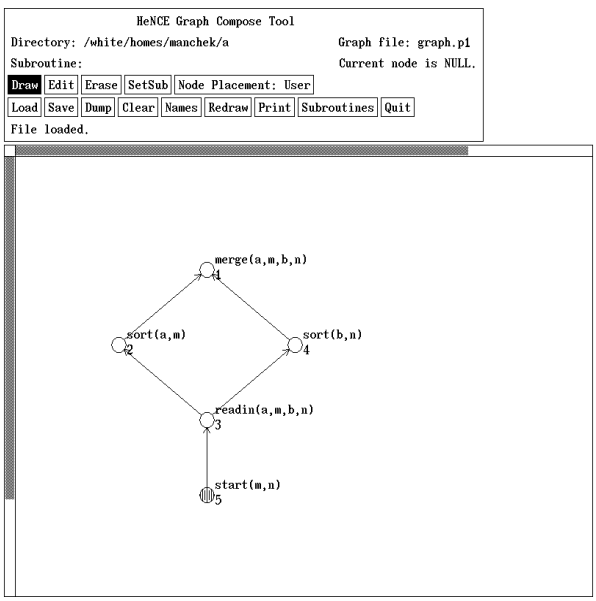

FIG. 2. *Drawing the graph using the HeNCE graphical interface.*

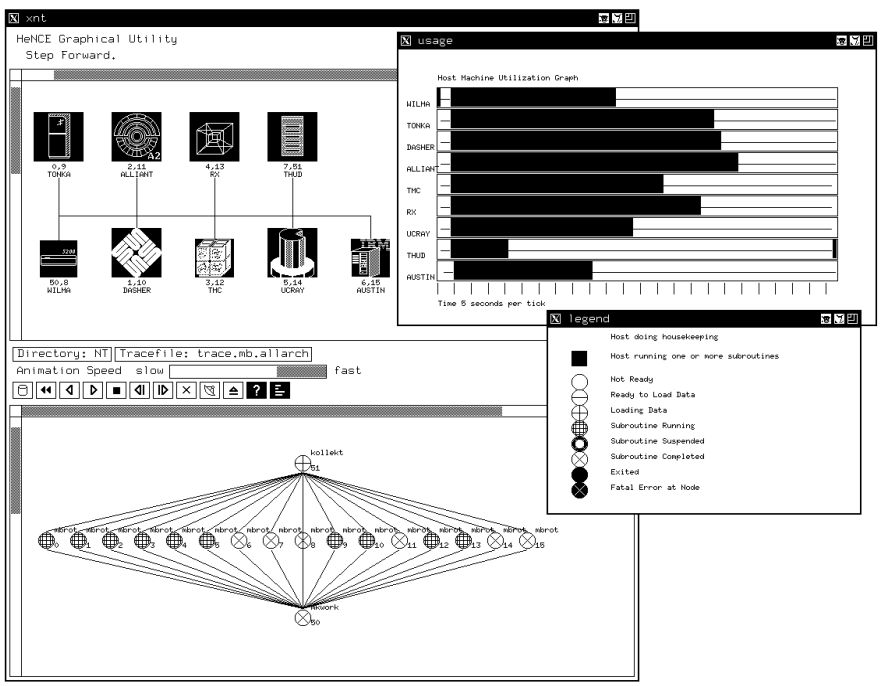

FIG. 3. *Trace mode of the HeNCE graphical interface.*

computers together to achieve the computational power necessary to solve Computational Grand Challenge Problems.

REFERENCES

[1] B. Ginatempo, J. B. Staunton, The electronic structure of disordered alloys containing heavy elements - an improved calculational method illustrated by a study of CuAg alloy. J. Phys. F Vol. 18, 1827 (1988).

[2] D. D. Johnson, D. M. Nicholson, F. J. Pinski, B. L. Györffy, G. M. Stocks, Total energy and pressure calculations for random substitutional alloys, Phys. Rev. B, Vol. 41, 9701 (1990).

[3] G. M. Stocks, W. M. Temmerman, B. L. Györffy Complete solution of the Korringa-Kohn-Rostoker coherent potential approximation: Cu-Ni alloys, Phys. Rev. Letter, Vol. 41, 339 (1978).

