International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 7, July - 2013

Efficiency of GPU over CPU in Geometric Processing of 3D Objects

NV.Veenaadeeve, Chithra.R, Manju.M, K. P. Soman

Dept. Of Computational Engineering and Networking
Amrita Vishwa Vidyapeetham
Affliated to Amrita University

Abstract

The parallel computation is one of the most important and flourishing research field nowadays. The GPUs are supporting the parallel processing on various applications in many ways which in turn has made it more popular especially in the fields where the computation of large number of data in short time is required. The parallel nature of the GPU is making if more special and effective than general purpose CPUs. Thus the invasion of the GPUs in the general purpose applications has given rise to the introduction of new area of research called as the GPGPU or General Purpose GPU. Mesh process if one of the most important 3D modelling technique which is helping in the geometric processing of the complex 3D models. In this paper, our work mainly concentrate on the study of efficiency of implementing mesh processing in GPU and the efficiency is measured according to the time constraints.

.

1. Introduction

Graphics Processing Unit(GPU) is a processor with ample computational resources and is widely used nowadays due to its advantages over Central Processing Unit(CPU). The high memory capability, efficiency in managing different jobs and programmability made the GPU a demanding platform for wide variety of applications. Demand for reducing the computational time has been increased as the complexity of the data being processed increased currently. Handling those large amounts of data is a tedious process while performing the processing steps on CPU. Since different tasks can be performed in parallel, GPU reduces the computational time to a great extent. It has its application in Computer graphics, Heavy industries, scientific research and so on.

We know the Darwin's theory of survival of the fittest. It is applicable even to the advancing technology. Since we always choose the most efficient and comfortable technique for any application and are always in search for that today's technology is growing with a racing speed. This is similar in the case of computer programming. Thus for making the application faster and efficient we should shift our concentration of processing from CPU to Graphical processing unit. This is the field that will survive in this ever-growing field of computer science. Processing in GPU is almost similar to human brain processing. A normal person can drive a vehicle, listen to music and perceive things around him all at the same time. Here different parts of our brain are divided automatically so as to handle all these activities parallel. Similarly with the help of GPU computers, it is possible to process tasks in parallel so that it can complete the tasks within a short time.

Computer graphics is one of the emerging fields in Computer Science in which 3D modelling 3D modelling is gaining much attention recently because of its wide variety of applications in gaming technology, film industries etc. Mesh processing is a 3D modelling technique in which the surfaces are modelled as using polygons. Polygon meshing is one of the popular methods used for mesh generation. The mesh is a data structure with a collection of information that is required for displaying 3D objects. The mesh size will be large and therefore it requires a large memory space to store these data. The operations are performed on meshes after loading the meshes to the CPU. For each vertex the operations are to be performed iteratively and therefore it takes more computational time. Also it requires more memory space. Here the GPU came into functionality. Because

of its parallel processing ability, each of the threads runs the program simultaneously and the result is given back to the CPU. Further operations will be performed in CPU.

The GPU combined with CPU will be able to reduce the computational time significantly. Several APIs are used along with GPU such as OpenCL, OpenGL, GLSL, HLSL and DirectX APLIs. GLSL and HLSL is shader languages used along with GPU. OpenGL API is helpful in creating as well as loading 3D objects. In this work OpenGL is used in loading meshes and then performing operations on meshes. The operations that can be performed on meshes are triangulation, smoothing, segmentation, fairing, deformation, remeshing, and so on. In this work study on meshing is performed.

2. Mesh

The mesh data structure is one such kind of structure in which the 3D models are defined on the basis of the vertices and edges. The vertices are the points which are actually forming the skeleton of the mesh. The edges are those data structures that are created by joining adjacent vertices using popular algorithms. The meshes can be defined like a truss which will be having the points and lines joining those points giving a perfect interpretation of meshes [1]. The mesh generation can be done only through the construction of the polygons on the surface of the model and the most popular among the process of polygon construction is the triangulation [1]. The triangulation is the process in which the vertices and the edges are given lots of importance because of which the triangulation could be done using some of the popular algorithms like Delaunay triangulation.

There are many application areas in which the mesh processing can be done and hence the importance of the mesh processing is getting increased. The simulations done using the mesh generation on the surface of the human skeleton can be used for forensic researches. The complex structure of the chemical molecules can also be studied clearly with the help of mesh processing on it. The mesh processing is making the processing in 3D models into a much simpler one because the complex 3D models are broken into simpler vertices in which the geometric processing can be done in a much simpler manner.

The 3D models are available in different formats like 3DS file, OBJ, Max, DSF etc. All these include information about the vertices, edges and some other attributes arranged in some particular way. For performing mesh processing first the mesh should be constructed on the information that is provided in the 3D models of the chosen format.

