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. 9304 _____ Афанасьев А.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2022

Цель работы.

Ознакомиться с работой с процессами и потоками в языке программирования C++.

Задание.

Выполнить сложение 2-х матриц M на N. Входные матрицы вводятся из файла (или генерируются). Результат записывается в файл.

1. Задание: 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.
2. Задание: выполнить задачу, разбив её на 3 потока.
   - Поток 1: заполняет данными входные матрицы (читает из файла или генерирует их некоторым образом).
   - Поток 2: выполняет сложение.
   - Поток 3: выводит результат.
3. Задание: разбить сложение на P потоков. Исследовать зависимость между количеством потоков, размерами входных данных и параметрами целевой вычислительной системы.

Выполнение работы.

Сложение матриц будет происходить в 3 этапа: считывание двух матриц из файла input.txt, сложение матриц поэлементно, и вывод получившейся матрицы в файл output.txt.

Задание 1.

Создание процессов происходит с помощью функции fork(). По возвращаемому значению можно определить в каком процессе выполняется код. Для передачи данных между процессами использовалась shared memory.

Задание 2.

Создание потоков использовался класс std::thread, в конструктор которого передается функция для вызова, а также ее аргументы. Для получения возвращаемого значения из функции использоваля класс std::promise.

Задание 3.

Создание потоков происходит аналогично методу из предыдущего задания. Для выполнения сложения выбирается количество потоков, на которые равномерно разделяются элементы матрицы.

Исследование зависимости между количеством потоком, размерами входных данных и параметрами вычислительной системы происходило путём перебора количества потоков от 1 до 10 с шагом 1, и от 10 до 100 с шагом 10.

В таблице 1 представлены результаты экспериментов для матриц 1000 на 2000 с разным кол-вом потоков.

Таблица 1 - Эксперименты над матрицей 1000 на 2000

Кол-во потоков	Время в мс.	Кол-во потоков	Время в мс.
1	41085	20	19634
2	20978	30	18622
3	17333	40	18046
4	15982	50	18025
5	21903	60	19254
6	19925	70	19522
7	18314	80	19387
8	18012	90	18955
9	18530	100	18485
10	18607		

Видно, что быстрее выполняется сложение, когда потоков 4, что обусловлено кол-вом ядер процессора (4 ядра, 8 HyperThreding потоков).

В таблице 2 представлены результаты экспериментов для матрицы 10 на 20 с разным кол-вом потоков.

Таблица 2 - Эксперименты над матрицей 10 на 20

Кол-во потоков	Время в мс.	Кол-во потоков	Время в мс.
1	138	20	339
2	81	30	484
3	120	40	662
4	105	50	813
5	116	60	862
6	131	70	1141
7	142	80	1131
8	171	90	1092
9	179	100	1563
10	169		

Видно, что наибольший прирост производительности дают 2 потока, что обусловлено маленьким размером матрицы.

Из обеих таблиц видно, что в случае большого количества потоков время обработки начинает замедляться. Отношение полезной работы потоков на работу по их созданию, удалению и переключению контекста становится заметно меньше: не в пользу полезной работы. Этот эффект проявляется намного быстрее на маленьких матрицах, чем на больших, ведь работы для сложения больших матрицах намного больше.

Выводы

В ходе выполнения лабораторной работы была изучена работа с процессами и потоками в языке программирования C++. Было реализовано три модуля: сложение двух матриц с помощью 3 процессов, сложение двух матриц с помощью 3 потоков, сложение двух матриц с помощью P потоков.

Экспериментально установлено:

1. На малом размере матриц более быстрый подсчёт обеспечивается 2-4 потоками из-за малых затрат на создание / смерть потока и смену контекста.

2. На большом размере матриц наиболее быстрый подсчет обеспечивается при использовании количества потоков приближенно равному количеству ядер процессора.