Volume 5 Issue II, February 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Performance Analysis of GPU V/S CPU for Image Processing Applications

B. N. Manjunatha Reddy[1], Dr. Shanthala S.[2], Dr. B. R. VijayaKumar[3]
[1]*Research Scholar, Dept. of TCE, Professor, Dept. of ECE, Bengaluru*
[2]*Professor & Head, Dept. of TCE Bangalore Institute of Technology, G.A.T., Bengaluru.*
[3]*Professor& Dean, Global Academy of Technology*

Abstract: ***Image processing grosses much more time to perform the convolution in image filtering on CPU, since the computation demand of image filtering is enormous. Contrast to CPU, Graphics Processing Unit (GPU)is a good way to accelerate the image processing. By comparison and analysis,it has reached a conclusion that GPU is appropriate for processinglarge-scale data-parallel load of high-density computing.CUDA(Compute Unified Device Architecture) is a parallel computing architecture established by NVIDIA. CUDA is highly suitable for general purpose programming on GPU which is a programming interface to use the parallel architecture for general purpose computing. This paper stressesthe possible gain in time which can be attained on comparison and analysis of GPU over CPU implementation and the research results shows that the GPU implementation can achieve a speed up of more than 60%of time in comparison with CPU implementation of image processing. GPU has great potential as high-performance co-processor.***
Keywords — GPU, Parallel Computing, CUDA, Speedup, Image Processing, MATLAB.

I. INTRODUCTION

Many realistic applications in the areas such as digital image processing, Pattern Recognition, analysis of programming languages etc., the speed is a significant factor. Since huge sum of data is to be processed, efficient low complexity algorithms are required. Image processing and filtering application has usually been a time and resource consuming task in the presence of more and more pixels as the image size increases. A sequential image processing approach has been replaced by a parallel programming approach commonly done through a CPU as the number of available cores has increased [1].

However, the speedup factor is limited to the number of cores present in the CPU. The presence of the multithreaded parallel programming capabilities provided by CUDA opened a door to increase the speedup factor by hundreds to thousands, limited by the number of cores in the GPU. NVIDIA CUDA architecture offers an efficient means to access massively-parallel threaded GPUs to achieve a higher degree of performance and energy efficiency. This paper focuses on performance comparison of CPU and GPU on some image processing applications.

A. Gpu and cpu

The graphics processing units are extremely parallel, rapidly gaining maturity as a powerful device for computationally demanding applications. The GPU's performance and potential will be the future of computing systems. A GPU is mainly designed for some particular type of applications with the following characteristics;

WhereComputational requirements are large: GPU must deliver an enormous amount of computing power to cover the requirements of complex real-time applications.

Parallelism is significant: The graphics pipeline system architecture is appropriate for parallelism.

Few years ago, GPUs were some fixed function processors built for three-Dimensional (3D) graphics. But now, the GPU has evolved into a powerful programmable processor, with both application programming interface (APIs) and the hardware focusing on the programmability aspects of the GPU. The result is a processor with huge arithmetic capability and streaming memory bandwidth, both substantially greater than a high-end CPU [3].

Volume 5 Issue II, February 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

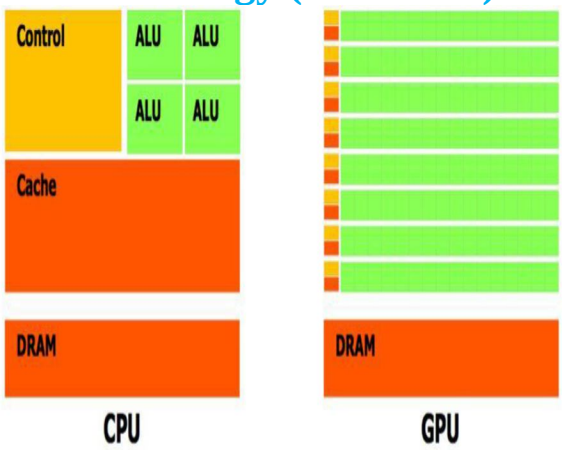

Figure 1: Basic CPU and GPU Architecture

As shown in figure1 [2], on comparing the GPU with CPU the basic difference is; CPU has few processing units with some cache and control units, 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.

The architectural comparison of CPU with GPU is 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 also in general computations[3][4].

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[9]. 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.

