Теория параллелизма

Отчет

Уравнение теплопроводности

Выполнил: Пучков Владислав, гр. 22933

Дата 20.05.2024

Цель: Реализовать решение уравнения теплопроводности в двумерной области с использованием разностной схемы (пятиточечный шаблон) на равномерных сетках. Программа должна учитывать линейную интерполяцию на границах и заданные значения в углах, ограничивать точность до 10^−6 и максимальное число итераций до 10^6. Реализация должна быть на C++ с использованием OpenACC для переноса на GPU. Необходимо сравнить производительность на CPU и GPU, провести профилирование и оптимизацию кода.

Используемый компилятор: pgc++ 23.11-0

Используемый профилировщик: NVIDIA Nsight Systems

Как производили замер времени работы: При помощи

Библиотеки chrono замерил время начала и конца

цикла итераций

**Выполнение на CPU**

**CPU**-onecore

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки | Время выполнения (сек) | Точность | Количество операций |
| **128\*128** | 0,65 | 0.000001 | 31000 |
| **256\*256** | 7,98 | 0.000001 | 110000 |
| **512\*512** | 103,4 | 0.000001 | 340000 |

**CPU**-multicore

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки | Время выполнения (сек) | Точность | Количество операций |
| **128\*128** | 0,52 | 0.000001 | 31000 |
| **256\*256** | 1,81 | 0.000001 | 110000 |
| **512\*512** | 34,1 | 0.000001 | 340000 |
| **1024\*1024** | 321,62 | 0.000001 | 1000000 |

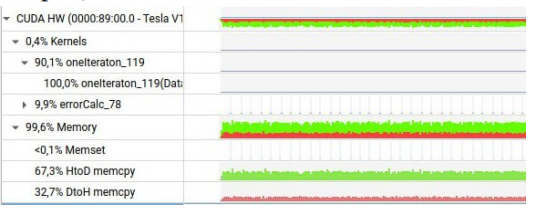
**Диаграмма** **сравнения** время работы CPU-one и **CPU**-multi

