Interactive Visualization of Large-Scale Architectural Models over the Grid

Strolling in Tang Chang'an City

XU Shuhong[1], HENG Chye Kiang[2], SUBRAMANIAM Ganesan[1], HO Quoc Thuan[1], KHOO Boon Tat[1] and HUANG Yan[2]

[1] *Institute of High Performance Computing, Singapore*

[2] *Department of Architecture, National University of Singapore*

Keywords: remote visualization, grid-enabled visualization, large-scale architectural models, virtual heritage

Abstract: Virtual reconstruction of the ancient Chinese Chang'an city has been continued for ten years at the National University of Singapore. Motivated by sharing this grand city with people who are geographically distant and equipped with normal personal computers, this paper presents a practical Grid-enabled visualization infrastructure that is suitable for interactive visualization of large-scale architectural models. The underlying Grid services, such as information service, visualization planner and execution container etc, are developed according to the OGSA standard. To tackle the critical problem of Grid visualization, i.e. data size and network bandwidth, a multi-stage data compression approach is deployed and the corresponding data pre-processing, rendering and remote display issues are systematically addressed.

1 INTRODUCTION

Chang'an, meaning long-lasting peace, was the capital of China's Tang dynasty from 618 to 907 AD. At its peak, it had a population of about one million. Measuring 9.7 by 8.6 kilometres, the city's architecture inspired the planning of many capital cities in East Asia such as Heijo-kyo and Heian-kyo in Japan in the 8th century, and imperial Chinese cities like Beijing of the Ming and Qing dynasties. To bring this architectural miracle back to life, a team led by Professor Heng Chye Kiang at the National University of Singapore (NUS) has conducted extensive research since the middle of 90s. Hundreds of full-scaled buildings and architectural landmarks, such as 55-meter Mingde Gate at the city's main entrance, Zhuque Avenue which was as wide as a 45-lane highway, Buddha Hall and Hanyuan Hall etc, have been digitally reconstructed (Heng 1999). How to enable users, especially remote users with normal personal computers, to access and interactively explore this splendid city becomes a very attractive research topic. The promise of Grid computing is a transparent, interconnected fabric to link data sources, computing (visualization) resources, and users into widely distributed virtual organizations

(Shalf and Bethel 2003). Lots of research and development efforts have been conducted on Grid computing issues (www.globus.org). However, Grid-enabled visualization, particularly interactive visualization of large-scale data, has not been well addressed and a wide gulf exists between current visualization technologies and the vision of global, Grid-enabled visualization capabilities. In this paper, we present a practical visualization infrastructure over the Grid for interactive visualization of large-scale architectural models. A multi-stage data compression approach, which happens in data pre-processing, rendering, and remote display stages, is deployed to tackle the critical problem of Grid-based visualization, i.e. data size and network bandwidth. The underlying Grid services, such as information service to enquire available data and visualization resources, visualization planner and execution container etc, are developed according to the OGSA standard (Foster 2001, 2003). Compared with other Grid-enabled visualization infrastructures, ours is specially designed to provide efficient services to the interactive visualization of large-scale architectural models and easy to implement. The related visualization issues have been systematically studied and integrated with the Grid services.

The reminder of this paper is organized as follows: Section 2 reviews the related remote visualization technologies and other Grid-enabled visualization approaches. A new infrastructure, A-VizGrid, is presented in Section 3. Section 4 introduces a multi-stage data compression approach and the corresponding techniques adopted. The development of a prototype system based on A-VizGrid is covered in Section 5 and conclusion is given in Section 6.

2 RELATED WORK

Grid-enabled visualization is a relative new area that originates from remote visualization and Grid services. Typically there are four approaches for remote visualization: (1) do everything on the remote server, (2) do everything but the rendering on the remote server, (3) use a local proxy for the rendering, and (4) do everything at the local site. SGI Vizserver (www.sgi.com/products/software/vizserver/) and VIRTUALE3D vCollab (www.virtuale3d.com) are two examples that use approach (1) and (4), respectively. In SGI Vizserver, rendering is executed at the server site. Rendered results are then captured from the server's frame buffer and sent to remote clients. To speed-up image transmission, users can interactively choose different compression ratio (up-to 1:32) to compress images. According to our test, the default image compression algorithm provided by Vizserver is inefficient and network latency is another problem. Using a different approach, vCollab copies the whole dataset to each of the collaborators. Rendering is carried out locally. This can reduce network latency but system rendering ability is heavily limited.

