Performance Analysis of Application Kernels in Multi/Many-Core Architectures

Karthik Balasubramanian, Dr. Ioan Raicu

Abstract— In recent years, advancement in technology and computing led to huge amounts of data being generated. Thus, High-Performance Computing (HPC) plays an ever growing role in processing these large datasets in a timely fashion. Our analysis consist of few important throughput computing app kernels which have high degree of parallelism and makes them excellent candidates for evaluation on high end multi-core CPUs, and many-core GPUs. In this work, we performed a performance comparison of important app kernels like Image Convolution, Histogram and Bilateral filtering in multi-core CPU, many-core NVIDIA GPUs in addition to comparing our research framework GPU enabled Many-Task Computing (GeMTC). GeMTC is an execution model and runtime system which enables NVIDIA GPUs to be programmed with many concurrent and independent tasks of potentially short or variable duration. In this work we provide a thorough performance analysis between CPU, CUDA, and the GeMTC framework. Through this we better understand the behavior of different applications that belong to the Many-Task Computing paradigm. The results show that the GeMTC framework shows promising results for Many-Task Computing workloads running on NVIDIA GPUs.

I. Introduction

In the past years, there is a huge number of digital content as more documents are being created in digital form ever before. The massive amount of data makes storing, cataloging, processing, and retrieving information challenging. In this work we have identified certain applications that can process huge amount of data and deliver appropriate content to the user. This work explores the performance evaluation of multi-core, many core and a framework built on many core architecture. Also this work attempts to correlate throughput computing characteristics with architectural features on today's CPU and GPU's.

Motivation: Prior to this work there is a research paper which compares the performance of CPUs with GPUs [1]. The work concludes CPUs and GPUs are much closer in performance (2.5X) than the previously reported orders of magnitude difference.

II. Background

In this work we have selected three applications, Histogram, Image convolution and Bilateral Filtering. All these applications are executed in CPU, GPU and GeMTC. These kernels have a large amount of data-level parallelism, which makes them a natural fit for modern multi-core architectures. In the following, we will see about the application kernels.

GeMTC [2, 5] is an execution model and runtime system that enables NVIDIA GPUs to be programmed with many concurrent and independent tasks of potentially short or variable duration.

1. **Histogram** computation is an important image processing algorithm which hashes and aggregates pixels from the continuous stream of data into a smaller number of bins. Here we used 256-bin histogram, which is suitable for image processing applications that require higher precision than 64 bins can provide.
2. **Image Convolution** is a common image filtering operation used for effects such as blur, emboss and sharpen. Each pixel is calculated independently, thus providing ample parallelism at both SIMD and thread level. Simple calculation of Image Convolution is depicted in Figure 1.[3]

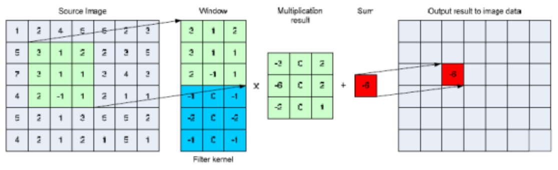

Figure 1: Image Convolution calculation

3. **Bilateral Filtering** is a common non-linear filter used in image processing for edge-preserving smoothing operations. The core computation has a combination of a spatial and an intensity filter. The neighboring pixel values and positions are used to compute new pixel values.

III. Architecture

This section describes the architecture for building the application kernels in GPU and GeMTC. Since CPU is a straightforward approach we will not list down here.

For GPU, for each application the device code and host code are placed in the same file. The application allocates memory and randomly generates the numbers. Then it does cudaMalloc to copy the data to device memory. The device code is called by taking all the blocks in the GPU and the number of threads passed as an argument.

For GeMTC, we need to use the GeMTC API to allocate memory and call the kernel application.

IV. Evaluation

This section evaluates the comparison between CPU, GPU and GeMTC. The target test bed for this implementation is GTX 670 GPU, a machine at DataSys Laboratory and AMD Phenom(tm) II X6 1100T Processor with 6GB RAM. For each of the application kernel, the performance of CPU, GPU and GeMTC are done. The performance is evaluated based on two parameters, throughput and FLOPS.

A. GPU

GPU version is evaluated by varying the threads with problem size. The problem size is started from 1 KB to 12 MB. This is because it needs to be compared to the GeMTC workloads, which will splits the task into number of warps the GPU consist of. In order minimize the startup cost for the GPU version, each application is executed for at least 5 – 10 seconds. For example for the problem size of 1Kb, the number of test performed is 1000000 times.

