Assignment IV - OpenCL: Discuss, Reflect, Contrast

DD2360
Simon Jäger (Group 6)

In my experience, I found CUDA a bit more friendly to deal with. In multiple aspects. One being the fact that there is less boilerplate code that needs to be made to get started in CUDA, compared to OpenCL. Also, developing the kernel as part of a string that is compiled during runtime was not a pleasant experience. As you cannot take advantage of your IDE to give you feedback about syntax errors and formatting. Perhaps this can be resolved if one loads an external file? If so, I would much rather prefer that approach.

Another observation was the readability of the code - I had the feeling that the CUDA code became much more concise and aesthetically pleasing to look at. I'd imagine this would lead to better and more maintainable code in the future. Perhaps even with the potential to share code and functions nicely? Whereas with OpenCL I'm not sure if that is possible - given that the kernel is compiled at runtime using the string input.

In terms of comparing the performance of my CUDA implementation with my OpenCL implementation I had to create a very different setup. For CUDA, I could leverage an Ubuntu environment with an NVIDIA Tesla V100. Whereas for OpenCL, which I could not get working there - I had to resort to my own desktop and run on Windows with an NVIDIA GeForce GTX 1080 Ti. As such, my results are not very comparable.

The OpenCL implementation was much slower. Most likely due to multiple reasons. One obviously being the hardware difference (NVIDIA Tesla V100 vs NVIDIA GeForce GTX 1080 Ti), another one being the fact that OpenCL most likely needs to abstract more than CUDA must. In-order to achieve the abstraction above different infrastructure. This, I presume, does come with a cost in terms of performance.

In my opinion, it was a great opportunity to try both CUDA and OpenCL. While I preferred the experience in CUDA - I understand where OpenCL makes more sense to use (e.g., for portability).