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*. Свёртка, сканирование, гистограмма

Выполнил: Днепров И. С.
Группа: 8О-407Б-17
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

Цель работы:

Ознакомление с фундаментальными алгоритмами на *GPU*: свёртка (*reduce*), сканирование (*blelloch scan*) и гистограмма (*histogramm*). Реализация одной из сортировок на *CUDA*. Использование разделяемой и других видов памяти. Исследование производительности программы с помощью утилиты *nvprof*.

Вариант 1. **Битоническая сортировка**

Требуется реализовать битоническую сортировку для чисел типа *int*.

Должна быть реализована адаптивная операция битонического слияния. Если данные помещаются в разделяемую память, то взаимодействие идёт через неё, если нет, то через глобальную память (т.е. необходимо реализовать несколько вариантов ядра).

Ограничения: $n \leq 256 \cdot 10^6$.

Все входные-выходные данные являются бинарными и считываются из **stdin** и выводятся в **stdout**.

Входные данные.

В первых четырех байтах записывается целое число n – длина массива чисел, далее следуют n чисел типа заданного вариантом.

Выходные данные.

В бинарном виде записывают n отсортированных по возрастанию чисел.

Пример входных-выходных данных. Десять чисел типа *int*, от 0 до 9.

Входной файл (*stdin*), *hex*:

```
0A000000 00000000 09000000 08000000 07000000 06000000 05000000
04000000 03000000 02000000 01000000
```

Выходной файл (*stdout*), *hex*:

```
00000000 01000000 02000000 03000000 04000000 05000000 06000000
07000000 08000000 09000000
```

Программное и аппаратное обеспечение

GPU

Название: GeForce GT 545
Размер глобальной памяти: 3150381056
Размер константной памяти : 65536
Размер разделяемой памяти: 49152
Регистров на блок: 32768
Максимум потоков на блок: 1024
Размер варпа: 32
Максимальные размеры блока: 1024 x 1024 x 64
Максимальные размеры сетки: 65535 x 65535 x 65535
Количество мультипроцессоров : 3