The advent of Grid computing has inspired people to explore Grid-enabled visualization technologies. In UK, researchers have been developing visualization middleware for e-science (www.visulization.leeds.ac.uk/gViz). They intend to develop a fully Grid-enabled extension of IRIS Explorer, to allow an e-scientist to run Grid applications through the IRIS Explorer user interface. Much of the Grid complexity will thus be hidden from the end-user. Among the few existing Grid related visualization systems, the Access Grid (www.accessgrid.org) has been widely used. It is an open-source remote video conference tool that integrates various resources including multimedia large-format displays, presentation and interactive environments, and interfaces to Grid middleware and to visualization environments. Kong et al. developed a collaborative visualization system over the Access Grid using the ICENI Grid middleware (Kong et al. 2003). Karonis et al. (2003) built a prototype visualization system using the Globus Toolkit, MPICH-G2, and the Access Grid. However, by nature Access Grid is designed for group-group remote video conferences. Even though digital cameras and print-screen stream can be used to share rendered results, the Access Grid system itself does not deal with rendering work directly. Heinzlreiter and Kranzlmuller (2003) introduced a Grid visualization kernel GVK based on the concept of visualization pipe and aimed to develop a universal Grid visualization service. Visualization pipe programming is usually not the best choice for interactive rendering of large-scale architectural models. Furthermore, in their prototype system, only one sided communication is considered and users cannot interact with the visualization process. Another infrastructure, RAVE, introduced by Grimstead et al. (2004) intended to make use of available resources, either local or remote, and react to changes in these resources. Their services connect to the data service, and request a copy of the latest data. This has the same limitations as the vCollab. In summary, Grid-based visualization technology is far from mature till today. How to efficiently make use of distant visualization resources and overcome network bandwidth limitations, especially for large-scale data, is still a research challenge.

3 A NEW GRID-ENABLED VISUALIZATION INFRASTRUCTURE

The Chang'an city is composed of hundreds of detailed 3D models. Hundreds of millions of polygons and numerous high-resolution textures have been used to model these 3D buildings that go far beyond the rendering capability of a single computer. Even though numerous software techniques have been developed to counterbalance the insufficiency of rendering hardware and enable game players to experience a relatively large virtual word using normal Personal Computers (PC), interactively visualizing Chang'an city with fidelity is still far beyond the capability of a normal PC. In general, software approaches improve rendering speed at the sacrifice of accuracy. For any visualization hardware, the number of triangles and texture size that can be handled is fixed. To "squeeze" the Chang'an city into the narrow graphics pipe of a PC graphics card, lots of the charming details of this splendid city will be lost. Thus, sharing the visualization burden over the Grid and making use of remote powerful rendering hardware becomes a natural choice for PC users who want to explore the Chang'an city and experience ancient Chinese architectural beauties. In an idea grid environment, data and visualization resources should be transparent to end users. Visualization tasks can be distributed and balanced among distant heterogeneous resources (visualization supercomputers, clusters, normal PCs, and laptops). However, heavily limited by current network bandwidth and internet technologies, it is obviously impractical to distribute a big real-time visualization task to remote resources which are linked by low speed networks and the internet. To achieve real-time rendering speed with acceptable latency, the networks among the distributed rendering resources have to be fast enough. For interactive visualization of large-scale architectural models like Chang'an city, a practical solution is to distribute the task to a limited number of powerful rendering facilities (visualization supercomputers and clusters) connected by high-speed networks to reduce data communications over networks. Data sources might be kept remotely. A data resource server can be used to coordinate data transmission. Based on these considerations, a grid-enabled visualization infrastructure A-VizGrid (Architectural Visualization Grid) is proposed as follows (Figure 1).

