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

ОТЧЁТ

по лабораторной работе № 4
по дисциплине «Параллельные алгоритмы»
Тема: «Параллельное умножение матриц»

Студент гр.1304 _____ Поршнев Р.А.
Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2024

Задание

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

2. Реализовать параллельный алгоритм "быстрого" умножения матриц (Штрассена или его модификации). Тестирование: проверить, что результаты вычислений реализаций 4.1 и 4.2 совпадают (в том числе на больших размерностях). Сравнить производительность с реализацией 4.1 на больших размерностях данных (порядка $10^4 \smile 10^6$)

3. Дополнительно (на отл.) реализация 4.2 должна быть масштабируема (работать на разном количестве потоков)

Выполнение работы.

1. В ходе выполнения данного пункта работы за основу был взят код третьего задания первой лабораторной работы, который был изменён и дополнен необходимыми функциями. Теперь программа принимает через CLI дополнительный аргумент – размер блока матрицы.

Добавлены следующие функции:

• void ExpandMatrix(Matrix &matrix, const int &blockSize) – данная функция принимает на вход матрицу, которую нужно дополнить нулями, чтобы её размеры были кратны размеру блока, который также передаётся в качестве аргумента.

• void MultiplyMatrices(const std::string &filename1, const std::string &filename2, const std::string &filename, int &n_threads, const int &blockSize) – данная функция принимает на вход пути к файлам, в которых хранятся левая и правая матрицы, файл, куда будет записана результирующая матрица, количество потоков и размер блока. В данной функции происходит распределение результирующих блоков по потокам. Каждый поток получает одинаковое количество блоков, которое нужно вычислить в результирующей матрице. Если не получается равномерно

распределить блоки, то оставшиеся блоки поручаются на вычисление первым запускаемым потокам.

• void MultiplyBlocks(const Matrix &matrix1, const Matrix &matrix2, Matrix &matrix, const int &startBlock, const int &endBlock, const int &blocksPerRowInMatrix1, const int &blocksPerRowInMatrix, const int &blockSize) – данная функция получает на вход левую и правую матрицы, результирующую матрицу, начальный и конечный блоки результирующей матрицы, которые необходимо вычислить, количество блоков в строке левой матрицы, количество блоков в строке результирующей матрицы и размер блока. Данная функция предназначена для того, чтобы определить какие блоки левой и правой матрицы необходимо умножить, чтобы получить данный блок результирующей матрицы.

• void MultiplyOperation(const Matrix &matrix1, const Matrix &matrix2, Matrix &matrix, const int &rowMatrix1, const int &columnMatrix1, const int &rowMatrix2, const int &columnMatrix2, const int &rowMatrix, const int &columnMatrix, const int &blockSize) – данная функция принимает на вход левую и правую матрицы, результирующую матрицу, начальную строку и столбец левой матрицы, начальную строку и столбец правой матрицы, начальную строку и столбец результирующей матрицы, блок матрицы. Функция выполняет перемножение блоков левой и правой матриц.

Зависимость времени умножения матриц от размера блока при использовании однопоточности и матриц порядка 1024 представлена в Таблице 1.

Таблица 1 – Зависимость времени умножения матриц от размера блока

Размер блока	Время, мс
2	13133
4	10582
8	9066
16	9938
32	9667
64	9147
128	9613
256	9529
512	10403

Исходя из данных в Таблице 1, можно сделать вывод, что максимальная эффективность блочного умножения при однопоточности достигается при размере блока, равным 8.

Зависимость времени умножения матриц порядка 1024 от количества потоков при использовании блока, размер которого равен 8, представлена на Рисунке 1.

