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. 0304 _____ Люлин Д.В.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург

2023

Цель работы.

Изучить алгоритм умножения матриц Штрассена, разработать его параллельную версию. Сравнить производительность разных алгоритмов.

Здание.

4.1. Реализовать параллельный алгоритм умножения матриц с масштабируемым разбиением по потокам.

Исследовать масштабируемость выполненной реализации с реализацией из работы 1.

4.2 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.

1. Масштабируемый алгоритм.

В работе был реализован улучшенный алгоритм умножения матриц, по сравнению с работой №1. Он также является многопоточным, но каждый поток теперь вычисляет определённые ряды в результирующей матрице, таким образом, потоки не проходят по всем элементам результирующей матрицы.

Так как потоки в новом алгоритме проходят не по всем элементам, а только по нужным, то для каждого потока определена своя ограниченная задача. Данное решение является более масштабируемым, с увеличением количества потоков (до *std::thread::hardware_concurrency()*) время выполнения задачи уменьшается.

2. ***Алгоритм Штрассена.***

Алгоритм Штрассена перемножает матрицы, размерность *n* которых является степенью 2. Он оперирует их подматрицами размера *n/2*, позволяя считать произведение рекурсивно. Исходные матрицы разбиваются на части:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

$$B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

Нужно вычислить 7 промежуточных матриц размерности *n/2*, из которых будет вычислена результирующая матрица. Вычисление происходит по формулам, представленным на рис. 1.

$$\begin{aligned}
D &= (A_{11} + A_{22})(B_{11} + B_{22}); \\
D_1 &= (A_{12} - A_{22})(B_{21} + B_{22}); \\
D_2 &= (A_{21} - A_{11})(B_{11} + B_{12}); \\
H_1 &= (A_{11} + A_{12})B_{22}; \\
H_2 &= (A_{21} + A_{22})B_{11}; \\
V_1 &= A_{22}(B_{21} - B_{11}); \\
V_2 &= A_{11}(B_{12} - B_{22}); \\
AB &= \begin{pmatrix} D & 0 \\ 0 & D \end{pmatrix} + \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix} + \begin{pmatrix} -H_1 & H_1 \\ H_2 & -H_2 \end{pmatrix} + \begin{pmatrix} V_1 & V_2 \\ V_1 & V_2 \end{pmatrix} \\
&= \begin{pmatrix} D + D_1 + V_1 - H_1 & V_2 + H_1 \\ V_1 + H_2 & D + D_2 + V_2 - H_2 \end{pmatrix}.
\end{aligned}$$

Рисунок 1. Вычисление произведения матриц по алгоритму Штрассена.

Произведения матриц вычисляются рекурсивно с помощью того же алгоритма. На малых матрицах простое умножение матриц оказывается эффективнее, и на большой глубине рекурсии вместо алгоритма Штрассена используется оно.

Вычисление подматриц можно выполнять параллельно, поэтому при первом вызове алгоритма данные вычисления запускаются в разных потоках. В рекурсии потоки не создаются, чтобы не создать условия потокового голодания.

Для реализации доступа на чтение к подматрицам без копирования данных был написан класс *MatrixSlice*, который ссылается на участок оригинальной матрицы. Для записи данных в участок матрицы в классе *Matrix* был создан метод *insert_part()*.

При сравнении алгоритмов было проверено, что они выдают одинаковые результаты, произведение матриц корректно. Сравнение производительности алгоритмов приведено на графике на рис. 2.

