МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ

Кафедра ВМСС

Лабораторная работа №4

По курсу “Распределенные вычисления”

по теме

“Программирование графических процессоров средствами nVidia CUDA (Compute Unified Device Architecture). Использование графических процессоров в качестве сопроцессора для ускорения вычислений”

|  |  |
| --- | --- |
| **Выполнил:** | Анисимов А. Д.  Мишин А. А. |
| **Группа:** | А-08м-19 |
| **Бригада:** | №1 |

Москва, 2019 г.

**Цель работы:**

Создание программы, выполняющей заданные вычисления на GPU с использованием средств nVidia CUDA.  
Варианты задания:

|  |  |
| --- | --- |
| № | **Формула** |
| 1 | Y[i] = A[i]\*B[i] + A[i]\* A[i]\* A[i]\*C[i]/B[i] |

1. Параметры вычислительной системы:

CPU: Intel Core i5-8500, 3.90 ГГц

6 физических ядер, 6 логических ядер.

RAM: 16 Gb DDR4

GPU: nVidia GeForce GTX 970

* Версия CUDA: 5.2
* Ядер CUDA: 1664
* VRAM: 4095 MB
* Max Block Size: 1024\*1024\*64
* Max Warp Size: 32
* Max Grid Size: 2147483647\*65535\*65535

Операционная система Win10 Pro 1903

Следует упомянуть: в проекте одна программа содержит все варианты построение параллелизма согласно заданию, однако вызываются как подпрограммы после выбора пользователя. Код главного (управляющего) модуля программы приведён ниже:

int main(){

int count[3] = { pow(2,14),pow(2,16) };

int blocks[5] = {1,2,4,8,16};

int blocksize[6] = {1,4,32,64,256,512};

for (int i = 0; i < 2; i++)

{

CPU(count[i]);

}

for (int i = 0; i < 2; i++)

for (int j = 0; j < 5; j++)

for (int k = 0; k < 6; k++)

GPU(count[i], blocks[j], blocksize[k]);

system("Pause");

return 0;

}

2. Далее представлена подпрограмма для вычисления на CPU:

int CPU(int N)

{

float\* vectA = new float[N];

float\* vectB = new float[N];

float\* vectC = new float[N];

float\* vectY = new float[N];

float checksumm = 0;

//заполняем массивы данными

for (int i = 0; i < N; i++) {

vectA[i] = static\_cast<float>(i \* 1.1);

vectB[i] = static\_cast<float>(i \* 2.7) + 1;

vectC[i] = static\_cast<float>(i \* 3.14);

}

auto startc = std::chrono::high\_resolution\_clock::now();

//Вычисления

for (int j = 0; j < cyclelength; j++)

for (int i = 0; i < N; i++)

vectY[i] = ((vectA[i] \* vectB[i]) + ((vectA[i] \* vectA[i] \* vectA[i]) \* vectC[i] / vectB[i]));

auto finishc = std::chrono::high\_resolution\_clock::now();

auto duration = std::chrono::duration\_cast<std::chrono::microseconds>(finishc - startc).count();

cout << "Chrono: N = " << N << "; CPU Time = " << duration / 1000.0 << " ms\n";

for (int i = 0; i < N; i++) {

checksumm += vectY[i] / checksize;

//cout << "Result[" << i << "]:=" << vectY[i] << endl;

}

printf("checksumm = %f\n\n\n", checksumm);

return 0;

}3. Далее представлена подпрограмма, организующая параллелизм с использованием GPU:

\_\_global\_\_ void addKernel(float\* vectA, float\* vectB, float\* vectC, float\* vectY, int S)

{

int k;

//номер блока в задаче \* размер блока + номер потока в текущем блоке

int idx\_thread = blockIdx.x \* blockDim.x + threadIdx.x;

int m = 0;

for (int j = 0; j < cyclelength; j++)

for (k = 0; k < S; k++)

{

m = idx\_thread \* S + k;

vectY[m] = ((vectA[m] \* vectB[m]) + ((vectA[m] \* vectA[m] \* vectA[m]) \* vectC[m] / vectB[m]));

}

}

int GPU(int N, int blocks, int blocksize)

{

float\* vectA = new float[N];

float\* vectB = new float[N];

float\* vectC = new float[N];

float\* vectY = new float[N];

float\* devA;

float\* devB; //указатели на память для ГПУ

float\* devC;

float\* devY; //указатели на память для ГПУ

float elapsedTime; //для подсчета времени средствами CUDA

//float checksumm = 0;

cudaEvent\_t start, stop; //идентификаторы событий

cudaEventCreate(&start); //инициализация события start

cudaEventCreate(&stop); //инициализация события stop

float m = 3;

float checksumm = 0;