[4] Ulf von Barth Density Functional Theory for Solids, *Electronic structure of complex systems*, ed. Phariseau and Temmerman, NATO ASI Series, Plenum Press, (1984).

[5] SUN Network Programming Manual Part Two: Protocol Specification, (1988).

grals typically involves the independent evaluation of a function at different locations and the merging of these data into a final value. The two most obvious locations for parallelization in the KKR-CPA algorithm are in the integration over the Brillouin zone and the integration over energy. Each location was evaluated in terms of the available parallelism and the required communication overhead incurred by splitting the algorithm at that point. The Brillouin zone integration is the main step in each CPA iteration. The disadvantage of parallelizing the Brillouin zone integration is the large amount of communication volume that would be required. For this reason the integration over energy was parallelized. Typically, this integration involves the evaluation of the single site Green's function for between 200 and 1000 energies in order to determine the charge density for the next self-consistency iteration. Each of these tasks involves the iterative solution of the CPA equations for the given energy and requires significantly more computation than communication.

The parallel implementation is based on a master/slave paradigm to reduce memory requirements and synchronization overhead. In our implementation one processor is responsible for reading the main input file, which contains the number of processors to be used, the problem description, and the location of relavent data files. This master processor also manages the LDA-DFT charge self-consistency iteration. The slave processors require only enough memory to solve the CPA equations for a single energy, which presently requires 7 Mbytes. Memory reduction was a crucial consideration for porting the KKR-CPA code to the Intel iPSC/860 because the individual nodes only contain 8 Mbytes of RAM. If only one processor is requested, then a subroutine is called that calculates the tasks serially one after another. When more than one processor is requested, a pool-of-tasks scheme is used to accomplish dynamic load balancing. In this scheme, the tasks are arranged in a queue in approximate order of decreasing difficulty and assigned to idle slave processors as they become available. Thus, all processors are busy as long as there are tasks in the queue.

Additional modifications were required to allow this KKR-CPA code to execute on multiple heterogeneous supercomputers. On a Cray YMP two options are now available in the code. The user can specify a number of tasks to initiate simultaneously on the Cray multiprocessor. Ideally one per processor. A second option is to set a Cray multitasking switch in the input file and then initiate just one multitasking task. So far we have found the first option to be satisfactory and simpliest. To use an Intel/860 we had to write an intermediate routine that runs on the Intel host and performs three functions. It allocates a number of node processors as specified by the master process. It receives messages from other PVM processes, converts these messages to Intel format, then passes the messages to the appropriate nodes (and visa versa). Finally, it releases the nodes when the problem is finished. Because of the size of the KKR-CPA code (16000 lines of Fortran 66), we have not ported the code to run on Thinking Machines CM2, which requires applications to be written in Fortran 90.

5. Results. So far the results on a network of heterogeneous supercompters have been to show the viability of using such a system. Test programs consisting of simple tasks like solving the Mandelbrot problem have been run on a DECStation, Cray XMP, iPSC/860, CM2 configuration.

The KKR-CPA code was initially run on a network of IBM RS/6000's. Using six model 530's and four model 320's an execution rate of 207 Mflops was measured. This is comparable to running it on a single processor of a Cray YMP.

The KKR-CPA code was also run on a virtual machine consisting of an RS/6000, a Cray XMP, and 16 nodes of the iPSC/860. Performance was not measured during these runs because the Cray had many users on it and because only a few of the iPSC/860 nodes were used. What these runs do show is the viability of hooking several Cray and Intel super-

the parallelism of a computation and to aid the programmer as much as possible in the tasks of designing, compiling, executing, debugging, and analyzing the parallel computation. Central to HeNCE is a graphical interface which the programmer uses to perform these tasks. This tool supports visual representations of many of its functions.

In HeNCE, the programmer is responsible for explicitly specifying parallelism by drawing graphs that express the parallelism. The user directly inputs the graph using tools which are part of the HeNCE environment. Each node in a HeNCE graph represents a procedure written in either Fortran or C. There are three types of arcs in the HeNCE graph. One represents dependency; a second represents looping; and the third represents piplined sections. An arc from one node to another represents the fact that the tail node of the arc must run before the node at the head of the arc. Figure 2 shows an example graph-drawing session using HeNCE.

