Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

**«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Кафедра вычислительной техники |
| кафедра |

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1**

«Изучение сценария работы пользователя с гибридными кластерными системами»

|  |  |  |
| --- | --- | --- |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_ | Тарасов С. А. |
|  | подпись, дата | инициалы, фамилия |
| Студент КИ20-07Б, 032050342 | \_\_\_\_\_\_\_\_\_\_\_\_ | Таханов А. М. |
| номер группы, зачетной книжкой | подпись, дата | инициалы, фамилия |

Красноярск 2023

**Содержание**

[1 Задание 3](#_Toc146733944)

[2 Код программы 4](#_Toc146733945)

[3 Результат программы 8](#_Toc146733946)

# Задание

Изучить работу пользователя с гибридными кластерными системами и основными утилитами ОС Unix. Получение навыков работы составление сценариев выполнения на языке BASH.

Вариант: вычисление модуля(длины) вектора

# Код программы

#include <cuda\_runtime.h>

#include <curand\_kernel.h>

#include <cmath>

extern "C" {

#include <stdio.h>

\_\_host\_\_ void host\_get\_vector\_length(float\* coordinates, float\* vector\_length, int N) {

    \*vector\_length = 0;

    for(int i = 0; i < N; i++) {

        \*vector\_length += pow(coordinates[i], 2);

    }

    \*vector\_length = sqrtf(\*vector\_length);

}

\_\_global\_\_ void device\_get\_vector\_length(float\* coordinates, float\* vector\_length, int N) {

    int i = blockDim.x \* blockIdx.x + threadIdx.x;

    if (i == 0) {

        \*vector\_length = 0;

    }

    if (i < N) {

        atomicAdd(vector\_length, (coordinates[i]\*coordinates[i]));

    }

}

\_\_global\_\_ void d\_fill\_uniform(

    float \*a, int n, float r, unsigned long long seed) {

    int i = blockDim.x \* blockIdx.x + threadIdx.x;

    if (i < n) {

        curandState\_t state;

        curand\_init(seed, i, 0, &state);

        a[i] = -r + 2 \* r \* curand\_uniform(&state);

    }

}

}

#ifndef REDEFINE

    #define VEC\_LEN 51200000

    #define VEC\_LEN\_INC 512000

    #define CHECK\_FIRST 51200

    #define BLOCK\_SIZE 128

    #define FNAME\_STAMPS "timings.stmp"

    #define PRECISION 10e-2

    #define SEED 27

    #define VEC\_MAX\_ABS\_VAL 101

#endif

#define VEC\_MEM\_SIZE (VEC\_LEN \* sizeof(float))

#define calc\_grid\_size(m) ((m + BLOCK\_SIZE - 1) / BLOCK\_SIZE)

#define ts\_to\_ms(ts) (ts.tv\_sec \* 10e3 + ts.tv\_nsec \* 10e-6)

int main() {

    float \*host\_coordinates \_\_attribute\_\_ ((aligned (64)));

    float \*host\_vector\_length \_\_attribute\_\_ ((aligned (64)));

    float \*host\_fromdevice\_vector\_length \_\_attribute\_\_ ((aligned (64)));

    host\_coordinates = (float\*)malloc(VEC\_MEM\_SIZE);

    host\_vector\_length = (float\*)malloc(sizeof(float));

    host\_fromdevice\_vector\_length = (float\*)malloc(sizeof(float));

    float \*device\_coordinates, \*device\_vector\_length;

    cudaMalloc((void\*\*)&device\_coordinates, VEC\_MEM\_SIZE);

    cudaMalloc((void\*\*)&device\_vector\_length, sizeof(float));

    d\_fill\_uniform<<<calc\_grid\_size(VEC\_LEN), BLOCK\_SIZE>>>(device\_coordinates, VEC\_LEN, VEC\_MAX\_ABS\_VAL, SEED);

    cudaMemcpy(host\_coordinates, device\_coordinates, VEC\_MEM\_SIZE, cudaMemcpyDeviceToHost);

    host\_get\_vector\_length(host\_coordinates, host\_vector\_length, CHECK\_FIRST);

    device\_get\_vector\_length<<<calc\_grid\_size(CHECK\_FIRST), BLOCK\_SIZE>>>(device\_coordinates, device\_vector\_length, CHECK\_FIRST);

    cudaMemcpy(host\_fromdevice\_vector\_length, device\_vector\_length, sizeof(float), cudaMemcpyDeviceToHost);

    \*host\_fromdevice\_vector\_length = sqrtf(\*host\_fromdevice\_vector\_length);

    if (fabs(\*host\_vector\_length - \*host\_fromdevice\_vector\_length) > PRECISION) {

       printf("Panic!\n");

       return -1;

    }

    printf("h\_length: %f\n", \*host\_vector\_length);

    printf("d\_length: %f\n", \*host\_fromdevice\_vector\_length);

    float h\_time;

    timespec h\_start, h\_stop;

    float d\_time;

    cudaEvent\_t d\_start, d\_stop;

    cudaEventCreate(&d\_start);

    cudaEventCreate(&d\_stop);

    FILE\* file = fopen(FNAME\_STAMPS, "w");

    fprintf(file, "Vector Length, CPU Time, GPU Time\n");

    for (int m = VEC\_LEN\_INC; m <= VEC\_LEN; m += VEC\_LEN\_INC) {

        clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, &h\_start);

        host\_get\_vector\_length(host\_coordinates, host\_vector\_length, m);

        clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, &h\_stop);

        h\_time = (ts\_to\_ms(h\_stop) - ts\_to\_ms(h\_start)); // time in ms

        cudaEventRecord(d\_start);

        device\_get\_vector\_length<<<calc\_grid\_size(m), BLOCK\_SIZE>>>(device\_coordinates, device\_vector\_length, m);

        cudaEventRecord(d\_stop);

        cudaEventSynchronize(d\_stop);

        cudaEventElapsedTime(&d\_time, d\_start, d\_stop); // time in ms

        fprintf(file, "%d, %f, %f\n", m, h\_time, d\_time);

    }

    free(host\_coordinates);

    free(host\_vector\_length);

    free(host\_fromdevice\_vector\_length);

    cudaFree(device\_coordinates);

    cudaFree(device\_vector\_length);

    return 0;

}

# Результат программы

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание