Теория параллелизма

Отчет

Лабораторная работа 7

Выполнил Кошелев Никита, 22931 27.05.24

Цель работы: Реализовать и оптимизировать решение уравнение теплопроводности (разностная схема –пятиточечный шаблон) в двумерной области на равномерных сетках с использованием GPU и директив OpenACC.

Используемый компилятор: pgc++

Используемый профилировщик: “Nsight Systems”.

Замер времени работы: Библиотека “chrono”

Выполнение на CPU

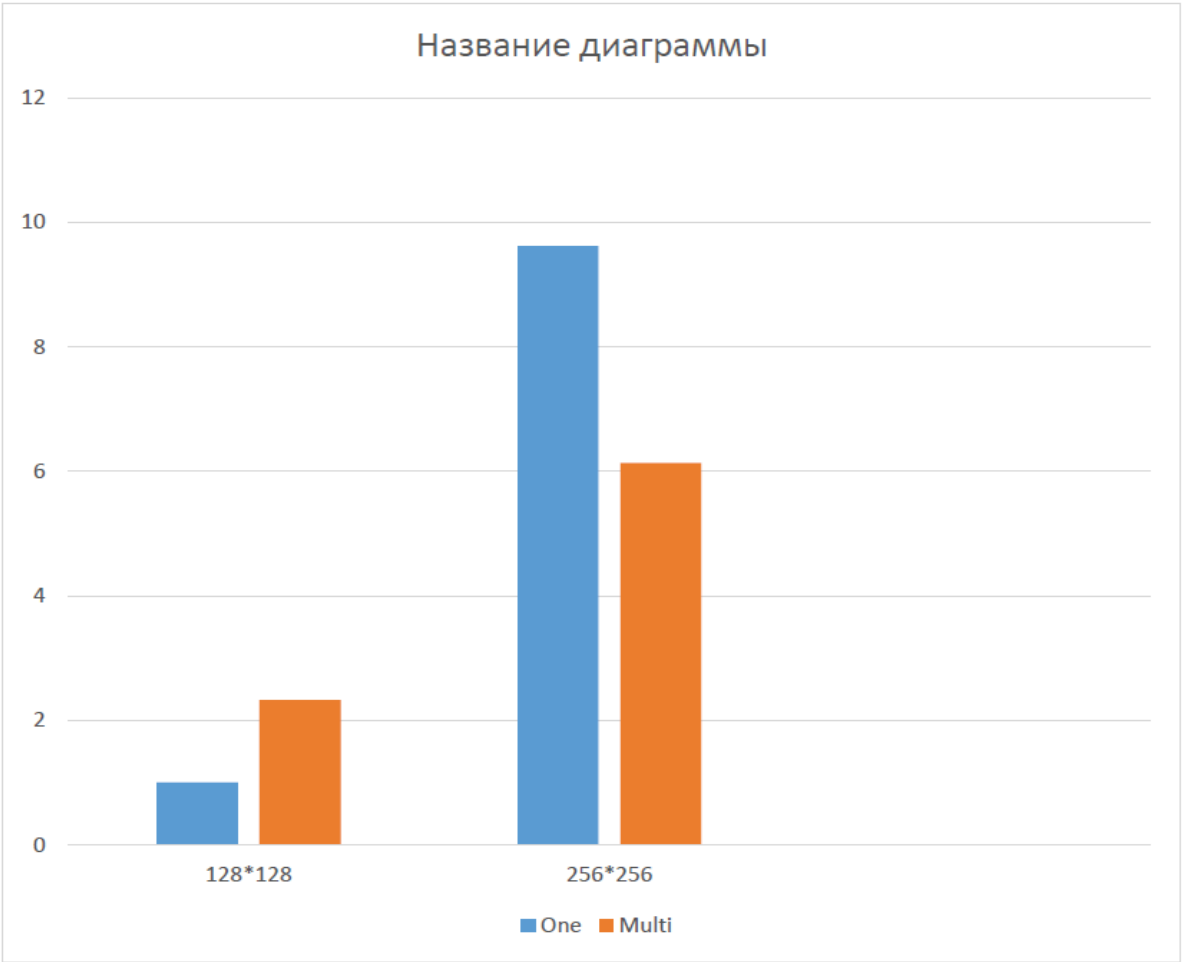
CPU-onecore

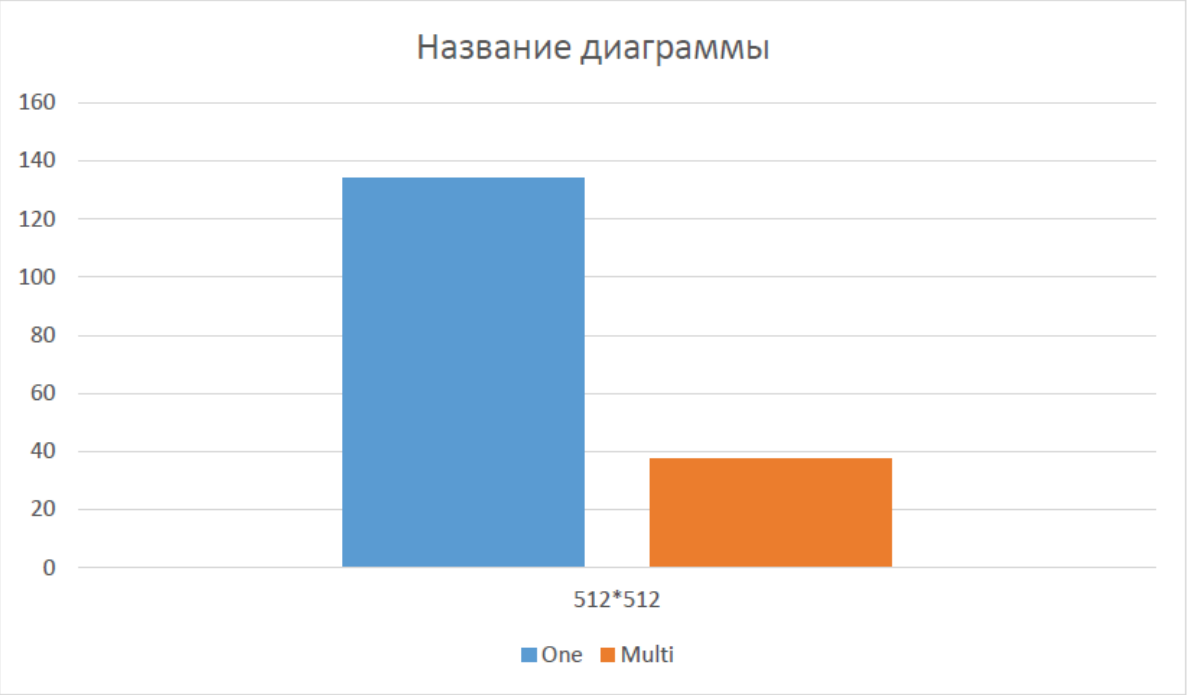
Размер сетки	Время выполнения	Точность	Количество операций
128*128	0.672911	0.000001	30101(30074)
256*256	9.620933	0.000001	102901(102885)
512*512	134.230674	0.000001	339601(339599)

CPU-multicore

Размер сетки	Время выполнения	Точность	Количество операций
128*128	2.329963	0.000001	30101(30074)
256*256	6.132657	0.000001	102901(102885)
512*512	37.022035	0.000001	339601(339599)
1024*1024	329.625234	>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	124.794780	>0.000001	1000000	Неоптимизированнный вариант
2	62.007211	>0.000001	1000000	Замена swap (swap через указатели)
3	38.979210	>0.000001	1000000	Возвращение ошибки каждые 100 операций

