

Performance Analysis of a 3D Parallel Volume Rendering Application on Scalable Tiled Displays

Xingfu Wu¹, Valerie Taylor¹, Jason Leigh², and Luc Renambot²

¹ Department of Computer Science, Texas A&M University, College Station, TX 77843

² Electronic Visualization Laboratory, University of Illinois at Chicago, Chicago, IL 60607
{wuxf@cs.tamu.edu, taylor@cs.tamu.edu, spiff@evl.uic.edu, luc@evl.uic.edu}

Abstract

A graphics cluster with new display technology and off-the-shelf, inexpensive components has made possible affordable large-scale high-resolution display systems. In this paper, we present the performance results from analyzing a 3D parallel volume rendering application, called Volatile, executed on a practical scalable tiled display infrastructure. Volatile allows users to update transfer functions, to rotate/zoom a 3D scene at real-time rates on scalable tiled displays. Prophecy system is used to monitor and analyze the performance of the visualization application, focusing on the performance characterization of I/O, communication, computation, and scalability. The experimental results indicate that the data access time dominates the application execution time, requiring at least 90% of the execution time for the datasets with more than 16MB.

Keywords---Parallel volume rendering, visualization applications, Linux graphics cluster, performance analysis.

1. Introduction

Today, a new renaissance in 3D graphics area is under way, driven by fully programmable GPU (Graphics Processing Unit) that delivers thousands of times the graphics power available ten years ago. Combining parallel processing with modern, high-level programming languages, today's GPU can process tens of millions of vertices per second and rasterize hundreds of millions or even billions of fragments per second. The GPU exceeds the CPU in the number of transistors present in each microchip [1]. For example, Intel's 2.4 GHz Pentium 4 has 55 million transistors; NVIDIA used over 125 million transistors in the original GeForce FX GPU. Further, the GPU is much faster for graphics tasks, such as rendering 3D scenes, than a general-purpose CPU. Currently, the single-chip GPUs, designed for PCs and video game consoles are far more powerful and much cheaper than any previous graphics systems.

Current high-speed general-purpose networks, such as 1Gigabit/10Gigabit networks, are fast enough to handle the demanding tasks of routing streams of graphics primitives. Systems built with such networks and off-the-shelf GPUs and PCs, are being used to provide graphics clusters, which are more economical than expensive graphics supercomputers. Further, the current trend in building high-resolution display systems is to tightly couple inexpensive LCD/TFT monitors to provide a large-scale high-resolution display system for detailed scientific visualizations with an increased pixel density, such as the GeoWall [3] or Lightning-2 [9]. The graphics clusters can drive the large-scale high-resolution display systems to possibly replace the limited output resolution of standard devices such as monitors and video projectors.

In this paper, we present a practical scalable tiled display infrastructure that consists of a five-node Linux graphics cluster. Further, we present the performance results from analyzing a 3D parallel volume rendering application, called Volatile, executed on the infrastructure. The Prophecy system [11] is used to instrument, monitor and analyze the performance of the application. The analysis focuses on the performance characterization of I/O, communication, computation, and scalability of Volatile.

The remainder of this paper is organized as follows. Section 2 presents the visualization cluster and a scalable tiled display infrastructure. Section 3 describes three major components of the 3D parallel volume rendering application, Volatile. Section 4 uses Prophecy to monitor and explore the performance and behaviors of Volatile. Section 5 concludes the paper, and discusses the further research work.

2. OptIPuter Visualization Cluster

Our five-node visualization cluster is one OptIPuter [6] node deployed at Texas A&M University [15]. It is a linux graphics cluster with dual Opteron 240 w/1MB OD cache and nVidia Quadro FX 3000 with 256MB of video memory on each node. The operating system is SuSe Linux 9.0 for 64bit AMD Opteron version. To support

3D graphics, we installed OpenGL, Cg [1], and Mesa 6.0.1. To support MPI programs, we installed MPICH2.

