Теория Параллелизма

Отчет

Уравнение теплопроводности

21933, Круковский Василий Сергеевич
30.04.23

Цель работы:
Реализация приближённого вычисления уравнения теплопроводности при заданных начальных условиях.
Произвести вычисления на CPU и GPU и сравнить.

Используемый компилятор:
pgc++

Используемый профилировщик:
nsys (NVIDIA Night System) с OpenACC trace

Как производился замер времени работы:
Замер производился с использованием библиотеки chrono

Результат выполнения на CPU:

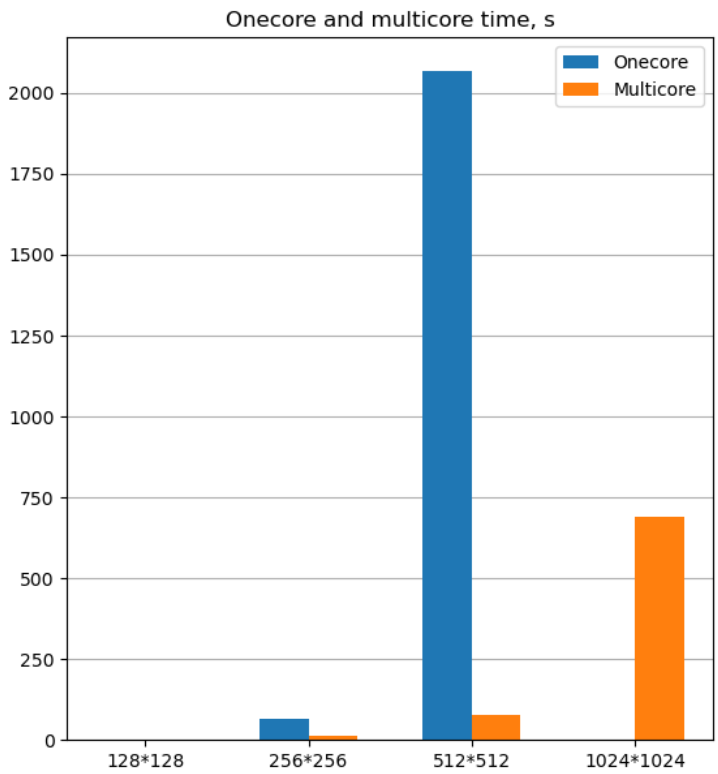
1.1 Onecore

Размер сетки	Время выполнения, сек	Точность	Количество итераций
128*128	2.62	9.98e-07	30080
256*256	64.21	9.98e-07	102912
512*512	2067.725	9.93e-07	339968

1.2 Multicore

Размер сетки	Время выполнения	Точность	Количество итераций
128*128	1.5	9.98e-07	30080
256*256	12.8	9.98e-07	102912
512*512	76.17	9.93e-07	339968
1024*1024	689.576	1.37e-06	1000000

2.1 Диаграмма сравнения выполнения на CPU



Результат выполнения на GPU:

Размер сетки	Время выполнения, сек	Точность	Количество итераций
128*128	0.2	9.98e-07	30080
256*256	0.72	9.98e-07	102912
512*512	6.3	9.93e-07	339968
1024*1024	66.5	1.37e-06	1000000

Этапы оптимизации программы:

Этап 0:
Результат:

Размер сетки	Время выполнения, сек	Точность	Количество итераций
128*128	0.448	0.104	100
256*256	1.132	0.106	100
512*512	4.167	0.107	100
1024*1024	25.235	0.107	100

Этап 1:
Вместо использования kernels, проанализировать места распараллеливания самостоятельно и использовать эффективные директивы: parallel, collapse, present, independent, reduction.
Результат:

Размер сетки	Время выполнения, сек	Точность	Количество итераций
128*128	0.229	0.104	100
256*256	0.219	0.106	100
512*512	0.217	0.107	100
1024*1024	0.209	0.107	100