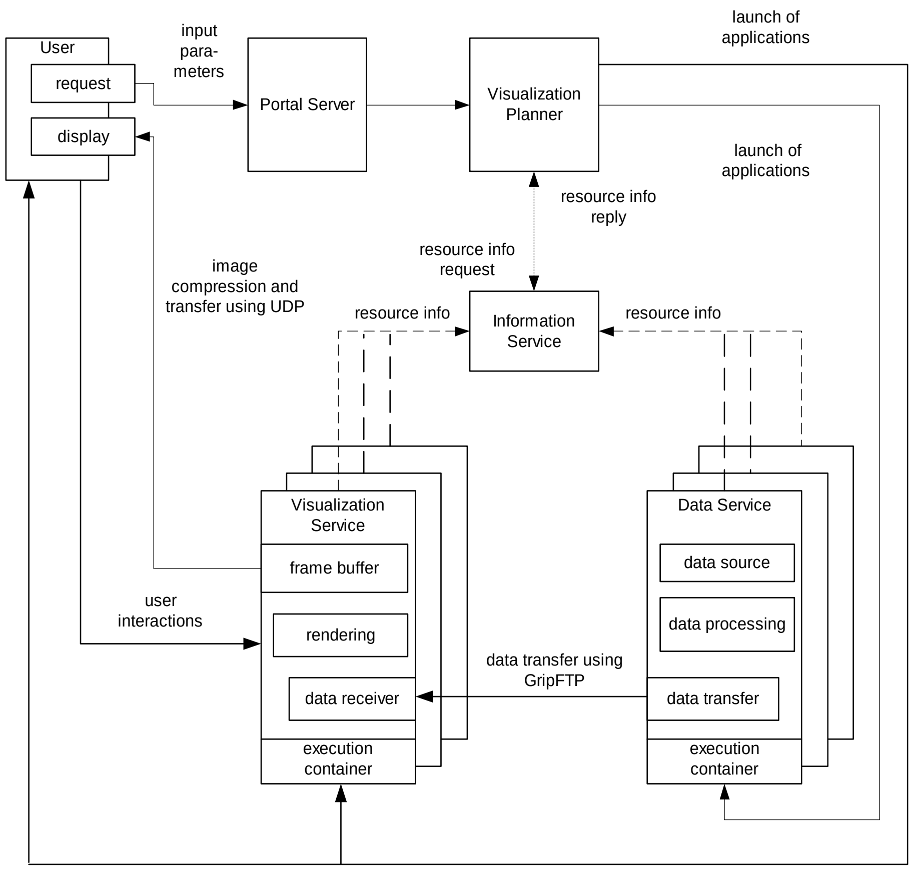


Figure 1 Infrastructure of A-VizGrid

Grid resources are used to do computation and handle data. On top of a resource, a visualization or data service can be deployed. These services are registered into a centralized information service so that they can be located by a visualization planner. A visualization or data service runs within its execution service container which deals with communications over the Grid using Grid-compliant protocols. The following steps are performed to deploy and utilize this Grid-enabled system:

1) The user sends a visualization request to Portal Server. The contents of this request include the districts and buildings that the user wants to navigate, the rendering quality that the user prefers, and what kind of visualization software installed in the user's machine etc.

2) The Portal Server authorizes and forwards the visualization request to the Visualization Planner.

3) The Visualization Planner sends a data resource information request to Information Service. The replied information, including the location, data file size, number of triangles and texture size etc of each available data resource, is returned to the Visualization Planner and used to estimate the visualization resources required to handle these data in real time. Then, the Visualization Planner sends a visualization resource information request to the Information Service and gets the information on available visualization resources.

4) The Visualization Planner allocates the corresponding data resources and visualization resources based on user requirements and the reply from the Information Service. As data transfer speed over networks is usually much slower than that between cluster nodes or using computer system bus, an important criterion of choosing visualization resources is to minimize large data communications over networks.

5) The allocated data are sent to the Visualization Service site using GridFTP. For distributed data files, a data service server is usually used to consolidate and merge them. Data processing and transfer applications are launched by the Visualization Planner through the corresponding execution container at Data Service site.

6) Once data transfer is completed, the Visualization Planner launches the rendering application at Visualization Service site through its execution container. In case a rendering task cannot be handled by a visualization supercomputer/cluster and needs to be distributed among a few rendering facilities, a visualization service master is used to synchronize rendering jobs and handle user interactions.

7) Meanwhile, the Visualization Planner launches the remote visualization application. The rendered results at Visualization Service site are captured and transferred to the user continually. To overcome network bandwidth limit, the rendered results are compressed first before being sent out and the UDP protocol is used. For a detailed analysis of different communication protocols for remote visualization, see (Renambot et al. 2003).

