Министерство цифрового развития, связи
и массовых коммуникаций Российской Федерации

Сибирский государственный университет
телекоммуникаций и информатики

Кафедра прикладной математики и кибернетики

ЛАБОРАТОРНАЯ РАБОТА №3

По дисциплине: «Программирование графических процессоров»

Выполнили:

Студенты 3 курса группы ИП-111
Корнилов А.А.,
Попов М.И.,
Толкач А.А.

Проверил:

Профессор кафедры ПМиК
Малков Е.А.

Новосибирск, 2024

Задание: 1. Напишите программу сложения двух векторов на GPU.

2.Зафиксируйте длину векторов равной 1<<20.

3. Проведите серию запусков программы варьируя количество нитей в блоке, 1, 16, 32, 64,128, …, 1024, для запуска ядра.

3. Определите зависимость времени выполнения ядра, вычисляющего сумму векторов, от конфигурации нитей. Используйте для определения времени события CUDA и профилировщики.

Цель: априорное понимание влияния конфигурации нитей на производительность выполнения кода на GPU.

Выполнение работы:

Для первого задания была написана простоя программа для сложения двух векторов используя GPU, количество векторов задано n = 1<<20, для определения времени работы используется CudaEvent и STL библиотека Chrono

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include <stdio.h>
#include <iostream>
#include <chrono>
using namespace std;
const int n = 1<<20;
typedef std::chrono::milliseconds ms;
typedef std::chrono::nanoseconds ns;

__global__ void vectorAdd(const float* a, const float* b, float* c, int n) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < n) {
        c[i] = a[i] + b[i];
    }
}

int main() {
    float elapsedTime;
    int blockSize = 1024;
    int numBlocks = n;
    cudaEvent_t start, stop;
    chrono::time_point<chrono::system_clock> start_chrono, end_chrono;

    cout << "Enter threads num: ";
    cin >> numBlocks;

    float* d_a, * d_b, * d_c;
    cudaMalloc((void**)&d_a, n * sizeof(float));
    cudaMalloc((void**)&d_b, n * sizeof(float));
    cudaMalloc((void**)&d_c, n * sizeof(float));

    float* h_a = new float[n];
    float* h_b = new float[n];
    for (int i = 0; i < n; ++i) {
        h_a[i] = i;
```

```
        h_b[i] = i * 2;
    }

    cudaMemcpy(d_a, h_a, n * sizeof(float), cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, h_b, n * sizeof(float), cudaMemcpyHostToDevice);

    cudaEventCreate(&start);
    cudaEventCreate(&stop);

    cudaEventRecord(start, 0);
    start_chrono = chrono::system_clock::now();
    vectorAdd <<< numBlocks, blockSize >>> (d_a, d_b, d_c, n);
    cudaEventRecord(stop, 0);
    end_chrono = chrono::system_clock::now();

    cudaEventSynchronize(stop);
    cudaEventElapsedTime(&elapsedTime, start, stop);

    float* h_c = new float[n];
    cudaMemcpy(h_c, d_c, n * sizeof(float), cudaMemcpyDeviceToHost);

    cout << "CUDA Event time: " << elapsedTime << endl
         << "Chrono time: " << chrono::duration_cast<ms>(end_chrono -
start_chrono).count() << "ms"
         << endl << chrono::duration_cast<ns>(end_chrono - start_chrono).count() << "ns";

    delete[] h_a;
    delete[] h_b;
    delete[] h_c;
    cudaFree(d_a);
    cudaFree(d_b);
    cudaFree(d_c);

    return 0;
}
```

Листинг 1 – программа LR03_1.cu

Команда компиляции и результат работы программы, количество выбранных нитей выбирается с клавиатуры:

