

Теория параллелизма

Отчет

Уравнение теплопроводности

Выполнил: Пучков Владислав, гр. 22933

Дата 27.05.2024

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++, используя функционал CUDA, CUB для распараллеливания алгоритма на графический процессор.

Используемый компилятор: pgc++ 23.11-0

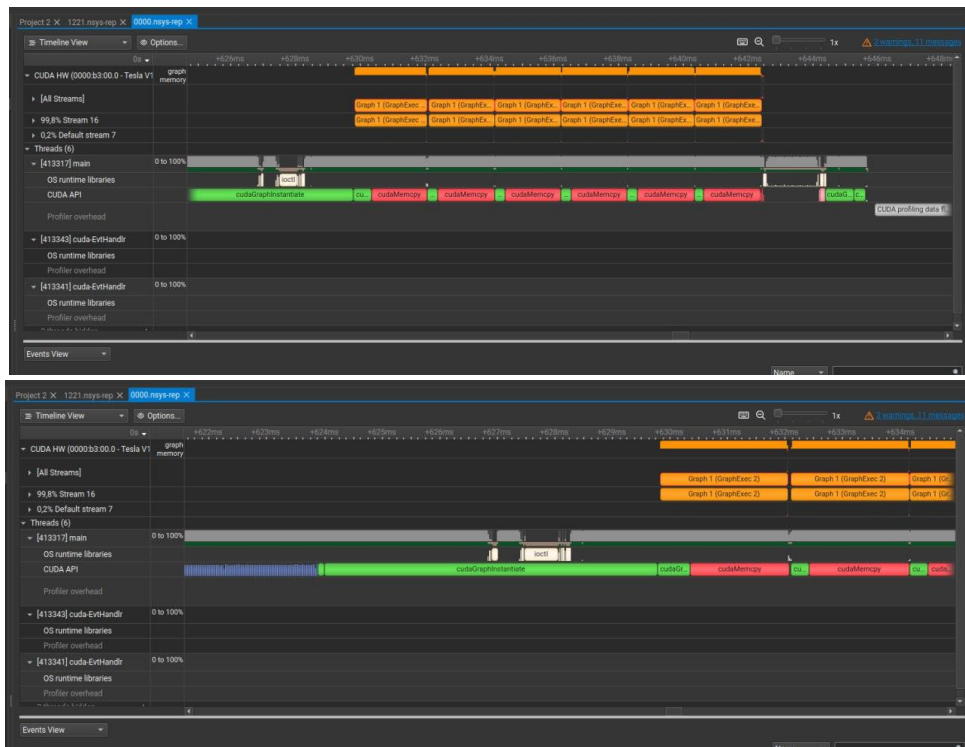
Используемый профилировщик: NVIDIA Nsight Systems

Как производили замер времени работы: При помощи Библиотеки chrono замерил время начала и конца цикла итераций

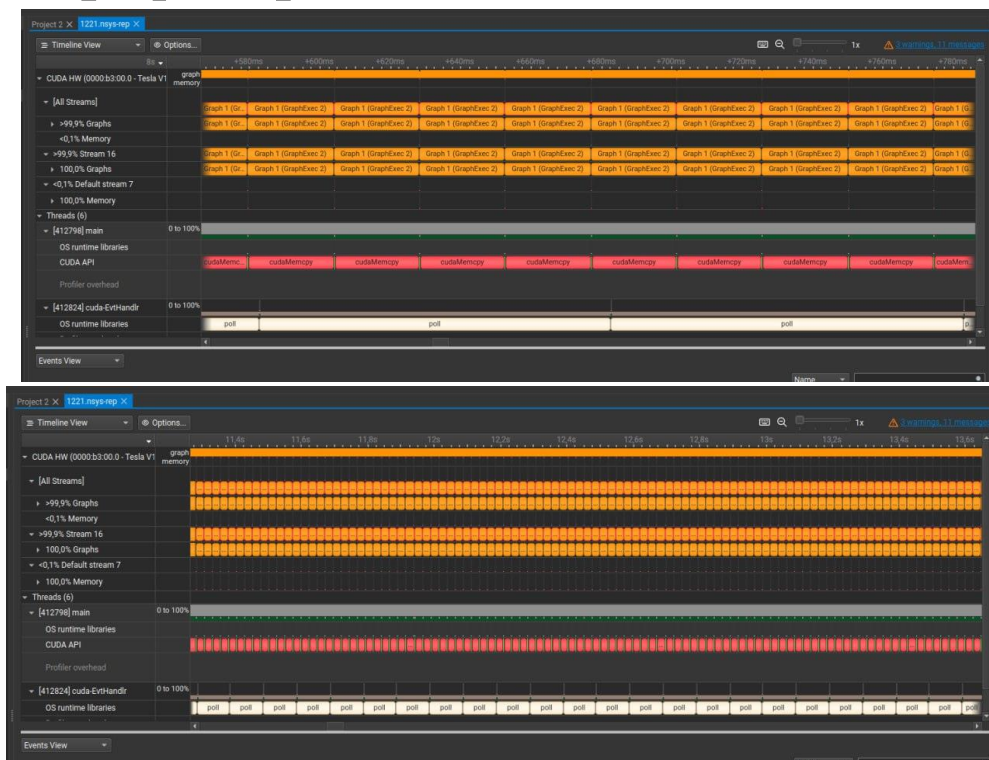
GPU (CUDA)