2.1. Graphics Pipeline Architecture

Figure 1 provides an overview of the graphics pipeline from the CPU to the GPU on each node. To render a scene, the application, which is running on the CPU, must send data and instructions across to the graphics device. The application communicates with the graphics device through the driver and AGP (Accelerated Graphics Port) bus. Once the device has the data, it processes the data using the graphics chip itself, and writes the data out to the frame buffer in the video memory. The video memory typically stores static or semi-static geometry, the command stream, textures, and the frame buffer. On the graphics chip, there are some caches to handle buffering and to ensure that the pipeline is running as efficient as possible.

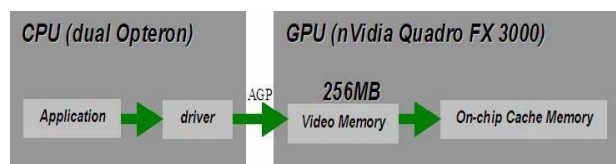


Figure 1 Overview of graphics pipeline from the CPU to the GPU on each node

To take advantage of the graphics pipeline, a 3D parallel volume rendering application, called Volatile, was developed using Cg [1] and OpenGL to directly control the shape, appearance, and motion of objects drawn using programmable GPUs at real-time rates.

2.2. Scalable Tiled Display Infrastructure

The display infrastructure consists of a five-node Linux visualization cluster with a Gigabit network, a 2x2 tiled display, and a LCD 20.1' master monitor. The tiled display consists of four LCD 20.1' panels arranged in 2x2 configuration for a whole resolution of 3200x2400 pixels. Each node is connected to one LCD monitor of the tiled display using a DVI cable.

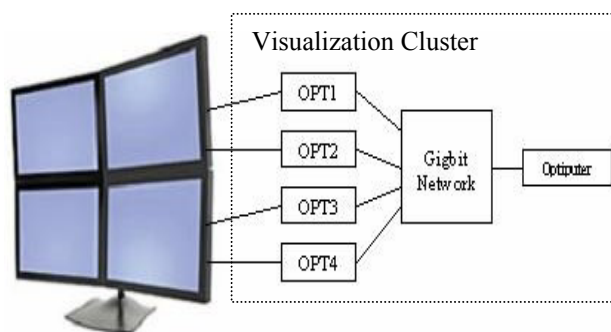


Figure 2. Scalable tiled display infrastructure

The tiled display can scale with an increase in the cluster size [16]. When scaling up the number of nodes of the cluster, it is easy to scale up the number of LCD monitors, making possible higher resolution tiled displays. Further, the messages that are broadcasted to nodes are small in size and are independent of the scene size and number of tiles. This also ensures the scalability of Volatile for larger scene sizes on larger tiled displays.

3. 3D Parallel Volume Rendering Application: Volatile (Vol-a-Tile)

In this section, we describe the framework of Volatile using three scenarios. A sort-first strategy is used to distribute the graphic updates in Volatile.

3.1. Framework of Volatile

Volatile was developed by the Electronic Visualization Laboratory at UIC [8, 13]. It is a volume visualization tool for large-scale, time-series scientific datasets rendered on high-resolution scalable displays. These large-scale datasets can be dynamically processed and retrieved from remote data stores over optical networks using OptiStore --- a system that filters raw volumetric data and produces a sequence of visual objects such as iso-surfaces or voxels. Volatile focuses on large volumetric data and time-dependent simulation results, such as ultra-high-resolution seismic and microscopy data. Volatile consists of three major components shown in the Master node of Scenario 1 in Figure 3:

- Data server, **Optistore**
- Main graphics program, **Volvis**
- Transfer function Editor, **tfUI**

Optistore is the data server, which stores objects as 3D volumes or geometry. The Optistore is designed to assist visualization dataset handling, including data management, processing, representation and transport. It is built upon existing functionality in VTK [12] such as the marching cubes, gradient estimation, data reduction and sampling filters.

Volvis handles the rendering, scalable displaying and user interaction. All the nodes in Volatile (master and clients) have a dedicated link to the Optistore to retrieve the datasets. The master node handles user interaction and any transfer function updates from the transfer function editor, **tfUI**; the master node uses MPI to broadcast function updates to the clients in order to maintain a consistent graph on each client. The broadcast messages are very small, containing only the operations to be performed on the scene, and are independent of the scene size or the number of tiles. This ensures the scalability of Volatile for larger scene sizes on larger tiled displays. The master node does not calculate any client's scene view. The client nodes are responsible for

processing commands from the master node and rendering to their respective view frustums.

tfUI is the user interface for transfer function selection based on the Simian system developed at University of Utah [2]. The color and opacity can be selected using the classification widgets. These widgets can be overlaid and then rasterized to a 2D texture, which is sent to Volvis. The 2D histogram for the dataset is retrieved from Volvis and displayed to guide the user in the selection.