3. Mesh Processing in CPU

The mesh processing is one of the most flourishing areas for various applications especially for performing study on the geometry of the complex multi dimensional objects. [3] The representation of the meshes will be done based on the polygons that are built over the surface of the 3D objects thus defining the structure of the 3D object to its best. The most common way of performing mesh processing in CPU is done on mathematical software like Matlab making it easier for programming.

Though the Matlab is performing all the mesh processing steps using various algorithms it is taking more time for loading a particular mesh because of the high number of vertices. Hence the users are using MEX files which are generally said as Matlab executable file giving a parallelising efficiency while loading the meshes. The Matlab will not support .3ds format of the 3D object representation. Most of the 3D modelling software as Autodesk, Maya are producing its output 3D models as output in 3ds format only. Hence those complex models are expected to be converted from 3ds format to .obj format by the user before using it as an input to the Matlab processing steps. Even after using MEX files and .obj format the time take for loading the complex 3D object is very high. It is because of the use of the single processing thread for loading all the vertices of the 3D model at the same time.

This drawback especially the time inconsistency of the mesh processing works on the CPU has given rise to the use of GPU programming as a solution to tackle the problem. Most of the individuals are suggesting many number of GPU programming API in conjunction with the CPU also as an alternative method to get the results of mesh processing in a much efficient manner.

Our work concentrates mainly on loading a mesh using GPU which utilizes parallel processing techniques efficiently in meshing process. We then perform a comparative study on mesh loading between CPU and GPU based on time constraints. Here triangulation is done on the vertices by using certain triangulation methods and algorithms giving better results especially within the time constraints making the process to more time efficient.

4. Graphic Processing Unit

GPU is evolved to have large number of parallel threads and multiple cores. The main driving force for this evolution is the real time graphics performance needed to render complex and high resolution 3D scenes for games. A graphics programmer usually write a single threaded program that draw a single pixel and the GPU runs multiple instances of these threads in parallel so as to draw multiple pixels in parallel. Nvdia's CUDA platform was one of the widely adopted programming models in present days. OpenCL an open standard defined by khronos group is another widely supported platform which allows for the development of code for both GPUs and CPUs based on probability. OpenCL solutions are usually supported by AMD, Intel, Nvidia and ARM. It is actually the GPGPU development platform used widely in many areas in the world for research. But to directly support graphics applications OpenCL is not handy. It is better to go with OpenGL so as to get better result within improved time and less difficulty.

All the logics which will boost the single instruction stream performance are removed in such a way that each and every chip can carry more ability to program and features in a chip. Thus in the architecture of GPU around 16 processing cores are there which turn is helping in the process of enabling parallel processing. Thus the instructions can be executed at a particular time and their parallelized version can share the memory and hence can execute the various instructions in all the logical units of the processor. In modern GPUs the pixels and the vertices where the actual processing is done will be considered and are then parallelized to make the image processing or the 3D processing to be completed in a much faster manner using limited memory space within certain time.

5. Meshing in GPU

The meshes by itself are defined as the data structure in which the various features of a 3D model or object's surface could be represented using the vertices of it. The edges are built on the surface joining the vertices to study the geometry of the 3D object which was considered to be the most difficult task in 3D processing. The polygons built by joining the vertices of the 3D object using the application of certain algorithm on the edges is considered to be the face of the meshes over the surface of the 3D object.

Thus the data width of the mesh processing is wide and vast which expects more amount of processor memory to be free while performing the processing steps. In CPU there will be a single processor for loading and also for performing the further processing operations over the surface of the 3D object in the mesh because of which the time taken for meshing in CPU is high.

[2] The GPU is having 16 processors which in turn can perform the process of loading large data parallel on the threads of each processor. Thus because of the parallel structure and also because of the availability of more number of processor threads the meshes are getting processed in a much faster manner. The vertices and fragments which are generally said as the polygons of the meshes are loaded in parallel in the GPU so that the time taken for loading a particular mesh in the GPU memory is comparatively low.

6. Mesh Implementation in GPU

GPU processing requires the PC or workstation to be equipped with graphics drivers. These drivers have to be installed properly before moving to GPU programming. GPU supports different APIs such as OpenGL, OpenCL, DirectX and so on. Here we have chosen OpenGL for implementation because of its various advantages. OpenGL helps in rendering 2D and 3D graphics applications and is also very easy for the users to handle. Its OS independent feature makes it portable. OpenGL has to be installed into the PC by properly adding its corresponding headers and library files to the directories. Any compilers can be used for compiling the OpenGL program, e.g.: Visual Studio for windows.