**Выполнение на GPU   
Этапы оптимизации** на сетке 1024\*1024

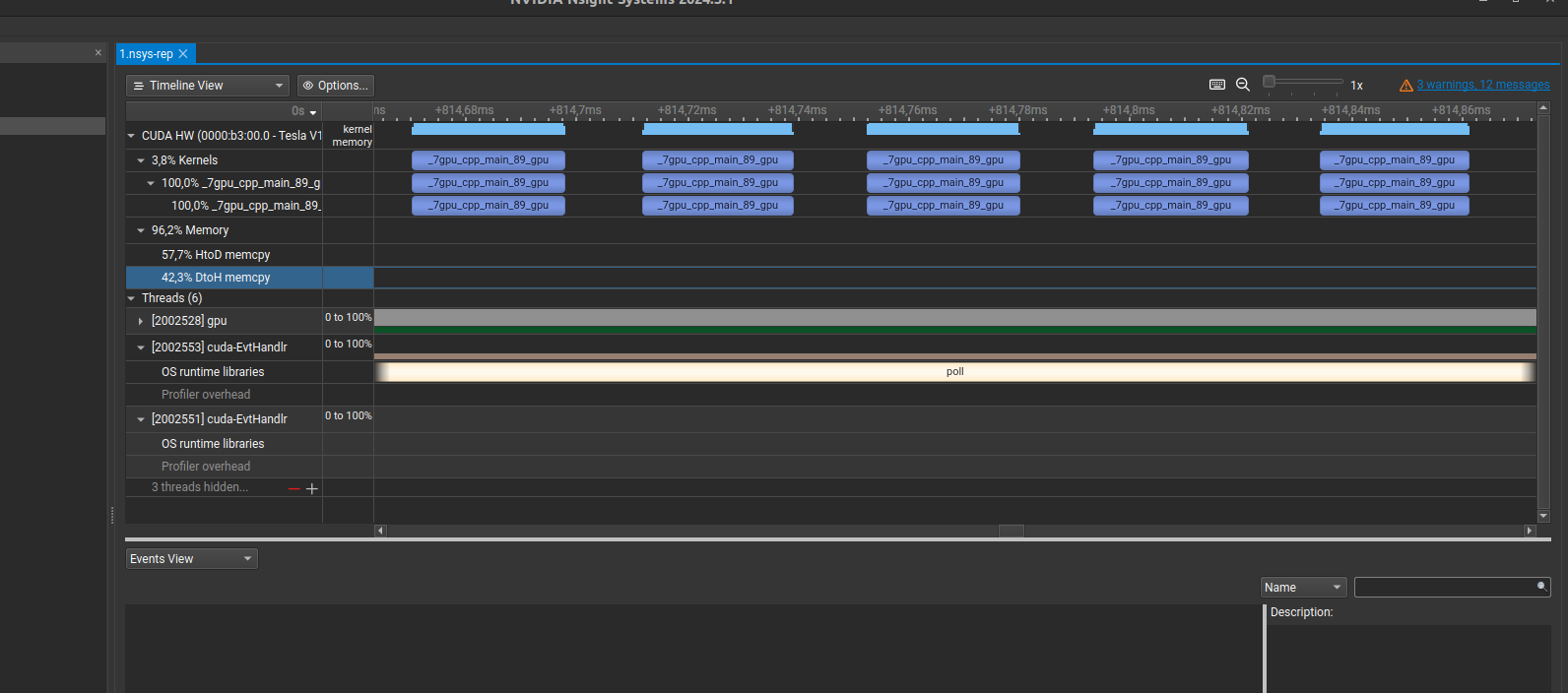
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Этап | Время выполнения | Точность | Количество итераций | Комментарии |
| 1 | 71,83 | >0.000001 | 1000000 | Замена swap через temp на swap через указатели |
| 2 | 44,56 | >0.000001 | 1000000 | Возвращение ошибки каждые 1000 операций |
| 3 | 42,768 | >0.000001 | 1000000 | Замена распараллеливания цикла на #pragma acc parallel loop independent collapse(2) vector vector\_length(256) gang num\_gangs(1024) present(currentMatrix, previousMatrix) |
| 4 | 34,987 | >0.000001 | 1000000 | Выбор самой свободной видеокарты через библиотеку #include <openacc.h> и команду acc\_set\_device\_num( N, acc\_device\_nvidia ); |

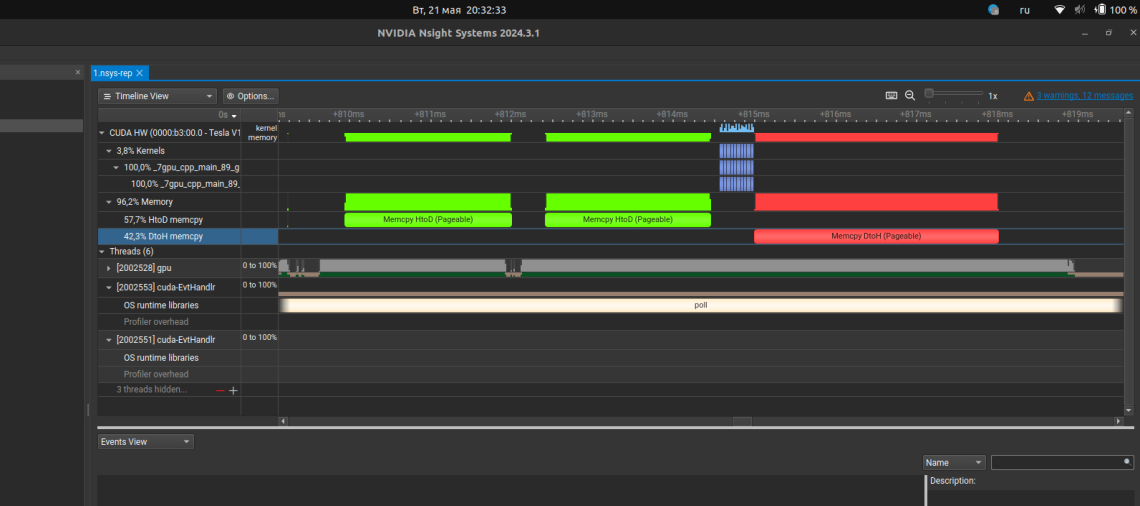
**Диаграмма оптимизации**

Профилирование:



