Performance Modeling of Interactive, Immersive Virtual Environments for Finite Element Simulations

Valerie E. Taylor
EECS Department
Northwestern University

Milana Huang
Electronic Visualization Laboratory
University of Illinois at Chicago

Thomas Canfield *Rick Stevens*
Mathematics and Computer Science Division
Argonne National Laboratory

Daniel Reed *Stephen Lamm*
Computer Science Department
University of Illinois at Urbana-Champaign

To Appear in the *International Journal on Supercomputer Applications and High Performance Computing*

Performance of Virtual Environments

Contact: Valerie E. Taylor
Assistant Professor
EECS Department
Northwestern University
2145 Sheridan Road
Evanston, IL 60208-3118
voice: (847) 467-1168
FAX: (847) 467-4144
taylor@eecs.nwu.edu

Abstract

Interactive, immersive virtual environments allow observers to move freely about computer generated 3D objects and to explore new environments. The effectiveness of these environments is dependent upon the graphics used to model reality and the end-to-end lag time (i.e., the delay between a user's action and the display of the result of that action). In this paper we focus on the latter issue, which has been found to be equally important as frame rate for interactive displays. In particular, we analyze the components of lag time resulting from executing a finite element simulation on a multiprocessor system located in Argonne, Illinois connected via ATM to the interactive visualization display located in San Diego, California. The primary application involves the analysis of a disk brake system that was demonstrated at the Supercomputing 1995 conference as part of the Information Wide Area Year (IWAY) project, which entailed the interconnection of various supercomputing centers via a high-bandwidth, limited-access ATM network. The results of the study indicate that the major components of the end-to-end lag are simulation, synchronization, and rendering times; the use of the ATM network resulted in the network time comprising only a small fraction of the end-to-end lag time.

1 Introduction

Interactive, immersive visualization allows observers to move freely about computer-generated three-dimensional objects and to explore new environments. This technology can be used to extend our perception and understanding of the real world by enabling observation of events that take place in spaces that are remote, protracted or dilated in time, hazardous, or too small or large to view intricate details. The resulting three-dimensional virtual environment can be a distortion of reality projected onto a physical framework that enables the display of non-visual, physical information, such as temperature, velocity, electric and magnetic fields, and stresses and strains.

Virtual reality strives to be a more natural user interface to complex data, allowing the scientist to focus on the analysis of the data rather than manipulation of the analysis environment [Bishop et al., 1992]. The human senses are more accustomed to the three-dimensional world; the ability to process three-dimensional spatial information has been honed over many years of evolution [Kalawsky, 1993]. In engineering, this technology may be incorporated into the product design cycle, allowing virtual prototyping and testing of products prior to their physical construction. Hence, interactive, immersive three-dimensional visualization is an important medium for scientific applications.

Interactive, immersive visualization of scientific simulations involves four major subsystems: graphics, display, simulation, and communications. The descriptions of these subsystems particular to our system are as follows. The graphics subsystem performs the calculations needed to render the objects used in the display. The display subsystem consists of the projection screen, projectors, interactive devices, and tracking sensors needed to physically realize the virtual environment; the user interacts with the virtual objects via such display subsystem devices as a head tracker or hand-held wand (similar to a mouse). The simulation subsystem performs the calculations for the interactive analysis of the scientific phenomenon. Because phenomena of interest often require complex models to capture dynamic interactions, parallel computation usually is needed to reduce the execution time. The last subsystem consists of the connections used to communicate information between the user (via the display) and the graphics system and between the graphics and simulation systems.

A critical issue facing interactive virtual environments is the end-to-end lag time (i.e.,

the delay between a user action and the display of the result of that action.) Like any closed loop adaptive system, if the lag is too great, users find it difficult to maintain fine control over system behavior and complain that the system is unresponsive. Indeed, Liu *et. al.* [Liu et al., 1993] found lag time to be as important as the frame rate for effective use of immersive displays.

Lag has been studied in the context of teleoperated machines, head-mounted displays, and telepresence systems [Liu et al., 1993, Wloka, 1995]. Liu *et al.* [Liu et al., 1993] conducted experiments on a telemanipulation system and found the allowable lag time to be 100 ms (0.1s) and 1000 ms (1s) for inexperienced and experienced users, respectively. In [Taylor et al., 1995], the work on lag models was extended to include scientific simulations with interactive, immersive environments. That work, however, focused on simulations executed on one processor; multiple processors for the simulations were not used or analyzed in the lag model. Further, all of the computer components were located at one site, interconnected via a local area network. The goal of this paper is to analyze the components of lag resulting from simulations executed on multiprocessors connected to the virtual environment via wide area networks. With the emergence of high-speed networks and distributed computing resources, the frequency of remote access and distributed collaboration is rising rapidly.

We have conducted an extensive case study of a visualization system to display the results of a finite element simulation of a contact-impact problem, in particular a disk brake system. This application was demonstrated at the Supercomputing 1995 conference as part of the Global Information Infrastructure Testbed for the IWAY (Information Wide Area Year) project. The display system consisted of an ImmersaDesk^{TM} — a single wall of a CAVE^{TM} (Cave Automatic Virtual Environment) [Cruz-Neira et al., 1993]. The simulation was executed on the IBM SP located at Argonne National Laboratory; the display system was located at the San Diego Convention Center. The IBM SP was connected to the display system via an ATM OC-3c network as part of the IWAY project. Although our analysis is specific to this context, the concepts presented in this paper can be extended easily to other scientific applications.

To understand the contributors to lag, we instrumented all major processes in the system and constructed a performance model of the contributors to end-to-end lag: rendering, tracking, local network connections to the parallel system, parallel simulation, and various

types of synchronization lags. Our lag model decouples the viewpoint lag (not involving the simulation) from the interaction lag (using the simulation results). Our analysis indicates that the major component of viewpoint lag is the rendering time. For interaction lag, the majority of the time is comprised of simulation and synchronization times.

The remainder of the paper is organized as follows. In §2, we discuss previous work, followed by an outline of the supercomputer visualization environment in §3. The general model for end-to-end lag is given in §4, followed by the case study's findings in §5. This is followed by a discussion of methods for reducing the lag in §6. Finally, we summarize our results in §7.

