Technical Report ITL-98-2
July 1998

**US Army Corps of Engineers**
Waterways Experiment Station

# Interactive Computational Steering in Distributed, Heterogeneous High Performance Computing Environments

*by Alex R. Carrillo, John E. West*







Prepared for Headquarters, U.S. Army Corps of Engineers

Technical Report ITL-98-2
July 1998

# Interactive Computational Steering in Distributed, Heterogeneous High Performance Computing Environments

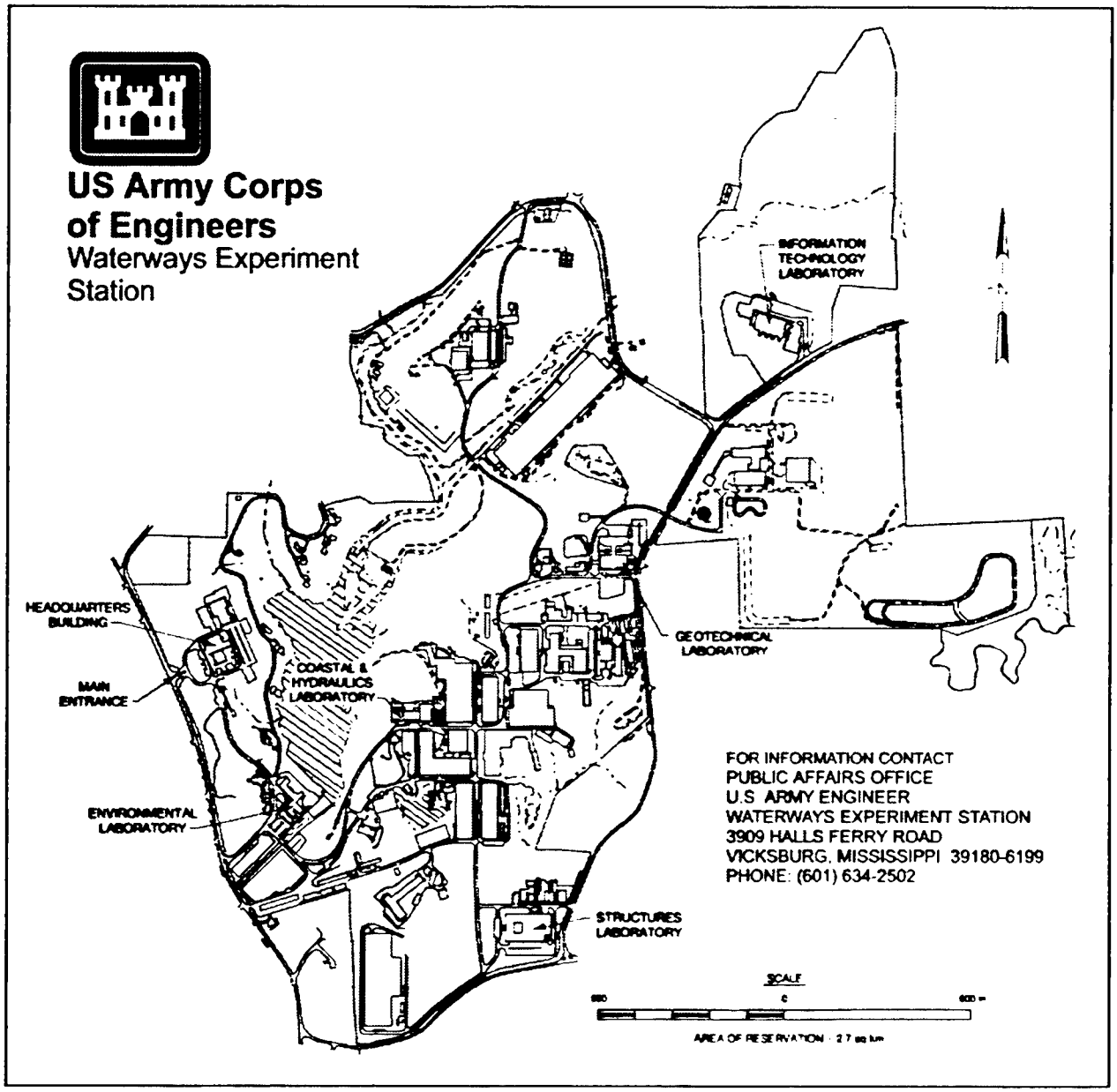
by Alex R. Carrillo, John E. West

U.S. Army Corps of Engineers
Waterways Experiment Station
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Final report



Prepared for U.S. Army Corps of Engineers
Washington, DC 20314-1000




**Waterways Experiment Station Cataloging-in-Publication Data**

Carrillo, Alejandro R.

Interactive computational steering in distributed, heterogeneous high performance computing environments / by Alex R. Carrillo, John E. West ; prepared for U.S. Army Corps of Engineers.

35 p. : ill. ; 28 cm. — (Technical report ; ITL-98-2)

Includes bibliographic references.

1. High performance computing — Simulation methods. 2. Soil dynamics —Mathematical models. I. West, John E. II. United States. Army. Corps of Engineers. III. U.S. Army Engineer Waterways Experiment Station. IV. Information Technology Laboratory (U.S. Army Engineer Waterways Experiment Station) V. Title. VI. Series: Technical report (U.S. Army Engineer Waterways Experiment Station) ; ITL-98-2.

TA7 W34 no.ITL-98-2

# Contents

# List of Figures

## Preface

This report[1] presents the results of the creation of an interactive modeling system. The system described uses distributed computational resources and high performance networking to achieve real-time modeling of large soil masses, although the techniques are applicable to a wide variety of scientific and engineering simulations. This work has been recognized with both a Gold Medal in the High Performance Computing Challenge awarded at Supercomputing '96 in Pittsburgh, PA, and the 1997 Director's Research and Development Award at the Waterways Experiment Station.

This research was performed at the U.S. Army Engineer Waterways Experiment Station (WES), Vicksburg, MS, by Alex R. Carrillo and John E. West, both of the DoD High Performance Computing Center, Information Technology Laboratory (ITL), WES. David A. Horner, Geotechnical Laboratory, WES, and Dr. John F. Peters, Structures Laboratory, WES, are the authors of the original version of the soil modeling application. This work was funded in part by the DoD High Performance Computing Modernization Office, and was made possible by a grant of computer resources at the DoD CEWES Major Shared Resource Center. The work was under the direction of Dr. N. Radhakrishnan, Director, ITL.

The authors would like to thank Dr. M. M. Stephens of the Army High Performance Computing Research Center for helpful discussions and contribution of the original version of the code used to visualize output of the soil model. The authors would also like to thank K. Gaither for her insightful comments leading to the incorporation of the texture map approach to simplifying the particle geometry.

During preparation of this report, Dr. Robert W. Whalin was Director of WES. COL Bruce K. Howard, EN, was Commander.

[1] The contents of this report are not to be used for advertising, publications, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such commercial products.

## Abstract

The ability to visually interact with and guide a running simulation can greatly enhance understanding of both the nature of a numerical model and the characteristics of the underlying physical process. Until recently, interaction with large-scale simulations was inhibited not only by computational limitations, but by network and graphics capabilities as well. However, the increasing power of large high performance computing (HPC) systems, coupled with the advent of new network hardware and innovative techniques for displaying large data sets, has eliminated many of the barriers. It is now possible to link large computational resources with powerful remote graphics workstations to create fully interactive, distributed applications.

This report discusses the development of a distributed, interactive modeling system capable of providing responsive visual feedback to user input. The system is built around a discrete element soil model being used to simulate the behavior of soil masses under large deformations. During the process of designing this system to be interactively responsive to the user for systems of several hundred thousand particles, a range of technical problems were encountered, including the computational burden of the discrete element method (DEM), real-time visualization of large particle masses, and communication of data over large distances between heterogeneous resources using a variety of networks. This report highlights how each of those obstacles was addressed and discusses the effectiveness of the final system.

# Chapter 1

# Introduction

Historically, large scale numerical modeling has been dominated by batch-oriented processing, inhibiting researchers from efficiently inserting experience and insight into the simulation process. Interaction with large scale simulations was further limited by network and graphics capabilities. However, the increasing power of large HPC systems, coupled with the advent of new network hardware and innovative techniques for displaying large data sets, has eliminated many of these barriers. It is now possible to link large computational resources with powerful remote graphics workstations to create fully interactive, distributed applications in areas previously limited to batch processing. Such capabilities can greatly enhance understanding of both the nature of the physical process and the characteristics of the underlying model. During model development, real-time feedback and computational steering can provide researchers with vital clues to algorithmic errors or program bugs, shortening the time needed to create and validate the numerical application. In production, interaction can lead to a shorter, more effective research and development product cycle. Interaction also makes possible the selective storage of simulation results, greatly reducing (or eliminating) disk requirements by limiting output to critical times and/or regions.

