Report 3

Introduction to CUDA and OpenCL

Grid Size Evaluation and Unified Memory tests.

Dariusz Biela

Karol Sawicki

1. Introduction

The third laboratory classes was the first time we were working with matrix multiplication kernels. We prepared two of them – one with traditional index mapping and one with stride grid loop in the kernel. This loop jumps in large data sets to process more than one object in one thread.
Each kernel we were testing with two data sets, one prepared with copying the data from the host to the device and one with Unified Memory handling.
To check errors we prepared simple wrapper for handling all of them. We were using it by calling CUDA functions that could return cudaError_t as argument for our wrapper, because it is simple way to do this.

2. Calculations and measurements

We decided to measure time for memory allocation, initialization and kernel execution. With that measurements we could compare two ways of memory handling: classic mallocs and more elegant Unified Memory.

Figure 1 Graph comparing time for each way of processing

The first graph is inconsistent, three of our curves are really similar, but the first one gains much higher values, we can observe that at the end it decreases to the same level as others. It could be caused by some other tasks executed on the server at the same time, not especially by the GPU.

To make sure that the code was good we decided to repeat measurements for classic memory allocation with first kernel.

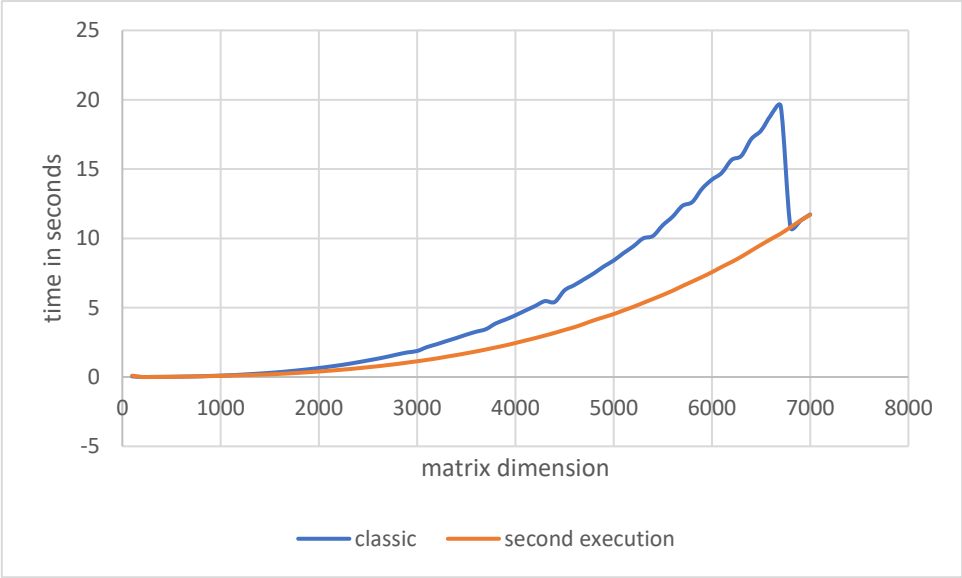

Figure 2 First and second run comparison

At the second graph we can see that everything is fine and values from the second run are a way more similar to the others.

To see more details we prepared graphs comparing memory allocation types and one that compares kernels.

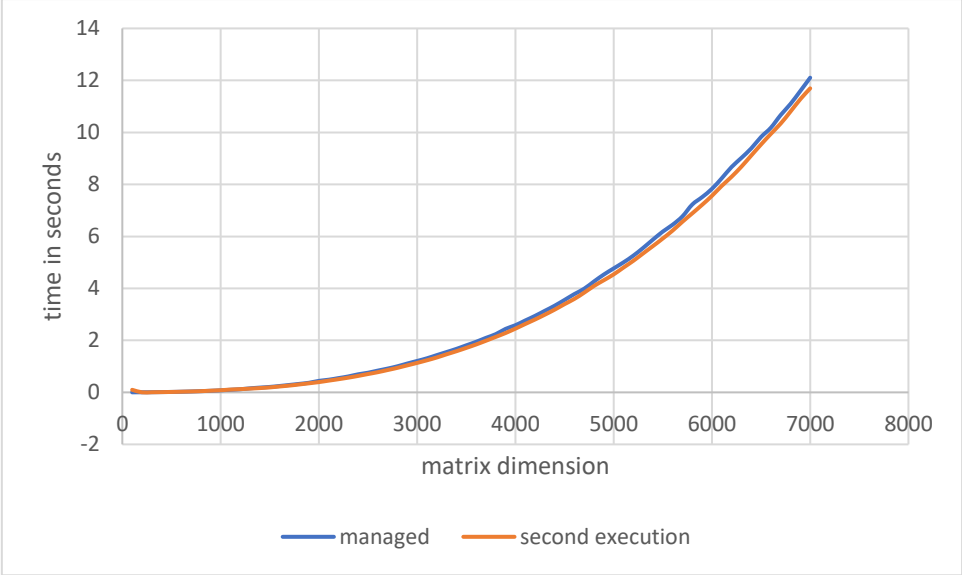

Figure 3 First kernel, second execution stands for classic memory handling from our second run

