Теория параллелизма

Отчет

Решение уравнения теплопроводности с использованием метода Якоби

Выполнила Новикова Полина Павловна, гр. 23931

20.05.2025

Цель работы: 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 1006. Реализация должна быть на C++ с использованием OpenACC для переноса на GPU. Необходимо сравнить производительность на CPU и GPU, провести профилирование и оптимизацию кода.

Компиляторы:
nvc++ 23.11-0

Визуализатор параллельного кода:
NVIDIA Nsight Systems

Инструмент для измерения времени работы:
chrono.

Выполнение на CPU (по факту добавления критических оптимизаций)

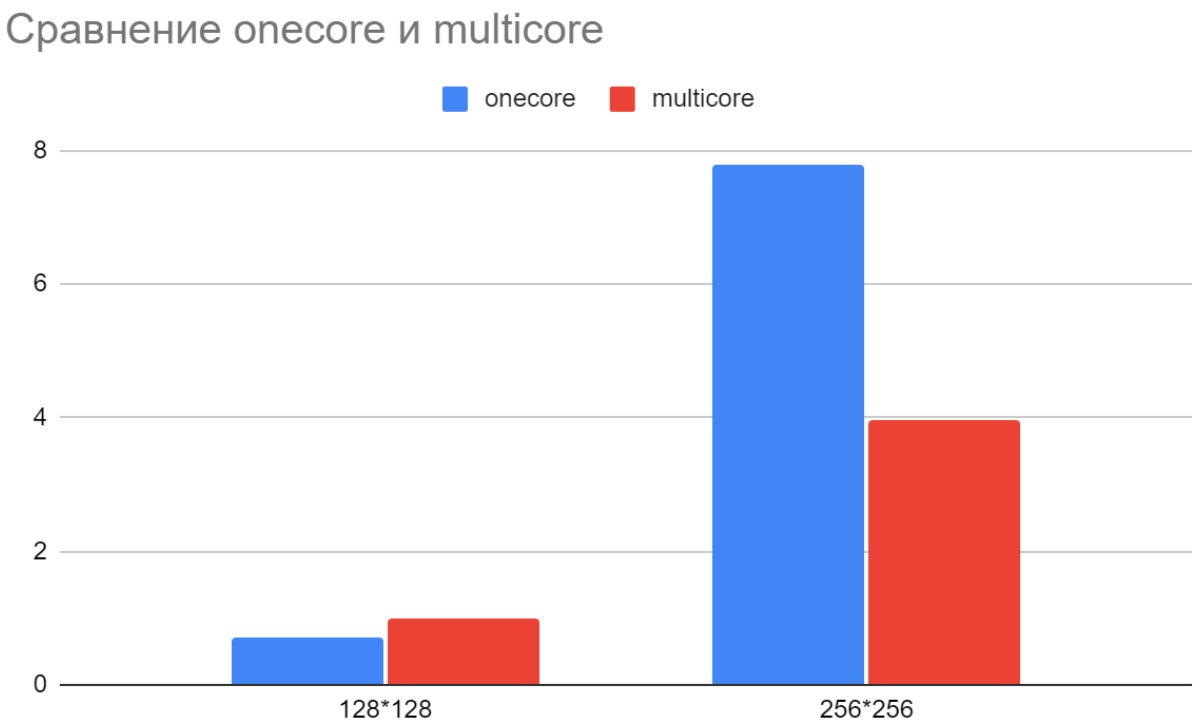
CPU-onecore

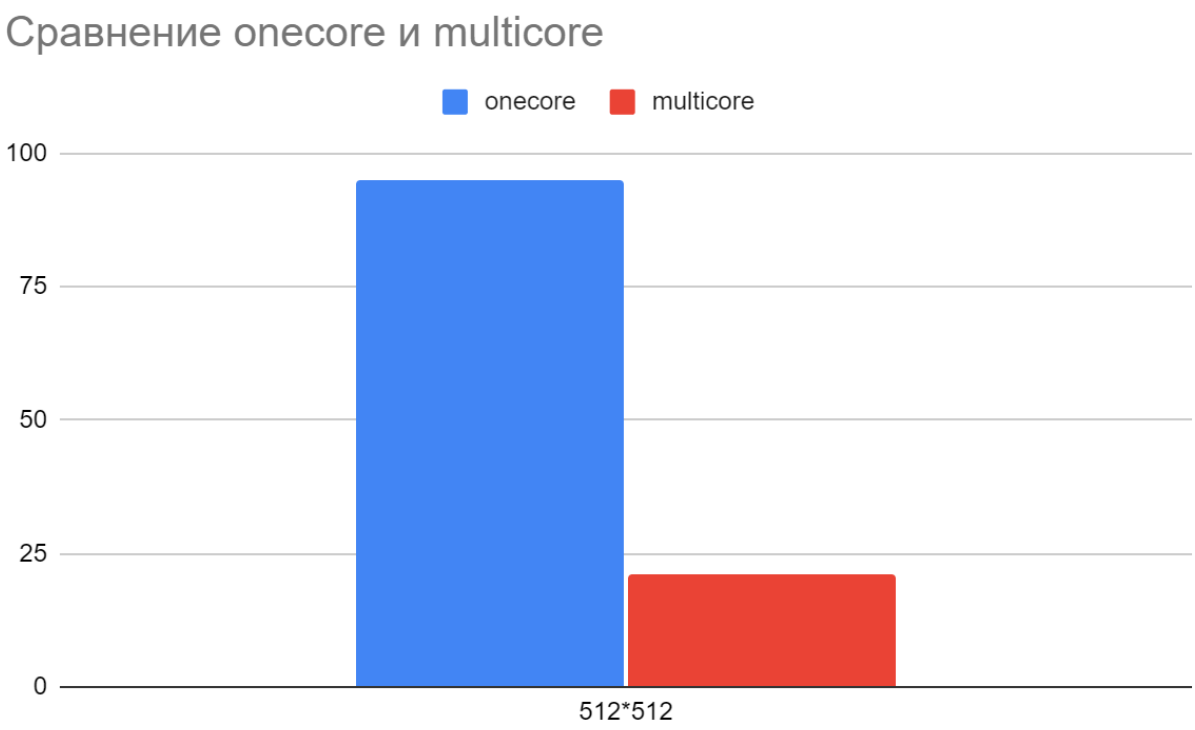
Размер сетки	Время выполнения, сек	Точность	Количество итераций
128*128	0.722	1e-6	40000
256*256	7.799	1e-6	110000
512*512	94.848	1e-6	340000
1024*1024	1273.686	1e-6	1000000

