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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Параллельные алгоритмы»

Тема: Параллельное умножение матриц

Студент гр. 0304	____________________	Максимов Е.А.
Преподаватель	____________________	Сергеева Е.И.

Санкт-Петербург

2023

Цель работы.

Исследование производительности параллельных алгоритмов.

Постановка задачи.

Реализовать параллельный алгоритм умножения матриц с масштабируемым разбиением по потокам. Исследовать масштабируемость выполненной реализации с реализацией из работы №1.

Реализовать параллельный алгоритм “быстрого” умножения матриц (Штрассена или его модификации). Проверить, что результаты вычислений реализаций совпадают. Сравнить производительность на больших размерностях данных (порядка 10^4 .. 10^6).

Выполнение работы.

Алгоритм умножения матриц с масштабируемым разбиением по потокам был взят из лабораторной работы №1. Для оптимальной работы алгоритма было определено количество потоков равное $\sqrt{N}$, где N — размер матрицы.

Для реализации алгоритма Штрассена были написаны функции, перечисленные ниже.

`void multiplyStrassen(vector<Matrix> matrices*, int N)`, `void multiplyStrassenRoutine(vector<Matrix> matrices*, int recursionDepth)` — основные функции, реализующие алгоритм Штрассена.

`Vector<Matrix> split(Matrix A)`, `Matrix join (vector<Matrix> Cs)` — обратные функции: разбивает матрицу на 4 подматрицы меньшего размера, объединяет 4 матрицы в одну (слева направо, сверху вниз).

`operator+(Matrix A, Matrix B)`, `Matrix operator-(Matrix A, Matrix B)` — вспомогательные функции для краткой записи сложения и вычитания объектов типа `Matrix`.

Максимальная глубина рекурсии равна 3, минимальный размер матрицы, при котором увеличивается глубина рекурсии, равен 4.

Если размер матрицы не равен 2^N, то перед работой алгоритма Штрассена матрица расширяется до минимально возможного N, пустые ячейки заполняются нулями, а после работы алгоритма уменьшается до первоначального размера.

Пусть A, B — входные матрицы. Вычисляются следующие промежуточные M_i матрицы, перечисленные ниже.

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22});$$
$$M_2 = (\mathrm{A}_{12} - \mathrm{A}_{22})(\mathrm{B}_{21} + \mathrm{B}_{22});$$
$$M_3 = (\mathrm{A}_{21} - \mathrm{A}_{11})(\mathrm{B}_{11} + \mathrm{B}_{12});$$
$$M_4 = (\mathrm{A}_{11} + \mathrm{A}_{12})\mathrm{B}_{22};$$
$$M_5 = (\mathrm{A}_{21} + \mathrm{A}_{22})\mathrm{B}_{11};$$
$$M_6 = \mathrm{A}_{22}(\mathrm{B}_{21} - \mathrm{B}_{11});$$
$$M_7 = \mathrm{A}_{11}(\mathrm{B}_{12} - \mathrm{B}_{22}).$$

Умножение матриц производится рекурсивно этим же алгоритмом.

Затем вычисляется результирующая матрица C на основе полученных матриц M_i по формуле ниже.

$$\mathrm{C} = \begin{pmatrix} \mathrm{C}_1 & \mathrm{C}_2 \\ \mathrm{C}_3 & \mathrm{C}_4 \end{pmatrix} = \begin{pmatrix} \mathrm{M}_1 + \mathrm{M}_2 + \mathrm{M}_6 - \mathrm{M}_4 & \mathrm{M}_7 + \mathrm{M}_4 \\ \mathrm{M}_6 + \mathrm{M}_5 & \mathrm{M}_1 + \mathrm{M}_3 + \mathrm{M}_7 - \mathrm{M}_5 \end{pmatrix}.$$

Для измерения времени работы алгоритмов была использована библиотека `chrono`. Результаты тестирования представлены в приложении А.

Выводы.

В ходе лабораторной работы было проведено исследования эффективности алгоритмов умножения матриц. Результаты показали, что алгоритм Штрассена эффективнее, чем умножение с масштабируемым разбиением по потокам, при больших размерностях матрицы ($N > 10^3$).

Практическим результатом лабораторной работы является программный код, реализующий алгоритм Штрассена.

ПРИЛОЖЕНИЕ А
ТЕСТИРОВАНИЕ

Таблица А1 — Сравнение эффективности программ

№	Размерность матрицы	Время работы с масштабируемым разбиением по потокам, мс	Время работы с алгоритмом Штрассена, мс
1	4	<1	2
2	16	1	54
3	64	3	60
4	256	62	160
5	512	263	417
6	1024	1782	2302
7	2048	21723	18426