Toward Real-time Interactive Virtual Prototyping of Mechanical Systems: Experiences Coupling Virtual Reality with Finite Element Analysis

Tom Canfield, Terry Disz, Michael E. Papka and Rick Stevens
Mathematics Computer Science Division
Argonne National Laboratory
Argonne, IL 60439

Milana Huang
Electronics Visualization Laboratory
University of Illinois at Chicago
Chicago, IL

and

Valerie Taylor and Jian Chen
Electrical Engineering and Computer Science Department
Northwestern University
Evanston, IL

ABSTRACT

Virtual prototyping involves a synthesis of engineering methodology and immersive, three-dimensional visualization technology. Ideally, this is a process in which computational models are used in place of physical models in the development of a new product or design concept. If used successfully, virtual prototyping can lead to more rapid product design and development.

Software is currently being developed that will enable virtual prototyping of mechanical systems in the CAVE[1] (CAVE Automatic Virtual Environment) at Argonne National Laboratory. This software has two principal components: (1) fast simulation software, FIFEA (Fast Implicit Finite Element Analysis), for analyzing mechanical systems and (2) virtual reality display software for visualizing results and allowing user interaction. This paper discusses various issues related to the coupling of finite element software to the CAVE display system.

[1] CAVE and ImmersaDesk are trademarks of the University of Illinois Board of Trustees.

Keywords: Virtual Reality, CAVE, Virtual Prototyping, Finite Elements

INTRODUCTION

Virtual prototyping of mechanical systems can be accomplished on many levels by integration of immersive three-dimensional visualization technology and powerful tools to do computational simulations interactively. 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 enable viewing of intricate details. 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 [2]. Visualization and navigation in this environment allow one to preview new design concepts for "look" and "feel." Design function can be exercised and tested by using tools to do kinematic analysis and multibody dynamics simulation in this environment. Design performance can be tested by incorporating appropriate analysis tools, for example, finite element models to do stress and/or thermal analysis. The challenge is to perform these functions quickly and accurately enough to enable effective virtual prototyping.

SYSTEM COMPONENTS

The system we describe comprises four components: (1) visualization and simulation environment, (2) graphics-rendering machinery, (3) parallel supercomputer, and (4) software system.

Visualization and Simulation Environment

A unique capability exists in the Futures Laboratory at Argonne National Laboratory for developing tools and testing virtual prototyping concepts. Within this laboratory are a CAVE and an ImmersaDesk (a one walled CAVE) to do immersive three-dimensional visualization [3] . These two devices are networked with the IBM SP supercomputer via regular ethernet or high-speed OC3 ATM connections. Figure 1 shows the hardware components of the system, which are described in detail below.

The CAVE is a multi-person, room-sized, high-resolution, three-dimensional video and audio environment. The CAVE is a theater $10\times10\times9$ feet, made up of three rear-projection screens for walls and a down-projection screen for the floor. Electrohome 8000 projectors throw full-color workstation images onto the screens at 96 Hz, giving $2{,}000\times2{,}000$ linear pixel resolution to the surrounding composite image. Computer-controlled audio provides sound to multiple speakers within the CAVE. A user's movements are tracked with tethered electromagnetic sensors: one sensor tracks head movements, the other a hand-held wand. Stereographics' LCD stereo shutter glasses are used to separate the alternate fields going to the eyes. A SGI Onyx with three Reality Engines is used to create the images that are projected onto the walls and floor. 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. Other observers can passively share the virtual reality experience by wearing LCD glasses that are not tracked.

The ImmersaDesk is a lower-cost, more portable, and smaller alternative to the CAVE. The ImmersaDesk 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 a single stereo display. This image is projected on 4×6 foot screen with a resolution of $1{,}024\times768$.

Graphics-Rendering Hardware

The SGI Onyx is a shared-memory multiprocessor with a high-performance graphics subsystem. The system at Argonne National Laboratory has 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. The Onyx is used to drive the CAVE virtual environment interface as discussed above. The ImmersaDesk uses only one RealityEngine2 graphics pipeline connected to a single Electrohome Marque 8000 high-resolution project to project images onto the one translucent screen.

Parallel Supercomputer

The simulation environment consists of a 128 processor IBM PowerParallel SP with a high-performance input/output system. Each SP node has an RS/6000 cpu with 128 MB of memory and a 1 GB local disk and is connected to other SP nodes via a high-speed interleaved switch. Some of the nodes are equipped with ATM, HIPPI, 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.