The U.S. Army Engineer Waterways Experiment Station is developing three-dimensional numerical methods for evaluating the off-road performance of military land vehicles in various soil conditions. These computational models will supplement field tests and reduce dependence on full-scale testing, ultimately resulting in less expensive, more efficient vehicles. As part of that effort, the current work not only develops an efficient model for large scale soil simulations, it also incorporates this model into a system capable of providing interactive visual feedback to user input.

Ultimately, the driving force behind any interactive modeling system is the model itself. The development of such a system is useless if the model cannot handle meaningful problem sizes at "real-time" computational rates. Previous work in interactive modeling of soil masses by Moshell and Li [18][21] and Burg and Carrington [5] has examined methods for incorporating kinematic and dynamic soil models into real-time simulation and training environments. However, these efforts have concentrated on creating "visually plausible" models of soil behavior to increase the sense of realism in a synthetic environment, not creating models for engineering analysis. Some of this research [18] modeled soil deformation by vehicle systems using Newtonian mechanics, but this work focuses primarily on agglomeration of individual particles into representative groupings of homogeneous, isotropic soil particles to

create a reasonable visual approximation. The primary focus of these efforts has thus been real-time simulation while maintaining a sensible relationship to physically accurate models.

The current work differs from previous efforts in that our primary focus is the creation of a physically accurate soil model and its incorporation into a real-time system for engineering analysis. Before an interactive system was considered, a thorough evaluation of the model capabilities and efficiencies was investigated. Extensive single and multi-processor optimizations were performed to produce a model which could compute solutions fast enough for meaningful interaction, yet remain physically accurate. Further capabilities were gained for the overall system by subdividing computational tasks and spreading them to distributed resources, each subtask being run on (and optimized for) the particular architecture best suited for that subtask's computational algorithm. The result is a system which is distributed over multiple heterogeneous HPC resources[1] and the user's local workstation. Information is transmitted between the user and various system components using a combination of conventional and high-bandwidth networks, as appropriate.

[1]For simplicity, the term "HPC resources" in this report refers only to high end computational resources.

# Chapter 2

# System Overview

There are three critical issues to be addressed in creating an effective interactive simulation for engineering analysis: *i)* computation of the simulation, *ii)* communication between the computational components, and *iii)* visualization of each solution. While performance targets for each of these factors will vary depending upon the goals of the application, each of these issues represents a potential stumbling block to effective interaction. Further, the configuration combining each of these components adds additional hurdles, especially in heterogeneous environments (i.e., machine numerical representations, communication library differences).

The simplest configuration relies solely upon the user's workstation and has the advantage of utilizing computational resources which are often under the control of the researcher performing the work, thus removing competition for over-subscribed HPC resources. However, even with current advancements, most workstations cannot effectively support large simulations interactively. A second possibility is to rely primarily on HPC resources which can meet the computational demands imposed by the simulation. However, as HPC resources are not typically collocated with researchers and even more rarely equipped with direct graphics capabilities, the final display must still appear on the user's local workstation. The most widely-applicable method of doing this is by using X Windows to display graphics generated on the remote HPC machine, which results in generally poor graphics performance. Furthermore, all user interaction must be managed by the HPC resource, which is not an efficient use of these resources, and may be in direct conflict with resource allocation policies.

A third option, and the approach employed in the current work, is to distribute the total workload between both HPC assets and the user's workstation. The goal in the current work is an application that supports an interaction/simulation/visualization cycle which approaches interactive rates for usefully large simulations. Initially, the computations of the system were divided into two primary parts: the numerical simulation, designed for execution on an HPC resource; and the visualization, designed for the user's local workstation. However, it quickly became apparent that interactively rendering particle systems in the several hundred thousand particle range would be beyond the graphics capabilities of most researchers' workstations. The problem was resolved by dividing the visualization into two subtasks, allowing a distribution of the total workload. The first subtask, the data reducer, determines which particles are visible to the user, given current location and viewing direction information. The second subtask actually renders and displays those particles. Since

the rendering and display are performed on the user's local workstation, the second subtask is also given responsibility for user interactions and system coordination. The result is a system which is composed of three parts, each distributed to the computational resource for which it is best suited and linked via the appropriate network technology (as defined by the bandwidth requirements between each task). In selecting the specific architecture on which to place each task, care was taken to select the architecture most suited to that task's computational profile.

Trial runs of this system have been successfully performed on a variety of hardware platforms. One such test, discussed in detail later, placed the numerical simulation on a CRAY T3E, the data reducer on a CRAY C90, and the control and graphics on an SGI workstation. Using a HiPPI "simulation-reducer" connection and an ATM "reducer-graphics" connection, the user was provided with a fourteen frame-per-second computational steering and visualization capability for a 100,000 particle simulation.

# Chapter 3

# Numerical Simulation

Many applications for soil-structure interaction and particle physics involve large discontinuous deformations of particulate media. Such problems include soil plowing, penetrometers, pile driving, soil-tire interactions, hopper flows, tire/track-soil interactions, and mass movements by avalanche. For a particular problem, particulate medium may deform as a solid, flow as a fluid, or behave as individual particles. All of these "phases" may play important roles in the analysis, yet at present no one model exists that can account for all of the different characteristics of soil behavior. The DEM is an alternative to the continuum description for particulate mechanics problems. "Particle model" is a generic term for the class of DEMs where the representation of a physical phenomenon involves use of discrete particles that interact only at inter-particle contact points. A particle model consists of a set of particles in which each has an individual collection of attributes (e.g., mass, particle position, velocity) and some constitutive relationships describing the interaction among particles. The particle attributes evolve according to the equations of motion.

Particle models have successfully been applied to wide variety of problems in plasma physics, astrophysics, fluid dynamics, and molecular dynamics [4] [14] [19] [20]. Cundall [9] is attributed with being one of the first to use particle modeling techniques for evaluating soil and rock mechanics problems. Several researchers have used DEMs to model granular assemblies [1] [7] [16]. Ng and Dobry [22] used DEMs to model small strain cyclic loading. Their simulation results agreed closely with trends found in laboratory tests of sands. Shukla and Sadd [25] used DEMs to investigate how mechanical stress waves propagate in granular material and how they are influenced by media microstructure.

The predominant disadvantage of DEMs in soil simulations is the enormous computational requirement: typically, the maximum number of particles that can be feasibly handled in DEM computations is no more than a few tens of thousands. This number is much less than the number of soil particles in a small-scale laboratory test. For example, a laboratory direct shear test of medium sand will contain over 200,000 particles and triaxial test soil specimens can contain over a million particles. In research applications on fundamental mechanisms, the DEM is used as a surrogate to laboratory experiments because it is possible to extract information that cannot be measured in real tests. In this context validation is essential, meaning that a significant increase in problem size is needed to place the DEM on a firm experimental base. Furthermore, simulations of large deformations in practical engineering problems require orders-of-magnitude more particles than the laboratory tests. The

optimizations and algorithmic modifications discussed below have resulted in a system which can interactively handle a few hundred thousand particles with a very reasonable response time. Much larger simulations are possible when one relaxes the requirement for interactive response, making the current implementation applicable to a wide range of modeling scenarios.

## 3.1 Algorithm Development: Single Processor

The DEM algorithm consists of three main sections: *i)* calculation of the boundary and global body forces, *ii)* calculation of the forces between elements, and *iii)* integration of the equations of motion (to move particles and global objects). The length of the time step is limited by a critical time step which depends on the natural frequency of the particle interaction and damping. Individual particles in this application have been represented as spheres, and the soil mass may contain particles of varying radii. The introduction of shear forces differentiate this model from "smooth particle" models in that a previous contact history between particles must be maintained.

The initial effort of the project focused on the implementation of the DEM algorithm and soil physics, with little or no attention given to computational performance. Each section of the algorithm outlined above looped over all particles, performing computations on a particle-by-particle basis. Contacts were determined by calculating distances between all particles, and redundant contact and force calculations were eliminated by limiting the checks to the current particle and those with a larger index number. The resulting model, though simple, had a run time that grew exponentially with the number of particles simulated. The bulk of the computation occurred in the force calculations between particles. As shown in Figure 3.1 only problems in the couple thousand particle range could be reasonably run.

The initial performance evaluation concentrated on single processor performance on a CRAY Y-MP and was aimed solely at code structure. With no major algorithmic modifications being made, the emphasis was on improved vectorization, improved input/output, and the elimination of redundant work. Though, as shown by the second curve in Figure 3.1, performance was improved by at least an order of magnitude, it was still insufficient to meet project goals and it still scaled exponentially with the number of particles. Even for a parallel implementation to be successful, single processor capability would have to be in the tens of thousands of particles. Thus, although performance increases due to restructuring were significant, further single processor improvements were still necessary.

