**Филиал федерального государственного бюджетного образовательного учреждения высшего образования**

**«Национальный исследовательский университет «МЭИ»**

**в г. Смоленске**

Кафедра вычислительной техники

Направление: 09.04.01. «Информатика и вычислительная техника»

Профиль: «Программное обеспечение средств вычислительной техники и

автоматизированных систем»

Лабораторная работа №4

«**Параллельное программирование с использованием**

**технологии CUDA**»

по курсу:

«Вычислительные системы»

Студент: Старостенков А.А.

Группа: ВМ-22(маг)

Вариант: 19

Преподаватель: Федулов А.С.

Смоленск, 2023

**Задание**

1. Написать, отладить, скомпилировать и запустить на гибридном вычислительном кластере СФМЭИ программу вычисления суммы числового ряда: , где  - общий член ряда, с использованием технологии CUDA. Вариант задания (общий член ряда) выбрать в таблице (по номеру журнала).
2. Предусмотреть замер времени выполнения программы, контроль правильности вычисления суммы.
3. Вычисление членов ряда производить на GPU. Редукцию (суммирование) элементов массива для получения итогового значения суммы можно выполнить на CPU.
4. Получить зависимость времени выполнения параллельной программы от числа отрезков разбиения интервала интегрирования n.
5. При максимальном n исследовать влияние параметров запуска ядра CUDA на время выполнения параллельной программы. Произвести замер времени пересылок между устройством и хостом, времени счета на GPU и времени суммирования членов ряда на CPU. Проанализировать полученные результаты.
6. Все полученные зависимости оформить в виде графиков.



**Ход работы**

1. Написать, отладить, скомпилировать и запустить на гибридном вычислительном кластере СФМЭИ программу вычисления суммы числового ряда: , где - общий член ряда, с использованием технологии CUDA. Вариант задания (общий член ряда) выбрать в таблице (по номеру журнала).

2. Предусмотреть замер времени выполнения программы, контроль правильности вычисления суммы.

#include <cuda.h>

#include <stdio.h>

*// Переопределяем количество итераций цикла*

#define N 1000000

*// Переопределяем максимальное значение для GK110*

#define THREADS\_PER\_BLOCK 1024

*// Функция - ядро*

*\_\_global\_\_* void get\_el(float *\**dev\_el)

{

    float index = threadIdx.x + blockIdx.x \* blockDim.x;

    if (index <= N)

    {

        float znam = (10 \* (index + 1) \* (index + 1) - 2 \* (index + 1) - 3);

        dev\_el[(int)index] = 3.0 / znam;

    }

    else

    {

*return*;

    }

}

int main(void)

{

    float gpu\_calc\_time, gpu\_send\_time;

    cudaEvent\_t start, stop;

*// Создание событий*

    cudaEventCreate(&start);

    cudaEventCreate(&stop);

*// Host копии*

    float \*el;

*// Device копии*

    float \*dev\_el;

*// Выделение памяти для device и host элементов*

    int size = N \* sizeof(float);

    el = (float \*)malloc(size);

    cudaMalloc((void \*\*)&dev\_el, size);

*// Отсечка*

    cudaEventRecord(start, 0);

    get\_el<<<(int)(N / THREADS\_PER\_BLOCK) + 1, THREADS\_PER\_BLOCK>>>(dev\_el);

*// Отсечка*

    cudaEventRecord(stop, 0);

    cudaEventSynchronize(stop);

    cudaEventElapsedTime(&gpu\_calc\_time, start, stop);

*// Копирование результата в CPU*

*// Отсечка*

    cudaEventRecord(start, 0);

    cudaMemcpy(el, dev\_el, size, cudaMemcpyDeviceToHost);

*// Отсечка*

    cudaEventRecord(stop, 0);

    cudaEventSynchronize(stop);

    cudaEventElapsedTime(&gpu\_send\_time, start, stop);

*// Освобождение памяти device*

    cudaFree(dev\_el);

*// Переменные времени и суммы*

    float cpu\_t;

    double sum = 0;

*// Вычисление суммы числового ряда*

*// Отсечка*

    cudaEventRecord(start, 0);

    for (float i = 0; i < N; i++)

    {

        sum += el[(int)i];

    }

*// Отсечка*

    cudaEventRecord(stop, 0);

    cudaEventSynchronize(stop);

    cudaEventElapsedTime(&cpu\_t, start, stop);

    printf("Количество итераций: %d\n", N);

    printf("Cумма числового ряда: %.6f\n", sum);

    printf("Время вычисления в gpu: %.4f мс\n", gpu\_calc\_time);

    printf("Время пересылок: %.4f мс\n", gpu\_send\_time);

    printf("Время суммирования в cpu: %.4f мс\n", cpu\_t);

    printf("Общее время: %.4f мс\n", gpu\_calc\_time + gpu\_send\_time + cpu\_t);

    cudaEventDestroy(start);

    cudaEventDestroy(stop);

*return* 0;

}

3. Вычисление членов ряда производить на GPU. Редукцию (суммирование) элементов массива для получения итогового значения суммы можно выполнить на CPU.