B. GeMTC

For GeMTC, the problem size is varied from 100 KB to 1GB. In GTX 670, there are 84 warps so each warp takes the problem/84 warps MB. So there are 84 tasks pushed to GeMTC and 84 workers are parallel performing the task.

C. CPU

For CPU, the histogram application is executed 84 times with 6 application running in parallel at a time. Even for CPU, the task size is reduced 1KB to 12 MB. So that it can be compared to GeMTC workloads.

I. HISTOGRAM

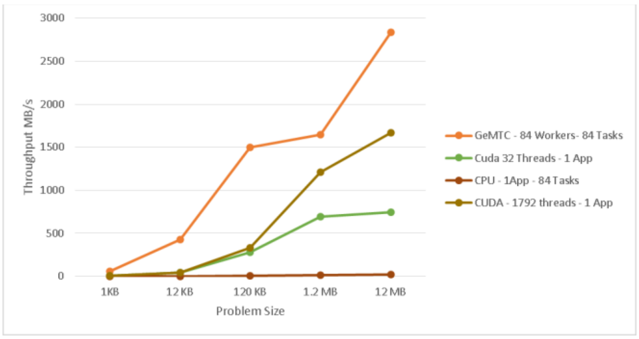

Figure 2: Histogram Throughput Comparison

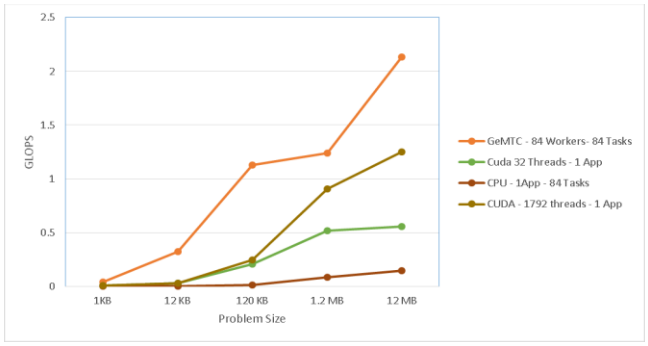

Figure 3: Histogram Flops Comparison

In Figure2 and 3, the comparison between GeMTC, CPU and GPU are shown. The histogram takes image size and the mask width as the input and it takes 3 *n operations. CPUs have low start up compared to CUDA or GeMTC but the cost is ignored by executing for long time. From the figure it is evident that CPU perform very poor when compared to GPU or CUDA. GeMTC problem size are very high when compared to CUDA so the throughput is very high. GeMTC has a peak of 3GB/s for 1GB problem size which is divided into 84 tasks.

II. IMAGE CONVOLUTION

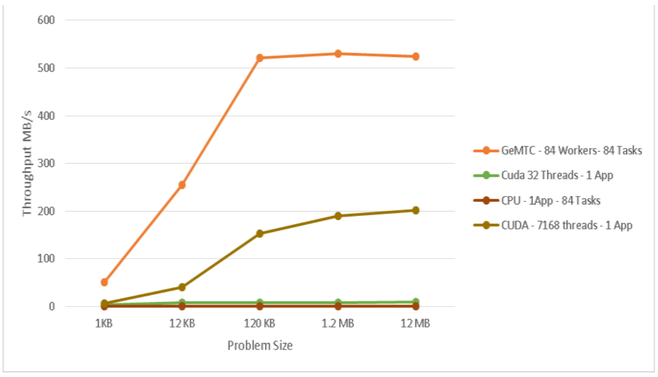

Figure 4: Image Convolution Throughput Comparison

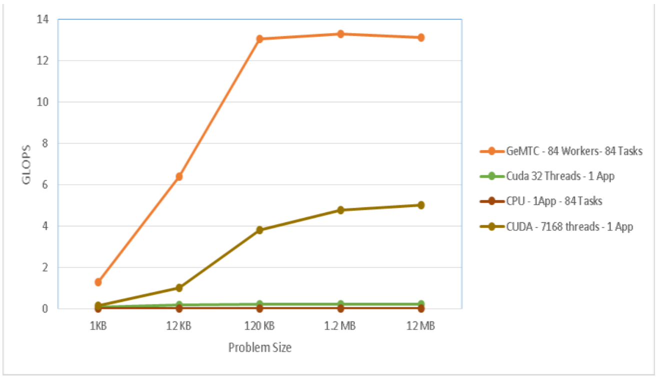

Figure 5: Image Convolution Flops Comparison

In Figure 4 and 5, the comparison between GeMTC, GPU and CPU are depicted. The CPU performs very poor compared to GeMTC and GPU. The GeMTC workloads starts from 100 KB to 1 GB and it divides the workload to 84 workers. Each worker runs on 32 threads. These applications are executed multiple times in order to neglect the variations involved.