Once the graph is complete, HeNCE will automatically write the parallel program including all the communication and sychronization routines using PVM calls. HeNCE tools exist to assist the user in compiling this program for a heterogeneous environment.

Graphical tools in HeNCE allow the user to dynamically configure a parallel collection of machines into a parallel virtual computer. During execution, HeNCE automatically maps procedures to the machines in the heterogeneous network based on a user defined cost matrix. HeNCE also collects trace and scheduling information which can be displayed in real time or saved to be replayed later. Figure 3 shows a snapshot of this animated tracing. The lower window displays the graph, and the graph nodes change color depending on what activities are being performed in the particular procedures. For example, sending messages, awaiting messages, or executing. The upper window displays the heterogeneous network, and again the color of the machine-icons reflect the activities on the various machines. Node numbers are displayed under each icon so the user can see which tasks have been assigned to which computer. Also shown in Figure 3 is a utilization graph that displays the busy and idle times of each of the machines in the parallel virtual computer.

4. Parallelizing a Grand Challenge. During the last few years, ORNL material scientists and their collaborators at the University of Cincinnati, SERC at Daresbury, and the University of Bristol have been developing an algorithm for studying the physical properties of complex substitutionally disordered materials. A few important examples of physical systems and situations in which substitutional disorder plays a critical role in determining material properties include: metallic alloys, high-temperature superconductors, magnetic phase transitions, and metal/insulator transitions. The algorithm being developed is an implementation of the Korringa, Kohn and Rostoker coherent potential approximation (KKR-CPA) method for calculating the electronic properties, energetics and other ground state properties of substitutionally disordered alloys [3]. The implementation allows the treatment of materials having complex atomic lattices and having any number of disordered sublattices. The algorithm also solves the underlying quantum mechanical equations within a fully relativistic framework [1]. This is necessary for materials containing heavy elements such as barium in which even the outer, bonding, electrons are travelling at a significant fraction of the velocity of light.

The KKR-CPA method extends the usual implementation of local density approximation to density functional theory (LDA-DFT) [4] to substitutionally disordered materials [2]. In this sense it is a completely first principles theory of the properties of substitutionally disordered materials requiring only the atomic numbers of the species making up the solid as input.

The KKR-CPA algorithm contains several locations were parallelism can be exploited. These locations correspond to integrations in the KKR-CPA algorithm. Evaluating inte-

large level of granularity. During execution, multiple *instances* of each component may be initiated. Figure 1 depicts a simplified architectural overview of the PVM system.

