From CUDA to OpenCL: Towards a Performance-portable Solution for Multi-platform GPU Programming

A summary by Shawn Anderson

‘From CUDA to OpenCL: Towards a Performance-portable Solution for Multi-platform GPU Programming’ is a paper from 2012 in which the authors try to understand why OpenCL has such poor performance-portability, and if there are tweaks they can make to improve it.

OpenCL is an open source programming interface for writing parallel code to be executed across heterogeneous sets of computing devices such as CPUs and GPUs. It is designed with device portability in mind by the Khronos group. It is not, however, performance portable, meaning that its performance is variable across devices and architectures. This is mainly due to it’s very expensive kernel initialization, which CUDA beats by an order of magnitude. The authors explore the barriers to performance portability in pursuit of CUDA level portability with OpenCL.

The authors implement two kernels in OpenCLfor their experiments: a triangle solver, and matrix multiplication. They implement the kernels in such a way to optimize portability between two state of the art graphics cards at the time: Nvidia Tesla C2050 and ATI Radio 5870. The authors observe two key factors in performance portability. These factors are data copying from global memory, and compiler optimizations. They find that auto-tuning is key for improving performance portability. Auto-tuning is “collecting and generating multiple kernel versions, implementing the same algorithm optimized for different architectures, and heuristically selecting the best performing one”. Auto-tuning has historically been very popular for generating near optimal numeric libraries on CPUs.

The authors are able to achieve a portability solution that maintains 50 percent peak performance on both the Nvidia and ATI cards. Key contributions are identification of initialization overhead as the main barrier to portability, and the use of auto-tuning in efforts to improve portability. Ultimately they conclude that the implemented kernels, which include matrix multiplication, and a triangle solver, are not highly portable.