Adaptive Resources Selection Framework for Grid Enabled Visualization Pipeline

Aboamama Atahar Ahmed, Universiti Teknologi Malaysia
Muhammad Shafie Abd Latiff, Universiti Teknologi Malaysia
Kamalrulnizam Abu Bakar Universiti Teknologi Malaysia

Zainul Ahmad Rajion

Universiti Sains Malaysia

Summary

Scientific data visualization is a process of transforming numerical data into a pictorial format conceivable by humans. The datasets generated by medical detectors and simulations is increasing in size and complexity. Additionally, the conventional desktop computers are not sufficient to process this datasets due to memory overwhelming phenomenon which causes the desktop to be in unresponsive state. The current implementation of remote visualization techniques specifically real time visualization takes the direction of reducing the size of the datasets which is known to give less details and precision of the visualization. On the top of that, the increasing size of datasets and the continuous demand for computational power results an urgent need for grid computing infrastructure for real time remote visualization. However, the current grid computing implantations introduce new challenges for remote real time visualization such as resources discovery and real time automatic resources selection. This paper investigates how the automatic resources selection mechanism could be used to support real time remote visualization of large medical datasets on the grid environment. We show our Adaptive resources selection framework for grid enabled visualization pipeline. Our results shows better performance of distributed parallel Isosurface of large partitioned datasets implemented as grid services. We support our findings with practical implementation of grid enabled visualization prototype, and our proposed grid mapping function algorithm for automatic resources selection to visualize large medical datasets, (circa 11 million polygons) on modest resources machine.

Key words:
Visualization, Grid computing, Medical datasets, visualization techniques, thin clients, Globus toolkit, VTK.

1. Introduction

Scientific data visualization is a process of transforming a numerical data into a pictorial format understandable by humans. The datasets generated by medical detectors and digital scanners is normally large in size and algorithmically complex. Moreover, the large size of datasets introduces several challenges for visualization researchers where the computational powers required for visualizing this datasets exceeded the ability of conventional desktop machine provided to the researchers. The current access methods to large datasets remotely are also associated with several issues such as the nature and services of infrastructure provided to access the datasets. In addition to that, there are several technologies introduced to provide access methods such as client server architecture and remote access to a cluster of machines [1] [2][3] [4] and recent studies focus on grid computing as an alternative to tackle the problem of data intensive applications. Grid computing as defined by OGSA [5] introduces the concept of sharing the resources of geographically distributed machines. This resulted through the utilization of current high speed networks therefore this grid infrastructure allows sharing of processing power memory and storage and forms like powerful virtual super computers. However, current grid implementation introduces new challenges especially for visualization community such as implementations of remote visualization on the grid. There is some introduced research studies focused on enabling existing visualization systems although these studies introduce solutions for specific systems but there are arising issues with the current implementation of these solutions specifically for real time visualization pipeline. This paper investigates how the resources discovery mechanism could be implemented to allow automatic formation of real time visualization pipeline on the grid environment. Our results show better performance of our grid mapping function in utilizing the discovered resources and making best use of the resources with comparison of current manual configuration of the pipeline. We support our findings with practical implementation of real time visualization pipeline prototype for large medical datasets.

Manuscript received December 5, 2007
Manuscript revised December 20, 2007

2. Related work

The scientific data visualization was sparked by landmark NSF report 'Visualization in Scientific Computing' by McCormick [6]. The introduced visualization concept was based on breaking down the dataflow of the visualization process to smaller distributed processes. The smaller processes can be placed on distributed locations which are interconnected by network to form a modular visualization. Each part can contribute as an independent modular to form the overall visualization process. However, the existing grid enabled visualization systems are in the direction of translating the existing dataflow concept presented by Haber and McNabb [7] as described in Figure [1].