III. BILATERAL FILTERING

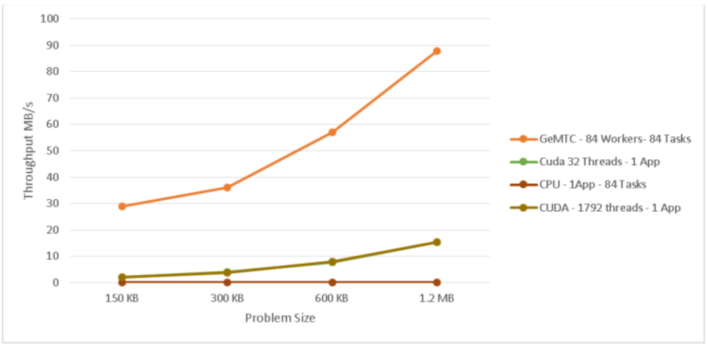

Figure 6: Bilateral Filtering Throughput Comparison

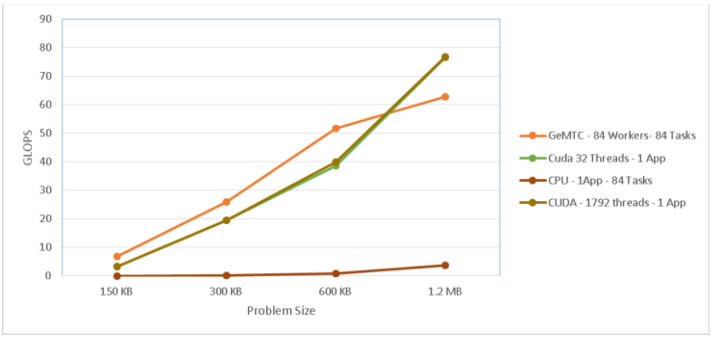

Figure 7: Bilateral Filtering Flops Comparison

In figure 7 and 8, the comparison for GPU, CPU and GeMTC are depicted. The CPU performs very slowly when compared to GPU and GeMTC. The GeMTC workloads ranges from 1 Megapixel to 8 Megapixel. Each pixel holds 12 bytes of R, G and B values. The 32 thread CUDA and whole GPU performs almost same. The algorithm is referred from [4]. There can be many improvements done to improve the performance of the application. Since this application is compute intensive, we see very less variation when the number of thread increases.

V. CONCLUSION AND FUTURE WORK

Thus in this work, we analyzed the performance analysis of GPU, GeMTC and CPU. And we see that GPU and GeMTC performs really well when compared to CPU. And there is more speedup compared to CPU. We believe many factors contributed to the reported large gap in performance, such as which CPU and GPU are used and what optimizations are applied to the code.

Future Work includes explore additional application for GeMTC, improving locking mechanism in GeMTC to improve total run time. Integrating with Swift/T.

REFERENCES

[1]. Lee, Victor W., et al. "Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU." ACM SIGARCH Computer Architecture News. Vol. 38. No. 3. ACM, 2010..

[2]. Krieder, Scott, and Ioan Raicu. "GeMTC: GPU enabled many-task computing." Illinois Institute of Technology, Department of Computer Science, PhD Oral Qualifier (2013).

[3]. Podlozhnyuk, Victor. "Image convolution with CUDA." NVIDIA Corporation white paper, June 2097.3 (2007).

[4]. Bilateral Filtering with CUDA,Lasse Klojgaard Staal, University of Aarhus

[5]. Scott J. Krieder, Justin M. Wozniak, Timothy Armstrong, Michael Wilde, Daniel S. Katz, Benjamin Grimmer, Ian T. Foster, Ioan Raicu. “Design and Evaluation of the GeMTC Framework for GPU-enabled Many-Task Computing”, ACM HPDC 2014