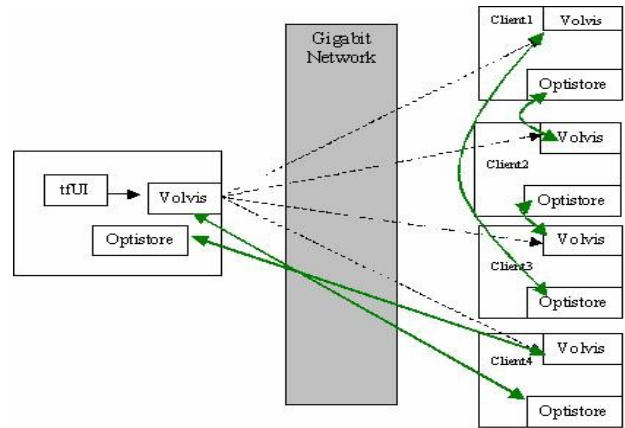
For Volatile, all graphics commands from the API on the master node are marshaled and sent to all client nodes instead of being executed locally. This makes all cluster-related issues hidden from users, and makes the system easy to use and operate like using a single graphics workstation. In contrast, other systems such as WireGL [4], allow users to issue graphics commands directly to each node. WireGL requires cluster-related background, which can make it hard to use because of the need to maintain a consistent scene graph throughout the cluster.

The performance analysis experiments presented in Section 4 utilize three different scenarios for the cluster systems. The scenarios, which explore different configurations of the three Volatile components, are given in Figure 3 and described below:

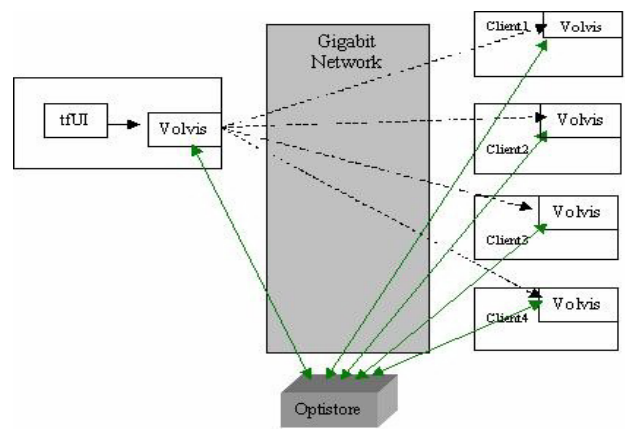
Scenario 1: Each data server (Optistore) is set to be local such that each volvis contacts its local data server to get raw data.

Scenario 2: Each data server (Optistore) is set to be remote such that each volvis contacts its corresponding remote data server to get raw data.

Scenario 3: Only one data server (Optistore) is set to be outside the cluster so that each volvis contacts the remote data server to get raw data over the network.

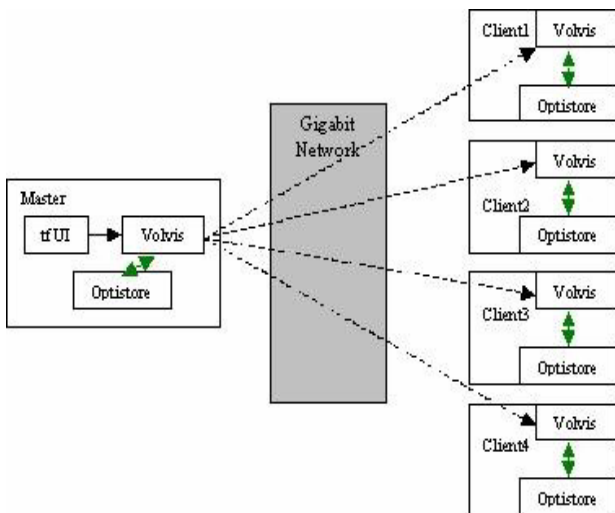


Scenario 2



Scenario 3

Figure 3. Three scenarios of Volatile



Scenario 1

3.2 Sort-first partitioning

Parallel graphics architectures can be classified according to the step in the graphics pipeline at which data is redistributed [5]. This redistribution step is the transition from object parallelism to image parallelism, and the location of this redistribution has significant implications for the architecture's communication needs. In this section, we describe the sort-first strategy to distribute the graphic updates in Volatile.