2 Previous Work

In [Wloka, 1995], Wloka presents a thorough analysis of lag time in a multiprocessor virtual reality system; the multiprocessors execute the calculations necessary to render an image. The focus is on viewpoint lag. He identifies the various sources of lag time: *input device lag* — the time required to obtain position and angle measurements for the input device, *application lag* — application-specific processing of the input device mechanism, *rendering lag* — the time to render and display the data, *synchronization lag* — the average time the sample is waiting between processing stages, and *frame-rate induced lag* — the time between changes in the display. In Wloka's system, the application-specific processing is directly dependent on one user input device. In contrast, we analyze an existing system with two input devices: a head tracker (which affects the viewpoint lag) and a wand (which affects the viewpoint and interaction lags). Methods for reducing the lag in our system must consider the relationship between the two lags. Further, our system includes a parallel system for simulation and a shared-memory multiprocessor for rendering, interconnected via a wide area network. Therefore, we consider two additional sources of lag: the network lag and simulation lag.

In [Mine, 1993], Mine characterizes the relative performance of magnetic tracking technologies, including two magnetic trackers from Ascension Technology and two from Polhemus. This characterization focused on reducing end-to-end delay in head-mounted systems, with emphasis on tracking lag; no attention is given to other sources of lag. In contrast, we consider all the sources of lag.

Methods for reducing lag are an active area of research. Such methods include prediction [Deering, 1992, Friedmann et al., 1992, Liang et al., 1991], time-critical computing [Funkhouser et al., 1993, Holloway, 1991, Wloka, 1993], and use of parallelism. Prediction methods use extrapolation to reduce tracker lag by predicting future input data based upon past data. These methods assume the other components of lag have constant times. This is generally not the case, especially for systems that include scientific simulations on supercomputers. Time-critical computing trades computation time for computation accuracy, which is not advisable for directly reducing lag. Parallelism reduces the lag by increasing the computing resources used for the computation. Our system uses exploits the parallelism in the simulation and rendering calculation.

3 Visualization and Simulation Environment

Any study of virtual environment overheads presumes some basis for experimentation. Our environment for interactive, immersive visualization and simulation consisted of a 128-node IBM SP system located at Argonne National Laboratory and a ImmersaDesk^{TM} located at the San Diego Convention Center. Figure 1 shows the interconnection of the hardware and software components, which are described in detail below.

3.1 Display Component

The ImmersaDesk^{TM}, the display component, creates a wide field of view by rear-projecting stereo images on a 4x5 foot translucent panel tilted at a 45 degree angle; see Figure 2. This display system is a lower cost, more portable, and smaller alternative to the CAVE^{TM}, a room composed of rear-projection screens [Cruz-Neira et al., 1993]. The ImmersaDesk^{TM} provides the illusion of data immersion via visual cues, including wide field of view, stereo display, and viewer-centered perspective. A Silicon Graphics (SGI) Power Onyx computes the stereo display, with a resolution of 1024×768 for each image. The Onyx alternately draws left and right eye images at 96 Hz, resulting in a rate of 48 frames per second per eye. These images are sent to an Electrohome video projector for display. Infrared emitters are coupled with the projectors to provide a stereo synchronization signal for a CrystalEyes LCD glasses worn by each user. These glasses have a sampling rate of 96 Hz that is matched

to the projection system; the eyes and brain fuse the alternate left and right eye images to provide stereo views.

Tracking is provided by an Ascension SpacePad unit with two inputs. One sensor tracks head movements; the other tracks a hand-held wand. These sensors capture position and orientation information on head and wand movements at rate of 10 to 144 measurements per second [SpacePad, 1995]. The existing system is configured to operate in the range of 100 measurements per second. The buttons on the wand are sampled at a rate of 100 Hz.

The location and orientation of the head sensor are used by the SGI Onyx to render images based on the viewer's location in the virtual world. Hence, subtle head movements result in slightly different views of the virtual objects, consistent with what occurs in reality. The wand has three buttons and a joystick that are connected to the Onyx via an IBM PC, which provides A/D conversion, debounce, and calibration. Other observers can passively share the virtual reality experience by wearing LCD glasses that are not tracked.

3.2 Graphics Component

The SGI Onyx is a shared-memory multiprocessor with a high-performance graphics subsystem. The system used in our experiments had 128 MB of memory, 10 GB of disk, four R4400 processors and three RealityEngine2 graphics pipelines. Each RealityEngine2 has a geometry engine consisting of Intel i860 microprocessors, a display generator, and 4 MB raster memory [SGI]. The Onyx is used to drive the virtual environment interface as discussed above. The ImmersaDesk(TM), however, uses only one RealityEngine2 graphics pipeline connected to a Electrohome Marque 8000 high-resolution project to project images onto the one translucent screen.

The visualization code executed on the SGI Onyx consists of three processes, main, rendering, and tracking, that manage, respectively, communication with the parallel simulation, calculations for surface graphics, and management of interactive commands. The code is explained further in §4.

3.3 Simulation Component

The simulation component consists of a 128-processor IBM SP with a high-performance input/output system. Each SP node has 128 MB of memory and a 1 GB local disk and was connected to other SP nodes via a high-speed network. Some of the nodes were equipped with ATM and Ethernet interfaces. Collectively, the SP system is also connected to 220 GB of high-speed disk arrays and an Ampex DST-800 automated tape library.[1]

3.4 Interconnections

The interconnections used in the system consists of the devices used by a scientist to interact with the display system and the interconnection of the simulation and graphics components. The scientist controls the field of view by changing his or her head position or manipulating the wand buttons and joystick; the simulation is also modified via the wand. The simulation and graphics components are interconnected via the ATM OC-3c (peak bandwidth of 155 Mbps) used with the IWAY project.

4 Performance Model

The performance model presented in this section is consistent with that given in [Taylor et al., 1995]. It is presented here for completeness. Given two input devices, there are two classifications of interactions:

- *movement of the head tracker*: this type of interaction causes a change to the field of view; the data sent to the simulation process is not modified — this lag is defined as Q_{view}
- *movement and clicking of wand buttons*: this type of interaction can change the field of view or the simulation process, dictated by the code-defined wand buttons and joystick — this lag is defined as $Q_{interact}$. In this paper, the focus is on wand modification to the simulation process to distinguish the two lags.

