МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

**Лабораторная работа №1**

**по курсу «Параллельная обработка данных»**

**Сортировка чисел на GPU. Свертка, сканирование, гистограмма.**

Выполнил: Гамов П.А.

Группа: 8О-407Б-18

Преподаватели: К.Г. Крашенинников,

А.Ю. Морозов

Москва, 2022

**Условие**

Ознакомление с фундаментальными алгоритмами GPU: свертка

(reduce), сканирование (blelloch scan) и гистограмма (histogram). Реализация одной из

сортировок на CUDA. Использование разделяемой и других видов памяти.

Исследование производительности программы с помощью утилиты nvprof.

Вариант 1

Требуется реализовать битоническую сортировку для чисел типа int.

Должна быть реализована адаптивная операция битонического слияния. Если

данные помещаются в разделяемую память, то взаимодействие идет через неё, если

нет, то через глобальную память (т.е. необходимо реализовать несколько вариантов

ядра).

Ограничения: n ≤ 256 \* 10^6

**Программное и аппаратное обеспечение**

nvcc 7.0

Ubuntu 14.04 LTS

|  |  |
| --- | --- |
| Compute capability | 6.1 |
| Name | GeForce GTX 1050 |
| Total Global Memory | 2096103424 |
| Shared Mem per block | 49152 |
| Registers per block | 65534 |
| Max thread per block | (1024,1024,64) |
| Max block | (2147483647, 65535, 65535) |
| Total constant memory | 65536 |
| Multiprocessor’s count | 5 |

**Метод решения**

Метод решения построен на применении полу очистителей в правильной последовательности.

**Описание программы**

Организованны два ядра, один работает на глобальной памяти, второй на shared memory. Процесс битонного слияния преобразует битонную последовательность в полностью отсортированную последовательность. Алгоритм битонной сортировки состоит из применения битонных преобразований до тех пор, пока множество не будет полностью отсортировано.

**Результаты**

|  |  |  |
| --- | --- | --- |
|  | 10^5 | 10^6 |
| 10, 1024 | 1.3 sec | 10.9 sec |
| 256, 1024 | 1.3 sec | 9.1 sec |
| 1024, 1024 | 1.3 sec | 10.1 sec |

user74@server-i72:~/5lab pod$ nvprof ./a.out < data.t > res.t

==5680== NVPROF is profiling process 5680, command: ./a.out

==5680== Profiling application: ./a.out

==5680== Profiling result:

Time(%) Time Calls Avg Min Max Name

94.67% 9.98508s 171 58.392ms 21.442ms 102.20ms B\_shared(int\*, int, int, int)

5.22% 550.80ms 105 5.2458ms 3.9359ms 51.023ms B\_global(int\*, int, int, int)

0.06% 5.8555ms 1 5.8555ms 5.8555ms 5.8555ms [CUDA memcpy HtoD]

0.05% 5.5261ms 1 5.5261ms 5.5261ms 5.5261ms [CUDA memcpy DtoH]

==5680== API calls:

Time(%) Time Calls Avg Min Max Name

99.27% 10.5374s 276 38.179ms 3.9406ms 102.21ms cudaDeviceSynchronize

0.59% 63.099ms 1 63.099ms 63.099ms 63.099ms cudaMalloc

0.11% 11.582ms 2 5.7912ms 5.6785ms 5.9039ms cudaMemcpy

0.02% 1.9692ms 276 7.1340us 5.0420us 40.058us cudaLaunch

0.00% 440.85us 83 5.3110us 177ns 186.95us cuDeviceGetAttribute

0.00% 161.87us 1104 146ns 110ns 6.7420us cudaSetupArgument

0.00% 147.30us 1 147.30us 147.30us 147.30us cudaFree

0.00% 83.072us 1 83.072us 83.072us 83.072us cuDeviceTotalMem

0.00% 82.165us 276 297ns 227ns 1.1430us cudaGetLastError

0.00% 62.342us 276 225ns 186ns 3.1600us cudaConfigureCall

0.00% 51.499us 1 51.499us 51.499us 51.499us cuDeviceGetName

0.00% 2.1160us 2 1.0580us 418ns 1.6980us cuDeviceGetCount

0.00% 679ns 2 339ns 255ns 424ns cuDeviceGet

**Выводы**

Я смог написать алгоритм битонной сортировки, который как и метод чет-нечет был разработан как раз на применении многопоточности, рад что познакомился с таким культовым и быстрым способом сортировки чисел.