Early Experiences with Distributed Supercomputing on I-WAY: First Principles Materials Science and Parallel Acoustic Wave Propagation *

George A. Geist, II James A. Kohl Donald M. C. Nicholson
Philip M. Papadopoulos † Bart D. Semeraro William A. Shelton
Computer Science and Mathematics Division

G. Malcolm Stocks Yang Wang
Metals and Ceramics Division

Oak Ridge National Laboratory
Oak Ridge TN, 37831-6367

Abstract

In this paper, we present experiences relating to two simulations running on a dedicated network of supercomputers that are connected and scheduled with I-WAY. Each code uses PVM as the message passing layer so that the heterogeneity of the computers becomes transparent. The first application is a unique O(N) first principles materials science simulation that requires a large number of processors and memory to simulate materials properties. The algorithm scales well and efficiently overlaps computation with communication to tolerate large message latencies. The second application is a parallel acoustic wave propagation code that scales well, but requires message exchanges

*Research supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy, under contract DE-AC05-96OR22464 with Lockheed Martin Energy Research Corporation

†Author to whom correspondence should be addressed, phil@msr.epm.ornl.gov

at each iteration and therefore does not tolerate large message latencies. Both simulations produce on-line output which is sent to AVS using a custom library for collecting data from distributed applications. This library supports the dynamic attachment of several "viewers" to a single distributed application to facilitate on-line collaboration.

These computationally intensive codes require hundreds to thousands of processors to achieve acceptable performance. The I-WAY testbed provides access to the needed resources by allowing a researcher to use a virtual I-WAY machine (VIM), which is a collection of supercomputing, visualization, and network resources from around the country. We have slightly modified PVM to operate within the I-WAY scheduling and authentication mechanisms and present the user with a familiar interface to run existing programs on this new testbed. New challenges are presented when attempting to write efficient code for geographically distributed computers. For example, the message passing fabric is now heterogeneous with widely varying bandwidth and latency figures for the different network links. The applications presented here were successfully ported to the current I-WAY environment.

1 Introduction

Today, it is possible to have supercomputers from around the country working in concert on a single task. However, two important components of the computing infrastructure that make this practical have been missing, namely, a high-speed dedicated national network and software to simplify the task of submitting and running jobs across geographically distributed computers. I-WAY represents the beginnings of the next-generation high-performance computing network and a software environment that presents a uniform interface to the nation's supercomputing sites.

We present two applications that have been run successfully in the I-WAY environment and give suggestions on improving the interface and usability of I-WAY-connected machines. The applications are a first principles materials science code that determines the lowest-energy lattice configuration of a crystalline solid and gives the spin magnetization of each individual atom in the crystal. This simulation scales well and, for properly sized problems, can productively consume the CPU resources of thousands of processors. The second application is an acoustic wave simulation that is used for seismic analysis. This code simulates a seismic “thump” and propagates the sound wave through a known geological configuration. Seismograms are generated so that commercial seismic analysis codes can attempt to reconstruct the geology. This code also scales well but has a greater need for high-speed interconnects to maintain efficiency. Both simulations represent applications that are suitable for high-speed distributed computing.

Although raw performance is always of great interest, it becomes necessary to develop methods that monitor the progress of an ongoing calculation. This monitoring becomes even more difficult in the heterogeneous and distributed environment of I-WAY. We present a steering and visualization library, called CUMULVS (pronounced cumulus), that allows a user to interactively visualize fields of distributed data and steer parameters in the calculation. This on-line monitoring can yield a “movie” of the data as the calculation is progressing. The problem of monitoring the state of the virtual machine and associated network connections is not addressed in this paper, but is clearly a necessary diagnostic tool that should be part of a more mature I-WAY system.

This paper is organized as follows. Section 2 gives details and results for seismic simulation. Section 3 gives details and results for the materials science code. Section 4 overviews the library used for steering and visualization of the application. Section 5 explains some issues involved in making PVM work in the I-WAY environment and outlines some broad suggestions for making I-WAY a continuing success. Finally, Section 6 gives our conclusions.

2 Parallel Acoustic Wave Propagation

The seismic code simulates the propagation of an acoustic signal through a heterogeneous media by solving the scalar wave equation,

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} - \Delta u = f(x,t).$$

Here, $c(x)$ represents the local velocity of acoustic waves, $u(x,t)$ is the pressure field, and $f(x,t)$ is the source term. This simulation has been used to create a synthetic seismic dataset that will eventually be used to calibrate seismic analysis codes. The simulation is a finite difference approximation to the three-dimensional wave equation. Second-order centered differences are used to discretize the time terms. Tenth-order centered differences for the spatial term. Mesh spacing is uniform in all three dimensions. The computational mesh is regular and Cartesian.