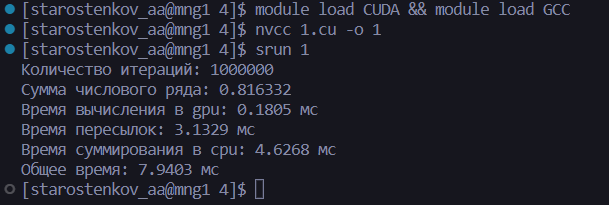


Рисунок 1 - Программа вычисления суммы числового ряда с использованием технологии CUDA

4. Получить зависимость времени выполнения параллельной программы от числа отрезков разбиения интервала интегрирования n.

5. При максимальном n исследовать влияние параметров запуска ядра CUDA на время выполнения параллельной программы. Произвести замер времени пересылок между устройством и хостом, времени счета на GPU и времени суммирования членов ряда на CPU. Проанализировать полученные результаты.

Для выполнения заданий 4 и 5 необходимо произвести измерения работы программы при различных значениях числа итераций. Были выбраны следующие значения N: 100, 1000, 10000, 100000, 1000000, 10000000.

На рисунках 2 - 7 представлены результаты вычислений.

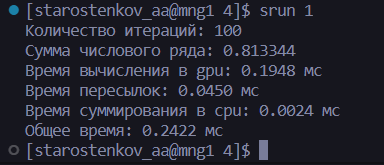


Рисунок 2 – Измерение времени при N = 100

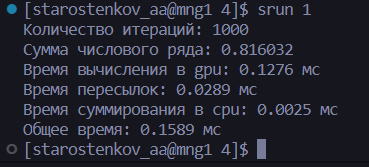


Рисунок 3 – Измерение времени при N = 1000

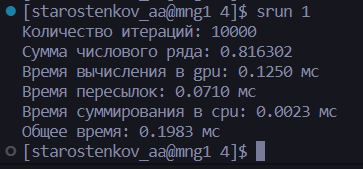


Рисунок 4 – Измерение времени при N = 10000

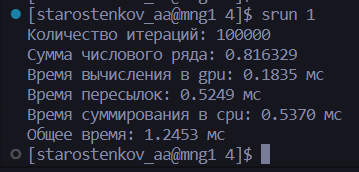


Рисунок 5 – Измерение времени при N = 100000

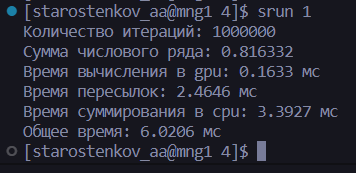


Рисунок 6 – Измерение времени при N = 1000000

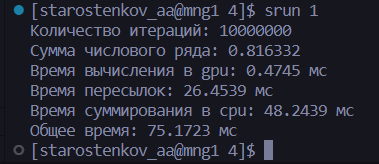


Рисунок 7 – Измерение времени при N = 10000000

Для наглядности все результаты вычисления времени занесены в таблицу 1.

Таблица 1 – Результаты измерения времени

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кол-во итераций | Время вычисления в gpu | Время пересылок | Время суммирования в cpu | Общее время |
| 100 | 0,1948 | 0,0450 | 0,0024 | 0,2422 |
| 1000 | 0,1276 | 0,0289 | 0,0025 | 0,1589 |
| 10000 | 0,1250 | 0,0710 | 0,0023 | 0,1983 |
| 100000 | 0,1835 | 0,5249 | 0,5370 | 1,2453 |
| 1000000 | 0,1633 | 2,4646 | 3,3927 | 6,0206 |
| 10000000 | 0,4745 | 26,4539 | 48,2439 | 75,1723 |

На рисунке 8 представлены графики зависимости времени от числа итераций.

Рисунок 8 – График зависимости времени от числа итераций

**Выводы**:

По данному графику видно, что время выполнения ядра при любом количестве итераций не меняется. Это связано с тем, что ядро не содержит циклы, которые зависят от количества итераций. Для вычисления членов ряда N потоков запускаются параллельно.

Время пересылки, суммирования в cpu и общее время ведут себя одинаково. При N равному интервалу от 100 до 100000 время меняется несущественно, но после 10000 время возрастает прямо пропорционально N.