International Journal of Computer Science and Network (IJCSN)
Volume 1, Issue 4, August 2012 www.ijcsn.org ISSN 2277-5420

Accelerating MATLAB Applications on Parallel Hardware

[1]Kavita Chauhan, [2]Javed Ashraf

[1] NGFCET, M.D.University
Palwal,Haryana,India

[2]AFSET, M.D.University
Dhauj,Haryana,India

Abstract

MATLAB has been used widely in scientific and engineering community. The tool provides set of commands for different set of problems. The functions or commands related to complex fields such as 3 D Image/Graphics processing have been proved very helpful for quick verification and simulations. MATLAB in fact provides functions in a plethora of other areas too. Despite the inherent convenience it provides, it comes at a cost of poor performance, which becomes a hindrances to the scientific discoveries in R & D. The goal of this work is to increase the performance of the tool. Towards this end we have used GPUs having hundreds of processing cores. We implemented several different algorithms using MATLAB on the GPU found that we got several times speed up as compared to the CPU.

Keywords: MATLAB, NVIDIA's CUDA, JACKET, Accelerating, PARALLEL HARDWARE, GPU

I. INTRODUCTION

MATLAB is the high-performance language for technical computing computation, visualization, and programming. However, MATLAB uses an interpreter which slows down the processing, especially while executing loops. In order to accelerate MATLAB's processing we are using NVIDIA's CUDA parallel processing architecture .We run MATLAB program code of different applications on CPU and GPU using JACKET and calculate the elapsed time. Here, we would find that due to the GPU, processing time reduces and we get output faster.

We are using Graphic Processing Unit(GPU) as a parallel hardware. GPU's are available at low costs. GPUs consist of large number of computing multiple
cores and individual threads are processed in these cores. These cores are called Streaming Processors
(SPs); SPs are organized into independent multiprocessor units called Streaming Multiprocessors (SMs), groups of SMs are contained within texture/processor clusters (TCPs).

In this we will take different applications and run them on CPU & GPU .Algorithm will be same for both. Some of the applications are Multiplication, FFT, Image filtering, pi calculation. Software which we needed are -CUDA SDK, CUDA toolkit, MATLAB, JACKET. Before we show how much speed of MATLAB is enhanced, we must know some terms as briefly explain below.

A. NVIDIA's CUDA

CUDA(Compute Unified Device Architecture) is a parallel computing platform and programming model. It increases the computing performance by harnessing the power of the graphics processing unit (GPU). Through a C –like programming interface, CUDA technology gives computationally intensive applications access to the latest GPUs . CUDA program can be executed on either the host (CPU) or a device such as a GPU as CUDA consists of one or more phases .

B. JACKET

Jacket is developed by AccelerEyes .It is a computing platform which enable GPU acceleration of MATLAB-based codes. Jacket allows interfacing the programs written in other languages, including C, C++, CUDA, and OpenCL with GPU. Jacket enables standard MATLAB code to run on any NVIDIA CUDA-capable GPU, from the GeForce 8400 to the Tesla C1060.

C. GFOR

In the MATLAB coding we are using GFOR/GEND. If the iterations are independent then GFOR loop is used to simultaneously launch all of the iterations of a FOR-loop on the GPU. Standard FOR-loop performs each iteration sequentially where as Jacket's GFOR-loop performs each iteration at the same time.

Some of the features and functions of Jacket which are supported within GFOR-loops are :-

- element-wise arithmetic (addition, subtraction, multiplication, division, POWER, EXP)
- FFT, FFT2, and their inverses IFFT, IFFT2
- Transpose etc.

To enhance the speed of MATLAB Processing, Graphic card is connected. We have used Nvidia Geforce GTX 480 version. Jacket software is also downloaded, as it is an important parameter. CUDA Software Development Kit (SDK) is used to detect the GPU & show its details whereas CUDA Toolkit for Compile/debugger. Once all the software are installed, we write MATLAB program for different applications & run it on CPU and GPU. To run code on GPU, we need JACKET .Jacket enables standard MATLAB code to run on any NVIDIA CUDA-capable GPU. Jacket would convert MATLAB code into CUDA processing code .As explain before that GPU offer a large number of computing cores, it will reduce the run time of different applications. Output of MATLAB applications codes are explain below.