We all know pixel value is the finest data that can be taken from a 2D image. These pixels when arranged in a specified order will render an image. If we remove some of the pixel the image will look almost the same but removal of large number of pixels will reduce the clarity and in turn will damage the image. Thus we can say that significant amount of pixels are necessary for the perfect rendering of an image. Similarly vertices are the finest data with respect to 3D models. Since loading a mesh is the first step of mesh processing applications, we are mainly concentrating on loading meshes into the computer. Loading a mesh itself is a time consuming process since it consists of many information such as vertices, faces and edges. Also the mesh size will be considerably larger than that of images. The loading of a 3D model is based on accessing the vertices of the model. Once it is accessed it is easy to create and load the original 3D model. Every file format for a 3d model has information about its vertices. This can be used for processing and rendering of a 3D model. Here we have used a 3DS files for processing based on our convenience. Also we are performing a comparison between CPU and GPU efficiency based on this loading process. The algorithm for the loading of 3D meshes in GPU and displaying it is given below:

Algorithm:
Step 1: Read input file of 3D models with any format (3DS, OBJ, MAX)

Step 2: Compute the length of the file.

Step 3: Starting from the first vertex read three consecutive vertices.

Step 4: Perform triangulation using the command GL_TRIANGLES.

Step 5: Repeat the above step until the end of the file.

Step 6: Display the model using the OpenGL command 'glutDisplayFunc'.

3D mesh is a data structure that is available in many formats. The information present in each type of file may differ and will be available in different sizes. This file is loaded into the GPU processor and length of the file is determined. The OpenGL consists of many inbuilt commands which are very helpful in performing operations on the vertices of the mesh. Using these commands, for displaying the mesh firstly a window pane is created. The color for the screen as well as the position and the size of the window can also be set. Operations such as scaling, translate and rotate can be performed on the loaded mesh by simply passing parameters to those functions.

Once the vertices are read, we can perform triangulation using the command GL_TRIANGLES [4]. This will perform triangulation without any intersection. This triangulation process is done in parallel using the GPU processor. In the case of multi-core CPU, this process is performed in a sequential manner. As a result of this, the model loading will also takes more time. The use of GPU processor will in turn results in sufficient reduction of time.

The mesh is loaded and operations are performed on the vertices and the final output is to be displayed on the window generated. Display window can be created and set prior to the display. We can set the position of the window with respect to the top left portion of the display screen. It is also possible to vary the size of the screen as well as change the background color. OpenGL consists of several built-in functions to perform these operations. For example, to set the colorglClearColor can be used. Similarly glDisplayFunc command helps with display the loaded mesh.

7. Results and Discussion

The goal of this work is to compare the time taken for loading and displaying three dimensional models of varying sizes between CPU and GPU. The large size of 3D input files affects the processing time inversely. The sequential approach of problem solving in CPU makes it more complex. Along with this complexity if the size of input file is comparatively large, this may result in increased execution time. Using GPU processor all these problems can be solved efficiently.

Loading a 3D model in CPU can be performed using different languages. Matlab is a mathematical tool which consists of several built-in functions for handing many applications in the field of image processing, signal processing, 3D processing and so on. This has an advantage over other languages such that users with little programming knowledge but with very good background of maths can use this for solving problems. The recent versions of Matlab include several packages for3D processing. Here we are loading a 3D model in CPU using Matlab. The mesh once loaded will be displayed along with its execution time. It is found that the size of the model is having an inverse effect over execution time. So further mesh processing applications will eventually takes more time. Also the 3DS files has to be converted to .obj format before loading into Matlab. This made us think for processing using GPU.

GPU is well known for its parallel computing capability. OpenGL API is used along with GPU because of its suitability in graphics rendering. 3DS files can be directly loaded into GPU with the help of OpenGL command. Even if the size of model is inversed to execution time, the delay in time while comparing with CPU, will be less. This is mainly because the processing is done by several threads in each block concurrently.

Loading a Mesh in CPU

1. Convert 3DS file to OBJ file (This has to be done with some conversion software like Maya)
2. Copy the converted file into the Matlab directory.
3. Read the file using Matlab function.
4. Plot the loaded file.
5. Compute the time.

The output obtained while processing using CPU is as given below. Also we can see the execution time for the corresponding process.

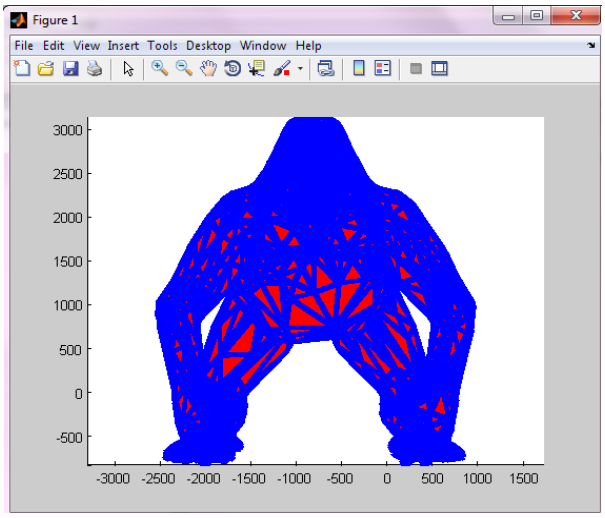

Figure 1: GORILLA 3D model(Matlab)

The Execution Time= 32.1387 Sec

Loading a 3D model in GPU

1. Read a 3DS file directly as input.
2. Perform triangulation as per the algorithm described above.
3. Display the triangulated output in the window screen.
4. Compute the time taken for loading and displaying the mesh.

