МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Кафедра вычислительной математики и программирования

спецкурс «Параллельные и распределенные вычисления»

ОТЧЕТ

Лабораторная работа № 1

Выполнил: Полей-Добронравова Амелия
Группа: М8О-114М-22
Преподаватель: Семенов С. А.

Москва, 2022

Содержание

1. Постановка задачи

На вход программа принимает длину последовательности, начальный элемент и шаг арифметической прогрессии. В функции-ядре gpu заполнить массив элементами данной арифметической последовательности. Со стороны cpu вывести сформированный массив на экран.

2. Описание решения

В main на cpu я выделяю память для массива: один массив - память cpu для вывода результата, второй массив - память gpu для обработки потоками.

Функция-ядро gpu в зависимости от индекса рабочего потока выбирает себе элементы массива для заполнения. Шаг выбора элемента массива offset высчитывается в зависимости от числа потоков в блоке.

3. Аппаратное обеспечение и ПО

Компилятор nvcc версии 7.0(g++ версии 4.8.4) на 64-х битной Ubuntu 14.04 LTS.
Параметры графического процессора:
Compute capability : 6.1
Name : GeForce GTX 1050
Total Global Memory : 2096103424
Shared memory per block : 49152
Registers per block : 65536
Max threads per block : (1024, 1024, 64)
Max block : (2147483647, 65535, 65535)
Total constant memory : 65536
Multiprocessors count : 5

4. Основные моменты кода

```
__global__ void kernel(int* v1, long long n, int a0, int d) {
    long long i = blockDim.x * blockIdx.x + threadIdx.x;
    long long offset = blockDim.x * gridDim.x;

    while (i < n) {
        v1[i] = a0 + d * i;
        i = i + offset;
    }
    v1[0] = a0;
}

int main() {
    long long n;
    scanf("%lld", &n);

    int a0;
    scanf("%d", &a0);

    int d;
    scanf("%d", &d);

    int* v1 = (int*)malloc(n * sizeof(int)); //элементы прогрессии

    int* dev_v1;
    CSC(cudaMalloc(&dev_v1, sizeof(int) * n));
    cudaEvent_t start, end;
    CSC(cudaEventCreate(&start));
    CSC(cudaEventCreate(&end));
    CSC(cudaEventRecord(start));

    kernel<<<256,256>>>(dev_v1, n, a0, d);

    CSC(cudaEventRecord(end));
    CSC(cudaEventSynchronize(end));
    float t;
    CSC(cudaEventElapsedTime(&t, start, end));
    CSC(cudaEventDestroy(start));
    CSC(cudaEventDestroy(end));
    printf("time: %.10lf\n", t);

    CSC(cudaMemcpy(v1, dev_v1, sizeof(int) * n, cudaMemcpyDeviceToHost));
    CSC(cudaFree(dev_v1));
```

5. Результат работы программы

```
(base) radmin@radmin-A7-X1:~/Documents$ ./lab1
7
20
-10
time: 0.0051199999
20 10 0 -10 -20 -30 -40
```

6. Сравнение скорости выполнения на CPU и GPU

При запуске программы с различными значениями *N* видно, что вычисления на видеокарте произвелись быстрее, чем на процессоре компьютера, при *N*>32.

Время выполнения программы при различных значениях *N*:

N	GPU время выполнения, мс	CPU время выполнения, мс	t_{CPU}/t_{GPU}
4	0.000061440002	0.001	1,6
8	0.000230080001	0.001	5
16	0.000229759999	0.001	5
32	0.000147839999	0.001	7
64	0.000140800001	0.001	7
128	0.000082879998	0.001	2
256	0.000133440001	0.001	2

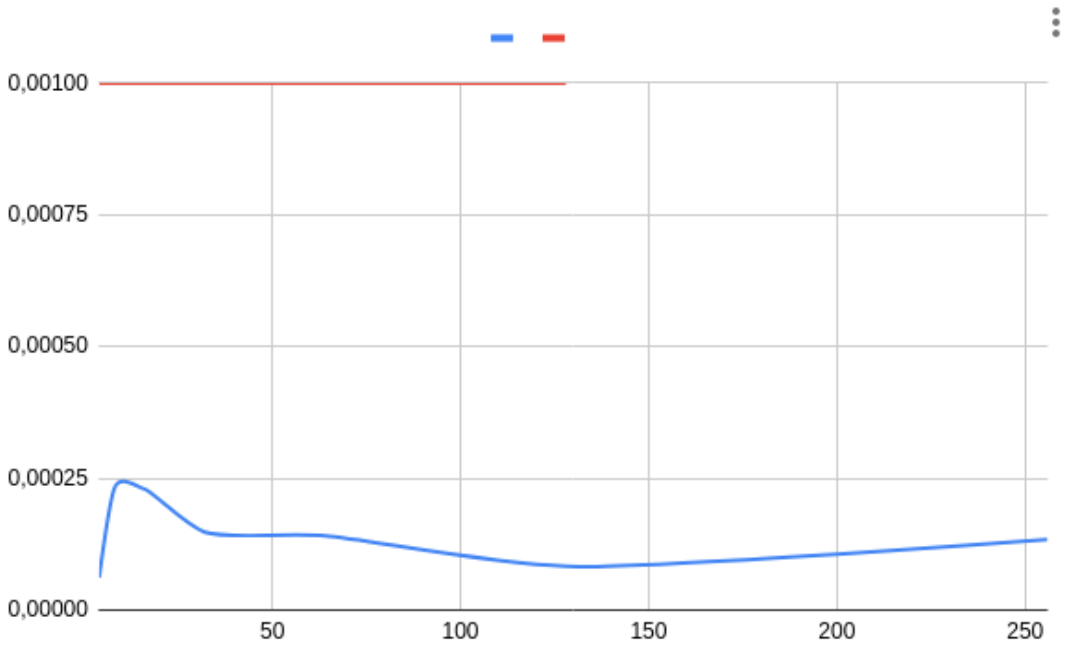

Рис. График зависимости времени выполнения программы от длины арифметической последовательности *N*.

7. Выводы

В Лабораторной работе №1 проведен анализ работы различных программ по решению задачи вычисления арифметической прогрессии используя технологию cuda.

Во второй части Лабораторной работы происходило сравнение времени выполнения на gpu и cpu.

8. Приложения