Performance profiling had clearly established the particle contact checks as the primary computational bottleneck. The all-against-all particle distance check, though simple, scales computationally as $O(N^2)$, making it impractical for large problems. By using a link-cell type method [13][15] to create a neighbors list [27], the contact check was reduced to an $O(N)$ operation, dramatically reducing the computational requirement, as seen by the final curve in Figure 3.1. This method divides the physical space of the simulation into a 1 grid of cells. A simple calculation then determines each particle's "owning" cell. The cell size is set to greater than or equal to the maximum particle diameter, limiting possible contacts to particles within a cell and the 26 surrounding cells. Larger cell sizes can decrease the frequency with which one has to update the neighbors list, but add considerably to the number of potential contacts. Since the creation of the neighbors list amounted to only

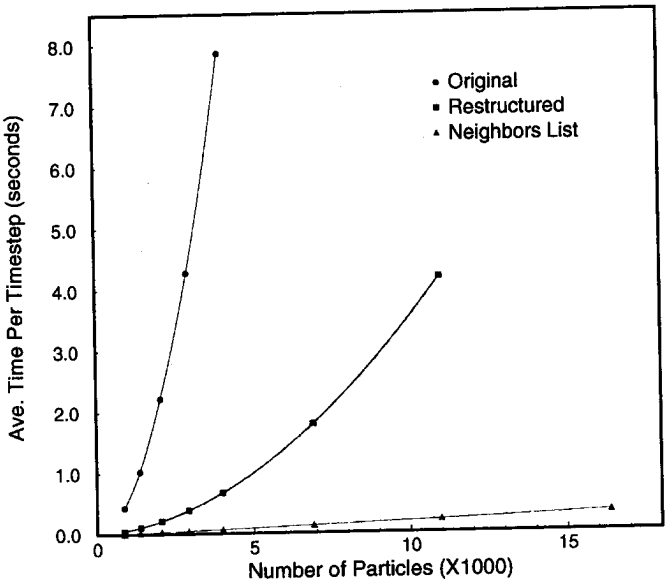



Figure 3.1: Single processor CRAY Y-MP CPU time per time step vs. number of particles for the three phases of single processor optimization.

5%-15% of the total runtime, it was found to be more efficient to minimize the cell size and update each time step.

## 3.2 Algorithm Development: Multi-Processor

Though single processor optimizations substantially reduced runtime, it was clear that a parallel implementation would be necessary to run large simulations with numerous time steps. The neighbors list implementation, though dramatically improving runtime, had created much smaller vector lengths, thus degrading vector processor performance. The smaller vector lengths, coupled with the explicit nature of the DEM, made the model an ideal candidate for non-vector parallel architectures. The success of parallelization efforts in the molecular dynamics discipline [2][23] further supported the effort. Initially, fine-grained parallelism (loop level) was explored using shared memory with compiler directives, but was eventually dismissed in favor of a more robust message passing implementation. The message passing version uses a single program, multiple data (SPMD) approach with domain decomposition. PVM [3] was the message passing library used for development, but was subsequently converted to MPI [8]. maintain portability.

Since the simulations tend to be fairly rectangular in nature, the solution domain is decomposed by subdividing the volume grid-wise in the horizontal directions, producing columnar sub-domains. Planar sub-domains are possible by limiting the subdivisions along a single axis to one, but planar sub-domains tend to create too few sub-domains for smaller

problems when large numbers of processors are desired. A three-dimensional decomposition was not considered because it would not only add to the message passing, but would also risk increased load balance problems due to the greater particle motion in the vertical direction. Analysis showed that most of the load imbalance for the current problems with this decomposition was in the edge domains, and that by increasing the size of the edge domains a load balance could be achieved. Figure 3.2 shows the fractional percentage of computation and communication without load balancing for a 50,000 particle simulation using an 8 by 8 decomposition on a 64 processor nCUBE2 system. Figure 3.3 shows the same problem with the load balancing. (Red/dark grey represents computation, while green/light grey represents communication, including wait times associated with load imbalance.) As can be seen by these figures, this approach was sufficient for the current problems. However, if model geometries were to become more complicated, a three-dimensional decomposition with a more complex data distribution would need to be explored.

The message passing element of the parallel implementation was carefully tuned to minimize its effects. To avoid unnecessary synchronization bottlenecks, all particle information relevant between domains/processors is exchanged once at the beginning of each time step. Bordering particles are treated as "ghost" particles and are only used as contributing elements. All particle information is stored in temporary arrays so it can be passed all at once. This minimizes message latency and maximizes message sizes, at the cost of somewhat more memory (though the model's memory requirement is still dominated by the burden of saving previous contact histories). Overall, the message passing version is very similar to the single processor version. The primary differences are the wrapper routines dealing with particle ownership and passing, a few global reductions to handle global forces and larger objects, and some minor logic to appropriately handle non-owned or "ghost" particles. A more detailed description of the work can be found in [6].

## 3.3 Performance

The parallelization effort substantially increased the number of particles that can be modeled interactively, and test runs of up to one million particles have been successfully performed in batch mode. The model has been run on several HPC architectures and configurations. Figure 3.4 shows the average time per time step for a 100,000 particle simulation for several architectures. Figure 3.5 shows the average time per time step for solutions of varying sizes on a CRAY C90, a CRAY T3E, and an SGI Power Challenge Array.

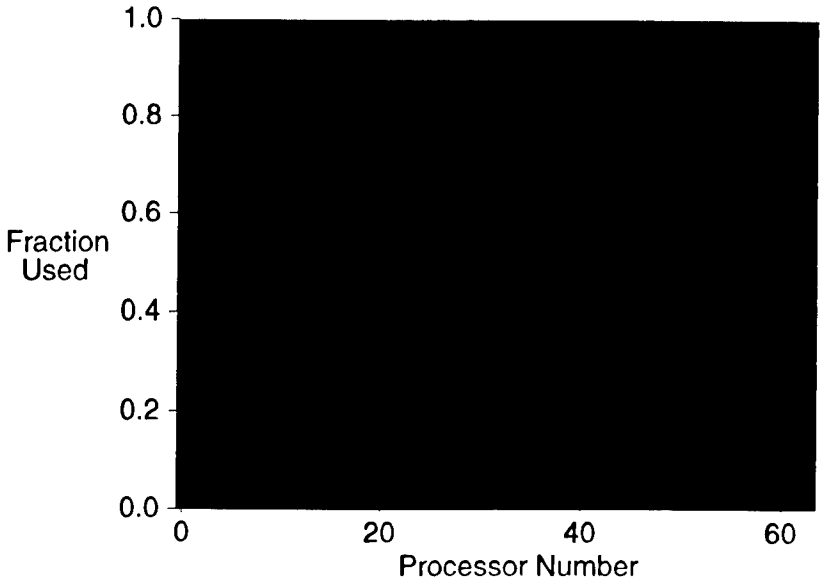



Figure 3.2: Percentage of CPU time (bottom/red) vs. percentage of communication time (top/green) for a 50,000 particle simulation on a 64-processor nCUBE2 without load balancing.

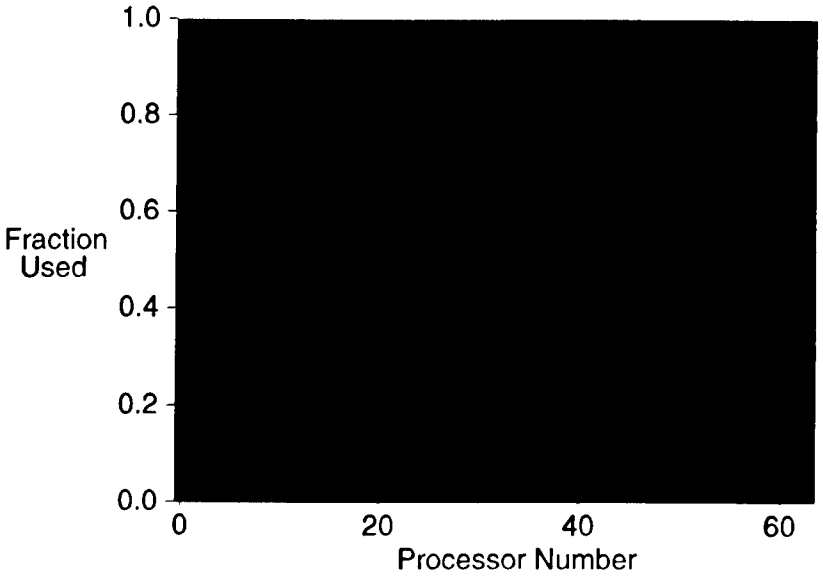



Figure 3.3: Percentage of CPU time (bottom/red) vs. percentage of communication time (top/green) for a 50,000 particle simulation on a 64-processor nCUBE2 with load balancing.

