Conf-9409278--4

PNL-SA-24897

VISUALIZATION OF PARALLEL MOLECULAR DYNAMICS SIMULATION ON A REMOTE VISUALIZATION PLATFORM

J. B. Nicholas
T. Y. Lee[a]
C. S. Raghavendra[a]

September 1994

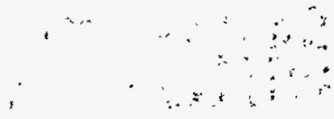
Presented at the
GViz 1994 Conference
September 8, 1994
Richland, Washington

Prepared for
the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest Laboratory
Richland, Washington 99352

(a) Washington State University, Pullman, Washington

DISCLAIMER



Visualization of Parallel Molecular Dynamics Simulation on a Remote Visualization Platform

Tong-Yee Lee, C.S. Raghavendra
School of EE/CS
Washington State University
Pullman, WA 99164

John B. Nicholas
Molecular Science Research Center
Pacific Northwest Laboratory
Richland, WA 99352

Abstract

Visualization requires high performance computers. In order to use these shared high performance computers located at national centers, we need an environment for remote visualization. Remote visualization is a special process that uses computing resources and data that are physically distributed over long distances. In our experimental environment, a parallel raytracer is designed for the rendering task. It allows us to efficiently visualize molecular dynamics simulations represented by three dimensional ball-and-stick models. Different issues encountered in creating our platform are discussed, such as I/O, load balancing, and data distribution.

1 Introduction

Scientific visualization is rapidly becoming an important part of scientific discovery, enabling researchers to explore more of their data with greater efficiency and increased comprehension. Generally, visualization [1] has been characterized as a process consisting of numerical simulation, rendering, data conversion, and image animation. This whole process requires very intensive computation power and sophisticated graphics hardware.

Several software packages for scientific visualization are available [2, 3]. However, these packages generally run only on workstation-class machines that are often not powerful enough to produce real-time visualization. This is particularly true if we visualize huge datasets with sophisticated rendering algorithms, such as raytracing. Furthermore, with significant increases in the size of the dataset, memory limitations can become a problem on workstations. Another approach is to use high-end graphics workstations as the visualization clients and submit the most computationally intensive rendering tasks to a local Supercomputer Server. The San Diego Supercomputer Center uses this approach to provide volume visualization service in the NetV system [4]. However, not all universities and research centers can afford expensive supercomputer and advanced graphics hardware. Another approach is to use a network of workstations. With the increasing speed and decreasing cost of advanced workstations, an affordable graphics platform can be created by grouping many workstations through a local network. We previously experimented with visualizing oceanography simulation [5] and thermal fluid dynamics [6], using Silicon Graphics or Sun workstations connected by a local network. In this configuration, we used a client-server model, where a master node handles scheduling of both simulation and rendering tasks among all clients.

The approach we present in this paper is to use a local workstation to view data created and rendered on a high performance parallel computer at a remote site. In this method, both the simulation and rendering computations can all be done in parallel machines. Thus, image frames are the only data transferred between the two sites. The massive amount of numeric data produced by the simulation is rendered on the parallel computer, avoiding the need to transfer the data to the local workstation. In this paper, we will address our experience in building this kind of remote visualization platform. The current experimental environment (see Figure 1) includes a few local Silicon Graphics workstations and many workstations with X terminals connected by 10 Mbytes/sec Ethernet. Remote access to the Intel Delta Touchstone machine at Caltech is provided through a 56Kbytes/sec data transfer line. Finally, to demonstrate the feasibility of our work, we present a visualization of a molecular dynamics simulation on our graphics platform.

2 Overview of Our Remote Visualization Platform

At Washington State University (WSU), we have active research in areas such as molecular dynamics, electromagnetic wave scattering, and fluid dynamics. There is increasing need to provide a visualization environment for researchers to use in analyizing their data. Through our collaboration with Pacific Northwest Laboratory (PNL), we have access to the Intel Delta parallel computer at Caltech. Figure 2 shows the main visualization cycle in our platform. There are four main processes.

determined by the memory size of the system and the size of datasets. As in [12], a software-implemented cache with the least recently used (LRU) strategy is devised to exploit ray coherence. Initially, each group of processors holds a copy of the whole bound volume hierarchical tree. Within each group, the top n level of tree are duplicated on each processor and the remaining subtree nodes are evenly distributed among the processors. During raytracing, if any subtree is missed locally, it is fetched from a specific processor at the same group. This subtree is then stored and maintained in local cache memory. For detailed discussion, see [13].