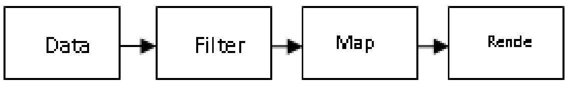

Figure [1] Haber-McNabb Visualization Pipeline

The existing visualization systems such as AVS Express [8], VTK [9], IBM Data Explorer, OpenGL VizServer [10] and IRIS Explorer [11] are generally available today and used to visualize a variety of large volume of data including medical data. For example, a system such as VTK is making use of wide range of visualization algorithms and VTK supports parallelism through the use of threads. However, these visualization systems are generally designed to work in single high capabilities hardware machine. Despite the fact that VTK was designed in an object oriented fashion, during the design of this system there were no considerations to be supported in the grid environment. Other projects are in a direction to extend the capabilities of these visualization systems. For instance, gViz project was designed to extend IRIS Explorer. However, the possible integration in grid environment should be based on the design of internal components of these systems. Therefore, the challenge now is in providing a flexible and effective mechanism to support remote access to the resources. Current implementations of grid enabled visualization are often tied to expensive hardware and powerful graphic support. In addition to that, the different network bandwidth and different output devices between the rendering location and presentation location produces output suitable for one device and not suitable for another. On the other hand, the existing grid visualization applications often make assumptions on the available resources 'render local and render remote methodologies'. The following are some of Grid enabled visualization applications and projects. RAVE [12] is a grid enabled visualization system that reacts and responds to available heterogeneous resources. RAVE implements techniques to make use of both remote and local resource according to the participating machines from high capabilities machines to Small PDA's. The gViz project [13] is another grid enabled visualization application. The idea was to incorporate the grid in the internal components of the IRIS Explorer [11]. The E-Demand [14] is another grid enabled visualization project focusing on the use of Grid services to support stereoscopic visualization in a distributed environment. The E-demand application considered as PSE "problem solving environment" on the grid. OGSA [5] presents each model as an entity. Multi rendering services can be deployed to form a collaborative environment. The SuperVise [15] is another grid implementation. In SuperVise Project, the phases of visualization pipeline such as filtering and geometry transformation are distributed across the grid. The user selects the data then the SuperVise selects the appropriate resources and form the visualization pipeline. The Distributed Visualization System [16] is visualization application that uses frameless buffer for rendering to distribute the pixel images between several machines. Each machine receives subset of the pixels to render it and submit the rendered part to create the full image, but each machine must have the original copy of the full image. Some other visualization applications do not rely totally on software in their implementations for instance Visapult [1] is a visualization framework with the ability to render a huge amount of datasets (of the order of 1-5 Tb). Visapult uses parallel rendering hardware to carry out the high speed rendering processes. Using Cactus [17] the data are distributed amongst many parallel nodes for volume rendering, the rendered subset 2D image sent to the client for local rendering. Engel_vis [18] is another application that combines Local and Remote Visualization Techniques for Interactive volume rendering in medical applications. The application was implemented using java, java 2D and java 3D based on the client which communicate with a server implemented in C++ and OpenInventor. The methodology followed is to load the datasets from the client side. Clients send the datasets to slicing tool. The slicing tool inspects slices in axial, coronal directions, and transfers the volume data to the server application. The server, a stand-alone application that utilizes 3D texture mapping hardware renders images off-screen and sends back compressed images to the clients. The methodology presented will not work well for clients with limited capabilities for local rendering and geometry transformation. There are some other implementations of grid methods on visualization, such as stated in [19] the implementation was focused on developing a rendering pipeline and this implementation also utilizes Globus toolkits for interconnecting the pipeline components with support of Chromium technology for distributed rendering. Other recent implementation as mentioned in [20] where they describe the integration of VTK with Globus to have parallel graphics rendering pipeline on grid environment. However, the mentioned grid enabled visualization applications are well structured and designed to solve specific problem. Some of the applications provide the participating machines with no ability to do the rendering processes such as COVISE. Others assume that the participating machines support the rendering resources such as OpenGL VizServer 3.1. In addition, the other newly implemented techniques are trying to solve the rendering and resources support such as RAVE visualization application. RAVE is trying to develop a mechanism to figure out which machines support the rendering tasks and which machines have limited support for the rendering. Unlike other recent implementation [19] where the methods followed did not specifically focus on the issue of Isosurface rendering or volume rendering pipeline and how it could be done on a gird environment. Despite the fact that Isosurface rendering is one of the most important issues in remote or distributed visualization, our technical implementation shows the actual results of performing Isosurface on medical datasets located remotely and displays the results on modest resources machine. These findings are described with real implementation of grid computing environment particularly with Globus toolkit to provide transparent access to the available resources. Additionally, we present in this paper unique added functionality as small embedded Java programs for the resources discovery to suite our architecture. On other hand, and make best selection of the available discovered resources.

3- A Framework for Volumetric Visualization on the Grid Environment

3.1 Visualization Toolkit

VTK [9] is an open source library for 3D computer graphics and image processing and visualization. It is object oriented implementation of over 700 C++ class**es** and more than 350000 lines of code. The library created by Ken Martin, Will Schroeder and Bill Lorensen. This library organized in a form of kits. The kits are used to build application in sequential modules with related functionality. VTK supports a variety of dataset formats and visualization algorithms. The object oriented design of VTK allows the C++ library to be accessed with wrappers built in TCL, java or Python. We are using two algorithms provided by VTK *Marching Cubes* [21] for extracting the Isosurface from the datasets to produce the polygons and *Decimation* algorithm [22] to simplify the mesh and reduce the number of polygons produced by *Marching Cubes* stage.

3.2 Grid Services

The motivation behind choosing Globus [23] as our hosting grid container comes from the useful components that make up its grid services despite the known difficulties in the Globus configuration specifically for real time visualization operations. Our architecture utilizes very important Globus components such as Globus MDS (Monitoring and Discovery Services) and GRAM (Grid Resources allocation Manager) to discover the resources available on the grid pool and to be able to send and receive data between the components of the visualization pipeline. However, the integrations of Globus and VTK is not new method for grid configuration where there has been several attempts in past a couple of years [19] [20]. The only issue is that these particular implementations did not take into consideration the real time visualization pipeline. Therefore, the visualization pipeline for real time visualization demands extra consideration to be taken into account. Adjusting the visualization pipeline is a very difficult task especially when dealing with a very large datasets on one hand. This is normally due to unreliable nature of the resources on the grid and one would not know which resources suitable for which job. On the other hand, visualization operations such as Isosurface demands more computational power and most cases conventional computer not capable of providing this power. From the above facts we have decided to use MDS and embed our java algorithm to collect information from MDS which also utilizes Ganglia to get detailed information about the nodes automatically by using UsefulRP Resource Property Providers. This information organized in a form of static data such as host name Memory size and the current processor load for every node in the environment. The issue now is how to map our visualization tasks to these discovered resources. This is where our java algorithm comes to work. This task is done immediately after having our visualization pipeline started and users press on map grid function to get VTK data reader calculate the size of datasets and checks the complexity of datasets then map the result to the algorithms to propose the resources needed for this particular datasets according to the MDS resources query results.