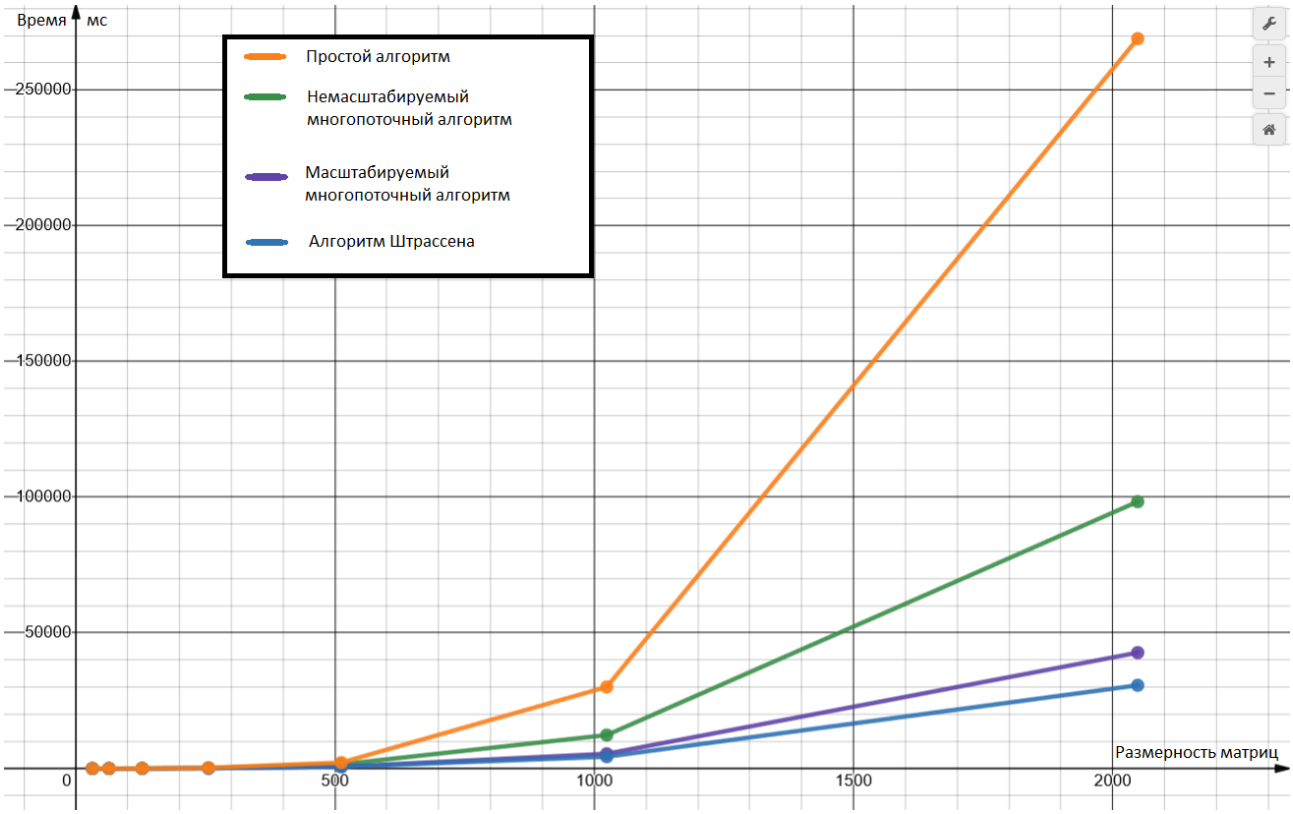

Рисунок 2. Сравнение производительности алгоритмов.

Видно, что параллельный алгоритм Штрассена оказался самым производительным, на втором месте – масштабируемый алгоритм, затем немасштабируемый алгоритм из работы №1. Самым медленным из представленных является простой алгоритм, имеющий кубическую сложность.

Выводы.

В ходе работы был разработан масштабируемый многопоточный алгоритм умножения матриц, в котором каждый поток вычислял только отведённые ему строки результирующей матрицы.

Был реализован параллельный алгоритм Штрассена, вычисляющий произведение матриц с помощью рекурсии. В результате сравнения алгоритмов он оказался самым быстрым. За ним идут масштабируемый и немасштабируемый алгоритмы. Самым медленным является простое умножение матриц.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Исходный код программы доступен в репозитории https://github.com/Astana-Mirza/parallel_algo/tree/master.