//заполняем массивы данными

for (int i = 0; i < N; i++) {

vectA[i] = static\_cast<float>(i \* 1.1);

vectB[i] = static\_cast<float>(i \* 2.7) + 1;

vectC[i] = static\_cast<float>(i \* 3.14);

}

////выделение памяти на ГПУ для массивов

cudaMalloc((void\*\*)&devA, N \* sizeof(float));

cudaMalloc((void\*\*)&devB, N \* sizeof(float));

cudaMalloc((void\*\*)&devC, N \* sizeof(float));

cudaMalloc((void\*\*)&devY, N \* sizeof(float));

//blocks = 16; //количество блоков потоков

//blocksize = 512; //потоков на один блок

int steps = static\_cast<int>(N / (blocks \* blocksize)); //количество элементов векторов, которые обрабатывает каждый поток

//фиксируем время начала выполнения

cudaEventRecord(start, 0);

auto startc = std::chrono::high\_resolution\_clock::now();

//копируем вектор 1 на место devA, размером таким-то, и с флагом что копируем с хоста -> ГПУ

cudaMemcpy(devA, vectA, N \* sizeof(float), cudaMemcpyHostToDevice);

cudaMemcpy(devB, vectB, N \* sizeof(float), cudaMemcpyHostToDevice);

cudaMemcpy(devC, vectC, N \* sizeof(float), cudaMemcpyHostToDevice);

cudaMemcpy(devY, vectY, N \* sizeof(float), cudaMemcpyHostToDevice);

//вычисляем на ГПУ

addKernel <<<blocks, blocksize>>> (devA, devB, devC, devY, steps);

//копируем себе назад результат операции

cudaMemcpy(vectY, devY, N \* sizeof(float), cudaMemcpyDeviceToHost);

cudaFree(devA);

cudaFree(devB);

cudaFree(devC);

cudaFree(devY);

cudaEventRecord(stop, 0);

cudaEventSynchronize(stop); //синхронизация host и device по событию stop

cudaEventElapsedTime(&elapsedTime, start, stop); //время = stop - start

auto finishc = std::chrono::high\_resolution\_clock::now();

auto duration = std::chrono::duration\_cast<std::chrono::microseconds>(finishc - startc).count();

//cout << "Chrono: N = " << N << ";"<<" Blocks = "<<blocks<<"; Blocksize = "<<blocksize<<"; GPU Time = " << duration / 1000.0 << " ms\n";

cout << duration / 1000.0 << endl;

//printf("CUDA: GPU Time = %f\n", elapsedTime);

for (int i = 0; i < N; i++) {

checksumm += vectY[i] / checksize;

//cout << "Result[" << i << "]:=" << vectY[i] << endl;

}

//printf("checksumm = %f\n\n\n", checksumm);

//printf("vect1 = %5.f\n", vect1[1023]);

//tracing tools such as Nsight and Visual Profiler to show complete traces.

cudaDeviceReset();

return 0;

}

7. Далее представлены времена выполнения и коэффициенты ускорения.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| GPU |  |  |  |  |  |  |  |
| Размер | Количество блоков | Размер Блока | Время (мс) | Размер | Количество блоков | Размер Блока | Время (мс) |
| 16384 | 1 | 1 | 480,53 | 65536 | 1 | 1 | 1667,80 |
|  | 1 | 4 | 108,95 |  | 1 | 4 | 433,70 |
|  | 1 | 32 | 19,07 |  | 1 | 32 | 73,18 |
|  | 1 | 64 | 9,78 |  | 1 | 64 | 38,09 |
|  | 1 | 256 | 4,55 |  | 1 | 256 | 17,76 |
|  | 1 | 512 | 4,82 |  | 1 | 512 | 18,06 |
|  | 2 | 1 | 209,52 |  | 2 | 1 | 835,12 |
|  | 2 | 4 | 54,78 |  | 2 | 4 | 217,33 |
|  | 2 | 32 | 9,82 |  | 2 | 32 | 37,43 |
|  | 2 | 64 | 5,12 |  | 2 | 64 | 19,20 |
|  | 2 | 256 | 2,68 |  | 2 | 256 | 9,53 |
|  | 2 | 512 | 1,78 |  | 2 | 512 | 9,11 |
|  | 4 | 1 | 104,90 |  | 4 | 1 | 417,92 |
|  | 4 | 4 | 27,59 |  | 4 | 4 | 109,10 |
|  | 4 | 32 | 5,13 |  | 4 | 32 | 19,25 |
|  | 4 | 64 | 3,04 |  | 4 | 64 | 10,06 |
|  | 4 | 256 | 1,37 |  | 4 | 256 | 5,17 |
|  | 4 | 512 | 1,15 |  | 4 | 512 | 4,99 |
|  | 8 | 1 | 52,91 |  | 8 | 1 | 209,65 |
|  | 8 | 4 | 14,26 |  | 8 | 4 | 55,15 |
|  | 8 | 32 | 2,93 |  | 8 | 32 | 10,00 |
|  | 8 | 64 | 1,82 |  | 8 | 64 | 5,55 |
|  | 8 | 256 | 1,07 |  | 8 | 256 | 3,10 |
|  | 8 | 512 | 0,93 |  | 8 | 512 | 2,09 |
|  | 16 | 1 | 26,85 |  | 16 | 1 | 105,42 |
|  | 16 | 4 | 7,44 |  | 16 | 4 | 28,13 |
|  | 16 | 32 | 1,89 |  | 16 | 32 | 5,63 |
|  | 16 | 64 | 1,30 |  | 16 | 64 | 3,48 |
|  | 16 | 256 | 0,83 |  | 16 | 256 | 2,55 |
|  | 16 | 512 | 0,80 |  | 16 | 512 | 2,46 |