The geological models used were an overthrust and salt dome model. The geology was specified via a spatial variation of the local acoustic wave speed c. The mesh spacing is 25 meters for the overthrust model and 20 meters for the salt model. The models represent large geological features and consist of several million grid points each. The salt model consists of approximately 83 million grid points and the overthrust model has close to 120 million grid points. The salt model is 13.5 km on a side and 4.2 km deep. The overthrust model is 20 km on a side and 4.65 km deep.

The synthetic seismic dataset project simulates a seismic survey on the computer. The survey is done by simulating many thousands of events called shots. Each shot consists of a signal generated at a particular point in the domain, the propagation of sound waves in the media, and the collection of time history data at an array of receivers in the domain. This is analogous to the physical case where data is collected in the field. The data for each shot is the acoustic pressure collected at thousands of receivers, hundreds of time a second. This results in a large amount of output data. This volume of data is then multiplied by the thousands of shots required to define a geology. The entire project represents too much computation to be done by one entity and is in fact a cooperative project involving several national labs and industrial partners.

This project could use Internet computing, similar to I-WAY, in several ways. The possibility of spreading the calculation over several machines would allow greater resolution in the models. The shot matrix could also be traversed more quickly and fully by taking advantage of shot level parallelism across many computers. We chose to explore the distributed computation aspect of the Internet calculation.

2.1 Parallel implementation

The parallel version of the code is based on a two-dimensional decomposition of the solution domain. The computational mesh is decomposed in both lateral directions. The vertical (depth) dimension is continuous and not distributed. The computational mesh has an equal number of points in each of the lateral directions and the processor mesh has an equal number of processors in each direction.

The kernel of the computation is the evaluation of the finite difference form of the wave equation,

$$u_{ijk}^{n+1} = 2u_{ijk}^{n} - u_{ijk}^{n-1} + c^2\delta^2(\Delta_{10}^{h}u^n),$$

where $\Delta_{10}^{h}u^n = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + O(h^{10})$ is the tenth-order approximation to the spatial derivative. The centered difference form of this term is a stencil with five points in each direction:

$$\frac{\partial^2 u}{\partial x^2} = \sum_{k=1}^{5} \beta_k \frac{u_{i+k} - 2u_i + u_{i-k}}{(kh)^2} + O(h^{10})$$

The evaluation of this term near processor boundaries requires a considerable amount of data from neighboring processors. The data overlap is 5 points deep. At each evaluation of the stencil, five times more data must be communicated than for a standard second order, three point stencil. Figure 1 shows the decomposition schematically. This figure also shows the dependency of the data on a processor. The central square of data is independent of the data on other processors. The stencil can be evaluated here without waiting for boundary data to be filled in. The dark strips along the sides of the domain are points which require data from neighboring processors to evaluate the stencil. The small corner areas require data from two neighbors to evaluate the stencil.

The numerical approximation uses absorbing boundary conditions on the sides and bottom of the domain and a free surface condition on the top. The sides and bottom also use an absorbing sponge layer boundary condition to improve the performance of the non-reflecting boundary condition used there. The sponge condition results in extra work being done on processors that have part of the domain that contacts the sides or bottom. This results in a load imbalance on the edge processors and results in a small performance degradation.

The calculation proceeds by evaluating the stencil at each of the points in the domain. The necessary communication is done via PVM. The next step of the calculation is to apply the sponge boundary condition. As mentioned above this phase results in a slight load imbalance. The third part of the calculation is the generation of the source signal at the appropriate point in the domain. The time step is tested periodically to determine if output of the receiver data is required. This overall process is repeated several thousand times per shot.

2.2 Results

In this section we present some results for a sample calculation. The domain is a 3-layer heterogeneous media. The domain has 20 meter spacing in each direction and extends 100 points in each direction. The propagation speeds are 2200, 3200, and 4200 meters per second in the top, middle, and bottom thirds of the domain respectively. Figures 2 – 6 show a wave propagating downwards from the shot point in the top center of the domain. The figures show the reflections from the interface and the distortion of the wave as it passes from one media to another.

Figure 2 shows the situation at 0.8 seconds into the calculation. The wave has propagated nearly to the interface of the first and second layer. Figure 3 shows the situation just after the wave enters the second layer, at 0.96 seconds. The wave is distorted and reflected at the interface.