3.3 Visualization Pipeline Architecture

The architecture is constructed from combination of two different components. First component is VTK which is distributed based on the requirement of visualization operation. Additionally, we have VTK installed on every

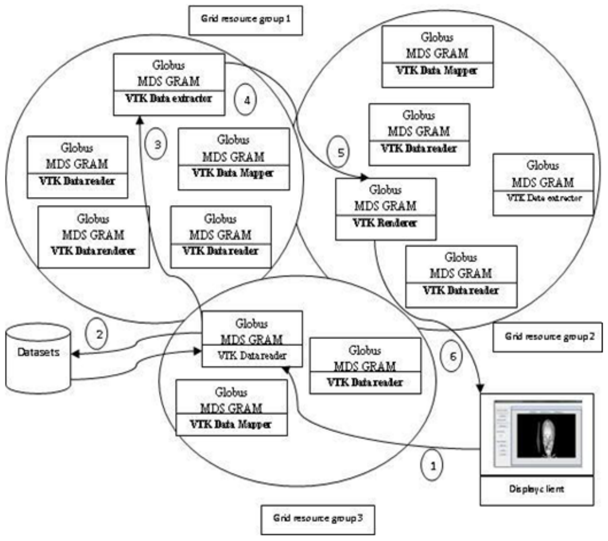
Grid resource group 1
Globus MDS GRAM VTK Data extractor
Globus MDS GRAM VTK Data Mapper
Globus MDS GRAM VTK Data reader
Globus MDS GRAM VTK Data renderer
Globus MDS GRAM VTK Renderer
Globus MDS GRAM VTK Data extractor
Grid resource group2
Datasets
Display client
Grid resource group3
1
2
3
4
5
6

Figure 2 Automatic Formation of Visualization pipeline on the grid environment

node except for the client machine where we do not expect enough resources such as memory, processing power and storage. However, in order to utilize the existing VTK interactivity we must have VTK java classes compiled installed as vtk.jar on the client. Additionally, we have GT4 which also installed on all the nodes participating in the architecture except for the display client. On The other hand, we have installed cogKt as jar file on the client to allow the queries to be passed and received by the client machine to The Globus MDS. The distribution of the architecture components is done based on grid services where we divide the architecture into several as follow Parallel Data Reader Service, Iso-surface extractor Services, Data Mapper, renderer and Display.

3.3.1 Parallel Data Reader

Data reader designed to read the selected datasets in parallel fashion, the datasets normally organized as slices specifically medical datasets. The architecture supports reading different types of datasets such as ASCII binary files or raw datasets formats and all the formats supported by VTK. The type of data reader is selected according to specified datasets and the data reader is able to read data from more than one location and append the data to one or more Iso-surface extractor. The size of the datasets is calculated at this stage. The parallel capabilities adds very important advantages to the over all architecture performance by speeding up the reading and recognition of

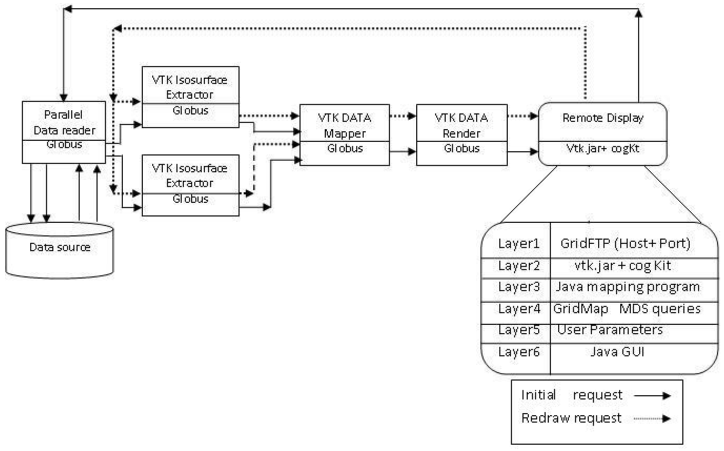

Figure 3 Grid Visualization Pipeline Architecture with Parallel Data reader and Redraw process requests of an Isosurface

the specified datasets. In addition to that dividing the datasets into chunks allows the following stages of the pipeline which are Isosurface extraction to be distributed according to load and the discovered resources on the grid. The datasets partitioning process is carried out immediately after checking the attributes of the datasets. In our particular case for medical datasets which supplied as numbered slices each portion of the datasets is read in parallel and assigned to Isosurface extractor. Additionally, the number of Isosurface Extractors to participate in the datasets extraction operations depends on two conditions. First condition is on the discovered resources in the grid. Second condition is the attributes of the datasets such as size and the number of the slices. The datasets partitioning and parallel reading capabilities provide the architecture speedup and enhance the performance in comparison with sequential reading methods. Firstly, reading the datasets in parallel will shorten the time needed to read large datasets although there is a network delay but our practical implementation shows better performance in time needed to read the datasets using parallel reading comparing with sequential pipeline without parallelism. Secondly, assigning partitioned datasets to more than one Isosurface Extractor will also give better performance and speedup the time need to extract polygons for later rendering stage. However, The Isosurface Extraction stage is known to take longer time when the entire datasets assigned to one single machine as our implementation show where with some Isosurface values resulted the memory overwhelmed and causes to machine to be in unresponsive state.

