

Review on GPU in MATLAB as MATCUDA

Santosh Kumar Sahu¹, Chetan Pise², Rahul Sathawane³, Sandip Kamble⁴

¹Asst. Prof., Dept. of IT, RGCER, Nagpur Maharashtra, INDIA

²Asst. Prof., Dept. of CT, YCCE, Nagpur, Maharashtra, INDIA

³Asst. Prof., Dept. of IT, RGCER, Nagpur Maharashtra, INDIA

⁴Asst. Prof., Dept. of CT, RGCER, Nagpur Maharashtra, INDIA

Abstract: Nowadays GPU is very frequent word in the market for tremendous computing within a short span of time. GPU means Graphics Processing Unit; it is a co-processor of CPU. The establishment of co-processor (GPU) is nearer to the Processor (CPU). This High Performance supercomputing have found the technology cost effective in domains as diverse as seismic imaging, electromagnetics, molecular dynamics, financial valuation, medical imaging and others, so it is called as General Purpose Graphical Processing Unit (GPGPU).

MATLAB means MATrixLABoratory is a high-level programming tool and user-friendly environment makes to write technical code. MATLAB is extensively used in a number of scientific fields, such as mathematics, digital signal processing, digital image processing. MATLAB uses an interpreter which slows down the processing, especially while executing loops. This becomes a performance bottleneck in programs that make excessive use of loops. In the field of digital image processing, for example, the operations performed on a matrix usually involve many nested loop structures. To accelerate MATLAB's processing, we can refer NVIDIA's CUDA(Compute Unified Device Architecture) parallel processing architecture. We mentioned that interfacing MATLAB with CUDA and parallelizing the most time-consuming portions of MATLAB's code, the processing can be speeded up significantly.

Keywords: CUDA, GPU, MATLAB, MATCUDA, NVIDIA.

1. Introduction

Parallel programming is a generic concept describing a range of technologies and approaches. However in general it describes a system whereby threads of instruction are executed truly in parallel over a shared or partitioned data source. General Purpose computation on Graphics Processing Units (GPGPU) is a new and active field. The main goal in GPGPU is to find parallel algorithms capable of processing concurrently huge amounts of data over a number of Graphic Processing Units (GPU). GPGPU involves using the advanced parallel Graphics Processing Unit devices now readily available for general purpose parallel programming. [1]

GPU computation has provided a huge edge over the CPU with respect to computation speed. Hence it is one of the

most interesting areas of research in the field of modern industrial research and development.

CPU it is a single-chip processor and the GPU has hundreds of cores as compared to the 4 or 8 in the latest CPUs. The primary job of the GPU is to compute 3D functions. Because these types of calculations are very heavy on the CPU, the GPU can help the computer run more efficiently. Though, GPU came into existence for graphical purpose, it has now evolved into computing, precision and performance. [2]

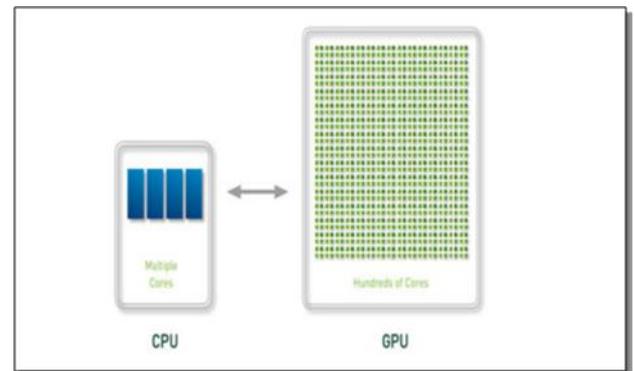


Figure 1: Comparison between CPU and GPU [2]

The evolution of GPU over the years has been towards a better floating point performance. NVIDIA introduced its massively parallel architecture called "CUDA" in 2006- 2007 and changed the whole outlook of GPGPU programming. The CUDA architecture has a number of processor cores that work together to munch the data set given in the application. GPU computing or GPGPU is the use of a GPU (graphics processing unit) to do general purpose scientific and engineering computing. The model for GPU computing is to use a CPU and GPU together in a heterogeneous co-processing computing model. The sequential part of the application runs on the CPU and the computationally-intensive part is accelerated by the GPU. From the user's point of view, the application is faster because it is using the better performance of the GPU to improve its own performance.