CPU

Название: Intel Core i7-3770

Частота: 3.40GHz

Размер кеша: 8192 KB

Количество ядер: 4

Количество потоков: 8

MEM

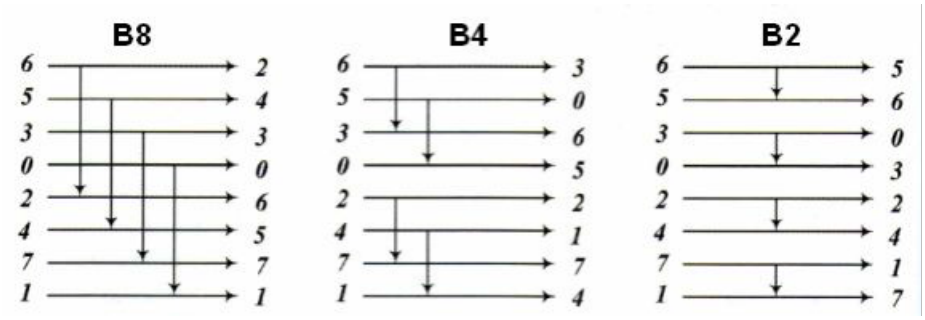
Размер: 15 GB

Тип: ddr3

Прочее

OS: Linux Ubuntu 16.04.6
Редактор: Atom

Метод решения

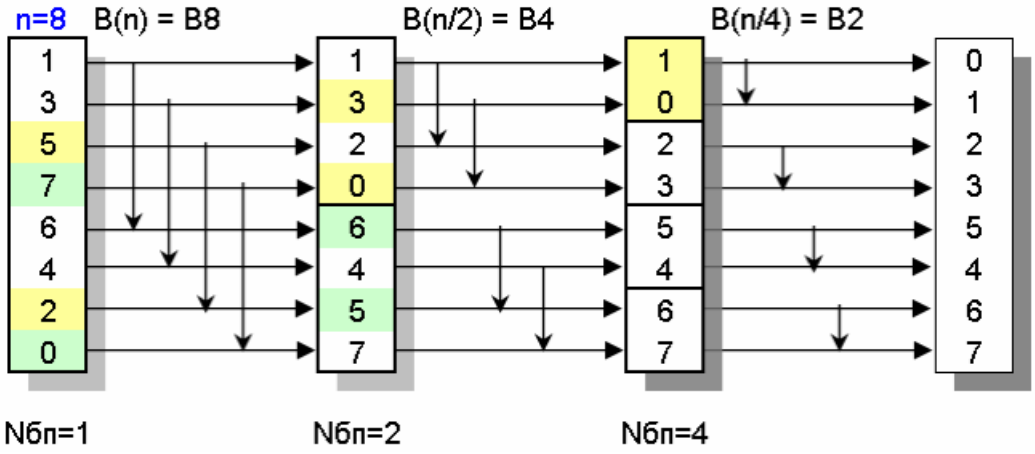


В основе метода – полуочиститель $B(n)$, упорядочивающий $X(i)$ и $X(i+n/2)$.

Битоническая последовательность:

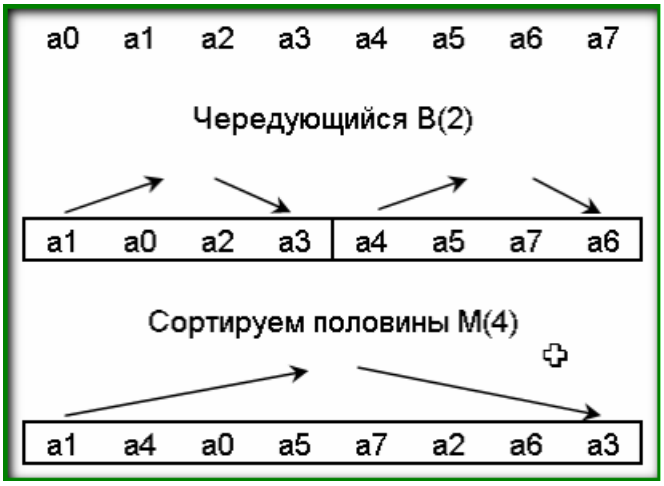
5 7 6 4 2 1 3

1 3 5 7 6 4 2



- Для битонической последовательности применим $B(n), B(n/2), \ldots, B(2)$.
- $M(n): B(n), B(n/2), \ldots, B(2)$ – битоническое слияние.
- $M(n)$ – сортирует произвольную битоническую последовательность.

Произвольная последовательность:



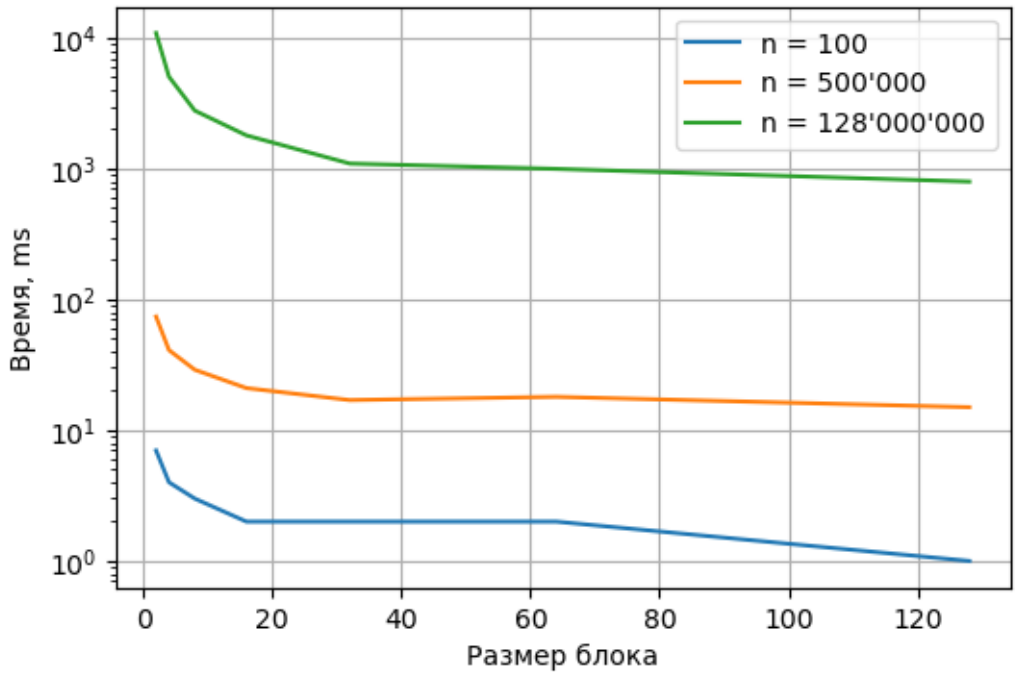
Последовательность разбивается на две части, каждая из частей сортируется. Все шаги – применение полуочистителя.