3.3.2 Distributed Isosurface extractor

The extraction process is geometry generation of the datasets and duration of this process is depending on the size, complexity of the datasets and in addition to the available resources (processing power, memory, storage) of the used machine. For these reasons dividing this process to be carried in more than one machine is necessary to avoid the memory overwhelming phenomenon which is considered the main problem when processing large datasets on one conventional desktop. Our implementation for extracting 3D grid from the datasets we used The Marching Cubes algorithm [21]. We have chosen this particular algorithm for geometry generation for several reasons. Firstly, modeling the dynamic changes of the visualization operations on the grid is a great challenge. However, for our particular Isosurface algorithm case, different Isovalue with the same datasets produces different number of generated polygons. Additionally, different quantities of polygons produced by the same Isovalue even with the same datasets with different time step. The quantity of generated polygons causes different performance of the entire extraction process and over all the performance of the pipeline. Secondly, this scenario is providing dynamic changes in the environment where the load is not fixed throughout the distributed visualization pipeline. The visualization requirements (datasets location and Isovalue) passed from the user is located at remote location to the starting server of the pipeline. Then, the pipeline is formed according to selected dataset size and generated polygons. Figure 3, the initial Isosurface drawing requests contain datasets address location and Isovalue. These parameters are passed to the starting pipeline server. The source of the datasets can be from static file or life feed from external programs. After reading and calculating the datasets size the server serialize the datasets to the assigned Isosurface extractors according to datasets size and the capacity of the extracting machines. The Isosurface extractor then deserializes the data and appends the datasets with *vtkappendFilter* implemented as grid service used to append datasets from several data extracting instances. After extracting the polygons from the datasets the extractor then serialize the resulted datasets to mapping service. The decimation process takes place after the extraction for datasets size reduction by reducing the number of polygons.
[1]

3.3.3 Data Mapper

Mapping service is responsible for taking datasets produced by Isosurface extractor and deserializes the data and maps it to one or more rendering service. Mapping and Isosurface extraction may be implemented as a single service. The resulted datasets serialized to the rendering service. The importance of mapping is to allow the discovery of grid available resources by querying the Globus MDS. The result of the query is used to assign the proper rendering nodes. The mapping service is also responsible for partitioning the resulted geometric datasets.

3.3.4 Renderer

Rendering is a process of transforming the geometric data into images. The rendering process is known to consume the available resources such as memory and storage. This particular problem is common for standard desktop computers where the rendering of large geometric datasets will consume CPU and available memory. For these reasons, our technique uses rendering services in the form of grid services. Each rendering service is registered in UDDI server and advertises itself to other services. The Globus MDS is used to discover the rendering resources in the grid. Then the render receives the assigned chunk of the datasets.

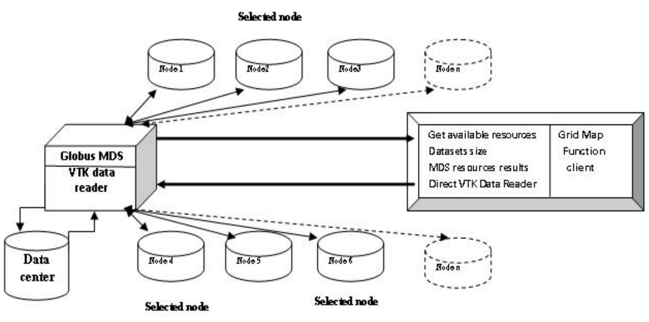

Figure 4 Grid Mapping Function

4 Resources Discovery Mechanism

The resources discovery mechanism starts at the grid mapping function Figure 4 located at client machine. The users must specify the starting point of visualization pipeline by specifying the address of the starting node and the address of data Service where the datasets stored. After supplying this information the data reader will automatically form the remaining visualization pipeline nodes. Data Reader works closely with Globus MDS to supply the necessary information about the available nodes as showing in figure 3. This information contains the available memory and storage capacity and processing power of each of the discovered node. Then according to calculated size of the datasets and with supplied information from the MDS the visualization pipeline will be formed. This technique is not built in the MDS therefore we had to write and amped small java algorithm programs to match this methodology the following scenario is a description of the proposed algorithm.