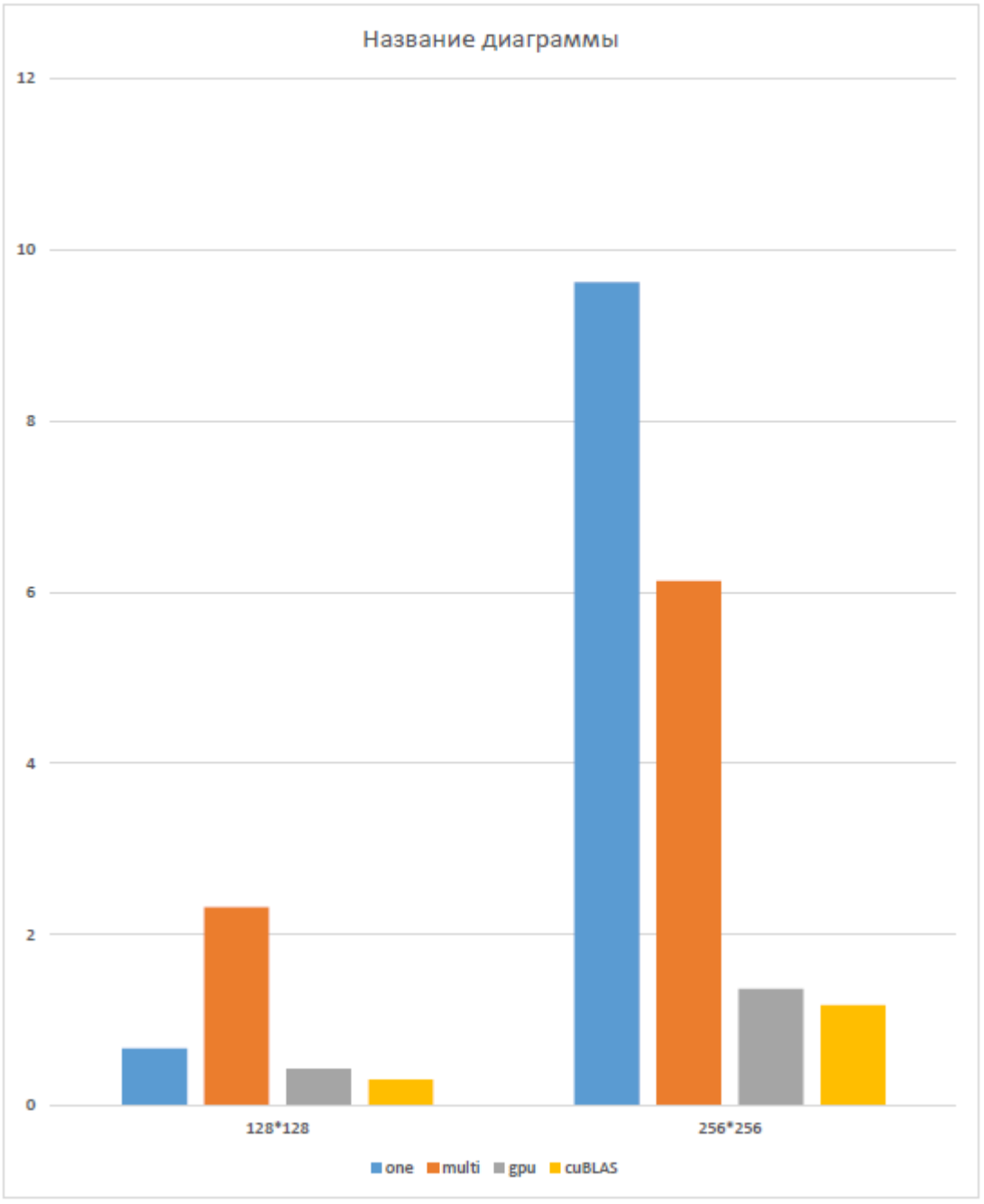
GPU - **оптимизированный** вариант

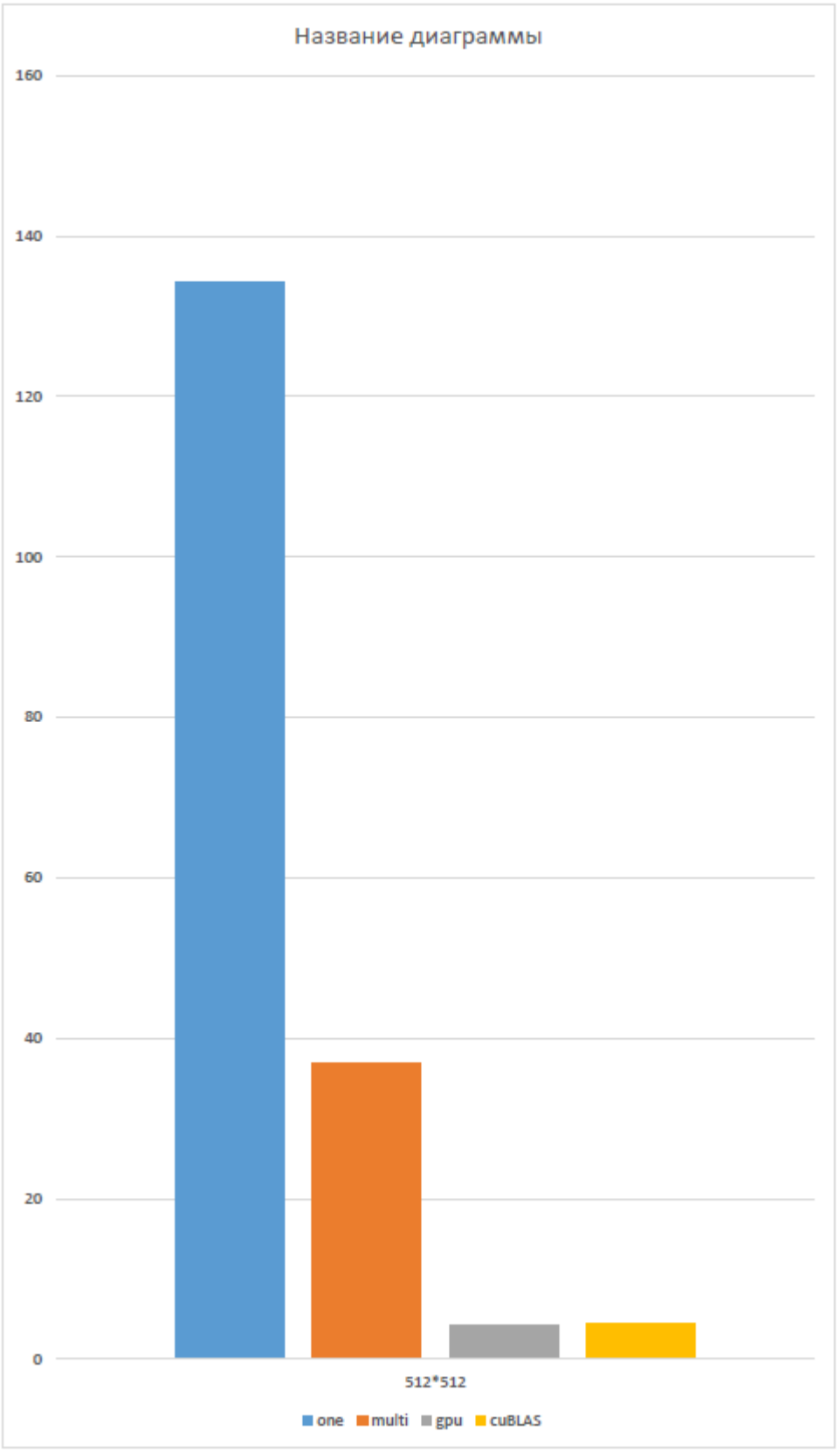
Размер сетки	Время выполнения(с)	Точность	Количество операций
128*128	0.434266	0.000001	30074(до возвращения ошибки раз в 100 раз) 300101
256*256	1.364479	0.000001	102885\102901
512*512	4.240394	0.000001	339599\339601
1024*1024	35.587202	>0.000001	1000000

