Automatic Visualization Pipeline Formation for Medical Datasets on Grid Computing Environment

Aboamama Atahar Ahmed, Muhammad Shafie Abd Latiff, Kamalrulnizam Abu Bakar, and Zainul Ahmad Rajion

***Abstract*—**Distance visualization of large datasets often takes the direction of remote viewing and zooming techniques of stored static images. However, the continuous increase in the size of datasets and visualization operation causes insufficient performance with traditional desktop computers. Additionally, the visualization techniques such as Isosurface depend on the available resources of the running machine and the size of datasets. Moreover, the continuous demand for powerful computing powers and continuous increase in the size of datasets results an urgent need for a grid computing infrastructure. However, some issues arise in current grid such as resources availability at the client machines which are not sufficient enough to process large datasets. On top of that, different output devices and different network bandwidth between the visualization pipeline components often result output suitable for one machine and not suitable for another. In this paper we investigate how the grid services could be used to support remote visualization of large datasets and to break the constraint of physical co-location of the resources by applying the grid computing technologies. We show our grid enabled architecture to visualize large medical datasets (circa 5 million polygons) for remote interactive visualization on modest resources clients**.**

***Keywords*—**Visualization, Grid computing, Medical datasets, visualization techniques, thin clients, Globus toolkit, VTK.

Manuscript received October 30, 2007. 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 is a PhD candidate at department of computer systems and communications, faculty of computer science and information technology,Universiti Teknologi Malaysia, 81310 UTM Skudai Malaysia (phone: (607)-555536503; fax: (607)-5565044; e-mail: atahar74@gmail.com).

Muhammad Shafie Abd Latiff PhD Bradford University, United Kingdom he is now a lecturer and head of computer system & communication department, faculty of computer science and information technology, Universiti Teknologi Malaysia, 81310 UTM Skudai Malaysia (phone: (607)-5532006; fax: (607)-5565044 ;e-mail: shafie@utm.my).

Kamalrulnizam bin Abu Bakar PhD Aston University, United Kingdom he is now Lecturer at department of computer systems and communications, faculty of computer science and information technology, Universiti Teknologi Malaysia, 81310 UTM Skudai Malaysia (phone: (607)-5532382; fax: (607)-5565044; e-mail: knizam@utm.my).

Zainul Ahmad Rajion is a medical doctor at School of dental sciences, health campus Universiti Sains Malaysia 16150 Kubang Kerian, Kelantan, Malaysia (phone: +609 766 3764; e-mail: zainul@kck.usm.my).

I. Introduction

SCIENTIFIC Visualization is a process of transforming a numerical datasets into pictorial format understandable by human. This datasets is normally very large in size and algorithmically complex. Therefore, processing this datasets with conventional desktop computer is not sufficient, where the machine will be overwhelmed with intensive processing of large datasets even with the latest development of visualization techniques. Distance visualization concerns providing remote users with access and utilization of remote powerful resources located at the remote location. However, real time visualization requires on demand or real time access to the remotely available resources. Additionally, the resources generally are not reliable in nature and distributed on different networks. This unreliable distribution of resources on different networks leads to output suitable for one device and not suitable for another. On the other hand, scientific visualization demands availability of powerful resources such as powerful graphics adapters and large memory and most often extended storage devices. Moreover, designing a visualization to run on single machine always results in specialist high cost super computers. These high-end powerful resources always need to be located in a secure location with limited access privileges. There are several techniques introduced to provide flexible methods for remote visualization. Unfortunately, these techniques are always based on client server paradigm where there is powerful computational resources do the visualization tasks at the backend. Other techniques are based on clustering methods by building a cluster of nodes to carry out the computational load [1], [2], [3], [4]. Unlike clusters, grid methods are designed to deal with unreliable resources where the cluster is a group of similar resources attached together to build extra computational power. Grid computing [5] is a term used to describe the process of sharing geographically distributed resources. This distributed computing infrastructure allows the sharing of processing power, memory, storage and high performance graphics in heterogeneous environment. We utilize grid technologies to provide transparent access to remotely located resources and implement isosurfacing algorithm architecture on grid environment. This paper

investigates the integration of grid services with scientific visualization and we support our findings with practical implementation of grid enabled remote visualization prototype for large medical datasets. We give an overview of our grid visualization architecture and describe our implementation and the results obtained. Our initial results show the performance of rendering large datasets located remotely.