Now a days CPU is capable of crunching more numbers than before. Still processing huge data or crunching large numbers puts a lot of burden on CPU. This can be done powerfully using Graphics Processing Unit. If we do complex calculations using GPU then we can free CPU time

cycles which can be used for other higher priority tasks. [1, 4]

On the other hand, MATLAB is a powerful tool for prototyping and analysis. MATLAB could be easily extended via MEX files to take advantage of the computational power offered by the latest NVIDIA graphics processor unit (GPU). [3]

2. MATLAB?

MATLAB (Matrix Laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages including C, C++, Java, Fortran and Python. Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. [3]

MATLAB can be used to analyze data, develop algorithms, and create models in a variety of application areas such as image and video processing, signal processing and communications, computational finance, machine learning, and computational biology.

MATLAB supports CUDA kernel development by providing a language and development environment for prototyping algorithms and incrementally developing and testing CUDA kernels. [11, 12, 13]

2.1 The Problem with MATLAB

MATLAB is a convenient but inefficient programming language of choice for scientists. There are two major problems with MATLAB as

Interpreted language

The fact that MATLAB is compiled language is not relevant to people who are using it as a user point of view but as a researcher its necessary to know that interpreted languages are often slower than compiled languages. The interpreted languages are optimized by JIT compiler which is approximately 2x slower than C

Most of the existing code and libraries are single-threaded

Since most of the MATLAB code is executed on a single thread, it is unable to harness the processing power of multiple cores which can allow us to process information and produce results at a much faster rate.

3. CUDA

CUDA (Compute Unified Device Architecture) is NVIDIA's GPU architecture featured in the GPU cards, positioning

itself as a new means for general purpose computing with GPUs. CUDA C/C++ is an extension of the C/C++ programming languages for general purpose computation. CUDA gives advantage of massive computational power to the programmer. This massive parallel computational power is provided by Nvidia's graphics cards.[5]

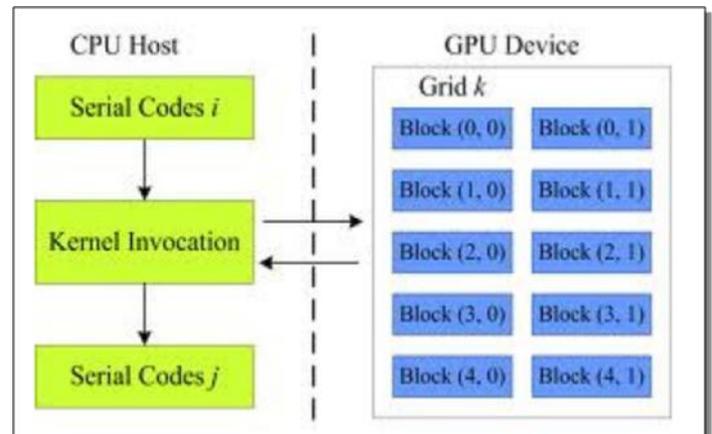


Figure 2: Flow of execution of GPU [5]

CUDA provides 128 co-operating cores. For running multithreaded applications there is no need of streaming computing in GPU, because cores can communicate also can exchange information with each other. CUDA is only well suited for highly parallel algorithms and is useful for highly parallel algorithms. If you want to increase performance of your algorithm while running on GPU then you need to have many threads. Normally more number of threads gives better performance. For the most of the serial algorithms, CUDA is not that useful. If the problem cannot be broken down into at least a thousand threads then using CUDA has no overall advantage. In that case we can convert serial algorithm into parallel one but, it is not always possible. As mentioned above to get best optimization you need to divide your problem into minimum thousand threads. Then performance of algorithm increases rapidly.

One advantage is that there is no need to write the whole programme using CUDA technology. If you are writing a large application, complete with a user interface, and many other functions, and then most of your code will be written in C++ or whatever your language of choice is. When you really want to do large mathematical computations then, you can simply write kernel call to call CUDA functions you have written. In this way instead of writing complete programme you can use GPU for some portion of the code where we need huge mathematical computations. [11, 12, 13]

3.1 Basic units of CUDA

CUDA Architecture splits the device into grids, blocks and threads in a hierarchical structure as shown in Figure 3. Since there are a number of threads in one block and a number of blocks in one grid and a number of grids in one GPU, the parallelism that is achieved using such a hierarchical architecture is immense.

