Кафедра прикладной математики и кибернетики

Выполнили:

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

Толкач А.А.

Проверил:

Профессор кафедры ПМиК  
Малков Е.А.

ЛАБОРАТОРНАЯ РАБОТА №3

По дисциплине: «Программирование графических процессоров»

Сибирский государственный университет  
телекоммуникаций и информатики

Новосибирск, 2024

Министерство цифрового развития, связи  
и массовых коммуникаций Российской Федерации

**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