4 Input/Output (I/O) Tasks

Most highly parallel computers have remarkably inadequate I/O and graphics capabilities, particularly for I/O intensive tasks such as graphics, which needs massive numbers of read/write operations on disk and data communications on the network. For efficient parallel rendering in many implementations, a parallel machine can be connected to a distributed frame buffer(DFB). For example, the AT&T Pixel machine, has multiple banks of video memory (VRAM), that can store and access the image data in an interleaved mode to display an image efficiently.

In our platform, we need to display an image on the remote site. Therefore, we do not adopt the above configuration. Simply, we output the result into a single file and then transmit the image to the remote site for display. In our implementation, all processors scatter their rendered results between different files. At the end of the computation one node will gather these results and reorganize them into a single file. After the scatter/gather process, a JPEG library is used to reduce the amount of data transfer. The JPEG compression/decompression technique, the ISO standard for still, high-quality images, is based on the 8*8 (two-dimensional) Discrete Cosine Transform (DCT), followed by psychovisual quantization and statistical coding. We can adjust the compression ratio to about 1/10 to 1/20 without too much loss of the original image data.

Network communication on the Delta and other workstations uses the Unix socket network interface [9]. Both client and server open a socket interface and connect to each other through the server's Internet address and a service port. In our environment, the server site is a Sun host located at Caltech which is in charge of file transfer between the Delta and local WSU sites. The Delta site could have near real-time data transfer to the local server host. This host is responsible for communicating with the other machines. Therefore, each node of the Delta can concentrate on its computationally intensive jobs and not suffer too much communication overhead while communicating with the other machines.

Processor(s)	Rendering timings (sec.)	Efficiency
1	1329.53	100%
16	83.23	99.8%
32	41.77	99.4%
64	20.94	99.2%
128	10.52	98.7%
256	5.46	95.1%
512	3.12	83.2 %

Table 1: Experimental results for raytracing zeolite structure.

An important issue for us in image display and animation is making the best use of the local display hardware without increasing cost. In our campus departments, there are many color X-terminals and some advanced Silicon Graphics workstations. In order to make our image output available for each potential user, we provide many image formats, such as JPEG and PPM. These formats can be viewed using many popular software packages, such as the "XV" viewer. For simple animations, we use the "MOVIE" utility in SGI to show a sequence of images.

5 Results of Molecular Dynamics Visualization

Molecular dynamics (MD) simulation provides a powerful method for investigating the microscopic and macroscopic properties of many chemical systems. Researchers have used MD to study gases and liquids, polymers, biological polymers such as proteins, and solids [14]. MD simulations require the computation of forces between the atoms or particles that make up the simulation system and integration of the equations of motion to propogate the system through time. The simulation computation is done in parallel on the Delta. We are interested in simulating system that contain 1000's to 100,000's of particles. In this paper, we will present a visualization of a zeolite. Zeolites are a large inorganic crystals and are vital to the chemical and petroleum industries.

We show the result of visualizing a zeolite structure composed of silicon and oxygen atoms. We use a ball-stick model to represent the geometry. The atoms are visualized by spheres of different color and size. The bonds between atoms are represented by cylinders. Figure 3 shows the raytraced image of the zeolite structure. The CPU times and parallel efficiencies for visualizing the zeolite using our raytracer are given in table 1. The results show that our parallel raytracer can achieve almost linear speedup performance.

Ideally, the compressed image data could be transferred over the internet in real time speed between WSU and Caltech. However, performance is degraded