Figure 4 shows the wave passing into the third layer. The wave is again reflected and distorted as it passes into the layer. The wave has also reached the side boundary of the domain with little reflection evident. The reflection from the interface of the first and second layers is nearing the top of the domain.

In Figure 5, the reflection from the first interface has reached the surface and been reflected. The reflection from the second interface has reached the first interface and has started to reflect and distort. The last Figure 6 shows the surface reflection and the third reflection from the first interface.

3 Materials Science

The development of new materials using only experimental laboratory techniques has become prohibitively expensive. Instead, the development cycle for new materials can be shortened with the appropriate use of computer simulations used in combination with experimental evidence to determine physical characteristics such as ductility. For atomistic

(figure)

Figure 1: processor mesh

(figure)

Figure 2: 0.8 sec

(figure)

Figure 3: 0.96 sec

(figure)

Figure 4: 1.28 sec

(figure)

Figure 5: 1.56 sec

(figure)

Figure 6: 1.68 sec

simulations to be useful in the discovery of new materials, codes should not use experimental parameters, be predictive, and generate parameters for input into other models that are used on much larger length scales (e.g. mesoscopic and continuum). These types of methods are referred to as first principles or *ab initio* meaning they are formulated with all the relevant physics and require no adjustable parameters.

The most computationally efficient *ab initio* methods are those based on the local density approximation (LDA) to density functional theory. These methods are used to treat the quantum-mechanical effects of electron exchange and correlation and result in a single-body formulation [9][10]. However, these methods exhibit rather poor $O(N^3)$ ($N =$ number of atoms) and are limited to systems of tens of atoms.

The introduction of massively parallel processors (MPP's) is providing scientists with the necessary resources to develop new theoretical and computational methods to simulate materials properties that were once thought to be intractable. The recent development of a parallel, first principles, LDA-based electronic structure algorithms that scale linearly with the number of atoms provides researchers with the necessary tools to accurately simulate materials comprised of 100's to 1000's of atoms. This new technique is a real-space multiple scattering, Green's function-based method termed large system multiple scattering (LSMS)[12]. In addition, a recent theoretical method has been developed that includes the effects of spin-dynamics within an LDA formalism [2]. The new method provides a description for calculating the torque on each moment due to all the other moments comprising the system so that the true magnetic structure can be determined. This is a change from previous techniques that have fixed the moment directions to be collinear. This advance allows for the investigation of the effects of both chemical disorder and spin dynamics on the magnetic properties of real materials.

In the next subsections, a brief description of the LSMS method will be given. Finally, the results of a 256 atom simulation of the chemically and magnetically disordered INVAR $Ni_{.35}Fe_{.65}$ alloy will be provided.

3.1 Large System Multiple Scattering

At the heart of the LSMS method is that, to a good approximation, the electron charge density on a particular atom at the i^{th} site, can be obtained by considering only the electronic multiple scattering process in a finite spatial region centered at that atom. The cluster of M atoms included in this region is referred to as the local interaction zone (LIZ) of the atom. Every atom in the system has its own LIZ and thus its own local electron densisty $\rho^i_M(\vec{r})$. The potential be used in the next self-consistent (SCF) iteration is obtained by solving Poisson's equation for the entire system where the electron charge density for the entire system is made up of a sum of the single-site densities:

$$\rho(\vec{r}) = \sum_{i=1}^{N} \rho^i_M(\vec{r})\sigma^i(\vec{r}) \tag{1}$$

where $\sigma^i(\vec{r})$ is the truncation function for the Voronoi polyhedron that confines atom i. This procedure is well suited to a parallel computing paradigm if one associates one or more atoms per node or task.

The calculation of the electronic structure is reduced to calculating the single particle Green's function for each of the atoms comprising the system from which $\rho^i_M(\vec{r})$ can be obtained in a standard way. For this, multiple scattering theory (MST) [11][8] is a particularly useful approach. In MST, one of the important quantities of interest is the scattering path matrix [13], which describes the scattering process of an electron beginning and ending at the i^{th} site and includes all possible scattering paths in real-space given by,

$$\underline{\tau}_M(\{\xi_i\};\epsilon) = [\underline{t}_M^{-1}(\{\xi_i\};\epsilon) - \underline{G}(i;,\epsilon)]^{-1} \tag{2}$$