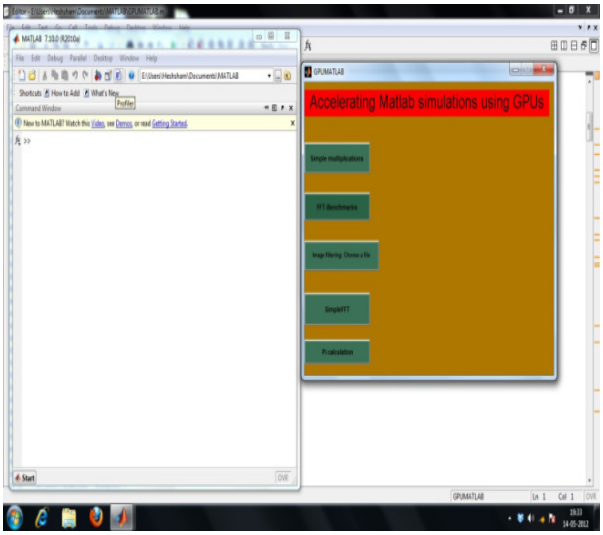
1.PI CALCULATION

Value of pi is calculated with the help of Monte Carlo Simulation. This technique is used where the underlying probabilities are known but the results are more difficult to determine. Value of pi is calculated by consider a square .One corner of square should at the origin of a coordinate system and length is 1 .Now consider inscribing a quarter of a circle of radius 1 inside of this & its area is pi/4.

Value of pi is calculated by find the relative area of the circle and square and then multiply the circle's area by 4.To find the area of circle we use following method :- for a point (X,Y) to be inside of a circle of radius 1, its distance from the origin (X^2+Y^2) will be less than or equal to 1. We can generate thousands of random (X,Y) positions and determine whether each of them are inside of the circle. Each time it is inside of the circle, we will add one to a counter. Repeat it for large number of points, the ratio of the number of points inside the circle to the total number of points generated will approach the ratio of the area of the circle to the area of the square. So the value of pi would simply be

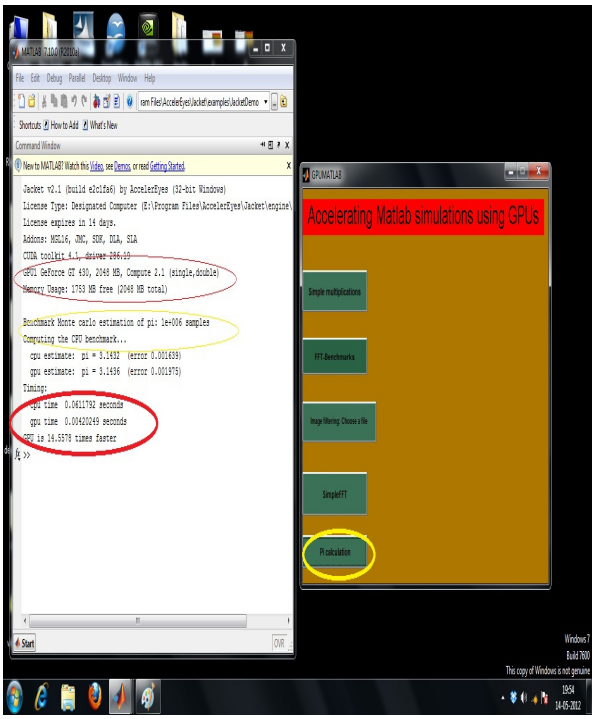
Pi = 4*(Number of Points inside circle)/ (Total Points Generated)

When the program code is run, we get output which is shown below:



Here we have taken five applications & run them on CPU & GPU to check the elapsed time.