The basic idea behind Volatile's data redistribution and that of other similar methods such as Griz [7] and WireGL [4], is the following: divide the raw volumetric data into smaller sub-volumes, distribute them to multiple nodes, render them independently and locally, and combine the resulting images on scalable tiled displays. It is also important to minimize the amount of data that must be communicated between nodes during the combination.

For Volatile, the screen is divided into disjoint regions, for which the number of regions is equal to the number of tiles. The client nodes, which equal the number of tiles, are responsible for all rendering

calculations, which affect their respective screen regions. The screen regions (view frustums) are calculated as percentages, and are configured individually. For example, the 2x2 tiled display has the following configuration:

1	3
2	4

For each tile from bottom-left onwards, the configuration file for the tiled displays is given below:

#	X0	X1	Y0	Y1
2	M_X	M_X+R_X	M_Y	M_Y+R_Y
1	M_X	M_X+R_X	M_Y+R_Y $+2*M_Y$	100
4	M_X+R_X $+2*M_X$	100	M_Y	M_Y+R_Y
3	M_X+R_X $+2*M_X$	100	M_Y+R_Y $+2*M_Y$	100

where M_X , M_Y are the percentages of X-mullion, Y-mullion of each LCD, respectively; R_X , R_Y are the percentages of width, height of each LCD according to the total width and height of the tiled display. Volatile uses sort-first strategy to partition a 3D scene graph along X and Y dimensions, and keeps the Z dimension in full range as a default. When scaling up the graphics cluster, it is easy to generalize the above method to support any size tiled displays.

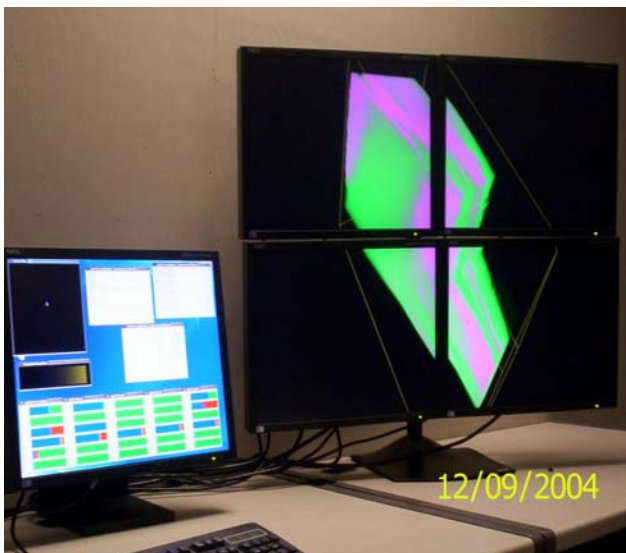


Figure 4. Snapshot of the whole tiled display with geological image using Volatile