Размер сетки	Время выполнения(с)	Точность	Количество операций
128*128	0,063	0.000001	31000
256*256	0,224	0.000001	103000
512*512	1,328	0.000001	340000
1024*1024	24,253	>0.000001	1000000

Профилирование на сетке 50x50

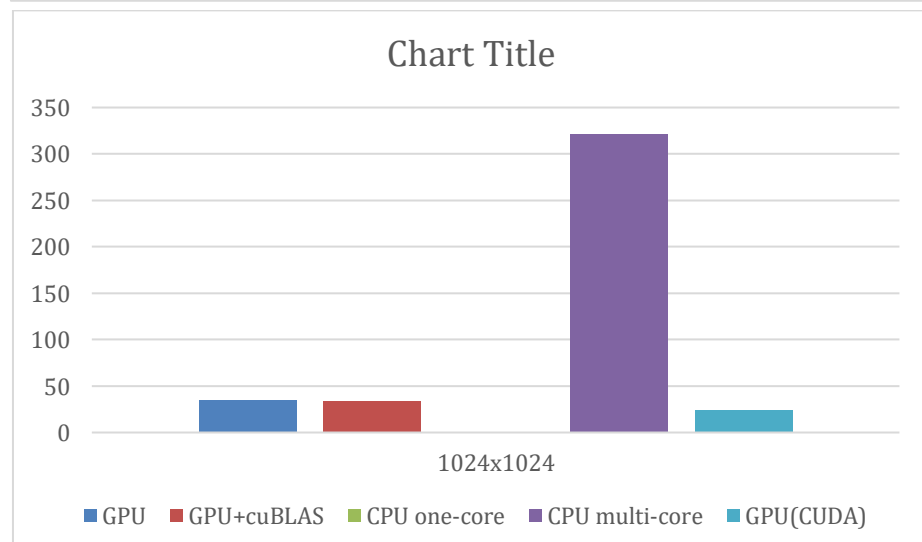
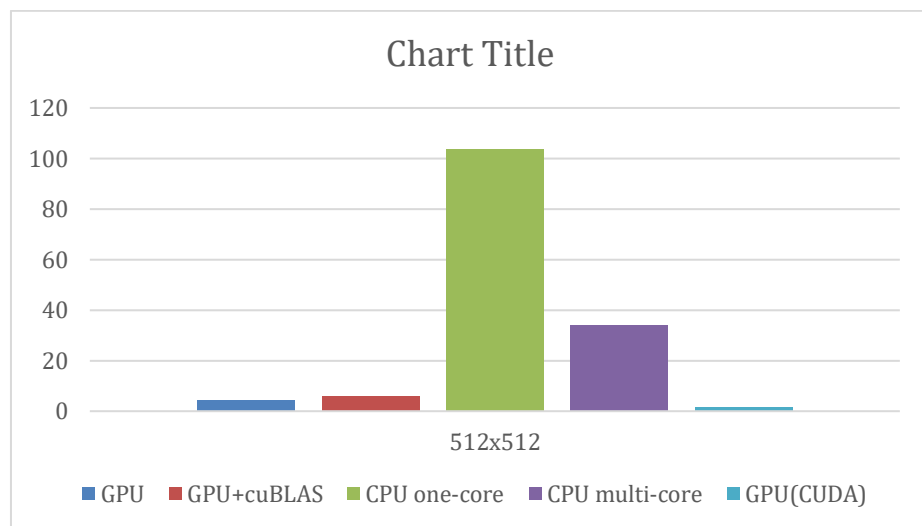
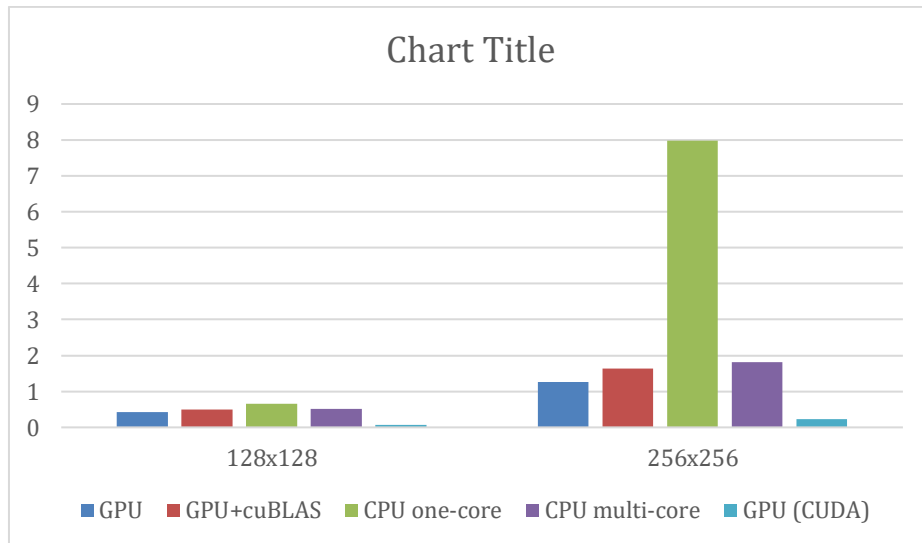