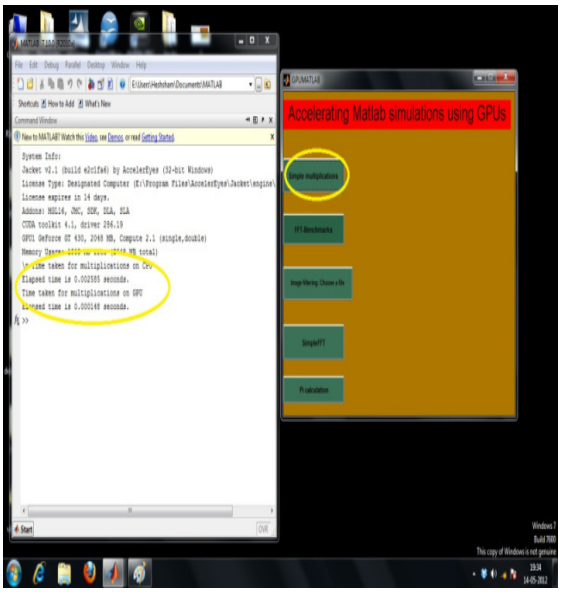
To calculate pi, select pi calculation push button .we get the information about graphic card, jacket and the most important- run time of this on CPU & GPU.



Here we can see from the output that elapsed time for calculating the value of pi on CPU is 0.0611792secs where as on GPU is 0.00420249secs.It is clear that time is reduced in GPU as compare to CPU.

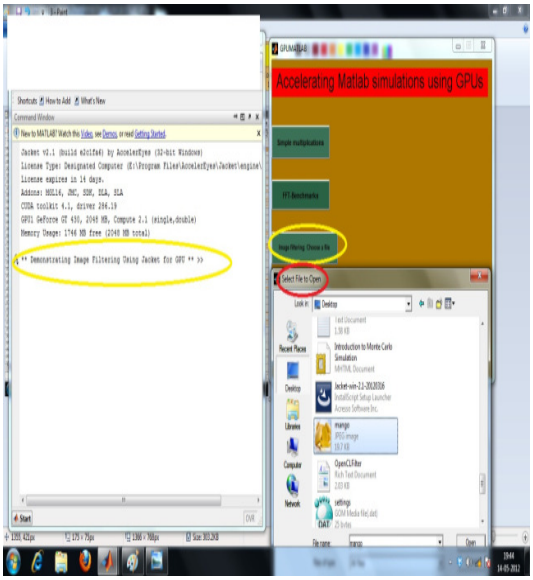
2) SIMPLE MULTIPLICATION

Here we have taken simple matrix multiplications, but for long matrix multiplication, it will be very useful. For research we need large no of calculations & if we use this method it would reduce the research time. When simple multiplication push button is selected , we get details of GPU,JACKET and also see that time taken for multiplication on GPU is 0.000148 and on CPU is 0.002488 .we have seen here that run time is reduce on GPU.

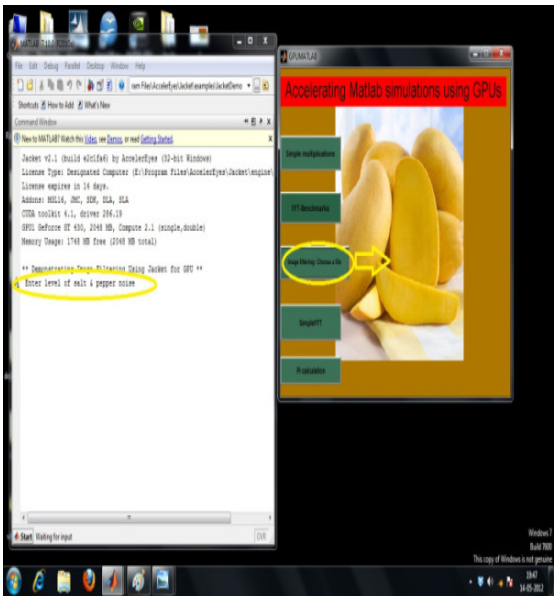


3) IMAGE FILTERING

In this application we will find out that how much time is taken in filtering/removing noise from an image.When this application is selected,we get information about gharaphic card,JACKET.Here we need to select an image as shown below.