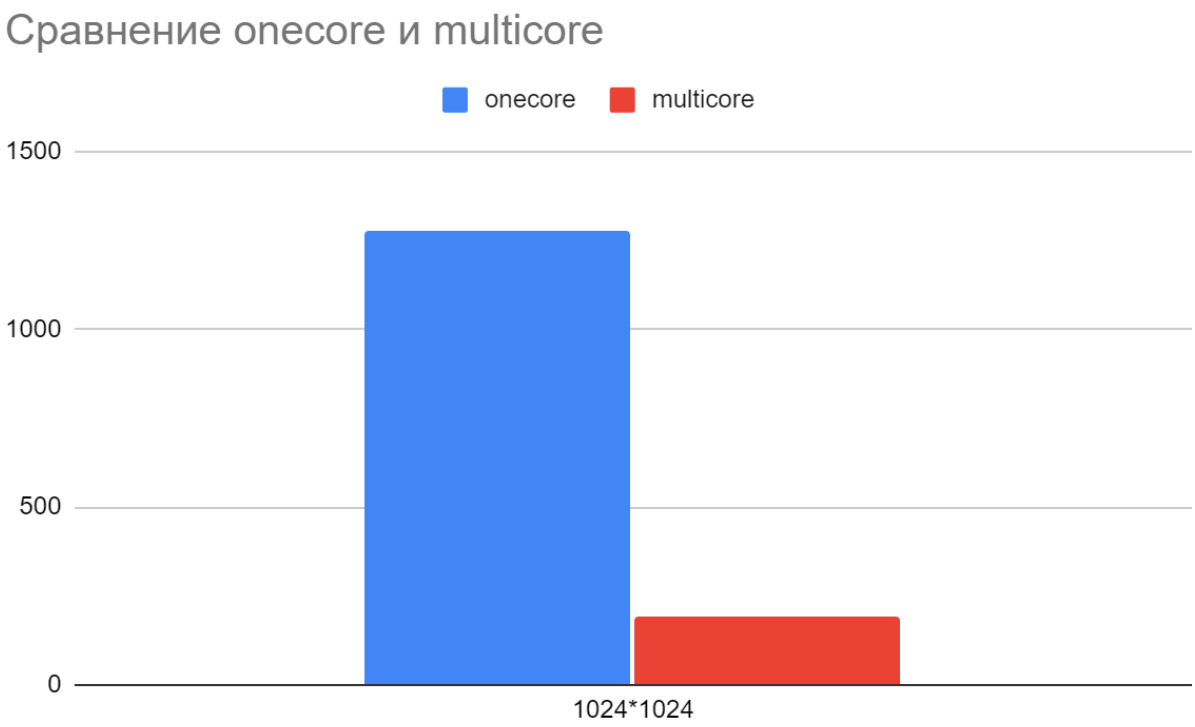
CPU-multicore

Размер сетки	Время выполнения, сек	Точность	Количество итераций
128*128	0.983	1e-6	40000
256*256	3.952	1e-6	110000
512*512	21.096	1e-6	3400000
1024*1024	194.531	1e-6	1000000