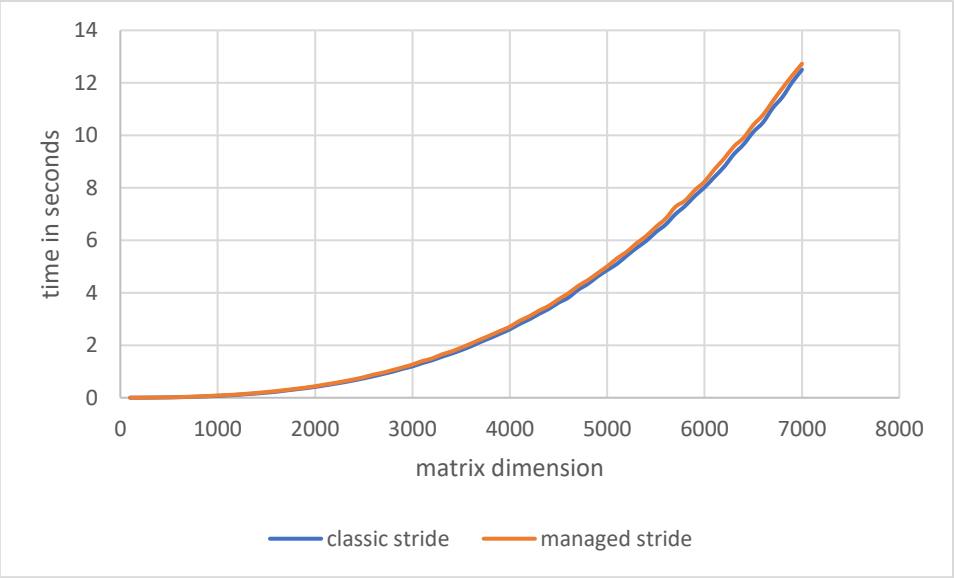

Figure 4 Second kernel, with loop

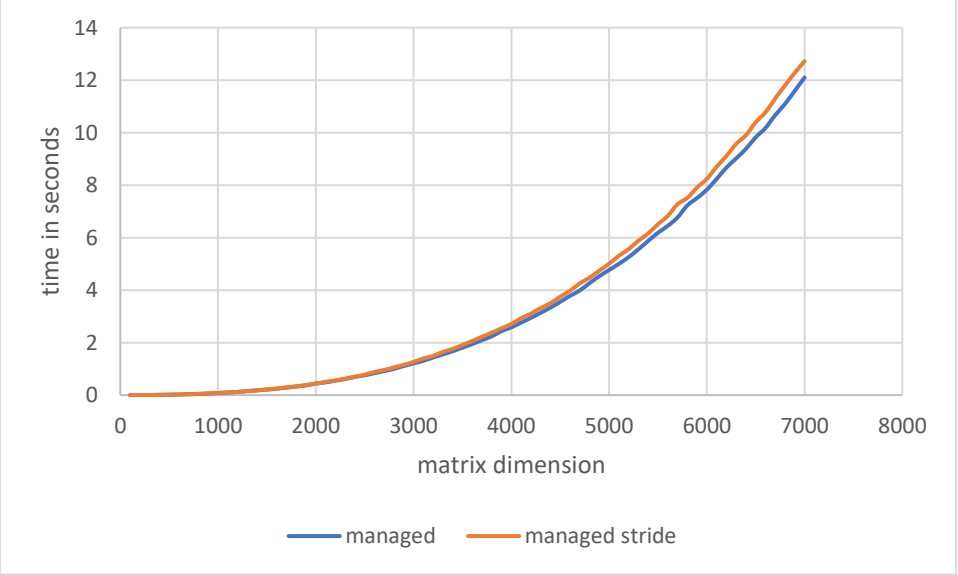

Figure 5 Comparison for the same memory handling with other kernels

3. Conclusions

We should always be sure that our test environment is free from other tasks, because they can impact measurements in a very meaningful way. If it is possible we should do measurements more than once.

We could observe that Unified Memory does not impact performance in a significant way, both graphs are reaching higher values, but it is really small difference. It is worth to use managed memory, because it not hurt the performance, but makes our code some much cleaner and easier.

Our both kernels are really similar, but we should know how to achieve that – each of them requires other processing grid layouts. The first one works well with block size that depends of data size, when the second – the one with loop – works well with the block size that has the value of power of two.