Профилирование на сетке 1024x1024



Вывод профилирования: Вычисления занимают примерно 86% от общей работы GPU, также благодаря использованию cudaGraph мы вызываем только его, а не каждую операцию отдельно, что также добавляет прирост оптимизации.

Диаграмма **CPU one-core**, **CPU multi-core**, **GPU** и **GPU+cuBLAS**,
GPU(CUDA)



10x10 (CUDA)

```
10 11.1111 12.2222 13.3333 14.4444 15.5556 16.6667 17.7778 18.8889 20
11.1111 12.2222 13.3333 14.4444 15.5556 16.6667 17.7778 18.8889 20 21.1111
12.2222 13.3333 14.4444 15.5556 16.6667 17.7778 18.8889 20 21.1111 22.2222
13.3333 14.4444 15.5556 16.6667 17.7778 18.8889 20 21.1111 22.2222 23.3333
14.4444 15.5556 16.6667 17.7778 18.8889 20 21.1111 22.2222 23.3333 24.4444
15.5556 16.6667 17.7778 18.8889 20 21.1111 22.2222 23.3333 24.4444 25.5556
16.6667 17.7778 18.8889 20 21.1111 22.2222 23.3333 24.4444 25.5556 26.6667
17.7778 18.8889 20 21.1111 22.2222 23.3333 24.4444 25.5556 26.6667 27.7778
18.8889 20 21.1111 22.2222 23.3333 24.4444 25.5556 26.6667 27.7778 28.8889
20 21.1111 22.2222 23.3333 24.4444 25.5556 26.6667 27.7778 28.8889 30
```

13x13 (CUDA)

```
10 10.8333 11.6667 12.5 13.3333 14.1667 15 15.8333 16.6667 17.5 18.3333 19.1667 20
10.8333 11.6667 12.5 13.3333 14.1667 15 15.8333 16.6667 17.5 18.3333 19.1667 20 20.8333
11.6667 12.5 13.3333 14.1667 15 15.8333 16.6667 17.5 18.3333 19.1667 20 20.8333 21.6667
12.5 13.3333 14.1667 15 15.8333 16.6667 17.5 18.3333 19.1667 20 20.8333 21.6667 22.5
13.3333 14.1667 15 15.8333 16.6667 17.5 18.3333 19.1667 20 20.8333 21.6667 22.5 23.3333
14.1667 15 15.8333 16.6667 17.5 18.3333 19.1667 20 20.8333 21.6667 22.5 23.3333 24.1667
15 15.8333 16.6667 17.5 18.3333 19.1667 20 20.8333 21.6667 22.5 23.3333 24.1667 25
15.8333 16.6667 17.5 18.3333 19.1667 20 20.8333 21.6667 22.5 23.3333 24.1667 25 25.8333
16.6667 17.5 18.3333 19.1667 20 20.8333 21.6667 22.5 23.3333 24.1667 25 25.8333 26.6667
17.5 18.3333 19.1667 20 20.8333 21.6667 22.5 23.3333 24.1667 25 25.8333 26.6667 27.5
18.3333 19.1667 20 20.8333 21.6667 22.5 23.3333 24.1667 25 25.8333 26.6667 27.5 28.3333
19.1667 20 20.8333 21.6667 22.5 23.3333 24.1667 25 25.8333 26.6667 27.5 28.3333 29.1667
20 20.8333 21.6667 22.5 23.3333 24.1667 25 25.8333 26.6667 27.5 28.3333 29.1667 30
```

Вывод CUDA: Если выполнять операцию редукции (вычисление максимального значения ошибки) на графическом процессоре реализовать через вызовы функций из библиотеки CUB на уровне блока и применить CUDA graph для итераций между подсчетами ошибки, дает очень большое время уменьшение время выполнения программы на каждом размере сетке из протестированных.