where ξ_i denotes the configuration of the atoms within the LIZ of atom i, and ϵ is the one electron energy parameter. The real space structure constant $\underline{G}(i;,\epsilon)$ is composed of an $M \times M$ array of free-particle propagation sub-blocks $\underline{g}^{jk}(\epsilon)$ that connects sites j and k and $j,k = 1,2,....,M$. The t-matrix $\underline{t}_M(\{\xi_i\};\epsilon)$, has M nonzero diagonal sub-blocks, corresponding to a single-site scattering matrix $\underline{t}^j(\epsilon)$, where $j = 1,2,...,M$. Each atomic site has its own spin-space local frame of reference and the single-site quantities are diagonal in this frame. Because each site can have a different direction with respect to the global frame, the calculation of $\underline{\tau}_M(\{\xi_i\};\epsilon)$ for each site is performed in the global frame. Calculation of the single-site quantities requires the transformation $\underline{\tau}_M(\{\xi_i\};\epsilon)$ by each site back to its local frame of reference. The electron density $\rho^i_M(\vec{r})$ is given by,

$$\rho^i_M(\vec{r}) = \frac{1}{\pi}\mathrm{Im}\int_{-\infty}^{\epsilon_f} d\epsilon \left\{ \sum_{LL'} Z^i_L(\mathbf{r};\epsilon)\left[\tau^{ii}_{M,LL'}(\xi_i;\epsilon)\right] Z^i_{L'}(\mathbf{r};\epsilon) - \sum_L Z^i_L(\mathbf{r};\epsilon) J^i_L(\mathbf{r};\epsilon) \right\}, \quad (3)$$

where ϵ_f is the Fermi energy, and L represents a combination of angular and azimuthal quantum numbers, l and m. The functions $Z^i_L(\mathbf{r};\epsilon)$ and $J^i_L(\mathbf{r};\epsilon)$ are regular and irregular solutions of the single-site Schrödinger equation. This real space method is highly scalable on MPP's because each node can be assigned the calculation of the τ-matrix elements, the single-site quantities for the atom mapped onto it. Construction of the τ-matrix at each site requires that each node invert a matrix of dimension $O(M)$ and this implies an, overall scaling proportional to M^3N. Since M is independent of the system size N, the algorithm scales as $O(N)$.

A schematic representation of the mapping of the LSMS algorithm onto a MPP platform is displayed in Figure 7, where without loss of generality, we assume a one atom per node case. Consider the i^th atom of an N-atom system and assume periodic boundary conditions and that the LIZ of atom i is confined to its nearest neighbor shell (atoms j,k,l,m in the illustration). The LSMS algorithm proceeds as follows. An initial guess of the potential $v^i(\vec{r})$, and electron density $\rho^i_M(\vec{r})$, for the i^th site, and the positions of all atoms in the system, is loaded onto the i^th-node. The node determines which other nodes are within its LIZ. It then calculates the t-matrix, $\underline{t}^i(\epsilon)$, corresponding to its own potential and requests and receives the t-matrices of other atoms (nodes), (i.e., $\underline{t}^j(\epsilon)$, $\underline{t}^k(\epsilon)$, $\underline{t}^l(\epsilon)$, and $\underline{t}^m(\epsilon)$) within its LIZ. In addition it sends its own t-matrix to the nodes for which it is one of the atoms (nodes) in the remote atoms' LIZ. At this point, the ith-node has sufficient information to calculate the real space τ-matrix elements using Eq. (2). This process is repeated for each ϵ where typically 30-40 energy points are required per SCF iteration. The number of messages per ϵ per atom is equal to M. At each ϵ the contribution to $\rho^i_M(\vec{r})$ is calculated, making this essentially a local process. Since all nodes execute this procedure, the total electron density is now known (Eq. (1)), and ϵ_f and the potential for the next SCF iteration can be constructed. Calculating the Madelung potential requires knowledge of the boundary conditions of the large cell, as well as the electric multi-pole moments on the remote sites which have to be exchanged among nodes. The complete process is then repeated until charge self-consistency is achieved.

3.2 Results

We have performed a non-relativistic 256-atom simulation of the substitutionally disordered INVAR alloy $Ni_{.35}Fe_{.65}$ at a lattice constant, a, equal to 6.76 a.u. (atomic units). The 256-atom cell is constructed by layering an underlying face-centered-cubic (FCC) lattice which, corresponds to a 4x4x4 tiling in the case considered here. Substitutional disorder refers to the random occupation of the atomic sites on a fixed underlying lattice where the probability of occupation of a particular alloying component is equal to its concentration.