[1] This input/output system eventually will be used for recording and playback of visualizations. However, that work is beyond the scope of this paper.

The operations that are executed based upon a user interaction are the following:

1. The sensors generate the position and rotation of the header tracker and wand; the personal computer records the position of the wand buttons (T_{track}) [input device lag]
2. The wand information is read by simulation side (T_{read}) [network lag]
3. The scientific application is simulated on the multiprocessor system (T_{sim}) [simulation lag]
4. The update from the simulation is sent to the graphics process (T_{write}) [network lag]
5. The graphics process uses the data from the simulation process and the tracker process to render a new image (T_{render}) [rendering lag]

In addition to the above lags there is also synchronization lag as described previously. We consider four sources of synchronization lag: (1) $T_{sync(TR)}$: the time from when the tracker measurement is available until the data is read by the rendering process, (2) $T_{sync(RS)}$: the time from when the rendering process has read the updated wand values until the values used by the simulation process, (3) $T_{sync(SR)}$: the time from when the data is available from the simulation process until written to the rendering process, and (4) $T_{sync(F)}$ the time from when the data is available in the frame buffer and the image is available on the screen.

Given the above sequence of operations, the following equations represent the lag time for the head tracker (Q_{view}) and the wand ($Q_{interact}$):

$$Q_{view} = T_{track} + T_{sync(TR)} + T_{render} + T_{sync(F)} \quad (1)$$

$$Q_{interact} = T_{track} + T_{sync(TR)} + T_{sync(RS)} + T_{read} + T_{sim} + T_{write} + T_{sync(SR)} + T_{render} + T_{sync(F)} \quad (2)$$

The derivation of these equations is discussed in the following section.

4.1 Lag Sources

The components of Q_{view} (or Viewpoint Lag) and $Q_{interact}$ (or Wand/Simulation Interaction lag) are depicted pictorially in Figure 3. All of the processes run asynchronously. Recall from §3 that the graphics components consists of three processes: main, rendering and

tracking; the simulation consists of one process. The main process runs on the Onyx and is responsible for forking the rendering and tracking processes (this is done only once) and communicating with the simulation process. Hence this process has three states: writing data via the network to the simulation process, reading data from the simulation process, and copying the new data to the SGI shared memory to be used by the rendering process. The one rendering process, Render0, reads the tracking data from the tracking process uses this data to render the left and right eye images.

The tracker process is responsible for continuously reading the tracking information from the serial SGI ports, scaling the data, and writing the data into a region of memory for reference by Render0. The tracker process is also responsible for initialization of the tracker and wand controls. Like the other processes, the tracker process operates asynchronously, reading tracker data as fast as the tracking system can produce it.

For the case of the CAVE(TM) display system, images can be displayed on four screens — the three surrounding walls and the floor. For this case, there will be four rendering processes forked by the main process and only rendering process Render0 will obtain the tracking data. This is done to insure that all four rendering processes are performing calculations in response to the same tracker data. All four rendering processes synchronize at the beginning of each frame.

The scientific simulation executed on the IBM SP system consists of three states: executing the simulation, sending data to the main process, and receiving data from the main process. Hence the simulation processes are interrelated to the graphics processes. The effects of these processors on the view and interaction lags are given below.

4.2 Lag Equations

The diagram in Figure 3 illustrates all the sources of lag that are used in the performance model. Assuming a wand and tracker event occurs as indicated in the diagram, we can trace the lag times that result in a scene update due to a head event (indicated in the diagram) and a scene update due to a wand event (indicated in the diagram). Head tracker events are distinguished from wand events in the header segment of the data sent to the tracker process.

When a head tracker event occurs, the tracker process reads the values from the SpacePad

ports, performs the calibration, and places these values in shared memory. The time to execute these operations is given by T_{track} in Equations (1) and (2). Typically, the tracker process is sampling the sensors faster than the rendering process can render a new display. Only the last sample obtained prior to the start of a new rendering cycle is read by the rendering process. Hence, the average "wait time" or synchronization lag is half the average tracker update time. This time corresponds to $T_{sync(TR)}$ in the equations. The tracker sample is used by Render0 to generate a new image, corresponding to T_{render} in the equation. When the new image data is available, it may not be displayed immediately. There is some wait time due to the frame rate and the scan rate of the projectors. The average of this synchronization time is half the frame and scan times per eye for stereo; this time is given by $T_{sync(F)}$ in the equations. The scene is now updated in response to the head tracker event. The summation of these four terms compose the viewpoint lag or Q_{view}.

When a wand tracker event occurs, the sensors are again sampled by the tracker process and read by Render0 process. This task corresponds to $T_{tracker}$ and $T_{sync(TR)}$ as described above. At this point, the analysis takes a different path from that taken with the viewpoint lag. Once the wand position has been read by Render0 process, it is used by the main process to forward to the simulation process. This wand data may not be read immediately by the simulation process. The average time that this data "waits" to be used is equal to half the time of the simulation process. This synchronization time corresponds to $T_{sync(RS)}$ in Equation (2). The actual communication of these values to the simulation side is given by T_{read}. The simulation time is denoted by the term T_{sim}. The updated simulation values are then sent to the rendering process, denoted by the term T_{write}. After the data has been sent, it may not be used immediately by the rendering processes. The average of this "wait" time is equal to half the average of the rendering time; this synchronization time is denoted as $T_{sync(SR)}$. Once the values have been read by the rendering processes, a new image is rendered and displayed corresponding to $T_{render} + T_{sync(F)}$. The summation of these nine terms compose the interaction lag or $Q_{interact}$.

5 Case Study: Analysis of a Disk Brake System

To assess the performance of the combined supercomputer/visualization system described in §3, we analyzed a finite element simulation of an automotive disk brake system. Braking

times generally are on the order of seconds or tens of seconds. Each time step, which is generally on the order of a few milliseconds, can require tens of minutes of execution time on a single processor. Hence, parallel systems are necessary for this application. Some critical issues related to automotive disk brakes are the heating and stressing of the material used for the disk and pads, the wearing on this material, and the pitch of the sound that occurs when the pads are applied to a rotating disk. The virtual environment interface to the analysis consists of a disk brake in a Porsche; see Figure 4. Using the virtual environment's wand, a scientist can manipulate the virtual environment to focus on various features (for example areas of high stress or high temperature) of the brake system. Further, while viewing the features, the scientist can use the wand to modify the simulation to test different conditions.