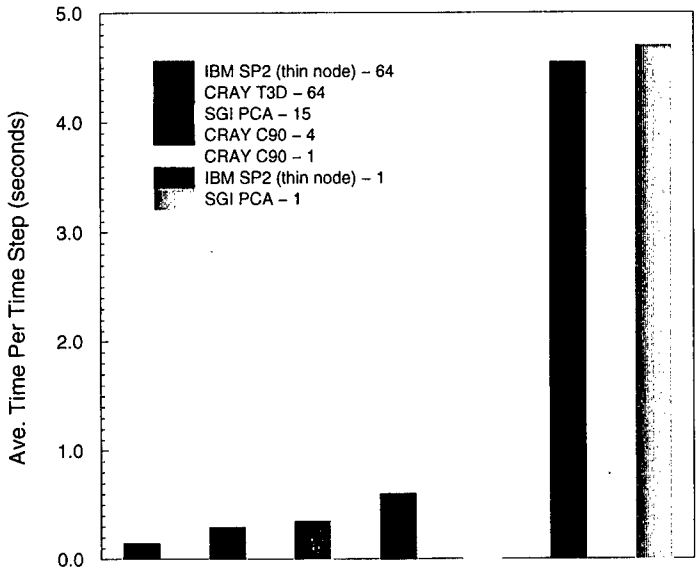



Figure 3.4: Average time per time step for a 100,000 particle solution on several HPC architectures.

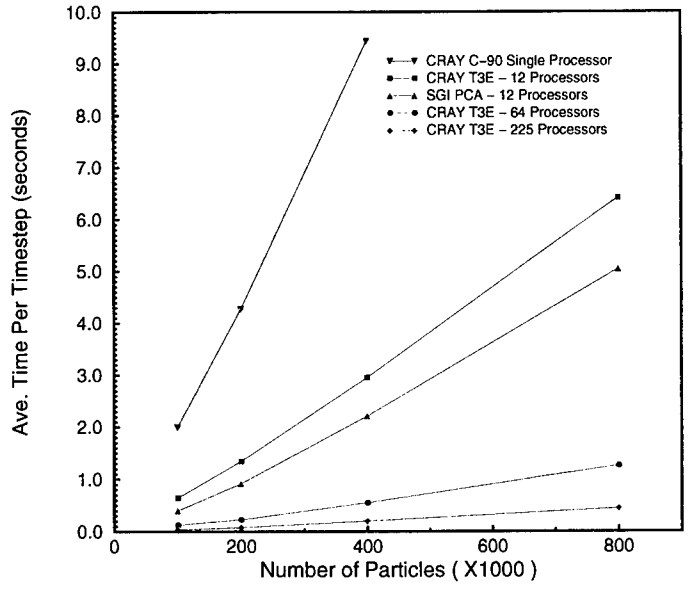



Figure 3.5: Average time per time step for various HPC architectures.

# Chapter 4

# Visualization

Data visualization and display are at the core of this system, providing the visual feedback necessary to understand how the model responds to various external inputs. Because the system is designed to support interactive exploration of soil-structure behavior, the information in the visualization must be easily assimilated (the user does not have long to study any single image), and each image must be generated and displayed quickly enough to avoid creating a bottleneck. The data being analyzed is a system of discrete particles; each particle has a radius (which may be different from other particles), a force, and a position $P = P(x, y, z)$ in $\Re^3$. The most straightforward approach for visualizing this data is to render the particles as spheres, a geometric primitive for which many graphics libraries provide high-performance rendering calls. This representation is highly intuitive, and it is a simple matter to indicate the force being exerted upon each sphere by coloring its surface appropriately. Furthermore, this approach implicitly conveys velocity vector information as particles are displaced during solution updates. This technique relies upon the graphics hardware of conventional desktop graphics workstations for rendering, rather than transferring these computations to an HPC resource as might have been required for more computationally intensive visualization techniques such as direct volume rendering.

Despite the advantages, rendering the particles as pseudo-colored spheres has several important disadvantages when implemented naively. The only computer hardware components proposed in the system thus far are the HPC resource performing the numerical simulation and the user's workstation performing the visualization. Maintaining smooth interaction with the data being visualized is problematic in this configuration for several reasons. First, transferring the results of a $100,000$ particle simulation (for example) to a user's workstation for visualization involves transmitting over $100,000$ particles $\times$ 5 floating point quantities per particle $\times$ 4 bytes per floating point number, or $2 \times 10^6$ bytes. Updating a solution of this size fast enough to maintain reasonable model interaction requires that the user's workstation and the HPC resource performing the simulation be very tightly coupled, a requirement which is considered too restrictive in a useful distributed system. This problem becomes an even larger obstacle when one considers that systems of several hundred thousand particles are desired. Secondly, supporting interactive rendering and visualization of particle systems this large requires very high performance graphics hardware not readily available to most researchers. Even if such hardware is assumed to be available, the size of particle systems simulated would quickly outstrip the interactive capabilities of the graphics hardware. Given

these network bandwidth and graphics constraints it is unlikely that large particle systems can be transferred, processed, and displayed fast enough to support real-time interaction on a user's workstation. The solution to this problem lies in the division of the data visualization into two distinct processes: a pre-visualization data reducing module which utilizes HPC resources to identify visible particles, and a local visualization module which displays only visible particles and manages user interactions.

## 4.1 Data Reduction

While the visualization module provides graphics and handles user interaction, the data reduction module is a critical element in facilitating interactive visualization of large particle systems. Each full solution is transmitted from the particle model to the data reducer as it becomes available, after which the data reducer, a modified ray caster [17][24], uses information about the user's current view to identify particles which are currently visible to the user. These visible particles are then sent to the user's workstation for visualization and display. ("Visible particles" in this context refers to particles which are not occluded by other items in the scene; for example, if the particles are held in a box and the user is viewing the particles from outside that box, then the only particles which are visible are those lining the outside faces of the box.) This approach is very effective - up to 95% of the particles in a simulation are not visible to the user at any one time and can thus be eliminated from the visualization process. In identifying and transmitting only visible particles to the local workstation, this module thus serves to reduce both the network bandwidth required between the user and simulation and the level of graphics performance required of the user's workstation.

### 4.1.1 Algorithm

The ray casting technique used in this system is very similar to those commonly used in direct volume rendering. Volume rendering techniques create a view of a three-dimensional data set by casting rays from the viewer's location through each pixel of a viewing window into the data, as shown in Figure 4.1. As the ray passes through the volume, samples are taken from the data set at intervals along the ray's path and are then mapped to a specific color and opacity. This color and opacity are added to the previously encountered values along the ray to produce a final pixel color for the ray in the image. For example, a ray passing from the viewer into a red cell and then a blue cell (each of equal opacity less than one) may produce a purple pixel in the final image.

In the current work, rather than rendering images directly from the data volume, the algorithm has been modified to serve as a visible surface determination engine (an overview of ray casting for visible surface determination is provided in [10]). The volume in this case is the bounding volume which contains the particles of the most recent solution. Using the same method as the soil model, particles are mapped to cells, creating a neighbors list. Rays are then cast from the viewer's eye through each pixel of the viewing window into the data volume, and the point of intersection for each of these rays with the bounding volume of the solution is used to map the ray to an individual cell. All particles contained in this and neighboring cells are checked for contact with the ray; if contact is found, the particle nearest

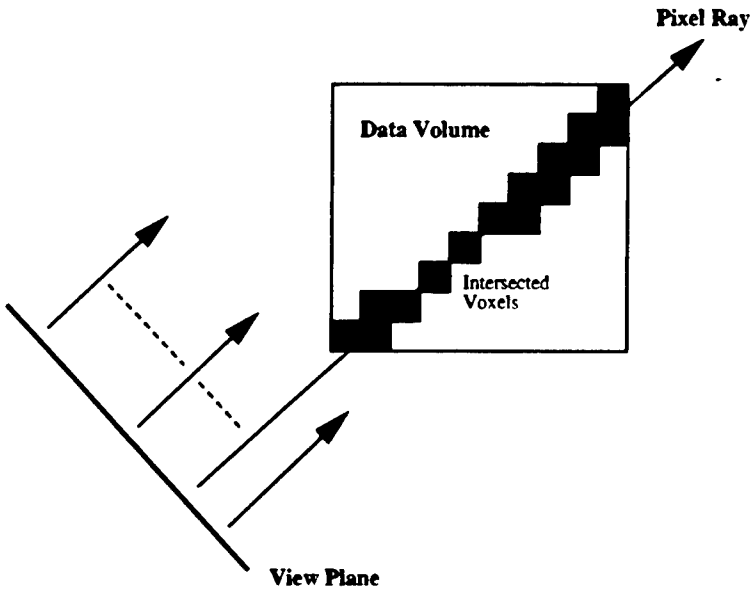



Figure 4.1: Ray casting.

the intersection point is marked as visible. If no contacts are found, the ray is advanced into successive cells of the volume. This process is repeated either until a contact is found, or the ray leaves the volume. After all rays have been cast, the marked particles are gathered into a temporary buffer and transmitted to the graphics workstation. The entire process, except for the creation of the neighbors list, is repeated as updates to the current view are received from the user. The creation of the neighbors list, unlike the ray casting which is static with the number or rays, grows computationally with the number of particles and is therefore only performed when an updated solution is received.

