Discuss, Relfect and Contrast

In this exercise, I'm going to briefly discuss the usability of OpenCL with my own experience, and its performance versus CUDA introduced by Nvidia on some simple test cases in past assignments.

Usability of OpenCL

In assignment II and III, I was required to utilize CUDA toolkit provided by Nvidia, and my experience was that CUDA is super easy to grasp and understand. The hard part in writing CUDA code is not brand new functions or grammars but the throughput-oriented structure that I need to understand. The correspondence between functions and their usage is pretty clear and easy to understand once I have known the organization and layout of GPU.

In this assignment, I'm required to switch to OpenCL, which makes me remember the time when I can use CUDA. The first impression of this library is verbosity, where I have to write tones of code just to implement a simple HelloWorld program. Another point I can not stand is that the arguments of functions are too hard to remember, which makes it difficult to write code without documentation. Overall, I think OpenCL is much less elegant and hard to use.

Performance

In the SAXPY application, I did simple element multiplication and addition. My expectation was they should have same performance, but surprisingly, the program written in OpenCL ran one time faster than the one written in CUDA. This is quite strange because I did the test on the same platform and both of them use global memory and the same block size. My assumption is that this difference lies in subtle differences in memory access patterns and some optimization techniques used by CUDA and OpenCL. In CUDA I specifically specified to use global memory, while in OpenCL the program may implicitly transfer data to shared memory, which leads to performance difference. There also might be some measurement errors which are hard to avoid.