```
Enter threads num: 16
CUDA Event time: 0.813696
Chrono time: 0ms
373300ns
```

Ниже приведена таблица (таб. 1) с запуском программы с количеством нитей от 1 до 1024. Также графики с использование CudaEvent (грф. 1) и таймера Chrono (грф. 2). По графикам можно сделать вывод что время выполнения уменьшается относительно увеличения количества нитей

Нити	EventTime	Chrono
1	0,116544	133800
16	0,068672	114200
32	0,019168	65100
64	0,035424	80400
128	0,012608	67500
256	0,026752	70600
512	0,028672	74100
1024	0,023776	72900

Таблица 1 – Запуск программы с различным количеством нитей

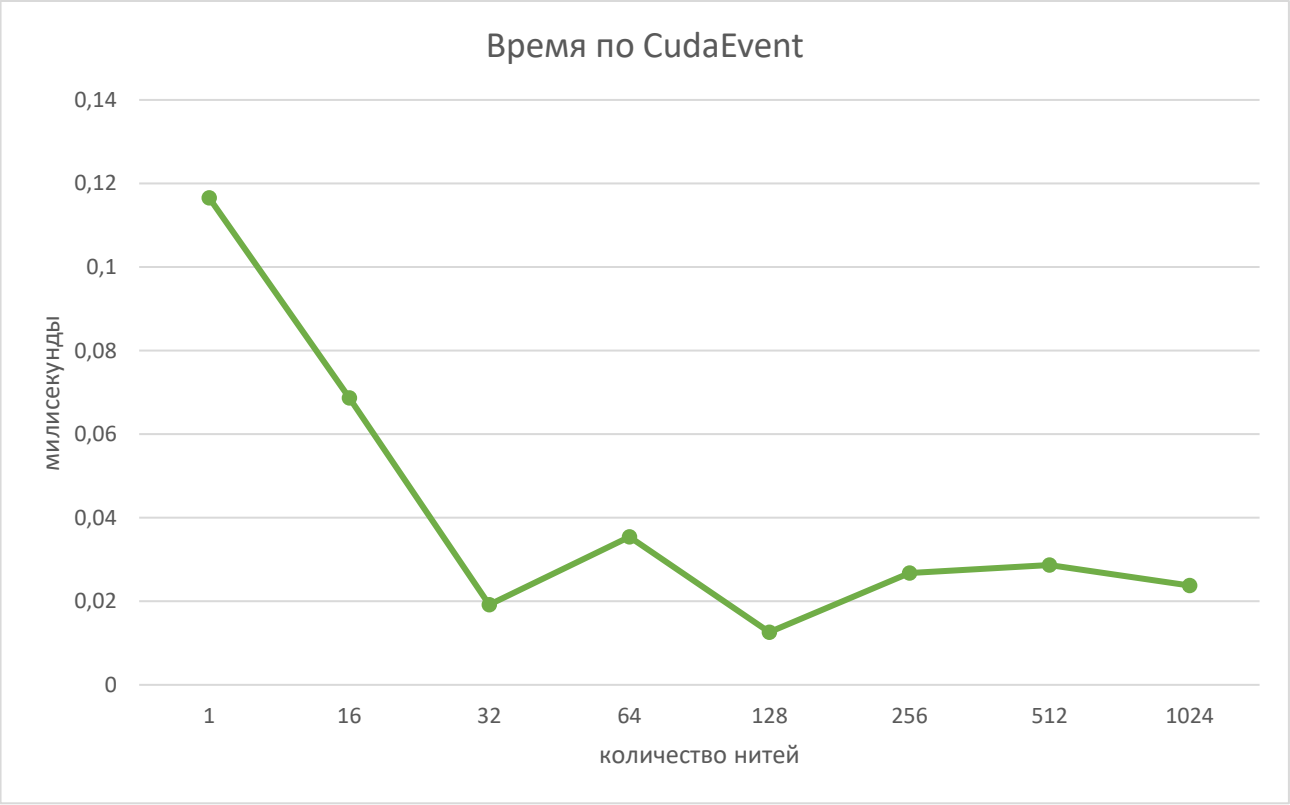

График 1 – время выполнения от кол-во нитей по таймеру CudaEvent

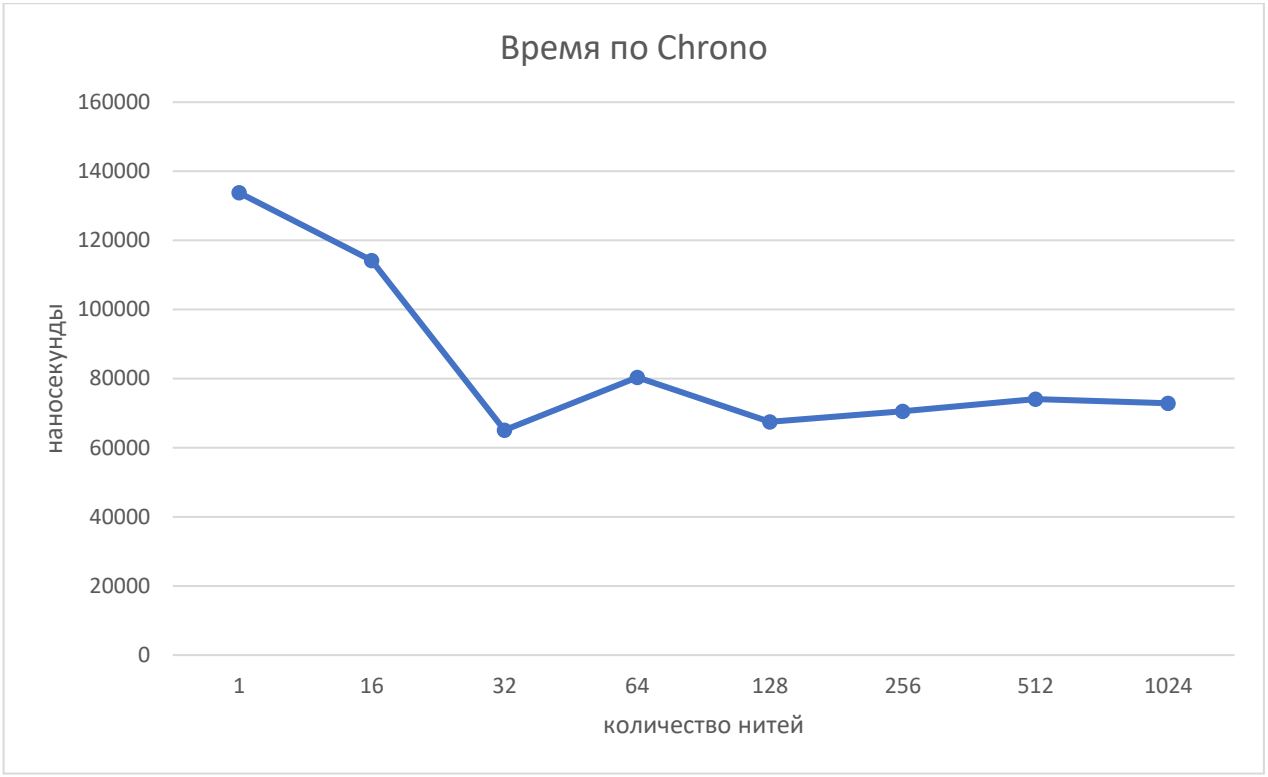

График 2 - время выполнения от кол-во нитей по таймеру STL::Chrono