Performance Analysis of GPU compared to Single-core and Multi-core CPU for Natural Language Applications

Shubham Gupta
Master of Technology,
School of Computing Sciences and Engineering,
VIT University, India

Prof. M.Rajasekhara Babu
School of Computing Sciences and Engineering,
VIT University, India

***Abstract*— In Natural Language Processing (NLP) applications, the main time-consuming process is string matching due to the large size of lexicon. In string matching processes, data dependence is minimal and hence it is ideal for parallelization. A dedicated system with memory interleaving and parallel processing techniques for string matching can reduce this burden of host CPU, thereby making the system more suitable for real-time applications. Now it is possible to apply parallelism using multi-cores on CPU, though they need to be used explicitly to achieve high performance. Recent GPUs hold a large number of cores, and have a potential for high performance in many general purpose applications. Programming tools for multi-cores on CPU and a large number of cores on GPU have been formulated, but it is still difficult to achieve high performance on these platforms. In this paper, we compare the performance of single-core, multi-core CPU and GPU using such a Natural Language Processing application.**

Keywords- NLP; Lexical Analysis; Lexicon; Shallow Parsing; GPU; GPGPU; CUDA; OpenMP.

I. INTRODUCTION

In recent times, CPU supports multi-cores each supports improved SIMD instruction sets. And recent GPU supports a large number of cores which run in parallel, and its peak performance outperforms CPU. In [1], comparison of performance on CPU, FPGA and GPU is done using some image processing applications. And in [4], performance analysis of CPU and GPU is performed on some medical image volume rendering application. In this paper, we compare the performance of GPU and CPU (single-core and multi-core) using a NLP application.

In many real-life applications in the areas such as syntactic pattern recognition, syntactic analysis of programming languages etc., the parser speed is an important factor. Since large sum of data is to be processed, efficient low complexity parsers are required. All the techniques of parsing in Natural Language Processing involve string matching as the single most important operation. Traditionally for many years, GPU is just used to accelerate some stages of the graphics rendering pipeline. However, after the programmability available on this chip, GPU opens a door to developers to take advantage of its ALUs besides graphics processing. Compared with CPU in their architectures, GPU is more suitable for stream computations; it can process data elements in parallel with SIMD & MIMD capability. And in many cases, people gain great performance improvement on GPU over CPU. So a totally new technique called **GPGPU** (General Purpose computation on GPU) emerges. And in recent years it became a hot research topic, not only in computer graphics domain, but also in other discipline areas.

II. GPU AND CPU

The graphics processing units (GPU) are highly parallel rapidly gaining maturity as a powerful engine for computationally demanding applications. The GPU's performance and potential will be the future of computing systems. A GPU is basically designed for some particular type of applications with the following characteristics.

- Where Computational requirements are large: GPU must deliver an enormous amount of compute power to cover the requirements of complex real-time applications.
- Parallelism is significant: The graphics

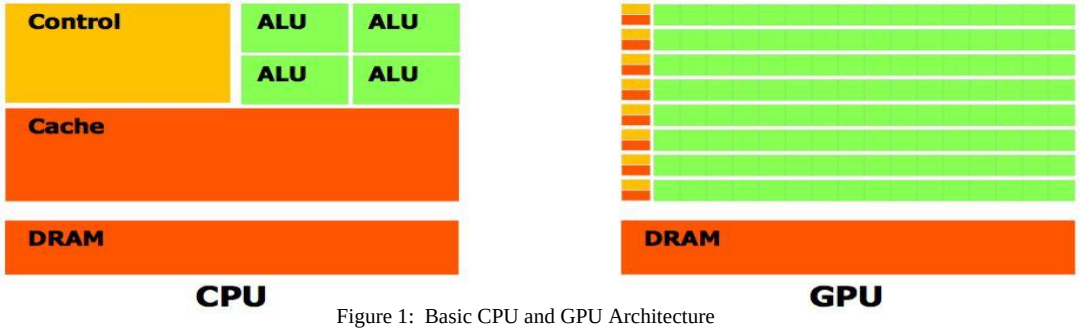

Figure 1: Basic CPU and GPU Architecture

pipeline system architecture is suitable for parallelism.

