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	999900

CPU multi-core

Размер сетки	Время выполнения	Точность	Кол-во итераций
128	1	1,00E-06	30100
256	3,7	1,00E-06	102900
512	13	1,00E-06	339600
1024	62	1,00E-06	999900

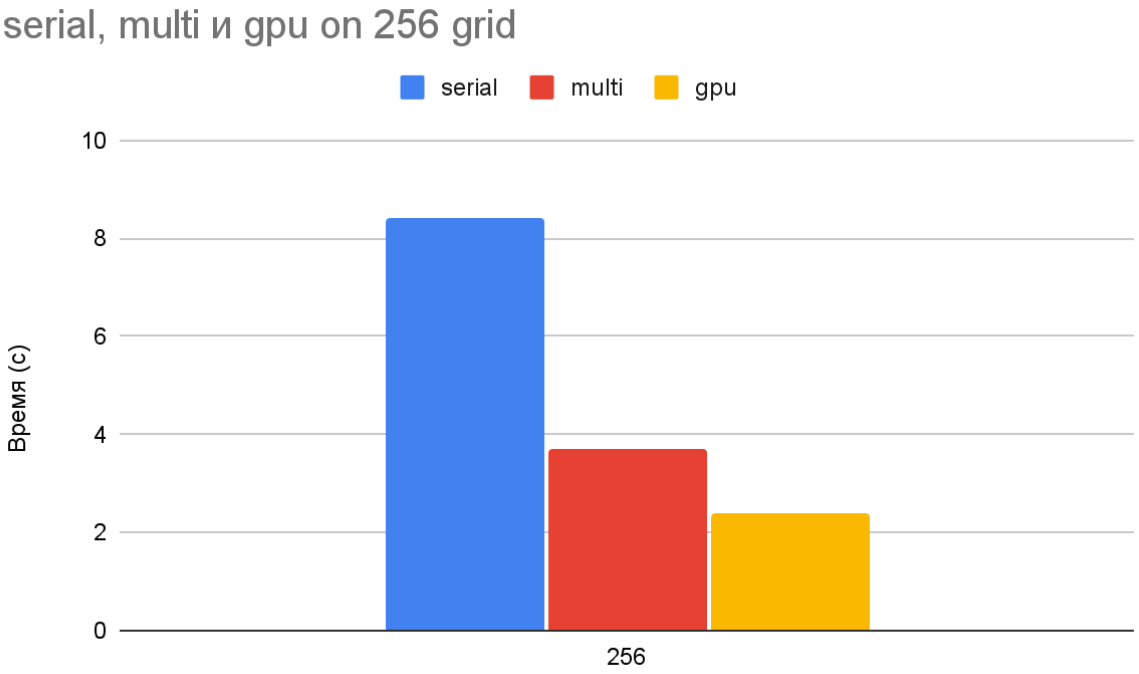
Диаграмма сравнения время работы CPU-one и CPU-multi