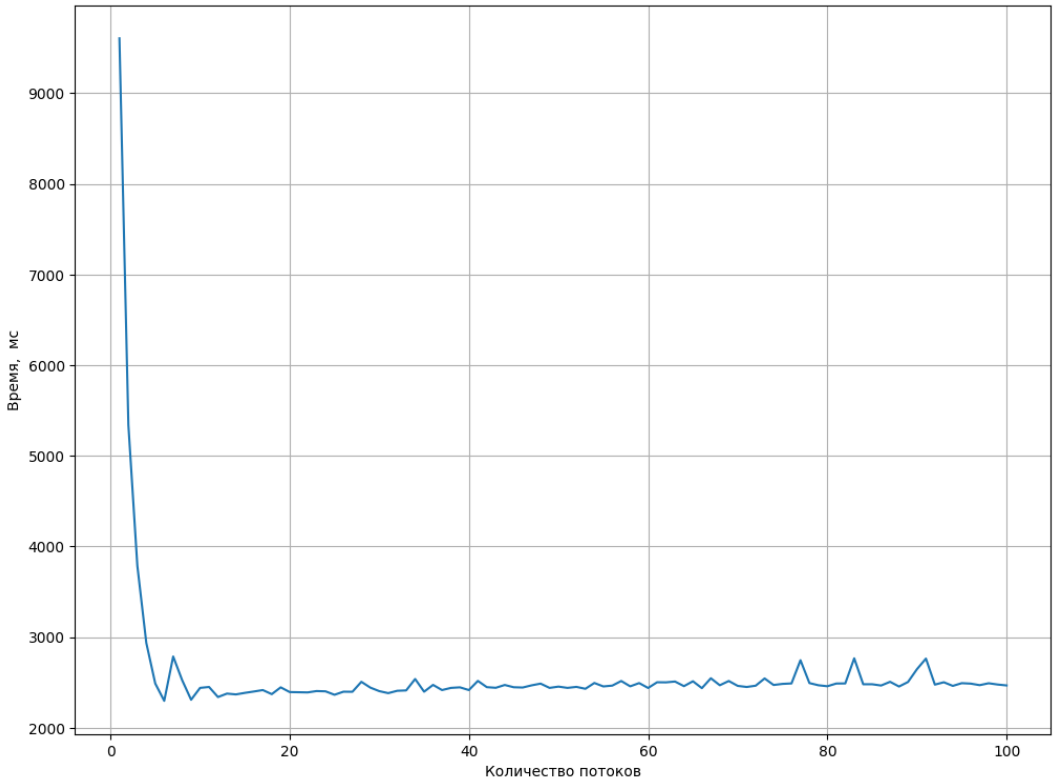

Рисунок 1 – Зависимость времени умножения матриц порядка 1024 от числа потоков

Таким образом, максимальная эффективность блочного умножения матриц порядка 1024 достигается при использовании блока размером 8 и 6-ти потоков.

В первой работе было установлено, что максимальная эффективность обычного умножения достигается при пяти потоках при условии, что размеры матрицы порядка 1024. Зависисимость времени умножения матриц при обычном и блочном умножении при наилучших конфигурациях для матриц порядка 1024 представлена на Рисунке 2.

