Report 8

Introduction to CUDA and OpenCL

Reduction Pattern.

Dariusz Biela

Karol Sawicki

1. Introduction

Our last laboratory classes was about reduction pattern – we had to prepare kernel that works with a lot of data at the start of execution and then data size reduces.

We were encouraged to focus on partial sums of vector, which contains floating point numbers.

To understand what happens and why reduction pattern is important we had to prepare naïve kernel that sums vector and kernel that uses thread synchronization and shared memory to perform a real data reduction.

2. Calculations and measurements

To start our investigations we prepared two kernels – one with simple atomic addition, performed by each thread on its own data; and the second one, which performs reduction. The second kernel works with thread synchronization, shared memory and atomic addition. These three tools allows us to get really big time savings when sum of vector elements is needed.

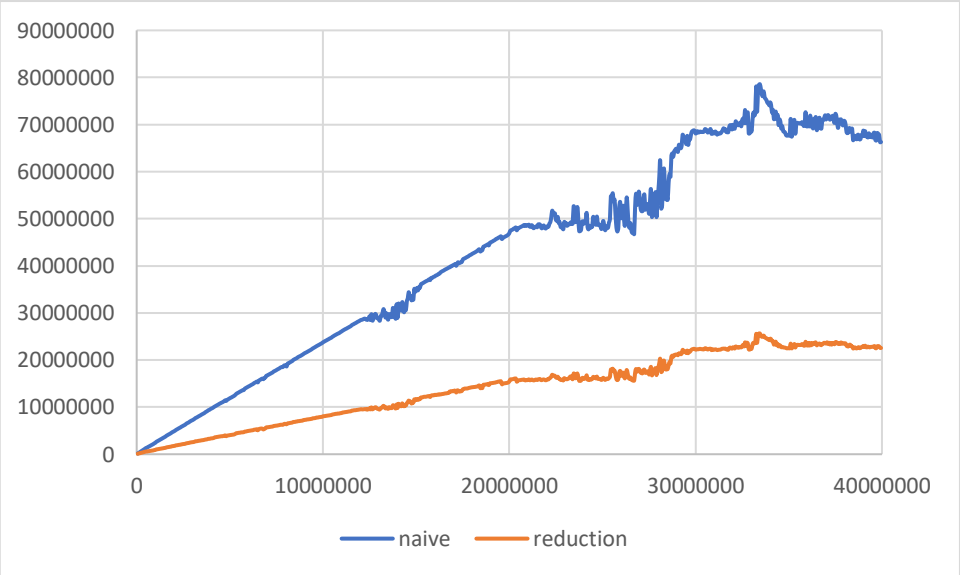

Figure 1 Comparison of time needed to perform addition of all vector elements with naive algorithm (blue) and algorithm, which uses reduction pattern (orange).

Our graph shows us, that naïve implementation can be 3 to 4 times slower than reduction pattern. With one vector it should not be problematic, but when we have thousands of them or more the problem is much bigger, especially when time is really important to us.

3. Conclusions

We do not need to know about advanced mechanics, when we have to create CUDA accelerated programs. Many problems can be solved with simple good known mechanics and the results will be good. However – knowing about advanced topics is really useful, when we need better performance. We can write program without thread synchronization or shared memory, the only needed instruction is atomic addition, but execution time can be so much worse without any of these more advanced mechanics.

Knowing about advanced mechanics will be preferable, while learning CUDA, because it can help us create most optimized programs, what should be the most important, when we are using GPU chips to perform our calculations.