All angular momentum summations were truncated at ℓ_{max}=3 and the von-Barth Hedin method [3] was used to calculate the exchange-correlation potential and energy. We used a muffin-tin approximation which assumes the charge density and potential are spherically symmetric within the muffin-tin radius, r_{mt}, and equal to a constant outside r_{mt}. In addition, the muffin-tin spheres are allowed to touch but not overlap and this condition is used to determine the muffin-tin radius. The initial magnitude and direction of the moments were obtained from a previous LSMS calculation on this same system except that the directions of the moments were restricted to be parallel to the spin axis of quantization (z-axis in spin space). At the end of each SCF iteration the new direction of the moments is determined in the global frame of reference in spin space by calculating the torque on a moment at the i^{th} site due to all the other moments on the remaining sites.

The calculation provided in this article is preliminary and is performed for a single LIZ equal to 13 atoms (first nearest neighbor shell). Even for this size LIZ the number of iterations needed for convergence of the total energy, charge density, magnetization density, magnitude of the magnetic moment and its direction was over 200 iterations (several weeks of continuous calculations or several months of wall clock time). The condition for convergence is satisfied when the L_2 norm of the densities and moment direction between iterations is less than 10^{-6} and the difference of the total energy and the magnitude of the moments (which are scalars) between iterations is less than 10^{-6} for all atoms in the system. Once self-consistency is achieved the equilibrium magnetic structure is determined for this LIZ.

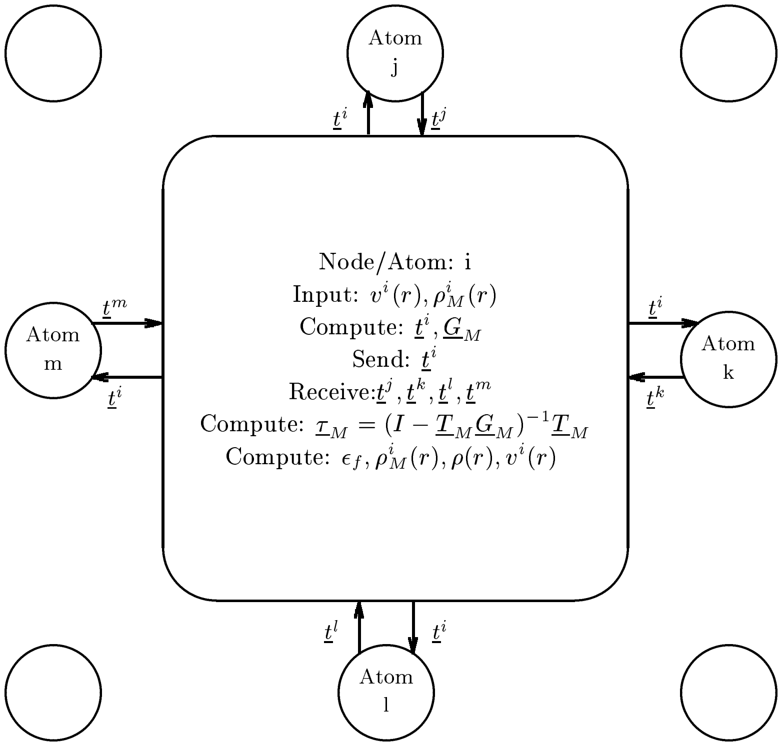

Figure 7: Schematic of a parallel SCF algorithm. A basic unit cell consists of N atoms. About each atom is defined a local interaction zone consisting of M atoms (labeled i, j, k, l, m in this example)

Figure 8 is a graphical representation of the directions of the Ni and Fe moments. Notice, that one of the directions is pointing anti-parallel with respect to the majority of other moments. We were able to determine which Fe site in the cell was responsible for this behavior. In addition, this site's first nearest-neighbor shell is composed of all Fe atoms. This local anti-ferromagnetic structure is exactly the magnetic structure of FCC Fe at this lattice constant. This suggests the magnetic correlations that give rise to the direction of the moments is short-ranged and may be mainly due to interactions with the first neighbor shell. However, these results are preliminary and we cannot draw any definitive conclusions.

In summary, this new approach of including spin-dynamics into the LSMS methods is providing researchers with the computational tools to investigate a variety of magnetic structures. In addition, I-WAY can provide the necessary computational resources to perform large scale simulations that would otherwise be infeasible on even a single large MPP.