Software System

Virtual prototyping of mechanical systems requires two principal components: (1) computational software to do the engineering analysis, and (2) virtual reality display software to do the visualization and allow user interaction. These processes involve multiple levels

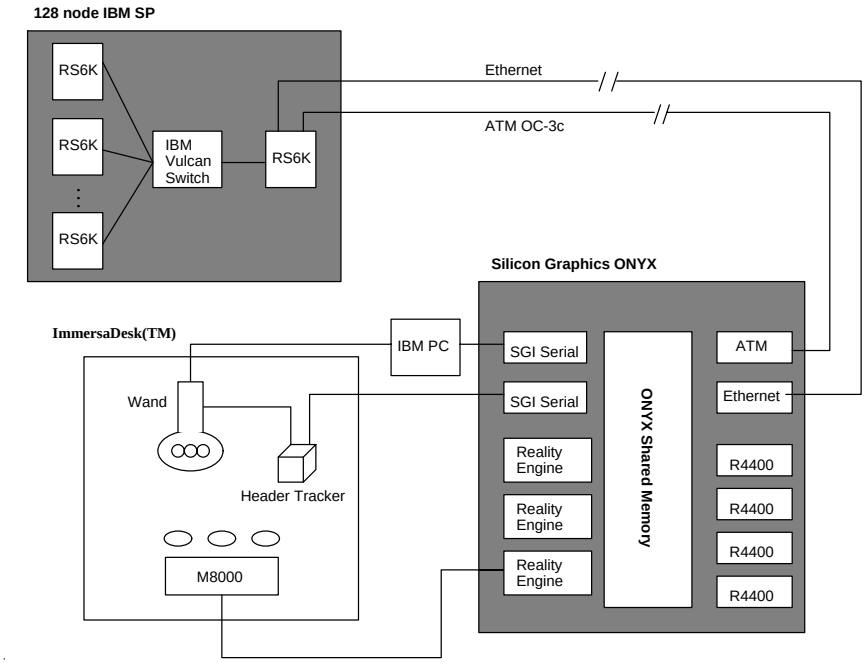

Figure 1: Parallel Computing/Visualization Environment.

of concurrency and can be implemented in a heterogeneous computing environment. Both components should run concurrently and must share data at intervals during the course of a simulation.

We are currently developing fast finite element software (FIFEA) to do analysis of mechanical system coupled with the virtual reality environment. This finite element program runs in parallel using MPI (Message Passing Interface) on an IBM SP [6], whereas the display process runs on a shared-memory multiprocessor SGI ONYX.

FIFEA

FIFEA uses implicit-finite elmenent methodology to do dynamic analysis of solid structures. It can be used to model the thermal-elastic response of materials during low-speed contact with friction. It 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 stresses. This formulation enables the analyis of a wide variety of effects within mechanical systems.

The use of stable implicit time integration offers the possibility of performing dynamic finite element analysis in real time. However, these methods can become inaccurate and lose high-frequency information when the sampling rate is too small. This limits the usefulness of the method in dynamic analysis to low speed mechanical systems. In this case speed depends on many factors, such element size and the frequency content of the driving forces.

In the pseudo-rigid body formulation, a set of multibody dynamic equations is derived assuming each body is rigid. These equations are solved to determine the motion of each body. The dynamic equations for momentum and energy are formulated in the moving frame of reference with additional Coriolis forces. The equations are then solved to determine the additional accelerations, velocities, displacements and temperatures by using unconditionally stable implicit time integration.

The coordinates in a moving body are defined by the following transformation:

$$X_i = X_i^C + A_{ij}(\hat{X}_j - \hat{X}_j^C) \ ,$$

where $\hat{X}_i$ are the coordinates of the material points in initial rest configuration, $\hat{X}_i^C$ are the coordinates of the body centroid in the rest configuration, X_i^C are the coordinates of the body centroid, and A_{ij} is the rotation of the body relative to the fixed reference frame. A_{ij} is an orthogonal matrix. The motion of the

body centroidal coordinates, X^C, and the rotation of the body, A_{ij}, are determined by the solution of the multibody dynamics equations [10]. A material point in a deforming body is given by

$$x_i = u_i + X_i = u_i + X_i^C + A_{ij}(\hat{X}_j - \hat{X}_j^C) \ ,$$

where u_i is the displacement relative to the moving coordinates of the body.

In the finite element formulation, a process of semi-discretization is used to interpolate the displacement and temperature fields over the elements:

$$\begin{aligned} u_i &= u_{ia} N_a(\xi_k) \ , \\ \theta &= \theta_a N_a(\xi_k) \ , \end{aligned}$$

where ξ_k are the local coordinates of the element and the subscript a indicates the node number. In this case summation over the nodal connectivity of the element is implied by the repeated subscript.

