

**МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)**

**Институт №8 «Компьютерные науки и прикладная математика»  
Кафедра 806 «Вычислительная математика и программирование»**

**Лабораторная работа №2  
по курсу «Параллельная обработка данных»**

**Работа с матрицам. Метод Гаусса.**

**Выполнил: В. В. Бирюков  
Группа: 8О-407Б  
Преподаватель: А. Ю. Морозов**

**Москва, 2022**

## Условие

**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.

**Вариант:** 3. Решение квадратной СЛАУ.

## Программное и аппаратное обеспечение

Характеристики графического процессора:

- Наименование: GeForce GT 545
- Compute capability: 2.1
- Графическая память: 3150381056 Б
- Разделяемая память на блок: 49152 Б
- Количество регистров на блок: 32768
- Максимальное количество потоков на блок: (1024, 1024, 64)
- Максимальное количество блоков: (65535, 65535, 65535)
- Константная память: 65536 Б
- Количество мультипроцессоров: 3

Характеристики системы:

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

Программное обеспечение:

- ОС: Ubuntu 16.04.6 LTS
- IDE: Visual Studio Code 1.72
- Компилятор: nvcc 7.5.17

## Метод решения

Для решения квадратной СЛАУ необходимо привести матрицу системы к верхнетреугольному виду, при помощи прямого хода метода Гаусса, затем найти решение обратным ходом ( $x_n$  тривиально находится из единственного элемента последней строки,  $x_{n-1}$  — через  $x_n$  и так далее).

Один шаг метода Гаусса состоит из нескольких этапов: выделение ведущего элемента (максимального в столбце), перестановка строки с ведущим элементом на первое место для данного шага, зануление текущего столбца. Суммарно необходимо выполнить  $n$  шагов.

Сложность прямого хода —  $O(n^3)$ , обратного —  $O(n^2)$ . Итого —  $O(n^3)$ .

## Описание программы

В файле lab4.cu расположен основной код программы. Функция `gaussian_solver_step` — вычислительное ядро — осуществляет финальный этап шага метода Гаусса. Функция `swap_rows` — другое ядро — переставляет местами две строки матрицы. Максимальный элемент столбца находится при помощи функции `thrust::max_element`. Таким образом весь прямой ход метода Гаусса осуществляется над матрицей в видеопамяти. Обратный ход не распараллеливается, так как его сложность значительно меньше, и целиком выполняется на CPU.

## Результаты

### Анализ программы профилировщиком

Конфигурация ядра:  $32 \times 32$ ,  $32 \times 32$ ; размер тестовой матрицы:  $2000 \times 2000$ . Из вывода nvprof убраны нечитаемые названия ядер thrust.

В ядре gaussian\_solver\_step потоки варпа обрабатывают столбцы матрицы:

Invocations	Event Name	Min	Max	Avg
Device "GeForce GT 545 (0)"				
Kernel: gaussian_solver_step(double*, int, int)				
2000	global_store_transaction	0	500208	161671
Kernel: swap_rows(double*, int, int, int)				
1994	global_store_transaction	6144	6144	6144

В ядре gaussian\_solver\_step потоки варпа обрабатывают строки матрицы:

Invocations	Event Name	Min	Max	Avg
Device "GeForce GT 545 (0)"				
Kernel: gaussian_solver_step(double*, int, int)				
2000	global_store_transaction	0	4002432	1335062
Kernel: swap_rows(double*, int, int, int)				
1994	global_store_transaction	6144	6144	6144

Видно, что во втором случае происходит гораздо больше обращений к глобальной памяти, т.е. объединения запросов не происходит. На практике это приводит к увеличению времени работы в 2-3 раза.

### Сравнение времени работы

Размер матрицы	100	500	1000	2000	5000	10000
Конфигурация	Время выполнения, мс					
CPU	14.8886	788.532	5958.03	47452.10	745217	-
$1 \times 1$ , $32 \times 32$	36.4768	256.299	1145.10	8840.49	114357	-
$32 \times 32$ , $32 \times 32$	53.7845	287.704	932.85	4461.41	43472	411956
$64 \times 64$ , $32 \times 32$	99.7329	547.723	1513.98	6305.97	51747	466804
$128 \times 128$ , $32 \times 32$	284.3100	1559.010	3640.97	10970.80	73901	518470

## Выводы

В ходе выполнения лабораторной работы я познакомился с параллельной реализацией метода Гаусса. При работе с матрицами сильно ощущается эффект распараллеливания, так как очень большое количество матричных операций имеют кубическую сложность.

Я наглядно убедился в важности объединения запросов к глобальной памяти — если его не происходит, время выполнения алгоритма увеличивается в 2-3 раза. Оптимальная конфигурация ядра: сетка блоков —  $32 \times 32$ , сетка потоков —  $32 \times 32$ .