Описание программы

Прототип	Тип	Описание
`__global__ void bitonic_merge(int *host_data, int n, int rank, int dist)`	Ядро	Применить битоническое слияние M_{rank} с использованием разделяемой памяти. Последовательность элементов длины *dist* сортируется в одном из направлений
`__global__ void bitonic_half_cleaner(int *host_data, int n, int block_index, int dist)`	Ядро	Применить битонический полуочиститель B_{iB} с использованием глобальной памяти. Последовательность элементов длины *dist* сортируется в одном из направлений
`int main()`	Функция	Главная точка входа в приложение

Результаты

№ теста	Количество чисел	Размер блока	Число блоков	Время, микросекунды
1	100	2	512	7
2		4		4
3		8		3
4		16		2
5		32		2
6		64		2
7		128		1
8	500’000	2		74
9		4		41
10		8		29
11		16		21
12		32		17
13		64		18
14		128		15
15	128’000’000	2		10921
16		4		5056
17		8		2782
18		16		1798
19		32		1095
20		64		997
21		128		797



Профилирование на удалённом сервере.

```
~/PGP_5$ nvprof -e l1_shared_bank_conflict -e divergent_branch ./a.out
==18579== NVPROF is profiling process 18579, command: ./a.out
Time to sort 25600000 numbers = 5585.00
BLOCK_SIZE = 1024, blocks = 512, threads = 1024
==18579== Warning: The following aggregate event values were extrapolated from limited profile data and may therefore be inaccurate. To see the non-agg
regate event values, use "--aggregate-mode off".
l1_shared_bank_conflict
==18579== Profiling application: ./a.out
==18579== Profiling result:
==18579== Event result:
Invocations                                Event Name         Min         Max         Avg
Device "GeForce GT 545 (0)"
        Kernel: B_apply_global(int*, int, int, int)
        120                    l1_shared_bank_conflict           0           0           0
        120                           divergent_branch           1       12158        1573
        Kernel: M_apply_shared(int*, int, int, int)
         25                    l1_shared_bank_conflict           0           0           0
         25                           divergent_branch     2639291    10586808     5260469
        Kernel: vector_set(int*, int, int)
          1                    l1_shared_bank_conflict           0           0           0
          1                           divergent_branch           0           0           0
```

Выводы

Получен навык распараллеливания вычислений в алгоритме поразрядной сортировки с помощью *CUDA*, изучены вопросы работы с разделяемой памятью. Был реализован параллельный вариант алгоритма битонической сортировки.

Битонная сортировка является одним из первых параллельных алгоритмов сортировки. Публикация этого алгоритма, наряду с также предложенным Бэтчером алгоритмом четно-нечетной сортировки слиянием, стимулировала развитие проектирования и анализа параллельных алгоритмов в общем и параллельной сортировки в частности.

Благодаря высокой параллельности, битонные сортировщики широко применяются в устройствах, нацеленных на массивные параллельные вычисления, таких как графические процессоры, но редко используется в современных суперкомпьютерах.

Во время выполнения работы возникли сложностей не возникло.

В ходе лабораторной выяснилось, что технология *CUDA* хорошо подходит для решения параллельных задачах.