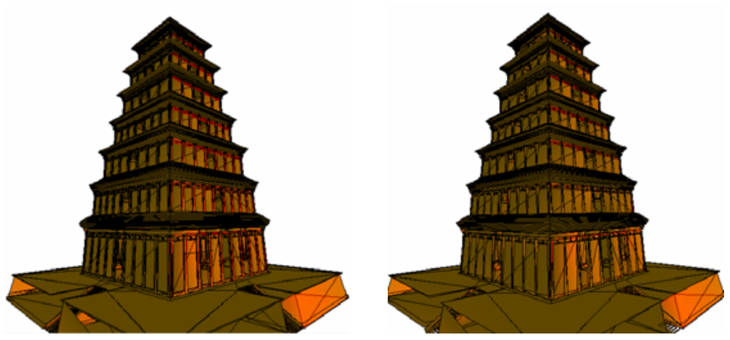
8) At the user site, the compressed images are decompressed and displayed. The user can interact with the remote rendering application using keyboard and mouse commands.

4 MULTI-STAGE DATA COMPRESSION

Grid services provide a mechanism for sharing a heavy computing or visualization burden with distant resources. Compared with the narrow graphics pipe of a single computer, this virtually combined graphics pipe provided by the Grid is much broad. Unfortunately, due to the limitations of current network bandwidth, the capability of this virtual graphics pipe is not unlimited. Especially for interactive visualization of large-scale data, we are unable to implement a pure hardware approach, simply chop a big visualization task and throw the sub-tasks into the virtual pipe. In many cases we still need software techniques as complements.

4.1 Data Pre-Processing

Original models of ancient Chinese buildings designed by commercial architectural software are usually not suitable for interactive rendering. They are composed of too many redundant polygons. 3D model simplification, as a data pre-processing step, is necessary. Surface simplification techniques have been extensively investigated. Hoppe's progressive meshes (Hoppe 1996) and Garland's quadric error metrics method (Garland and Heckbert 1997) have been widely used. Figure 2 shows an example of the Dayan tower simplification. In the picture, we can see that the original 18,756 triangles have been reduced to 9,378 without much visual difference. According to user requirements, 3D models can be simplified into different resolution levels. In our system, three versions of the city, i.e. high, medium and low resolution, are pre-obtained and saved in data source machines. The whole city is divided into several districts for user selection and fast rendering.



(a) Original Model with 18,756 Triangles (b) Simplified Model with 9,378 Triangles

Figure 2 Surface Simplification Applied to the Dayan Tower Model

4.2 Rendering Stage

Parallel rendering is necessary for large-scale data visualization. In a multiple CPU visualization supercomputer, the rendering pipe can be divided into three functional stages, i.e. APP, CULL and DRAW. Each stage can be a separate process. Cluster-based parallel rendering can be categorized into sort-first, sort-middle, sort-last, and hybrid approaches (Molnar et al. 1994). In a typical master-slave system, each rendering node (master and slaves) has a copy of the whole dataset to reduce data communication burden. The master node is used to synchronize the applications running at each rendering node. The rendered results can be collected from their frame buffers and assembled for remote display. According to the experiments done by Staadt et al. (2003), for interactive cluster-based rendering, this deployment is most effective. In a distributed visualization system that is composed of supercomputers and clusters linked by high-speed networks, a similar approach can be deployed. To fit large-scale data into the system memory of each rendering node, data paging technique can be used.