II. Previous 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 Fig. 1.

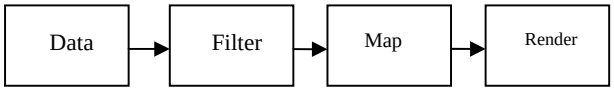


Fig. 1 Haber-McNabb Visualization Pipeline

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 a single powerful machine. Some 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 architecture to support remote access to the resources. Current implementations of grid enabled visualization are often tied to expensive hardware and powerful graphic support. 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] it incorporate the grid in the internal components of the IRIS Explorer [11].

The E-Demand [14] is a grid application which focuses 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 relay 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. 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. 1 Despite the fact that Isosurface rendering is one of the most important issues in remote or distributed visualization, therefore 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 have added some functionality as small embedded Java programs for the resources discovery to suite our architecture. On other hand we have utilized VTK (Visualization Toolkit) to provide the necessary visualization techniques particularly Isosurface algorithm and decimation algorithm.

III. A Framework for Volumetric Visualization on the Grid Environment

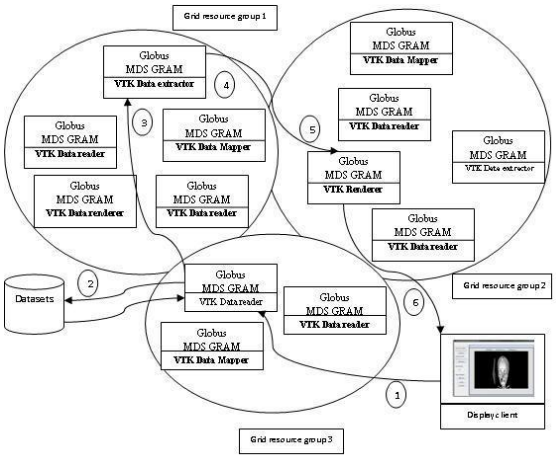


Fig. 2 Automatic formation of Visualization Pipeline on the grid

A. Grid Services

Fig. 2 shows the automatic formation of the visualization pipeline. We used Globus [23] to host our 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 and GRAM 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, adjusting the visualization pipeline is very difficult task specially 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, a visualization operation 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 using external resources property providers. The issue now is how to map our visualization tasks to these discovered resources. This is where our java algorithm comes to work.

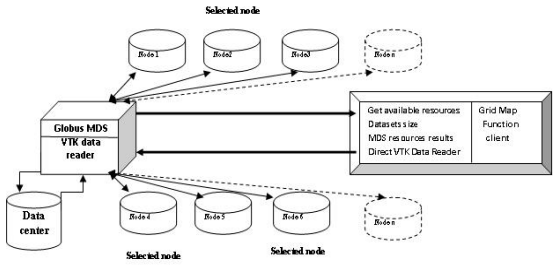


Fig. 3 shows grid Map Function

B. VTK and Globus Grid Services Integration

Fig. 3 shows the integration of Globus and VTK to form the visualization pipeline MDS finds the available resources and GRAM services direct the requests between the pipeline components in a form of RSL script. However, the architecture should have Globus installed on each node except for the display clients. The display clients should have COG Kit installed to allow the RSL script to map the GRAM jobs to other grid nodes. In addition to Globus installation, we separately implemented the installation of VTK modules on each node. That is by breaking out the visualization operations into small subtasks to be run as network connected modules. This way we achieved the distribution of workload between the grid nodes and avoid the visualization operations to overwhelm one single machine. For the display client, we must have VTK java packages as a jar file to give a flexible implementation and interaction features to allow real-time interaction with the scene. With these backend architecture components VTK and Globus, we only need to promote our services as grid services with WSDL and to discover these services as the visualization requirements. From the display client, the user will need to perform the grid mapping task as a mouse click to map the jobs to suitable resources which resulted based on MDS queries. Unlike other grid applications where the implementation of grid resources discovery use manual selection of resources, the resources discovery in our architecture is done in automatic way and the users are not required to have detailed knowledge of the grid nodes and the users will not worry about manual mapping and selection for resources.

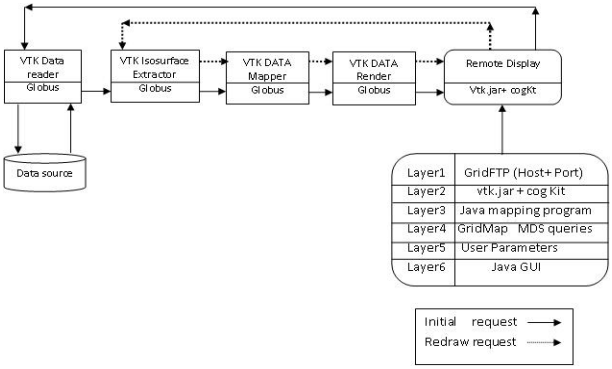


Fig. 4 Redraw process requests of an Isosurface

C. Grid Visualization Pipeline Architecture

Our grid visualization pipeline architecture is divided into several stages as follow Reader, Iso-surface extractor, Mapper, renderer and Display.

