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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Параллельные алгоритмы»
Тема: Реализация потокобезопасных структур данных с блокировками

Студент гр. 1304 _____ Шаврин А.П.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2024

Цель работы.

Изучить и реализовать потокобезопасную структуру данных с «грубой» и «тонкой» блокировкой, используя шаблон «производитель-потребитель».

Задание.

Реализовать итерационное (потенциально бесконечное) выполнение подготовки, обработки и вывода данных по шаблону “производитель-потребитель” (на основе лаб. 1 (части 1.2.1 и 1.2.2)). Количество производителей и потребителей должно быть изменяемым. Обеспечить параллельное выполнение потоков обработки готовой порции данных, подготовки следующей порции данных и вывода предыдущих полученных результатов. Использовать механизм “условных переменных”.

2.1 Использовать очередь с “грубой” блокировкой.

2.2 Использовать очередь с “тонкой” блокировкой

Выполнить тестирование п. 2.1 и 2.2, убедиться в корректности результатов.

В отчёте: сравнить производительность 2.1. и 2.2 в зависимости от количества производителей и потребителей.

Выполнение работы.

1. Для выполнения действий с матрицами был создан файл *matrix_operations.cpp,* который содержит следующие функции:

std::vector<std::vector<int>> generateMatrix(const int rows, const int columns, const int maxValue = 10) - функция генерации матрицы с заданными параметрами строк и столбцов, а также максимального по модулю значения для генерации. Функция возвращает двумерный вектор целых значений.

std::vector<std::vector<int>> multiplyMatrices(const std::vector<std::vector<int>>& A, const std::vector<std::vector<int>>& B) – функция перемножения переданных матриц A и B. Функция возвращает двумерный вектор – результат перемножения.

void multiplyMatricesByBlocks(const std::vector<std::vector<int>>& matrixA, const std::vector<std::vector<int>>& matrixB, std::vector<std::vector<int>>& result, const int threadsAmount) – функция блочного перемножения матриц

void thMultiplyMatrixBlock(const std::vector<std::vector<int>>& matrixA, const std::vector<std::vector<int>>& matrixB, std::vector<std::vector<int>>& result, std::vector<std::pair<int, int>> blockPositions) – функция перемножения выделенных блоков

void writeMatrixInFile(const std::string& filename, const std::vector<std::vector<int>>& result) – функция записи переданной матрицы в файл.

void printMatrix(const std::vector<std::vector<int>>& result) – функция вывода переданной матрицы в консоль.

2. Для реализации потокобезопасной очереди с “грубой” блокировкой был создан файл roughQueue.hpp, который содержит шаблонный класс, с реализованными методами push и pop для производителей и потребителей:

void push(T value) - метод добавления элемента value в очередь

void pop(T& value) – метод извлечения первого элемента из очереди и сохранение его по ссылке в value.

3. Для реализации потокобезопасной очереди с “тонкой” блокировкой был создан файл thinQueue.hpp, который содержит шаблонный класс, с реализованными методами push и pop для производителей и потребителей:

void push(T value) - метод добавления элемента value в очередь

void pop(T& value) – метод извлечения первого элемента из очереди и сохранение его по ссылке в value.

4. Для реализации поведения производителей были созданы файлы producer.hpp и producer.cpp, которые содержат следующие функции:

myPair produceMatrices() – метод производящий матрицы и возвращающий их в виде std::pair

void thProducer(QueueType& queue, const int power) – шаблонный метод реализующий производство с заданной мощностью.

5. Для реализации поведения потребителей были созданы файлы consumer.hpp и consumer.cpp, которые содержат следующие функции:

void processMatrices(const myPair& matrices) – метод обрабатывающий пару матриц (перемножение и запись результата в файл)

void thConsumer(QueueType& queue, const int power) – шаблонный метод реализующий потребление с заданной мощностью.

6. Для замера времени был создан файл timer.cpp, в котором описан класс Timer реализующий следующие методы:

void start() – функция старта таймера

void stop() – функция остановки таймера

double elapsed() const – функция возвращающая время в миллисекундах прошедшее либо с начала работы таймера, если он не остановлен, либо между началом и концом работы таймера.