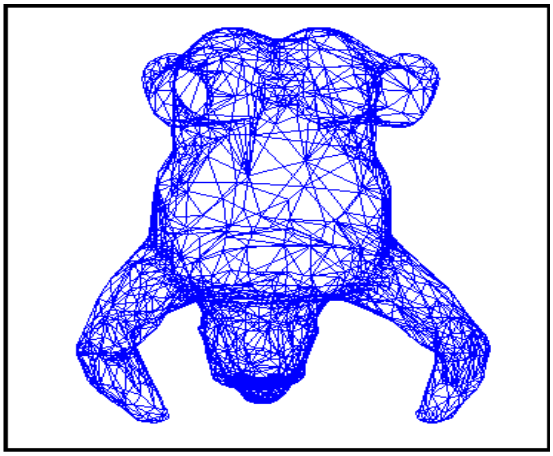

Figure 2: GORILLA 3D model(OpenGL)

Execution time taken= 0.40 Sec

According to the variation in size of input file we can see the different execution time in CPU and GPU. The table given below shows the comparison of execution time for various meshes in CPU and GPU.

Name of the 3D models	Number of vertices	Execution Time in CPU(Sec)	Execution Time in GPU(Sec)
Chicken.3ds	5592	7.6440	0.33
Dog.3ds	7956	9.6877	0.4
Boar.3ds	13470	18.9697	0.35
Sheep.3ds	15576	29.3594	0.36
Gorilla.3ds	21652	32.1387	.40

Table 1:Comparison between CPU and GPU execution time for 3D models

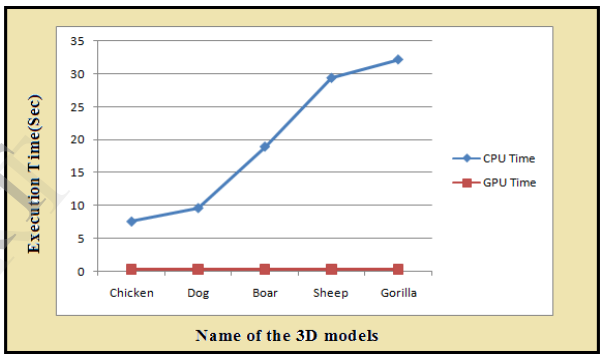

Figure 3: Graphical comparison between CPU and GPU execution

The graph shows the comparison of execution time of 3D models in CPU and GPU. Here the X-axis represents the model names we have loaded for comparison and Y-axis represents the execution time in seconds. The model names are arranged in ascending order staring from the model having smallest number of vertices. It can be clearly seen from the graph that the CPU execution time increases with increase in number of vertices whereas the GPU execution time remains almost same with increase in size of 3D models. Also we have found that loading 3D models of huge sizes using CPU exhibits excessively large time delay while it took only few seconds using GPU.

8. Conclusion

We have done the comparative study for loading and displaying the 3D models between CPU and CPU and successfully proved that the GPU processing is done faster than the CPU. There is considerable difference in the execution time taken for different 3D models. Also using CPU for computation requires conversion of

3DS files to some other executable formats. This needs to install some conversion software which is an extra work.

Once the models are loaded using GPU, we can perform various operations on these mesh models such as segmentation, deformation, morphing and so on. The execution time taken for such operations will be significantly lesser than that of the time taken for CPU because of the repetitive operations to be performed on the vertices.

9. References

[1] P.-O. Persson, G. Strang, "A Simple Mesh Generator in MATLAB". *SIAM Review*, Volume 46 (2), pp. 329-345, June 2004

[2] John D. Owens, Mike Houston, David Luebke, Simon Green, John E.Stone, and James C.Philips,"GPU Comuting", *Proceedings of IEEE*, May 2008, Vol 96, No.

[3] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, Bruno Levy,"Polygon Mesh Processing", *Published by A. K Peters/CRC Press*, pp. 250, 7th October 2010.

[4] Jackie Neider, Tom Davis,"OpenGL Programming Guide", *Addison-Wsley Longman Pubication Co. USA*, 1993.

[5] John D. Ownens, David Luebke, Govindaraju, Mark Harris, Jens Krunger, Aaron E. Lefohn, Timothy J. Purecell,"A Survey of General-Purpose Computation on Graphics Hardware", *In Eurographics 2005, State of Art*, pp. 21-51,August 2005.