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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Параллельные алгоритмы»

Тема: Основы работы с процессами и потоками

Студент гр. 0303 _____ Пичугин М.В.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург

2023

Цель работы.

Изучение и практическое применение работы с процессами и потоками на языке C++.

Задание.

Выполнить умножение 2х матриц:

1)Входные матрицы вводятся из файла (или генерируются).

2)Результат записывается в файл.

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

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

2) Процесс 2: выполняет умножение

3) Процесс 3: выводит результат

1.2.1 Аналогично 1.1, используя потоки (std::threads)

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

Исследовать зависимость между количеством потоков, размерами входных данных и параметрами целевой вычислительной системы. Сформулировать ограничения.

Выполнение работы.

Функция fill_matrix() из файла matrix.h заполняет матрицу. На вход функция получает матрицу, представляющую из себя вектор векторов, и её размерность. В процессе выполнения программы элементы матрицы заполняются случайными целочисленными значениями в диапазоне от 0 до 9.

Реализация умножения матриц при помощи процессов.

Умножение двух матриц при помощи процессов реализовано в файле process.cpp. Программа начинается с объявления переменных, необходимых для работы: трех матриц a, b и c размером n на n. Затем, функция fill_matrix() заполняет матрицы a и b случайными целочисленными значениями в диапазоне от 0 до 9.

Далее, функция pipe() создаёт канал, который будет использоваться для обмена данными между процессами. Функция возвращает два файловых дескриптора: fd[0] для чтения и fd[1] для записи.

Затем, функция fork() создаёт два дочерних процесса. Первый процесс будет выполнять умножение матриц, а второй - выводить результат.

В конце программы освобождается память, выделенная для буферов.

Реализация умножения матриц при помощи потоков.

Метод умножения двух матриц с использованием потоков реализован в файле thread1.cpp. Чтобы разделить задачу на потоки, создается конструктор класса thread из библиотеки thread, который принимает в качестве аргументов функцию и ее параметры. Наконец, чтобы дождаться завершения работы потока, используется метод join().

Реализация умножения матриц при помощи p-потоков.

Умножение матриц с использованием потоков реализовано в файле thread2.cpp. Чтобы равномерно разделить работу на p потоков, матрица разделяется на p частей по строкам первой матрицы. Все созданные потоки хранятся в векторе threads. После инициализации потоков программа итеративно вызывает метод join(), чтобы дождаться их завершения.

Исследовать зависимость между количеством потоков, размерами входных данных и параметрами целевой вычислительной системы.

Исследуем зависимость времени работы программы от количества потоков. Для этого возьмём постоянный размер матрицы равный 1000x1000. Результат зависимости времени умножения матриц от количества потоков представлен в табл. 1.

Таблица 1 — Зависимость времени работы программы от количества потоков

Количество потоков P	Время работы программы, ms
1	34372
2	21535
4	21234
8	20378

16	19624
32	19535

Из таблицы видно, что с ростом количества потоков, выделенных под умножение скорость работы программы линейно увеличивается, но это будет происходить до тех пор, пока количество потоков не превышает количество строк первой матрице, иначе в увеличении потоков не будет смысла.

Исследуем зависимость времени работы программы от размера входных данных. Результат зависимости времени умножения матриц от количества входных данных представлен в табл. 2.

Таблица 2 — Зависимость работы программы от входных данных

Размер входной матрицы	Время работы программы, ms
10x10	34372
100x100	21535
1000x1000	21234

Исходя из результатов таблицы, можно сделать вывод, что увеличение размерности матрицы замедляет выполнение работы программы.

Вывод.

В ходе лабораторной работы были изучены базовые понятия и методы работы с процессами и потоками на языке C++. Были реализованы три программы, которые выполняли умножение матриц различными способами:

- В первом случае задача была разделена на три процесса: один процесс создавал матрицы, другой выполнял их умножение, а третий выводил результат.
- Во втором случае задача была разделена на три потока, разделенные на задачи как в пункте выше.
- В третьем случае задача была разделена на произвольное количество потоков.

При исследовании зависимости времени работы программ от количества

потоков, размера входных данных и параметров целевой вычислительной системы были получены следующие результаты:

- Увеличение количества потоков не всегда приводит к уменьшению времени работы программы. Оптимальное число потоков зависит от размера входных данных и параметров целевой вычислительной системы.
- Увеличение размера входных данных приводит к увеличению времени работы программы.

На основании полученных результатов можно сделать вывод, что использование процессов и потоков может быть эффективным способом ускорения работы программ, но при этом необходимо учитывать ряд факторов, таких как размер входных данных, параметры целевой вычислительной системы и оптимальное число потоков.