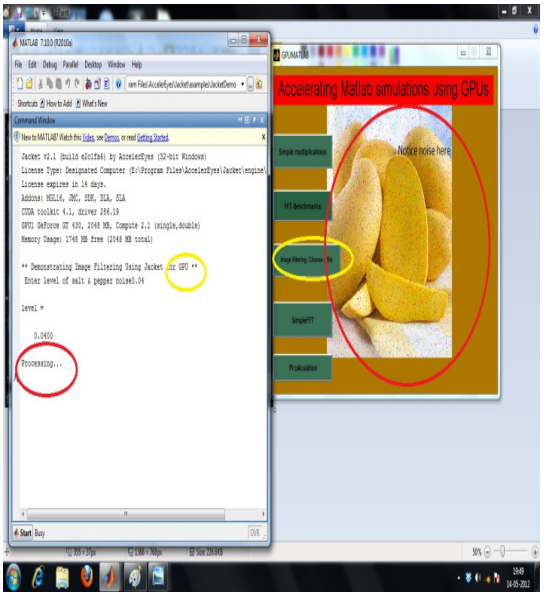


Here we selected mango image.

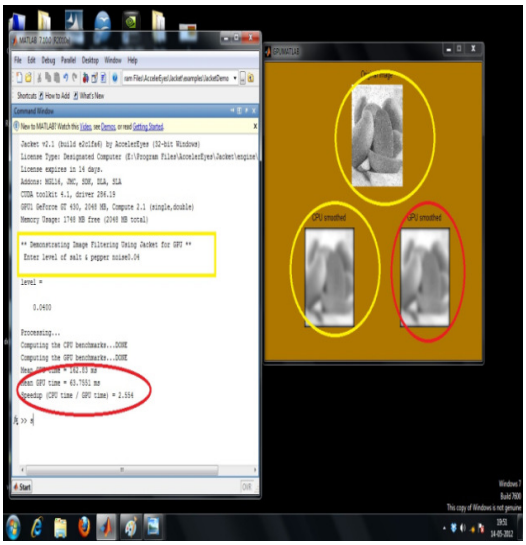


Once a file is selected, we introduced noise. Basically here we find out that how much time is needed in filtering/removing noise from an image.MATLAB code is run on both CPU & GPU, & see the time difference.

To calculate the processing time, first we need to introduce noise in an image.Here in the below figure we have introduced noise we have introduced noise



Here, Color image is converted into grey code and then, noise is removed with low pass filter .Time taken in processing on GPU & CPU and there images are shown below. We find that time taken on CPU is approximate 3 times than GPU.

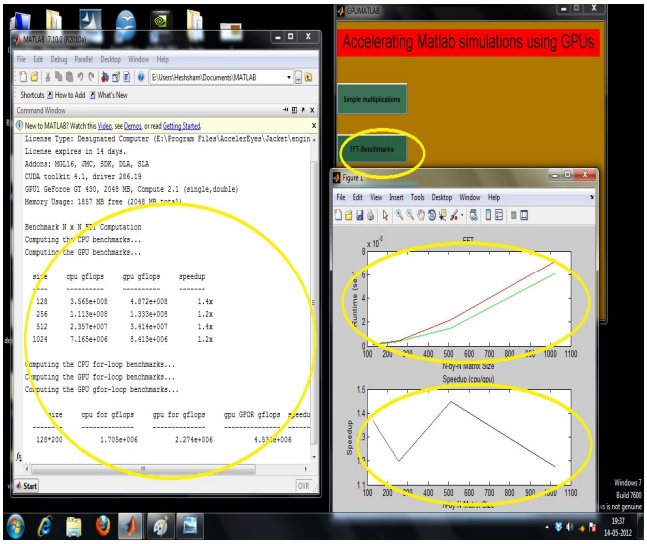


4) FFT(FAST FOURIER TRANSFORM) BENCHMARK

Fast Fourier Transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse.

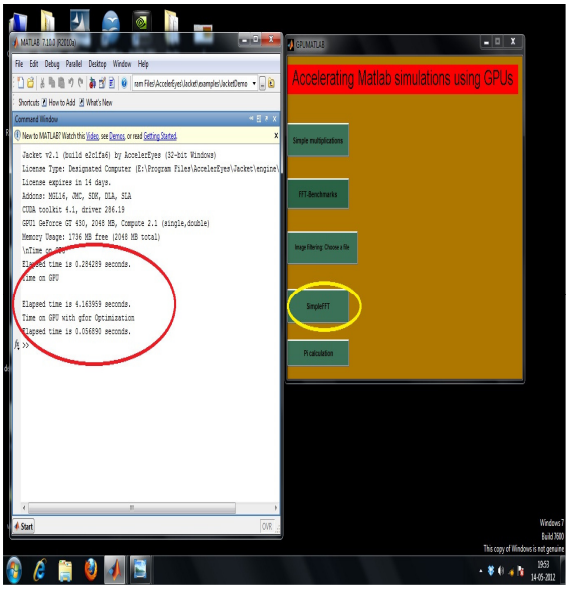
Y=fft(x)
Y = fft(X,n)
Y = fft(x) returns the discrete Fourier transform (DFT) of vector x, computed with a fast Fourier transform (FFT) algorithm.
If the input X is a matrix, Y = fft(X) returns the Fourier transform of each column of the matrix.
If the input X is a multidimensional array, FFT operates on the first non singleton dimension.