1. Data Reader

Data reader was designed to read different type of datasets, such as ASCII binary files or raw datasets format. The 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 datasets

size is calculated at this stage.

2. Isosurface extractor

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 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. Therefore 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 Isosurface value) passed from the user located at remote location to the starting server of the pipeline. Then, the pipeline is formed according to selected dataset attributes and number of generated polygons. Fig. 4 shows 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 to one or two Isosurface extractor according to datasets size and the capacity of the extracting machine. 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.

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.

4. Renderer

Rendering is a process of transforming the geometric data into images. The rendering process is known to consume the available resources memory and storage. This particular problem is common for standard desktop computers where the rendering of large geometric datasets consumes CPU and available memory. For these reasons, our technique uses rendering services in the form of grid services. Each rendering services 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.

IV. Grid Visualization Pipeline Features

This section describes the gird visualization features and the advantages of spreading the visualization pipeline on the grid.

A. Heterogeneous Support

The implementation of our pipeline as grid services 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 three machines two with Linux RedHat 9 and one with fedora core 3 implemented as data reader, Isosurface extractor and data Mapper respectively which are able to communicate with Windows XP.

2. Efficient Resources Utilization

The technique of distributing the visualization operations offers chance to other users to utilize the resources where the workload is divided to several machines, 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. Resources utilization is an important concept in the grid concept. In our architecture, we implemented resources utilization in two main points. The first point is to divide the visualization task as connected pipeline that helped us in distributing the load and avoid our machines to be overwhelmed with several operations in one node. The second point is in our implementation of automatic discovery of resources. Where we first discover the proper resources and we make best use of them.

3. Automatic Resource Discovery

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 resulted of the query is information of current load of each node and the memory, storage and CPU. For that purpose we wrote java client program as grid mapping function for resources selection which is done by comparing our calculated datasets size and the power of the available nodes.

V. 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 one 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

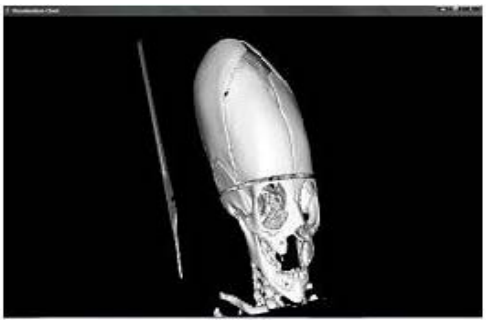
40 GB disk drives running on windows XP Professional. All the machines were linked with LAN cable 100MB Ethernet LAN. During the implementation there was extra demand for memory during the rendering process.

VI. Experimental Results

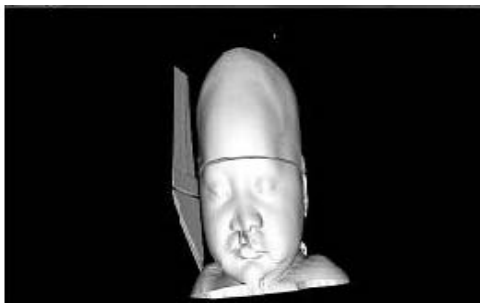
For our initial results, we used test models (CT scan of facial bone) in raw format that were obtained from Hospital Universiti Sains Malaysia and second model was the UNC head dataset converted to ASCII VTK Binary format was taken from public datasets archive Table I shows the models used in our experiment.

TABLE I
MODELS USED IN BENCHMARKS

Model Name	Number of Polygons	Size of Data File
Skeleton head	4.28 million	15.1MB
3D full head	11.17 million	23.1MB



(a)



(b)