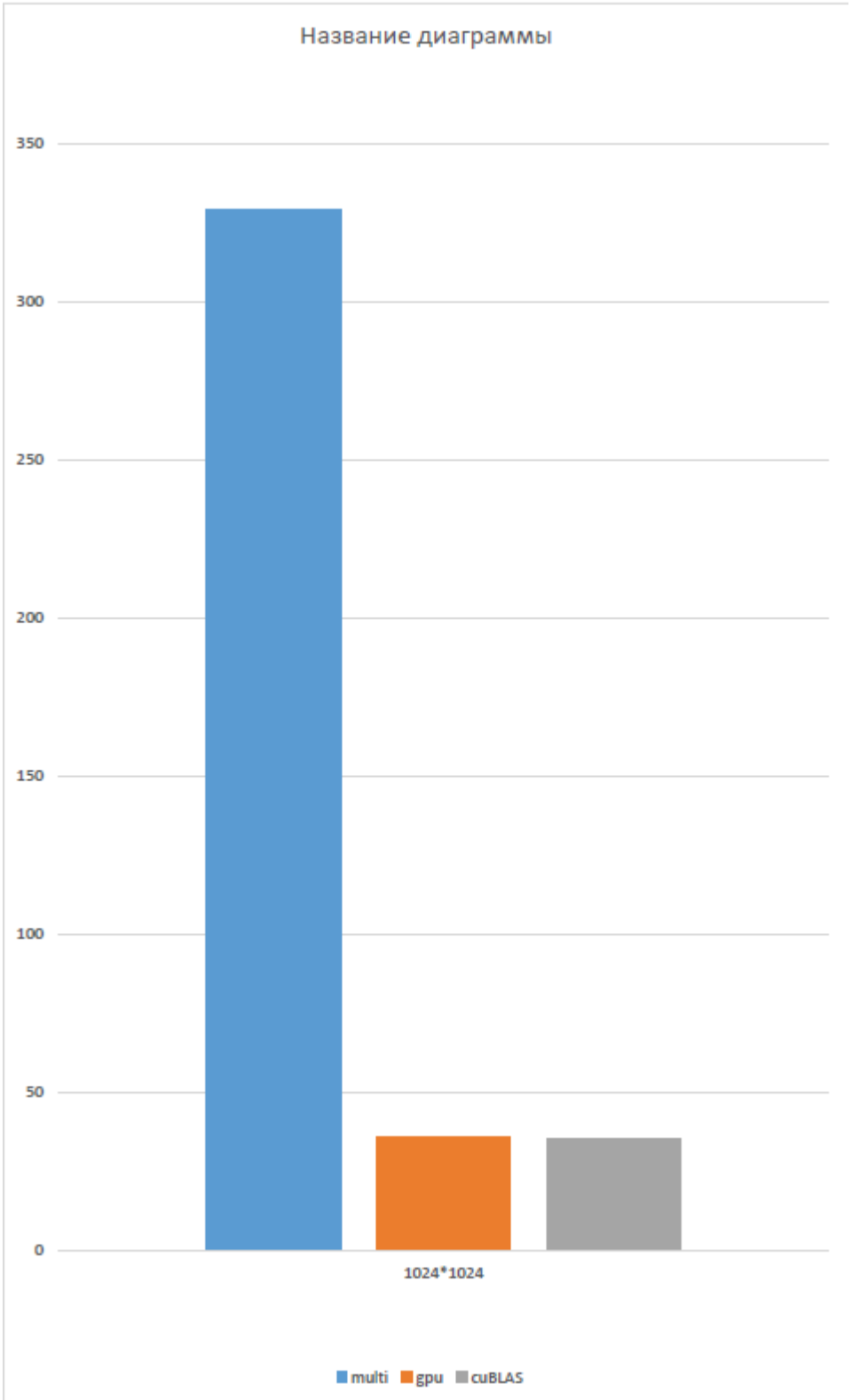
GPU - **оптимизированный** вариант+ cuBLAS

