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. 9304	____________________	Ламбин А.В.
Преподаватель	____________________	Сергеева Е.И.

Санкт-Петербург
2022

Цель работы.

Изучить основы реализации параллельного умножения матриц в языке программирования C++.

Задание.

Реализовать параллельный алгоритм умножения матриц. Исследовать масштабируемость выполненной реализации.

Реализовать параллельный алгоритм «быстрого» умножения матриц. Проверить совпадение результатов вычислений. Сравнить производительность на больших размерностях данных.

Выполнение работы.

Для хранения и удобства использования матриц был реализован класс *Matrix*. В данном классе были перегружены операторы [ ] для обращения к элементам матрицы, $+$ и $-$ для сложения и вычитания матриц, $*$ для параллельного умножения матриц. На каждой итерации выбирались строка i первой матрицы и столбец j второй матрицы, после чего вычислялся элемент c_{ij} результирующей матрицы. Таким образом, сперва вычислялись элементы матрицы, находящиеся на главной диагонали, а затем на каждой следующей итерации элементы на строке на 1 больше предыдущей (или на самой верхней строке, если предыдущий элемент находился на самой нижней). Пример реализации алгоритма для матриц 4×4 представлен на рисунке 1.

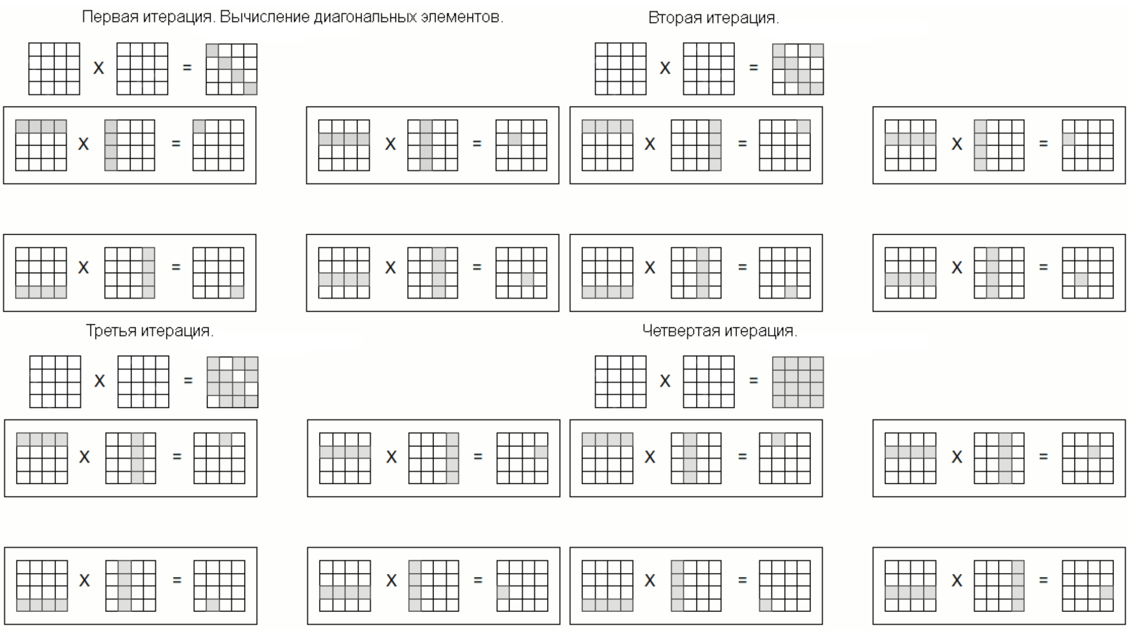

Рисунок 1 – Пример реализации параллельного алгоритма умножения матриц

Был реализован алгоритм Штрассена. Алгоритм работает с матрицами $2^k \times 2^k$, поэтому сперва матрицы дополняются нулями до ближайшего необходимого размера (в конце работы алгоритма матрицы будут возвращены к первоначальному размеру). Алгоритм предполагает работу с частями матриц, поэтому был реализован метод *split()* класса *Matrix*, возвращающий четыре подматрицы. Для обратного объединения был реализован статический метод *join()*. Алгоритм на каждом шаге проверяет размеры матриц: если размеры не превышают 64, то выполняется обычный метод параллельного умножения матриц. В остальных случаях каждая из матриц A и B разделяется на четыре части и вычисляются матрицы