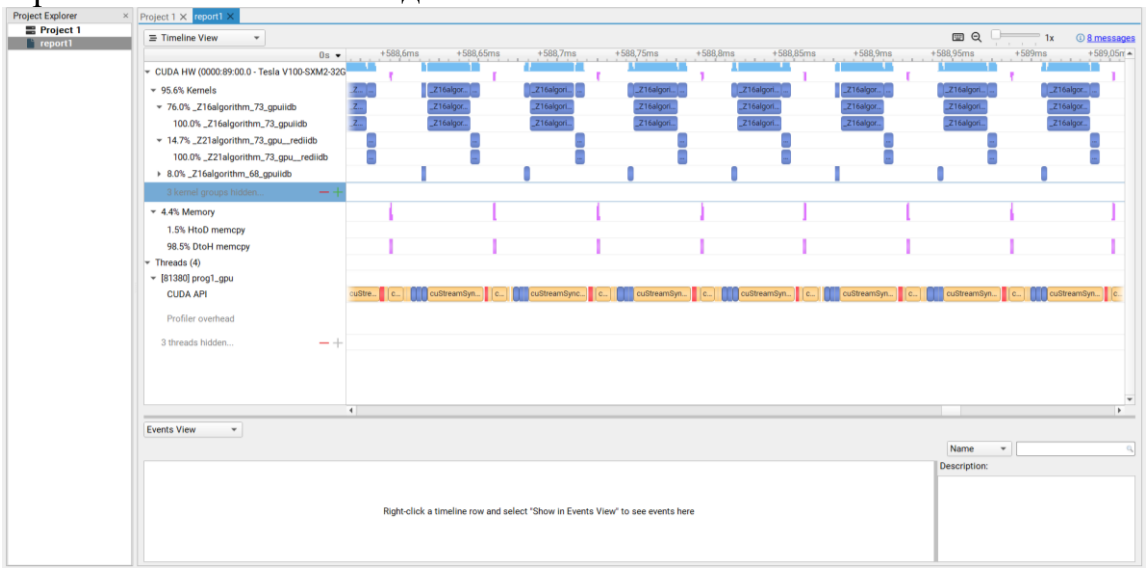
Этап 2:
Делать проверку на достижение необходимой точности не каждую итерацию, а, например, один раз за количество итераций равной размеру сетки. Размер сетки не должен превосходить количество итераций.

Результат:

Размер сетки	Время выполнения, сек	Точность	Количество итераций
128*128	0.225	0	100
256*256	0.220	0	100

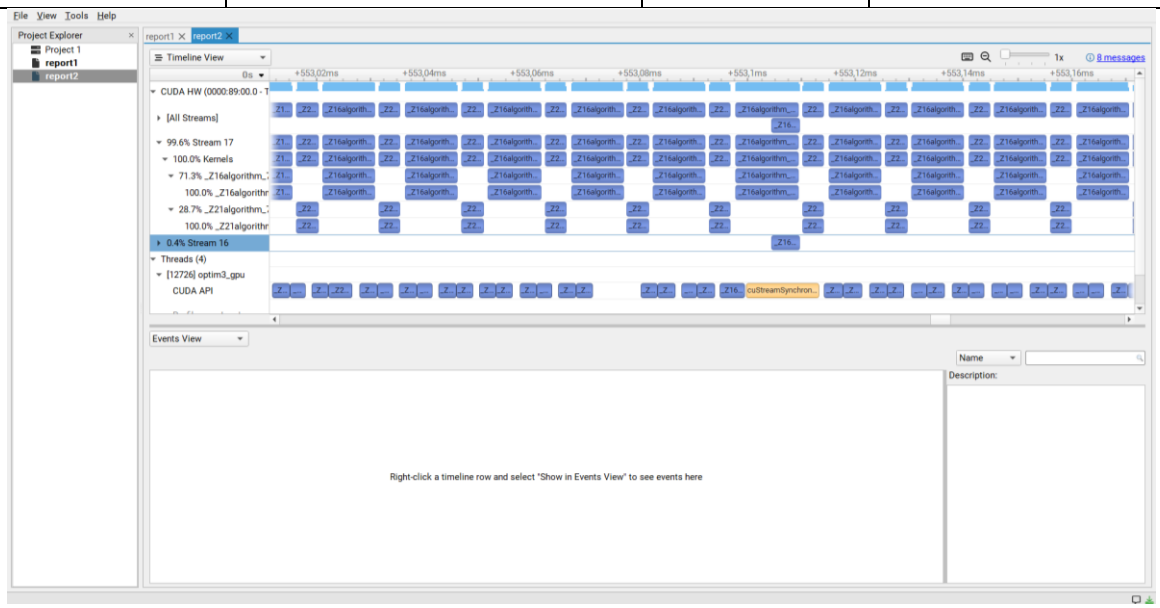
512*512	0.218	0	100
1024*1024	0.201	0	100

Этап 3:
Можно заметить в профилировщике, что некоторое время уходит на попытку синхронизации. Поэтому, можно разсинхронизировать основной цикл. Дополнительно, в случае загруженного устройства GPU, лучше переключиться на “свободное”.



Результат:

Размер сетки	Время выполнения, сек	Точность	Количество итераций
128*128	0. 214	0	100
256*256	0.208	0	100
512*512	0.221	0	100
1024*1024	0.201	0	100