Initial transients, such as closing the brake pads, take place in a few milliseconds, whereas the actual braking occurs over the longer time when the pads are clamped on the rotor. Stable-implicit time integration is required to model this long term braking. The disk brake system is modeled as a finite element problem using the FIFEA (Fast Implicit Finite Element Analysis) code developed at Argonne National Laboratory. This code performs dynamic analysis of solid structures using implicit finite element methodology. FIFEA employs a pseudo-rigid body formulation to decouple the large displacements and rotations due to rigid body motion from the small relative displacements and strains associated with elastic deformation and thermal stress. FIFEA can detect intermittent contact and calculate friction forces and heat generated at the contacts. The complete disk brake system consists of 3790 elements, 5636 nodes, and 22,544 degrees of freedom (4 degrees of freedom per node). Figure 5 is the finite element mesh of the disk brake. Given the problem size, the finite element simulation was executed on 8 processors of the IBM SP. Figure 6 shows an example of the temperature distribution of a rotating disk as modeled by FIFEA.

FIFEA makes extensive use of the PETSc (Portable, Extensible, Toolkit for Scientific computation) [Balay et al., 1995] to do linear algebra and to manipulate sparse matrices and vectors. FIFEA and PETSc use the MPI library [Gropp et al., 1994] for communication between the IBM SP processors. One IBM SP processors sends and receives results to and from the SGI Onyx using the CAVEcomm library [Disz et al., 1995].

5.1 Performance Instrumentation

To understand the temporal interaction patterns among the graphics software libraries, the SGI Onyx, and the finite element software on the IBM SP, we instrumented the graphics library and finite element software using the Pablo performance analysis environment [Reed et al., 1995, Reed et al., 1996]. The Pablo environment consists of (a) an extensible performance data metaformat and associated library that separates the structure of performance data records from their semantics, allowing easy addition of new performance data types, (b) an instrumenting parser capable of generating instrumented SPMD source code, (c) extensible instrumentation libraries that can capture event traces, counts, or interval times and reduce the captured performance data on the fly, (d) graphical performance data display and sonification tools, based coarse-grain graphical programming, that support rapid prototyping of performance analyses, and (e) a virtual environment for real-time display of dynamic performance data.

Using the Pablo environment, we instrumented the graphics library to capture tracker updates, drawing of left and right eye display frames, and internal library lock and buffer management. In addition, we instrumented the finite element code to capture communication between the SP and SGI Onxy using the CAVEcomm library and entry/exit to selected portions of the application code. Together, this instrumentation allowed us to quantify the relative contributions of tracking, rendering, remote communication, and application computation to lag.

5.2 Timing Data

Table 1 is the average time for each lag source for the simulation executed on 8 IBM SP processors with IWAY interconnection between Argonne and the San Diego Convention Center. The network values in parentheses correspond to the values collected using the Internet connection between the IBM SP at Argonne and the SGI Onyx at University of Illinois at Chicago. The mean lags and their standard deviations are based on 30 minutes of elapsed time. The values with no corresponding standard deviations correspond to the sources of synchronization lag, which are derived from other values. The $T_{sync(F)}$ value is based on a frame rate of 48 frames per second per eye and an average scan rate of 120 Hz for the Electrohome projectors. The average of this synchronization lag is equal to one half

of the frame-induced time.

The total network time, the summation of T_{read} and T_{write}, corresponds to the time to send the temperature data from the IBM SP to the SGI Onxy, modify the data structures on the SGI Onyx, and send the wand data back to the IBM SP. In particular, the time to send the simulation data from the IBM SP to the buffers is equal to T_{write} and the time to receive the simulation data on the SGI Onxy, process it, send the wand data, and receive the wand data on the IBM SP side is equal to T_{read}. For the IWAY, the total time is 822.26 ms; for the Internet the total time is 1360.57. This reduction is significant given that the data using IWAY went from Illinois to California and through the myriad connections in the San Diego Convention center as compared to going between two sites in the same state with the Internet.

Recall from §1 that Liu *et al.* [Liu et al., 1993] conducted experiments on a telemanipulation system and found the allowable lag time was 100 ms and 1000 ms for inexperienced and experienced users, respectively. The total lag times for the 8 processor case are $Q_{view} = 129.58$ ms and $Q_{interact} = 57370.05$ ms. The view lag is within the allowable tolerance, but the interaction lag is over one order of magnitude beyond the tolerance for experienced users and two orders of magnitude for the inexperienced users. As expected, 81 percent of the view lag is due to the rendering component. For the interaction lag, the simulation and synchronization times compose over 95Methods for reducing these times are discussed in the following section. The methods used to reduce the simulation time will also result in a reduction of the synchronization time, due specifically to a reduction to $T_{sync(RS)}$ (recall that $T_{sync(RS)}$ is approximated as half T_{sim}). Hence methods to reduce the simulation time will have a significant impact on reducing the interaction lag.

6 Lag-Reducing Methods

In this section we consider methods for reducing the end-to-end lag and highlight areas of future research. We focus on the rendering, simulation and synchronization lags, which can be major factors affecting the end-to-end lag as discussed in the previous section. In particular we focus on scene complexity, parallelism, and phase tuning.

6.1 Scene Complexity

The rendering lag is a function of the scene complexity and the geometry transformations. In general a scene consists of essential objects affected by the simulation and the background used to give the scientist the illusion of being in the remote environment. For the automotive disk brake application, the scene consists of a shaded or wire frame version of the Porsche, the rotating tires, and the disk with two pads. The image of the car and tires provides the context for the disk brake, conveying position and relative size of the brakes. These two images, the car and the tires, are not necessary for the analysis of the brake system. The elimination of these images can reduce the rendering time, thereby reducing the view lag, without sacrificing the interface to the simulation. Further research is necessary to develop a performance model of the rendering time based upon the essential and nonessential objects in a scene. This model can be used to determine the scene to be used for a given application based upon viewpoint lag constraints.

6.2 Parallelism

The simulation time was the major component of the interaction lag for the automotive disk brake system. This lag can be reduced by the use of more processor nodes. The number of processors nodes to be used for the problem is dictated by the interprocessor communication requirements. Further research is necessary to identify the communication requirements dictated by the algorithm used in the FIFEA code and the domain decomposition scheme and investigate methods for reducing these requirements.