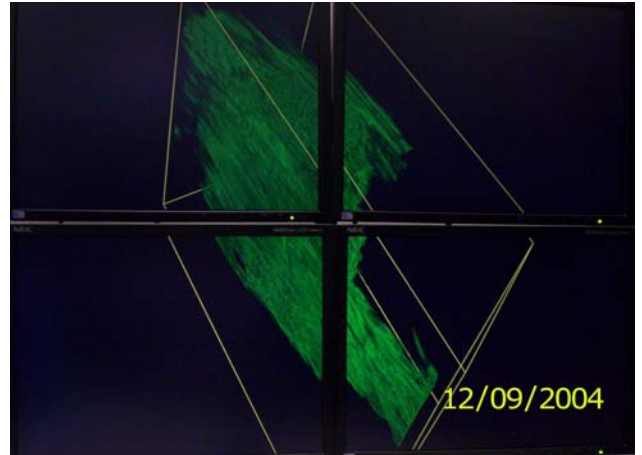


Figure 5. Snapshot of geological image after reducing the volume data using tfUI

Figure 4 provides a snapshot of the whole tiled display infrastructure with a geological image using Volatile, which consists of one master display and one 2x2 tiled display. The master display consists of the Volatile control panel window (top, black), the transfer function editor window (middle, black), and five performance monitoring windows (bottom). Users can interact with the scene on the tiled displays using a mouse, keyboard, or joystick. After using tfUI to update transfer functions (i.e. reduce the volume data and color), the result is given in Figure 5. This update process is real-time. Volatile also allows users to rotate, or zoom in/out a 3D scene, toggle roam volume mode, and cut the plane at real-time rates.

4. Performance Analysis

In this section, we monitor and explore the performance and behaviors of Volatile, and use the Prophecy system to analyze the performance of the visualization application.

We monitor the machine load, CPU, memory usage, network, and disk usage for each node of the cluster; the results of the monitoring are available on the master node. The performance monitors at the bottom of the master display shown in Figure 4 display the status of several system-based parameters. Each monitored component is displayed as a horizontal bar, which is separated into color-coded regions; each region represents a percentage of the resource that is being used. The Prophecy system is used to instrument the code such that we can collect the detailed performance data of the application with different datasets on different number of nodes in order to further analyze the application performance. Table 1 lists the different datasets used in these experiments.

Table 1. Raw datasets

Name	Dimensions	Size (KB)
Protein64x64x64	64x64x64	256
Fuel64x64x64	64x64x64	256
Furdave160x255x75	160x255x75	2989
Foot256x256x256	256x256x256	16384
Geo256x256x256	256x256x256	16384
Geo440x290x198	440x290x198	24673

