МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №4
по курсу «Параллельная обработка данных»

Сортировка чисел на GPU. Свертка, сканирование, гистограмма.

Выполнил: М.А.Трофимов
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

Цель работы. 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 (обязательно отразить в отчете).

Вариант 2: Сортировка подсчетом. Диапазон от 0 до 2^{24}-1.

Программное и аппаратное обеспечение

Характеристики GPU "NVIDIA GeForce GTX 950"

CUDA Driver Version / Runtime Version 11.4 / 11.4
CUDA Capability Major/Minor version number: 5.2
Total amount of global memory: 1997 MBytes (2094137344 bytes)
(006) Multiprocessors, (128) CUDA Cores/MP: 768 CUDA Cores
GPU Max Clock rate: 1278 MHz (1.28 GHz)
Memory Clock rate: 3305 Mhz
Memory Bus Width: 128-bit
L2 Cache Size: 1048576 bytes
Maximum Texture Dimension Size (x,y,z) 1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)
Maximum Layered 1D Texture Size, (num) layers 1D=(16384), 2048 layers
Maximum Layered 2D Texture Size, (num) layers 2D=(16384, 16384), 2048 layers
Total amount of constant memory: 65536 bytes
Total amount of shared memory per block: 49152 bytes
Total shared memory per multiprocessor: 98304 bytes
Total number of registers available per block: 65536
Warp size: 32
Maximum number of threads per multiprocessor: 2048
Maximum number of threads per block: 1024
Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)
Maximum memory pitch: 2147483647 bytes
Texture alignment: 512 bytes

Характеристики CPU Intel i5-4460

# of Cores 4
# of Threads 4
Processor Base Frequency 3.20 GHz
Max Turbo Frequency 3.40 GHz
Cache 6 MB Intel® Smart Cache
Bus Speed 5 GT/s
Intel® Turbo Boost Technology 2.0 Frequency 3.40 GHz
TDP 84 W

Характеристики RAM

Total 15 Gi
Swap 2 Gi

Операционная система: Ubuntu 20.04 LTE
IDE Sublime Text 3
Compiler nvcc for cuda 11.4

Метод решения

Алгоритм сортировки подсчётом реализован в три этапа:

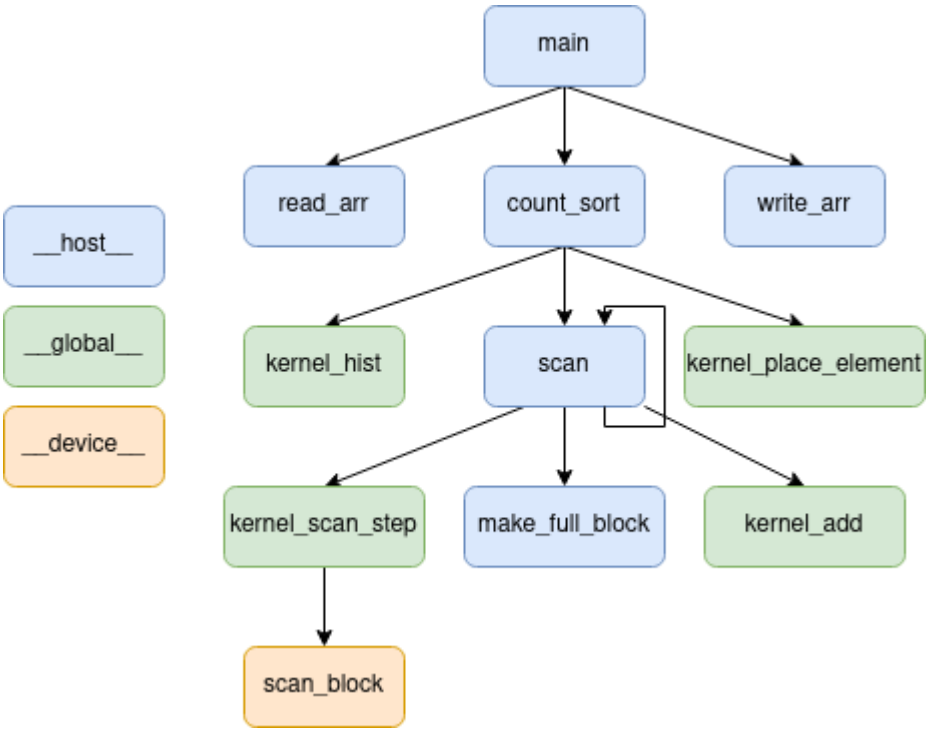
1. Построения гистограммы по входному массиву;
2. Применение сканирования полученной гистограммы;
3. Заполнение входного массива отсортированными числами.

Алгоритм гистограммы реализован наивным методом с использованием атомарных операций.