Few years ago, GPU's were some fixed function processors, built over the three dimensional (3D) graphics pipeline and with very little else to offer. But now, the GPU has evolved into a powerful programmable processor, with both application programming interface (APIs) and the hardware increasingly focusing on the programmability aspects of the GPU. The result is a processor with enormous arithmetic capability and streaming memory bandwidth, both substantially greater than a high-end CPU. [5]

As shown in fig.1 [3], on comparing the GPU with CPU the basic difference is; CPU has few processing units with some cache and control unit, but in GPU there are many more processing units with their own cache and control units with dedicated and specific works defined for them. GPUs are mostly with hundreds of cores which work in parallel to process the data, but in general CPUs processing is done on few cores with very little parallelism.

On architectural comparison with CPU, GPU are more suitable for stream computations, they can process data elements in parallel with SIMD & MIMD capability. So a new technique called **GPGPU** (**G**eneral **P**urpose computation on GPU) emerged and in recent years has become a hot research topic in not only graphics domain but in general computations.[2][5]

III. GPGPU (General Purpose GPU)

GPGPU is a combination between hardware components and software that allows the use of a traditional GPU to perform computing tasks that are extremely demanding in terms of processing power. Traditional CPU architectures available on the market cannot satisfy the processing demands for these specific tasks, and thus the market has moved on to GPGPU in order to achieve greater efficiency.

Few benefits of using a GPU for general purpose processes (GPGPU): [2] [5]

- Large performance benefits in many parallel coded applications. In some situations the GPU clearly performs better compared to a traditional CPU-based high performance computer.
- Purchase price: the prices of GPUs are somewhat similar to the market price of CPUs. This is a large advantage GPGPU has. In some cases, it would take multiple CPUs to match the performance of a GPGPU system. This means that in terms of cost, the GPGPU is a smarter choice.
- Technology refresh rate: GPU manufacturers develop new GPUs with a refresh rate that is much faster compared to that of the CPU market. The advantage the GPU has in this case is rather obvious, as its core technology is updated more frequently than that of the CPUs'.

From the Fig. 2 [2], it is clear that the floating-point operations per second on GPU very much exceed that of CPU. In other words, the computation power of GPU is stronger than that of CPU. The computation power is more reachable than other hardware, and in terms of the computation cost, GPU for per GFLOPS is much lower than CPU.

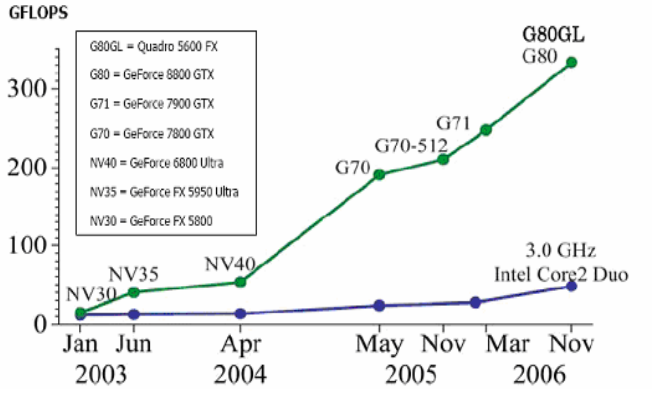

Figure 2: Floating-point operations on CPU and GPU (from NVIDIA)

IV. Architecture of GPU

The architecture of the GPU has progressed in a different direction than that of the CPU. Consider a pipeline of tasks that processes a large number of input elements, the output of each successive task is fed into the input of the next task. Data in multiple pipeline stages can be computed at the same time; that is pipeline shows the task parallelisms. As data in multiple pipeline stages can be computed at the same time; computing more than one element at the same time is data parallelism. To execute such a pipeline, a CPU would take a single element (or group of elements) and process the first stage in the pipeline, then the next stage, and so on. The CPU divides the pipeline in time, applying all resources in the processor to each stage in turn. GPU divides the resources of the processor among the different stages, such that the pipeline is divided in space, not time. The part of the processor working on one stage feeds its output directly into a different part that works on the next stage. In brief, there are many hundred cores on a GPU system; cache memory but with no cache coherency. (Fig. 3)