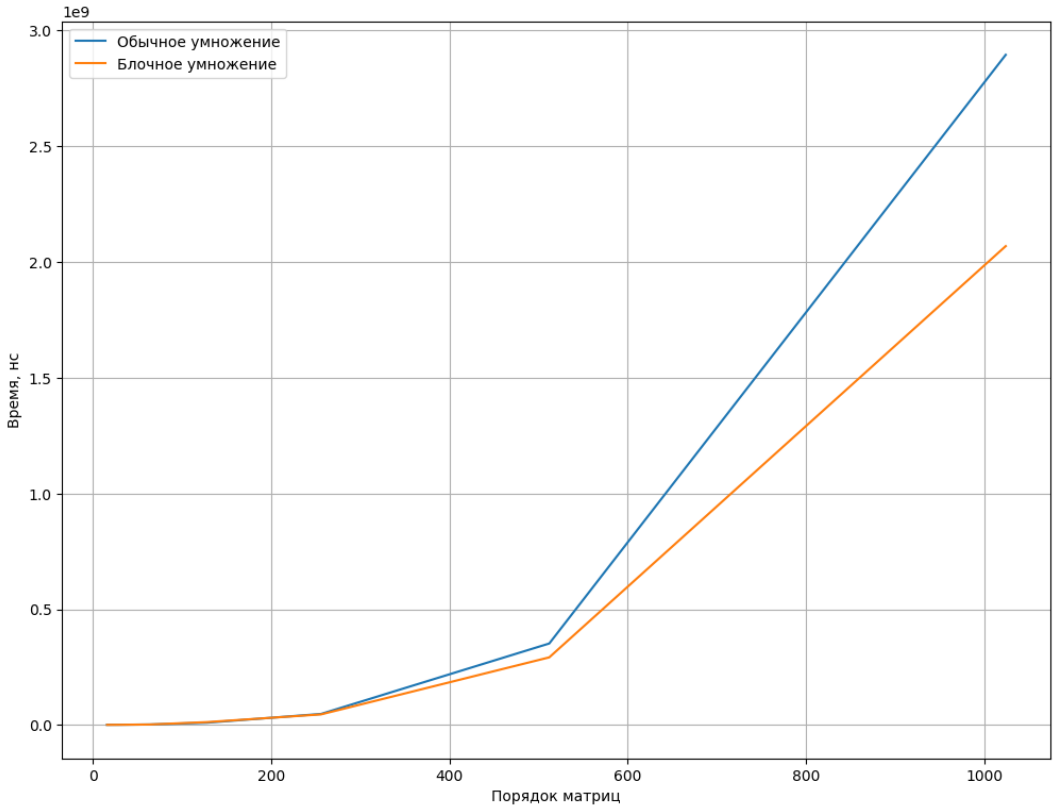

Рисунок 2 – Зависимость времени умножения матриц от их порядка

Исходя из данных на Рисунке 2 можно сделать вывод, что блочное умножение эффективнее обычного умножения матриц за счёт меньшее количества промахов в кэше.

2. В ходе выполнения данного пункта работы за основу также был взят код третьего задания первой лабораторной работы, который был изменён и дополнен необходимыми функциями.

Важно учитывать особенность передаваемого количества потоков, которое указавает на количество потоков, которые будут работать в максимально возможных глубоких частях рекурсии. Из-за особенности реализованной программы передаваемое число потоков должно быть степенью 7. Тогда фактическое число потоков вычисляется по следующей формуле:

$$N = \frac{7 \cdot (p-1)}{6}$$

Добавлены следующие функции:

• void ExpandMatrix(Matrix &matrix, const int &expandSize) – данная функция принимает на вход матрицу, которую необходимо расширить нулями до квадртной матрицы порядка expandSize.

• void MultiplyMatrices(const std::string &filename1,
const std::string &filename2, const std::string &filename, int &n_threads) – данная функция принимает на вход файлы, где хранятся левые и правые матрицы, файл, куда будет записана результируюшая матрицы и количество потоков. В данной функции происходит импорт матриц из файлов, запуск алгоритма Штрассена и запись результата перемножения в файл.

• Matrix add(const Matrix &matrix1, const Matrix &matrix2) – данная функция складывает переданные в качестве аргумента матрицы и возвращает результат.

• Matrix sub(const Matrix &matrix1, const Matrix &matrix2) – данная матрица отнимает от первой переданной матрицы и возвращает результат.

• void TransferData(Matrix &M11, Matrix &M12, Matrix &M21, Matrix &M22, const Matrix &matrix) – данная функция принимает на вход четыре матрицы, которые нужно заполнить данными из матрицы, которая состояла из данных 4-ёх матриц.

• void CollectBlocks(const Matrix &C11, const Matrix &C12, const Matrix &C21, const Matrix &C22, Matrix &matrix) – данная функция предназначена для объединения четырёх матриц в одну.

• Matrix ShtrassenAlgorithm(const Matrix &matrix1, const Matrix &matrix2, int n_threads) – данная функция реализует основую процедуру алгоритма Штрассена, которая заключается в рекурсивном вычислении семи вспомогательных матриц, которые формируют ответ на задачу. Для обеспечения параллельности операций используется асинхронность вычислений семи вспомогательных матриц. Функция принимает на вход перемножаемые матрицы и текущее количество доступных потоков.

Зависимость времени умножения матриц от количества потоков представлена в Таблице 2.

Таблица 2 – Зависимость времени умножения матриц от размера блока

Количество потоков при использовании алгоритма Штрассена	Время, мс
7	2044
49	1778
343	1724
2401	1785
16807	1786

Исходя из данных в Таблице 2, можно сделать вывод, что максимальная эффективность достигается при 343 потоках, что фактически равняется 399 потокам.

График зависимости времени умножения матриц от размера матриц при наилучших конфигурациях для матриц порядка 1024 представлен на Рисунке 3.

