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

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Параллельные алгоритмы»
Тема: Основы работы с процессами и потоками

Студент гр. 9303 _____ Халилов Ш.А.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2022

Цель работы.

Изучить работу процессов и потоков в языке C++.

Задание.

Выполнить поэлементное сложение 2х матриц M*N

Входные матрицы вводятся из файла (или генерируются).

Результат записывается в файл

1.1.

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 сложение

Процесс 3: выводит результат

1.2.1

Аналогично 1.1, используя потоки (threads)

1.2.2

Разбить сложение на P потоков.

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

Выполнение работы.

Для генерации входных данных был написан файл generate.cpp, в происходит генерация матриц по числам из интервала [0; 100] с фиксированные строки и столбцы, определяемые пользователем. После выполнения

получаются два файла - matrix_1.txt и matrix_2.txt, которые обнаруживаются сгенерированные матрицы.

1.1 Реализация с помощью процессов.

Реализация сложения двух матриц с помощью процессов выполнена в файле processes.cpp. Разбиение на три процесса происходит с помощью функции fork(), которая создаёт процессы-потомки. Для того, чтобы определить, какой процесс что выполняет используется идентификатор PID: если значение равно 0, то это потомок, и он выполняет свою часть, иначе, с помощью функции wait() происходит ожидание выполнения кода потомком.

1.2.1 Реализация с помощью потоков.

Реализация сложения двух матриц с помощью потоков выполнена в файл threads.cpp. Поток создается с помощью конструктора thread(), который принимает ссылки на функцию для выполнения, в нашем случае это лямбда-функция и ссылки на аргументы для выполнения функции. Ожидание исполнения потока для продолжения исполнения программы выполняется с помощью метода join().

1.2.2 Разбиение операции сложения на P потоков.

Реализация сложения двух матриц с помощью нескольких потоков выполнена в файле p_threads.cpp. В этом файле реализована функция sumVectors, которая принимает ссылки на 3 вектора, первые два вектора будут складываться по элементам, а результат записывается в 3-й вектор.
Все созданные потоки хранятся в векторе, и после их инициализации для каждого
потока вызывается метод join().

1.3 Исследование зависимости между количеством потоков, размерами входных данных и параметрами целевой вычислительной системы.

Зависимость времени сложения от количества потоков P представлена в табл. 1.

Количество потоков	Размер матрицы N*M	Время (секунды)
2	1000x1000	0.051721
4	1000x1000	0.060574
6	1000x1000	0.061114
8	1000x1000	0.057118
16	1000x1000	0.062645
32	1000x1000	0.060036
64	1000x1000	0.067171
128	1000x1000	0.069285
256	1000x1000	0.08322
512	1000x1000	0.096333

Таблица 1 — Зависимость времени сложения от количества потоков P.

Зависимость времени сложения от размера матриц при сложении с помощью 6 потоков представлена в табл. 2.

Количество потоков	Размер матрицы N*M	Время (секунды)
6	100x100	0.001159
6	1000x100	0.007449
6	100x1000	0.006401
6	1000x1000	0.092396
6	10000x1000	0.43962
6	1000x10000	0.498262
6	10000x10000	3.90624

Таблица 1 — Зависимость времени сложения от размера матриц.

Выводы.

В процессе выполнения лабораторной работы была изучена работа процессов и потоков в языке C++.

Было проведено исследование зависимости времени выполнения программы от различных параметров, и было выявлено:

1) Размер матриц влияет на время выполнения операции сложения: чем больше элементов в матрице, тем дольше осуществляется сложение;

2) Не всегда увеличение числа потоков ведёт к уменьшению времени выполнения програ