With the IWAY project, various supercomputer centers are interconnected by ATM as described previously. This project has made possible the use of a large number of processors from the aggregation of the machines at the different sites. Users are no longer limited by the number of processors at any given site. Further research is also necessary develop decomposition schemes for a network of supercomputers, which involves communication costs of the local network as well as the wide area network.

6.3 Phase Tuning

The synchronization lags were another major component of the interaction lag. In particular, the synchronization lag between the rendering and simulation, $T_{sync(RS)}$, dominated the synchronization lag. Recall that this lag corresponds to the interval from when the wand data has been read by the rendering process until it is used by the simulation process. Given that the wand updates from the scientist can occur at any time, the average value of $T_{sync(RS)}$ is equal to half the simulation time. One possible method for reducing this lag is to "phase tune" the various asynchronous processes. This may involve using predictors, based upon previous interactive inputs, to be used by the simulation process to estimate the wand inputs at the optimal time to reduce the synchronization lag. Another option involves adding a delay mechanism to the various asynchronous processes that can be tuned to reduce the phase miss match, thereby reducing the synchronization lags. Further research is necessary to explore these options and identify effective methods for reducing the synchronization lags.

7 Summary

In this paper we analyzed the lag time of a integrated supercomputer applications with interactive, immersive virtual interfaces. We conducted an extensive case study of a system used to display the results of a finite element simulation of the analysis of an automotive disk brake system. The simulation was executed on the IBM SP at Argonne National Laboratory and the virtual interface was available at the San Diego Convention Center. The simulation and virtual system were interconnected via ATM in conjunction with the IWAY project. The analysis entailed the comparison of a system using ATM versus Internet. The results of the study indicated that the view lag was within the allowable tolerance, but the interaction lag was over one order of magnitude beyond the tolerance for experienced users and two orders of magnitude for the inexperienced users. As expected, 81 percent of the view lag was due to the rendering component. For the interaction lag, the simulation and synchronization times composed over 95the transmission time using an ATM network between Illinois and California was approximately half of that using the Internet between two sites in Illinois.

The results of the study highlighted the importance of reducing the lag time to a tolerable range — 1 s to 0.1 s. Various methods to reduce the end-to-end lag for a supercomput-

ing/virtual system were also discussed. We considered scene complexity, parallel processing, and phase tuning. Future work entails quantifying the impact on end-to-end lag for the aforementioned methods.

Acknowledgments

The authors thank Micheal Papka, Terry Disz, Warren Smith, Jonathan Geisler, Lois Curfman, and Steve Tuecke for their long hours of help in making the various libraries work in harmony. The authors also acknowledge the IWAY leaders, Tom Defanti and Rick Stevens, for making this project possible and Jason Leigh for assisting with the development of the ImmersaDesk images.

The author at Northwestern University was supported by a National Science Foundation Young Investigator Award, under grant CCR–9357781. The authors at Argonne National Laboratory were supported by the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38. The author at the University of Illinois in Chicago was supported by a Laboratory Graduate Grant from Argonne National Laboratory. Finally, the authors at the University of Illinois in Urbana-Champaign were supported by the Advanced Research Projects Agency under ARPA contracts DAVT63-91-C-0029 and DABT63-93-C-0040, by the National Science Foundation under grants NSF IRI 92-12976, NSF ASC 92-12369, and NSF CDA 94-01124, and by the National Aeronautics and Space Administration under NASA Contracts NGT-51023, NAG-1-613, and USRA 5555-22.

References

[Balay et al., 1995] Balay, S., Gropp, W., Curfman McInnes, L., and Smith, B. 1995. PETSc 2.0 Users Manual. *Technical Report ANL-95/11.* Argonne National Laboratory.

[Bishop et al., 1992] Bishop, G., Fuchs, H. 1992. Research Directions in Virtual Environments. *Computer Graphics.* 26:153-177.

[Deering, 1992] Deering, M. 1992. High Resolution Virtual Reality. *Computer Graphics.* 26:195-202.

[Disz et al., 1995] Disz, T., Papka, M., Pellegrino, M., and Stevens, R. 1995. Sharing Visualization Experiences among Remote Virtual Environments. *Proceedings of the International Workshop on High Performance Computing for Computer Graphics and Visualization*, Swansea, United Kingdom.

[Friedmann et al., 1992] Friedmann, M., Starner, T., and Pentland A. 1992. Device Synchronization Using an Optimal Linear Filter. *Computer Graphics*. 25:57-62.

[Funkhouser et al., 1993] Funkhouser, T. and Sequin, C. H. 1993. Adaptive Display Algorithm for Interactive Frame During Visualization of Complex Virtual Environments. *Computer Graphics*. 27:247-254.

[Gropp et al., 1994] Gropp, W., Lusk, E., and Skjellum, A. 1994. *Using MPI Portable Parallel Programming with the Message-Passing Interface*. Massachusetss: MIT Press, Cambridge.

[Holloway, 1991] Holloway, R. L. 1991. Viper: A Quasi-real-time Virtual Worlds Application. *Technical Report TR92-0004*. University of North Carolina at Chapel Hill.

[Hughes, 1987] Hughes, T. 1987. *The Finite Element Method*. Englewood Cliffs: Prentice-Hall, Inc.

[Jones et al., 1992] Jones, M. and Plassmann, P. 1992. Solution of Large, Sparse Systems of Linear Equations in Massively Parallel Applications. *Proceedings of Supercomputing*.

[Kalawsky, 1993] Kalawsky, R. 1993. *The Science of Virtual Reality and Virtual Environments*, New York: Addison-Wesley.

[Liang et al., 1991] Liang, J., Shaw, C., and Green M. 1991. On Temporal-Spatial Realism in the Virtual Reality Environment. *Proceedings of the 1991 User Interface Software Technology*.

[Liu et al., 1993] Liu A., Tharp G., French L., Lai S., and Stark L. 1993. Some of What One Needs to Know about Using Head-Mounted Displays to Improve Teleoperator Performance. *IEEE Transactions on Robotics and Automation*. 9:638-648.