Большая часть времени GPU отправляет данные на каждой итерации, вместо вычислений





скорость работы была увеличена за счёт того что ошибка вычисляется не на каждой итерации, так как максимизация переменной среди потоков требует времени на синхронизацию между потоками.

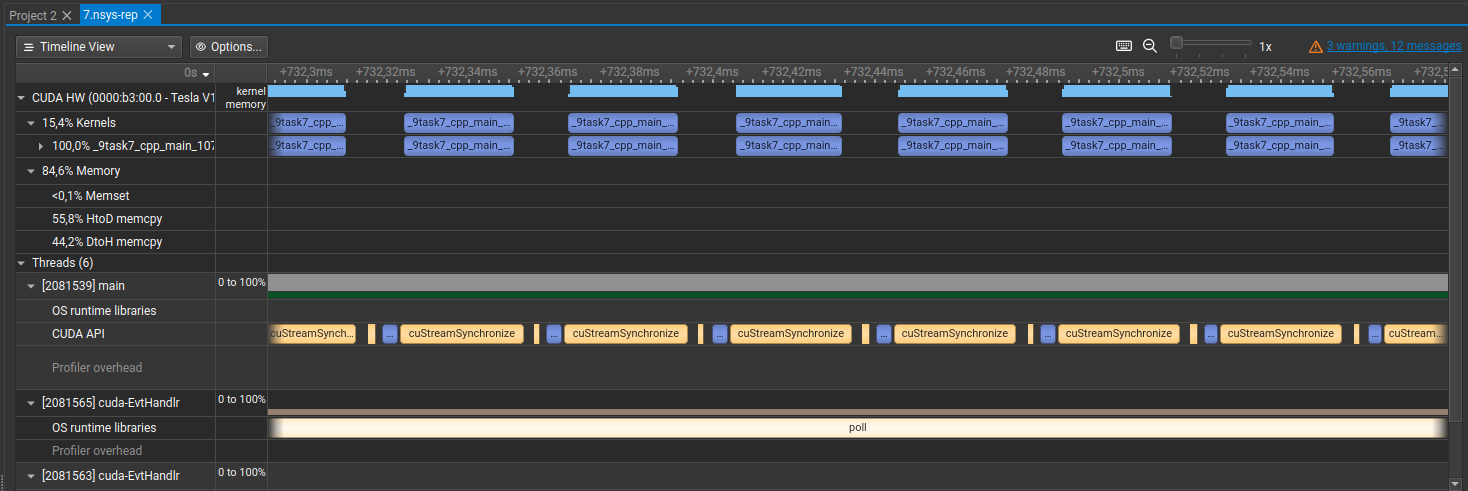
**GPU - оптимизированный вариант**

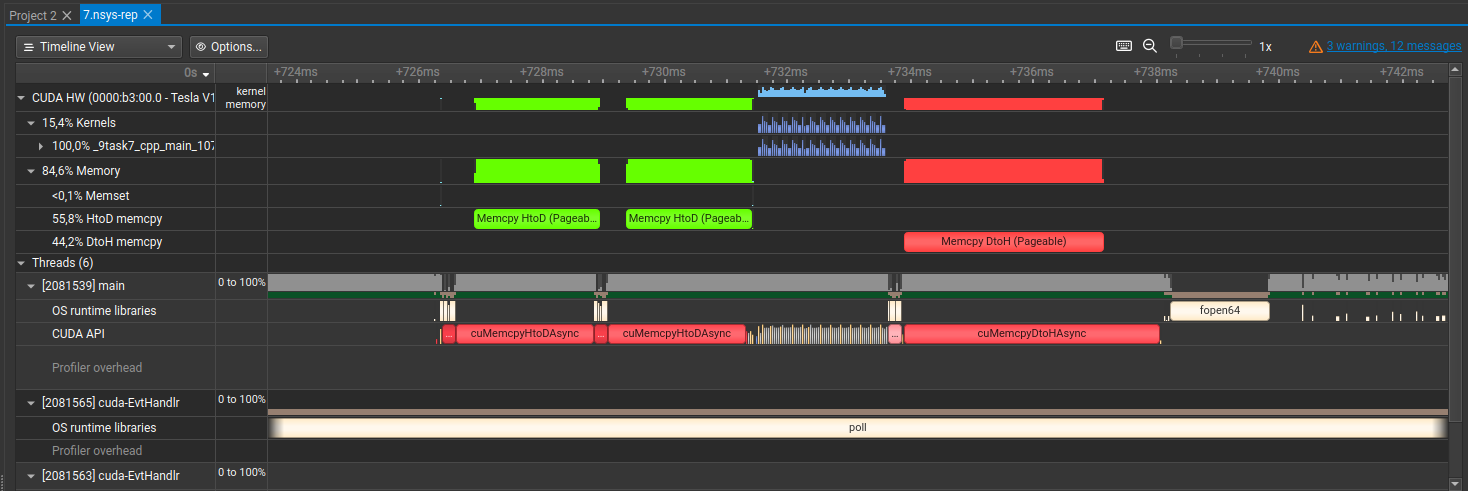
|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки | Время выполнения(c) | Точность | Количество операций |
| **128\*128** | 0,43 | 0.000001 | 40000 |
| **256\*256** | 1,27 | 0.000001 | 110000 |
| **512\*512** | 4,2 | 0.000001 | 340000 |
| **1024\*1024** | 34,8 | >0.000001 | 1000000 |

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