The success of this algorithm in selecting all visible particles for display depends upon the density with which the particles are sampled. Clearly if there are too few rays, or if the rays are not located close together, particles will be missed and the final visualization will not appear as a contiguous mass of particles. To ensure an appropriately dense sample, the object space extents of the window through which the rays are cast are redefined for each new view to match the minimum and maximum particle locations. This enables the volume to be sampled with the maximum number of rays regardless of the user's distance from the volume, and minimizes the number of pixels required in the view window. While this technique is not completely accurate for all views, it is nonetheless quite effective in delivering frames quickly enough for interaction while being accurate enough to maintain confidence in the visualization. The plot in Figure 4.2 shows the average number of reported visible particles in a $40,000$ particle solution as the number of rays used to sample the particle volume increases. Figure 4.2 indicates that over 94% of all visible particles have been identified with a matrix of as few as $200 \times 200$ rays. Although this level of accuracy is not ideal, the user benefit arising from the ability to explore the particle volume at a higher frame rate outweighs the information lost in not detecting the remaining 6% of visible particles. Future efforts are expected to focus on improving the scalability profile of this approach in order to reduce the need for this trade-off.

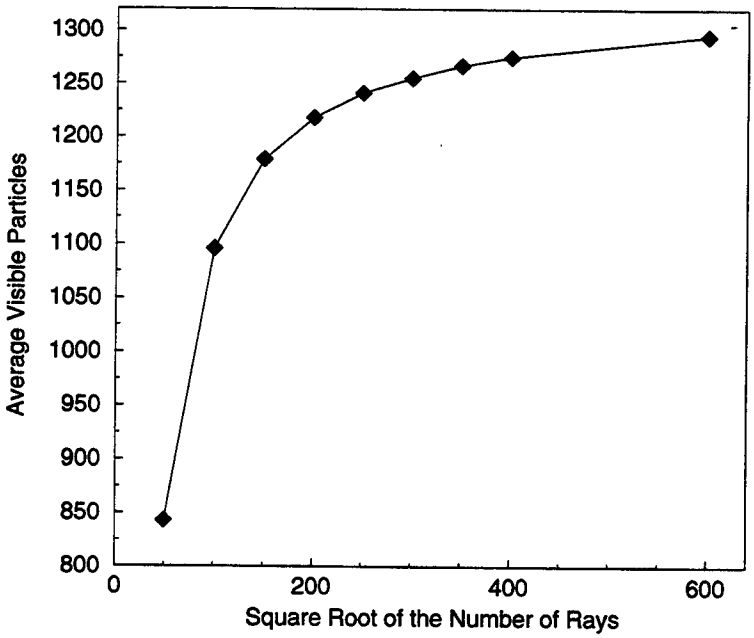



Figure 4.2: Number of visible particles detected as the sample density increases.

### 4.1.2 Performance

The significant drawback of this approach to visible particle determination is the time and computational resources needed to cast an adequate number of rays. Unlike the simulation, the data reducer must be capable of sustaining a frame rate sufficient to produce a "fluid" animation; indeed, it is the data reducer which is most responsible for maintaining the target interaction rate of the system. This component is critical to interactive performance because the user at their workstation does not see the effect of a scene manipulation action (*i.e.*, rotating the dataset to see a different view, etc.) until the data reducer delivers the resulting set of visible particles for display. Thus, the user cannot re-orient or explore the current solution interactively without receiving updates from the reducer at least ten times per second. Because of these stringent performance demands, the data reducer has been targeted for placement on HPC architectures. The algorithm parallelizes effectively on shared memory architectures because each ray traverses the volume independently of surrounding rays [12]. Furthermore, the algorithm vectorizes extremely well, making it well-suited to parallel vector processing systems such as the CRAY C90 or T90.

Communication requirements to the user's workstation scale reasonably well in that, as the number of particles in the simulation increases within a volume of fixed size, the number of visible particles increases only as the surface to volume ratio of the polygon bounding the system (limited by the resolution at which the data is sampled by the modified ray caster). The introduction of the data reducer thus relaxes the requirement for a very high bandwidth connection between the simulation and the user's workstation, enabling the visualization

and display to take place on an computer system close to the researcher, again facilitating the use of hardware resources in those tasks for which each is best suited. However, while the high bandwidth requirement between the user and the simulation has been relaxed, a new requirement for fast data transfer has been added between the data reducer and the simulation. As new solutions are generated by the simulation, they are transmitted to the data reducer, displacing the previous time step's solution as the soil mass being visualized. But, because the time steps in the particle simulation are usually very small, the distance a particle travels in one time step is not visually significant. There is very little advantage to the user in transmitting a solution at every time step to the data reducer. This allows the number of time steps between successive transmissions of solutions to the data reducer to be adjusted to correspond to the time required to deliver each solution. If a HiPPI connection is available between the reducer and the simulation, solutions are sent more frequently than if these components are connected by FDDI or OC-3 ATM connections. This enables a workable balance to be achieved between the solution rates of the simulation, the time to transmit a solution to the data reducer, and the time needed to reduce a view. As commodity networks speeds continue to grow to OC-24 and beyond, the need for this balancing may diminish.

Figure 4.3 summarizes the rate at which the reducer can complete visible particle determination (*i.e.*, frames generated per second) for solutions with varying numbers of particles. The results were generated on twelve processors of both a CRAY C90 and an SGI Power Challenge Array using $200 \times 200$ rays; the frame rate is determined by averaging the single frame computation times from 200 positions around the data set for both static and dynamic data sets. The SGI data is only included to demonstrate that the code is portable to other architectures and to highlight the observation that the algorithm as implemented performs more effectively on vector processors. The static data case is one in which the solver has not provided a new solution so that the neighbors list does not need to be rebuilt as it does in the dynamic data set case. Typically the user will view a single data set from many views before the solver updates a given solution; these curves therefore represent performance bounds on the frame rate delivered to the user. For a well-balanced simulation the performance will be closer to the top curve. Note that while there is a strong performance dependence upon the number of particles, the data reducer provides for a response within our target range even for simulations containing approximately 400,000 particles, and solutions involving up to 800,000 particles can be visualized effectively with some additional latency in response times.

## 4.2 Display and User Interaction

After the visible particles have been identified, they are transferred to the visualization module, which is responsible for processing and displaying these particles and for handling all user interactions within the distributed system. Figure 4.4 is a snapshot from the system which shows a representative visualization and the graphical user interface (GUI). The rendering and display tasks are implemented in C using OpenGL, with X/Motif used for the GUI. Both X and OpenGL are widely available, preserving some degree of portability for the system. The image shows a plow being forced through a mass of particles from left to right. As discussed above, each of the particles is rendered as a sphere and the forces

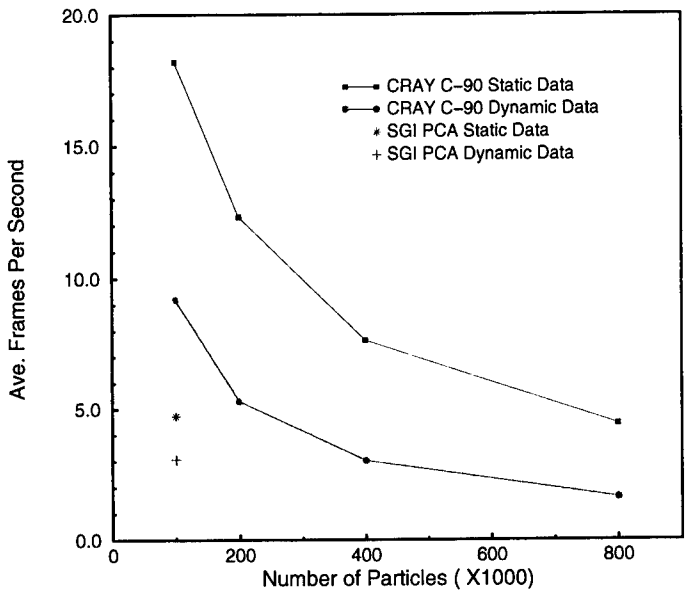



Figure 4.3: Data reducer performance (average frames per second) on 12 processors versus the number of particles.

exerted on each particle are indicated by colors which range from blue for the lowest values through green and yellow to red for the highest value. As the user explores the data set, updated viewing matrices and control information are sent to the data reducer for use in identifying the new set of visible particles. The user interacts with the particles of the soil model through the interactor (the plow in this image) which is controlled by the user with a mouse or similar input device, and is used to apply external forces to the particle system. The plow in Figure 4.4 is one of several simple interaction geometries currently supported. Specific applications to incorporate more complex interaction geometries (*e.g.*, tank tracks, etc.) are in development.