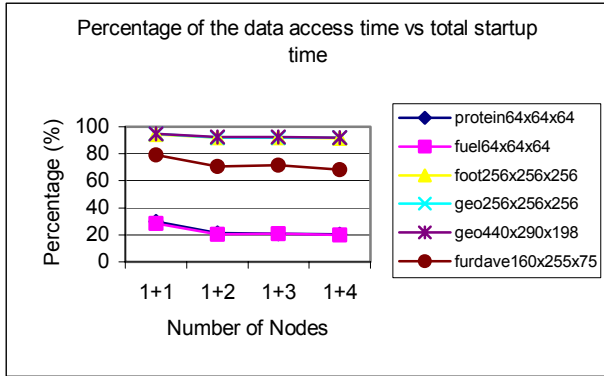


Figure 6. Performance of Scenario 1

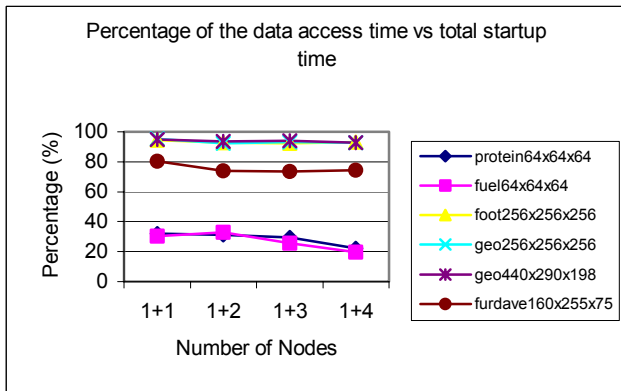


Figure 7. Performance of Scenario 2

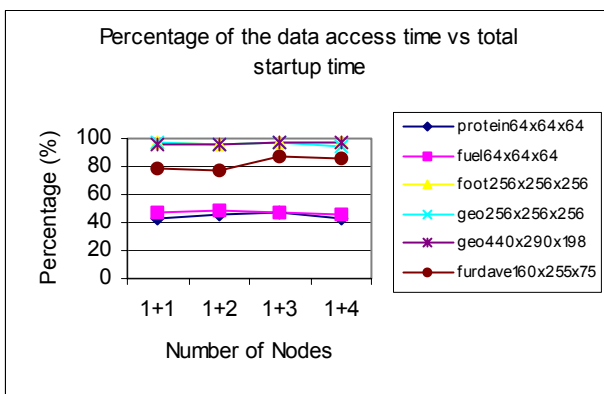
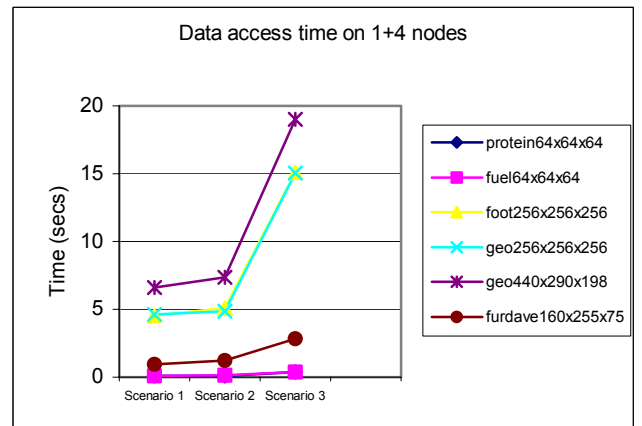
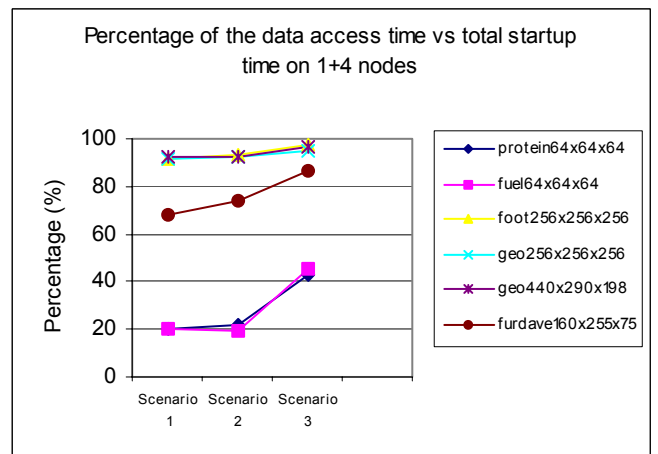


Figure 8. Performance of Scenario 3

Figures 6-8 provide the performance results on different number of nodes for three scenarios, where 1+4 nodes denotes 1 master server and 4 client nodes. These figures provide the percentage of data access time for different datasets and different number of processors. The percentage is given in terms of total startup time, which includes all of the processing and accessing necessary to obtain the initial display. These experimental results show that, with the increase in the dataset size, the data access time dominates the total startup time of Volatile. For small datasets, such as protein64x64x64 and fuel64x64x64, the percentage of the data access time is less than 50%; for large datasets, such as foot256x256x256, geo256x256x256, and geo440x290x198, the percentage is more than 90%. For the even node-screen partition, the total startup time does not change much with increase in the number of nodes.



(a) Data access time



(b) Percentage

Figure 9. Performance comparison of three scenarios

To compare the three scenarios, we use the performance data on only 1+4 nodes. Figure 9 indicates that the data access time increases significantly when going from scenario 1 to scenario 3. The increase occurs because of the increase in communication time due to the

one remote Optistore that all clients must access. The results for Scenario 1 and 2 are similar because of the use of the Gigabit network. Thus, when the raw datasets are remotely distributed, the use of a high speed network, such as an optical network in Scenario 2, can achieve similar results as that in Scenario 1. Further, the data access time in Scenario 3 increases significantly in Figure 9(a); this indicates that parallel I/O should be applied to improve the I/O performance when there is only one Optistore.

Conclusions and Further Work

In this paper, we presented the OptIPuter visualization cluster, which includes a scalable tiled display infrastructure. This infrastructure was used to execute a 3D parallel volume rendering application, called Volatile. We analyzed the performance of this application, with three different configurations that explored different methods for placing the image data or Optistore. The results indicated that for small datasets the percentage of the data access time is less than 50%; for large datasets the percentage is more than 90%. Further, for the even node-screen partition, the total startup time does not change much with increase in the number of nodes.