Диаграмма сравнения времени работы CPU-onecore и CPU-multicore




Сравнение onecore и multicore
onecore
multicore
100
75
50
25
0
512*512

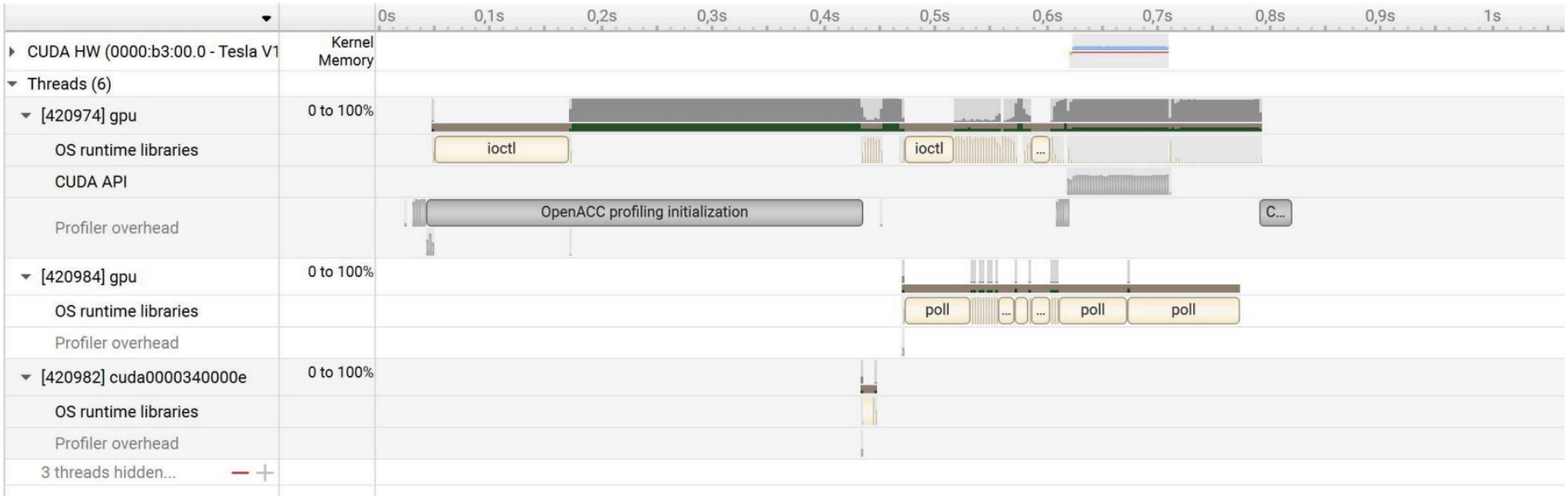

Сравнение onecore и multicore
onecore
multicore
1500
1000
500
0
1024*1024

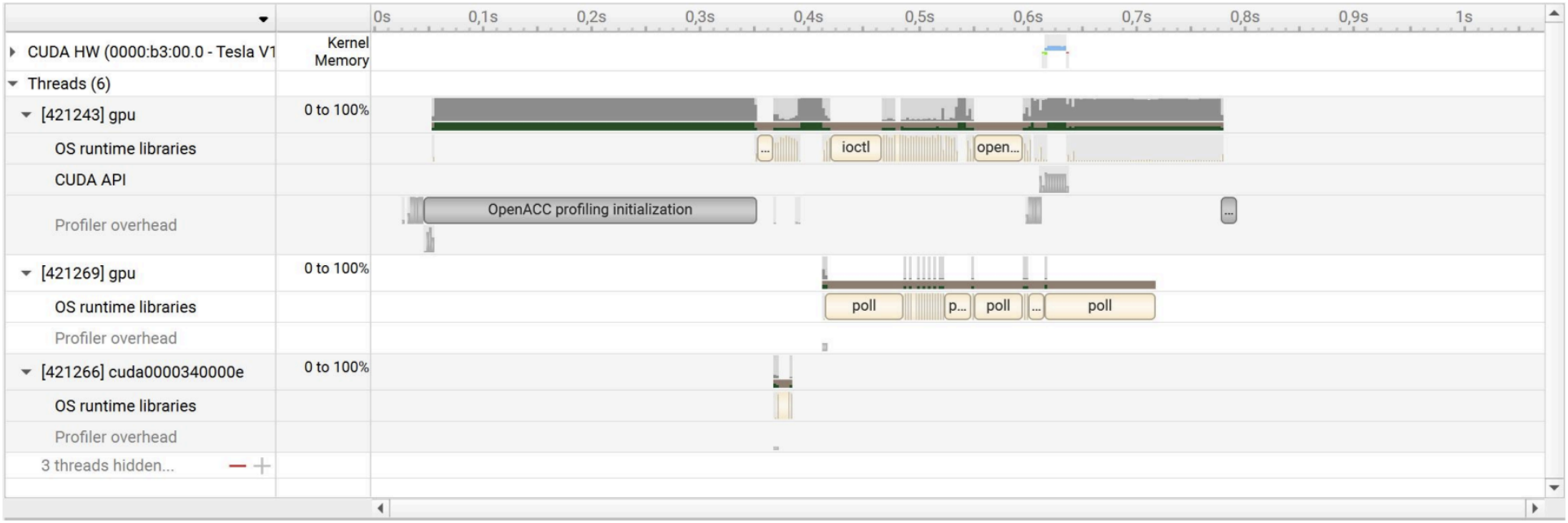
Выполнение на GPU

Измерения на матрице 1024*1024 с указанием оптимизаций для ускорения

№	Время выполнения, сек	Точность	Описание решения
1	98.902	1e-6	Без применения оптимизаций.
2	36.998	1e-6	Убран вывод промежуточного результата по окончании каждой итерации.
3	32.938	1e-6	Замена swap на temp через указатели.

Визуализация в nsys-ui (5000 итераций)




0s
0,1s
0,2s
0,3s
0,4s
0,5s
0,6s
0,7s
0,8s
0,9s
1s
CUDA HW (0000:b3:00.0 - Tesla V1
Kernel
Memory
Threads (6)
[421243] gpu
0 to 100%
OS runtime libraries
ioctl
open...
CUDA API
Profiler overhead
OpenACC profiling initialization
[421269] gpu
0 to 100%
OS runtime libraries
poll
p...
poll
poll
Profiler overhead
[421266] cuda0000340000e
0 to 100%
OS runtime libraries
Profiler overhead
3 threads hidden...

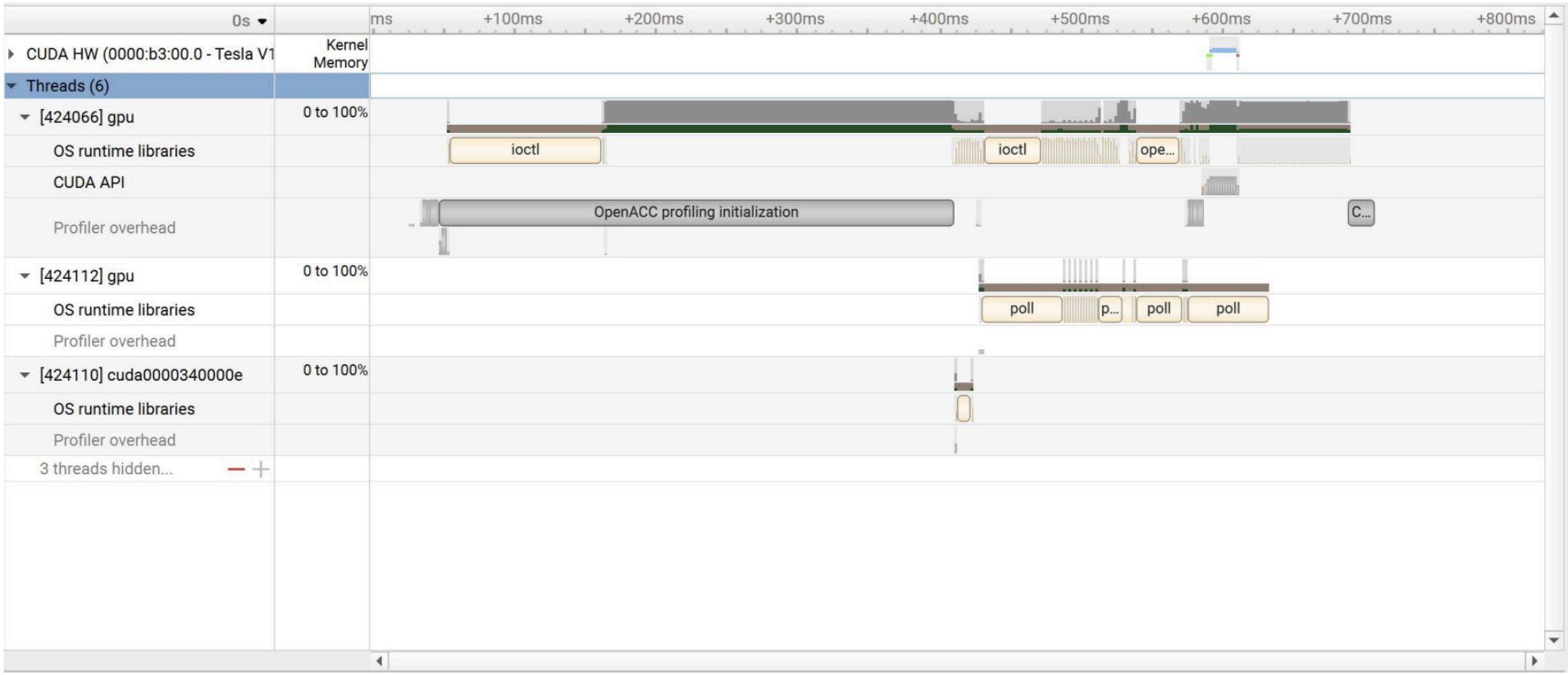
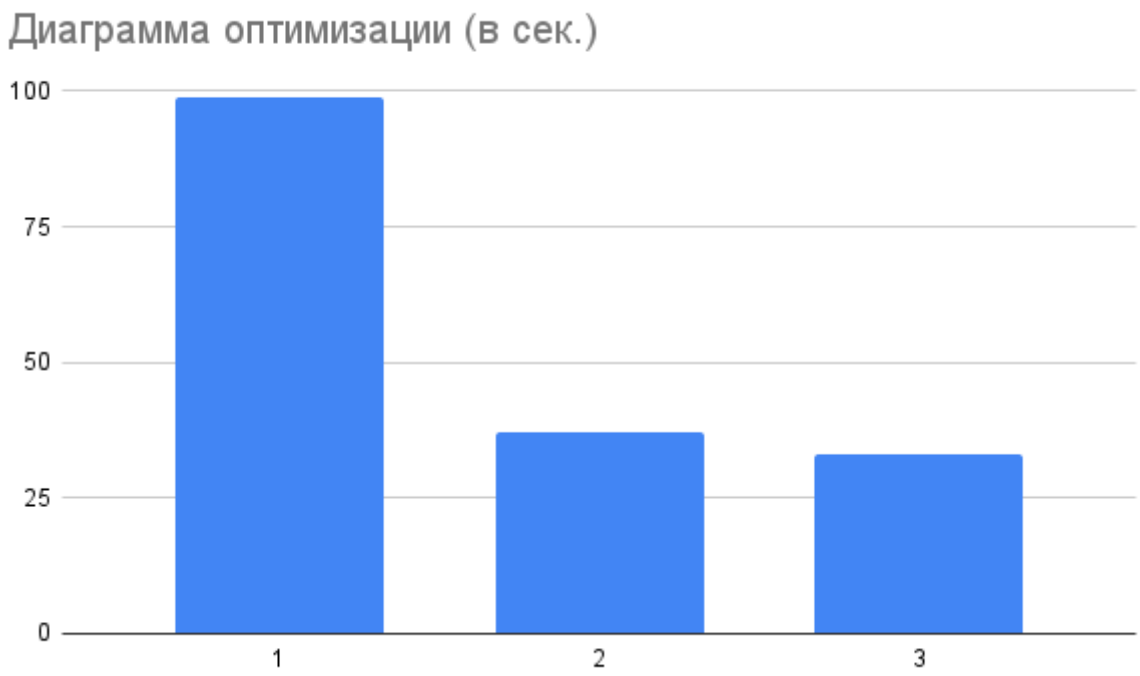