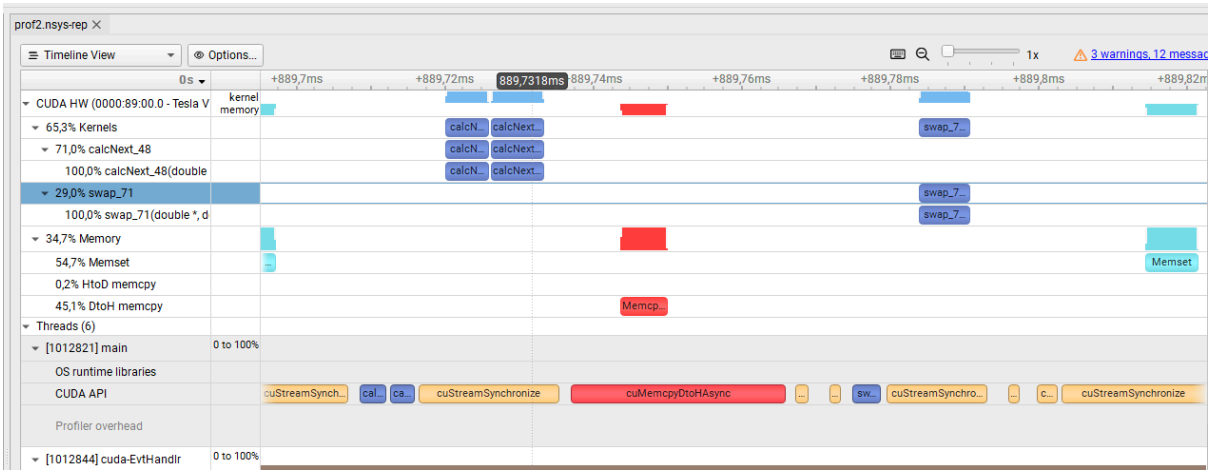
Размер сетки	Время выполнения(с)	Точность	Количество операций
128*128	0.305420	0.000001	300101
256*256	1.173664	0.000001	102901
512*512	4.485946	0.000001	339601
1024*1024	35.504507	>0.000001	1000000

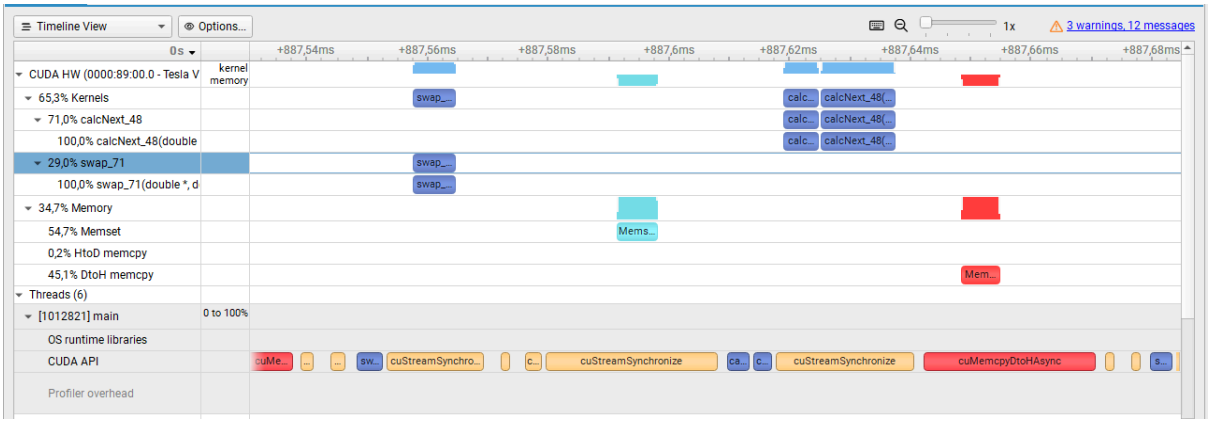
Диаграмма сравнения времени работы CPU-one, CPU-multi, GPU(**оптимизированный** вариант) для **разных** размеров **сеток**