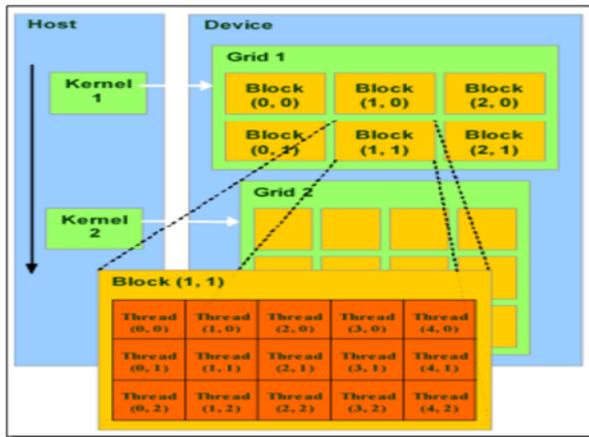


Figure 3: Units of CUDA [2]

4. GPU Acceleration using MATLAB

Using MATLAB for GPU computing lets you accelerate your applications with GPUs more easily than by using C or Fortran. With the familiar MATLAB language you can take advantage of the CUDA GPU computing technology without having to learn the intricacies of GPU architectures or low-level GPU computing libraries.

You can use GPUs with MATLAB through Parallel Computing Toolbox, which supports:

1. CUDA-enabled NVIDIA GPUs with compute capability 2.0 or higher. For releases 14a and earlier, compute capability 1.3 is sufficient.
2. GPU use directly from MATLABGPU-enabled MATLAB functions such as fft, filter, and several linear algebra operations
3. GPU-enabled functions in toolboxes: Image Processing Toolbox, Communications System Toolbox, Neural Network Toolbox, Phased Array Systems Toolbox, and Signal Processing Toolbox
4. CUDA kernel integration in MATLAB applications, using only a single line of MATLAB code
5. Multiple GPUs on the desktop and computer clusters using MATLAB workers in Parallel Computing Toolbox and MATLAB Distributed Computing Server

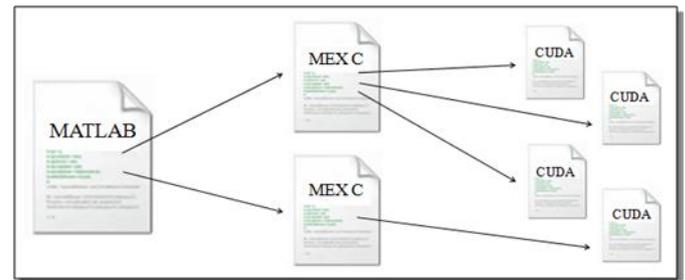


Figure 4: Implementing MEX files using CUDA

4.1 Implementation using CUDA

The Figure 4 depicts how MATLAB functions are accelerated using GPU. We first create the MEX(MATLAB EXE) files which are later converted into relative CUDA files. These files are then executed on the CUDA capable GPU and then the results are returned to the CPU. This process is much more efficient and streamlined as compared to other approaches. The same is shown in the figure 6 below.

It is a two-step process where we proceed in the following order:

Translation:

- convert MATLAB to C

Parallelization:

- C for multi-core CPU
- CUDA for GPU

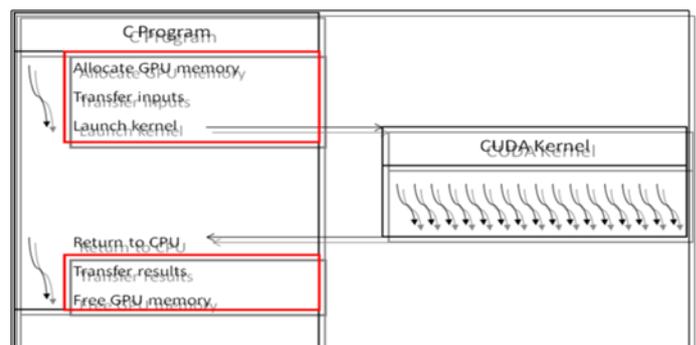


Figure 5: Implementation on CPU and GPU end

5. Applications

CUDA provided benefit for many applications implemented with MATLAB. Here list of some:

1. Seismic Database - 66x to 100x speedup
2. Molecular Dynamics - 21x to 100x speedup
3. Medical Imaging- CUDA programming is effectively used in the medical field for image processing. Using CUDA, MRI machines can now compute images faster than ever possible before.- 245x to 415x
4. Atmospheric Cloud Simulation - 50x speedup.