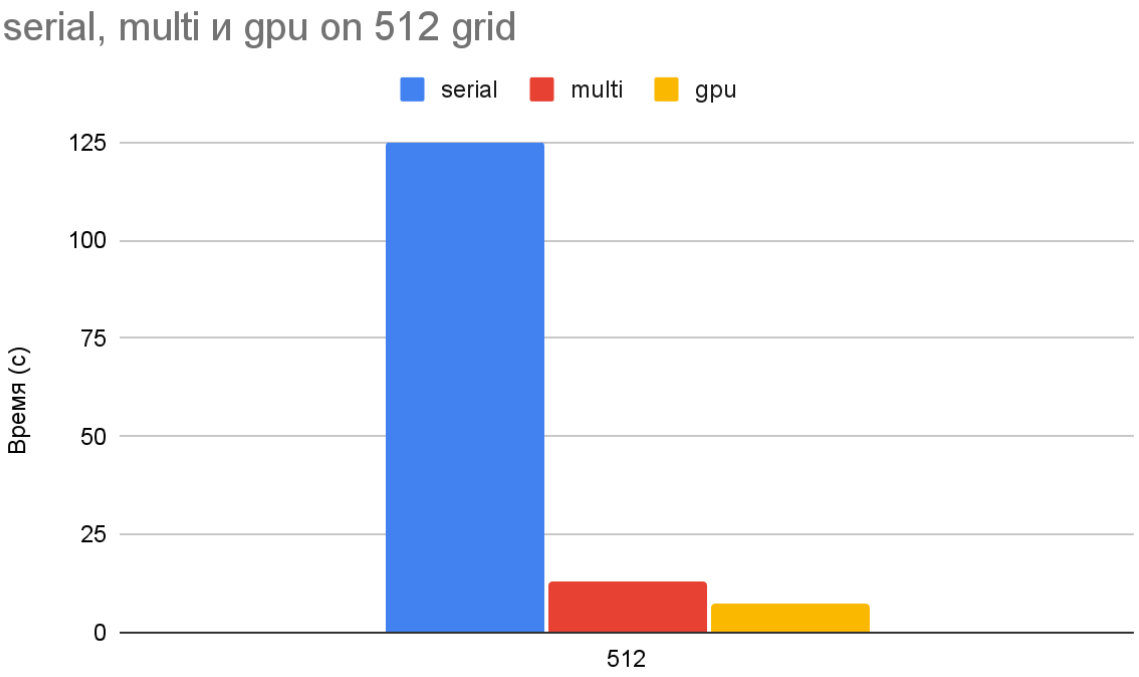
GPU

Размер сетки	Время выполнения	Точность	Кол-во итераций
128	0,7	1,00E-06	30100
256	3,2	1,00E-06	102900
512	11,8	1,00E-06	339600
1024	58	1,00E-06	999900

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

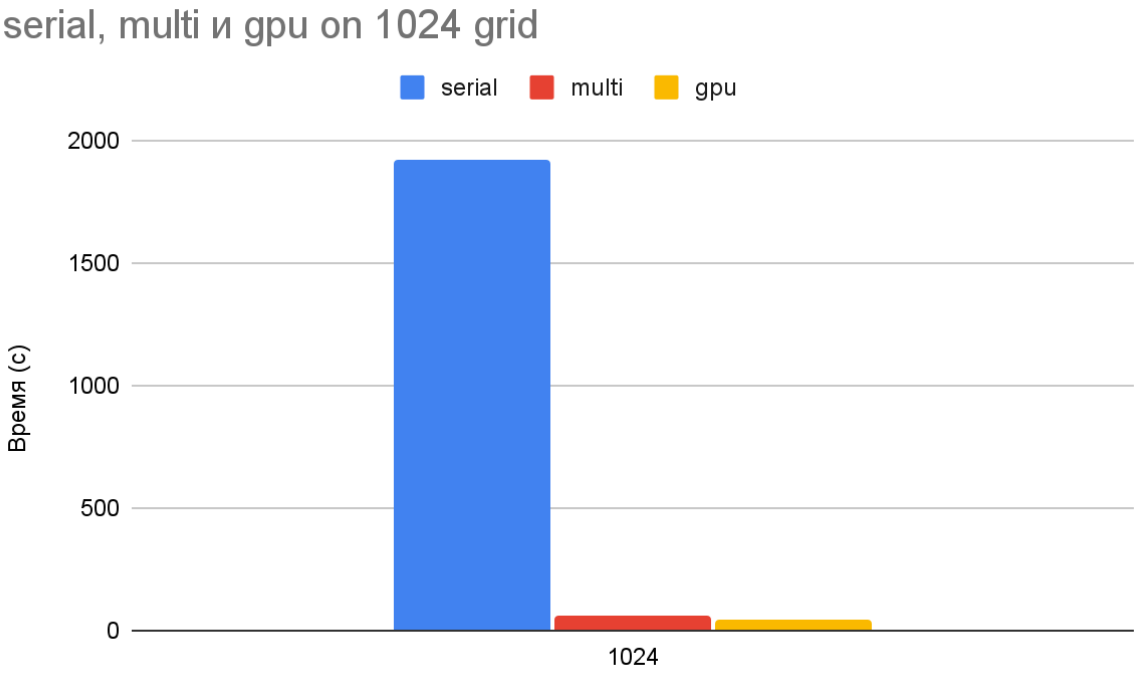
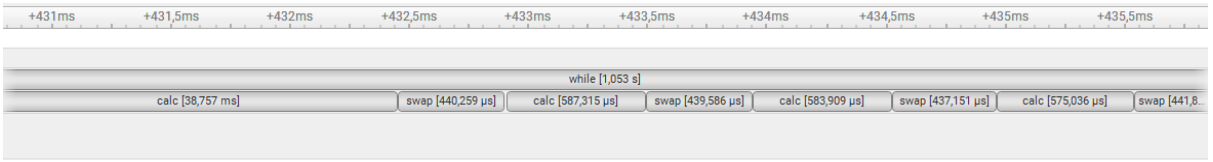


Диаграмма сравнения времени работы CPU-one, CPU-multi, GPU для разных размеров сеток