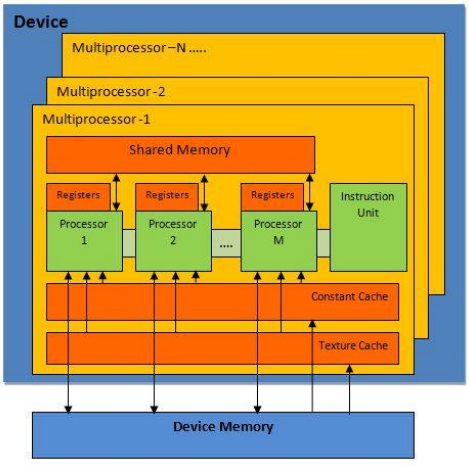

Figure 3: GPU Architecture

V. Programming Model for GPU

The programming over GPU follows a single instruction multiple-data (SIMD) programming model. For efficiency, the GPU processes many elements in parallel using the same program. Each element is independent from the other elements, and in the base programming model, elements cannot communicate with each other. All GPU programs must be structured in this way: many parallel elements each processed in parallel by a single program.

NVIDIA has developed **C**ompute **U**nified **D**evice **A**rchitecture (CUDA) which allows the use of the C programming language to code algorithms to execute on the GPU. CUDA enabled GPUs include data parallel cache. Besides the flexible interface for programming, it also supports memory scatter bringing more flexibilities to GPU.

CUDA (Compute Unified Device Architecture)

CUDA provides a C-like syntax for executing on the GPU and compiles offline. CUDA exposes two levels of parallelism, data parallel and multithreading. CUDA also exposes multiple levels of memory hierarchy: per-thread registers, fast shared memory between threads in a block, board memory, and host memory. Kernels in CUDA allow the use of pointers, general load/store to memory allowing the user to scatter data from within a kernel, and synchronization between threads in a thread block. However, all of this flexibility and potential performance gain comes with the cost of requiring the user to understand more of the low-level details of the hardware, notably register usage, thread and thread block scheduling, and behavior of access patterns through memory. All of these systems allow developers to more easily build large applications. [6][7]

VI. NLP Application

All the techniques of parsing in Natural Language Processing involve string matching as the single most important operation. We are using Lexical Analysis and Shallow Parsing as the NLP application to be implemented on single-core and multi-core CPU system (using OpenMP as programming model) and also on GPU system (using CUDA as their programming model).

As per the algorithm used, system performs lexical analysis and shallow parsing on a text input file in English language. To perform so, algorithm matches it with the knowledge base which is having complete list of possible words in English with their part of speech. The processing time for such input text file is high because of the size of the knowledge base which contains a huge amount of data to be processed for each and every word that is encountered by the algorithm from the input text file.

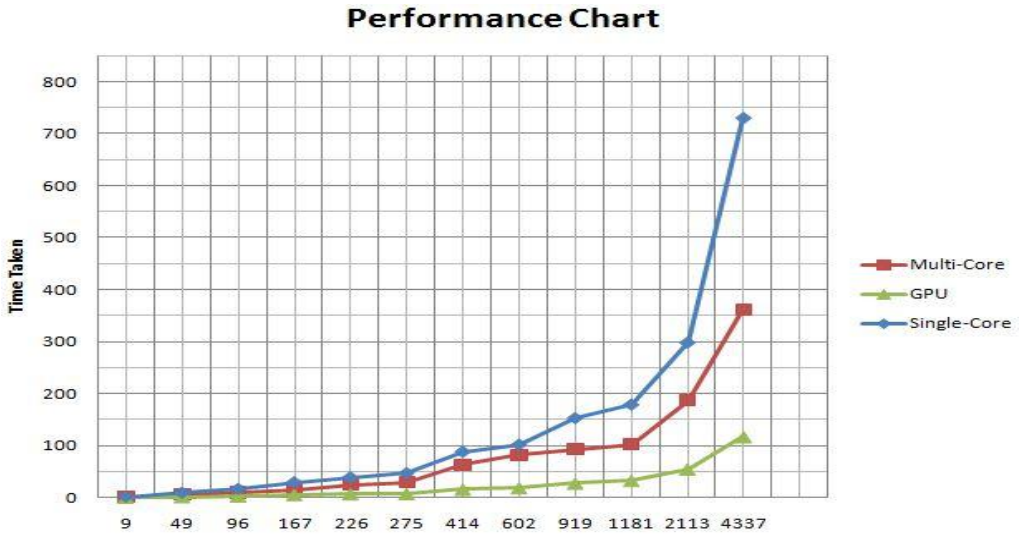

Figure 4: Performance Comparison Graph

VII. Performance Analysis

For the evaluation, we use the following platforms.