As with the simulation and the data reducer, care was taken to balance the performance of the visualization and GUI tasks with the rest of the system to avoid producing bottlenecks or overloading other modules. Even though the data reducer can eliminate as many as 95% of the particles before rendering, for large simulations this still leaves thousands of particles to be displayed. Each particle is drawn as a tessellated sphere, and an acceptable approximation to a sphere can be achieved with about 25 polygons. Each polygon must be processed by the graphics pipeline and rendered before final display. If a large number of particles are visible to the user, the graphics processing time can become too great for interactive rendering. In fact, for a simulation of large enough size this is guaranteed to happen: 5% of a one million particle solution is 50,000 particles - too many to render interactively on generally available hardware. The goal then was to improve the efficiency of the rendering phase and increase the number of particles which can be displayed and manipulated interactively. The visualization

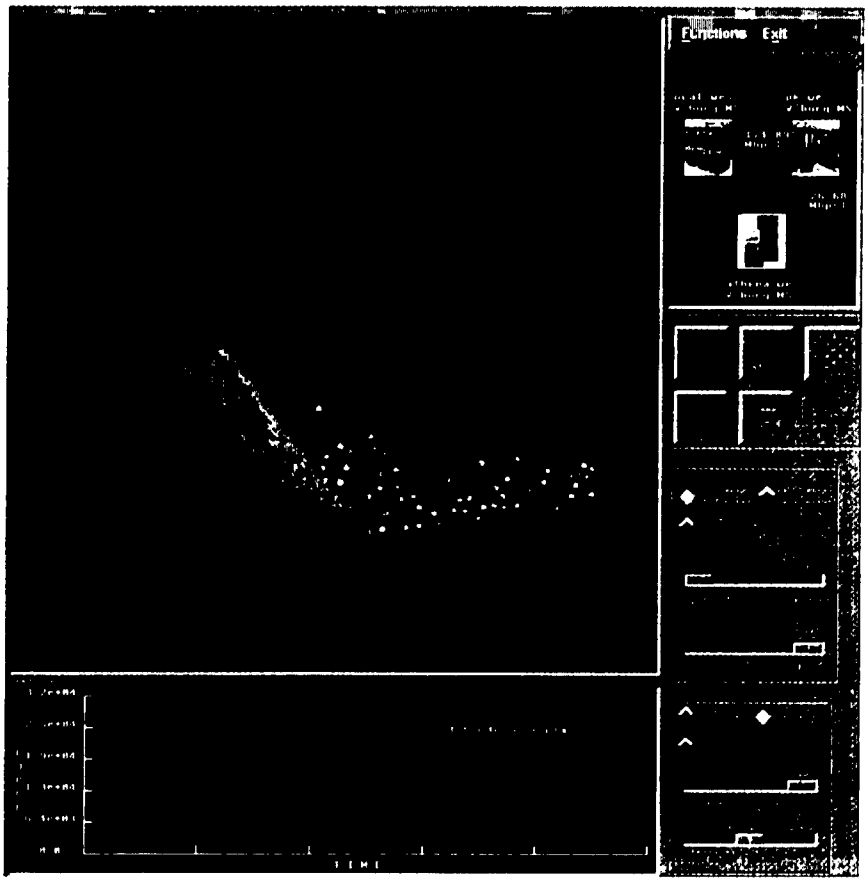



Figure 4.4: Interactive Soil Model.

code was profiled to highlight time consuming computations in the rendering process and identify which part of the graphics pipeline was most heavily stressed during display. The analysis was performed on a Silicon Graphics workstation, and exposed a bottleneck in the per-polygon stage of the graphics pipeline. This bottleneck was resolved using standard graphics pipeline tuning techniques [26], resulting in an overall rendering speed four times faster than the original code.

Despite this improvement, the distributed system still started to show signs of rendering speed limitations in handling simulations of particle systems with more than one to two hundred thousand particles. The primary cause of this difficulty is the number of polygons required to create a reasonably spherical particle. For users with a workstation which supports hardware texture mapping, there is an elegant solution to this problem which greatly simplifies the geometry and allows a significant improvement in the number of particles which can be handled interactively. Rather than building a spherical object from many polygons, each sphere is represented by a single polygon. This polygon (a square with side twice the radius of the sphere) is colored to indicate the magnitude of the force exerted upon the particle it represents, just as the spheres are. The polygon is then texture-mapped with the image of a lighted sphere and oriented dynamically so that it always faces the user, giving the appearance of a colored, smooth sphere when viewed from any angle [11]. Because texture mapping is done in hardware, this technique will allow approximately 20 to 25 times more spheres to be represented than the multi-polygon representation, and these textured spheres don't appear as faceted in the tessellated case. This improvement is the theoretical best-case, achievable only if the polygons in the texture-mapping approach are not sorted first. In practice, these polygons must be sorted by distance from the viewer before rendering to ensure that transparency effects are correctly computed. This introduces additional overhead which reduces the improvement to a factor of approximately ten. Added performance benefits are derived from the texture-mapping approach because the texture map includes a specular highlight, which means it is not necessary to perform lighting calculations. The overall performance improvement is significant; rendering $5,000$ spheres at 24 Gouraud shaded polygons per sphere on an SGI Onyx workstation (very large facets) requires 0.25 seconds per frame on average, while visualizing the same number of spheres using the texture mapping alternative requires only 0.03 seconds. More efficient sorting algorithms are being introduced to increase the performance even further.

Finally, a beneficial consequence of the strongly distributed design of this system is that changes in one portion of the system need not affect the remaining system components. Thus, modifications to the visualization module to provide support for a broader range of interaction devices, such as head-mounted displays, need not impact the remaining system modules. The system currently supports both standard desktop monitors and stereographic displays. Future work will include expansion of the stereographic capability to immersive interaction environments.

# Chapter 5

# Communication

In order for the individual software components to function as a system they must communicate with one another, and, as shown in Figure 5.1, information in this system travels along three data paths. Each of these data paths has widely varying bandwidth requirements, and the computational resources used in the system are heterogeneous and possibly geographically distributed. These factors had a key influence in developing the communication protocols used to transfer data among modules in the system.

There are several successful portable software libraries available in the public domain which support communication between heterogeneous architectures. Notable among these are PVM and MPI. Initially, we considered using one of these libraries as the software layer connecting the components of the distributed system. This would have enabled use of widely accepted and supported message passing software and shorted the development time by eliminating the need to write special-purpose communications routines for the application. However, early tests indicated that a general purpose message passing library would not provide all of the support necessary to facilitate interactive distributed simulation. First, message passing libraries often support communication between machines having different internal data representations using the eXternal Data Representation library (XDR), and our experiences indicate that on some machines, such as the CRAY C90, conversion of large messages with XDR is too slow for interactive computation. Second, after analysis of the bandwidth requirements between the various system components, it became clear that several different networking technologies would have to be employed to reach system goals. The most effective way to ensure optimal use of those technologies is to have direct access to the network programming layer.