4 CUMULVS Overview

Distributed applications need to be monitored at several different levels. During the debugging stage, for example, a programmer may want to view a program's use of message passing primitives and visually monitor how distributed data is being modified. In this section, we describe a "middle-ware" library called CUMULVS that enables steering and visualization of parallel programs. The library, in conjunction with a suitable front-end, allows the user to visualize distributed fields of data and to modify or "steer" program-defined parameters. The library is split into two pieces, a user library that is linked into existing parallel/distributed programs and viewer library that simplifies the creation of front-end viewer programs. We have successfully integrated our viewer library with AVS, Tcl/Tk, and tty-based viewer programs. CUMULVS is fundamentally dynamic. Viewers may attach and detach at anytime during a simulation, with multiple viewers looking at the same simulation at the same time. Viewers are potential serial bottlenecks when visualizing large data sets from a distributed application. To ease this problem, the user-library downsizes the dataset in parallel and sends only the desired information to each front-end viewer.

The CUMULVS library provides several important features for the computational scientist. It handles all of the details of collecting and transferring distributed data fields to the viewers, and oversees adjustments to steering parameters. The complete system also manages all aspects of the dynamic attachment and detachment of viewers to a running simulation. The following subsections overview the fundamentals and some implementation details of CUMULVS visualization and steering.

4.1 Visualization and Computational Steering

CUMULVS automatically handles the collection of visualization data from an application. Using a CUMULVS viewer, the user specifies a desired sub-region of a computational field to be visualized. The viewer then requests this region from the application via a request message that is handled by the user library routines. Each task in the distributed simulation need only identify its logical processor position in a data field decomposition, CUMULVS can then determine which data elements are present in each task. CUMULVS supports several data decompositions, including generic block-cyclic decompositions as described by HPF [1], and particle-type decompositions with user-defined accessor functions.

In addition to the sub-region boundaries, a sub-region request also includes a "cell size" for each dimension of the data. The cell size determines the stride of elements to be collected for that dimension, e.g., a cell size of 2 will obtain every other data element. This feature allows more efficient high-level overviews of larger regions by using only a sampling of the data points, while still providing the necessary details for smaller regions where every data point is required.

Once CUMULVS has collected the local task's data for the given sub-region, the data is sent to the viewer where it is automatically assembled into a coherent "frame" for animation. CUMULVS ensures that each viewer has a time-coherent view of the parallel data. An internal sequence number is maintained to determine where each application task is in the computation, and any attached viewers loosely synchronize with the application for each data frame they receive. The frequency of data frames can be set from the viewer, so the user can adjust how often frames are sent from the application, thereby reducing overhead and the loose synchronization effects.

Though CUMULVS's primary purpose is manipulating and collecting data from distributed or parallel applications, it is also useful with serial applications for the purpose of transferring data from the computation engine over a network to a visualization front-end.

CUMULVS supports coordinated computational steering of applications by multiple collaborators. The library guarantees that steering parameters remain coherent across any number of viewers. Consistency protocols are used to verify that all tasks in a distributed application apply the steering changes in unison. A locking scheme prevents conflicting adjustments to the same steering parameter by different users. Even if geographically separated, scientists can work together to direct the progress of a computation without concern for the consistency of steering parameters among the distributed tasks.

4.2 Implementation Issues

The CUMULVS library consists of approximately 20,000 lines of C code, and can be integrated into applications written in either C or Fortran. Existing programs require only slight modifications to describe how particular data fields have been decomposed and which parameters can be steered by a viewer. To get a flavor for the changes required to an existing program, Figure 9 shows the actual calls needed to define and send the pressure fields in the seismic simulation. The order is very simple: initialize the CUMULVS internals; define a data decomposition (handle returned in decompId); define a field with the described data decomposition (handle returned in fieldId); and, at the end of the computational loop, send fields to attached viewers. This additional call at the end of the logical computational loop directs when data can be sent to a viewing front-end. In the case where no viewers are attached to the running simulation, this single call results in a single message probe to check for requested connections.

The current CUMULVS system evolved from an earlier prototype system that linked a PVM application to AVS for floating point data visualization and some simple steering operations [7]. CUMULVS completely generalizes this early system, and now supports all standard primitive data types with built-in type conversion when desired. CUMULVS also provides a fault-tolerant communication protocol so that failures in either an application or a viewer can be gracefully handled.

