МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Паралельнные алгоритмы»
Тема: Реализация потокобезопасных структур данных без блокировок

Студент гр. 0304 _____ Максименко Е.М.

Преподаватель _____ Сергеева Е.И

Санкт-Петербург
2023

Цель работы.

Изучение способов реализации потокобезопасных структур данных без блокировок. Изучение способов обеспечения безопасного освобождения памяти в потокобезопасных структурах данных без блокировок.

Задание.

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.

Выполнение работы.

1. Реализован класс ThreadSafeQueue для потокобезопасной работы с очередью. Класс ThreadSafeQueue предоставляет минимальный интерфейс очереди, который содержит такие методы, как push и pop.

Изначально очередь содержит только один фиктивный узел. Головной узел очереди всегда указывает на фиктивный узел. Инвариант очереди: если указатель на следующий за головным узел nullptr, то очередь пуста.

Для добавления узла в очередь выполняется операция сравнения с обменом для изменения ссылки на следующий элемент хвостового узла, а также изменения самого хвостового узла.

Для извлечения элемента из очереди проверяется пуста ли очередь: если не пуста, то выполняется сравнение с обменом головного узла. Вместо текущего головного узла записывается следующий за ним.

Для безопасного удаления узлов используется механизм «указателей опасности».

2. Измерение времени работы программы для очереди с блокировками и без блокировок в зависимости от количества производителей и потребителей.

Для измерения зависимости времени выполнения программы с различными типами блокировок использовалась платформа, параметры которой отражены на рис. 1.

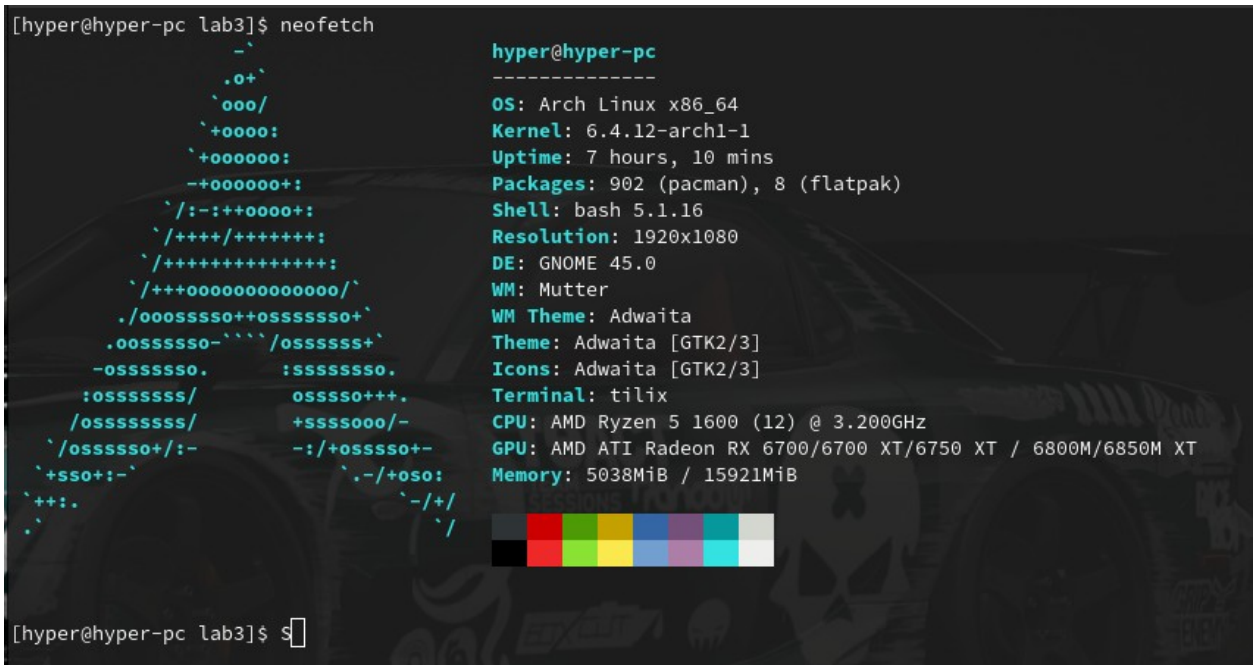

Рисунок 1. Параметры платформы, на которой производятся измерения

Для усреднения результатов программы запускаются 100 раз. Время, представленное в таблицах, указывается суммарное для 100 запусков. Количество задач, выполняемых программой — 6000. Измерения времени выполнения программы без блокировок см. в табл. 1, программы с «тонкими» блокировками — в табл. 2.

Таблица 1. Время выполнения программы без блокировок.

Количество потоков: Производители/ Потребители	**Real Time, сек**	**Sys. Time, сек**
2/2	16.215	0.282
12/12	5.378	0.575
12/1	5.564	0.520
1/12	44.438	0.615
500/500	13.743	6.904

500/10	6.735	4.526
10/500	6.498	4.278

Таблица 2. Время выполнения программы с «тонкими» блокировками.

Количество потоков: Производители/ Потребители	**Real Time, сек**	**Sys. Time, сек**
2/2	18.095	5.811
12/12	4.588	3.510
12/1	7.504	3.135
1/12	33.795	6.673
500/500	7.361	7.887
500/10	6.284	7.565
10/500	5.917	9.421

Как видно по табл. 1 и табл. 2, время выполнения одинакового количества операций с использованием очередей с «тонкими» блокировками и очередей без блокировок во многих рассматриваемых случаях (2/2, 12/12, 500/10, 10/500) примерно равно. Однако для большого количества потоков (500/500) программа с блокировками работала значительно быстрее (~20%). При небольшом количестве потребителей и большом количестве производителей (12/1) реализация без блокировок оказывается быстрее реализации с блокировками.

При большом количестве потоков в реализации с блокировками идет конкуренция за мьютекс, в реализации без блокировок большое количество потоков приводит к большому количеству неудач в операции CAS, которая является «тяжеловесной». Захват мьютекса оказывается в данном случае операцией более «дешевой», чем выполнение множества операций CAS для

получения результата. Также операции безопасного освобождения памяти в реализации без блокировок занимают значительное время, что также добавляет времени исполнения реализации без блокировок. Поэтому при большом количестве потоков реализация без блокировок проигрывает реализации с блокировками.

Выводы.

В ходе работы были изучены способы реализации потокобезопасных структур данных без блокировок. Были изучены способы обеспечения безопасного освобождения памяти в потокобезопасных структурах данных без блокировок.

Была разработана реализация потокобезопасной очереди без блокировок. Для безопасного освобождения памяти используется механизм «указателей опасности».

Было проведено сравнение реализации очереди без блокировок и очереди с «тонкими» блокировками. В большинстве сравниваемых случаев результаты были примерно одинаковыми. Однако для большого количества производителей и малого количества потребителей реализация без блокировок оказалась быстрее реализации с блокировками. С другой стороны, при большом количестве потоков производителей и потребителей очередь с блокировками оказалась быстрее очереди без блокировок.