0s
ms
+100ms
+200ms
+300ms
+400ms
+500ms
+600ms
+700ms
+800ms
CUDA HW (0000:b3:00.0 - Tesla V1
Kernel
Memory
Threads (6)
[424066] gpu
0 to 100%
OS runtime libraries
ioctl
ioctl
ope...
CUDA API
Profiler overhead
OpenACC profiling initialization
C...
[424112] gpu
0 to 100%
OS runtime libraries
poll
p...
poll
poll
Profiler overhead
[424110] cuda0000340000e
0 to 100%
OS runtime libraries
Profiler overhead
3 threads hidden...

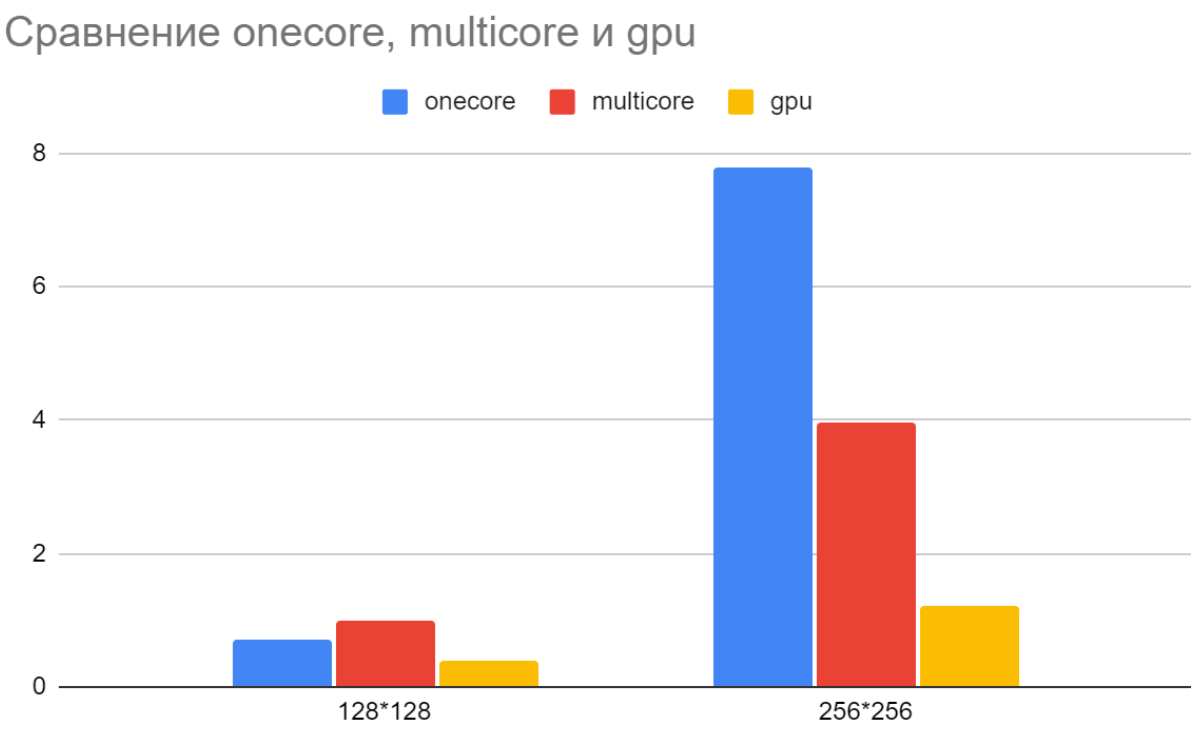
Диаграмма оптимизации

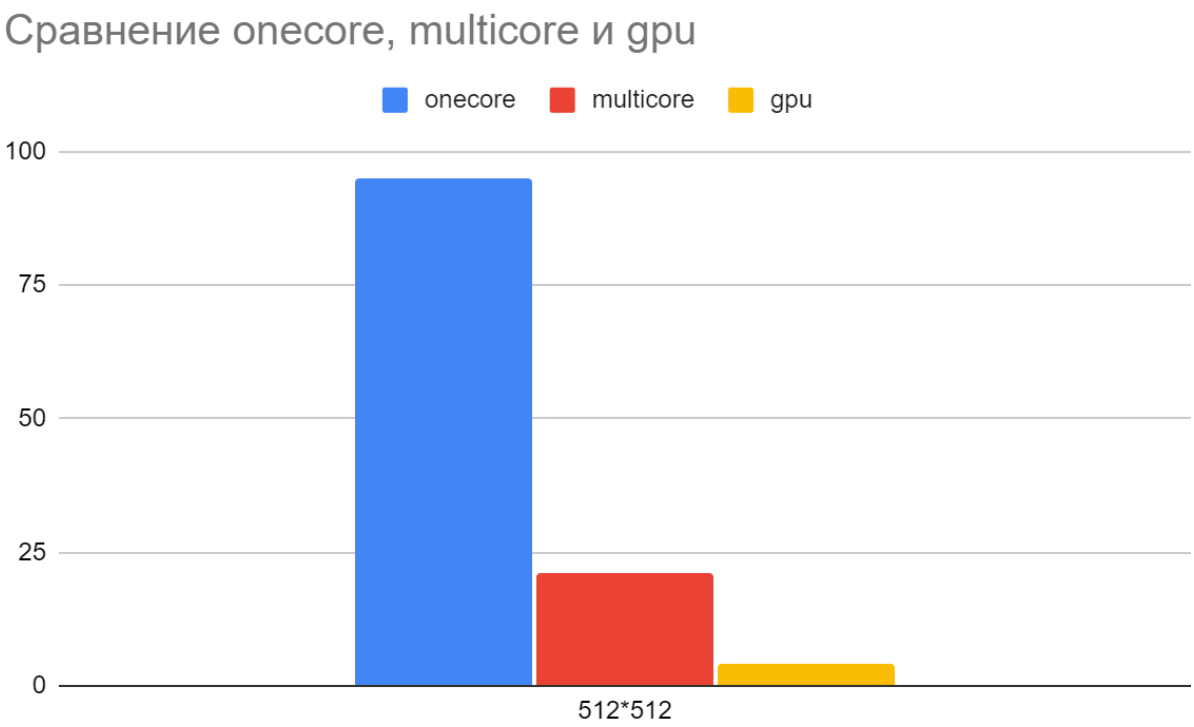


Измерения оптимизированного варианта на GPU

Размер сетки	Время выполнения, сек	Точность	Количество итераций
128*128	0.402	1e-6	40000
256*256	1.205	1e-6	110000
512*512	4.304	1e-6	340000
1024*1024	32.932	1e-6	1000000