[Mine, 1993] Mine, M. R. 1993. Characterization of End-to-End Delays in Head-Mounted Display System. *Technical Report TR93-001* University of North Carolina at Chapel Hill.

[Cruz-Neira et al., 1993] Cruz-Neira, C, Sandin, D. J., and DeFanti, T. 1993. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. *Proceedings of SIGGRAPH*.

[Reed et al., 1995] Reed, D. A., Shields, K. A., Scullin, W. H., Tavera, L. F., and Elford, C. L. 1995. Virtual Reality and Parallel Systems Performance Analysis. *IEEE Computer*. November.

[Reed et al., 1996] Reed, D. A., Elford, C. L., Madhyastha, T., Scullin, W. H., Aydt, R. A., and Smirni, E. 1996. I/O, Performance Analysis, and Performance Data Immersion. *Proceedings of MASCOTS*.

[SGI] *Silicon Graphics Onyx Installation Guide*. Document Number 108-7042-010.

[Smith et al., 1993] Smith, B. and Gropp, W. 1993. Scalable, Extensible, and Portable Numerical Libraries. *Proceedings of Scalable Parallel Libraries Conference*.

[SpacePad, 1995] *SpacePad Position and Orientation Measurement System Installation and Operation Guide*. 1995. Ascension Technology Corporation.

[Taylor et al., 1995] Taylor, V., Stevens, R., and Canfield, T. 1995. Performance Models of Interactive, Immersive Visualization for Scientific Applications. *Proceedings of the International Workshop on High Performance Computing for Computer Graphics and Visualization*. Swansea, United Kingdom.

[Wloka, 1993] Wloka, M. 1993. Time-critical Graphics. *Technical Report CS-93-50*. Brown University. Department of Computer Science.

[Wloka, 1995] Wloka, M. 1995. Lag in Multiprocessor Virtual Reality. *Presence*. 4:50-63.

Table 1: Lag values (8 processor IBM SP) using IWAY with Internet values in parentheses

Lag Component	**Mean (ms)**	**Std. Dev. (ms)**	$\%Q_{view}$	$\%Q_{interact}$
T_{track}	5.85	2.63	4.51	0.01
$T_{sync(TR)}$	2.92	—	2.25	0.005
T_{render}	105.81	11.01	81.66	0.18
$T_{sync(RS)}$	18788.16	—	NA	32.75
T_{read}	820.83 (1358.57)	49.38	NA	1.43
T_{sim}	37576.31	581.5	NA	65.50
T_{write}	2.26 (2.00)	0.17	NA	0.004
$T_{sync(SR)}$	52.91	—	NA	0.09
$T_{sync(F)}$	15.0	—	11.58	0.03
Total:	—	—	129.58 ms	57370.05 ms

Figure Captions

Figure 1: Parallel Computing/Visualization Environment

Figure 2: ImmersaDesk^{TM} Graphics Display

Figure 3: Components of Lag Time for Q_{view} and $Q_{interact}$

Figure 4: Virtual Disk Brake

Figure 5: Disk Brake Finite Element Mesh

Figure 6: Temperature Distribution (Rotating Disk)

