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 матрицами. Метод Гаусса

Выполнил: И.А. Мариничев
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников
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. Использование двухмерной сетки потоков. Исследование производительности программы с помощью утилиты nvprof.

Вариант 3: решение квадратной СЛАУ

Программное и аппаратное обеспечение

Compute capability	:	2.1
Name	:	GeForce GT 545
Total Global Memory	:	3150381056
Shared memory per block	:	49152
Registers per block	:	32768
Warp size	:	32
Max threads per block	:	(1024, 1024, 64)
Max block	:	(65535, 65535, 65535)
Total constant memory	:	65536
Multiprocessors count	:	3
Processor	:	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
RAM	:	16 GB
Drive	:	349G
OS	:	Ubuntu 16.04.6 LTS
IDE	:	Visual Studio Code
Compiler	:	NVIDIA (R) Cuda compiler driver V7.5.17

Метод решения

Считываем входные данные, при этом записываем матрицу и вектор свободных коэффициентов в памяти по столбцам, чтобы можно было искать максимум на непрерывных участках памяти. Далее запускаем на CPU цикл, который будет проходить по всем столбцам от первого до предпоследнего и выполнять прямой проход метода Гаусса с выбором главного элемента. После этого мы получим верхнетреугольную матрицу по которой можно будет за линейный цикл (обратный ход) найти каждую из неизвестных. В конце выводим полученный ответ и освобождаем память.

Описание программы

```
// переопределение оператора "()" для экземпляра struct comparator
__host__ __device__ bool operator()(double a, double b)

// функция меняющая местами две строки
__global__ void swap_rows(double *system, int curr_id, int max_id, int n)

// функция "зануления" всех элементов ниже данного
__global__ void subtract_row_from_rows_below(double *system, int curr_id, int n)

// прямой ход метода Гаусса
for (int i = 0; i < n - 1; i++) {
	// выполняем приведение типов
	thrust::device_ptr<double> system_ptr(dev_system + i * n);

	// ищем максимум в массиве на GPU
	max_id_ptr = thrust::max_element(system_ptr + i, system_ptr + n,
compare_by_absolute_value);
	max_id = max_id_ptr - system_ptr;

	if (i != max_id) {
		swap_rows<<<numBlocks, threadsperBlock>>>(dev_system, i, max_id, n);
		CSC(cudaGetLastError());
	}
	subtract_row_from_rows_below<<<numBlocks2D,
threadsperBlock2D>>>(dev_system, i, n);
	CSC(cudaGetLastError());
}

// находим вектор неизветсных x
for (int i = n - 1; i >= 0; i--) {
	x[i] = system[n * n + i];
	for (int j = n - 1; j > i; j--) {
		x[i] -= system[i + j * n] * x[j];
	}
	x[i] /= system[i * n + i];
}
```

Результаты

Конфигурация	Размер теста (время указано в ms)			
	O(10^2)	O(10^4)	O(10^6)	O(10^7)
<<<(32, 32), (32, 32)>>>	3.002048	51.896862	1061.817505	10175.867188
<<<(64, 64), (32, 32)>>>	7.687136	106.179970	2017.932495	17627.330078
<<<(128, 128), (32, 32)>>>	24.848608	297.967468	4871.255371	34100.488281
<<<(256, 256), (32, 32)>>>	94.211555	1072.585083	14384.940430	77266.398438
<<<(512, 512), (32, 32)>>>	369.886505	4127.046387	48633.371094	211602.203125
<<<(1024, 1024), (32, 32)>>>	1465.250000	16266.955078	178013.781250	672684.937500
CPU	4.0	598.0	3.32889e+06	1.66617e+08

Результаты профилирования

Ниже представлены результаты профилирования программы при n = 1000

1) конфигурация <<<(32, 32), (32, 32)>>>:

```
==7624== Event result:
Invocations                                Event Name         Min         Max         Avg
Device "GeForce GT 545 (0)"
        Kernel: subtract_row_from_rows_below(double*, int, int)
        999                           divergent_branch           0       30720        1032
        999                  global_store_transaction           6      125457       40583
        999                   l1_shared_bank_conflict           0           0           0
        999                         l1_local_load_hit           0           0           0

==7624== Metric result:
Invocations                                Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GT 545 (0)"
        Kernel: subtract_row_from_rows_below(double*, int, int)
        999                               sm_efficiency                   Multiprocessor Activity      66.49%      97.99%      90.06%
```

2) конфигурация <<<(256, 256), (32, 32)>>>:

```
==8093== Event result:
Invocations                                Event Name         Min         Max         Avg
Device "GeForce GT 545 (0)"
        Kernel: subtract_row_from_rows_below(double*, int, int)
        999                           divergent_branch           0      245760        8257
        999                  global_store_transaction           6      124872       40538
        999                   l1_shared_bank_conflict           0           0           0
        999                         l1_local_load_hit           0           0           0


==8093== Metric result:
Invocations                                Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GT 545 (0)"
        Kernel: subtract_row_from_rows_below(double*, int, int)
        999                               sm_efficiency                   Multiprocessor Activity      60.99%      82.92%      71.40%
```

Выводы

Алгоритмы с высокой сложностью, к которым относится метод Гаусса со сложностью O(n^3), довольно просто параллелятся и дают хороший выигрыш по времени. Как видно из результатов сравнения, GPU дает значительный прирост по времени, но почему-то только при конфигурации <<<(32, 32), (32, 32)>>>. Результаты профилирования показали, что в остальных конфигурациях растет дивергенция потоков, что и влияет на снижение производительности.