[Тероия Параллелизма](https://classroom.google.com/u/1/c/NTg0Nzg0MTE5Mzgy" \t "_self)

Отчет

Уравнение теплопроводности

Выполнил группа 21933, Рюмин Вадим Юрьевич

Дата 29.03.2023

Цели работы

Реализовать решение уравнение теплопроводности (пятиточечный шаблон) в двумерной области на  равномерных сетках (128^2, 256^2,  
512^2, 1024^2). Граничные условия – линейная интерполяция между углами области. Значение в углах – 10, 20, 30, 20. Ограничить точность – 10^-6 и максимальное число итераций – 10^6.

Используемый компилятор

pgcc

Используемый профилировщик

Nsight Systems

Как производили замер времени работы

Для однопоточной программы, при помощи библиотеки “time.h”

Для многопоточной программы, при помощи nvprof

Для программы на, при помощи GPU PGI\_ACC\_TIME=1

Выполнение на CPU

CPU-onecore

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки | Время  Выполнения(s) | Точность | Количество  итераций |
| 128\*128 | 1.5 | 10^-6 | 35085 |
| 256\*256 | 18 | 10^-6 | 123706 |
| 512\*512 | 240 | 10^-6 | 417181 |

CPU-multicore(./cpu\_m\_2)

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки | Время  Выполнения(s) | Точность | Количество  итераций |
| 128\*128 | 1,3 | 10^-6 | 34927 |
| 256\*256 | 5.4 | 10^-6 | 123706 |
| 512\*512 | 26 | 10^-6 | 417185 |
| 1024\*1024 | 291 | 40^-6 | 10^6 |

Диаграмма сравнения время работы СPU-one и CPU-multi

Выполнение на GPU

Этапы оптимизации на сетке 512\*512

(количество итераций при профилировании 100)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Этап No | Время  Выполнения (ms) | Точность | Количество  итераций | Комментарии  (что было сделано) |
| 1 | 874 | 0.107 | 100 | Добавил «#pragma acc data copyin(Anew[:sizearr\*sizearr],  A[:sizearr\*sizearr])» и «#pragma acc data present(A, Anew)»; |
| 2 | 3366 | 0.107 | 100 | Добавил «#pragma acc parallel reduction(max:err)»; |
| 3 | 900 | 0.107 | 100 | Добавил «#pragma acc loop independent» для каждого цикла; |
| 4 | 1072 | 0.107 | 100 | Добавил «#pragma acc data copy(err)», «#pragma acc data present(err)», а также «#pragma acc update device(err)» и «#pragma acc update host(err)»; |
| 5 | 900 | 0.107 | 100 | Добавил «#pragma acc data copyin (sizearr)» и «#pragma acc data present(sizearr)»; |
| 6 | 900 | 0.107 | 100 | Выполняю «#pragma acc update device(err)» и «#pragma acc update host(err)» каждые 50 итераций; |
| 7 | - | - | - | Выполняю «#pragma acc update device(err)» и «#pragma acc update host(err)» каждые «sizearr» итераций. |

Этапы оптимизации на сетке 512\*512 (Скриншоты)

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |

Диаграмма оптимизации

(по горизонтали номер этапа; по вертикали время работы)

GPU – оптимизированный вариант

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки | Время  Выполнения(s) | Точность | Количество  итераций |
| 128\*128 | 2.5 | 10^-6 | 34944 |
| 256\*256 | 6 | 10^-6 | 123904 |
| 512\*512 | 15 | 10^-6 | 417280 |
| 1024\*1024 | 43 | 40^-6 | 10^6 |

Диаграмма сравнения времени работы CPU-one, CPU-multi, GPU(оптимизированный вариант) для разных размеров сеток

Вывод: Программа при малых размерах (128\*128 и 256\*256) считается быстрее на CPU, чем GPU т.к. происходит относительно мало итераций. На GPU время расчётов сопоставимо со времени обмена данными, если их обновлять на CPU при каждой итерации. Получилось отправлять результат «err» через определённое количество итераций. Так как с размером матрицы нужно больше итераций её расчёта. Я обновляю «err» на CPU когда итерация кратна размеру матрицы.

Приложение

Ссылка на GitHub

[CS/Two at main · RyuminVadim/CS (github.com)](https://github.com/RyuminVadim/CS/tree/main/Two)

Выполнение на GPU

Программы для CPU и GPU(скриншот)

Программа CPU

Изображение выглядит как текст

Автоматически созданное описание

Программа GPU

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание