# Теория параллелизма

## Отчет

## Решение уравнения теплопроводности

Выполнил 22931, Чернов Иван Алексеевич

30.04.2024

Задача: Реализовать решение уравнение теплопроводности

Профилировщик: “Nsight Systems”.

# Выполнение на CPU

## CPU one-core

| Размер сетки | Время выполнения | Точность | Кол-во итераций |
|---|---|---|---|
| 128 | 0,7 | 1,00E-06 | 30100 |
| 256 | 8,4 | 1,00E-06 | 102900 |
| 512 | 124,9 | 1,00E-06 | 339600 |
| 1024 | 1921,3 | 1,00E-06 | 193500 |

## CPU multi-core

| Размер сетки | Время выполнения | Точность | Кол-во итераций |
|---|---|---|---|
| 128 | 1 | 1,00E-06 | 15700 |
| 256 | 3,7 | 1,00E-06 | 45100 |
| 512 | 13 | 1,00E-06 | 108500 |
| 1024 | 62 | 1,00E-06 | 193500 |

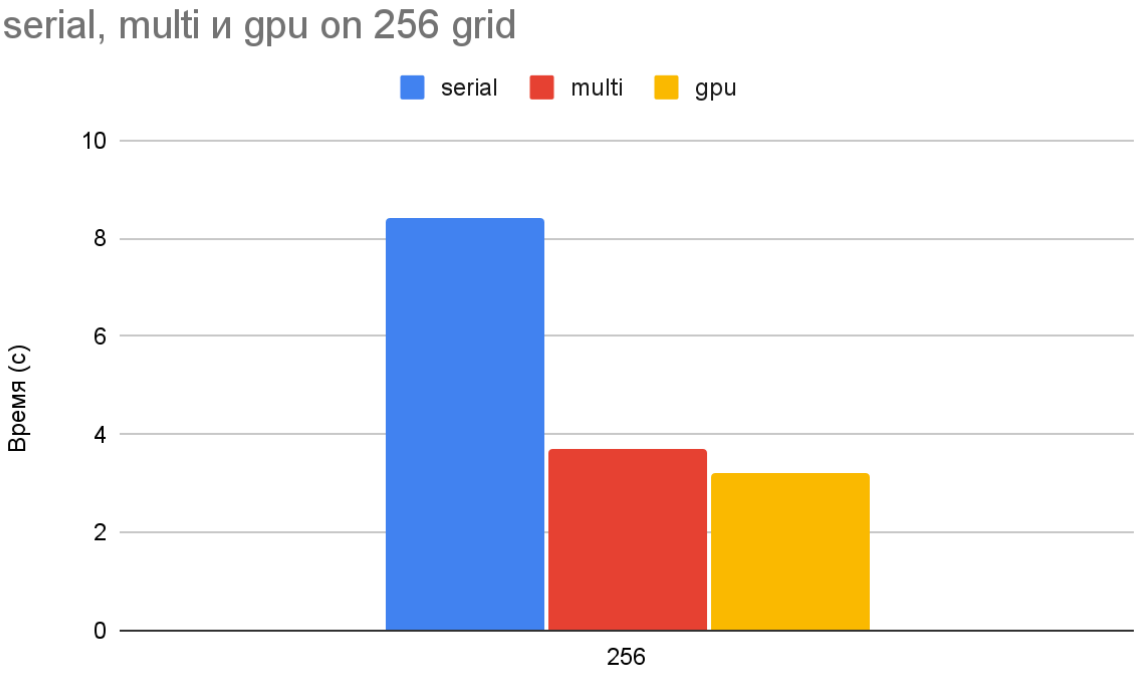
## Диаграмма сравнения время работы CPU-one и CPU-multi

