ACADEMIC
World Academic Union

ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 10, No. 2, 2015, *pp*.124-136

An In-depth Study on the Performance Impact of CUDA, OpenCL, and PTX Code

Puya Memarzia [1], Farshad Khunjush [2]

[1] School of Electrical and Computer Engineering, Shiraz University, pm@memarzia.com
[2] School of Electrical and Computer Engineering, Shiraz University, khunjush@shirazu.ac.ir
(*Received November 30, 2014, accepted February 26, 2015)*

Abstract. In recent years, the rise of GPGPU as a viable solution for high performance computing has been accompanied by fresh challenges for developers. Chief among these challenges is efficiently harnessing the formidable power of the GPU and finding performance bottlenecks. Many factors play a role in a GPU application's performance. This creates the need for studies performance comparisons, and ways to analyze programs from a fundamental level. With that in mind, our goal is to present an in-depth performance comparison of the CUDA and OpenCL platforms, and study how PTX code can affect performance. In order to achieve this goal, we explore the subject from three different angles: kernel execution times, data transfers that occur between the host and device, and the PTX code that is generated by each platform's compiler. We carry out our experiments using ten real-world GPU kernels from the digital image processing domain, a selection of variable input data sizes, and a pair of GPUs based on the Nvidia Fermi and Kepler architectures. We show how PTX statistics and analysis can be used to provide further insight on performance discrepancies and bottlenecks. Our results indicate that, in an unbiased comparison such as this one, the OpenCL and CUDA platforms are essentially similar in terms of performance.

Keywords: GPU; CUDA; OpenCL; PTX; Performance.

1. Introduction

The advent of General-Purpose computation on Graphical Processing Units (GPGPU) has long since opened the doors to utilizing GPUs for a wide variety of applications, and continues to gain momentum. GPUs are generally employed whenever an application requires heavy computational processing power, and has the potential to be split up into a large number of smaller tasks that can be performed in parallel. Software developers and researchers employ a variety of programming libraries and platforms to write GPGPU applications. Chief among these platforms is the Compute Unified Direct Architecture (CUDA) from Nvidia, and the Open Computing Language (OpenCL) from the Khronos Group. CUDA has established itself as the world's prominent GPU computing framework, and has traditionally received significantly higher attention from developers and researchers. Some studies have found CUDA to provide superior performance compared to OpenCL [6]. However, this appears to vary with the type of workload, and testing methodology [4]. In recent years, the OpenCL compiler and development tools have matured notably, and while it is still not a definitively superior programming platform, it is certainly more viable than it was in the past. GPGPU has been adopted on a broad variety of hardware, and the importance of portability (which is one of OpenCL's key features) and heterogeneous systems has risen dramatically. As a result, OpenCL has been increasingly gaining traction.

GPU applications are relatively complex compared to similar applications on the CPU. Many factors can affect a GPU application's performance. These include data transfers, access patterns, and the amount of parallelization. Writing efficient GPGPU applications is a challenging task. Different GPU architectures, programming platforms, algorithms, and memory spaces have the potential to influence an application's performance, significantly. Detailed analyses of these factors can help developers choose better hardware and programming platforms while writing better, and more efficient applications. They can also help with the development of better compilers and various developer tools, such as tuning guides, profilers, simulated GPUs, statistical performance models, modified compilers, and automatic optimization tools.

Nvidia GPUs are capable of executing both CUDA and OpenCL applications. This allows us to

implement a specific algorithm in CUDA, evaluate its performance on a set of data, and then port the code over to OpenCL, and repeat the procedure.

At their cores, the CUDA and OpenCL platforms share many similarities, such as the overall programming model, memory spaces, and a syntax that is very similar to C. One key similarity is the ability to save code to an intermediate language called Parallel Thread Execution (PTX). Since PTX is a low-level representation of the overlying code, it is directly related to performance, and can be useful for analyzing the different ways in which equivalent CUDA and OpenCL kernels are handled by their respective compilers. We combine PTX analysis with regular performance benchmarking. This provides us with a way to study the differences between CUDA and OpenCL from two different angles. PTX analysis can be beneficial to programmers because it can help them locate inefficiencies in the code. It can also be useful for compiler developers as it relates to compiler optimizations and performance.

Our goal is to compare and evaluate the differences between CUDA and OpenCL. We present a performance evaluation of a series of image processing kernels, implemented in both CUDA and OpenCL. The GPU kernels are implemented to be essentially identical.

The main contribution of this paper is to present a comprehensive performance comparison of the CUDA and OpenCL parallel programming platforms. Some of the key features of our work include: 1) we run the experiments on the latest GPU architectures from Nvidia (Fermi GF110 and Kepler GK104). 2) We measure kernel execution as well as data transfer times. 3) We perform statistical PTX analysis. 4) We take steps to ensure a fair comparison. 5) We explore the possibility of a correlation between the number of generated PTX instructions and the relative performance of the two platforms. We believe the sum of these features make our work distinctive from the prior work performed by others.

The reminder of this paper is organized as follows. Section 2 explores related work. A concise background on GPU architectures, GPU computing platforms, the PTX language, and image processing is provided in Section 3. We elaborate on our approach in Section 4, and describe our experimental methodology in Section 5. Our experimental results are presented and discussed in Section 6. We conclude the paper in Section 7, and outline future work in Section 8.

2. Related Work

Studying the performance impacts of GPGPU frameworks is not a novel concept. There have been relatively few comprehensive comparisons of the CUDA and OpenCL platforms. Some studies have focused on the performance differences between CUDA and OpenCL (with varying results), whereas others have attempted to explore the portability aspect of OpenCL. There has also been some work performed using simulated GPUs.

Fang et al. [4] presented an in-depth comparison of the CUDA and OpenCL platforms. They evaluated the performance of 16 benchmarks, consisting of real world and synthetic benchmarks, on the CUDA and OpenCL platforms. The tests were performed on an Nvidia GTX 480, GTX 280, and an ATI Radeon 5870. The authors used statistical PTX analysis to find out why their FFT benchmark exhibited the largest performance gap. Their experimental results suggest that OpenCL can perform on par with CUDA if the tests are performed in a fair manner.

Weber et al. [20] compared the performance and programmability of a Monte Carlo application on several platforms including Nvidia CUDA, and OpenCL. Their results indicate that the OpenCL platform provides portability between CPUs and GPUs, but may cause a drop in performance.

Karimi et al. [6] compared the performance of CUDA and OpenCL on a Monte Carlo simulation using near-identical kernels. They also explained the process involved in converting a CUDA kernel to an OpenCL kernel. Their results show CUDA consistently outperforming OpenCL in both data transfer times and kernel execution times. They used an Nvidia GTX 260 for their experiments.

A study on the performance and portability of OpenCL kernels was performed by Komatsu et al. [7]. The authors evaluate the performance several equivalent CUDA and OpenCL programs, and investigate the reasons behind their performance differences. Their results initially suggest that the CUDA kernels are significantly faster than their OpenCL counterparts are. They then demonstrate that OpenCL programs can perform similarly to CUDA programs, only if they are optimized by hand, or if the OpenCL compiler parameters are manually tuned for each device. The authors performed their experiments using a Radeon 5870, and an Nvidia Tesla C1060.