$$P_1 = (A_{11} + A_{22})(B_{11} + B_{22}),$$
$$P_2 = (A_{21} + A_{22})B_{11},$$
$$P_3 = A_{11}(B_{12} - B_{22}),$$
$$P_4 = A_{22}(B_{21} - B_{11}),$$
$$P_5 = (A_{11} + A_{12})B_{22},$$
$$P_6 = (A_{21} - A_{11})(B_{11} + B_{12}),$$
$$P_7 = (A_{12} - A_{22})(B_{21} + B_{22}).$$

После этого вычисляется результирующая матрица C по следующим правилам:

$$C = \begin{pmatrix} P_1 + P_4 + P_7 - P_5 & P_3 + P_5 \\ P_2 + P_4 & P_1 - P_2 + P_3 + P_6 \end{pmatrix}.$$

Для размеров матриц от 64 до 4096, являющихся степенями 2, была вычислена производительность. По полученному графику (рисунок 2) видно, что алгоритм Штрассена эффективен обычного алгоритма начиная с размера 512×512.

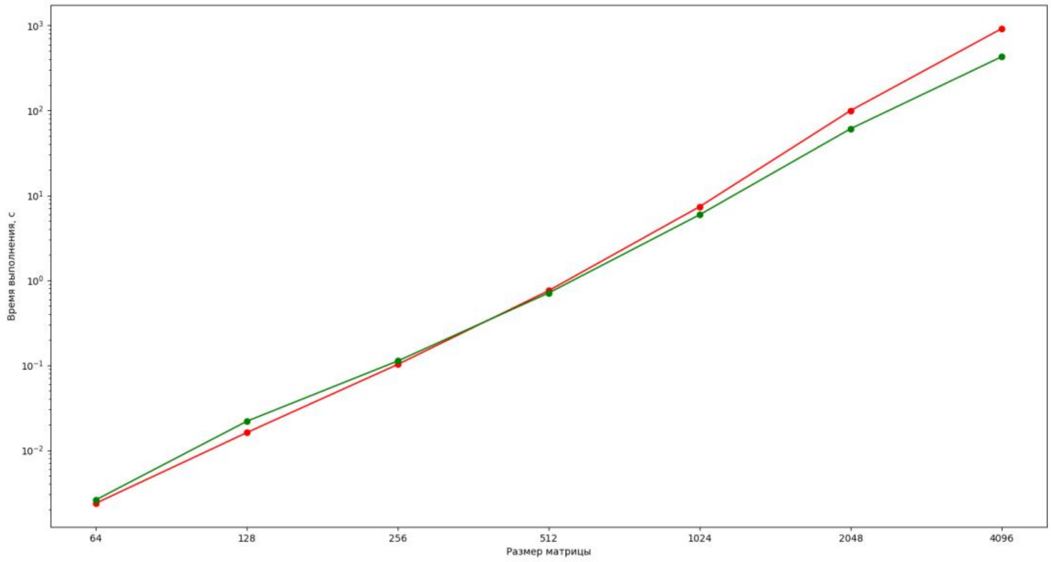

Рисунок 2 – Сравнение обычного алгоритма (красный) и алгоритма Штрассена (зелёный)

Выводы.

В ходе лабораторной работы были изучены основы реализации параллельного умножения матриц в языке программирования C++. Было реализовано два параллельных алгоритма умножения матриц. Произведено сравнение производительности реализованных алгоритмов.