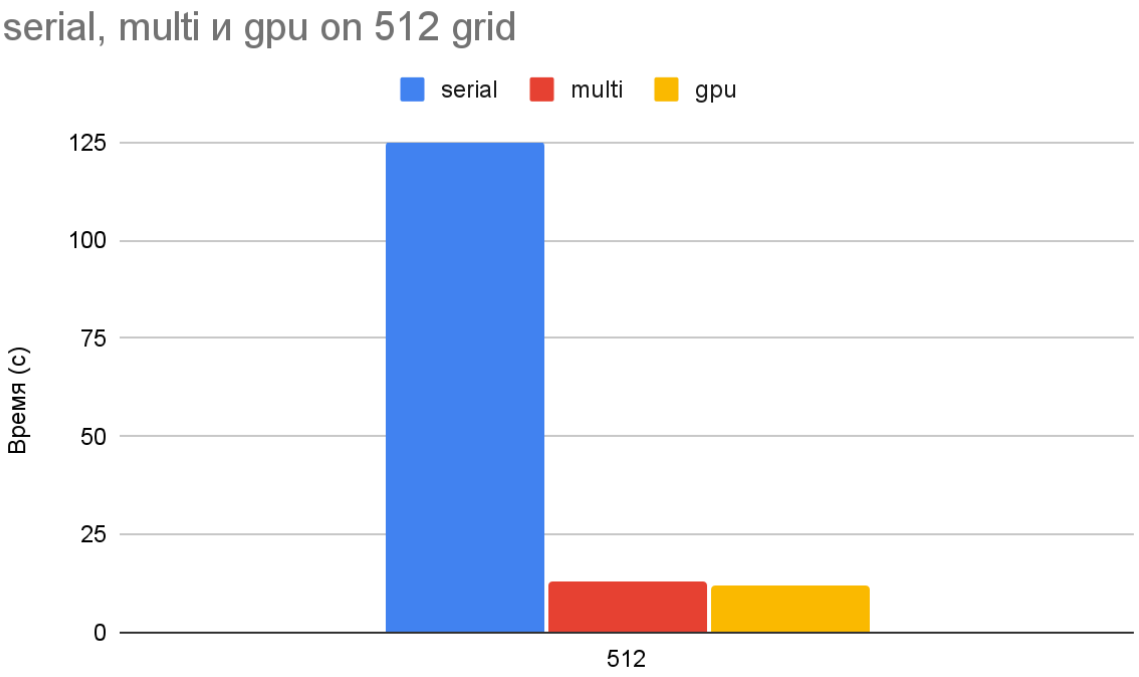
## GPU

| Размер сетки | Время выполнения | Точность | Кол-во итераций |
|---|---|---|---|
| 128 | 0,7 | 1,00E-06 | 15700 |
| 256 | 3,2 | 1,00E-06 | 45100 |
| 512 | 11,8 | 1,00E-06 | 108500 |
| 1024 | 58 | 1,00E-06 | 193500 |

## GPU Optimisation on 512x512 grid

| Этап N | Время выполнения | Точность | Описание |
|---|---|---|---|
| 1 | 0,65 | 1,00E-06 | BaseLine |
| 2 | 0,55 | 1,00E-06 | Использование всех видеокарт |
| 3 | 0,58 | 1,00E-06 | Вызов отдельной функции расчета ошибки каждую сотую итерацию |
| 4 | 0,6 | 1,00E-06 | Вызов отдельной функции расчета ошибки каждую тысячную итерацию |


serial, multi и gpu on 256 grid
serial
multi
gpu
10
8
6
4
2
0
Время (с)
256


serial, multi и gpu on 512 grid
serial
multi
gpu
125
100
75
50
25
0
Время (с)
512