Алгоритм сканирования реализован через рекурсию, где на каждом шаге рекурсии мы “разбиваем” массив входной на блоки размера равного удвоенному количеству потоков, которые мы вызываем, для каждого блока выполняем скан на разделяемой памяти. Использовался алгоритм Blelloch Scan, но с небольшой модификацией, благодаря чему получился не исключающий скан, а включающий, т.е. префиксная сумма в scan[i] = a[0] + a[1] + … + a[i]. Затем из каждого блока копировался последний элемент в новый массив, для которого снова запускался скан, пока количество блоков не станет равным одному. Поскольку предполагается, что все блоки одинакового размера, то массив с последними элементами блоков дополняется нулями, пока размер массива полученный не станет кратным размеру блока. На обратном пути рекурсии параллельно каждый блок увеличивается на значение, соответствующее этому блоку из созданного массива.

Алгоритм заполнения элементами массива заключается в том, что мы запускаем параллельное заполнение массива так, что каждый iый блок ядра обрабатывает число i.

Описание программы



*_arr - функции для ввода-вывода массива;
count_sort - сортировка подсчётом;

kernel_hist - построение гистограммы;
scan - рекурсивная функция запускающая процесс сканирования на каждом шаге;
kernel_scan_step - сканирование блоков одного массива;
scan_block - функция сканирования блока с разделяемой памятью алгоритмом Blelloch scan;
make_full_block - функция для определения размера массива, чтобы этот массив был кратен размеру блока;
kernel_add - увеличение каждого блока на соответствующее значение.

Результаты

Конфигурация	Тест размера 135	Тест размера 13500	Тест размера 1350000	Тест размера 135000000
На CPU	1304.9ms	1292.96ms	1371.34ms	5273.9ms
1,32	4745.96ms	4694.1ms	4732.38ms	6422.49ms
32, 32	233.266ms	211.726ms	208.347ms	433.712ms
64,64	119.49ms	110.913ms	105.054ms	301.328ms
256,256	182.261ms	166.191ms	161.451ms	360.342ms
512,512	315.951ms	293.621ms	290.686ms	484.519ms
1024, 1024	576.247ms	587.141ms	568.434ms	765.19ms

```
==20276== Profiling result:
==20276== Event result:
Invocations                                Event Name         Min         Max         Avg       Total
Device "NVIDIA GeForce GTX 950 (0)"
    Kernel: kernel_scan_step(unsigned int*, unsigned int)
          3                             global_store          96      786432      262304      786912
          3                              global_load          64      524288      174869      524608
          3                    shared_ld_bank_conflict        1575    12902400     4303425    12910275
          3                    shared_st_bank_conflict         945     7741440     2582055     7746165
    Kernel: kernel_place_element(unsigned int*, unsigned int*, unsigned int)
          1                             global_store    16777215    16777215    16777215    16777215
          1                              global_load  1073741824  1073741824  1073741824  1073741824
          1                    shared_ld_bank_conflict           0           0           0           0
          1                    shared_st_bank_conflict           0           0           0           0
    Kernel: kernel_add(unsigned int*, unsigned int*, unsigned int)
          2                             global_store         192      524224      262208      524416
          2                              global_load         384     1048448      524416     1048832
          2                    shared_ld_bank_conflict           0           0           0           0
          2                    shared_st_bank_conflict           0           0           0           0
    Kernel: kernel_hist(unsigned int*, unsigned int*, unsigned int)
          1                             global_store           0           0           0           0
          1                              global_load     4218750     4218750     4218750     4218750
          1                    shared_ld_bank_conflict           0           0           0           0
          1                    shared_st_bank_conflict           0           0           0           0

==20276== Metric result:
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "NVIDIA GeForce GTX 950 (0)"
    Kernel: kernel_scan_step(unsigned int*, unsigned int)
          3                          branch_efficiency                          Branch Efficiency     100.00%     100.00%     100.00%
          3                            global_hit_rate          Global Hit Rate in unified l1/tex       0.00%       0.00%       0.00%
          3                             local_hit_rate                             Local Hit Rate       0.00%       0.00%       0.00%
    Kernel: kernel_place_element(unsigned int*, unsigned int*, unsigned int)
          1                          branch_efficiency                          Branch Efficiency     100.00%     100.00%     100.00%
          1                            global_hit_rate          Global Hit Rate in unified l1/tex       0.00%       0.00%       0.00%
          1                             local_hit_rate                             Local Hit Rate       0.00%       0.00%       0.00%
    Kernel: kernel_add(unsigned int*, unsigned int*, unsigned int)
          2                          branch_efficiency                          Branch Efficiency     100.00%     100.00%     100.00%
          2                            global_hit_rate          Global Hit Rate in unified l1/tex       0.00%       0.00%       0.00%
          2                             local_hit_rate                             Local Hit Rate       0.00%       0.00%       0.00%
    Kernel: kernel_hist(unsigned int*, unsigned int*, unsigned int)
          1                          branch_efficiency                          Branch Efficiency     100.00%     100.00%     100.00%
          1                            global_hit_rate          Global Hit Rate in unified l1/tex       0.00%       0.00%       0.00%
          1                             local_hit_rate                             Local Hit Rate       0.00%       0.00%       0.00%
```

Выводы

Как видно, параллелизация достаточно неплохо улучшает время работы сортировки, однако видно, что есть минимум (конфигурация 64, 64), после которого увеличение количества блоков и потоков не даёт выигрыша.