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		Борисовский В.Ю.
Преподаватель		Сергеева Е.И.

Санкт-Петербург

2022

Цель работы.

Реализовать корректную работу потоков используя шаблон “производитель-потребитель”.

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.

Был написан класс threadsafe_queue, в котором была реализована fine-grained структура данных очереди. В структуру добавлен один фиктивный узел, что позволяет в методе push использовать всего один мьютекс для хвоста очереди, а в методе wait_for_data (вспомогательный метод для извлечения данных из очереди с ожиданием) хоть и блокируются два мьютекса на хвост и на голову очереди, но блокирование мьютекса на хвост используется только для чтения хвоста и сравнения его с текущей головой очереди, что позволяет блокировать мьютекс хвоста на очень короткое время. Таким образом в отличие от грубой блокировки данный вариант позволяет почти независимо читать и записывать данные в такую структуру, в то время как грубая блокировка полностью блокировала буфер и на чтение и на запись.

Инвариант структуры данных

- `tail->next==nullptr`.
- `tail->data==nullptr`.
- `head==tail` означает, что список пуст.
- Для списка с одним элементом `head->next==tail`.
- Для каждого узла `x` списка, для которого `x!=tail`, `x->data` указывает на экземпляр `T`, а `x->next` – на следующий узел списка. Если `x->next==tail`, то `x` – последний узел списка.
- Если проследовать по указателям `next`, начиная с головы списка, то рано или поздно мы достигнем его хвоста.

Исследование зависимости между количеством потоком, размерами входных данных и параметрами вычислительной системы.

Исследование для одного потребителя и производителя.

Таблица 1 - Сравнение размера входных данных и времени вычисления для одного потока при грубой синхронизации структуры:

Время вычисления(милисек.)	Размер входных данных (число матриц 2x2 для суммирования)
7	100
32	1000
285	10000

Таблица 2 - Сравнение размера входных данных и времени вычисления для одного потока при тонкой синхронизации структуры:

Время вычисления(милисек.)	Размер входных данных
19	100
76	1000
497	10000

Исследование для нескольких потребителей и производителей.

В таблице 3 представлено общее время работы программы в зависимости от кол-ва производителей и потребителей при грубой синхронизации структуры:

Один производитель и два потребителя (милисек.)	Два производителя и один потребитель (милисек.)	Два производителя и два потребителя (милисек.)	Размер генерируемых данных (число матриц 2x2 для суммирования)
32	35	25	1000
393	177	188	10000

В таблице 4 представлено общее время работы программы в зависимости от кол-ва производителей и потребителей при тонкой синхронизации структуры:

Один производитель и два потребителя (милисек.)	Два производителя и один потребитель (милисек.)	Два производителя и два потребителя (милисек.)	Размер генерируемых данных (число матриц 2x2 для суммирования)
58	35	37	1000
486	282	334	10000

Выводы.

В ходе выполнения лабораторной работы была реализована

программа на языке программировании C++ для попарного сложения матриц, использующая в качестве структуры данных *fine-grained* очередь.