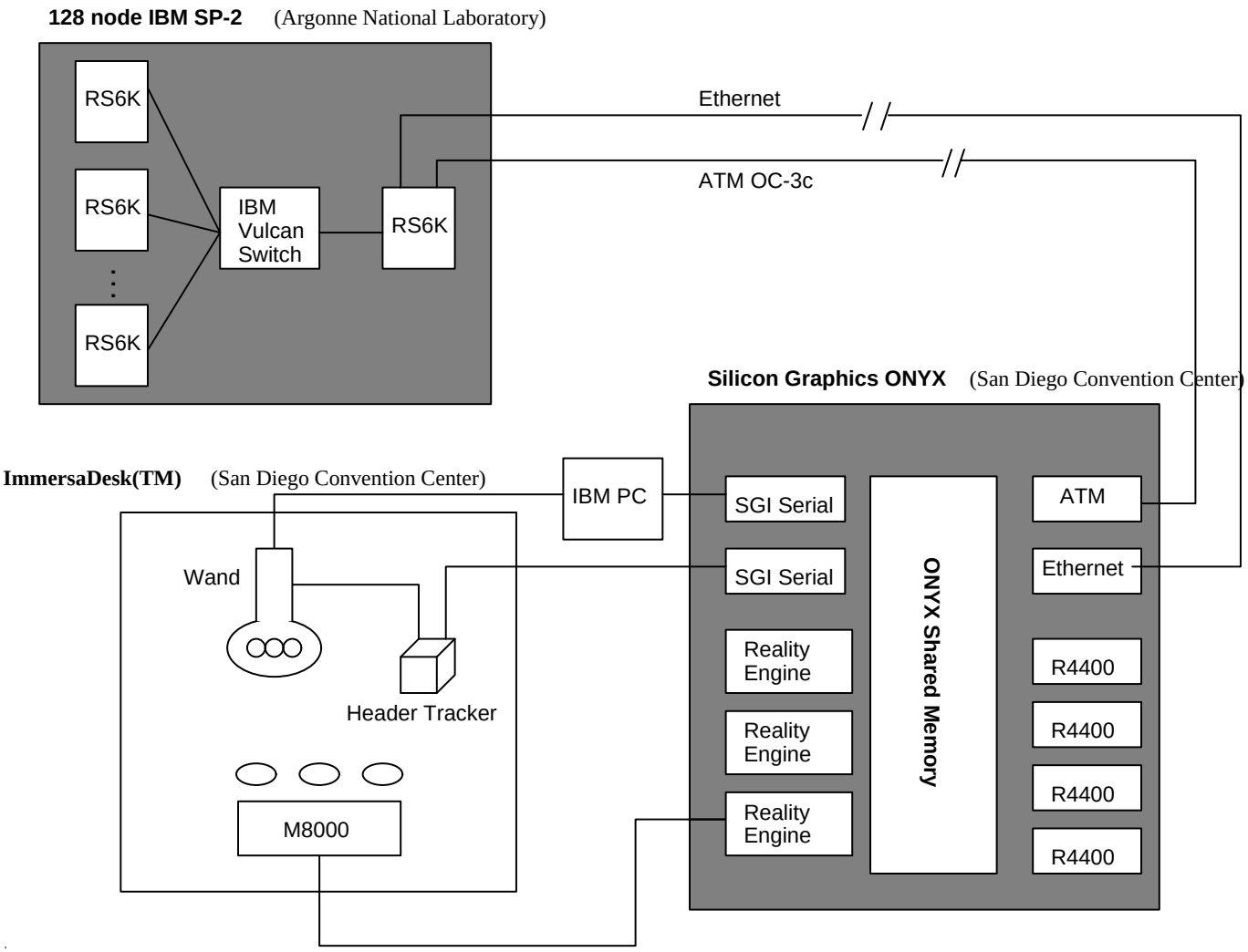

Figure 1: Parallel Computing/Visualization Environment

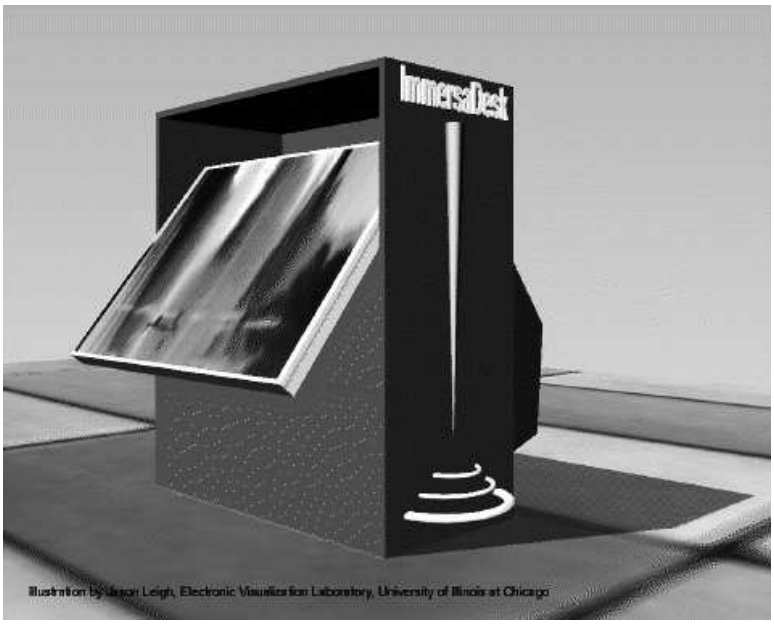

Figure 2: ImmersaDesk^{TM} Graphics Display

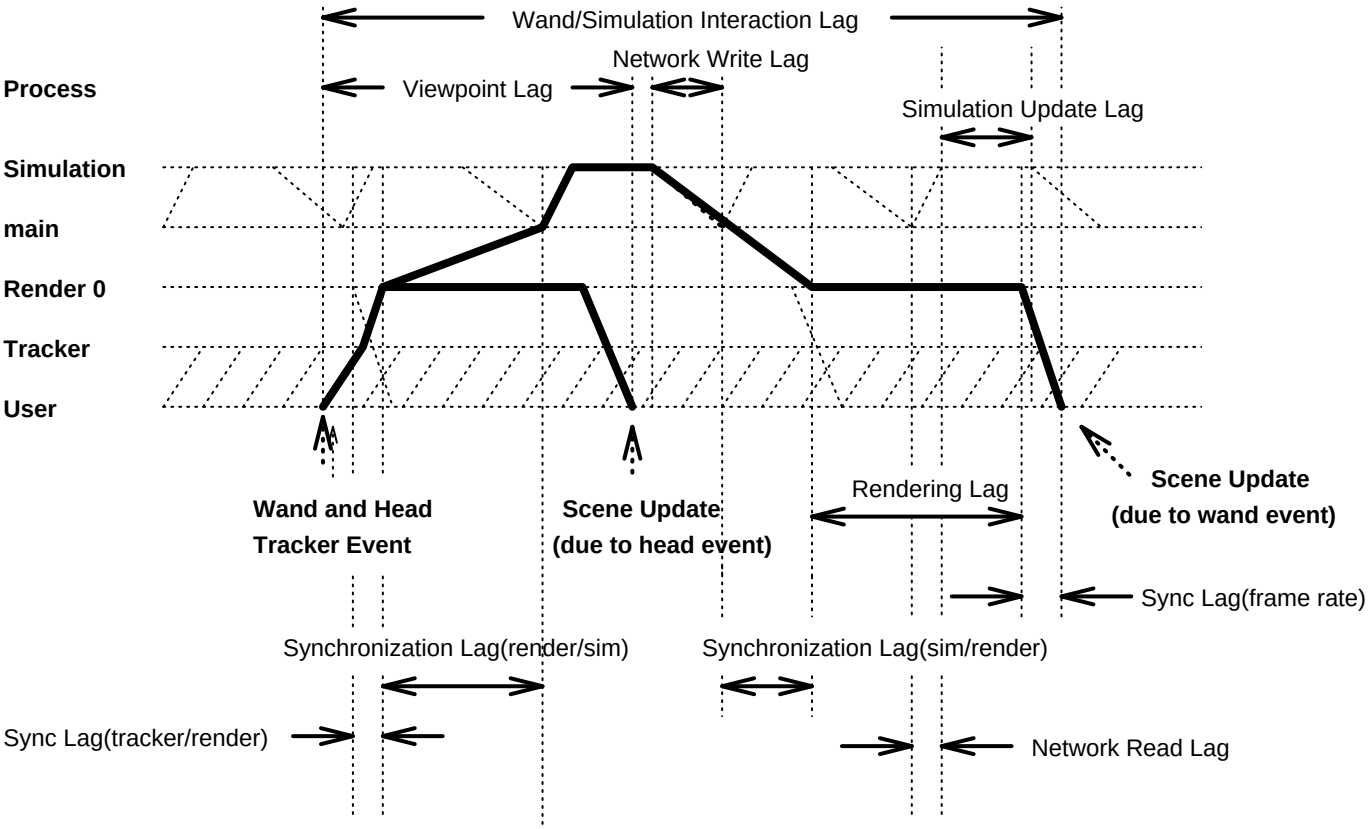

Figure 3: Components of Lag Time for Q_{view} and $Q_{interact}$

