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 _____ Тиняков С.А.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург

2022

Цель работы.

Изучить основы работы с процессами и потоками.

Задание.

Выполнить поэлементное сложение 2х матриц M*N. Входные матрицы вводятся из файла (или генерируются).

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

- Процесс 1 заполняет входные матрицы (читает из файла или генерирует их некоторым образом)
- Процесс 2 выполняет сложение
- Процесс 3 выводит

2. Аналогично пункту 1 только с использованием потоков

3. Разбить сложение на P потоков. Исследовать зависимость между количеством потоков, размерами входных данных и параметрами целевой вычислительной системы

Выполнение работы.

Программа generate_matrix генерирует матрицу с заданными размерами и сохраняет в заданный файл. Существует возможность сохранения данных как в текстовом виде, так и в бинарном. Генерация значений происходит в диапазоне [-1000, 1000] (изменяется одним define'ом в коде).

Программа read_elem выводит значение элемента расположенного в заданной ячейке (нумерация с 1). Данная программа работает только с матрицей записанной в бинарном виде.

Программа process_sum производит сложение матриц при помощи процессов. Главный процесс порождает два процесса чтения, которые считывают данные из файла в память. Для обмена данными между

процессами использует shared memory, которую выделяет и удаляет главный процесс. После завершения работы чтения главный процесс создаёт процесс, который производит суммирование. После суммирования ещё один процесс производит запись данных в файл.

Программа thread_sum производит сложение матриц при помощи потоков. Главный поток порождает два потока чтения, которые считывают данные из файлов. Под хранение матриц главный поток выделяет память в куче. После создаётся P потоков для суммирования матриц. Значение P задаётся пользователем, по умолчанию используется один поток. После суммирования создаётся поток, который записывает полученную матрицу в файл. Программа может работать с матрицами как в текстовом представлении, так и в бинарном.

Были исследовано общее время и время суммирования в зависимости от количества потоков и размера матриц. Полученные данные представлены в виде графиков на рис. 1-6.