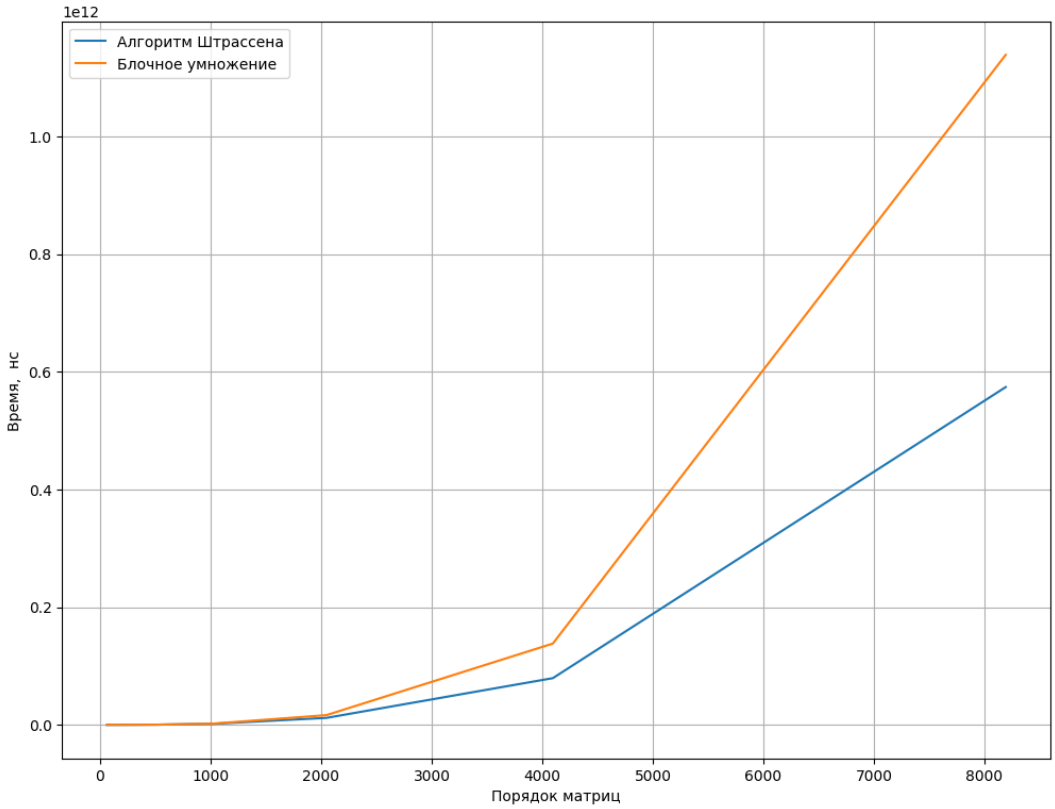

Рисунок 3 – Зависимость времени умножения матриц от их порядка

Исходя из данных на Рисунке 2 можно сделать вывод, что алгоритм Штрассена эффективнее обычного алгоритма умножения матриц и его блочной модификации за счёт меньшего числа умножений.

Вывод

В ходе выполнения работы изучено применение потоков в UNIX-подобных системах для решения практической задачи: перемножение матриц. Было выполнено следующее:

• Построена таблица зависимости времени умножения матриц от размера блока для матрицы порядка 1024, исходя из которой можно сделать вывод, что оптимальный размер блока бля умножения больших матриц равен 8.

• Получена зависимость времени умножения матриц порядка 1024 от числа потоков при размере блока, равном 8.

• Установлено, что для больших матриц оптимальное количество потоков равно 6.

• Исследована эффективность обычного алгоритма умножения матриц и блочного алгоритма при наиболее эффективных конфигурациях, в результате чего было установлено, что блочный алгоритм превосходит обычный алгоритм.

• Установлено, что для матриц порядка 1024 в алгоритме Штрассена максимальная эффективность достигается при количестве потоков, равном 343.

• Экспериментально установлено, что алгоритм Штрассена превосходит блочный и обычный алгоритмы перемножения матриц.

Реализовано две программы на языке C++, первая из которых перемножает матрицы с помощью блочного алгоритма, а вторая – перемножает матрицы с помощью алгоритма Штрассена.