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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Параллельные алгоритмы»

Тема: Параллельное умножение матриц

Студент гр. 0303 _____ Пичугин М.В.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург

2023

Цель работы.

Изучение и практическая реализация алгоритма Штрассена для перемножения матриц.

Задание.

1. Реализовать параллельный алгоритм умножения матриц с масштабируемым разбиением по потокам. Исследовать масштабируемость выполненной реализации с реализацией из работы 1.

2. Реализовать параллельный алгоритм “быстрого” умножения матриц (Штрассена или его модификации). Проверить, что результаты вычислений реализаций 4.1 и 4.2 совпадают. Сравнить производительность с реализацией 4.1 на больших размерностях данных (порядка 10^4 – 10^6)

Выполнение работы.

Для выполнения данной лабораторной работы был расширен класс Matrix из предыдущих лабораторных работ. В нём были определены операторы суммы и вычитания для удобства дальнейших вычислений, а также сравнения для проверки итоговых вычислений. Также были реализованы две функции:

- parallel(): умножает переданные в качестве аргументов матрицы с масштабируемым разбиением по потокам.

- strassen_alg(): умножает переданные на вход матрицы по алгоритму Штрассена.

Параллельный алгоритм умножения реализован следующим образом:

Каждый поток вычисляет элементы результирующей матрицы, начиная с i + k * n, где n — общее количество потоков, а k — целое число от 1 до m, где m — размерность результирующей матрицы.

Алгоритм Штрассена работает только с квадратными матрицами, размерность которых является степенью двойки. Чтобы выполнить умножение матриц произвольной размерности, необходимо предварительно расширить матрицы до нужного размера. Для этого были реализованы два вспомогательных метода: prepare_mat() и expand_mat().

Метод prepare_mat() расширяет матрицу, добавляя к ней нулевые строки и столбцы. Метод expand_mat() расширяет матрицу, добавляя к ней нулевые строки и столбцы, а также зеркально копируя элементы матрицы в новые строки и столбцы.

Алгоритм Штрассена вычисляет следующие вспомогательные матрицы:

$$
\begin{aligned}
D &= (A_{11} + A_{22})(B_{11} + B_{22});\\
D_1 &= (A_{12} - A_{22})(B_{21} + B_{22});\\
D_2 &= (A_{21} - A_{11})(B_{11} + B_{12});\\
H_1 &= (A_{11} + A_{12})B_{22};\\
H_2 &= (A_{21} + A_{22})B_{11};\\
V_1 &= A_{22}(B_{21} - B_{11});\\
V_2 &= A_{11}(B_{12} - B_{22});
\end{aligned}
$$

На основе этих вспомогательных матриц, вычисляются элементы результирующей матрицы:

$$
\begin{pmatrix}
D + D_1 + V_1 - H_1 & V_2 + H_1 \\
V_1 + H_2 & D + D_2 + V_2 - H_2
\end{pmatrix}
$$

Чтобы сравнить эффективность алгоритмов умножения матриц, рассмотрим результаты их работы на тестовых данных. Для умножения по строкам и масштабируемого умножения использовалось одинаковое количество потоков: 7. Результаты времени работы программы представлены в таблице 1.

Размерность матрицы	Параллельное умножение по строкам	Масштабируемое параллельное умножение	Алгоритм Штрассена
64x64	5	2	5
128x128	29	17	25
256x256	135	137	90
512x512	994	1127	531
1024x1024	13083	15207	3848
2048x2048	140324	230419	44281

Согласно результатам таблицы, масштабируемое параллельное умножение матриц работает быстрее алгоритма Штрассена для небольших матриц. Однако, для больших плотных матриц алгоритм Штрассена работает намного быстрее. В среднем масштабируемое параллельное умножение матриц работает так же, как и умножение матриц из лабораторной работы 1.

Вывод.

В ходе лабораторной работы были изучены и реализованы два алгоритма умножения матриц: масштабируемое параллельное умножение и алгоритм Штрассена.