Диаграмма сравнения времени работы CPU-onecore, CPU-multicore, GPU-optimized для разных размеров матриц


Сравнение onecore, multicore и gpu
onecore
multicore
gpu
8
6
4
2
0
128*128
256*256


Сравнение onecore, multicore и gpu
onecore
multicore
gpu
100
75
50
25
0
512*512

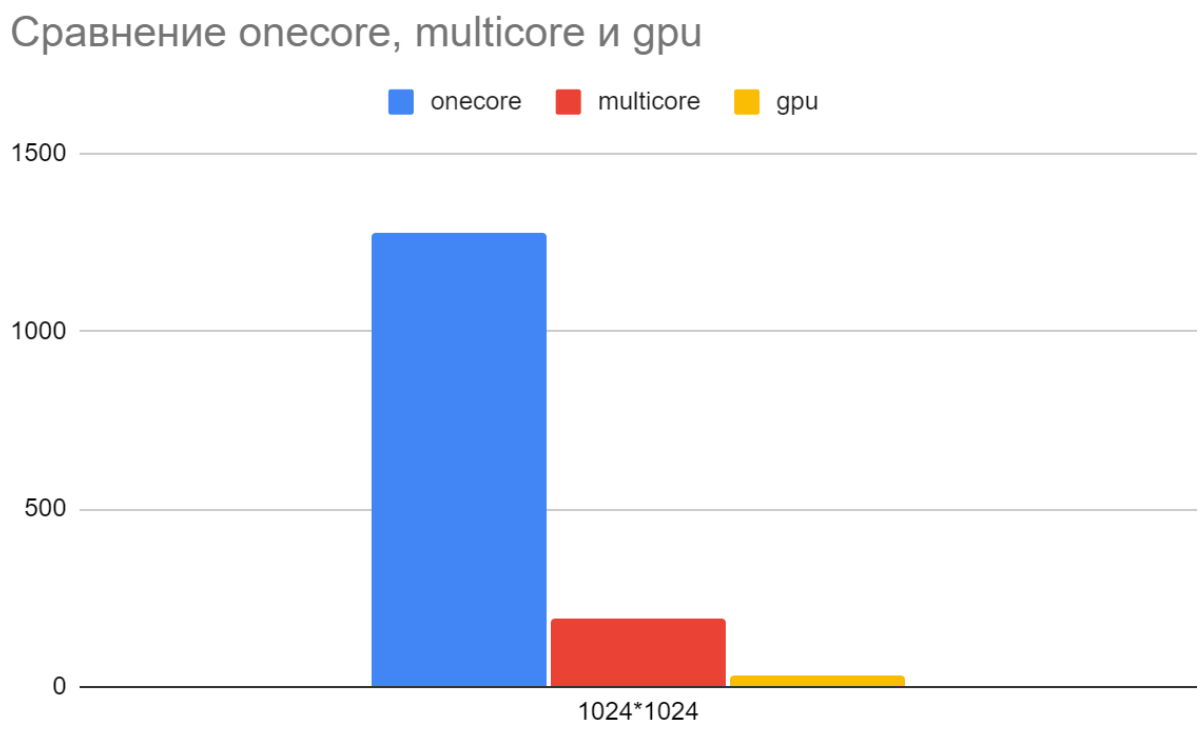
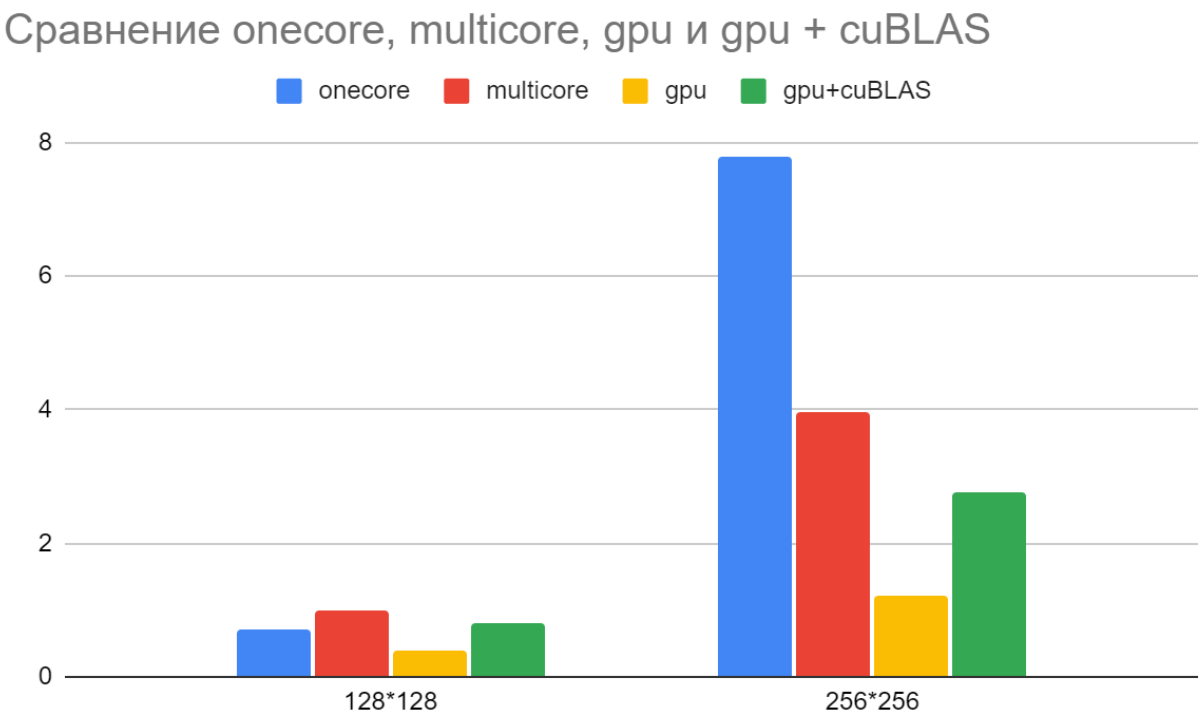
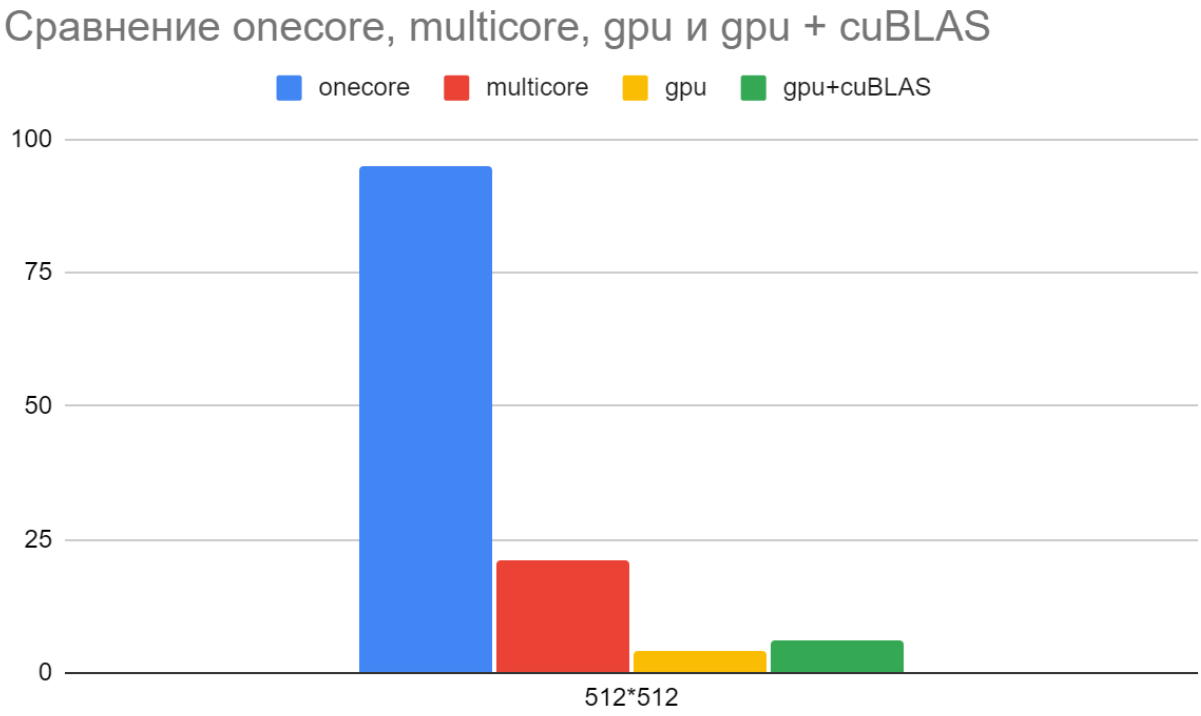
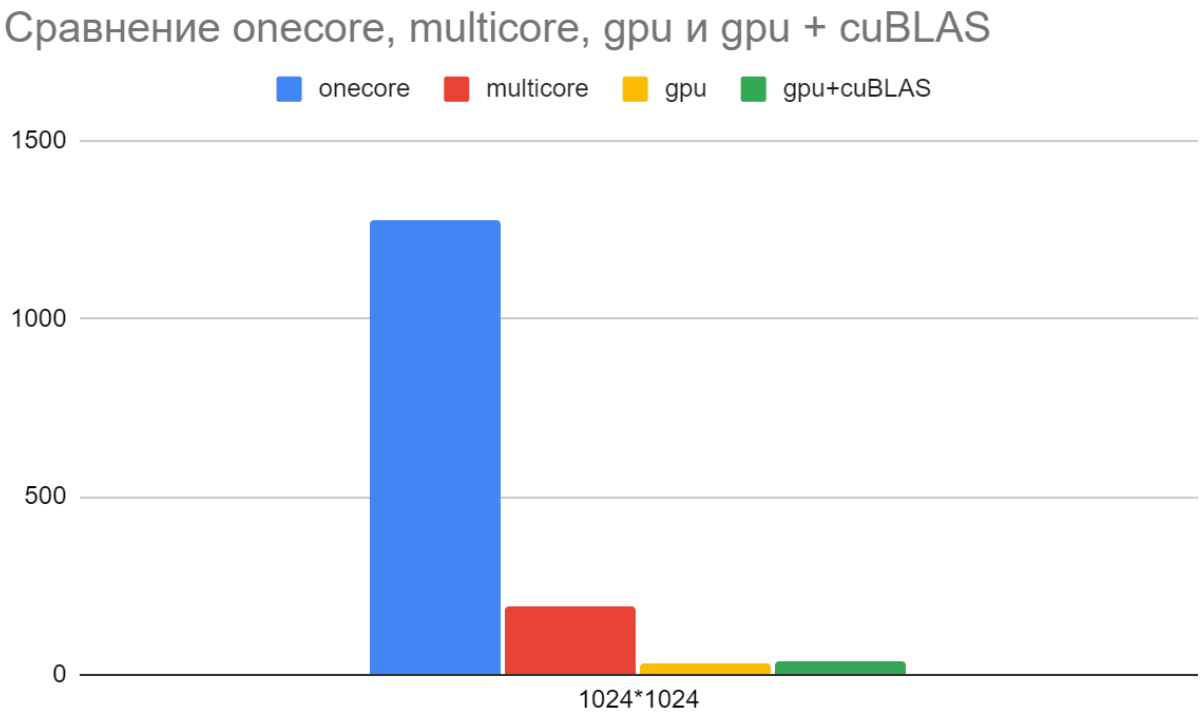


