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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Параллельные алгоритмы»

Тема: Основы работы с процессами и потоками

Студент гр. 0303 _____ Афанасьев Д.В.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2023

Цель работы.

Изучение основы работы с процессами и потоками в языке программирования C++.

Задание.

Лабораторная состоит из 3х подзадач, которые выполняют одинаковую задачу с использованием процессов или потоков. Выполнить умножение 2х матриц. Входные матрицы вводятся из файла (или генерируются). Результат записывается в файл.

Подзадачи:

1. Выполнить задачу, разбив её на 3 процесса. Выбрать механизм обмена данными между процессами.

Процесс 1: заполняет данными входные матрицы (читает из файла или генерирует их некоторым образом).

Процесс 2: выполняет умножение

Процесс 3: выводит результат

2. Аналогично 1.1, используя потоки (std::threads).

3. Разбить умножение на P потоков (“наивным” способом по по строкам-столбцам).

Выполнение работы.

Общая часть:

Были реализованы два класса Matrix и MultiplicationMatrix. Первый класс хранит матрицу, для него были перегружены операторы считывания и вывода в поток. Второй класс перемножает матрицы. Так при создании класса возможна ошибка, при которой матрицы перемножить невозможно, был создан метод, который возвращает объект класса или выбрасывает исключение в случае ошибки.

Подзадача 1:

Для создания процессов использовалась функция fork(). Для взаимодействия между процессами использовалась разделяемая память при помощи указателя.

Подзадача 2:

Для создания процессов использовался класс std::thread. Для получения результата выполнения задачи потоком использовались классы std::promise и std::future.

Подзадача 3:

Для создания потоков использовались класс std::thread и функция std::async с параметром std::launch::async, для выполнения в отдельном потоке. Умножение было распределено на потоки, путем выдачи каждому потоку задачи на получение строки в матрице, получаемой при умножении. Результаты из потоков собирались при помощи класса std::future, который возвращается функцией std::async.

Исследование проводилось путем запуска перемножение матриц с разным количеством потоков. Количество поток рассматривалось от 1 до 50 с шагом в 5. Были рассмотрены матрицы размером 32 на 32 и 1024 на 1024. Результаты представлены в табл. 1 и табл. 2, соответственно. Исследование проводилось на процессоре с 6 ядрами и 12 виртуальными потоками.

Таблица 1 – Исследование умножение матриц 32 на 32

Количество потоков	Время выполнения, мкс	Количество потоков	Время выполнения, мкс
1	3396	30	1201
5	1094	35	1049
10	972	40	1049
15	1091	45	1184
20	1050	50	1101

25	1149		

Таблица 2 – Исследование умножение матриц 1024 на 1024

Количество потоков	Время выполнения, мкс	Количество потоков	Время выполнения, мкс
1	35231761	30	5484566
5	7293873	35	5436820
10	6246739	40	5365890
15	6794694	45	5362840
20	5726128	50	5360402
25	5634955		

В результате исследования можно сделать вывод, что максимальная производительность достигается при количестве поток близком к количествум виртуальных потоков процессора, но при этом можно заметить, что на больших данных можно добиться лучшей производительности, при использовании функции std::async, которая освобождает поток после завершения выполнения задачи.

Выводы.

В ходе выполнения лабораторной работы были получены практические основы работы с процессами и потоками на языке C++. Было проведено исследовании в ходе, которого было выяснено, что оптимальное количество потоков, используемое в программе, должно равняться количеству виртуальных потоков процессора.