In order to determine the elastic displacements and temperatures, it is necessary to solve the partial differential equations for momentum and energy,

$$\begin{aligned} \rho \ddot{u}_i &= T_{ij,j} + b_i - \rho \ddot{X}_i \ , \\ \rho c_p \dot{\theta} &= h_{i,i} + \sigma \end{aligned}$$

where ρ is the density, T_{ij} is the Cauchy stress, b_i is the body force, c_p is heat capacity per unit mass, h_i is the heat flux, σ is a heat source or sink, and $\rho \ddot{X}_i$, is a Coriolis force.

After applying a Galerkin formulation and a fair amount of algebraic manipulation one obtains a set of matrix equations [7] in the form,

$$\mathbf{Kq} = \mathbf{f}$$

where $\mathbf{K}$ is the global "stiffness" matrix, $\mathbf{q}$ is the vector of independent variables and $\mathbf{f}$ is the vector of forcing terms. These equations and the multibody dynamics equations are solved at every time step during the course of the simulation.

Parallelism in FIFEA

FIFEA was designed to run on distributed memory computer architectures using message passing. Without modifying the source code the program may be compiled to run on a network of one or more desk-top workstations, workstation clusters, or a massively parallel MIMD (multiple instruction multiple data) supercomputer.

FIFEA is written in C with some low-level routine written in Fortran. FIFEA makes extensive use of the PETSc (Portable, Extensible, Toolkit for Scientific Computation) to do linear algebra and to manipulate sparse matrices and vectors [1]. All of the message passing within FIFEA uses MPI. This is either done explicitly or implicitly in PETSC, which is also written on top of MPI.

MPI is standardized interface that enables message passing on wide variety of parallel computer systems, especially those with distributed-memory architectures [6]. The main advantage of using MPI is portability. However, in machines were the manufacturers have implemented MPI layered on top of their native message-passing hardware, little or no loss in performance occurs. The same code, developed and tested on a network of workstations, can be recompiled to be run on a massively parallel supercomputer.

PETSc is a suite of data structures and routines that enables the numerical solution of large-scale problems that arise when considering systems of partial differential equations. It is usable from FORTRAN, C, and C++ programs. It also uses the message passing standard MPI for all interprocessor communications.

PETSc has an extensive set of tools that have been designed to facilitate computation on unstructured grids [11]. In FIFEA we have employed these tools to reduce development time, thereby permitting us to concentrate on the proper discretization of the finite element equations and the multibody dynamics.

We use PETSc to define distributed vectors and matrices, to do assembly operations on these distributed data structures, to apply constraints, and to solve the resulting systems of linear equations. PETSc gives us quite a bit of flexibility. By design, our current systems are symmetric and positive definite. Such systems can be solved effectively using a PETSc's solver, BlockSolve95 [8]. Without recompiling FIFEA, we can change the matrix storage structure and solver methodology by simply changing a set directives in the PETSc startup file. If physics is introduced that renders the equations nonsymmetric, PETSc allows us to change the matrix storage structure and solution method without rewriting the PETSc interface.

The parallelism of FIFEA is achieved at several levels: the finite element computations, the repetitive parallel assembly, the parallel contact detection, and the solution of large systems of global equations. The current version of FIFEA can employ one of two methods to do domain decomposition and to distribute the clusters of approximately the same number of elements among the N_p processors. The first is Jones and Plassmann's simple cutting algorithm [9], and the second is a version of Farhat's greedy algorithm [5]. This choice is left to the user. Once the elements have been distributed to the processors, only trivial parallelism is needed to compute the individual contributions of the finite elements to the global stiffness matrix and force vector.

One feature of FIFEA is its ability to detect surface interactions and calculate contact forces between bodies for multibody dynamics and finite element analysis. This is accomplished by extracting the surface mesh from the finite element mesh. Only node-on-face contacts are considered. The search is accomplished by encapsulating the clusters of elements on each processor with parallel pipeds. Structured grids are constructed within each parallel piped to provide an addressing scheme that is used to restrict the search. It is similar to the pigeonhole scheme described by Whirley and Engelmann [13].

Coupling of the Simulation and Graphics

A program was written to display the results of FIFEA simulations in the CAVE environment. It was used to analyze an automotive disk brake system. The program allows the user to select various features of the display and control various aspects of the simulation. It can be used to display the external geometry of the finite element model, the domain decomposition, and the results, such as the temperature and stresses, projected onto the surface of the model. In the present implementation the user can to reset the simulation in its current configuration or restart it from the initial conditions. The mode of analysis can be changed from finite element simulation to rigid-body mode by menu selection. The image of a car is used to provide context in the demonstration of the disk brake model as illustrated in Figure 2.

The virtual reality display program was executed on the SGI Onyx. It consists of three distinct processes (communication, rendering, and tracking) that manage, respectively, communication with the parallel simulation, calculations for surface graphics, and interactive commands. The program is written using the CAVE Library [3], to do the low level graphics rendering, local parallel processing, head tracking, and interactions with the wand. It is a library of C functions and macros that control its operation and simplifies the development of applications for the CAVE. It includes functions to keep all the devices synchronized, produce the correct perspective for each wall, keep track of the walls that are in use, and provide the applications with the current state of all the CAVE elements.

