Chapter 15

CUDA or OpenCL: Which is Better? A Detailed Performance Analysis

Mayank Bhura
National Institute of Technology Karnataka, India

Pranav H. Deshpande
National Institute of Technology Karnataka, India

K. Chandrasekaran
National Institute of Technology Karnataka, India

ABSTRACT

Usage of General Purpose Graphics Processing Units (GPGPUs) in high-performance computing is increasing as heterogeneous systems continue to become dominant. CUDA had been the programming environment for nearly all such NVIDIA GPU based GPGPU applications. Still, the framework runs only on NVIDIA GPUs, for other frameworks it requires reimplementation to utilize additional computing devices that are available. OpenCL provides a vendor-neutral and open programming environment, with many implementations available on CPUs, GPUs, and other types of accelerators, OpenCL can thus be regarded as write once, run anywhere framework. Despite this, both frameworks have their own pros and cons. This chapter presents a comparison of the performance of CUDA and OpenCL frameworks, using an algorithm to find the sum of all possible triple products on a list of integers, implemented on GPUs.

INTRODUCTION

Of recent, multi-core and many-core processors have far surpassed the performance of the sequential processor. Since the advent of GPGPUs, their inherent parallel architecture coupled with the much higher amount of bandwidth and floating operations per second (FLOPS), there has been an increase in the use of high-end graphic processors over the past many years. High Definition graphics has led its way from the glory of gaming industry to the scientific realm of higher floating point calculations. Complex operations are executed in parallel in a multithreaded environment with enormous computa-
tional horsepower. Increasing parallelism rather than clock rate has been the motive ever since. Titan, a supercomputer built for use in science projects, is the first hybrid model consisting of both CPUs and GPUs to achieve 17.59 petaFLOPS in speed, its theoretical peak value being 27 petaFLOPS. It consists of 18,688 AMD Opteron 6274 16-core CPUs and 18,688 Nvidia Tesla K20X GPUs.

Over the years, many frameworks have been developed to efficiently utilize the parallelism and bandwidth of our modern day GPUs. CUDA (Compute Unified Device Architecture), developed by Nvidia, has eliminated the need of graphic API. Similarly, the APP (Advanced Parallel Processing) framework, developed by ATI, allows ATI’s GPUs to work together with CPUs to achieve an even greater amount of scalability and parallelism. OpenCL (Open Computing Language) on the other hand, is a standard framework for parallel programming on Heterogeneous Systems. Its portability enables various platforms to be tested without the need to rebuild the programs from scratch. All these frameworks have been developed over a long time of research, starting from the high-level shading languages such as HLSL and GLSL, and research is bound to further reach new limits in the era of parallel computing.

But when it comes to decide which framework to choose, deep thought has to be given in to decide the parameters of comparison, along with proper implementations in the respective frameworks to have a fair comparison. There are all kinds of applications. Each of them can have their own domain of tasks. Thus, it may not be suitable to fix a framework for creating all types of applications. Each framework has its own pros and cons, which makes it mandatory that for a good performance utilization, we first decide which one to choose for our application. Thus, we need to be able to compare and decide as to which of the frameworks are suitable for a given computational task, on a given computational environment.

This is exactly what this chapter is for. This chapter investigates the portability vs performance feature of the two frameworks, CUDA and OpenCL, over various parameters, through a common problem: finding the sum of all triple products over an increasing list of real numbers. Though simple, this problem requires large amount of multiplication operations. Moreover, the list consists of floating point numbers, which requires precision in calculations. This enables us to easily move forward to the implementation issues. It also enables the readers to understand the problem and its complexity, with just a basic level of knowledge in algorithms, which in turn would make it possible for many beginners to understand the matter being presented in this chapter, with ease.

For the implementation hardware, the authors decided to go with NVIDIA GPUs as the best choice for comparison. The reason being, CUDA only supports NVIDIA hardware, while OpenCL can be run on many other GPUs, including that of NVIDIA. Moreover, it cannot be said that CUDA is at an advantage while running it on its own vendor’s hardware. As the chapter will disclose, the performance readings have not been too different from each other.

Efforts are to further optimize the algorithm implementations in terms of parallel execution with minimum overheads, and compare the kernel runtime of the two frameworks for the same problem. This chapter also studies how the execution times of both frameworks change as load on GPU increases. In the upcoming sections we discuss some of the related work and later move on towards discussing the algorithm, optimization strategies and comparison of performance.

**Background**

There have been debates regarding the better and more suitable one between OpenCL and CUDA. CUDA currently has only one vendor, NVIDIA, while OpenCL is a vendor-neutral framework, making it possible to run OpenCL code on any hardware, since it is portable. On the other hand, CUDA