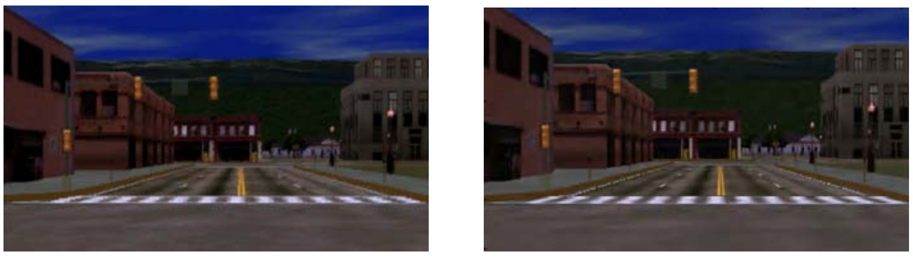
Level of Detail (LOD) is a widely used technique for interactive rendering of large models. Traditional LOD techniques are based on pre-built models at different detail levels. Applications automatically choose a level to draw based on the model's distance from viewpoint. These level models increase system memory request. Another problem is the distracting "popping" effect when the application switches between detail levels. To avoid these problems, view-dependent continuous LOD methods for real-time rendering can be used. The basic idea is to dynamically generate the simplification model according to viewpoint, object and screen resolution. Here we introduce a simple and fast dynamic LOD algorithm based on vertex clustering operations. Before rendering, each vertex is weighted according to its relative curvature value. These values are used to measure the relative importance of each vertex. During rendering, the radius of a bounding sphere centred at vertex *P* is online calculated using the following formula.

$$R = \sqrt{\frac{K \times W \times H}{SW \times SH}} \times \frac{\|OP\|}{\|OP'\|} \times \left(\frac{90^o}{90^o - \theta} \right)^2 \qquad (1)$$

Where the size of near clipping plane is $W \times H$, screen window resolution is $SW \times SH$, rendering error is within K pixels, the projection of vertex P on near clipping plane is P', O is viewpoint, $\|OP\|$ and $\|OP'\|$ are the lengths of line segments and θ is the angle between line OP and the central viewing line (viewing direction). All the vertices inside this bounding sphere are clustered into a single vertex. After vertex clustering, a simplified mesh is obtained. To further improve rendering speed, before vertex clustering process, visibility culling and back-facing culling calculations are carried out to filter out invisible vertices.

4.3 Remote Display Stage

To make use of remote visualization resources, one of the biggest challenges is how to send the rendered results over limited network bandwidth with acceptable latency. When we use Grid services to search for available visualization resources, an important consideration is the network bandwidth and physical distance between the user and visualization resources. We use UDP protocol to transfer rendered results (images). A wavelet-based image compression and decompression algorithm is also integrated with the A-VizGrid system. This algorithm is much efficient than that provided by SGI Vizserver. Figure 3 shows a comparison result. Picture 3(a) is the result using the default image compression algorithm provided by Vizserver. The compression ratio is 1:32. Compared with the result using wavelet compression (Figure 3(b)), its visual quality is obviously worse, even if the wavelet algorithm used a much higher compression ratio 1:128.



(a) 1:32 Compressed by Vizserver **(b) 1:128 Compressed by wavelets**

Figure 3 Image Compression using Different Algorithms

5 PROTOTYPE DEVELOPMENT

Based on the A-VizGrid infrastructure, a prototype system is being developed. The data resources are located at NUS. Available visualization resources include a SGI Onxy2 with four InfiniteReality Graphics pipes at the Institute of High Performance Computing (IHPC), a 5-node Linux cluster with 10 AMD64 CUPs and Nvidia Quadro FX3000G graphics cards at IHPC, and a SGI Onyx3000 at the Nanyang Technological University. The information service, visualization planner and execution container etc, are OGSA-based Grid services. Execution containers and the portal server have been implemented using Java (Ho and Khoo 2004). In the SGI supercomputers, we use OpenGL Performer with enhanced culling and view-dependent continuous LOD algorithms to handle large-scale architectural models. The rendered results are remotely displayed using Vizserver integrated with a wavelet-based image compression/decompression algorithm. In the visualization cluster system, we use Chromium to run OpenGL applications (Humphreys et al. 2002). We also use the CAVELib to control and synchronize OpenGL Performer applications. Figure 4 and 5 are two screen snapshots of the Chang'an city.