Phase 1- Starting the pipeline process

1- Connect to datasets Center
2- Calculate the size of the datasets → check the number of slices
3- Request the available node
4- Estimate the necessary power for each visualization task
5- Arrange the nodes according to the predicted power from step 4
6- Assign the visualization task to each node

Phase 2- Redraw Request process

1- Request the available resources
2- Reassign the visualization operation to each node
3- Apply the Isovalue changes to the assigned Isosurface extractors

The proposed algorithm is based on two stages. The first stage is at the starting point and at the very beginning of the pipeline launch which is described as follow. First the data reader checks the datasets format and chooses the suitable reader to read the datasets. The next step is to calculate the size of the selected datasets. Secondly the data reader connects to MDS and gets the available resources and rearranges the discovered resources according to the calculated size of the datasets. If the selected datasets is larger than the power of the first select machine then the data reader assigns more than one Isosurface extractor and that depends on the size of the datasets. The third step is the distribution of the visualization operations which is done based on the power of available resources and the attributes of the datasets. The following step is assigning the tasks and visualization operations to the selected nodes. In case the discovered machines are not suitable for the rendering process then two machines will be selected for the task. On the other hand, if the available resources are not sufficient enough for the selected datasets then the system will suggest adding more resources. The second form of the algorithm process is related to the redrawing process of the Isosurface. This part is dependent on the users entered Isovalue for redraw process. At this stage the datasets have been already read and there will be no need for rereading the datasets. The Isosurface extractor will resend the resurfaced datasets to the Mapper then to the renderer. We found that this procedure worked well because there will be no change in the pipeline sequence unless there is node breakdown. Therefore, for this particular case the Globus MDS will have regularly checked on the available resources. Additionally, our embedded java program will query the MDS at both stages at the initial pipeline launch and at every redraw process to avoid the nodes breakdown problems. On the other hand, The Data Reader Service acts as Data filter to check on the selected datasets for visualization and automatically select the suitable data reader for this particular format. This process is entirely done with supplied reader included in VTK and integrated in the reader service. The users will not have to worry about selecting readers for specific datasets since the datasets is converted if required to reduce the size for better transmission purposes with consideration on keeping the detailed precision of the datasets.

5- Adaptive Grid Visualization Pipeline Features

This section describes the adaptive gird visualization pipeline and the proposed framework for resources discovery features and the advantages of techniques introduced in the architecture.

5.1 Efficient Resources Utilization

The implemented techniques for distributing the visualization pipeline on the grid and the distributions of real time visualization operations into several machines give the architecture several advantages. Firstly, automated load distribution on the first launch of the pipeline and the capabilities of partitioning the datasets with parallel reading of the data service allow the architecture to utilize more resources for geometry generation which is known to consume the available resources of the machine. Secondly, the architecture is not only distributing the visualization operations into grid nodes, but is making best use of the discovered nodes where the visualization operations are distributed based on datasets attributes and the nodes with better resources are assigned more slices of datasets. This way the overall visualization pipelines will have better load balancing at the execution time and the user will have better interaction with the datasets. Unlike other grid enabled visualization systems such as stated in [13] in their implementation the visualization operations take over the memory of the entire used machine and the user will have to wait for the operations to complete.

5.2 Heterogeneous Support

The architecture was designed with consideration of platform independency. Additionally, the selected internal services and used components such as Globus Toolkit and VTK Library exist as an open source and provide java wrappers for these reasons we focused our implementation for clients and our imbedded components such as grid mapping function to be in java language where java is known to be platform independent. Therefore, the grid visualization pipeline allows different hardware and different operating systems to communicate and exchange the data without worrying about underlying configuration. As an example, for our testbed, we have five nodes; Three with Linux RedHat 9 and one with fedora core 3 implemented as Parallel data reader; we implemented Isosurface extractor on two nodes and the choice of number of Isosurface extractors to be used is depend on the visualization requirements. The other node works as data Mapper. This provides heterogeneous grid visualization pipeline which is able to communicate with client installed on Windows XP with modest resources and visualize large datasets without overwhelming one single machine.

5.3 Automatic Resource Discovery

For resources discovery, we utilize Globus MDS together with our embedded Grid Mapping Function to make the best use of the discovered resources on the grid. Our resources discovery mechanism starts from the display client node as the user executes the grid mapping task provided in GUI. The MDS then query the resources available in the grid and registers the resources in the system. The results of the query are information of current load of each node and the memory, storage and CPU. For better utilization of the discovered resources we wrote java client program for selection mechanism that is done by comparing our calculated datasets size and the power of the available nodes. On the other hand, our system is publically available to other users. The resources are advertised as grid services and registered in UDDI server. This allows automatic discovery of resources and give the system flexibility to add or remove services to the pipeline as required.

6. Testbed Implementation

The resources we used for testbed implementation include 2 HP workstations equipped with NVidia GeForce 4MX Go graphics, 512 MB of RAM and 2.87 GHz CPU running on Linux RedHat 9, and two with NVidia nv10 GeForce 256 SDR graphics card , 256 MB of RAM running Linux Fedora core 3. At the client user, we used HP Notebook equipped with Intel(R)Pentium(R)4 CPU 2.80GH, Graphic Adapter ATI Mobility IGP 340M/345M , 512 MB of RAM and ST94011A 40 GB disk drives running on windows XP Professional. All the machines were linked with LAN cable 100MB Ethernet LAN. During the implementation there was extra demand for memory during the rendering process.