B. Architecture of GPU

The research carried out on NVIDIA based GPU hardware using CUDA, a general purpose parallel computing architecture. The architecture of the GPU has developed in a different direction than that of the CPU. The design of the GPUs is forced by the fast growing video game industry that exerts marvellous economic pressure for the ability to perform a massive number of floating-point calculations per video frame in advanced games. The general philosophy for GPU design is to optimize for the execution of huge number of threads. Figure 2 shows the architecture of a typical GPU today. It is organized into 16 highly threaded streaming Multiprocessors (SMs). A pair of SMs forms a building block. Each SM has 8 streaming processors (SPs), for a total of 128 (16*8) SPs. Each SP has a multiply-add (MAD) unit and an additional multiply (MUL) unit. Each GPU currently comes with 4 megabytes of DRAM. These DRAMs differ from the system memory DRAMs on the motherboard in that they are essentially the frame buffer memory that is used for graphics. For graphics applications, they hold high-definition video images, and texture information for 3D rendering as in games. But for computing, they function like very high bandwidth off-chip cache, though with somewhat more latency regular cache or system memory.

C. Programming model for gpu

The programming in 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.

NVIDIA has developed CUDAwhich allows the use of the C programming language to code algorithms to execute on the GPU. CUDA enabled GPUs comprise data parallel cache. Besides the flexible interface for programming, it also supports memory scatter bringing more flexibilities to GPU.

Volume 5 Issue II, February 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

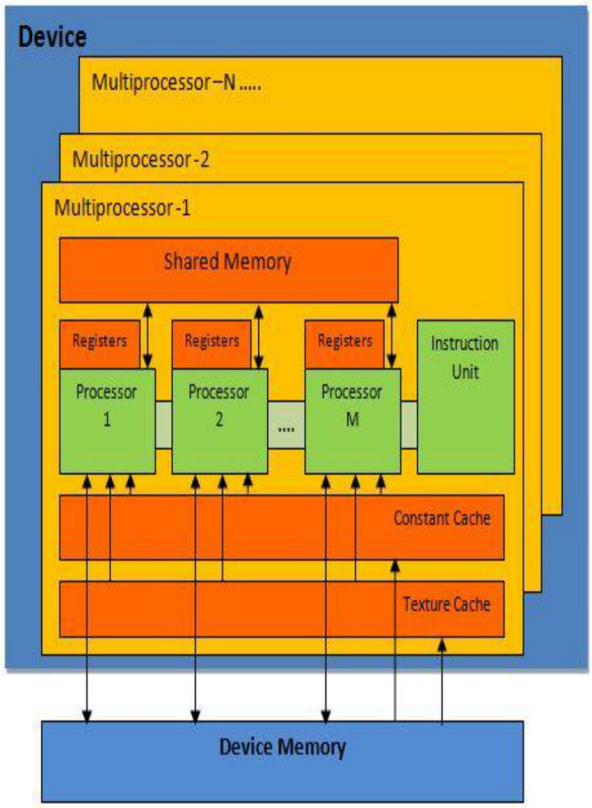

Figure 2: GPU Architecture

CUDA offers a C-like syntax for executing on the GPU. 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 permit 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 schedulingand behaviour of access patterns through memory. All of these systems allow developers to more easily build large applications. [5][6]

D. Experimental results

In order to relate the performance of GPU with CPU implementations, the experimental Settings are as follows:
CPU:Intel(R) Core (TM) i5-4590 at 3.30 GHz and 4GB of memory;
GPU: NVIDIA GeForce 970 GTXwith 1664CUDA cores,4GB of memory, 1050MHZ clock and CUDA version of 2.1.
System: Windows 7with 64 bit OS.
Edge detection is one of the most important paradigms of Image processing[7]. Images comprise millions of pixel and each pixel information is independent of its neighbouring pixel. More specifically, this paper focuses on Compute Unified Device Architecture as its parallel programming platform and observes the possible gain in time which can be attained for edge detection in images of pixel size of 1800 x 1200. A well-known algorithm SOBEL [8] for edge detection is used in this research work.
Event timers have been implemented to calculate the time taken for execution of each section of the code. These event timers show the speedup is achieved for the actual work by GPU and CPU.
In his paper,edge detection of the image is computedon both GPU and CPU to test the performance of GPUv/s CPU. Figure 3 shows the sceen shot of the code processed on both GPU and CPU.

Volume 5 Issue II, February 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

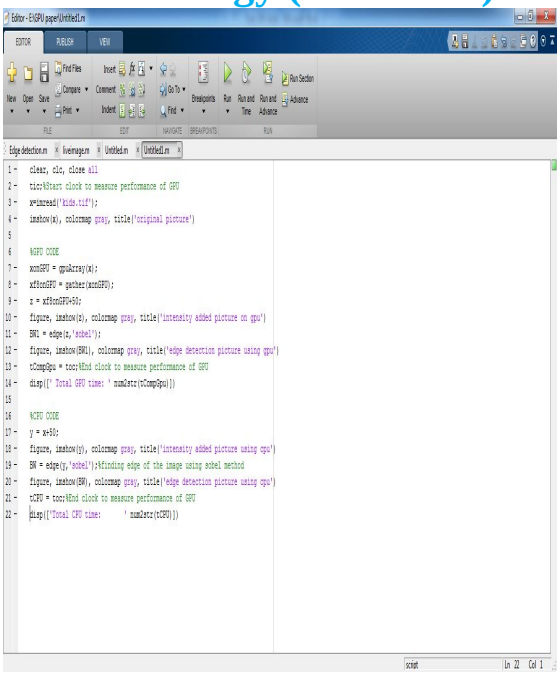

Figure 3: Screen shot of the code.