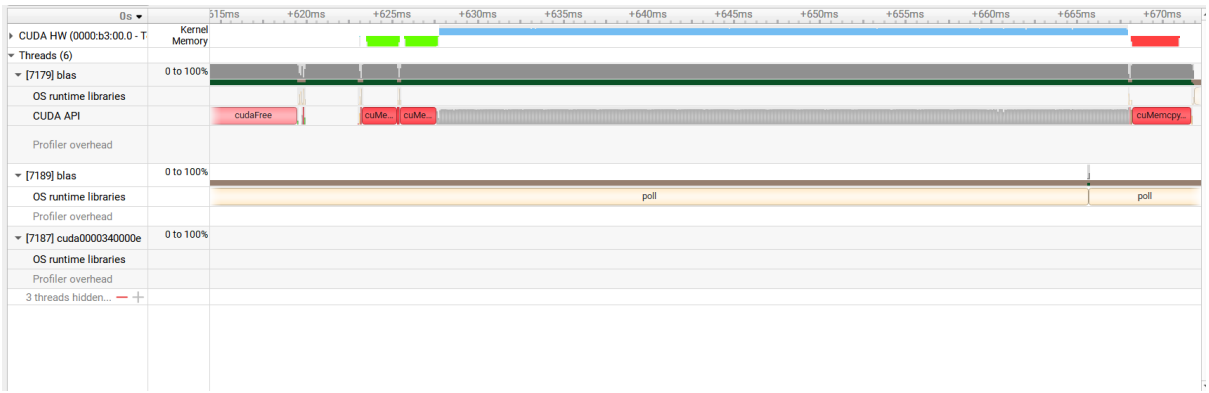
Диаграмма сравнения времени работы GPU + CUBLAS




Сравнение onecore, multicore, gpu и gpu + cuBLAS
onecore
multicore
gpu
gpu+cuBLAS
100
75
50
25
0
512*512


Сравнение onecore, multicore, gpu и gpu + cuBLAS
onecore
multicore
gpu
gpu+cuBLAS
1500
1000
500
0
1024*1024