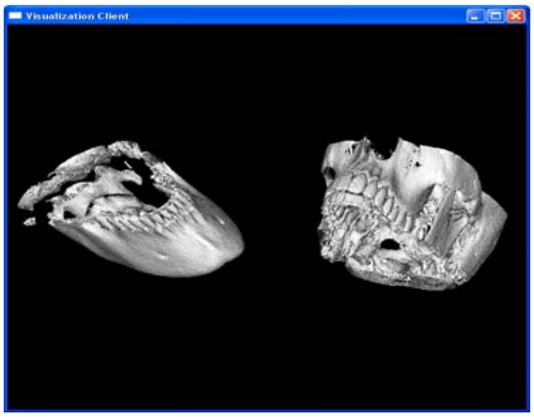

Figure 5 Dental Scan with Isovalue 1600

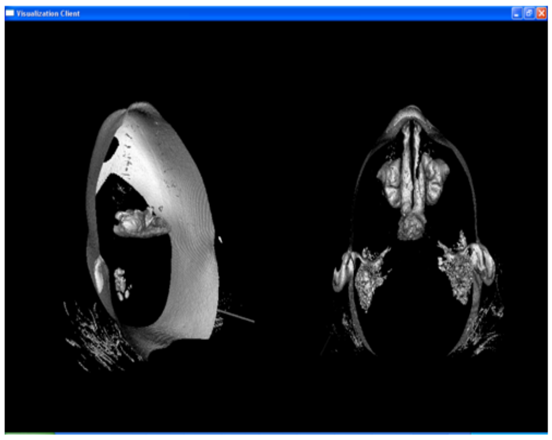

Figure 6 Dental Scan with Isovalue 500

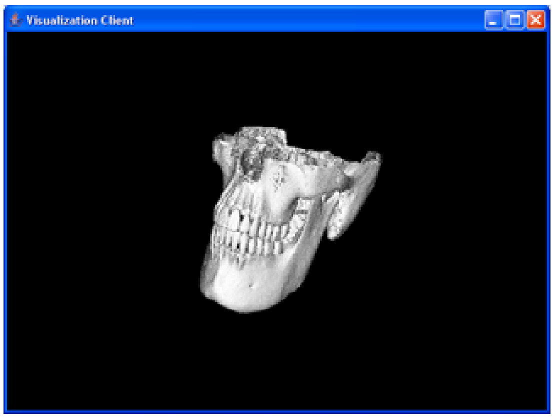

Figure 7 Dental Scan with Isovalue 1600

7. Experimental Results

For our presented results, we used test models (CT scan of facial bone) in raw format that were obtained from Hospital Universiti Sains Malaysia and second model was the 3D Dental Scans dataset converted to VTK raw format was taken from public datasets archive Table 1 shows the models used in our experiment.

Table 1: Models used in benchmarks

Model Name	Number of Polygons	Size of Data File
Skeleton head	4.28 million	15.1MB
3D Dental Scan	14.62 million	58 MB

The first raw skeleton datasets consists of 121 slices of 256*256 * 256 producing file size 15.1MB. The second model is 3D dental scans consists of 167 slices of 256*256 * 256 producing file size 58 MB. The datasets first read in parallel by Parallel Data reader services as for model of 3D dental scans datasets during the experiment. The data Reader services partitioned the datasets into two portions. The slices 1 to 81 are read concurrently with the slices from 82 to 167 with one Parallel Data reader Service. The implemented Parallel Data Reader Services was based on vtkVolume16Reader java class and modified to read the datasets in Parallel and work as grid services located at Skudai.fsksm.utm.my node which is the starting of the pipeline. Then portions of the portioned datasets sent to Assigned Isosurface Extractors immediately after completing the reading stage. At our Isosurface Extractors we used marching cubes and a polygon decimation algorithm for geometry generation The vtkmarchingcubes algorithm was used to extract the Isosurface from the supplied datasets and we used and vtkDecimatePro was to reduce the number of produced polygons from the previous step. For better illustration purpose the figure 5 Dental Scans Datasets shows the visualization client with two visualized objects. The object on the lift is showing the lower part of the dental scan was processed on Mewah.fsksm.utm.my with Isovalue 1600. Additionally, the object on the right of the same figure was processed on Kulai.fsksm.utm.my Machine with the same Isovalue. And the same condition for Figure 6 the only different is with Isovalue 500. Figure 7 shows visualization client with combined portions of the portioned Datasets on the client machine. However, The Mapping Service was needed before the datasets sent to the visualization client to reduce the load of appending the datasets to form one single object. We used vtkAppendFilter at Mapping Service to append the datasets together. This stage is also responsible of mapping the produced polygons to the selected Render Services then to the display client. Figure 8 shows the performance of the selected Isosurface Extractors for dental scan datasets and the number of produced polygons for each Isovalue.