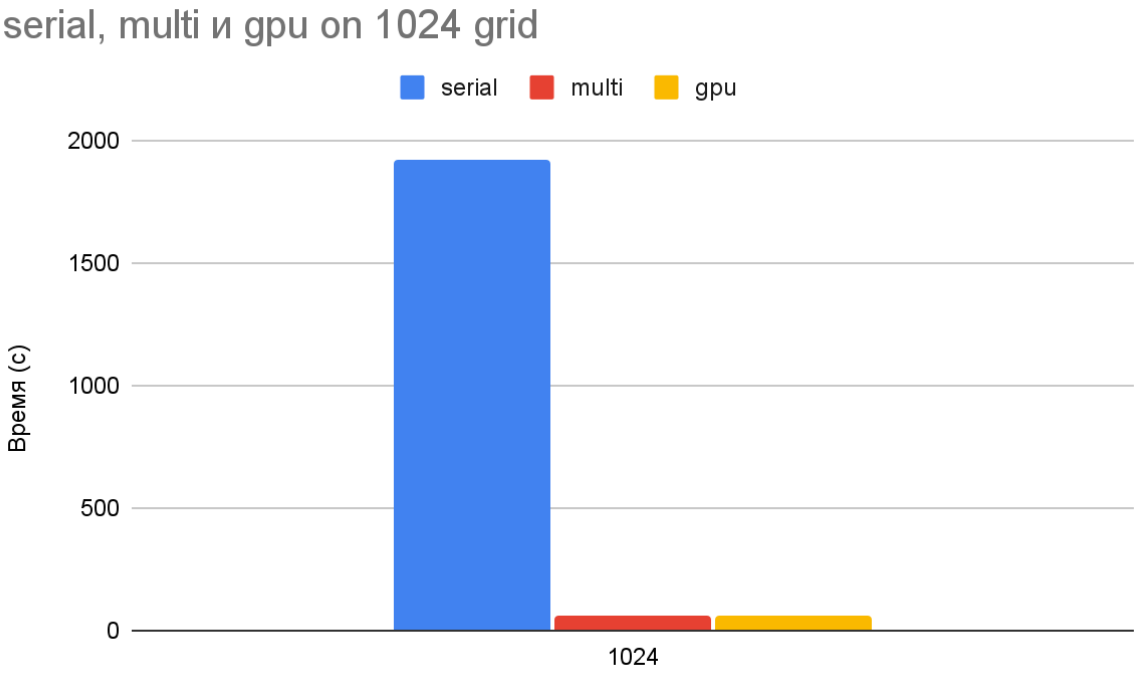
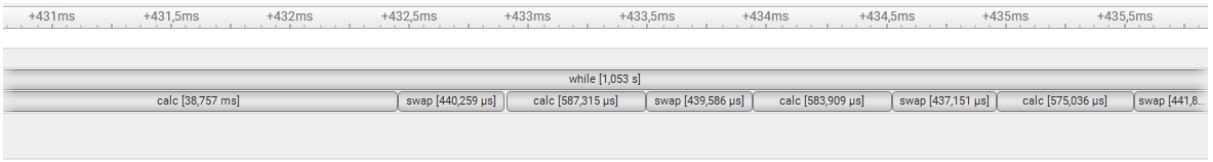
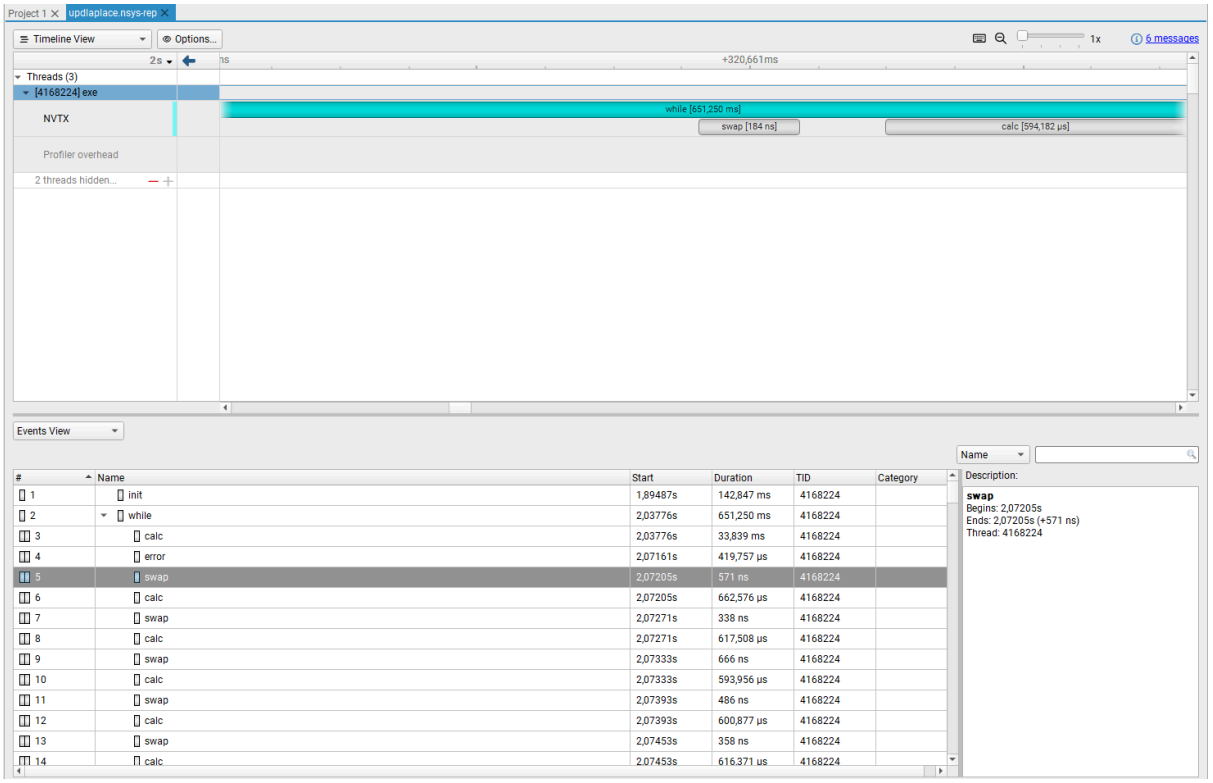



