Introduction to CUDA and OpenCL
Lab 3 report

19.03.2020
Piotr Litwin
Paweł Skalny

Introduction

During lab we were testing difference in time between 1D1D and 2D2D configuration of block (first *D) and grid (second *D) dimentions. We were comparing times for matrix addition, hadamrd product and dyadic product. Also creating a program dividem in multiple function such as initilisation, processing, testing and printing.

Creating program

We created a program in two configurations: 1D1D and 2D2D, which represent 1 dimention block and 1 dimention grid, and 2 dimention block and 2 dimention grid. Underneath table repesents the number of elements, thread, blocks that we setup in program. Program executes three function which calculate matrix addition product, Hadamard product and dyadic (tensor) product and also time of execution for each of them on GPU (device) and CPU (host).

Number of elements	Threads 1D1D	Blocks 1D1D	Threads 2D2D	Blocks 2D2D
10	256	1	16 x 16	1 x 1
100	256	1	16 x 16	1 x 1
1000	256	1	16 x 16	2 x 2
10000	256	1	16 x 16	7 x 7
100000	256	2	16 x 16	20 x 20
1000000	256	4	16 x 16	63 x 63
10000000	256	13	16 x 16	198 x 198
100000000	256	40	16 x 16	625 x 625

As we can easily see, 2D2D grid contains much more threads overall than 1D1D grid. That's because in 1D1D case we launched the kernel for each column, and then we performed calculation in a common loop while in 2D2D case we launched the kernel for each element in the matrix.

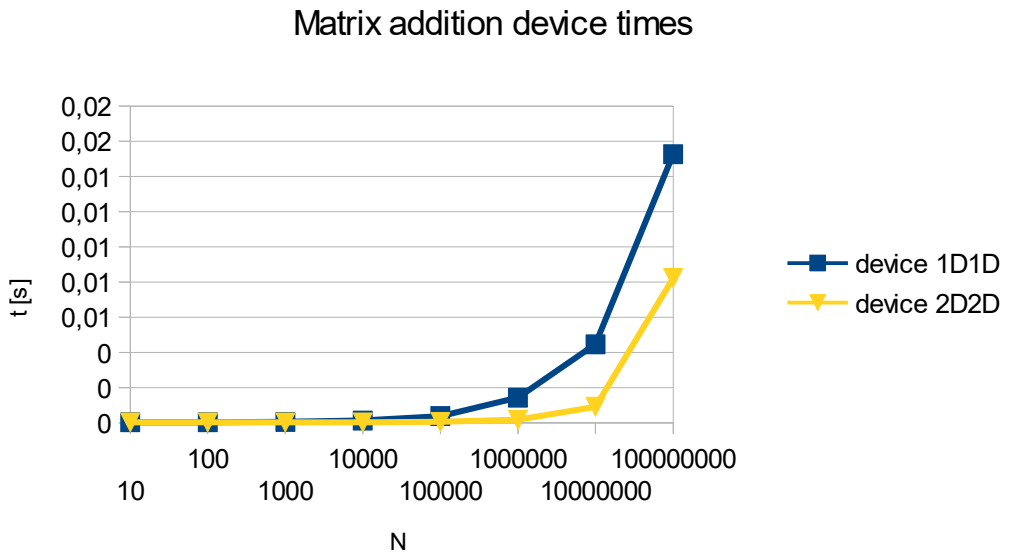
Results

All measurements were made ten times and then we calculated the average. Times are provided in µs (microseconds).

Matrix addition product

Below we have times of execution in respect to number of elements. First table has results from 1D1D configuration and 2D2D configuration. The chart below represents difference in times between configurations.