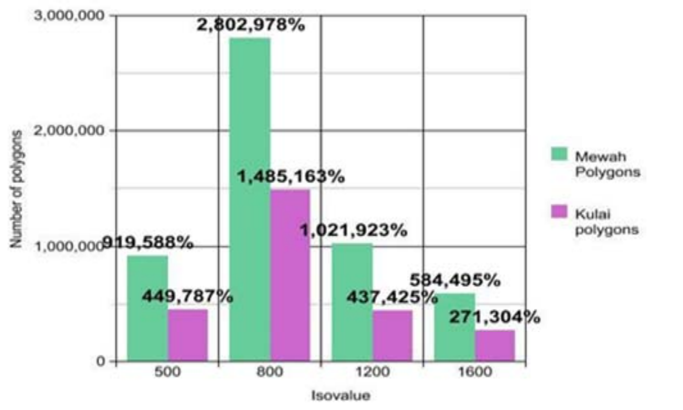

Figure 8 Mewah and Kulai Isosurface Extractors with different Isovalue and the produced polygons for dental scans Datasets

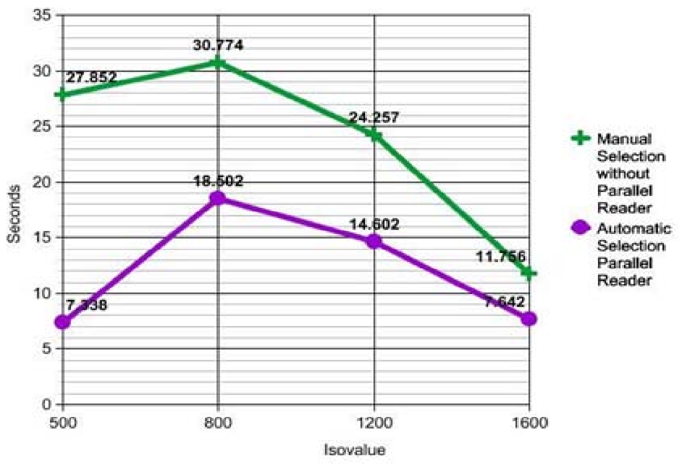

Figure 9 Automatic resources Selection with Parallel Reading with comparison of Manual resources Selection

Table 2: Improvement of Automatic resources Selection

	Manual (s)	Automatic (s)	Improvement
Isovalue 1600	27.852	7.338	73.65%
Isovalue 1200	30.774	18.502	39.87%
Isovalue 800	24.257	14.602	39.80%
Isovalue 500	11.756	7.642	34.99%

8. Automatic Resources Selection.

The partitioning and distributing the datasets to more than one Isosurface Extractor give the architecture several advantages over the conventional methods. There are some other methods and tools that are able to distribute the assigned jobs and tasks in environment similar to the cluster. In addition to that, for cluster case the user already has information about his/ her position in the cluster queue. Unfortunately, these conditions are not possible with real time grid visualization pipeline where the grid is built on unreliable environment and implementing the queue methodology will cause the undesired outcome for the entire pipeline performance. However, for our architecture, we implemented automatic resources selection based on grid mapping function and MDS queried results. Figure 9 shows the performance gained from automatic resources selection with parallel reading in comparison of manual resources selection. The results were based on two conditions. First we implemented manual resources selection by specifying the Isosurface extractor to extract without parallel reading for different Isovalue. We noticed that the manual resources selection causes The Isosurface Extraction node to be in unresponsive state with certain Isovalue where it worked well with the second condition by distributing and portioning method with two automatically selected Isosurface extractions.

9. Conclusion and future work

We presented architecture with Adaptive resources discovery mechanism for real time visualization pipeline for large medical datasets. The implementation was based on partitioning the datasets and parallel reading capabilities and automatic resources selection for real time formation of grid enabled visualization pipeline. Our presented results show the improvements of the introduced techniques in comparison of the existing manual selection without parallel and partitioning techniques. Additionally, our presented findings were based on practical implementation of grid visualization pipeline prototype. Our future work will focus on distributed rendering and we will more focus on applying real time distributed rendering techniques to improve the performance. We showed how resources discovery mechanism could be implemented to support automatic formation of real time visualization pipeline on grid.

References

[1] Bethel .W, Tierney. Brian, Lee . J, Gunter .D, Lau S (2000): Visapult Using High-Speed WANs and Network Data Caches to Enable Remote and Distributed Visualization, 2000 IEEE.

[2] Xiaoyu Zhang, Chandrajit Bajaj, William Blanke : 2001 Scalable Isosurface Visualization of Massive Datasets on COTS Clusters : Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics

[3] Engel K Sommer .O, Ernst C, Ertl T. (2000): Remote 3D Visualization using Image- Streaming Techniques. 2000

[4] Brett Beeson1,2, Mark Dwyer1, David 2005 : Server-side Visualization of Massive Datasets Thompson3 Proceedings of the First International Conference on e-Science and Grid Computing (e-Science'05)

[5] Foster, C. Kesselman, Nick .K. M., Tuecke .S (2002): The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Technical report, Globus, February 2002.

[6] McCormick B. H., DeFanti T. A., Brown M. D. (1987), "Visualization in Scientific Computing", Computer Graphics 21 1-14.

[7] Haber, R.B. and McNabb, D.A. 1990. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In: Visualization in Scientific Computing, Shriver, B., Neilson, G.M., and Rosenblum, L.J., Eds., IEEE Computer Society Press, 74-93.

[8] Upson, C., Faulhaber, T., Kamins, D., Schlegel, D., Laidlaw, D., Vroom, J., Gurwitz, R. and van Dam, A. 1989. The Application Visualization System: a Computational Environment for Scientific Visualization, IEEE Computer Graphics and Applications 9, 4, 30- 42.