The results for the image processing and filtering of both GPU and CPU execution times are given as follows and the screen shot is shown in figure 4.

Total GPU time: 0.19718

Total CPU time: 0.3254

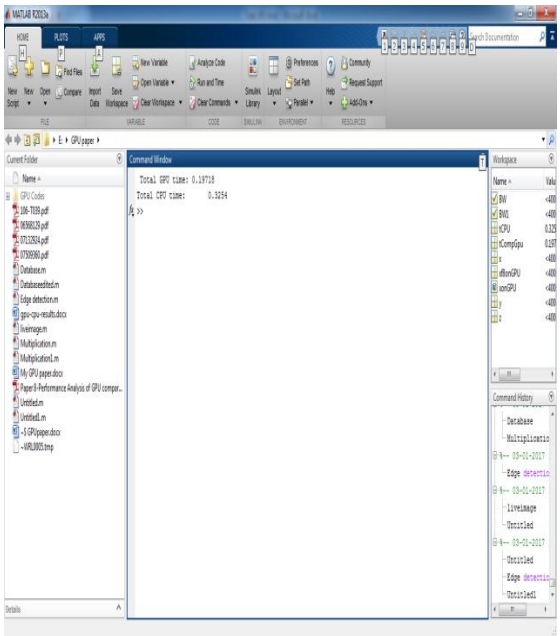

Figure 4: Screen shot of the processing time.

The result shows that the execution on GPU can get a speedup of about 61% as compared with the filtering implementation on CPU. The results of the edge detection of an image and the intensity added images processed on both CPU and GPU are shown in figures 5a to 5e.

Volume 5 Issue II, February 2017
ISSN: 2321-9653

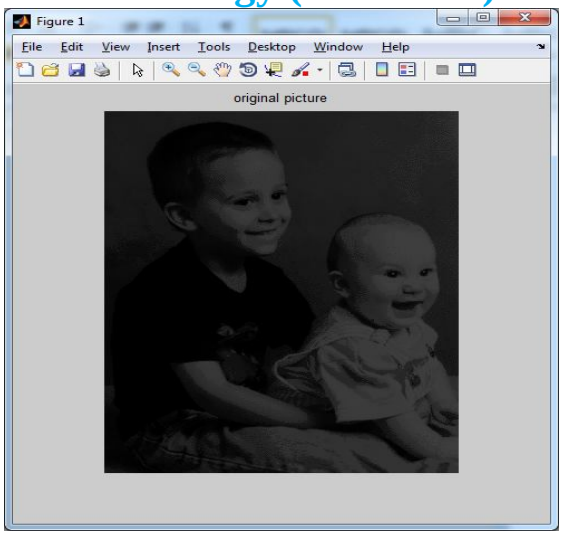

Figure5a: Original image.

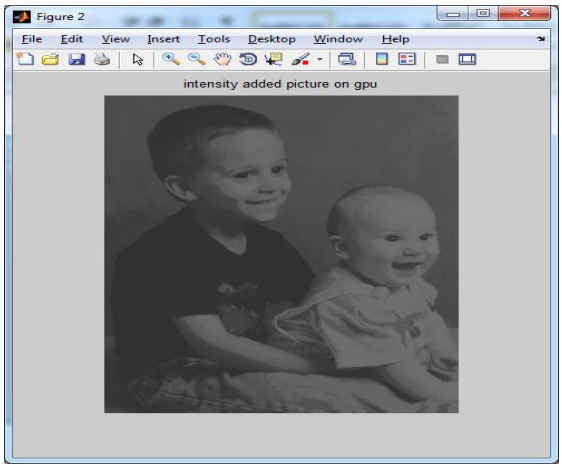

Figure5b: Intensity added image in GPU.

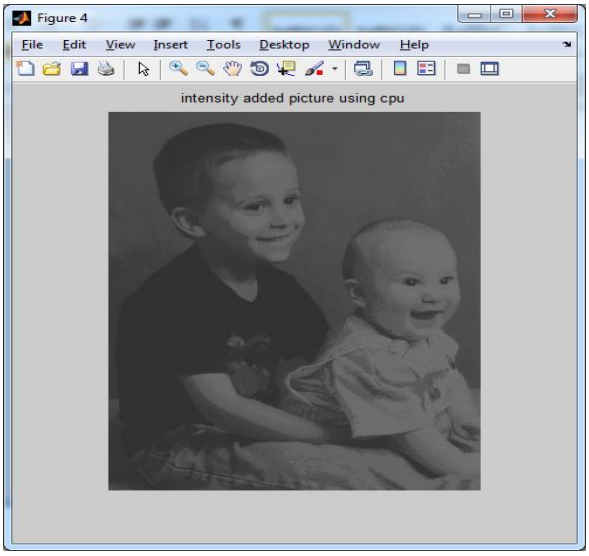

Figure5c: Intensity added image in CPU.