Here depending upon the size, we get CPU & GPU gflops & its speed up also. FLOPS (float-ing point operations per second) is a measure of a computer's performance i.e, instructions per second. GFLOPS means giga FLOPS. In this we compute the FFT for 'for –loop benchmarks' on CPU & GPU. And compute FFT for 'gfor-loop benchmarks'.
First Graph shows the comparison between CPU & GPU run time , here red colour indicate processing on CPU and green line indicate processing on GPU. We can clearly see the runtime differences.
Second graph shows the speed-up time.

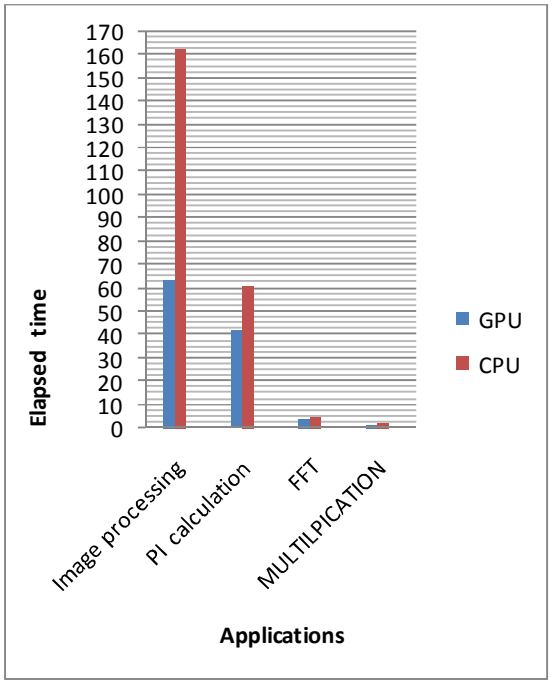


5) SIMPLE FFT

In this, we calculate the Elapsed time on both GPU & CPU, and found that time is reduced four times on GPU. Time on GPU with gfor optimization is also calculated, which is further reduced as shown below.



We have shown that processing speed of different applications of MATLAB can be enhance using GPU and JACKET & this also eliminates complex CUDA C,C++ programming. The following chart show the speed up on Graphics card .Also we expect much more speed up on latest GPUs having 1600 processor cores



Thus this work concludes that we can reduce time for discoveries in R & D and accelerate research and contribute for the social economic growth of the world.

REFERENCES

[1] Jun Zhang , Shiqiang hu . 'A GPU-accelerated real-time single image de-hazing method using pixel-level optimal de-hazing criterion' , 15 july2011.

[2] Chatterjee Subarna,Chakrabarti Amlan. 'International journal of recect trends in engineering' ,vol .1 ,may 2009 .

[3] Oberhuber T., SuzukiI A., Vacata J .: "New Row-grouped CSR format for storing the sparse matrices on GPU with implementation in CUDA", Acta Technica 56: 447-466, 2011.

[4] Tang Min, Jieyi Zhoa ,Ruofeng Tong: "GPU accelerated Convex Hull Computation", accepted by SMI'2012.

[5] Viabhav Vineet ,Narayana .P.J, "CUDA cuts: Fast graph cuts on the GPU," cvprw, pp.1-8, 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008

Oberhuber T., Suzuki A., Vacata J., Žabka V., "Image segmentation using CUDA implementations of the Runge-Kutta-Merson and GMRES methods", Journal of Math-for-Industry, 2011, vol. 3, pp. 73–79 .