МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

**Лабораторная работа № 4**

**по курсу «Программирование графических процессоров»**

**Работа с матрицами. Метод Гаусса.**

Выполнил: Ф.М. Шавандрин

Группа: 8О-408Б

Преподаватель: А.Ю. Морозов

Москва, 2022

# **Условие**

# Необходимо вычислить *LU*-разложение квадратной матрицы: *A = LU*, где *A* *-* матрица *n x n, L –* нижняя треугольная матрица, с единичными элементами на диагонали, *U –* верхняя треугольная матрица. Дополнительно нужно получить вектор перестановок строк *p,* где *p[i]* содержит номер строки с которой произошла перестановка на *i-ой* итерации.

# **Входные данные:** На первой строке задано число *n* - размер матрицы. В следующих *n* строках, записано по *n* вещественных чисел -- элементы матрицы.

# .

# **Выходные данные:** Необходимо вывести на *n* строках, по *n* чисел -- элементы

# матриц *L* и *U,* объединенные в одну матрицу. Далее записываются *n* элементов вектора перестановок *p*.

# **Цель работы:** Использование объединения запросов к глобальной памяти. Реализация метода Гаусса с выбором главного элемента по столбцу. Ознакомление с библиотекой алгоритмов для параллельных расчетов *Thrust*. Использование *двухмерной сетки потоков.* Исследование производительности программы с помощью утилиты *nvprof.*

**Вариант 4.** LU-разложение матрицы.

# **Программное и аппаратное обеспечение**

## GPU:

* Название NVIDIA GeForce GT 545
* Сompute capability: 2.1
* Графическая память: 3150381056
* Разделяемая память: 49152
* Константная память: 32768
* Количество регистров на блок: 32
* Максимальное количество нитей: (1024, 1024, 64)
* Максимальное количество блоков: (65535, 65535, 65535)
* Количество мультипроцессоров: 3

**Сведения о системе:**

* Процессор: Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
* ОЗУ: 15 ГБ
* HDD 500 ГБ

**Программное обеспечение:**

* OS: Ubuntu 16.04.6 LTS
* Текстовый редактор: Vim
* Компилятор: nvcc

# **Метод решения**

Для нахождения *LU-*разложения матрицы будем использовать метод Гаусса с выбором главного элемента по столбцу (максимального по модулю). Если на текущей итерации индекс стобца матрицы не равен индексу стобца с главным элементов, то меняем местами строки матрицы с теми же индексами. Иначе считаем коэффициенты *l* и выполняем итерацию метода Гаусса — для заданной подматрицы из каждой строки вычитаем главную строку, домноженную на соответствующий коэффициент.

# **Описание программ****ы**

# В программе реализовано три ядра. Ядро *kernel\_swap\_rows* принимает на вход две строки и меняет их местами. Для выполнения такой операции используется одномерная сетка потоков.

# Ядро *kernel\_gauss* принимает на вход не всю исходную матрицу, а только ту часть, в которой нужно применить метод Гаусса. Таким образом, мы будем изменять значения только тех элементов матрицы, которые будут использоваться в дальнейшем. Для выполнения этой операции используется двухмерная сетка потоков.

# Ядро *kernel\_calc\_l* принимает на вход строку и считает для неё коэффициенты *l* для дальнейшего построения матрицы *L.* Для выполнения данной операции используется одномерная сетка потоков.

# Нахождение главного элемента по столбцу реализовано с использованием библиотеки *Thrust*. Для того, чтобы было удобнее считать максимум по столбцам, будем хранить исходную матрицу по столбцам.

# **Результаты**

# Для тестирования программы на GPU, будем сравнивать её с программой на CPU, обрабатывающей матрицы размерами 100 х 100, 500 х 500, 1000 х 1000 соответственно.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Размер матрицы | | 100 х 100, мс | 500 х 500, мс | 1000 х 1000, мс | |
| Размер сетки ядра | CPU | 3.78 | 271.32 | | 1734.65 |
| <<<(1,32), (1,32)>>> | 43.12 | 451.11 | | 2380.22 |
| <<<(32,32), (32,32)>>> | 36.91 | 298.55 | | 1496.34 |
| <<<(1,128), (1,128)>>> | 51.15 | 511.78 | | 2325.07 |
| <<<(1,256), (1,32)>>> | 42.78 | 440.63 | | 2471.47 |

# Применим утилиту *nvprof* для исследования производительности программы. В качестве теста буду использовать матрицу размером 100 х 100. Проанализируем результат работы только для своих ядер (ядра от *Thrust* рассматривать не буду).

# 

# На скриншоте видно, что в ядре, выполняющим метод Гаусса, параметр *global\_store\_transaction*, отвечающий за количество обращений (транзакций) к глобальной памяти, ниже числа элементов матрицы. Таким образом, происходит оптимизация программы путем объединения запросов к глобальной памяти.

# **Выводы**

# В данной лабораторной работе реализовал метод Гаусса на *GPU*, с помощью которого научился находить *LU*-разложение квадратной матрицы. Также познакомился с библиотекой *Thrust*, в которой реализованы алгоритмы для параллельных рассчетов на *CUDA.* Для анализа программы впервые использовал профилировщик *nvprof*.