Communication paths are therefore created and maintained using a simple two-layer communications protocol implemented specifically for this system. The first layer abstracts communication operations as a set of high level routines, which simplifies the process of substituting new communications technology into the system. The second layer performs the data communications. Routines in this layer vary according to the architecture and networking technology employed. For example, native HiPPI communication routines can be used between various machines and the CRAY C90, while the ATM interface on the C90 is accessed through the UNIX stream socket mechanism because direct programmer access to the ATM programming interface routines is not supported. Where data conversion is necessary routines tuned to the machine doing the conversion are employed. For example,

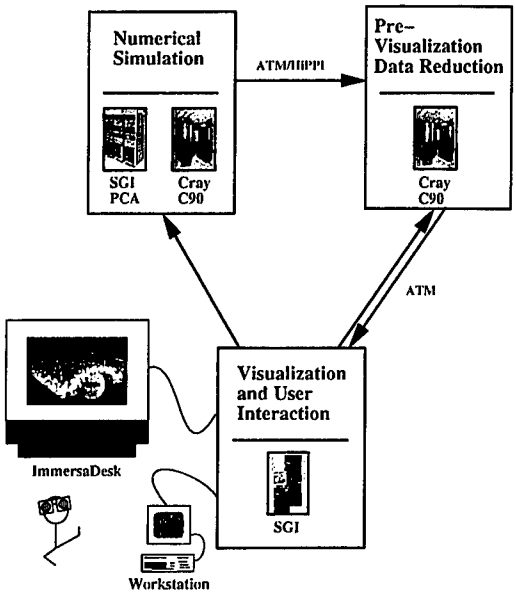



Figure 5.1: System overview.

when the data reducer is placed on a CRAY C90, the set of visible particles must be converted to IEEE floating point representation before being sent to the SGI workstation for display. This conversion is accomplished using the vendor-supplied routine `CRI2IEG()`. Note that in this case it is advantageous to perform the conversion before transmitting the data, as the IEEE single precision floating point representation only consumes four bytes while a single precision Cray floating point quantity uses eight bytes. While this approach is developer-intensive and not as general as using MPI or PVM, it does provide the maximum opportunity for high performance communication. Furthermore, a degree of portability is maintained for the system since it is possible to use UNIX sockets to communicate between all components in the system on machine/network interface combinations for which specialized communications routines have not been developed.

As mentioned earlier, the bandwidth requirements of the various data communication paths in the system vary widely according to function. The path between the visualization and the simulation updates the simulation module with the current position and orientation of the interaction geometry. This connection requires less than 100 bytes per update to transfer interactor control information, and even at twenty updates per second this amounts to a small quantity of data. The simulation-data reduction connection is used to transfer the entire solution at each update, the size of which depends upon the number of particles in the simulation, while the connection to the data visualization module transmits the currently visible particles, the number of which depends upon both the number of particles in the system and the user's current view. For example, a $100,000$ particle solution requires $100,000 \times 5 \times 4 = 2 \times 10^6$ bytes be transferred to the data reducer per solution update (assuming that four byte floats are used during data transfer). If an average of $4,000$ of these particles is visible to the user at any one time, $4,000 \times 5 \times 4 = 8 \times 10^4$ bytes are transferred to the user's local workstation for processing and display per view update. It is expected that view updates will be sent on average more frequently than solution updates

as the user manipulates the view of the particle volume. The visualization-data reducer connection should therefore be capable of handling at least $10 \times 80,000$ bytes per second, a rate manageable by ATM networks over longer distances. However, even assuming the maximum OC-3 transfer rate of 155 Mbits/s is available, an ATM connection will limit the update rate between the simulation and the reducer to less than ten solutions per second for a 100,000 particle simulation, a figure generally too low for acceptable interaction. Worse, it is not realistic to assume that the maximum theoretical rate is attainable. Supporting reasonably responsive interaction thus requires a sustainably high bandwidth connection between these two elements of the system. In practice this is not an overly burdensome restriction, as many modern HPC centers currently support tightly coupled computing environments. For example, a HiPPI connection between two HPC assets in the same center is not uncommon. It is still possible, of course, to use lower bandwidth connections (for example if tightly coupled HPC resources are not available), but the simulation should restrict the number of solutions it sends to the data reducer per second such that the total amount of data transferred is not greater than the bandwidth of the network connection.

A final optimization made in system communications was the separation of I/O tasks in the numerical simulation to a dedicated processor. As mentioned earlier, the numerical simulation is parallelized using MPI, and it was relatively simple to designate one processor as responsible for all communications with the data reducer and the user's workstation. The remaining processors in the simulation compute the new solutions. By using asynchronous messages between the compute processors and the I/O processor, and by adjusting the number of time steps between print intervals to match the send time between the simulation and reducer, computation can be completely overlapped with I/O, greatly reducing idle processor time. A similar approach is being explored for the data reducer.

# Chapter 6

# Results

To provide an indication of the present capabilities of the system, a test simulation for a 100,000 particle system was performed using a 16-processor CRAY C90 for the data reduction, 101 processors[1] of a CRAY T3E for the numerical simulation, and a Silicon Graphics Power Onyx Infinite Reality system for the final display. The two CRAYs are connected to one another via a HiPPI channel. The connection between the SGI and the C90 (data reducer) is an OC-3 ATM link, while the much smaller bandwidth requirements between the T3E (simulation) and the SGI permit use of a standard Ethernet connection. The ATM connection between the C90 and the SGI workstation is facilitated by special-purpose hardware known as a Bus-Based Gateway, or BBG. Cray Research does not support a native ATM capability on the C90; the BBG facilitates ATM communication by translating between the HiPPI and ATM protocols. Because the HiPPI channel is used by the C90 to communicate with both the numerical simulation and the user's workstation (via the BBG/ATM link) the competition for resources is expected to impact the achieved bandwidth to both processes.

The physical problem being simulated was a user-controlled plow being forced through a mass of heterogeneous particles. In order to balance the speed of the simulation with the needs of the rest of the system, the simulation was set to send new solutions to the data reducer at ten time step intervals. This resulted in reasonably smooth particle motion in response to user actions on the plow while preventing the data reducer from being overwhelmed by the flow of new data from the numerical simulation. Evaluation runs using smaller send intervals yielded some reduction in the responsiveness of the data reducer in allowing the user to smoothly manipulate the particle mass. Larger send intervals resulted in poor system response to user actions on the plow. Dedicating I/O tasks on the data reducer to a separate process, as was done for the numerical simulation, would reduce the impact of the send interval choice on the frame rate delivered to the user and is currently under development.

The workstation used for the visualization module supports hardware texture-mapping, so a single textured polygon was used to represent each sphere. In this configuration, the data reducer was able to reduce an average of fourteen frames per second while manipulating the 100,000 particle data set, providing a response well within what is typically accepted as interactive performance. With a send interval of ten time steps, the numerical simulation

[1] 100 for computations, and 1 (as discussed previously) for I/O.

delivered new solutions to the data reducer at a rate of between five and seven solutions per second, corresponding to a computation rate of 50-70 solutions per second. At seven solutions per second the perceived response of the system to user actions on the plow is slightly less than optimal, but still smooth enough to be perceived as interactive. One factor which affected the delivery of solutions to the data reducer was competition for HiPPI bandwidth on the C90. Also, the TCP/IP protocol was used for communications over this link rather than native HiPPI. The end result is an effective bandwidth of only 100 Mbits/sec, significantly less than the 800 Mbits/sec maximum for this device. Future efforts will extend support for native HiPPI communications between the T3E and the C90.

The average number of visible particles delivered to the user during the course of the simulation (which involved manipulating the dataset through 360 degrees several times) was 3.5% of the total particles in the simulation, indicating that the data reducer serves its purpose well in reducing the bandwidth requirement between it and the display module. These particles were delivered, on average, at 70 Mbits/sec over the ATM connection between the two modules. This rate is only a fraction of the theoretical maximum of 155 Mbits/sec. The chief culprits in this low performance are the overhead associated with the TCP/IP protocol and competition for I/O service from the CRAY C90 through the Bus-Based Gateway.

# Chapter 7

# Conclusions

A distributed, interactive particle simulation system has been developed which allows real-time assessment of the response of vehicles and vehicle systems in a large, three-dimensional soil mass. The system has been developed to help evaluate the off-road performance of military land vehicles in various soil conditions, and will supplement field experiments, ultimately resulting in less expensive, more efficient vehicles designed with less dependence on full-scale experimentation. To facilitate the highest possible level of interactivity, computational tasks in the system are divided into three major subtasks, each optimized for the particular architecture best suited to that subtask's execution. The result is a system which is distributed over multiple heterogeneous computational resources, with information transmitted between the user and various system components using ATM and HiPPI networks.

The system is strongly distributed, which means that changes in one portion of the system need not affect the remaining system components. Thus modifications to a particular module, such as the numerical simulation or additional support for non-traditional display devices, can be made with no impact on the remaining system. Furthermore, the application has been designed to support a range of modeling objectives, and can be used in computation-only mode for very large simulations as well as interactive mode. Where practical the various components have been designed for portability by using standard interface libraries such as MPI, OpenGL, and X/Motif.

