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: *И. П. Попов*
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.
В качестве вещественного типа данных необходимо использовать тип данных double. Библиотеку Thrust использовать только для поиска максимального элемента на каждой итерации алгоритма. В вариантах(1,5,6,7), где необходимо сравнение по модулю с нулем, в качестве нулевого значения использовать 10^{-7} . Все результаты выводить с относительной точностью 10^{-10}.

Вариант 3. Решение квадратной СЛАУ.
Необходимо решить систему уравнений Ax = b, где A -- квадратная матрица n x n, b -- вектор-столбец свободных коэффициентов длинной n, x -- вектор неизвестных.

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.
$n \leq 10^{-4}$.
Выходные данные. Необходимо вывести n значений, являющиеся элементами вектора неизвестных x.

Программное и аппаратное обеспечение

NVIDIA GeForce GTX 1660 Super:
Compute capability: 7.5
Dedicated video memory: Typically 6 GB (may vary by manufacturer)
Shared memory per block: 49152 bytes
Register per block: 65536 bytes
Total constant memory: 65536 bytes
Max threads per multiprocessor: 2048
Max threads per block: 1024

CPU AMD Ryzen 3600X
Physical cores: 6
Threads: 12
Base clock speed: 3.8 GHz
Boost clock speed: 4.4 GHz
L1 cache: 384KB (per core)
L2 cache: 512KB (per core)

L3 cache: 32 MB (shared)
Chip lithography: 7 nm

16 Гб RAM
1 ТБ HDD
OS – Windows 11 ProWSL, IDE – VS Code, compiler - nvcc

Метод решения

Программа, использует метод Гаусса с частичным выбором (partial pivoting) для решения системы линейных уравнений.

Описание программы

Для доступа к элементу используется формула matrix[j*n + i], где j - номер столбца, i - номер строки. Это представление выбрано для хранения матрицы "по столбцам", что упрощает реализацию метода Гаусса.

Алгоритм заключается в преобразовании матрицы в верхнюю треугольную форму при помощи метода Гаусса. На каждой итерации метода выбирается максимальный элемент в столбце из позиций (i, i) - (n, i). Если максимальный элемент не соответствует текущей позиции (i, i), происходит обмен строк. Затем каждая строка под текущей складывается с текущей строкой с использованием определенной формулы, таким образом обнуляя элемент под текущим диагональным элементом.

Представление матрицы в верхней треугольной форме позволяет решить уравнение за квадратичное время, просто выражая каждую переменную X, начиная с нижней строки. Решение уравнения выполняется на центральном процессоре (CPU).

Результаты

Все измерения представлены в ms
Конфигурация представлена размерностью функции swapLines(сверху) и columnSwap(снизу)

	10*10	100*100	1000*1000
<<<(16, 16)>>>, <<<(16, 16), (16, 16)>>>	1.319232	11.667744	1015.581482
<<<(64, 64) >>>, <<<(32, 32), (16, 16)>>>	1.512800	11.868640	1020.329590
<<<(128, 128) >>>, <<< (32, 32), (32, 32)>>>	1.522048	13.062144	1003.864075

<<<(512, 512) >>> <<< (64, 64), (32, 32)>>>	2.095424	17.721151	1025.087036
<<<(1024,1024) >>> <<<(128, 128), (32, 32)>>>	3.933632	36.483391	1052.189697
CPU	3.015972	7.200488	404515.837495

В качестве теста используется матрица 1000x1000 элементов.
columnSwap(double *, int, int), 2023-Nov-10 16:45:38, 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

Выводы

Применение параллельных вычислений на GPU с использованием CUDA технологии позволяет эффективно распараллеливать операции над матрицами и векторами.

Из проведенных тестов видно, что вычисления на GPU проходят в 2-3 раза быстрее, чем на CPU. Это связано с параллельной обработкой данных на множестве ядер GPU, что позволяет эффективно использовать ресурсы устройства. Благодаря использованию библиотеки Thrust и оптимизированным алгоритмам параллельных вычислений, удалось уменьшить количество запросов к глобальной памяти, что также положительно сказалось на производительности программы.