Fig. 5 (a) Isosurface with Isovalue of 1200 (b) Isosurface with Isovalue of 600

The raw skeleton consists of 121 slices of 256*256 * 256 producing file size 15.1MB as reported in the table. The datasets were processed by marching cubes and a polygon decimation algorithm. The two models are shown in the screenshots from the visualization client in Fig. 5. The used algorithms in our architecture are vtkmarchingcubes to extract the Isosurface and vtkDecimatePro to reduce the number of produced polygons from the first step. (Fig. 5 A) shows Isosurface of 15.1 MB datasets at client with Isovalue 1200. (Fig. 5 B) shows the Isovalue 600. And it is enough for skin surface for this particular datasets. To analyze and exchange our datasets via the pipeline, we used VTK at each node of the pipeline installed along with GT4. In our architecture, the implemented VTK java classes imports GT4 packages for easy programs integration. WSDL are used to advertise our services, such as render services. Our implementation is not restricted to particular platform. Our visualization pipeline components are distributed and advertized as grid services then published by UDDI server. Users only need to query the MDS for available services. However, for our initial implementation for resources discovery we utilize MDS included with GT4 installation. From the users perspective this underlying configuration is hidden. The only task for client user is to press on map grid function to query the resources available and map the required visualization operation to the proper available resources.

VII. Pipeline Performance

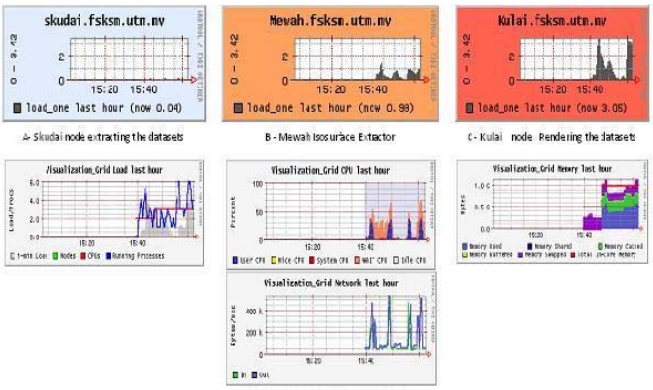


Fig. 6 The pipeline performance

Fig. 6 shows the pipeline performance where we have complete installation of Ganglia [24] for distributed monitoring system on one client machine which connects to other nodes on the grid using Ganglia Monitoring Daemon and Ganglia Meta Daemon. The reason for choosing Ganglia for pipeline monitoring is that it has a good support for Globus and produces accurate measurement output for unreliable resources and provided flexibility in heterogeneous environment. These conditions are provided in grid computing environment. We support the performance Measurement for the pipeline with analytical approach as described by [25] where the overall pipeline performance is calculated by calculating individual machines. The reason for choosing this method is that, in order to have accurate performance modeling for sequential pipeline as in our case the performance of an Isosurface algorithm, we notice that different numbers of polygons and points with different Isovalue even with the same datasets. Therefore, the number of produced polygons results in radically different performance characteristics for the entire pipeline execution. In our experiment we showed better interactivity performance of Isosurface of 15.1 MB datasets located at our (Skudai.fsksm.utm.my) starting pipeline node for reading then datasets passed to Isosurface extraction node

(Mewah.fsksm.utm.my) and then passed to rendering service at (Kulai.fsksm.utm.my) then to client display (HP_Mobile.fsksm.ut.my) notebook. Although measuring the performance in unreliable grid environment is not straightforward task, particularly where the start of the pipeline should always in contact with display client through out the pipeline execution. Therefore, in this specific implementation we utilized the components of VTK sending and receiving java rmi triggers in real time with support of update extent compiled classes in VTK distribution.

VIII. Resources Mapping and Discovery

The grid mapping is used to map the resources. Our current implementation of mapping is done by using RSL script as GRAM implementation of GT4. The user only need to specify the starting node of the pipeline and the location of the datasets that he or she needs to visualize and the isosurface value required for visualization. The rest of the operation will be done automatically without having the user worry about extra configuration. The embedded java algorithm for resources mapping is responsible for pipeline formation at very initial stage by requesting the available resources as MDS queries and receiving the size of the datasets from VTK data reader then the users will have to implement grid map Function task after specifying the above parameters.

IX. Conclusion and Future Works

We presented our implementation of grid enabled remote visualization architecture. We gave a brief description of our technical implementation and showed the possible integration of grid services and valuable support for scientific visualization particularly on medical datasets. We decomposed the visualization pipeline in distributed machines and developed our visualization services as grid services registered and published to public UDDI server. We show the usefulness of distributing the workload between several machines and how to utilize the Globus GRAM services to automatically launch the pipeline. Our next aim is to distribute the rendering process. Specifically, we are interested in applying parallel algorithms for this implementation. We were able to interactively visualize large number of polygons circa 9 million polygons at the client with java installed on modest resources machine.

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: available online http://vis.lbl.gov/Publications/2004/Bowman-PGV-LBNL-56977.pdf