In designing this system to be interactively responsive to the user, a range of technical problems were encountered. The computational burden of the discrete element method was addressed by optimizing the computations with both single and multi-processor techniques. Real-time visualization of large particle systems was accomplished by first dividing the visualization task into subtasks, and then tuning each subtask for the target architecture. The speed of the display stage was further improved with the use of textures to greatly simplify particle geometry. Also, communication support for ATM, HiPPI, and standard Ethernet communications in coordinating the flow of data between the various heterogeneous computational components in the system presented a significant challenge. Finally, as each component was developed, special efforts were taken to ensure that the computation and communications requirements of each portion was satisfied and optimized without introducing significant new bottlenecks in the remaining system.

# Chapter 8

# Future Work

Future development efforts will address several concerns in the present system. The results generated by simulations with the current discrete element model have successfully verified the model against laboratory-scale experiments (several hundred thousand particles) of soils subjected to large deformations. The DEM has unique benefits for soil mechanics problems from a material science standpoint, but despite the computational advances of the current work its applicability remains limited to laboratory-scale problems. However, using contact statistics from the current DEM simulations, a smoothed discrete element method (SDEM) is being developed to extend simulations to the field-scale (*i.e.*, several orders-of-magnitude more particles in the soil masses simulated). This smoothed particle system is intended to model the macroscopic behavior of granular media. The method being developed is similar to the particle-in-the-cell method used in smoothed particle hydrodynamic codes as described by Hockney and Eastwood [14]. Follow-on efforts will include the development of constitutive laws for cohesive soils, the introduction of a pore fluid, and enhanced boundary conditions.

As mentioned earlier, the scalability profile of the visible particle determination method needs to be improved to reduce the need to trade-off accuracy in the number of particles selected as visible for real-time performance. Efforts will also be directed at improving the efficiency of the communications layer by adding native support for a wider range of networking technologies. Furthermore, additional research in the visualization module is anticipated to extend the user interaction paradigms to immersive technologies.

Finally, the infrastructure developed in the current work is applied to a specific soil-structure interaction problem. There are no intrinsic design elements which would prevent this system from being applied to more diverse application domains, and future efforts at generalizing the techniques to other applications are expected.

# Bibliography


[1] R. J. Bathurst and L. Rothenburg. Micromechanical aspects of isotropic granular assemblies with linear contact interactions. *Journal of Applied Mechanics*, 15:12–23, 1988.

[2] D. Beazley, P. Lomdahl, N. Gronbech-Jensen, R. Giles, and P. Tamayo. Parallel algorithms for short-range molecular dynamics. *World Scientific's Annual Reviews in Computational Physics*, 3, 1995.

[3] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. Sunderam. A user's guide to PVM: Parallel virtual machine. Technical Report ORNL/TM-11826, Mathematical Sciences Section, Oak Ridge National Laboratory, September 1991.

[4] W. Benz. Smooth particle hydrodynamics, a review. In *The Numerical Modeling of Linear Stellar Pulsations*, pages 269–288. Kluwer Academic Publishers, 1990.

[5] J. Burg, J. M. Moshell, and *et al.* Behavioural representation in virtual reality. In *Proceedings of Behavioral Representation Symposium.* Institute for Simulation and Training, 1991. Orlando, FL.

[6] A. R. Carrillo, D. A. Horner, J. F. Peters, and J. E. West. Design of a large scale discrete element soil model for high performance computing systems. In *Proceedings of Supercomputing 1996*, November 1996. Pittsburgh, PA.

[7] J. Christoffersen, M. M. Mehrabadl, and S. Nemat-Nasser. A micromechanical description of granular material behavior. *Journal of Applied Mechanics*, 48:339–344, 1981.

[8] L. Clark, I. Glendinning, and R. Hempel. The MPI message passing interface standard. Technical report, 1994. Available at ftp://par.soton.ac.uk/pub/mpi/paper.ps.

[9] P. A. Cundall. Rational design of tunnel supports: A computer model for rock mass behavior using interactive graphics for the input and output of geometrical data. Technical Report MRD-2-74, Missouri River Division, Corps of Engineers, September 1974.

[10] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice, 2nd ed.* Addison-Wesley Publishing Company, 1990.

[11] K. Gaither, R. Moorhead, S. Nations, and D. Fox. Visualizing ocean circulation models through virtual environments. *IEEE Computer Graphics & Applications*, 17(1):16–19, Jan-Feb. 1997.

[12] S. Green. *Parallel Processing for Computer Graphics.* MIT Press, 1991.

[13] G. S. Grest, B. Dunweg, and K. Kremer. Vectorized link cell Fortran code for molecular dynamics simulations for a large number of particles. *Comp. Phys. Comm.*, 55:269–285, 1989.

[14] R. W. Hockney and J. W. Eastwood. *Computer Simulation using Particles.* Adam Hilger Publishing, 1988.

[15] R. W. Hockney, S. P. Goel, and J. W. Eastwood. Quiet high-resolution computer models of a plasma. *Journal of Computers in Physics*, 14:148–158, 1974.

[16] D. Horner, J. Peters, S. Howington, and R. Hryciw. Effects of grain size distribution representation on the physics of particulate materials. In *Proceedings of the Institute for Mechanics and Material Workshop on Mechanics and Statistical Physics of Particulate Material*, June 1994.

[17] M. Levoy. Volume rendering: Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(5), May 1988.

[18] X. Li and J. M. Moshell. Modeling soil: Realtime dynamic models for soil slippage and manipulation. In *Proceedings of SIGGRAPH '93.* Association of Computing Machinery, July 1993. Anaheim, CA.

[19] R. H. Miller. Experimenting with galaxies. *American Scientist*, pages 152–163, April 1992.

[20] J. J. Monagahan. Smoothed particle hydrodynamics. *Annual Review of Astronomy Astrophysics*, 30:543–574, 1992.

[21] J. M. Moshell, X. Li, and *et al.* Nap-of-earth flight and the realtime simulation of dynamic terrain. In *Proceedings of International Society for Optical Engineering*, April 1990.

[22] T. T. Ng and R. Dobry. A non-linear numerical model for soil mechanics. *International Journal for Numerical and Analytical Methods in Geomechanics*, 16:247–263, 1992.

[23] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computers in Physics*, 117:1–19, 1995.

[24] P. Sabella. A rendering algorithm for visualizing 3-D scalar data. *Computer Graphics*, 22(4):51–58, August 1988. Proceedings of SIGGRAPH '88.

[25] A. Shukla and M. H. Sadd. Wave propagation and dynamic load transfer due to explosive loading in heterogeneous granular media with microstructure. Technical Report Prepared Under Contract No. F49620-89-C-0091, U.S. Air Force Office of Scientific Research, August 1990.

[26] Silicon Graphics, Inc. *Graphics Library Programming Tools and Techniques*, 11 1991. Document Number 007-1489-010.

[27] L. Verlet. Computer experiments on classical fluids: I. thermodynamical properties of Lennard-Jones molecules. *Physics Review*, 159:98–103, 1967.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | July 1998 | Final report |

**4. TITLE AND SUBTITLE**
Interactive Computational Steering in Distributed, Heterogeneous High Performance Computing Environments

**5. FUNDING NUMBERS**

**6. AUTHOR(S)**
Alex R. Carrillo, John E. West

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
U.S. Army Engineer Waterways Experiment Station
3909 Halls Ferry Road, Vicksburg, MS 39180-6199

**8. PERFORMING ORGANIZATION REPORT NUMBER**
Technical Report ITL-98-2

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
U.S. Army Corps of Engineers
Washington, DC 20314-1000

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**
Available from National Technical Informance Service, 5285 Port Royal Road, Springfield, VA 22161.

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

The ability to visually interact with and guide a running simulation can greatly enhance understanding of both the nature of a numerical model and the characteristics of the underlying physical process. Until recently, interaction with large-scale simulations was inhibited not only by computational limitations, but by network and graphics capabilities as well. However, the increasing power of large high performance computing (HPC) systems, coupled with the advent of new network hardware and innovative techniques for displaying large data sets, has eliminated many of the barriers. It is now possible to link large computational resources with powerful remote graphics workstations to create fully interactive, distributed applications.

This report discusses the development of a distributed, interactive modeling system capable of providing responsive visual feedback to user input. The system is built around a discrete element soil model being used to simulate the behavior of soil masses under large deformations. During the process of designing this system to be interactively responsive to the user for systems of several hundred thousand particles, a range of technical problems were encountered, including the computational burden of the discrete element method (DEM), real-time visualization of large

(Continued)

**14. SUBJECT TERMS**
Discrete element method
Distributed visualization
High-performance computing
Interactive simulation
Ray casting
Soil dynamics

**15. NUMBER OF PAGES**
35

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | | |

13. (Concluded).

particle masses, and communication of data over large distances between heterogeneous resources using a variety of networks. This report highlights how each of those obstacles was addressed and discusses the effectiveness of the final system.