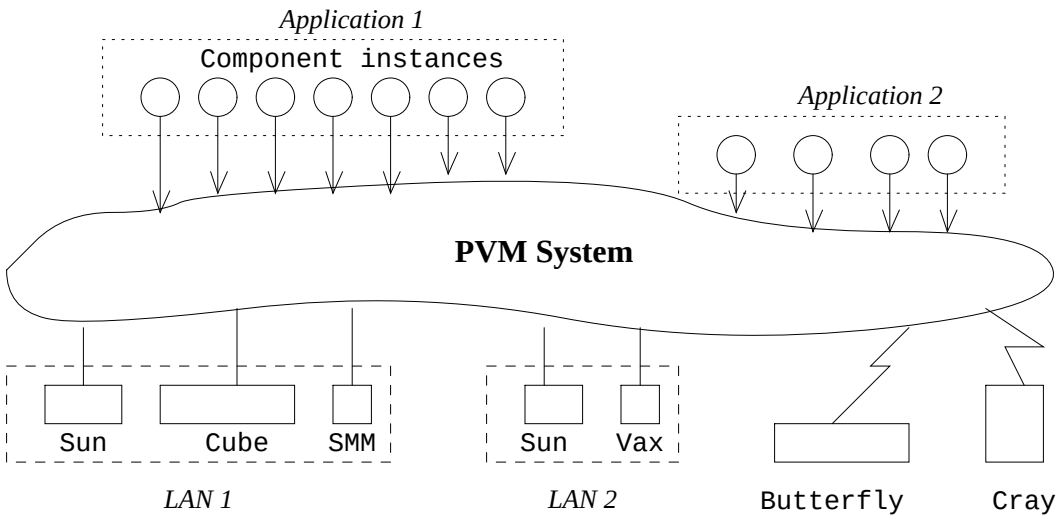

FIG. 1. *PVM Architecture Model*

Application programs view PVM as a general and flexible parallel computing resource that supports a message passing model of computation. This resource may be accessed at three different levels: the *transparent* mode in which component instances are automatically located at the most appropriate sites, the *architecture-dependent* mode in which the user may indicate specific architectures on which particular components are to execute, and the *low-level* mode in which a particular machine may be specified. Such layering permits flexibility while retaining the ability to exploit particular strengths of individual machines on the network. The PVM user interface is strongly typed; support for operating in a heterogeneous environment is provided in the form of special constructs that selectively perform machine-dependent data conversions where necessary. All communication done between PVM processes uses the external data representation standard (XDR) [5]. Inter-instance communication constructs include those for the exchange of data structures as well as high-level primitives such as broadcast, barrier synchronization, mutual exclusion, and rendezvous.

Application programs under PVM may possess arbitrary control and dependency structures. In other words, at any point in the execution of a concurrent application, the processes in existence may have arbitrary relationships between each other and, further, any process may communicate and/or synchronize with any other. This is the most unstructured form of crowd computation, but in practice a significant number of concurrent applications are more structured. Two typical structures are the tree and the "regular crowd" structure. We use the latter term to denote crowd computations in which each process is identical; frequently such applications also exhibit regular communication and synchronization patterns. Any specific control and dependency structure may be implemented under the PVM system by appropriate use of PVM constructs and host language control flow statements.

PVM is available through netlib. To obtain a copy, send mail to `netlib@ornl.gov` with the message: `send index from pvm`.

3. HeNCE. While PVM provides low-level tools for implementing parallel programs, HeNCE provides the programmer with a higher level abstraction for specifying parallelism. The HeNCE philosophy of parallel programming is to have the programmer explicitly specify

SOLVING COMPUTATIONAL GRAND CHALLENGES USING A NETWORK OF HETEROGENEOUS SUPERCOMPUTERS*

ADAM BEGUELIN, JACK DONGARRA
OAK RIDGE NATIOANL LABORATORY AND UNIVERSITY OF TENNESSEE

AL GEIST
OAK RIDGE NATIONAL LABORATORY

BOB MANCHEK
UNIVERSITY OF TENNESSEE

VAIDY SUNDERAM
EMORY UNIVERSITY

Abstract. This paper describes experiments connecting a Cray XMP, Intel iPSC/860, and Thinking Machines CM2 together over a high speed network to form a much larger virtual computer. The purpose of the experiments is to demonstrate the power and flexibility of the PVM (Parallel Virtual Machine) system to allow programmers to exploit a diverse collection of the most powerful computers available to solve Grand Challenge problems.

1. Introduction. Wide area computer networks have become a basic part of today's computing infrastructure. These networks connect a variety of machines, representing an enormous computational resource. In this paper we describe two software packages we are developing to facilitate the use of a network of heterogeneous computers. The first, PVM (Parallel Virtual Machine) allows utilization of the network of machines as a single computational resource. The second, HeNCE (Heterogeneous Network Computing Environment), is a graphical tool to assist the user in writing and analyzing parallel programs. We describe the parallelization of a large materials science application code including modifications required to run on heterogeneous supercomputers. Finally we present results from our initial tests of this software.

2. PVM. PVM is a software package that enables concurrent computing on loosely coupled networks of processing elements. PVM may be implemented on a hardware base consisting of different machine architectures, including single CPU systems, vector machines, and multiprocessors. These computing elements may be interconnected by one or more networks, which may themselves be different (e.g. Ethernet, the Internet, and fiber optic networks). These computing elements are accessed by applications via a library of standard interface routines. These routines allow the initiation and termination of processes across the network as well as communication and synchronization between processes.

Application programs are composed of *components* that are subtasks at a moderately

* This research was supported by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy, under contract DE-AC05-84OR21400 and the National Science Foundation's Science and Technology Center Cooperative Agreement No. CCR-8809615.