|  |  |  |
| --- | --- | --- |
| CPU | Размер | Время (мс) |
|  | 16384 | 11,938 |
|  | 65536 | 47,746 |

**КОЭФФИЦИЕНТЫ УСКОРЕНИЯ:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер | Количество блоков | Размер Блока | Коэфф. Ускорения |  | Размер | Количество блоков | Размер Блока | Коэфф. Ускорения |
| 16384 | 1 | 1 | 0,02 |  | 65536 | 1 | 1 | 0,03 |
|  | 1 | 4 | 0,11 |  |  | 1 | 4 | 0,11 |
|  | 1 | 32 | 0,63 |  |  | 1 | 32 | 0,65 |
|  | 1 | 64 | 1,22 |  |  | 1 | 64 | 1,25 |
|  | 1 | 256 | 2,62 |  |  | 1 | 256 | 2,69 |
|  | 1 | 512 | 2,48 |  |  | 1 | 512 | 2,64 |
|  | 2 | 1 | 0,06 |  |  | 2 | 1 | 0,06 |
|  | 2 | 4 | 0,22 |  |  | 2 | 4 | 0,22 |
|  | 2 | 32 | 1,22 |  |  | 2 | 32 | 1,28 |
|  | 2 | 64 | 2,33 |  |  | 2 | 64 | 2,49 |
|  | 2 | 256 | 4,46 |  |  | 2 | 256 | 5,01 |
|  | 2 | 512 | 6,71 |  |  | 2 | 512 | 5,24 |
|  | 4 | 1 | 0,11 |  |  | 4 | 1 | 0,11 |
|  | 4 | 4 | 0,43 |  |  | 4 | 4 | 0,44 |
|  | 4 | 32 | 2,33 |  |  | 4 | 32 | 2,48 |
|  | 4 | 64 | 3,93 |  |  | 4 | 64 | 4,75 |
|  | 4 | 256 | 8,73 |  |  | 4 | 256 | 9,24 |
|  | 4 | 512 | 10,41 |  |  | 4 | 512 | 9,57 |
|  | 8 | 1 | 0,23 |  |  | 8 | 1 | 0,23 |
|  | 8 | 4 | 0,84 |  |  | 8 | 4 | 0,87 |
|  | 8 | 32 | 4,07 |  |  | 8 | 32 | 4,78 |
|  | 8 | 64 | 6,57 |  |  | 8 | 64 | 8,61 |
|  | 8 | 256 | 11,17 |  |  | 8 | 256 | 15,39 |
|  | 8 | 512 | 12,82 |  |  | 8 | 512 | 22,86 |
|  | 16 | 1 | 0,44 |  |  | 16 | 1 | 0,45 |
|  | 16 | 4 | 1,60 |  |  | 16 | 4 | 1,70 |
|  | 16 | 32 | 6,32 |  |  | 16 | 32 | 8,48 |
|  | 16 | 64 | 9,20 |  |  | 16 | 64 | 13,72 |
|  | 16 | 256 | 14,44 |  |  | 16 | 256 | 18,69 |
|  | 16 | 512 | 14,89 |  |  | 16 | 512 | 19,43 |

8. Вывод:

* В ходе исполнения лабораторной работы удалось продемонстрировать ускорение, при использовании графического ускорителя как сопроцессора для ускорения вычислений.
* Стоить отметить, что ускорение проявляется только при большом количестве блоков потоков, и большом количестве потоков на блок, так как ядра графического процессора значительно слабее ядра процессора из-за различий в архитектуре и требований к данным ядрам.

9. Исследования ускорения на различных видеокартах nVidia

****