7. Для сравнения обеих структур был написан файл main.cpp, который содержит следующие функции:

void testQueue(const int producersCount, const int consumersCount, const int producerPower, const int consumerPower) – шаблонная функция теста потокобезопасной структуры очереди, которая создает переданное количество производителей и потребителей с переданными значениями мощностей.

int main(int argc, char argv[])* – функция которая получает в качестве аргументом количество производителей и потребителей а также их мощности и запускает и замеряет время тестов для обеих видов очередей.

8. Сравнение производительности потокобезобасных очередей.

По таблице 1 был построен график:

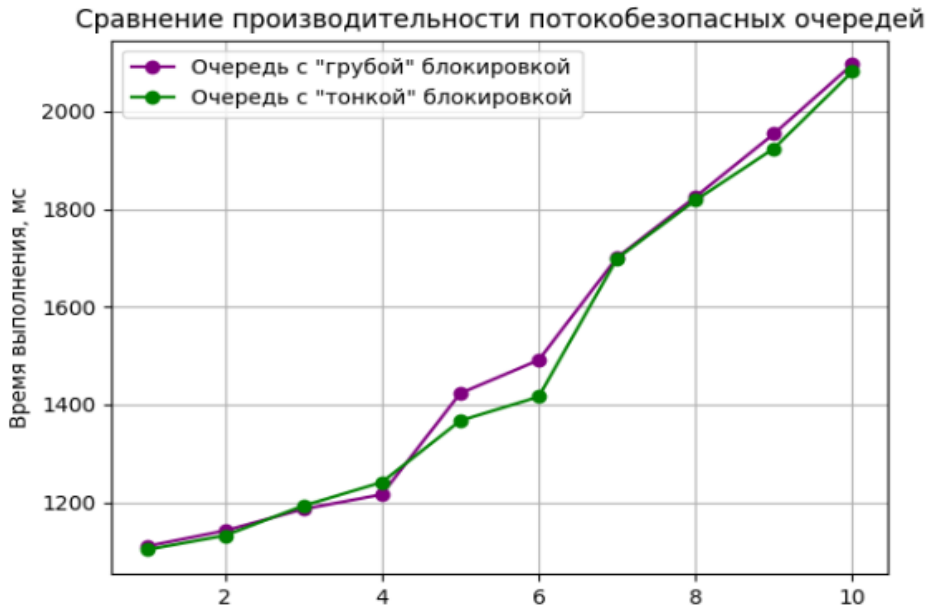

Рисунок 1. Сравнение производительности потокобезопасных очередей

По графику можно сделать вывод о том, что очередь с «грубой» блокировкой менее производительна, чем очередь с «тонкой» блокировкой.

Результаты тестирования см. в приложении А.

Выводы.

В ходе работы была изучена и реализована потокобезопасная структура данных с «грубой» и «тонкой» блокировкой, используя шаблон «производитель-потребитель».

В результате сравнения очередей в зависимости от количества потребителей и производителей установлено, что очередь с «грубой» блокировкой менее производительна, чем очередь с «тонкой» блокировкой, так как при использовании «грубой» блокировки потоки-потребители будут ждать освобождения целой очереди и простаивать, в то время как при использовании «тонкой» блокировки потоки-потребители будут выполнять больше работы и меньше простаивать, так как гранулярность блокировок мельче и больше работы выполняется не под защитой блокировок.

ПРИЛОЖЕНИЕ А

ТЕСТИРОВАНИЕ

Таблица 1 – Сравнение производительности потокобезопасных очередей для размеров входных матриц 240x240, размера очереди 7 и мощностях потребителей и производителей 10

№ п/п	Параметры производителей и потребителей	Время выполнения в мс для “грубой” блокировки	Время выполнения в мс для “тонкой” блокировки
1.	1	1110	1103
2.	2	1142	1132
3.	3	1186	1193
4.	4	1216	1241
5.	5	1423	1367
6.	6	1491	1416
7.	7	1701	1699
8.	8	1825	1818
9.	9	1954	1924
10.	10	2096	2081