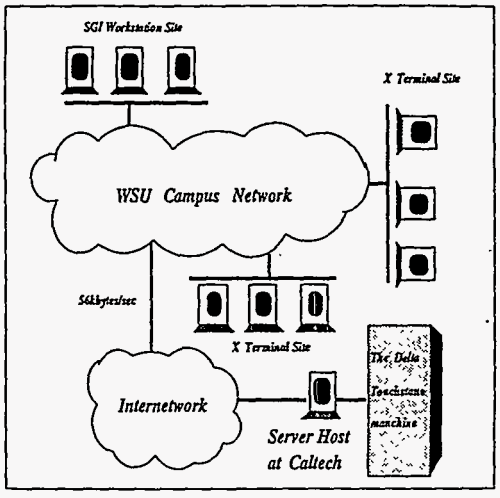


Figure 1: Remote Visualization Platform

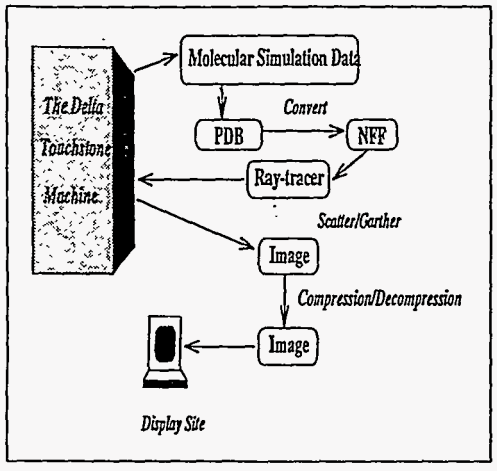


Figure 2: Visualization Cycle

- *Data simulations and data acquisitions*: Researchers submit their simulation jobs on the Intel Delta parallel computer and then obtain results in a standard output format such as the Protein Data Bank (PDB) file format. In our implementation, the PDB file is converted to a standard graphics scene format, NFF(Neutral File Format) [7].

- *Image rendering*: With an NFF file, an efficient, parallel render implemented on the Delta Touchstone machine is responsible for generating high quality of images.

- *Data communication and data compression/decompression*: The bandwidth of the internet in our environment cannot offer real-time image transfer. Thus, we exploit image data compression/decompression techniques, such as JPEG [10], to reduce the amount of data transferred between WSU and Caltech.

- *Image display and animation*: Within the limits of advanced graphics hardware, two options are provided: display of 24bit/pixel images on SGI workstations and 8 bit/pixel on X terminals. A simple animation can be done on the SGI workstations.

3 Parallel Rendering on the Delta Touchstone

In the past, there have been many efforts to build parallel renders on parallel machines [8]. Depending on how data parallelism is exploited, most of these approaches are classified into image space and object space methods. The main design issues are the schemes used for load balancing and data distribution. Raytracing the geometric surface data and raycasting volume data are popular techniques in visualization. In principle, they are very much alike. In our previous work [11], we designed and compared different image space load balancing strategies. The experimental results showed that no matter what the rendering situation was our method would always be able to provide very good performance. Here, we outline the load balancing scheme presented in [11].

N processors are logically organized in a ring topology. The screen is divided into a number of $a \times a$ square regions and these regions are assigned among the processors in an interleaved fashion. A processor will first raytrace its assigned regions. When a particular node, P_i becomes idle, it will request extra work from successive nodes on the ring P_{i+1}, ..., P_N, P_1,...,P_{i-1}, until it finds a node that is active. This active node, P_j, dispatches an $a \times a$ region to this request. The node will remember that node P_j was the last node from which a request was made. The next time it is idle, it will start requesting work from node P_j, bypassing nodes between P_i and P_j. An additional feature of this strategy is that if in the meantime, node P_j becomes idle and remembers that it received work from node P_k, node P_i will jump from P_j directly to P_k without asking for work from nodes between P_j and P_k. In this strategy, node P_i stops its search for work if it ends up at itself or at some node that it has already visited or skipped in a search step. This ensures that the search will end when all nodes are idle. An $a \times a$ square region is used to control the granularity of workload. In our experience, we control this size in the range of 2×2 to 5×5 to obtain good results.

With the increase in the size of datasets, we can not afford to duplicate the data in a single processor. We need an efficient method for partitioning and redistributing data. In our implementation, a shared virtual memory is designed to hold many copies of datasets in the whole system. The replication ratio is

[12] Green and et. al, "A Highly Flexible Multiprocessor Solution for Ray Tracing," In the Visual Computer 6,2(March 1990), 67-73.

[13] Tong-Yee Lee and et. al, "Load Balancing of Parallel Raytracing Algorithm with Hierarchical Tree Decomposition," submitted to HICSS-28, Jan. 3-6, 1995.

[14] "Computer Modeling of Fluid, Polymers, and Solids," book by C.R.A. Catlow, S. C. Parker, and M.P. Allen. Kluwer Academic Publishers, 1990.