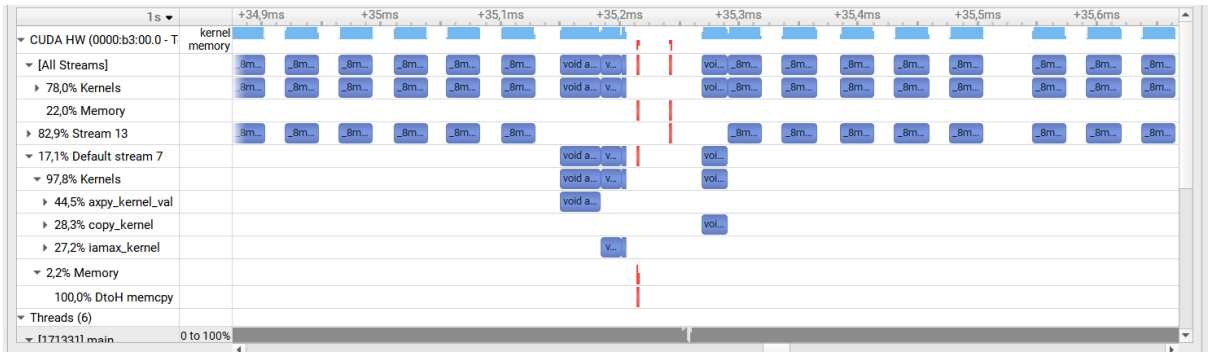


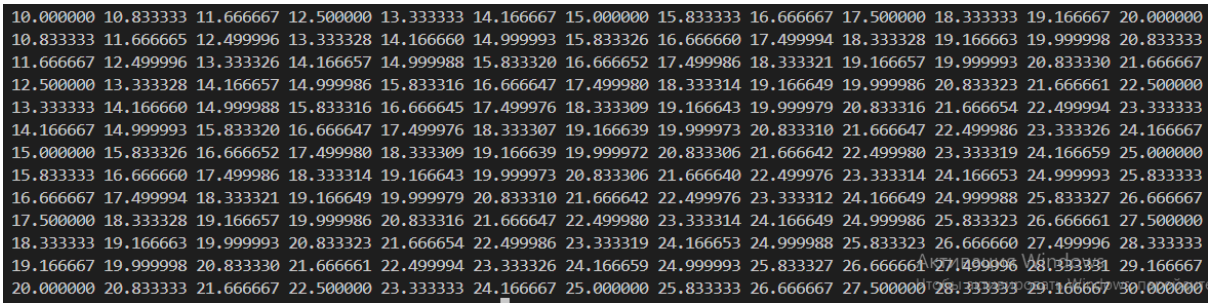

Название диаграммы
160
140
120
100
80
60
40
20
0
512*512
one
multi
gpu
cuBLAS


Название диаграммы
350
300
250
200
150
100
50
0
1024*1024
multi
gpu
cuBLAS


prof2.nsys-rep
Timeline View
Options...
1x
3 warnings, 12 messag
0s
+889,7ms
+889,72ms
889,7318ms
+889,74ms
+889,76ms
+889,78ms
+889,8ms
+889,82m
CUDA HW (0000:89:00.0 - Tesla V
kernel
memory
65,3% Kernels
71,0% calcNext_48
100,0% calcNext_48(double
29,0% swap_71
100,0% swap_71(double *, d
34,7% Memory
54,7% Memset
0,2% HtoD memcpy
45,1% DtoH memcpy
Threads (6)
[1012821] main
0 to 100%
OS runtime libraries
CUDA API
Profiler overhead
[1012844] cuda-EvtHandlr
0 to 100%
cuStreamSynch...
cuStreamSynchronize
cuMemcpyDtoHAsync
cuStreamSynchro...
Memset
Memcp...


Timeline View
Options...
1x
3 warnings, 12 messages
0s
+887,54ms
+887,56ms
+887,58ms
+887,6ms
+887,62ms
+887,64ms
+887,66ms
+887,68ms
CUDA HW (0000:89:00.0 - Tesla V
kernel
memory
65,3% Kernels
71,0% calcNext_48
100,0% calcNext_48(double
29,0% swap_71
100,0% swap_71(double *, d
34,7% Memory
54,7% Memset
0,2% HtoD memcpy
45,1% DtoH memcpy
Threads (6)
[1012821] main
0 to 100%
OS runtime libraries
CUDA API
Profiler overhead
cuStreamSynchro...
cuStreamSynchronize
cuMemcpyDtoHAsync
Mems...
Mem...