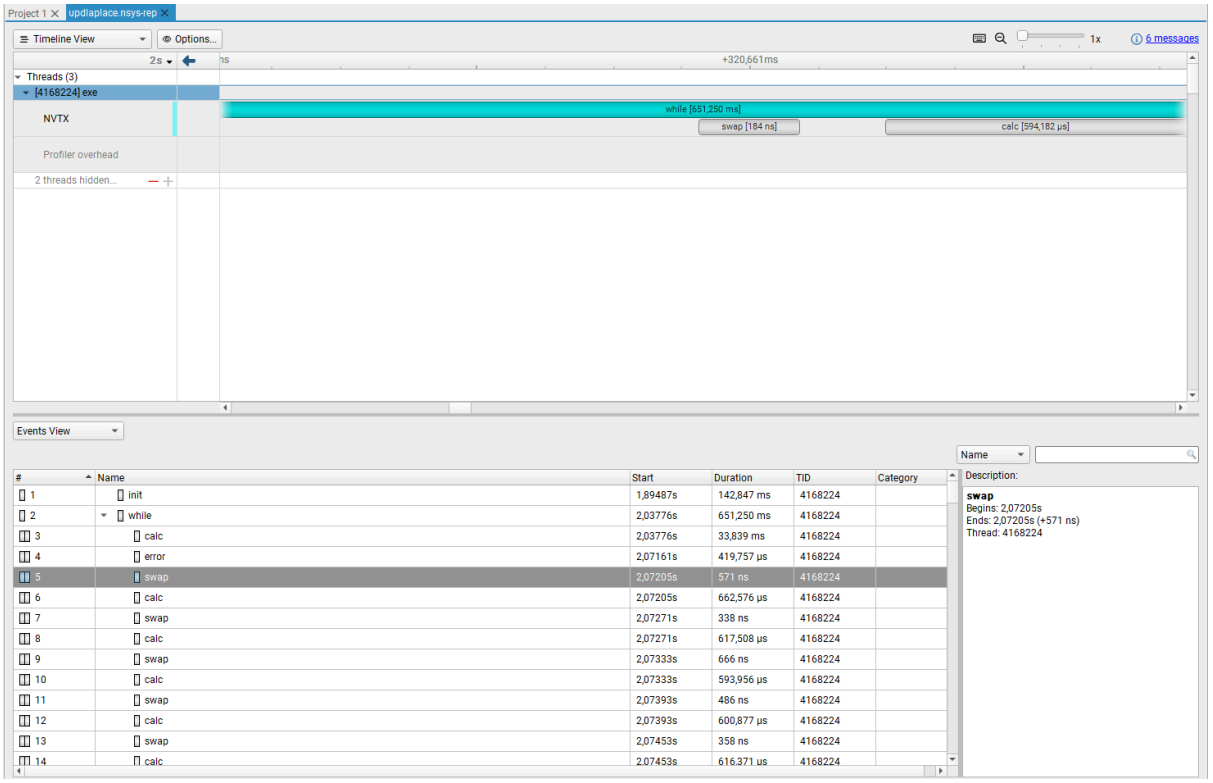
Общая таблица

grid size	serial	multi	gpu	gpu cuda	cuBLAS
128	0,7	0,7	0,6	0,6	0,7
256	8,4	3,7	2,4	3,1	2,4
512	124,9	13	7,6	7,7	7,8
1024	1921,3	62	48	50	51

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(){
    cublasHandle_t handler;
    cublasStatus_t status;

    status = cublasCreate(&handler);
    if (status != CUBLAS_STATUS_SUCCESS) {
        std::cerr << "cublasCreate failed with error code: " << status << std::endl;
        return 13;
    }
    double error = 1.0;
    int idx = 0;
    double alpha = -1.0;
    #pragma acc data present (A, Anew) wait
    #pragma acc host_data use_device(A, Anew)
    {
        status = cublasDaxpy(handler, n * n, &alpha, Anew, 1, A, 1);

        if (status != CUBLAS_STATUS_SUCCESS) {
            std::cerr << "cublasDaxpy failed with error code: " << status << std::endl;
            exit (13);
        }


        status = cublasIdamax(handler, n * n, A, 1, &idx);
        if (status != CUBLAS_STATUS_SUCCESS) {
            std::cerr << "cublasIdamax failed with error code: " << status << std::endl;
            exit (13);
        }
    }

    #pragma acc update host(A[idx - 1])
    error = std::fabs(A[idx - 1]);

    #pragma acc host_data use_device(A, Anew)
    status = cublasDcopy(handler, n * n, Anew, 1, A, 1);
    if (status != CUBLAS_STATUS_SUCCESS) {
        std::cerr << "cublasDcopy failed with error code: " << status << std::endl;
        exit (13);
    }
    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