Figure 2: Virtual Disk Brake.

Communications between the display program and the finite element program is accomplished through the use of the CAVEComm library. The CAVEComm library is a set of routines designed to do communications between virtual environments and supercomputers. It enables the development of supercomputer simulations with virtual reality visualizations that can be displayed at multiple sites, with each site interacting, viewing, and communicating about the results being discovered.

RESULTS

The disk brake model was used in the virtual prototyping system to identify potential bottlenecks and evaluate performance. The disk brake model is composed of two brake pads and a rotor. These bodies are divided into 3,790 elements with 5,635 nodes. FIFEA solves the multibody dynamics equations for the three bodies and for the displacements and temperatures at the 5,635 nodes. Since there are four degrees of freedom per node, a linear system of algebraic equations with 22,540 equations must be solved at each time

CPUs	1	2	4	8	16
Time (sec)	1100	530	361	255	240
Speedup	1.00	2.08	3.05	4.31	4.58

Table 1: Execution times and speedup for the disk brake problem.

CPUs	1	2	4	8	16
Solves	78.9	72.4	72.1	66.6	53.1

Table 2: Percentage of time spent doing matrix solves on the disc brake problem.

step.

The measurement of lag times for analysis of the disk brake in the virtual prototyping environment have been reported previously [12].

Computational experiments were performed to determine effective use of the IBM SP for this size problem. A short FIFEA run with problem initialization and one dynamic time step was timed on 1, 2, 4, 8, and 16 CPUs. The results of these measurements, shown in Table 1, indicate that this size problem gains little performances increase by using more than eight processors on the IBM SP.

SUMMARY

In our measurements we have identified the major contributors to lag time to be the time to send the data for visualization and the time to do the simulation. The communication times can be improved by the use of high-speed networking. The overwhelming amount of time spent in the simulation is solving matrix equations (see Table 2). This computation is necessary for implicit time integration of the finite element equations. The time required, however, may be decreased by improvements in iterative algorithms, alternative domain decompositions, and reformulation of the finite element equations to improve the conditioning of the matrices.

We are currently addressing many of the issues raised in this paper actively pursuing solutions to reduce simulation time and lag in virtual prototyping.

Acknowledgments

Work by researchers at Argonne Natinoal Laboratory was supported by the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38. The Work at Northwestern University was supported by an NSF Young Investigator award.

References

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

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

[3] C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. *ACM SIGGRAPH '93 Proceedings*, Lawrence Erlbaum Associates, pp. 135–142, 1993.

[4] T. L. Disz, M. E. Papka, M. Pellegrino and R. Stevens, Sharing Visualization Experiences among Remote Virtual Environments, *High Performance Computing for Computer Graphics and Visualization Proceedings of the International Workshop on High Performance Computing for Computer Graphics and Visualization*, Springer-Verlag, pp. 217–237, 1995.

[5] C. Farhat, A Simple and Efficient Automatic FEM Domain Decomposer, *Computers and Structures*, Vol. 28, pp. 579–602, 1988.

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

[7] T. J. R. Hughes, *The Finite Element Method*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.

[8] M. T. Jones and P. E. Plassmann, Solution of Large, Sparse Systems of Linear Equations in Massively Parallel Applications. In *Proceedings of Supercomputing '92*, IEEE Computer Society Press, 1992.

[9] M. T. Jones and P. E. Plassmann, Computational Results for Parallel Unstructured Mesh Computations, *Computing Systems in Engineering*, Vol. 5, No. 4–6, pp. 297–309, 1994.

[10] A. A. Shabana, *Dynamics of Multibody Systems*, John Wiley & Sons, New York, 1989.

[11] B. Smith and W. Gropp, Scalable, Extensible, and Portable Numerical Libraries, *Proceedings of Scalable Parallel Libraries Conference*, pp. 87–93, 1993.

[12] V. Taylor, M. Huang, T. Canfield, R. Stevens, D. Reed, and S. Lamm, Performance Modeling of Interactive, Immersive Virtual Environments for Finite Element Simulations, *High Performance Computing for Computer Graphics and Visualization*, Proceedings of the International Workshop on High Performance Computing for Computer Graphics and Visualization, M. Chen, P. Townsend, and J. A. Vince, eds., Springer-Verlag, pp. 238–252, 1995

[13] R. G. Whirley and B. E. Englemann, Slidesurfaces with Adaptive New Definitions (SAND) for Transient Analysis, *Proceedings of the ASME: New Methods in Transient Analysis*. AMD Vol. 143, pp. 65–71, 1992.