МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

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

Лабораторная работа №3

по курсу «Параллельная обработка данных»

Сортировка чисел на GPU. Свертка, сканирование, гистограмма.

Выполнил: Д. А. Ваньков
Группа: 8О-407Б-17
Преподаватели: А.Ю. Морозов,
К.Г. Крашенинников

Москва, 2020

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.

Вариант 5. Сортировка чет-нечет с предварительной битонической сортировкой.

Программное и аппаратное обеспечение

Graphics card: GeForce 940M

Размер глобальной памяти: 4242604032

Размер константной памяти: 65536

Размер разделяемой памяти: 49152

Максимальное количество регистров на блок: 65536

Максимальное количество потоков на блок: 1024

Количество мультипроцессоров: 3

OS: Linux Ubuntu 18.04

Редактор: CLion, Atom

Машины в кластере:

1. Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz, 16 Gb, GeForce GTX 1050, 2 Gb
2. Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz, 16 Gb, GeForce GT 545, 3 Gb
3. Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz, 16 Gb, GeForce GTX 650, 2 Gb
4. Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz, 12 Gb, GeForce GT 530, 2 Gb
5. Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz, 8 Gb, GeForce GT 530, 2 Gb

Все машины соединены гигабатным ethernet и находятся в подсети 10.10.1.1/24.

Версии софта: mpirun 1.10.2, g++ 4.8.4, nvcc 7.0

Метод решения

Так как сортировка чет-нечет состоит из нескольких этапов я разбил свой код на части. В первом этапе я реализовал битоническую сортировку с слиянием. В main я выделил память для графического процессора, предварительно добив ее до размера, удовлетворяющему 2^n числами INT_MAX. Затем в цикле, до 1024 (максимальный размер блока) я сортирую каждый из блоков, проходя по всему массиву чисел, тем самым завершая предварительный этап. На втором этапе я выполняю 2 * n, где n — это количество блоков разбиения, проходов. Первый проход представляет собой четный проход с перестановкой элементов в порядке возрастания, каждый из элементов массива помещается в свой поток, также используется разделяемая память для второго этапа. Второй проход представляет собой слияние блоков их пере разбиение, происходит это с использованием разделяемой памяти. От начального и конечного блоков откусываются половинки, которые становятся самостоятельными блоками, остальные блоки разбиваются на 2 части и сливаются в один. Из-за того, что все элементы были упорядочены по возрастанию, а для битонического слияния требуется разнонаправленное упорядочивание в одной половинке блока используется инвертирование индексов. Последний этап представляет собой возвращение данных с GPU и их вывод в бинарном виде в stdout.

Описание программы

Данный цикл пробегает по всему массиву и сортирует все блоки.

```
for (int k = 2; k <= upd_size; k *= 2) {
if (k > BLOCK_SIZE)
break;
// Merge and split step
for (int j = k / 2; j > 0; j /= 2) {
bitonic_sort_step<<<NUM_BLOCKS, BLOCK_SIZE>>>(dev_data, j, k, upd_size);
CUDA_ERROR(cudaGetLastError());
        }
}
```

Для сортировки он использует ядро с функцией, которую я описал на device-е.

```
__device__ void swap_step(int* nums, int* tmp, int size, int start, int stop, int step, int i) {
        // Using shared memory to store blocks and sort them
__shared__ int sh_array[BLOCK_SIZE];
// Step for bitonic merge inside merging
for (int shift = start; shift < stop; shift += step) {
// New start pointer
tmp = nums + shift;
```

```
// Right side
if (i >= BLOCK_SIZE / 2)
sh_array[i] = tmp[BLOCK_SIZE * 3 / 2 - 1 - i];
else
sh_array[i] = tmp[i];
__syncthreads();
// From half
for (int j = BLOCK_SIZE / 2; j > 0; j /= 2) {
unsigned int XOR = i ^ j;
// The threads with the lowest ids sort the array
if (XOR > i) {
if ((i & BLOCK_SIZE) != 0) {
// Step descending, swap(i, XOR)
if (sh_array[i] < sh_array[XOR])
thrust::swap(sh_array[i], sh_array[XOR]);
} else {
// Step ascending, swap(i, XOR)
if (sh_array[i] > sh_array[XOR])
thrust::swap(sh_array[i], sh_array[XOR]);
}
}
__syncthreads();
}
// Back from shared to temporary
tmp[i] = sh_array[i];
}
}
```

Эта функция вызывается внутри ядра и используется в качестве итераций чет-нечет сортировки. В данном цикле вызывается ядро, которое вызывает функцию, описанную выше.

```
for (int i = 0; i < 2 * (upd_size / BLOCK_SIZE); ++i) {
kernel_bitonic_merge_step<<<NUM_BLOCKS, BLOCK_SIZE>>>(dev_data, upd_size, (bool)(i % 2), true);
CUDA_ERROR(cudaGetLastError());
```

}

Ядро для основной сортировки выглядит аналогично функции на device-е.

Результаты

Для того, чтобы проанализировать работу алгоритма на больших данных я воспользовался профилировщиком nvprof. В сводном режиме по умолчанию nvprof представляет обзор ядер графического процессора и копий памяти в исполняемом файле. Сводка группирует все вызовы одного и того же ядра вместе, показывая общее время и процент от общего времени приложения для каждого ядра. В дополнение к сводному режиму nvprof поддерживает режимы GPU-Trace и API-Trace, которые позволяют видеть полный список всех запусков ядра и копий памяти, а в случае режима API-Trace - все вызовы API CUDA.