5. Fast video transcoding -raw computing power of GPUs can be harnessed in order to transcode video much faster than ever before.

Video Enhancement-Complicated video enhancement techniques often require an enormous amount of computations. Ex: ArcSoft was able to create a plug-in for its movie player which uses CUDA in order to perform DVD up scaling in real time!

And many more applications in different domains are present. The figure below shows how efficient is CUDA in varied applications:

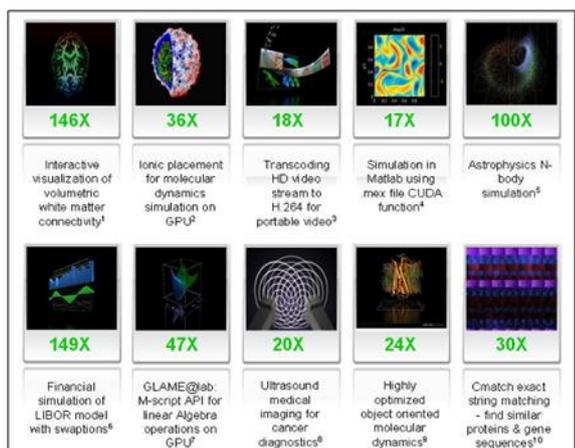


Figure 6: Speedups using GPU as compared to CPU [6]

6. Conclusions:

After the comparison between CUDA and the other paradigms of parallel computing and its application in numerical computations for image processing using MATLAB, it is clear that the future of parallel processing is very much in the hands of the Nvidia's.

The future work could be aimed to extend the set of applications to cover even more areas of image and video processing with MATLAB and CUDA.

References:

[1].ChetanPise, ShailendraShende, "NVIDIA Graphics Card for Parallelization of BFS Graph Algorithm using CUDA," Elsevier Proceedings of 3rd International Conference on Recent Trends in Engineering & Technology (ICRTET'2014).

[2].RajdeepKaur, "Survey on Medical Image Registration using Graphics Processing Unit," International Journal of Advanced Research in Computer Science and Software Engineering ISSN: 2277 128X, Volume 5, Issue 9, September 2015.

[3].MassimilianoFatica, Won-Ki Jeong, "Accelerating MATLAB with CUDA," By NVIDIA.

[4].ChetanPise, ShailendraShende, "Parallelization of Graph Algorithms on GPU using CUDA,"IJCAT International Journal of Computing and Technology, ISSN : 2348 - 6090, Vol. 1, Issue 3, April 2014.

[5].PreetiKaur, "Implementation of image processing algorithm onthe parallel platform using MATLAB," International Journal ofComputer Science & Engineering Technology (IJCSET), ISSN:2229-3345, Vol. 4 No. 06 Jun 2013.

[6].Sarah Tariq, "An Introduction to GPU Computing and CUDAArchitecture,"NVIDIA Corporation 2011.

[7].Dan Connors, "Exploring Computer Vision and Image Processing Algorithms in Teaching Parallel Programming," Department of Electrical Engineering University of Colorado Denver.

[8].Rafael C. Gonzalez, Richard E. Woods, "Digital ImageProcessing," University of Tennessee, MedData Interactive.

[9].Nvidia: <https://developer.nvidia.com/cuda-zone>

[10]. AntoninoTumeoPolitecnico di Milano", "Massively ParallelComputing with CUDA", © NVIDIA Corporation 2008.

[11]. Book: Jason Sanders, Edward Kandrot, "Programming based on CUDA".

[12]. Book: CUDA C Programming Guide.

[13]. Book: David B. Kirk, Wen-Mei W. Hwu, "Programming Massively Parallel Processors".

BIOGRAPHIES:

	Santosh Kumar Sahu working as Asst. Prof. in Dept. of IT, RGCER, Nagpur Maharashtra, INDIA. Specialization Design and analysis of algo., Data Structure, Parallel computing.
	Chetan Pise Working as a Asst. Prof in Dept. of CT, YCCE, Nagpur, Maharashtra, INDIA. Specialization Parallel computing and GPU-CUDA.
	Rahul Sathawane working as Asst. Prof. in Dept. of IT, RGCER, Nagpur Maharashtra, INDIA. Specialization DataBase Mngt. System Computer Graphics, AI.
	Sandip Kambleworking as a Asst. Prof. in Dept. of CT, RGCER, Nagpur Maharashtra, INDIA. Specialization Programming, Data Structure, Algorithms.