The concepts of collecting data from an application and passing steering parameters to

Figure 8: Magnetic Spin Moments for all 256 atoms: The cluster of larger arrows corresponds to Fe atoms. Cluster of smaller arrows corresponds to Ni Atoms. Notice that most moments point in the same direction with a single Fe moment pointing anti-parallel.

```
C -   Initialize CUMULVS Structures
      call stvfinit( SEISMICGRP, STVTAG, nproc, inst, info)

C - Define the Decomposition type
      call stvfdecompdefine(3,axisType,axisInfo,axisInfo1,glb,gub,
     +                 prank,pshape,decompId)

C - Define Pressure field with above decomposition
      call stvffielddefine(u1,'pressure',decompId,
     +    declare, STVFLOAT, paddr, STVVISCP, fieldId)

C - Top of main Computational Loop
      do 100 npdt = tmp_npdt,ntps,2
                 .
                 .
                 .

C - Send visualization information to any attached front ends
        call stvfsendtofe(info)

100   continue
```

Figure 9: CUMULVS Library Calls made by the Seismic Simulation

an application are straightforward, but there are several underlying issues that add significantly to the complexity of the underlying system. The primary goal in the development was to keep the overhead and synchronization costs to the distributed simulation as low as possible. A variety of request, connection, and synchronization protocols have been developed to maintain consistency among the many distributed application tasks without introducing any deadlock conditions. These protocols had to be dynamic to allow viewers to attach at will, and yet had to be tolerant of faults and failures. Efficient general algorithms have been formulated for the packing and unpacking of data in different data decompositions – obtaining every "Nth" element within a sub-region becomes significantly more complicated when working with arbitrarily mixed block and cyclic decompositions. Finally, the viewer/application interfaces had to be generalized to support a variety of viewers with different data and synchronization requirements. The end result is a general system that automatically and efficiently handles all of these challenging details, with a minimal amount of user specification or effort.

5 PVM and I-WAY

Access to each I-WAY computing resource is controlled using the I-WAY scheduling system. The scheduler groups resources into Virtual I-WAY Machines (VIMs) and allows a user to schedule time on a named collection. Each supercomputing site is made up of the computing resource and an I-POP (I-WAY point-of-presence) machine. Local I-POPs interface with local schedulers and with remote I-POPs (Figure 10). The I-POPs allow a user to submit and run jobs in a uniform manner without regard to site-specific scheduling policies. I-POPs also take care of authenticating a user to remote sites. The I-WAY developers directly support MPI and Nexus applications. We worked with the I-WAY scheduler development team to modify PVM to operate within the authentication constraints. Furthermore, we allow both I-WAY and non-I-WAY machines to be configured into a single parallel virtual machine (PVM). This feature is critical to our applications where the computation is running on I-WAY resources but the visual monitoring is being run at the user's home institution which may not be a part of the I-WAY. The modification to PVM is managed in two parts.

First, the user selects a single I-POP as the master daemon of the parallel virtual machine. I-WAY hosts are added to the virtual machine using I-WAY-specific mechanisms like irsh, which is a variant of rsh that interfaces with the naming and scheduling mechanisms of I-WAY. Non-I-WAY hosts can be added to the virtual machine using standard PVM methods. Another modification was to change the default spawning algorithm in PVM. Instead of the usual round-robin assignment of tasks to all hosts in the virtual machine, tasks are assigned in a round-robin fashion to all hosts but the master I-POP. This means that the master daemon is used for little more than machine configuration and authentication. These simple modifications make it possible for users to build large virtual machines that are heterogeneous at the network and hardware levels.

5.1 Possible Directions for I-WAY Development

In this subsection, we present observations on what is needed to make I-WAY a continuing success. The basic I-WAY architecture is sound and successfully treads the line between centralized scheduling and local site control with a system that is useful to programmers and acceptable to resource providers. However, there are some architectural changes that would seem appropriate for the next generation I-WAY, as well as several interface/development/ease-of-use issues that need addressing. VIMs are currently determined by the scheduler and are statically allocated. Hence, only certain configurations of machines are available to the user. In addition to these static configurations, the scheduler should also support a mode where the user requests the types and quantity of needed resources. For example, a user may specify a VIM of 100 SP-2 Nodes, 500 Paragon nodes and a cave. The scheduler should return a time when the available configuration could be achieved for the user. This would essentially be a custom VIM. Next, the geographical distribution and heterogeneity of the computing sites makes the management of consistent software difficult for the user. User accounts on I-POP machines use AFS to provide a single file name space, but not all of the I-WAY resources support AFS and the user is forced to manually copy and build programs on each physical machine that makes up a VIM. An integrated interface that allows the user to distribute and compile his code for the different architectures is essential. There are several existing candidate systems that could be adopted by the I-WAY software team. Legion [6] and Pade [4] are just two such possible systems. In addition, the interface to the scheduler should be formalized so that other messaging systems could be ported to I-WAY and still maintain the desired level of I-WAY security. Finally, user-level network monitoring tools should be developed so that the sanity of long-haul connections can be checked independently from the user's communication layer.