[9] Will Schroeder, Ken Martin, and Bill Lorensen, The Visualization Toolkit: An Object-Oriented Approach To 3D Graphics. Second Edition. Prentice Hall. Upper Saddle River, NJ. 1998.

[10] SGI. SGI OpenGL VizServer 3.1. Data sheet, SGI, March 2003.

[11] Walton, J.P.R.B. (2004). NAG's IRIS Explorer. In: Visualization Handbook, Johnson, C.R. and Hansen, C.D., Eds., Academic Press (in press). Available at http://www.nag.co.uk/doc/TechRep/Pdf/tr2_03.pdf

[12] Walker D. W. , Grimstead .I (2004): Resource aware visualization environment. http://www.wesc.ac.uk/projects/rave/.2004

[13] Wood. J, Brodlie, K., J. Walton. (2003) gViz – visualization and steering for the grid. In Proceedings of the UK All Hands Meeting 2003, http://www.nesc.ac.uk/events/ahm2003/AHMCD/pdf/030.pdf. , http://www.visualization.leeds.ac.uk/gViz.

[14] Charters, S., Holliman, N.S. and Munro, M. 2003. Visualization in e-Demand: Grid Service Architecture for Stereoscopic Visualization, Proceedings of UK e-Science Second All Hands Meeting.

[15] Osborne .J, Wright .H, (2003) SuperVise: Using Grid Tools to Support Visualization. In Proceedings of the Fifth International Conference on Parallel Processing and Applied Mathematics (PPAM 2003),
[16] Mahovsky .J, Benedicenti. L (2003): Architecture for Java-Based Real-Time Distributed Visualization. IEEE Transactions on Visualization and Computer Graphics, 9(4):570 – 579, October December 2003.
[17] Allen .G, Benger. W, Goodale. T, Hege H.-C, Lanfermann . G , Merzky . A, Radke. T , Seidel .E, Shalf .J (2000): The Cactus Code: A Problem Solving Environment for the Grid. In Proceedings of the Ninth International Symposium on High Performance Distributed Computing (HPDC'00), pages 253–262. IEEE, August 2000
[18] Engel K. et al.. (2000): Combining Local and Remote Visualization Techniques for Interactive Volume Rendering in Medical Applications. 2000
[19] Lorensen, William and Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics (SIGGRAPH 87 Proceedings) 21(4) July 1987, p. 163-170)
http://www.cs.duke.edu/education/courses/fall01/cps124/resources/p163-lorensen.pdf
[20] Ade J. Fewings and Nigel W. John, "Distributed Graphics Pipelines on the Grid," IEEE Distributed Systems Online, vol. 8, no. 1, 2007, art. no. 0701-o1001.
[21] Dutra, Rodrigues, Giraldi, Schulze, "Distributed Visualization Using VTK in Grid Environments," ccgrid, pp. 381-388, Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07), 2007
[22] William J. Schroeder , Jonathan A. Zarge , William E. Lorensen, Decimation of triangle meshes, ACM SIGGRAPH Computer Graphics, v.26 n.2, p.65-70, July 1992
[23] Thomas Sandholm and Jarek Gawor. Globus Toolkit 3 Core - A Grid Service Container Framework. Globus Toolkit 3 Core White Paper, July 2003.
[24] M. L. Massie, B. N. Chun, and D. E. Culler, The Ganglia Distributed Monitoring System: Design, Implementation, and Experience, Parallel Computing, Vol. 30, Issue 7, July, 2004
[25] Ian Bowman 2004 Performance Modeling for 3D Visualization in a Heterogeneous Computing Environment: http://vis.lbl.gov/Publications/2004/Bowman-PGV-LBNL-56977.pdf

Acknowledgments

This research is supported by the Ministry of Science, Technology and Innovation Malaysia and collaboration with Research Management Centre, Universiti Teknologi Malaysia.

Aboamama Atahar Ahmed received Higher Diploma in computer programming and system analysis from Higher Institute of comprehensive profession Libya and M.S. degrees in Information and Multimedia Technology from Unitar University Malaysia in 1997 and 2004, respectively. he is now PhD candidate at Department of Computer Systems and Communications, Faculty of Computer Science and Information Technology,Universiti Teknologi Malaysia.

Muhammad Shafie Abd Latiff received the B.S. 1986 and M.S. 1995 degrees in Computer Science (Universiti Teknologi Malaysia) and PhD in Modeling, Simulation and Animation in Virtual Environments (Bradford University, United Kingdom) in 2002. he is now a senior lecturer. and Head of Computer System & Communication Department , Faculty of Computer Science and Information Technology, Universiti Teknologi Malaysia

Kamalrulnizam bin Abu Bakar received the B.S 1996 in Computer Science, Universiti Teknologi Malaysia and M.S. in Computer Communication & Networks, Leeds Metropolitan University, UK. in 1998 , PhD in Computer Science (Network Security), Aston University, UK. he is now senior lecturer. at Department of Computer Systems and Communications, Faculty of Computer Science and Information Technology, Universiti Teknologi Malaysia.

Zainul Ahmad Rajion received M.S. PhD from Adelaide Australia now is medical doctor at School of Dental Sciences, Health Campus Universiti Sains Malaysia