A Distributed Scientific Data Visualization Framework

Alan M. Shih*, Ramkrishna Chakrabarty**

* University of Alabama at Birmingham, Birmingham, AL 35294, USA
(Tel: 205-934-8461; e-mail: ashih@uab.edu)
** Pennsylvania State University, University Park, PA 16802, USA
(e-mail: rkchak@psu.edu)

Abstract: High-fidelity scientific simulations generate large amounts of data that need to be analyzed. Scientific visualization using computer graphics has become one of the major approaches to analyzing and interpreting such data. However, due to the ever-growing amount of data generated by the simulations, it is now a major challenge for scientists to visualize and analyze such extreme-scale datasets to explore the crucial information contained within. The efficiency of visualization techniques and approaches is therefore immensely important. There are two major challenges in addressing this issue. First, the management of these huge quantities of data generated from various scientific experiments and simulations. Second, achieving a higher frame rate for visualization of the enormous amounts of data is necessary. In this paper we present a distributed visualization framework that addresses these two challenges by designing efficient and scalable database-based data storage implementations for high-performance computing data. We also provide a proof-of-concept distributed visualization technique for visualizing streamlines that uses the abovementioned storage mechanism.

Keywords: Scientific Visualization, High Performance Computing, Linux Clusters

1. INTRODUCTION

High-performance visualization systems such as tiled display walls have become a popular way to display scientific data in high resolution. The tiled display wall is achieved by the highly accurate alignment of multiple projectors and is powered by a high-performance graphics cluster. With a configuration of nine - projectors each at a 1024 x 768 resolution, a tiled display can support a combined resolution of 3072 x 2304 pixels, which is equivalent to nearly 7.1 million pixels. The use of such scalable display systems has enabled the display of high-resolution images with astonishing clarity. However, most of the tiled display wall systems today deal only with the rendering portion of the entire visualization pipeline, and thus do not efficiently use the computing power of the high-performance rendering cluster that drives the display wall. The entire visualization pipeline is implemented in a serial manner that reads data and is executed on the master node. Since visualization is a computation-and data-intensive process, this high-performance cluster can be used for computation as well as the data storage involved in visualization.

In this research effort is that if the high-performance cluster is also used to perform complex computations involved in the entire visualization pipeline over and above rendering, then hypothetically, the overall performance will be significantly improved. There are two major challenges to addressing this issue. First, there is the efficient management of these huge quantities of data generated from various scientific experiments and simulations. Second, improving computations for visualization involving enormous amount of data is needed.

2. DESIGN

An end-to-end framework has been implemented for distributed scientific data management and visualization. The distributed nature of the framework is primarily due to the distributed implementation of the visualization algorithms. This design consists of following main components:

1. Master Workstation. This is a high-end workstation meant to initiate the algorithm, oversee the running of the framework, and render the geometry obtained at the end of the visualization pipeline.
2. Visualization Cluster. This should be a computational cluster used for executing the compute-intensive visualization pipeline. The cluster of computers helps in faster execution of the visualization algorithms facilitated by task parallelism and distribution.

3. Data Storage. This can be applied to any modality, such as a database, a parallel database, a regular data file, or a data grid, depending on the user’s choice.

The dependence of the framework on the various dispersed resources for executing various tasks of the visualization pipeline makes it a distributed visualization framework. This framework has been implemented by designing and developing a collection of software tools that facilitate in the execution of the pipeline. The software that drives the framework comprises of the following components:

*Data Storage:* This component implements the various types of data storage paradigms supported by the framework. This includes data files, databases, and parallel databases. However, this component can be easily extended to other modalities, such as data grids.

*Data Abstraction Layer:* This software acts as a wrapper surrounding the data storage component, thus enabling the visualization component to be independent of the data storage implementation.

*Distributed Visualization Engine:* This software module of the framework implements the visualization algorithms for a distributed environment. In this work a distributed version of streamline visualization technique is developed.

### 3. DATA ABSTRACTION LAYER

One of the most important features of the proposed distributed framework is its interoperability across central databases, parallel databases or data file as the data storage implementation. However, it will be very inefficient to implement visualization techniques if it has to interface them with all of the three different types of data storage supported. On the other hand, it will be difficult to implement new data storage types if it is necessary to interface them with all of the visualization techniques implemented in the framework. This creates the need for a Data abstraction layer between the DVE and data storage. This way, the DAL eliminates the need for different visualization techniques to be interfaced with specific data storage implementations. DAL exposes a standard API on both sides, allowing DVE and data storage to interact with each other and thus remain transparent to each other. It will also allow different applications other than visualization to access the data through the same APIs provided in DAL. The DAL is developed keeping only mesh-related data in mind.

There are several salient features in DAL:

1. The DAL hides all of the complexities of all the different data storage implementations such as central and parallel databases.
2. DAL provides a simple, common, and unified interface (API) for the client application.
3. When using the DAL API, the client application requires no database knowledge. For example, the client need not create any tables. DAL encapsulates table creation, SQL statements to fetch data, and all the rest database specific jobs.
4. This design helps make the DAL scalable to other types of database that does not have the concept of tables such as Object Oriented Databases or the Data Grid.
5. It is implemented as a C++ class library.

### 4. BENCHMARKING

The test was conducted on a 10-node dual-processor Linux cluster. Here a test dataset was initially uploaded to all of the three supported data storage implementations. Then 1, 2, 4, and 8 streamlines are generated on 1, 2, 4, and 8 nodes with two processors each respectively.

In this metric, the times taken by different data storage implementations are compared with a varying number of streamlines computed. This test is conducted to study the effect of the number of streamlines and thereby the number of processors used simultaneously on different data storage implementations. The results from this metric show that the times taken by all of the three different data storage implementations do go up as the number of streamlines computed is increased. However, the data file implementation becomes slower than the database implementations as the number of simultaneous streamlines computed is increased. As is clear from Figure 1, the data file implementation is faster when just one streamline is computed. However, as the number of streamlines
computed is increased, the number of processes accessing the same file also increases, and thus the data file implementation becomes slower than the database implementations. Following are some of the snapshots of the visualizations done using the distributed visualization framework.

![Comparison between times taken to compute varying number of streamlines](image)

Figure 1: Comparison between times taken to compute varying number of streamlines for the test dataset

5. CONCLUSION

Distributed visualization is useful in increasing the performance of visualization techniques for large datasets. This research effort studied the effect of different data storage implementations, such as data files, central databases, and parallel databases, for high-performance computing data on distributed visualization techniques. Efficient data storage and management remains one of the challenges of any large data visualization system. The main motivation of this work is to explore the performance differences of data storage implementations using database and traditional flat data files in the distributed visualization framework, especially in the case of large datasets. To achieve this objective, an elaborate distributed visualization framework was developed that supports data files, central databases, and parallel databases. A distributed version of the streamline algorithm was also developed to serve as the test-bed algorithm for comparison. Various fast data retrieval methods were implemented to improve the performance, such as the octree search algorithm and cell caching. Also, a novel scheme was developed to store a hierarchical and irregular data structure such as octree in a database.

The results of the metrics used for measuring the performance of the distributed visualization framework have shown that although data files turn out to be the most efficient in fetching a cell, in the case of computing an entire set of streamlines, the parallel database is the fastest. This can be attributed to various kinds of sophisticated search mechanisms already built into databases. Also, in databases there is concurrent synchronous access to multiple users, which is absent in normal data files, making them a slower data storage implementation.