1s
+34,9ms
+35ms
+35,1ms
+35,2ms
+35,3ms
+35,4ms
+35,5ms
+35,6ms
CUDA HW (0000:b3:00.0 - T
kernel
memory
[All Streams]
78,0% Kernels
22,0% Memory
82,9% Stream 13
17,1% Default stream 7
97,8% Kernels
44,5% axpy_kernel_val
28,3% copy_kernel
27,2% iamax_kernel
2,2% Memory
100,0% DtoH memcpy
Threads (6)
0 to 100%



```
10.000000 10.833333 11.666667 12.500000 13.333333 14.166667 15.000000 15.833333 16.666667 17.500000 18.333333 19.166667 20.000000
10.833333 11.666665 12.499996 13.333328 14.166660 14.999993 15.833326 16.666660 17.499994 18.333328 19.166663 19.999998 20.833333
11.666667 12.499996 13.333326 14.166657 14.999988 15.833320 16.666652 17.499986 18.333321 19.166657 19.999993 20.833330 21.666667
12.500000 13.333328 14.166657 14.999986 15.833316 16.666647 17.499980 18.333314 19.166649 19.999986 20.833323 21.666661 22.500000
13.333333 14.166660 14.999988 15.833316 16.666645 17.499976 18.333309 19.166643 19.999979 20.833316 21.666654 22.499994 23.333333
14.166667 14.999993 15.833320 16.666647 17.499976 18.333307 19.166639 19.999973 20.833310 21.666647 22.499986 23.333326 24.166667
15.000000 15.833326 16.666652 17.499980 18.333309 19.166639 19.999972 20.833306 21.666642 22.499980 23.333319 24.166659 25.000000
15.833333 16.666660 17.499986 18.333314 19.166643 19.999973 20.833306 21.666640 22.499976 23.333314 24.166653 24.999993 25.833333
16.666667 17.499994 18.333321 19.166649 19.999979 20.833310 21.666642 22.499976 23.333312 24.166649 24.999988 25.833327 26.666667
17.500000 18.333328 19.166657 19.999986 20.833316 21.666647 22.499980 23.333314 24.166649 24.999986 25.833323 26.666661 27.500000
18.333333 19.166663 19.999993 20.833323 21.666654 22.499986 23.333319 24.166653 24.999988 25.833323 26.666660 27.499996 28.333333
19.166667 19.999998 20.833330 21.666661 22.499994 23.333326 24.166659 24.999993 25.833327 26.666661 27.499996 28.333331 29.166667
20.000000 20.833333 21.666667 22.500000 23.333333 24.166667 25.000000 25.833333 26.666667 27.500000 28.333333 29.166667 30.000000
```

```
10.000000 11.111111 12.222222 13.333333 14.444444 15.555556 16.666667 17.777778 18.888889 20.000000
11.111111 12.222220 13.333330 14.444440 15.555550 16.666662 17.777773 18.888886 19.999998 21.111111
12.222222 13.333330 14.444438 15.555547 16.666657 17.777768 18.888880 19.999994 21.111108 22.222222
13.333333 14.444440 15.555547 16.666655 17.777765 18.888876 19.999989 21.111103 22.222218 23.333333
14.444444 15.555550 16.666657 17.777765 18.888874 19.999985 21.111098 22.222213 23.333328 24.444444
15.555556 16.666662 17.777768 18.888876 19.999985 21.111096 22.222209 23.333324 24.444439 25.555556
16.666667 17.777773 18.888880 19.999989 21.111098 22.222209 23.333322 24.444436 25.555551 26.666667
17.777778 18.888886 19.999994 21.111103 22.222213 23.333324 24.444436 25.555549 26.666663 27.777778
18.888889 19.999998 21.111108 22.222218 23.333328 24.444439 25.555551 26.666663 27.777776 28.888889
20.000000 21.111111 22.222222 23.333333 24.444444 25.555556 26.666667 27.777778 28.888889 30.000000
```

Вывод: ускорение программы наибольшее при следующих оптимизациях:

1. Замена копирования массива на swap указателей, это занимает меньше времени(если смотреть на профилировании, >>1 мс)

2. Возврат с gpu на cpu ошибки при большом количестве операций, не обязательно проверять ошибку каждый раз, так как копирование занимает также время

Ускорение с помощью cuBLAS:

1. На матрицах меньшего размера ускорение есть, при увеличении матриц разница между cuBLAS и GPU с директивами openACC незначительна