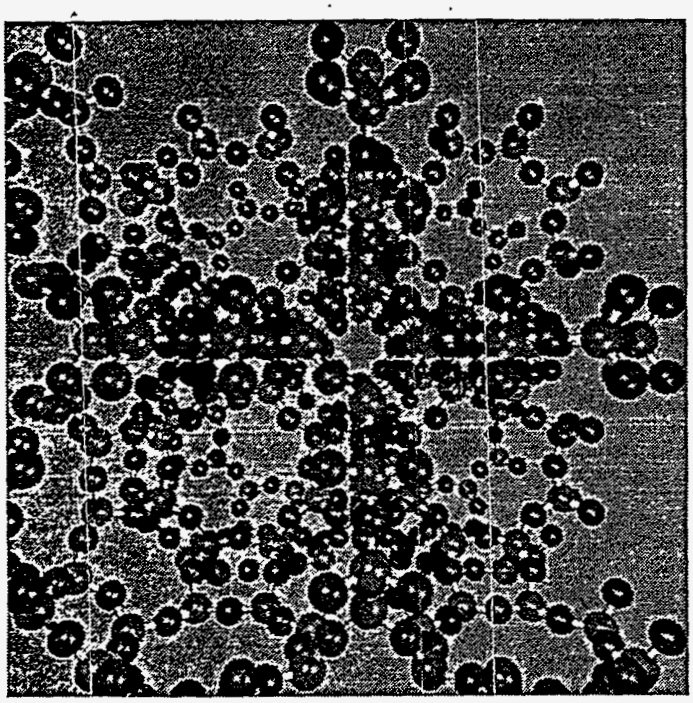


Figure 3: The zeolite structure

by the current heavy load on the internet. In our experiment, we found that it takes about 1 - 10 seconds for data transfer. This I/O problem is the main bottleneck in our implementation. In the future when a higher speed network is available, real time data transfer should be possible.

6 Conclusion and future works

In this paper, we present a platform for the remote visualization of scientific data. We attempt to make the best use of our available software and hardware. We demonstrate the feasibility of our method on a simple visualization of an MD simulation. Our parallel raytracer can achieve near linear speedup and provides a very high quality image. However, we cannot yet achieve real time rendering due to data transfer bottlenecks. Thus, we are investigating other techniques that can speed data visualization. For example, we are interested in other methods of visualizing scientific data, such as volume and polygon rendering. These methods can provide additional insight into scientific data with reasonable rendering rates. We are also working on a parallel data compression/decompression technique; the performance of our current visualization platform is degraded by our single node JPEG implementation. When both visualization and data compression can be done in real time, network I/O will be the main bottleneck in a remote visualization system. Hopefully, with the advent of a Gigabyte network, a real-time remote visualization system will be achieved.

Acknowledgements

This research is supported in part by the NSF Grant No. MIP-9296043. This research was performed in part using the Intel Touchstone Delta System operated by Caltech on behalf of the Concurrent Supercomputing Consortium. Access to this facility was provided by Pacific Northwest Laboratory, a multiprogram national laboratory operated for the U.S. Dept. of Energy by Battelle Memorial Institute under contract DE-AC06-76RLO 1830. We also give special thanks to Mark VandeWettering and Independent JPEG Group for providing invaluable sequential MTV raytracer and JPEG source code.

References

[1] Computer, A Special Issue on Scientific Visualization, August 1989.

[2] Dyer, D.S., "A Dataflow Toolkit for Visualization," IEEE Computer Graphics and Applications, Volume 10, Number 4, July 1990.

[3] Khoros Vision Lab, "Visual Programming System and Software Development Environment for Data Processing and Visualization," Khoros User's Manual, University of New Mexico, Version 1.0, 1990.

[4] T. Todd Elvins and David R. Nadeau, "NetV: An Experimental Network-based Volume Visualization System," IEEE Proceedings of Visualization , 1991.

[5] Tong-Yee Lee, "Visualization of Ocean Wave on DVP environment," Master thesis, National Taiwan University, 1990.

[6] Shih Stephen,"Distributed Thermal Fluid Dynamics Simulations," project report of EE/CS Dept. at Washington State University, 1992.

[7] E. Haines, "A Proposal for Standard Graphics Environments," IEEE Computer Graphics and Applications, July, 1987.

[8] 1993 Parallel Rendering Symposium Proceedings, IEEE and ACM press, San Jose, CA. October 25-26, 1993.

[9] W. Richard Stevens, "UNIX Network Programming," Prentice Hall, 1991.

[10] Gregory K. Wallace, "The JPEG Still Picture Compression Standard," IEEE Transactions on Consumer Electronics, December, 1991.

[11] Tong-Yee Lee and et. al, "Experimental Evaluation of Load Balancing Strategies for Ray Tracing on Parallel Processors," ICPP'94 International Conference on Parallel Processing, Aug. 15-19, 1994, Illinois.