

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

Максименко Е.М.

Преподаватель

Сергеева Е.И

Санкт-Петербург

2023

Цель работы.

Изучение алгоритмов параллельного умножения матриц. Изучение алгоритма Штрассена.

Задание.

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. Реализован класс матрицы `Matrix`, который поддерживает такие операции с матрицей, как получение размерностей (`GetRows`, `GetCols`), получение (`Get`) и установка (`Set`) элементов. Также поддерживается операция извлечения подматрицы `Slice`, необходимая для работы алгоритма Штрассена. Для работы алгоритма Штрассена реализована операция заполнения элементов матрицы элементами подматрицы `Insert`. Также присутствуют операции генерации элементов (`RandomFill`) и ввода (`Read`)/вывода (`Print`) матрицы. Реализованы операции сложения и вычитания матриц, а также операция сравнения двух матриц.

2. Реализован класс `Algorithms`, содержащий алгоритмы умножения матриц: последовательный алгоритм (`MultiplicationSerial`), параллельный алгоритм (`MultiplicationParallel`), алгоритм Штрассена (`MultiplicationStrassen`).

Также присутствует вспомогательный метод DotMultiplication для умножения строки матрицы на столбец другой матрицы.

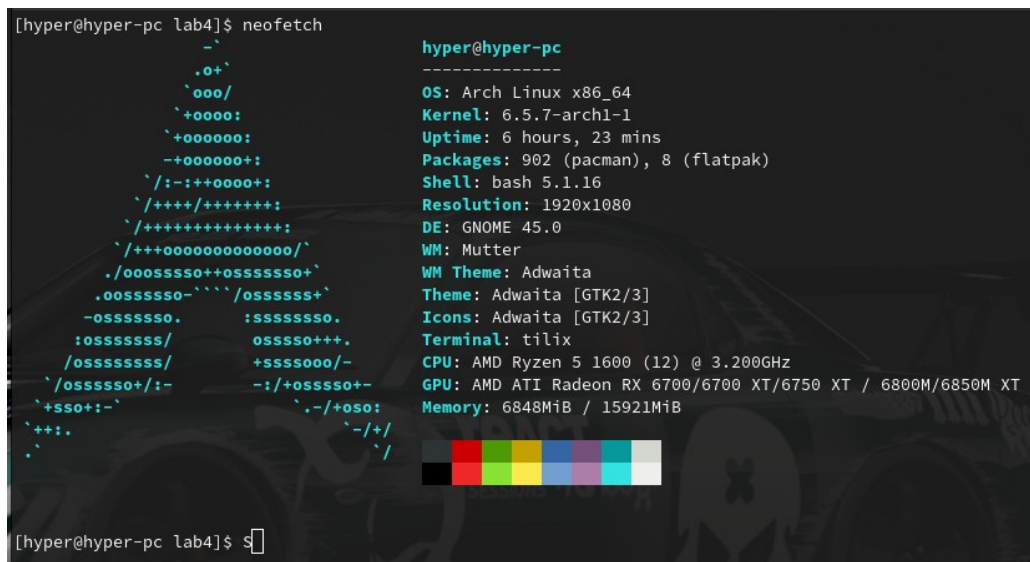
Параллельный алгоритм умножает элементы, за которые ответственен определенный поток (его индекс передается в качестве параметра, как и количество потоков). Это позволяет обеспечить максимальную загрузку каждого потока задачами.

Алгоритм Штрассена на первом этапе выполняет задачи параллельно, после первого этапа — последовательно. Это позволяет не создавать большое количество потоков умножения, что может привести к снижению производительности. Если размеры матриц меньше 16x16, то умножение производится последовательно.

2. Измерение времени работы алгоритмов умножения матриц.

Алгоритм параллельного умножения 4.1 повторяет реализацию параллельного алгоритма умножения матриц из работы 1, поэтому сравнение реализаций не уместно.

Для измерения зависимости времени выполнения программы с различными способами умножения матриц использовалась система, параметры которой отражены на рис. 1.



```
[hyper@hyper-pc lab4]$ neofetch
      .o+`
      `ooo/
      +oooo:
      +oooooo:
      -+oooooo+:
      /:-++oooo+:
      /++++/+++++++:
      /+++++++/+++++++:
      /++o000000000000/`
      ./000SSSS0++0SSSS0+`
      .0SSSSSS0-````/0SSSSS+`
      -0SSSSSS0.      :SSSSSS0.
      :0SSSSSSS/      0SSSS0+++.
      /0SSSSSSS/      +SSSS000/-
      /0SSSSS0+/:~    -:/+0SSSS0+-
      `+SS0+:-`      `.-/+0S0:
      `++:.          `.-/+/:
      .              `./`

hyper@hyper-pc
-----
OS: Arch Linux x86_64
Kernel: 6.5.7-arch1-1
Uptime: 6 hours, 23 mins
Packages: 902 (pacman), 8 (flatpak)
Shell: bash 5.1.16
Resolution: 1920x1080
DE: GNOME 45.0
WM: Mutter
WM Theme: Adwaita
Theme: Adwaita [GTK2/3]
Icons: Adwaita [GTK2/3]
Terminal: tilix
CPU: AMD Ryzen 5 1600 (12) @ 3.200GHz
GPU: AMD ATI Radeon RX 6700/6700 XT/6750 XT / 6800M/6850M XT
Memory: 6848MiB / 15921MiB
```

Рисунок 1. Параметры платформы, на которой производятся измерения

Для усреднения результатов программы запускаются 10 раз. Время, представленное в таблицах, указывается суммарное для 10 запусков. В расчеты включена генерация матриц, но для одинаковых размеров и разных типов умножения на результаты она не влияет. Количество потоков в параллельном умножении — 7. Измерение времени работы алгоритмов см. в табл. 1. В табл. 1 приведено реальное время выполнения задач в секундах (Real Time).

Таблица 1. Время работы программ умножения матриц.

Размер матриц	Последовательное умножение, сек	Параллельное умножение, сек	Алгоритм Штрассена, сек
64x64	0.144	0.062	0.081
128x128	0.491	0.256	0.328
256x256	2.811	0.962	1.211
512x512	21.678	5.339	5.810
1024x1024	170.963	36.100	33.270
2048x2048	4242.940	911.251	217.464

Как видно по табл. 1, для всех измерений последовательное умножение оказывается самым медленным. Для небольших матриц (до 512x512) параллельный алгоритм показывает себя лучше, так как он не использует копирование матриц, в отличие от алгоритма Штрассена. При этом для больших матриц (1024x1024 и более) время выполнения алгоритма Штрассена меньше времени выполнения параллельного алгоритма.

Выводы.

В ходе работы были изучены алгоритмы параллельного умножения матриц, изучен алгоритм Штрассена.

Было проведено исследование скорости выполнения последовательного, параллельного алгоритмов умножения матриц и

алгоритма Штрассена в зависимости от размеров матриц. Последовательный алгоритм неприменим для больших матриц, так как скорость его работы по сравнению с параллельным алгоритмом и алгоритмом Штрассена сильно ниже. Алгоритм параллельного умножения в 7 потоков работает быстрее алгоритма Штрассена до размеров матрицы 512×512 , после этого значения алгоритм Штрассена начинает работать быстрее параллельного алгоритма умножения. Для матриц 2048×2048 скорость работы алгоритма Штрассена оказалась в ~ 4 раза выше скорости работы параллельного алгоритма умножения для 7 потоков.

Таким образом, для больших квадратных матриц (1024×1024 и более), размер которых является степенью 2, алгоритм Штрассена более предпочтительный.