замера получается как среднее между 10-ю выполнениями полного цикла эксперимента (от создания производителей/потребителей до их окончания работы), генерация матриц в расчёт не бралась.

За основу бралось перемножение двух квадратных матриц размером 300x300, рассматривались перемножения при различных значениях количества производителей и потребителей. В таблицах 1 и 2 приведены результаты, получившиеся для очереди с “грубой” и “тонкой” блокировкой в прошлой лабораторной работе. Результат измерения работы очереди без блокировки приведён в таблице 3. За результат бралось время работы в микросекундах.

Таблица 1 - Очередь с “грубой” блокировкой

C \ P	5	10	15	20
5	226678	195817	180526	178807
10	249802	203297	179979	167131
15	250428	211113	197065	165655
20	259571	224624	212432	186514

Таблица 2 - Очередь с “тонкой” блокировкой

C \ P	5	10	15	20
5	227580	206442	205720	209875

10	210515	240066	257401	264066
15	215869	259096	282113	290876
20	220439	278875	308858	329071

Таблица 3 - Очередь без блокировки

C \ P	5	10	15	20
5	126209	131359	135374	129895
10	183743	161207	175394	146211
15	173223	156007	166307	155073
20	179883	152349	161067	152356

Как можно заметить, очередь без блокировок во всех ситуациях работала быстрее, чем очереди с блокировками. Также по результатам видно, что очередь без блокировок работает быстрее, когда количество производителей больше, чем количество потребителей.

Выводы.

В ходе выполнения лабораторной работы были изучены потокобезопасные очереди без блокировок при их работе по шаблону производитель/потребитель. Для этого была реализована очередь на основе однонаправленного списка.

Было выявлено, что очередь без блокировок работает быстрее, чем очередь с блокировкой, причем лучшие показатели производительности были в случае, когда количество производителей превышало количество потребителей.