Volume 5 Issue II, February 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

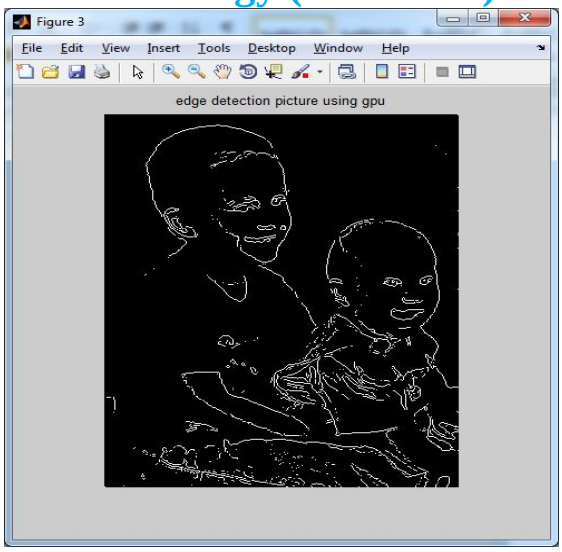

Figure5d: Edge detection image in GPU.

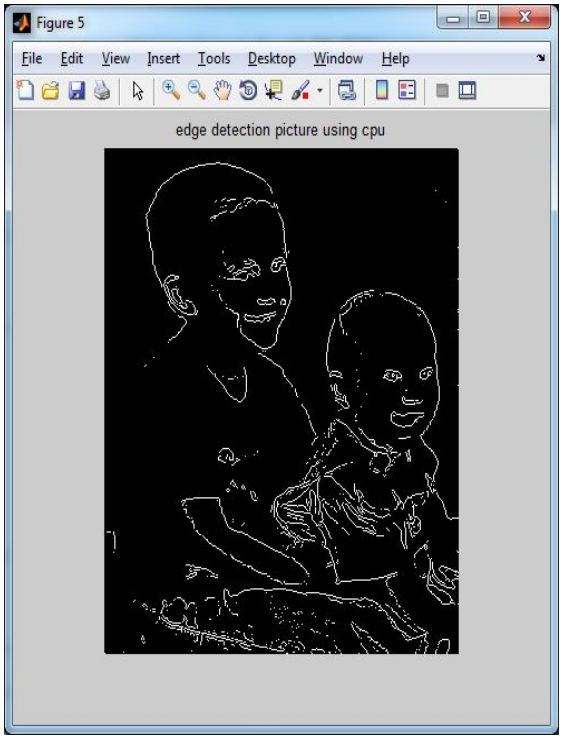

Figure5e: Edge detection image in CPU.

II. CONCLUSION

GPU-based parallel computing has potential to process images very fast. In this work, the impact of CUDA-accelerated GPU and CPU computing on image processing performance is analysed. Image processing and filtering through sequential C and parallel CUDA programs are implemented. CUDA Events are used to measure the GPU execution time. According to the research results, image processing and filtering is more efficient when done through CUDA programming on GPU. This is because computations are done simultaneously inparallel by fully exploiting the available processing resources to increase speedup.In this work, speedup of up to 62%has achieved for the given image.

A future extension of this work is to consider overhead caused by the CUDA copy operation between the host and the devices that may decrease the speedup.

Volume 5 Issue II, February 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

REFERENCES

[1] E. Young and F. Jargstorff, "Image processing and video algorithms with CUDA," 2008.
[2] 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.
[3] 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.
[4] Enhua Wu, University of Macau; Youquan Liu, Chinese Academy of Sciences, China; "Emerging Technology about GPGPU", Circuit and Systems, 2008. APCCAS 2008. IEEE.
[5] Nvidia CUDA; "NVIDIA CUDA C Programming Guide", Version 4.0
[6] John Stratton, "Compute Unified Device Architecture Application Suitability", Computer Science and Engineering, University of Illinois; IEEE Computer Society & American Institute of Physics.
[7] Mikhail Smelyanskiy "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.
[8] Sanjanaashree p, "Accelerating encryption/decryption using GPU's for AES algorithm", International Journal of Scientific & Engineering Research, volume 4, issue 2, february-2013 ISSN 2229-5518
[9] Karthik Balasubramanyam, Prabhu, P., Jablin, J., Johnson, N., Beard, S., and august, D. "Automatic CPU-GPU communication management and optimization", In Proc. of ACM Conference on Programming Language Design and Implementation (2014).