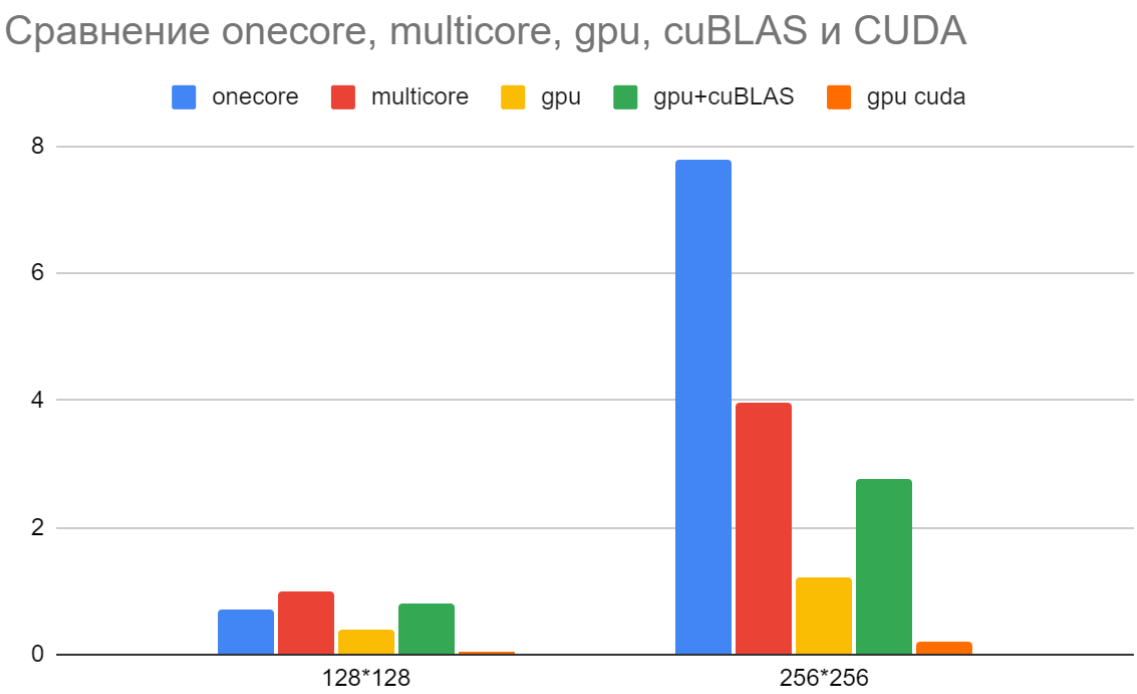
Размер сетки	Время выполнения, сек	Точность	Количество итераций
128*128	0.799	1e-6	40000
256*256	2.761	1e-6	110000
512*512	6.229	1e-6	340000
1024*1024	35.409	1e-6	1000000

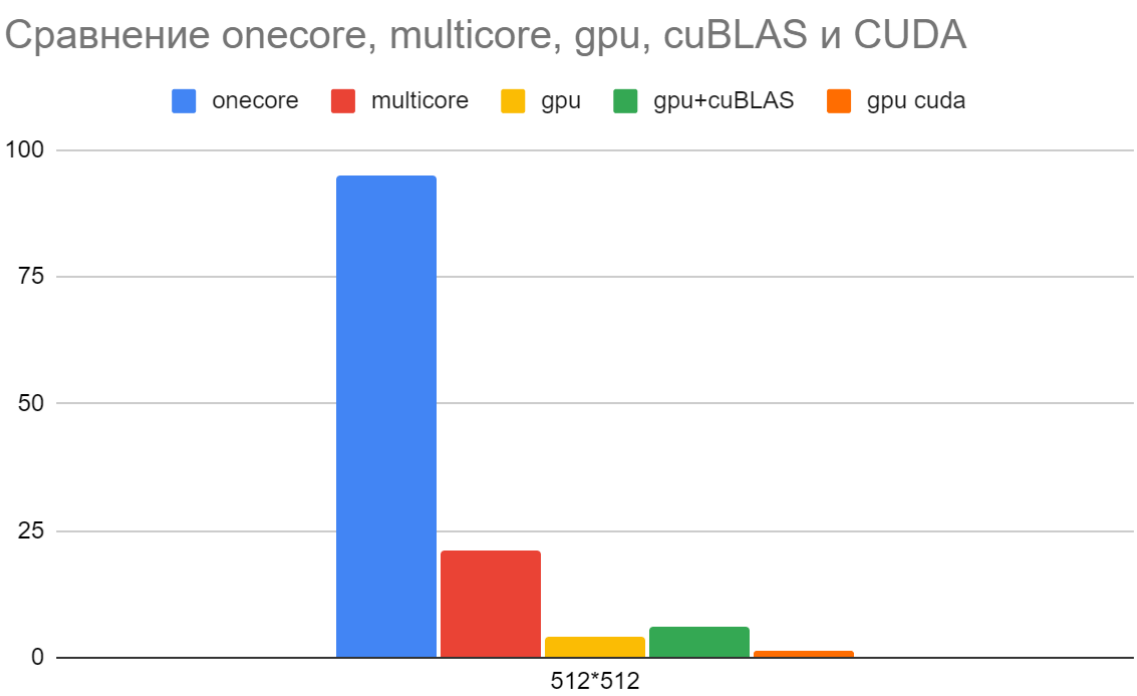


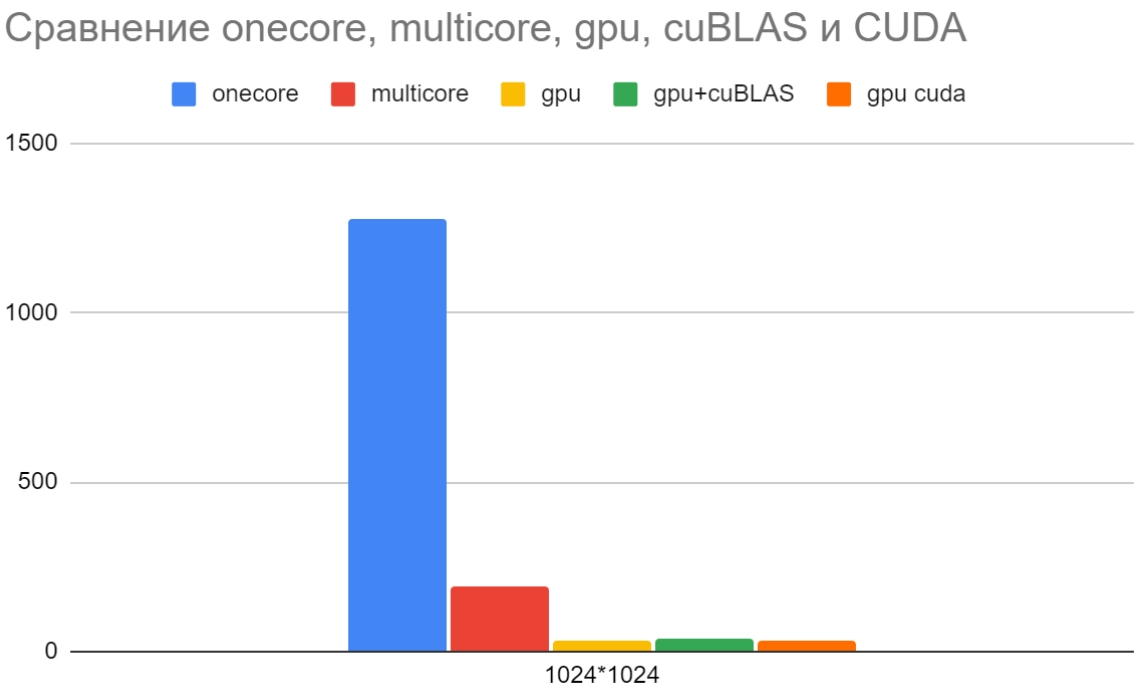
Сравнение времени выполнения на GPU (CUDA)

