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 матрицам. Метод Гаусса.

Выполнил: Р.С. Лисин
Группа: 8О-406Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2023

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. Использование двухмерной сетки потоков. Исследование производительности программы с помощью утилиты nvprof (обязательно отразить в отчете).

Вариант 3. Решение квадратной СЛАУ.:

Входные данные. На первой строке задано число n - размер матрицы. В следующих n строках, записано по n вещественных чисел - элементы матрицы. Далее записываются n элементов вектора свободных коэффициентов. $n \leq 10000$.

Выходные данные. Необходимо вывести n значений, являющиеся элементами вектора неизвестных x.

Программное и аппаратное обеспечение

В качестве графического процессора использую видеокарту NVIDIA Tesla T4.

```
Compute capability : 7.5
Name : Tesla T4
Total Global Memory : 15835398144
Shared memory per block : 49152
Registers per block : 65536
Warp size : 32
Max threads per block : (1024, 1024, 64)
Max block : (2147483647, 65535, 65535)
Total constant memory : 65536
Multiprocessors count : 40
```

В качестве редактора кода использовался Jupyter Notebook в Google Colab.

Метод решения

Чтобы решить квадратную СЛАУ, нужно привести матрицу к верхнетреугольному виду с помощью прямого хода Гаусса, выделяя максимальный элемент в столбце (ведущий элемент) и выполняя перестановку строки с ведущим элементом на первое место для данного шага. После этого нужно найти решение с помощью обратного хода метода Гаусса. На GPU реализуем шаг прямого хода метода Гаусса и обмен местами строк матрицы. Обратный ход имеет меньшую сложность, и реализуется на CPU.

Описание программы

В ядре swap производится обмен местами двух строк матрицы. В ядре gauss_method реализуется прямой шаг метода Гаусса. Максимальный элемент в столбце ищется при помощи библиотеки thrust.

Результаты

Исследуем производительность программы с помощью утилиты ncu, так как compute capability равно 7.5. В качестве теста используется матрица 1000x1000 элементов.

```
gauss_method(double *, int, int), 2023-Oct-27 14:15:38, Context 1, Stream 7 Section: Command
line profiler metrics ---------------------------------------------------------------------- -
-------------- ------------------------------
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.avg 0
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.max 0
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.min 0
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.sum 0
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.avg 0
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.max 0
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.min 0
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.sum 0
l1tex__t_sectors_pipe_lsu_mem_global_op_ld.avg sector 15,368.90
l1tex__t_sectors_pipe_lsu_mem_global_op_ld.max sector 614,756
l1tex__t_sectors_pipe_lsu_mem_global_op_ld.min sector 0
l1tex__t_sectors_pipe_lsu_mem_global_op_ld.sum sector 614,756
l1tex__t_sectors_pipe_lsu_mem_global_op_st.avg sector 7,025
l1tex__t_sectors_pipe_lsu_mem_global_op_st.max sector 281,000
l1tex__t_sectors_pipe_lsu_mem_global_op_st.min sector 0
l1tex__t_sectors_pipe_lsu_mem_global_op_st.sum sector 281,000
sm__sass_inst_executed_op_local.avg inst 0 sm__sass_inst_executed_op_local.max inst 0
sm__sass_inst_executed_op_local.min inst 0 sm__sass_inst_executed_op_local.sum inst 0
smsp__branch_targets_threads_divergent 0
```

Отметим, что в ядре gauss_method количество обращений к памяти внутри варпа меньше количества элементов матрицы. Следовательно, происходит объединение запросов к глобальной памяти.

Рассмотрим время работы программы на различных тестах при различных размерах сетки и на CPU. Будем замерять непосредственно время работы алгоритма. В качестве тестов используется квадратная матрица с разными размерами. Результаты приведены в таблице ниже.

Размер сетки ядра / Размер матрицы	100x100, мс	500x500, мс	1000x1000, мс
CPU	6.63181	806.461	7110.16
<<<(1, 32), (1, 32)>>>	42.007584	1755.519653	14104.806641
<<<(32, 32), (32, 32)>>>	29.524223	332.577515	991.482666
<<<(64, 64), (32, 32)>>>	21.367104	306.134094	1014.023926
<<<(128, 128), (32, 32)>>>	21.410944	307.419037	1002.386841
<<<(128, 128), (64, 64)>>>	21.695232	339.031586	1015.358032

Алгоритм на CPU справляется медленнее чем на GPU. Это безусловно связано с тем, что в данном случае распараллеливание в разы ускоряет работу алгоритма.

Выводы

В четвёртой лабораторной работе я реализовал метод Гаусса для решения квадратной СЛАУ на GPU. Графический процессор ускоряет в разы нахождение корней системы уравнений. Также я познакомился с утилитой ncu для профилирования программ на GPU.