6 Conclusions and Future Work

The idea of connecting a wide variety of geographically separated computers to solve very large problems has enticed researchers since the inception of network computing. The problem of relatively slow data transfer rates over the Internet has cooled the enthusiasm somewhat. Recent popular interest in the Internet as a national data super highway has made it likely that great increases in data transfer speed and volume are likely in the near future. Researchers are once again examining the possibility of using widely distributed network computing for solving large scientific problems.

In this work we have explored some aspects of distributed computing in the context of the I-WAY project. Two scientific applications were exercised on the I-WAY system. Each code stood to benefit differently from the distributed computing environment. The material science code could take advantage of several thousand nodes in one simulation. The seismic simulation had a maximum size with several hundred processors for a single run being optimal. Greater parallelism could be exploited by distributing over multiple virtual machines the several thousand runs needed to characterize a geology. The details of how both applications exploit parallelism were examined above. In addition, details of an interactive parallel visualization and steering tool were presented. This tool was used with the applications on the I-WAY system.

Our experience with I-WAY was not unusual for a project in its infancy. We had some difficulty with the I-WAY software being slow and not as smooth as we would have liked it to be. We were able to use the system in spite of this difficulty. Suggestions for improvement and enhancements to the system are made in the text. We remain enthusiastic about the prospect of this form of distributed computing in general and for the continuation of the I-WAY project in particular.

References

[1] *High Performance Fortran Language Specification, Version 1.1,* Rice University, Houston, TX, November, 1994.

[2] V. P. Antropov, M. I. Katsnelson, M. van Schilfgaarde, and B. N. Harmon, *Ab Initio Spin Dynamics in Magnets*, Phys. Rev. Let., 75 (1995), p. 729.

[3] U. von Barth and L. Hedin, *A Local Exchange-correlation Potential for the Spin-polarized Case: I*, J. Phys. C, 5 (1972), p. 1629.

[4] J.E. Devaney, R Lipman, M. Lo, W.F. Mitchell, M. Edwards, C.W. Clark, *The Parallel Applications Development Environment (PADE) User's Guide,* National Institute of Standards, 1995.

[5] G. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam, *PVM: Parallel Virtual Machine, A User's Guide and Tutorial for Networked Parallel Computing*, The MIT Press, 1994.

[6] A.S. Grimshaw, W.A. Wulf, J.C. French, A.C. Weaver, and P.F. Reynolds, Jr., *A Synopsis of the Legion Project,* University of Virginia, Technical Report No. CS-94-20, June, 1994.

[7] J. A. Kohl, P. M. Papadopoulos, *A Library for Visualization and Steering of Distributed Simulations using PVM and AVS,* High Performance Computing Symposium, Montreal, Canada, July 10-12, 1995.

[8] W. Kohn and N. Rostoker, *Solution of the Schrödinger Equation in Periodic Lattices with an Application to Metallic Lithium*, Phys. Rev., 94 (1954), p. 1111.

[9] P. Hohenberg and W. Kohn, *Inhomogeneous Electron Gas*, Phys. Rev., 136 (1964), p. 864.

[10] W. Kohn and L. J. Sham, *Self-consistent Equations Including Exchange and Correlation Effects*, Phys. Rev., 140 (1965), p. 1133.

[11] J. Korringa, *On the Calculation of the Energy of a Bloch Wave in a Metal*, Physica, 13 (1947), p. 392.

[12] Y. Wang, G. M. Stocks, W. A. Shelton, D. M. C. Nicholson, Z. Szotek and W. M. Temmerman, *An $O(N)$ Multiple Scattering Approach to Electronic Structure Calculations*, Phys. Rev. Let., 75 (1995), p. 2867.

[13] B. L. Györffy and M. J. Stott, in *Band Structure Spectroscopy of Metals and Alloys* , edited by D. J. Fabian and L. M. Watson (Academic, New York, 1973), p. 385

Figure 10: A Subset of I-WAY Sites. I-POPs are used for scheduling and configuration of VIMs.