Размер сетки	Время выполнения, сек	Точность	Количество итераций
128*128	0.54	1e-6	31000
256*256	0.206	1e-6	103000
512*512	1.330	1e-6	340000
1024*1024	30.781	1e-6	1000000

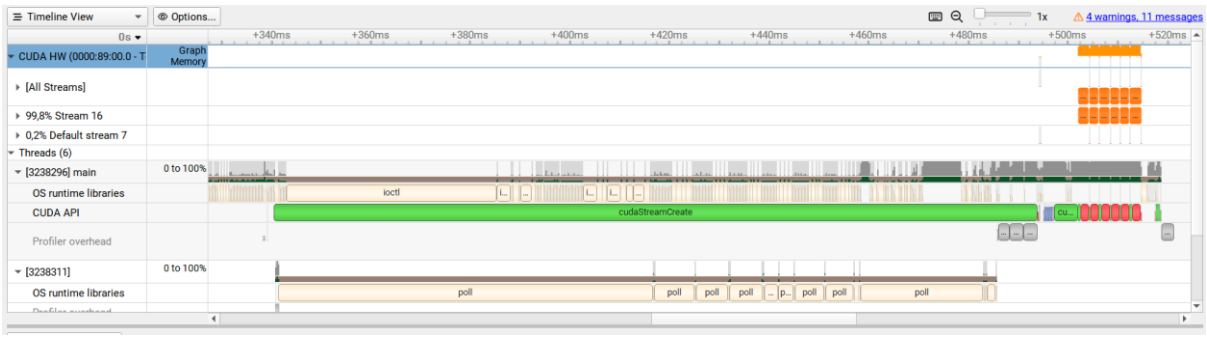
Диаграмма сравнения времени работы GPU + CUBLAS



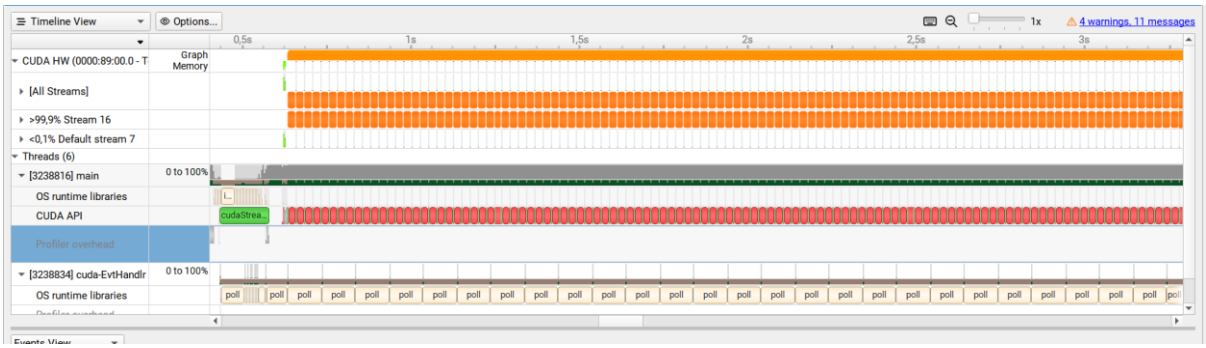

Сравнение onecore, multicore, gpu, cuBLAS и CUDA
onecore
multicore
gpu
gpu+cuBLAS
gpu cuda
100
75
50
25
0
512*512

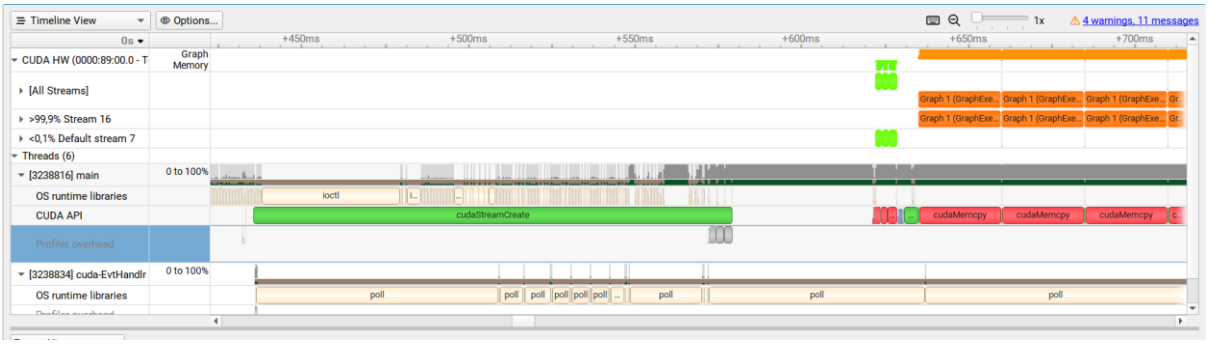

Сравнение onecore, multicore, gpu, cuBLAS и CUDA
onecore
multicore
gpu
gpu+cuBLAS
gpu cuda
1500
1000
500
0
1024*1024

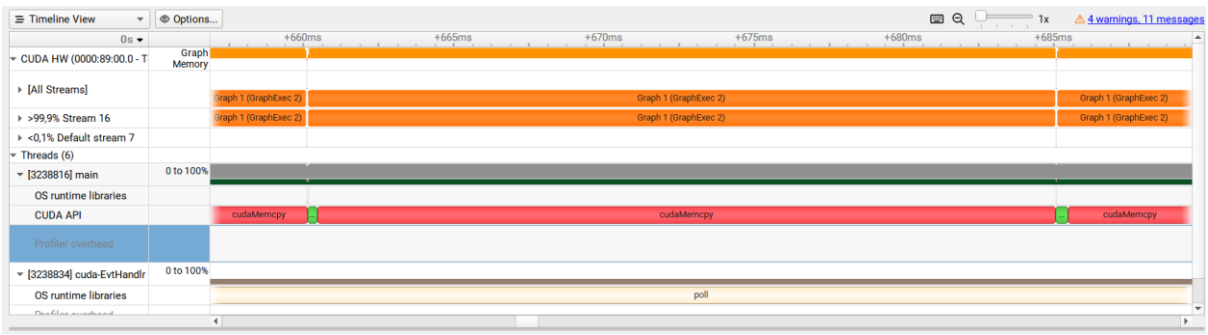
Сетка 50x50



Сетка 1024x1024







Вывод

Использование нескольких ядер CPU позволяет значительно увеличить производительность, в то время как на любых матрицах лучше всего себя показывает GPU. Разница в работе между программами , где ошибка ищется через директивы openACC или cuBLAS незначительна. Вычисления с использованием CUDA дают наилучший результат, превосходящий вычисления на GPU.