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 Сологуб Н.А.

Преподаватель Сергеева Е.И.

Санкт-Петербург
2023

Цель работы.

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.

1) Функции файла main.cpp

Был реализованы функции, взаимодействующие с экземплярами класса Matrix и обеспечивающие параллельное умножения матриц. Функция **void Create(Matrix& first, Matrix& second,Matrix& result,int N)** инициализирует набор из трёх матриц, две из которых перемножаются, а в третью заносится результат умножения. Функция **void StrassenMultiple(Matrix& first, Matrix& second,Matrix& result)** реализует параллельное умножение матриц методом Штрассена. На вход функции поступает матрицы размерностью 2^k, первая и вторая матрица дробится на 4 подматрицы, с помощью которых формируются новые матрицы через 10 операций сложения/вычитания и 7 операций умножения, что обеспечивает более меньшую ассимптотическую сложность, чем обычное умножение матриц. Функция запускает рекурсию до тех пор, пока размер матриц не достигнет 2, затем происходит обычное умножение матриц и результат возвращается. Рекурсия вызывается в четырёх потоках, что обеспечивает распараллеливание задачи. Функция **void MultipleExtended(Matrix& first, Matrix& second,Matrix& result,int startRow,int endRow)** обеспечивает масштабируемое умножения матриц за счёт

выделения каждому потоку определённого количества строк для умножения матриц. Функция **void MultipleSimple(Matrix& first, Matrix& second,Matrix& result,int shiftRow,int threadCount)** реализует умножения матриц из первой лабораторной.

2) Время выполнения параллельного умножения

Были проведены измерения времени параллельных умножений. При измерениях умножений обрабатывались матрицы размерностями 4,8, 16,32,64,128,256,512,1024,2048,4096 в 7 потоков для каждого варианта умножения. Результаты работы параллельных умножений матриц представлены в табл. 1

Таблица 1 - зависимость времени работы умножения матриц от метода умножения

Размерность матриц	Время масштабируемого параллельного умножения, с	Время параллельного умножения, с	Время параллельного умножения Штрассена, с
4	0.0012466	0.0013126	0.0019015
8	0.0012595	0.0013209	0.0024205
16	0.00127	0.0014428	0.0027481
32	0.0012997	0.0013364	0.0040139
64	0.0017698	0.002104	0.0079131
128	0.0070457	0.0089909	0.0216973
256	0.0358545	0.0603649	0.0848752
512	0.213983	0.605712	0.381595
1024	2.36056	4.39453	2.08541
2048	22.2422	29.8992	13.8839
4096	203.339	186.812	148.463

Из таблицы видно, что умножения Штрассена хорошо работают на больших размерностях данных, в то время как на малых размерностях матриц они немного отстают по времени.Также на скорость выполнения влияет глубина рекурсии, которая накладывает память на стек.

Выводы.

В ходе выполнения работы были реализованы параллельные алгоритмы умножения и исследованы их скорости работы при различном наборе данных