- NVIDIA GeForce G210M 1024MB (800MHz GDDR3, 16cores)
- CUDA version 2.1
- OpenMP
- Intel Core 2 Duo CPU P8600 (2.40GHz, 2CPUs) and Intel C++ Compiler 10.0

As per the algorithm used, for some particular size of file (in terms of number of words in a file) our system processes input file (performs lexical analysis and shallow parsing) and finally provide us with the number of matches and number of part of speech in the provided input file.

TABLE I. Table of Experimental Results (in seconds)

no. of Words	**Time Taken**		
	Single-Core	Multi-Core	GPU
9	1.4538	0.9125	0.2662
49	8.945	5.684	1.5735

no. of Words	Time Taken		
96	16.376	9.9218	2.8652
167	29.327	15.6598	5.627
226	38.631	24.6598	6.936
275	46.561	29.3627	7.5648
414	87.871	63.4374	16.0373
602	101.6373	81.9269	18.6092
919	152.655	93.2981	27.514
1181	178.5342	102.2638	32.7635
2113	297.8587	186.8143	54.8736
4337	730.539	361.6552	117.272

Some of the results generated by the NLP algorithm used on processing input file of certain size (in terms of number of words) are shown in the above table (table 1).

The graph generated on some of the data generated (using data from the table given) on the implementation of algorithm used is displayed above (fig.4).

VIII. CONCLUTION

We have compared the performance of GPU with single-core and multi-core CPU (2cores) for a basic NLP application (lexical analysis and shallow parsing). The number of cores in Nvidia GeForce G210M is 16. As the results from the table (table 1) and the graph (fig.4) generated, shows that multi-core CPU has better performance than the single-core CPU but a GPU system has clearly overtaken them with much better performance over CPU for Natural Language Processing (NLP) applications. For the future enhancements, this algorithm can be improved and implemented on programmable GPGPUs more efficiently to give even improved performance.

REFERENCES

[1] Shuichi Asano, Tsutomu Maruyama and Yoshiki Yamaguchi, University of Tsukuba, Japan; "Performance Comparison of FPGA, GPU and CPU in Image Processing", IEEE-FPL 2009, pp. 127-131.

[2] Enhua Wu, University of Macau; Youquan Liu, Chinese Academy of Sciences, China; "Emerging Technology about GPGPU", Circuit and Systems, 2008.APCCAS 2008. IEEE.

[3] Vadali Srinivasa Murty, P.C.Reghu Raj and S.Raman, Department of Computer Science and Engineering, Indian Institute of Technology Madras (IITM); "Design of Language-Independent Parallel String Matching unit for NLP", 2003 IEEE International Workshop on Computer Architectures for Machine Perception (CAMP).

[4] Mikhail Smelyanskiy and et al. ; "Mapping High-Fidelity Volume Rendering for Medical Imaging to CPU, GPU and Many-Core Architectures", IEEE Transactions on Visualization and Computer Graphics, Vol. 15, Dec.2009.

[5] John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips; "GPU Computing", Proceedings of the IEEE Vol. 96, No. 5, May 2008.

[6] Nvidia CUDA; "nVIDIA CUDA C Programming Guide", Version 4.0

[7] John Stratton et al., University of Illinois; "Compute Unified Device Architecture Application Suitability", Computer Science and Engineering, University of Illinois; IEEE Computer Society & American Institute of Physics.

AUTHORS PROFILE

Shubham Gupta is Pursuing Master of Technology in Computer Science and Engineering (2009-11) from VIT University, Vellore, India. He did his B.E. (Computer Science and Engineering) from Swami Keshwanand Institute of Technology (Rajasthan University), Jaipur, India (2005-2009). His research interest includes Natural Language Processing, Multi-core Architectures and GPU programming.

Prof. M.Rajasekhara Babu a senior faculty member at the School of Computer Science and Engineering ,VIT University ,Vellore ,India . At, present he is pursing Ph.D from the same University in the area of "Multi-Core Architectures and Natural Language Processing". He had authored a number of national and international papers and articles in reputed journals and conferences