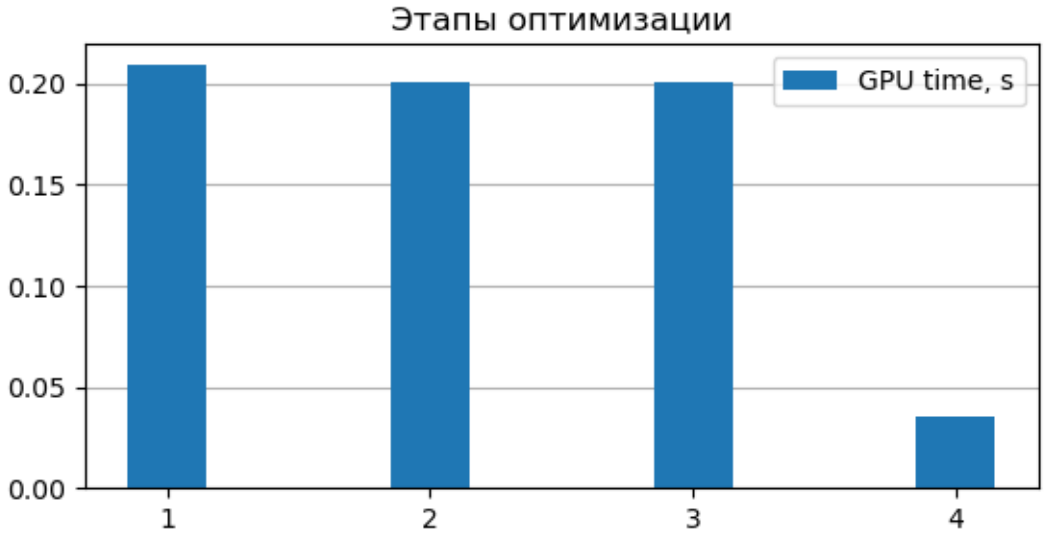


Этап 4:
Можно предположить, что скорость программы может увеличиться, если создавать массив не на CPU а сразу на GPU, для небольших размеров матрицы, это может дать прирост в скорости.
Также, вместо того, чтобы считать ошибку параллельно с элементом матрицы, можно считать ошибку отдельно в другом цикле, после полного просчёта матрицы.
Результат:

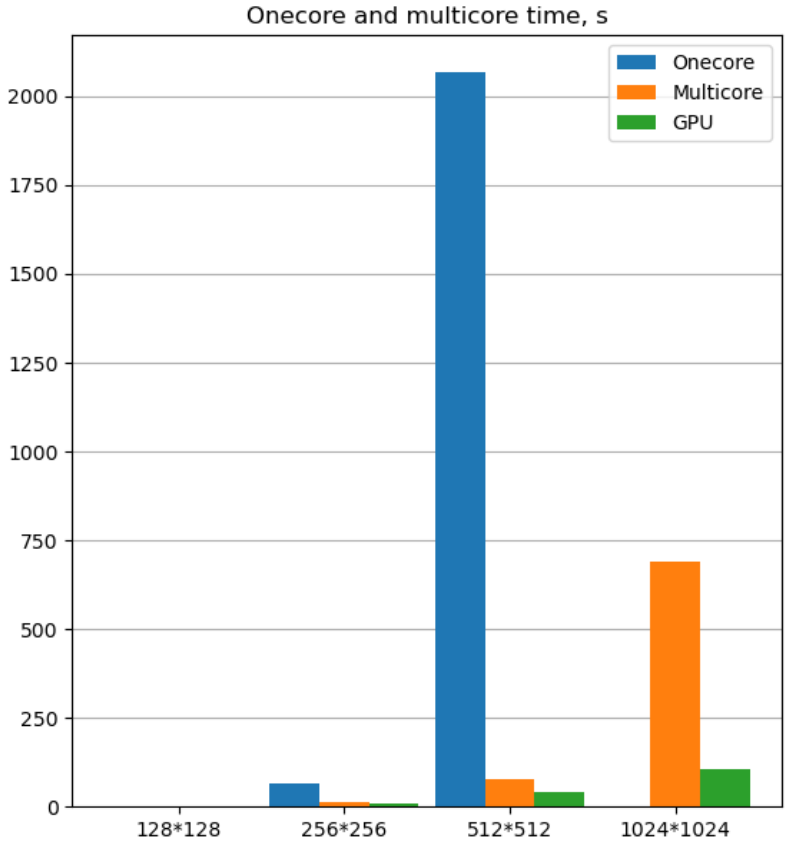
Размер сетки	Время выполнения, сек	Точность	Количество итераций
128*128	0.025	0	100
256*256	0.027	0	100
512*512	0.027	0	100
1024*1024	0.035	0	100

P.S. Точность равна нулю из-за того, что расчёт ошибки происходит, когда итерация кратна числу сетки. Минимальный размер сетки 128, а количество доступных итераций 100.

Вывод:
Наилучшие результаты для оптимизации это:
Расчёт матрицы на GPU, продуманный подход к выбору прагм acc а также расчёт ошибки не на каждой итерации.



Сравнение скорости выполнения программ на CPU и оптимизированного под GPU:



Вывод:

При правильном использовании инструментария OpenAcc, можно ускорить код содержащий циклы в разы, особенно для ускорителей, типа GPU.

Приложение:

Ссылка на GitHub: *https://github.com/MegaSear/paralellism_therm*