Ниже я привел пример профилирования примера приложения с использованием nvprof --print-gpu-trace. Тут можно заметить, на каком графическом процессоре работало каждое ядро, а также размеры сетки, используемые для каждого запуска. Это оказывается очень полезно, если нужно убедиться, что приложение с несколькими графическими процессорами работает должным образом.

```
==12634== Profiling application: ./lab5 --benchmark -numdevices=2 -i=1
==12634== Profiling result:
   Start  Duration            Grid Size      Block Size     Regs*    SSMem*    DSMem*      Size  Throughput           Device   Context    Stream  Name
337.03ms  1.2800us                    -               -         -         -         -  4.0000KB  2.9802GB/s  GeForce GT 545         1         7  [CUDA memcpy HtoD]
338.45ms  5.6580us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [97]
338.47ms  5.5790us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [104]
338.48ms  5.3710us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [111]
338.49ms  5.8590us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [118]
338.50ms  5.8770us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [125]
338.51ms  5.8360us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [132]
338.52ms  6.6940us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [139]
338.53ms  6.2190us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [146]
338.55ms  6.2640us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [153]
338.56ms  6.2580us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [160]
338.57ms  7.0730us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [167]
338.58ms  6.4190us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [174]
338.59ms  6.4220us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [181]
338.60ms  6.4560us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [188]
338.61ms  6.4250us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [195]
338.63ms  7.3080us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [202]
338.64ms  7.1360us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [209]
338.65ms  7.1770us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [216]
338.66ms  7.1530us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [223]
338.67ms  7.2040us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [230]
338.68ms  7.1560us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [237]
338.69ms  7.9690us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [244]
338.70ms  7.8120us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [251]
338.72ms  7.7880us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [258]
338.73ms  7.7500us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [265]
338.74ms  7.7510us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [272]
338.75ms  7.7300us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [279]
338.76ms  7.7840us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [286]
338.77ms  8.7900us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [293]
338.78ms  8.3940us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [300]
338.79ms  8.3420us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [307]
338.81ms  8.2920us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [314]
338.82ms  8.3580us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [321]
338.83ms  8.1350us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [328]
338.84ms  8.1570us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [335]
338.86ms  8.1300us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [342]
338.87ms  9.5950us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [349]
338.88ms  8.8630us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [356]
338.89ms  8.8480us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [363]
338.90ms  8.8440us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [370]
338.91ms  8.8640us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [377]
338.92ms  8.8520us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [384]
338.93ms  8.8690us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [391]
```

```
338.98ms  9.7970us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [419]
338.99ms  9.8370us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [426]
339.00ms  9.8240us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [433]
339.01ms  9.8400us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [440]
339.02ms  9.8240us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [447]
339.03ms  9.8260us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [454]
339.04ms  9.8270us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [461]
339.06ms  9.8220us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [468]
339.07ms  9.7990us             (10 1 1)       (1024 1 1)        16  4.0000KB        0B         -           -  GeForce GT 545         1         7  bitonic_sort_step(int*, int, int, int) [475]
339.08ms  3.6160us                    -               -         -         -         -  4.0000KB  1.0550GB/s  GeForce GT 545         1         7  [CUDA memcpy DtoH]
340.50ms  9.6750us             (10 1 1)       (1024 1 1)        19  4.0000KB        0B         -           -  GeForce GT 545         1         7  kernel_bitonic_merge_step(int*, int, bool, bool) [483]
340.51ms  5.7940us             (10 1 1)       (1024 1 1)        19  4.0000KB        0B         -           -  GeForce GT 545         1         7  kernel_bitonic_merge_step(int*, int, bool, bool) [490]
340.52ms  3.6160us                    -               -         -         -         -  4.0000KB  1.0550GB/s  GeForce GT 545         1         7  [CUDA memcpy DtoH]

Regs: Number of registers used per CUDA thread. This number includes registers used internally by the CUDA driver and/or tools and can be more than what the compiler shows.
SSMem: Static shared memory allocated per CUDA block.
DSMem: Dynamic shared memory allocated per CUDA block.
```

Выводы

В этой работе я познакомился и научился реализовывать сортировку чет-нечет с помощи классических алгоритмов в области параллельной обработки данных. Было интересно узнать про еще один вариант сортировки чисел и реализовать его с помощью CUDA. Данный способ значительно ускоряет работу простой битонической сортировки из-за того, что блоки сортируются параллельно, не мешая друг другу.

Также я познакомился с профилировщиком nvprof. Это довольно полезный и удобный инструмент для профилирования кода заточенного под GPU.

Во время выполнения возникали проблемы с понимаем того, в какой момент и что нужно реализовать. По сути, используя комбинацию из 2 битонических сортировок, я получил сортировку чет-нечет, однако было не просто к этому прийти. Также возникала уже известная ошибка an illegal memory access was encountered, которая была связана с тем, что я пытался выделить больше памяти для размера блока, хотя на текущем кластере размер составляет 1024, как только я ограничил размер, ошибка исчезла.