For future work, we plan to get static/time-varying large datasets (up to several TBs) to test the visualization application on large-scale OptIPuter testbeds [6] that utilize wide-area optical networks. Further, kernel coupling [10, 14] techniques will be used to explore and quantify the interactions among the three components to better understand the performance requirements in terms of I/O, memory, computation, and communication.

Acknowledgements

This work is supported by the NSF OptIPuter Project (ITR grant ANI-0225642). We acknowledge the reviewers for their comments and suggestions.

References

- [1] Randima Fernando and Mark J. Kilgand, *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics*, Addison-Wesley Publishing Company, 2003. Also see the web site http://developer.nvidia.com/object/cg_toolkit.html.
- [2] Christiaan Gribble, Xavier Cavin, Mark Hartner, and Charles Hansen, Cluster-based Interactive Volume Rendering with Simian, *Tech. Report UUCS-03-017*, School of Computing, University of Utah, Sep. 3, 2003.
- [3] The Geowall Consortium, <http://www.geowall.org>.
- [4] Greg Humphreys, Matthew Eldridge, Ian Buck, Gordon Stoll, Matthew Everett, and Pat Hanrahan, WireGL: A Scalable Graphics System for Clusters, in *SIGGRAPH 2001*, ACM Press, 2001.
- [5] Steven Molnar, Michael Cox, David Ellsworth, and Henry Fuchs, A Sorting Classification of Parallel Rendering, *IEEE Computer Graphics and Applications*, 14(4), July 1994.
- [6] The OptIPuter project, <http://www.optiputer.net/>.
- [7] Luc Renambot, Tom van der Schaaf, Henri E. Bal, Desmond Germans, and Hans J.W. Spoelder, Griz: Experience with Remote Visualization over an Optical Grid, *Journal of Future Generation Computer Systems (FGCS)*, Elsevier Science Press, Volume 19, Issue 6, August 2003, pp. 871-882.
- [8] N. Schwarz, S. Venkataraman, L. Renambot, N. Krishnaprasad, V. Vishwanath, J. Leigh, A. Johnson, G. Kent, and A. Nayak, Vol-a-Tile – a Tool for Interactive Exploration of Large Volumetric Data on Scalable Tiled Displays, *IEEE Visualization 2004 (Poster)*.
- [9] G. Stoll, M. Eldridge, D. Patterson, A. Webb, S. Berman, R. Levy, C. Caywood, M. Taveira, S. Hunt, and P. Hanrahan, Lightning-2: A High-Performance Display Subsystem for PC Clusters, in *SIGGRAPH 2001*, ACM Press, 2001.
- [10] Valerie Taylor, Xingfu Wu, Jonathan Geisler, and Rick Stevens, Using Kernel Couplings to Predict Parallel Application Performance, *Proc. of the 11th IEEE International Symposium on High-Performance Distributed Computing (HPDC 2002)*, Edinburgh, Scotland, July 24-26, 2002.
- [11] Valerie Taylor, Xingfu Wu, and Rick Stevens, Prophecy: An Infrastructure for Performance Analysis and Modeling of Parallel and Grid Applications, *ACM SIGMETRICS Performance Evaluation Review*, Volume 30, Issue 4, March 2003.
- [12] The Visualization ToolKit (VTK), <http://www.vtk.org/>.
- [13] Vol-a-tile, <http://www.evl.uic.edu/cavern/optiputer/volatile.html>.
- [14] Xingfu Wu, Valerie Taylor, Jonathan Geisler, and Rick Stevens, Isocoupling: Reusing Coupling Values to Predict Parallel Application Performance, *the 18th International Parallel and Distributed Processing Symposium (IPDPS2004)*, Santa Fe, New Mexico, April 26-30, 2004.
- [15] Xingfu Wu and Valerie Taylor, Design and Development of a Linux Visualization Cluster for OptIPuter project, *Technical Report*, Department of Computer Science, Texas A&M University, 2004.
- [16] Xingfu Wu, *Performance Evaluation, Prediction, and Visualization of Parallel Systems*, Kluwer Academic Publishers, ISBN 0-7923-8462-8, Boston, USA, 1999.