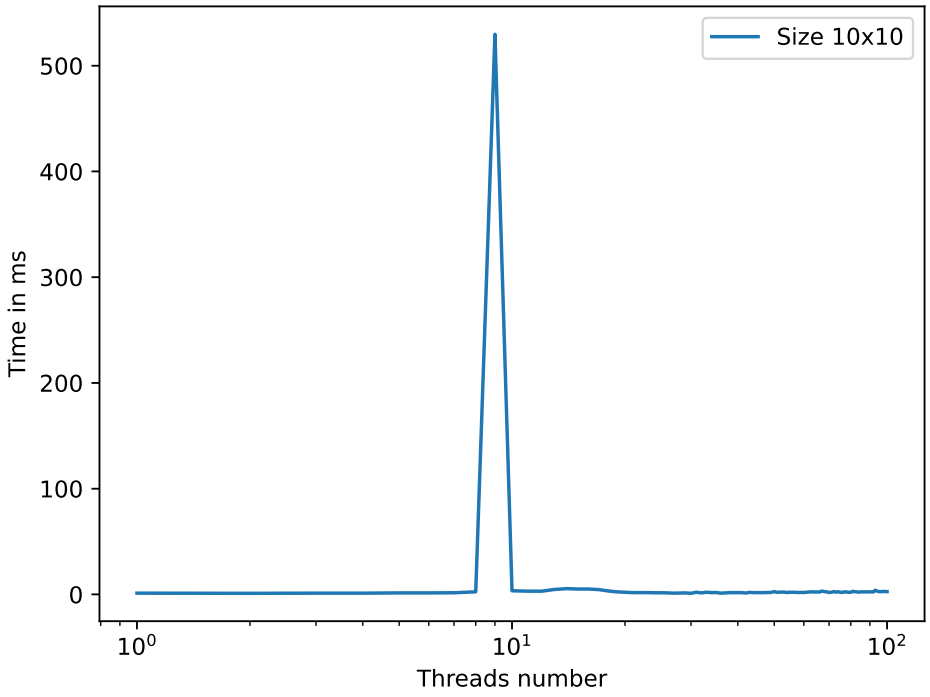

Рисунок 1 – Общее время выполнения для матрицы 10x10

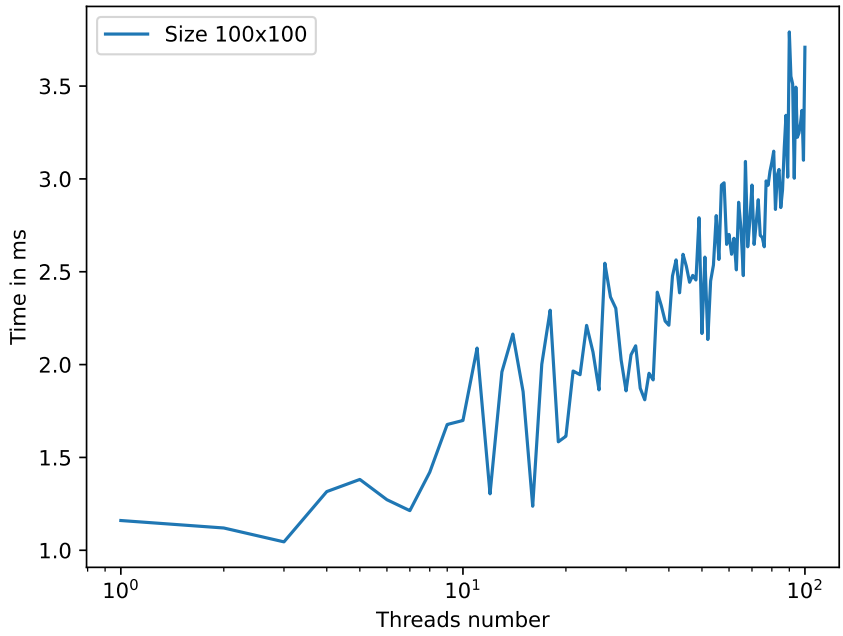

Рисунок 2 – Общее время выполнения для матрицы 100x100

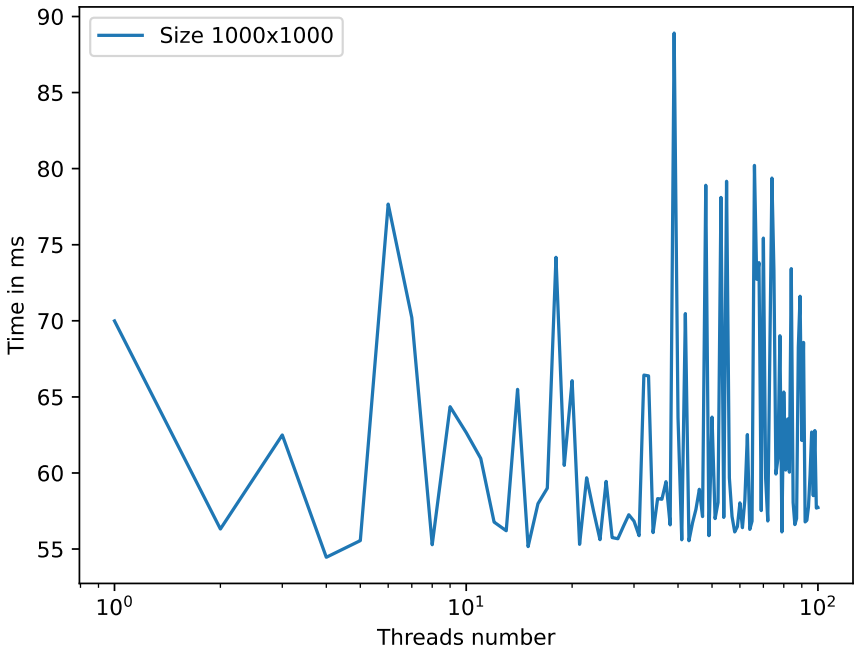

Рисунок 3 – Общее время выполнения для матрицы 1000x1000

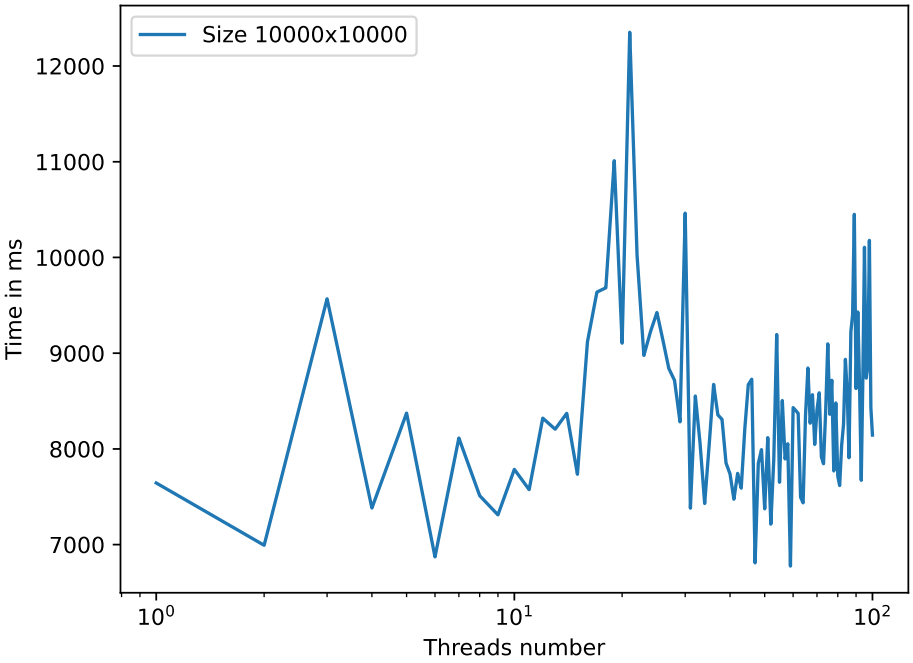

Рисунок 4 – Общее время выполнения для матрицы 10000x10000

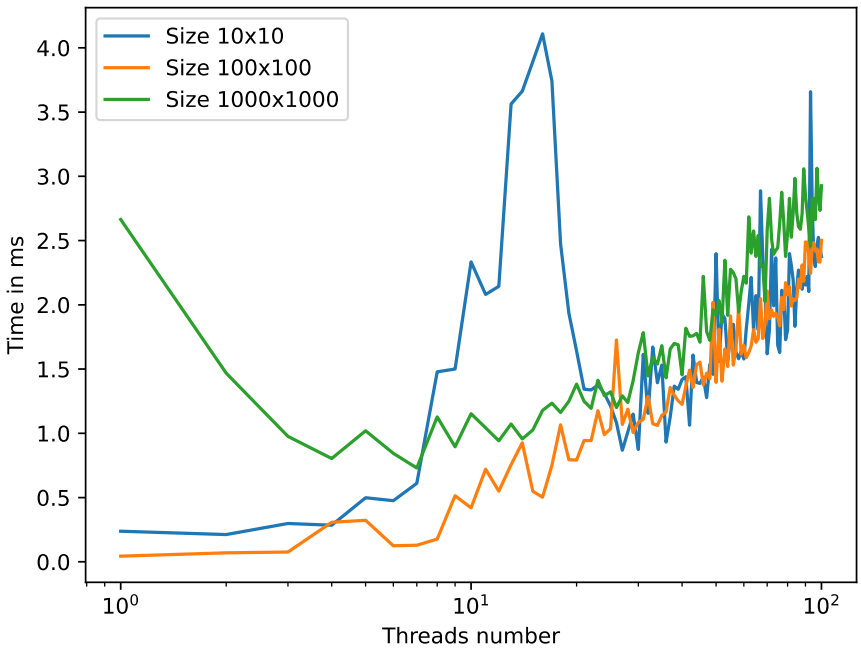

Рисунок 5 – Время суммирования для матриц 10x10, 100x100 и 1000x1000

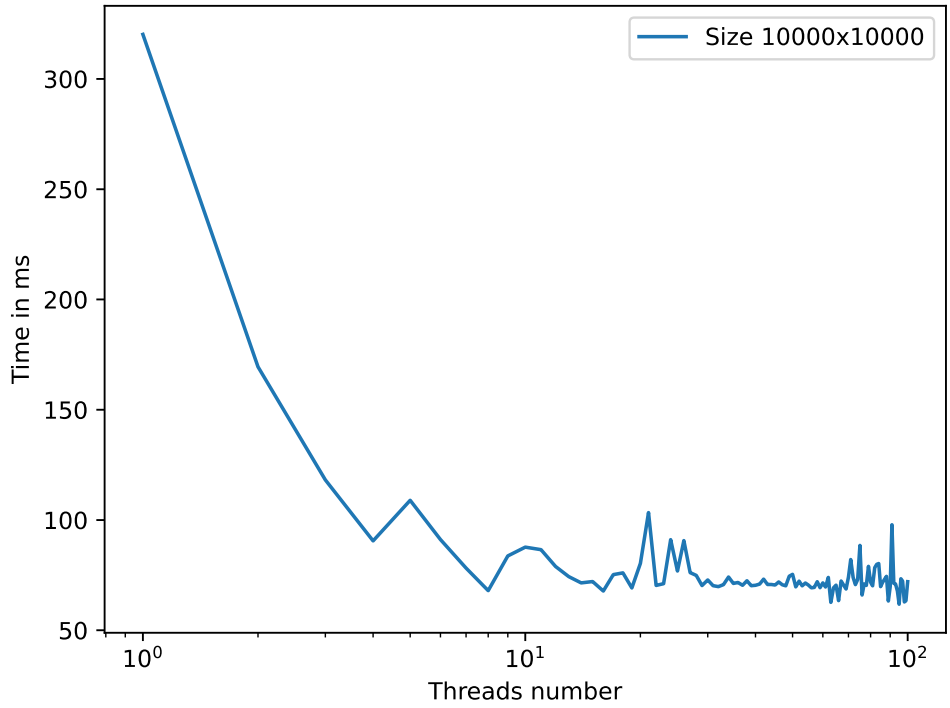

Рисунок 6 – Время суммирования для матрицы 10000x10000

Как видно из графиков, при размерах матрицы меньше 10000x10000 при увеличении количества потоков больших количеству ядер процессора происходит увеличение как общего времени суммирования, так и времени суммирования. Это происходит потому что времени больше уходит на создание потока, чем на его выполнение, к тому же начинает происходить конкуренция за процессорное время. При размере матрицы 10000x10000 данный эффект не так сильно выражен, потому что время создания потоков сильно меньше времени суммирования и потоки не конкурируют, а сменяют друг друга. Как можно заметить, оптимальным количеством потоков является количество ядер на процессоре.

Выводы.

Были изучены основы работы с процессами и потоками.

Было реализовано сложение матриц при помощи процессов и потоков, где каждая из частей задачи (чтение, суммирование, запись) происходит в

разных процессах/потоках. Также было исследованы влияние количества потоков и размера матриц на время работы программы. Было выяснено, что оптимальным количеством потоков является количество доступных ядер процессора. При дальнейшем увеличении количества потоков время выполнения начинает увеличиваться.