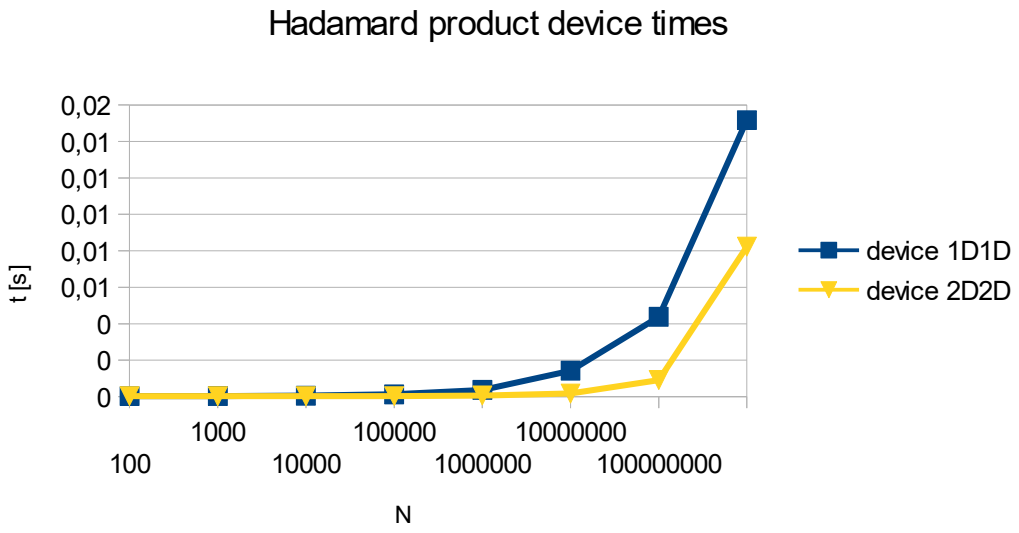
Number of elements	Time device 1D1D	Time device 2D2D	Time host
10	27,7	24,8	7
100	33,1	21,6	7,2
1000	53,1	21,5	14,1
10000	129	22	90,9
100000	381,2	50,5	1154,1
1000000	1427,5	173,5	17541,7
10000000	4463,5	913	156951,3
100000000	15271,6	8232,8	1156946



Hadamard product

Below we have times of execution in respect to number of elements. First table has results from 1D1D configuration and 2D2D configuration. The chart below represents difference in times between configurations.

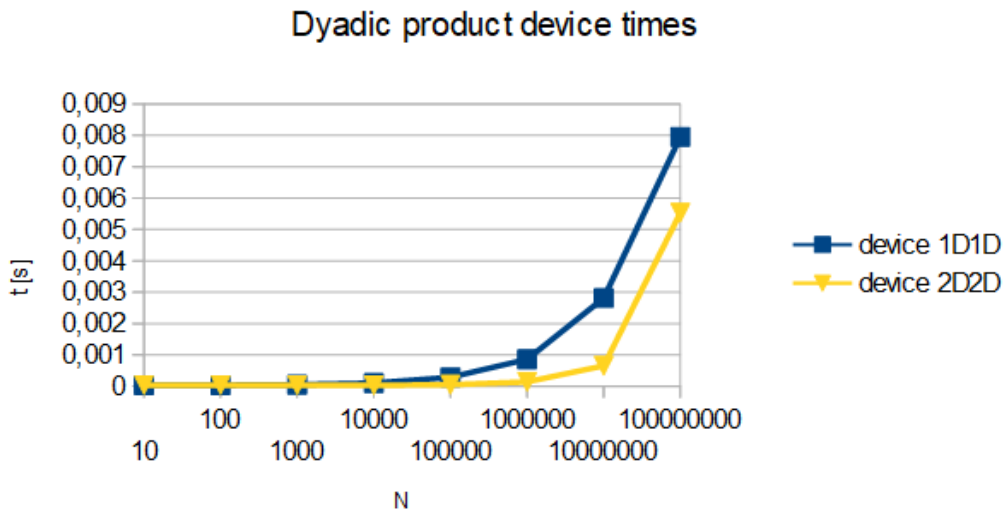
Number of elements	Time device 1D1D	Time device 2D2D	Time host
10	25,8	29	6,4
100	30,6	21,5	7,3
1000	52,2	24	14,5
10000	124,9	25,1	99,8
100000	370,6	59	1170
1000000	1418,1	171,1	16036,8
10000000	4380,2	911,7	157085,9
100000000	15167,2	8227,5	1156729,5



Dyadic product

Below we have times of execution in respect to number of elements. First table has results from 1D1D configuration and 2D2D configuration. The chart below represents difference in times between configurations.

Number of elements	Time device 1D1D	Time device 2D2D	Time host
10	36,1	33,5	11,5
100	37	32,9	11,5
1000	52,4	32,6	18,9
10000	104,5	31,7	86,1
100000	287,6	50,5	981,6
1000000	860,3	142,7	8979,1
10000000	2814,2	653,1	109382,7
100000000	7953,8	5533,6	919791,4



Conclusion

From the times in table and charts comparing them we see that in most of cases the 2D2D configuration is faster. This stems from the fact that this configuration will have more calculations done at the same time, among others, due to the use of more threads (as long as they are physically available). It seems that the dyadic problem was solved slightly faster, so we guess that repeated use of the same elements could increase the performance.