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

Студентка гр. 9303 _____ Москаленко Е.М.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2022

Цель работы.

Реализовать параллельные алгоритмы умножения матриц на языке программирования C++ и сравнить их производительность.

Задание.

4.1 Реализовать параллельный алгоритм умножения матриц.
Исследовать масштабируемость выполненной реализации.

4.2 Реализовать параллельный алгоритм “быстрого” умножения матриц (Штрассена или его модификации).

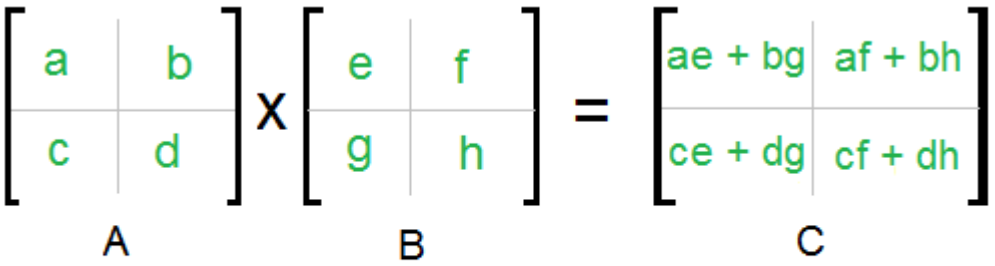
- Проверить, что результаты вычислений реализаций 4.1 и 4.2 совпадают.
- Сравнить производительность с реализацией 4.1 на больших размерностях данных (порядка 10^4 – 10^6)

Выполнение работы.

Было реализовано 3 алгоритма умножения квадратных матриц:

1) Алгоритм 1 делит матрицы на тайлы размером 2 на 2, затем параллельно находит все комбинации (произведения) тайлов, сохраняет их в хэш-мапу, после чего вычисляет элементы результирующей матрицы (сложение двух тайлов происходит с помощью одного или двух потоков).
   Сложность алгоритма O(N^3).
2) Алгоритм 2 представляет собой рекурсивный алгоритм Divide And Conquer.
   1. Матрицы A и B делятся на 4 матрицы размера N/2 x N/2.
   2. Значения считаются рекурсивно, то есть в каждой итерации рассматриваемая матрица делится на 4 части: ae + bg, af + bh, ce + dg and cf + dh.

В первые два вызова каждая часть матрицы обрабатывается в новом потоке. Если же глубина рекурсии больше 2, то с ней продолжает работать тот же поток (иначе количество потоков будет неизбежно возрастать и достигнет предела для ОС).



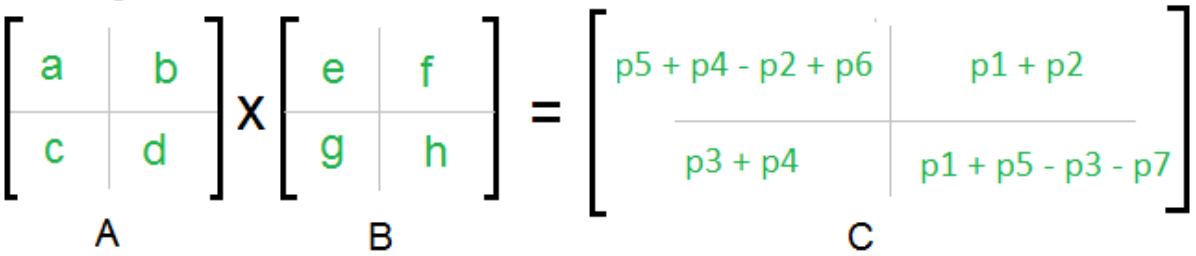
A, B and C are square metrices of size N x N
a, b, c and d are submatrices of A, of size N/2 x N/2
e, f, g and h are submatrices of B, of size N/2 x N/2

Сложность алгоритма все еще O(N^3).

3) Алгоритм 3 представляет собой алгоритм Штрассена. Он является модификацией Divide And Conquer, но значения матриц вычисляются по другим формулам (в результате 7 рекурсивных вызовов вместо 8).

p1 = a(f - h)
p2 = (a + b)h
p3 = (c + d)e
p4 = d(g - e)
p5 = (a + d)(e + h)
p6 = (b - d)(g + h)
p7 = (a - c)(e + f)

The A x B can be calculated using above seven multiplications.
Following are values of four sub-matrices of result C



A, B and C are square metrices of size N x N
a, b, c and d are submatrices of A, of size N/2 x N/2
e, f, g and h are submatrices of B, of size N/2 x N/2
p1, p2, p3, p4, p5, p6 and p7 are submatrices of size N/2 x N/2

При больших размерах матриц 2 и 3 алгоритм работают только до того, пока рассматриваемые суб-матрицы больше размера 64*64. После этого происходит наивное перемножение.
Сложность алгоритма Штрассена по времени O(N^2.8074).

Проверим, что результаты работы всех трех алгоритмов совпадают:

Первый алгоритм (не рекурсивный, работа с тайлами 2*2)

```
Time taken by function: 79 milliseconds
```

```
109 216 251 229 200 214 206 158
114 162 183 177 115 182 164 146
180 251 270 301 185 273 282 151
174 268 302 208 182 252 201 178
202 238 257 232 120 242 222 182
143 209 253 212 159 195 210 134
163 189 228 194 115 209 186 173
136 189 201 219 129 205 208 116
```

Рекурсивный алгоритм Divide And Conquer:

```
Time taken by function: 4 milliseconds
```

```
109 216 251 229 200 214 206 158
114 162 183 177 115 182 164 146
180 251 270 301 185 273 282 151
174 268 302 208 182 252 201 178
202 238 257 232 120 242 222 182
143 209 253 212 159 195 210 134
163 189 228 194 115 209 186 173
136 189 201 219 129 205 208 116
```

Рекурсивный алгоритм Штрассена:

```
Time taken by function: 5 milliseconds
```

```
109 216 251 229 200 214 206 158
114 162 183 177 115 182 164 146
180 251 270 301 185 273 282 151
174 268 302 208 182 252 201 178
202 238 257 232 120 242 222 182
143 209 253 212 159 195 210 134
163 189 228 194 115 209 186 173
136 189 201 219 129 205 208 116
```

Сравнение производительности алгоритмов

Размер матрицы	**Время работы первого алгоритма, миллисекунды**	**Время работы алгоритма divide-and-conquer, миллисекунды**	**Время работы алгоритма Штрассена, миллисекунды**
16*16	106	13	18
64*64	37 241	608	488
256*256	120 000+	308	277
1024*1024	-	19 267	12 820
2048*2048	-	144 401	77 097

Выводы.

На языке программирования C++ были реализованы параллельные алгоритмы умножения квадратных матриц. На большой размерности матриц алгоритм Штрассена показал себя наиболее эффективным по времени.