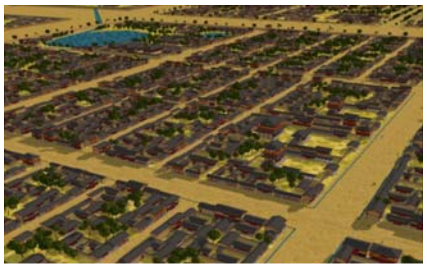


Figure 4 Bird's-Eye View of the City

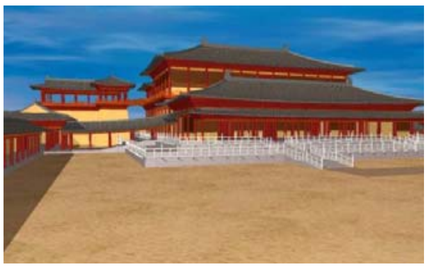


Figure 5 Navigation in the Linde Hall

6 CONCLUSION

A practical Grid-enabled visualization infrastructure has been presented in this paper. It intends to provide efficient services for interactive visualization of large-scale architectural models, such as Tang Chang'an city. It is easy to implement and the underlying Grid services are developed according to the OGSA standard. To tackle the critical problem of Grid visualization, i.e. data size and network bandwidth, a multi-stage data compression approach has been deployed and the corresponding data pre-processing, rendering and remote display issues have been systematically addressed.

ACKNOWLEDGEMENTS

We would like to thank the members of the Grid team at IHPC for valuable discussions and sharing of information with respect to the proposed grid-enabled visualization infrastructure. We also thank the Centre for Industrial Mathematics of NUS for use of their patented image compression algorithm. This research is supported by the Academic Research Fund of NUS under contract number R-295-000-037-112.

REFERENCES

Foster, I., C. Kesselman, and S. Tuecke. 2001. The anatomy of the Grid: enabling scalable virtual organizations. *The International Journal of High Performance Computing Applications* 15(Fall): 200-222.

Foster, I., and I. Kesselman. 2003. *The Grid 2: blueprint for a new computing infrastructure* [2nd ed.]. Morgan Kaufmannn.

Garland, M., and P.S. Heckbert. 1997. Surface simplification using quadric error metrics. In *ACM SIGGRAPH Proceedings*, 209-216. New York: ACM Press.

Grimstead, I.J., N.J. Avis, and D.W. Walker. 2004. RAVE: resource-aware visualization environment. In *Proceedings of the UK e-Science All-Hands Meeting*, ed. Simon Cox: 137-140. Nottingham.

Heinzlreiter, P., and D. Kranzlmuller. 2003. Visualization services on the Grid: the Grid visualization kernel. *Parallel Processing Letters* 13(June): 135-148.

Heng, Chye Kiang. 1999. *Cities of aristocrats and bureaucrats: the development of medieval Chinese cities.* Honolulu: University of Hawai'i Press.

Ho, Q.T., and B. Khoo. 2004. *Development of execution containers for Grid-based visualization*. Technical report (IHPC/SNC/Grid-04-11), Institute of High Performance Computing, Singapore

Hoppe, Hugues. 1996. Progressive meshes. In *ACM SIGGRAPH Proceedings*, 99-108.

Humphreys, G., M. Houston, R. Ng, and R. Frank. 2002. Chromium: a stream-processing framework for interactive rendering on clusters. In *ACM SIGGRAPH Proceedings*, 693-702. New York: ACM Press.

Karonis, N.T., M.E. Papka, J. Binns, and J. Bresnahan. 2003. High-resolution remote rendering of large datasets in a collaborative environment. *Future generation Computer Systems* 19(August): 909-917.

Kong, G., J. Stanton, S. Newhouse, and J. Darlington. 2003. Collaborative visualisation over the Access Grid using the ICENI Grid middleware. In *Proceedings of the UK e-Science All Hands Meeting*, ed. Simmon Cox: 393-396. Nottingham.

Molnar, S., M. Cox, D. Ellsworth, and H. Fuchs. 1994. A sorting classification of parallel rendering*.* *IEEE Computer Graphics and Applications* 14(July): 23–32.

Renambot L., T. Schaaf, H. Bal, D. Germans, and H. Spoelder. 2003. Griz: experience with remote visualization over an optical grid. *Future Generation Computer Systems* 19(August): 871-881.

Shalf, J., and E.W. Bethel. 2003. The Grid and future visualization architectures. *IEEE Computer Graphics and Applications* 23(March/April): 6–10.

Staadt, O.G., J. Walker, C. Nuber, and B. Hamann. 2003. A survey and performance analysis of software platforms for interactive cluster-based multi-screen rendering. In *Eurographics Workshop on Virtual Environments*, ed. Joachim Deisinger, and Andreas Kunz: 261-270. Zurich.