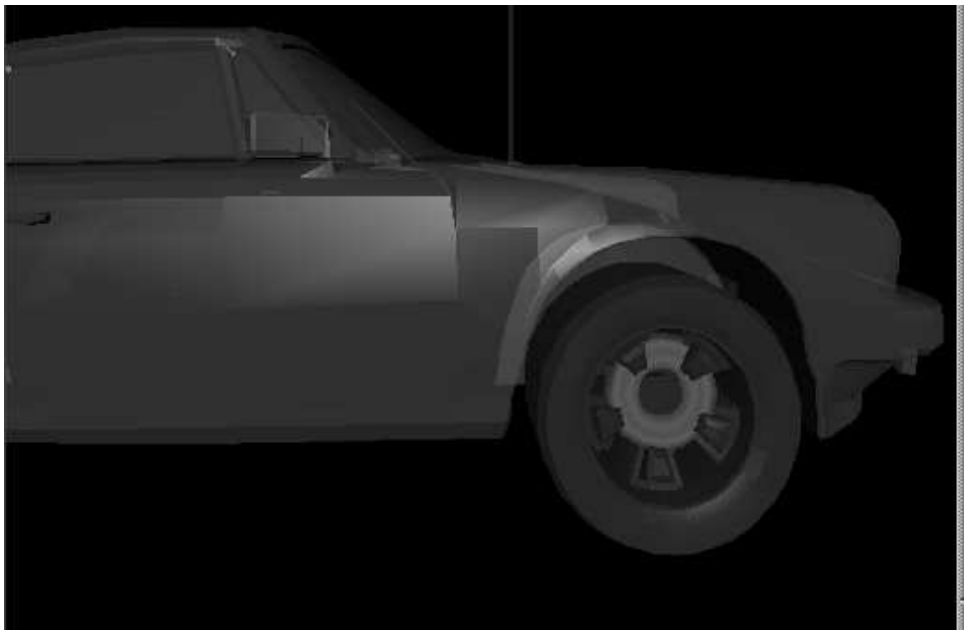

Figure 4: Virtual Disk Brake

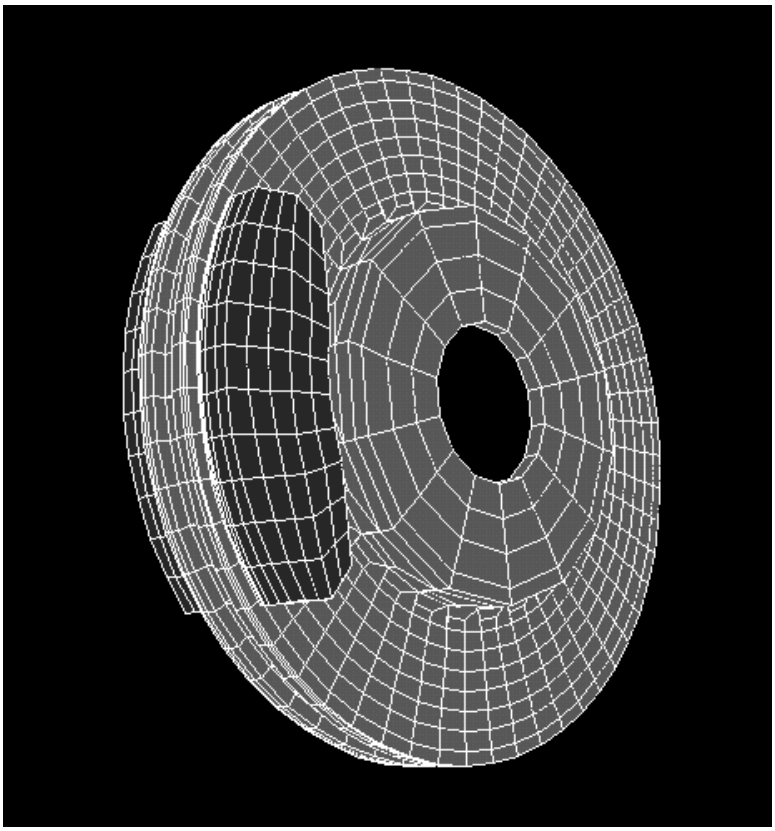

Figure 5: Disk Brake Finite Element Mesh

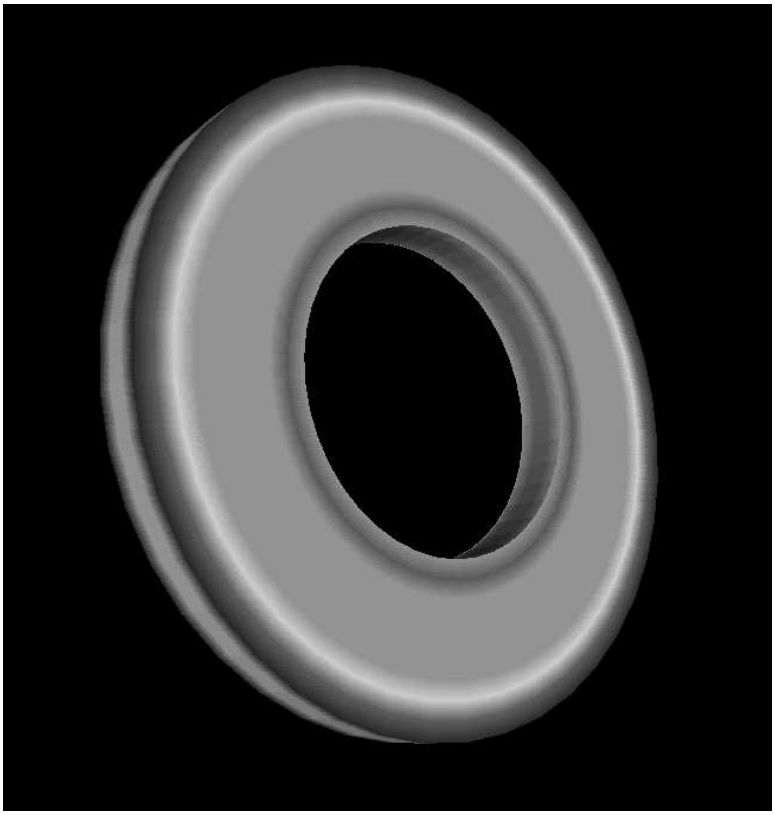

Figure 6: Temperature Distribution (Rotating Disk)