Диаграмма сравнения времени работы CPU-one, CPU-multi, GPU для разных размеров сеток

## Работа программы в профилировщике Nsight Systems




Переписан swap и расчет ошибки каждые 100 эпох




Вывод: программа отлично распараллеливается, но от перехода на gpu прироста нету.

## Расчет ошибки с помощью библиотеки cuBLAS

```
double Laplace::calcError(){
    double error = 0.0;
    double *d_A, *d_Anew, *d_error;
    double *h_C = new double[m * m];
    cudaMalloc(&d_A, m * n * sizeof(double));
    cudaMalloc(&d_Anew, m * n * sizeof(double));
    cudaMalloc(&d_error, m * n * sizeof(double));
    cudaMemcpy(d_A, A, m * n * sizeof(double), cudaMemcpyHostToDevice);
    cudaMemcpy(d_Anew, Anew, m * n * sizeof(double), cudaMemcpyHostToDevice);
    cublasHandle_t handle;
    cublasCreate(&handle);
    const double alpha = -1.0;
    const double beta = 1.0;
    cublasDgeam(handle, CUBLAS_OP_N, CUBLAS_OP_N, m, n, &alpha, d_Anew, m, &beta, d_A, m, d_error, m);
    cublasDasum(handle, m * n, d_error, 1, &error);
    cublasDestroy(handle);
    cudaFree(d_A);
    cudaFree(d_Anew);
    cudaFree(d_error);
    return error;
}
```

## Сравнение с обычным acc loop

| Размер сетки | cuBLAS, с | acc loop, с |
|---|---|---|
| 32 | 3,4 | 1,4 |
| 64 | 14,6 | 5,7 |
| 128 | 46,9 | 21 |