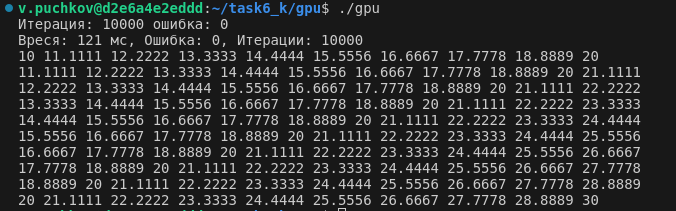
GPU - **оптимизированный** вариант + cuBLAS

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки | Время выполнения(c) | Точность | Количество операций |
| **128\*128** | 0,5 | 0.000001 | 40000 |
| **256\*256** | 1,63 | 0.000001 | 110000 |
| **512\*512** | 5,63 | 0.000001 | 340000 |
| **1024\*1024** | 33,82 | >0.000001 | 1000000 |

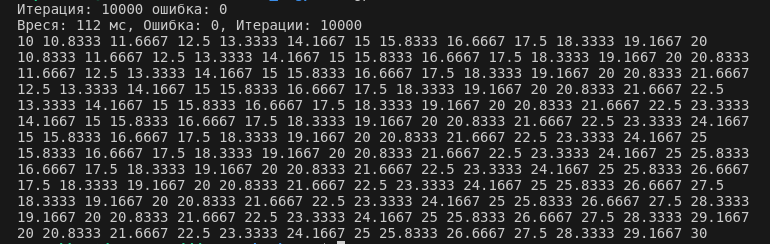




10х10



13х13



Вывод: Лучше всего GPU себя показывает на больших матрицах, на небольших размерах GPU, CPU-onecore, CPU-milticore – очень близки по времени. Возврат с GPU на CPU ошибки при большом количестве операций, необязательно проверять ошибку каждый раз, так как копирование занимает также время