CONF-9104158--6 8--6

DE91 016276

# The PVM System: Supercomputer Level Concurrent Computation on a Heterogeneous Network of Workstations

G. A. Geist
Mathematical Sciences Section
Engineering Physics and Mathematics
Post Office Box 2008, Building 6012
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6367

V. S. Sunderam
Department of Math and Computer Science
Emory University
Atlanta, GA 30322



*Research performed at the Mathematical Sciences Section of Oak Ridge National Laboratory under the auspices of the Faculty Research Participation Program of Oak Ridge Associated Universities, and supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract DE-AC05-840R21400 with Martin Marietta Energy Systems, Inc.

# The PVM System: Supercomputer Level Concurrent Computation on a Heterogeneous Network of Workstations

G. A. Geist

Mathematical Sciences
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6367

V. S. Sunderam

Department of Math and Computer Science
Emory University
Atlanta, GA 30322

**Abstract**

*The PVM (Parallel Virtual Machine) sytem enables supercomputer level concurrent computations to be performed on interconnected networks of heterogeneous computer systems. Specifically, a network of 13 IBM RS/6000 powerstations has been used to run superconductor modeling codes at more than 250 Mflops. This paper describes the PVM system and two example applications running on it.*

## 1 Introduction

Wide area computer networks have become a basic part of today's computing infrastructure. High speed networks connect a variety of machines representing an enormous computational resource. A team of researchers from Oak Ridge National Laboratory, the University of Tennessee, and Emory University have developed the PVM (Parallel Virtual Machine) system and programming environment to exploit the aggregate computing resources of such a heterogeneous network [1].

PVM is designed from the ground up with heterogeneity and portability as primary goals. As such it is one of the first software systems that allows machines with very different architectures and floating point representations to work together on a single computational task.

PVM enables applications written in either C or Fortran to spawn off and kill subtasks on other machines making up the virtual computer. PVM also enables the subtasks to syncronize and to send and receive data between themselves. The subtasks in turn can spawn off other subtasks. As such almost arbitrary control and dependency structures can exist between subtasks running under the PVM system.

A common method for using the PVM system is to link together several workstations during off hours to solve a computational problem that would normally be submitted to a mainframe. Initially, networks of Sun3 and Sun4 workstations were commonly used. With the introduction of the IBM RS/6000 workstation, whose floating point performance is nearly an order of magnitude higher than a Sun3, mainframe performance was now possible from a single workstation. The IBM RS/6000 workstations, in conjunction with high speed networks and the PVM software, has demonstrated that applications can attain supercomputer level performance in such environments.

The next section describes the basic features of the PVM system. To demonstrate the competitiveness of PVM, several real applications have been ported to this environment. Section 3 will describe two example applications that have achieved high performance rates. These are an electronic structures application and a modecular dynamics application run on a network of workstations.

## 2 PVM

The PVM system is composed of a suite of user-interface primitives and supporting software that together enable concurrent computing on a loosely coupled network of processing elements. These processing elements may be serial computers, vector computers, or multiprocessors. Figure 1 shows an architectural overview of PVM. The present version of the software has been tested with various combinations of Sun3, SPARCstation, DECstation, IBM RS/6000, Silicon Graphics IRIS, Sequent Symmetry, Alliant FX/8, Intel iPSC/2, Intel iPSC/860, Thinking Machines CM2, and Cray YMP computers. In addition, users can port PVM to new architectures by simply modifying a generic *makefile* supplied with the source and recompiling.

Besides heterogeneity and portability, PVM has several other distinuishing features. The PVM package is small (less than 300 Kbytes of C source code) and easy to install. It needs to be installed only once on each machine to be accessible to all users. Moreover, the installation does not require special privileges on any of the machines. Using PVM, each user can configure his or her own parallel virtual computer, which can overlap with other users' virtual computers. Several different network architectures can coexist in PVM. For example, Ethernet, a fiber optic network, and Internet can all be a part of a user's parallel virtual computer. In addition, several applications can run simultaneously on a single parallel virtual computer.

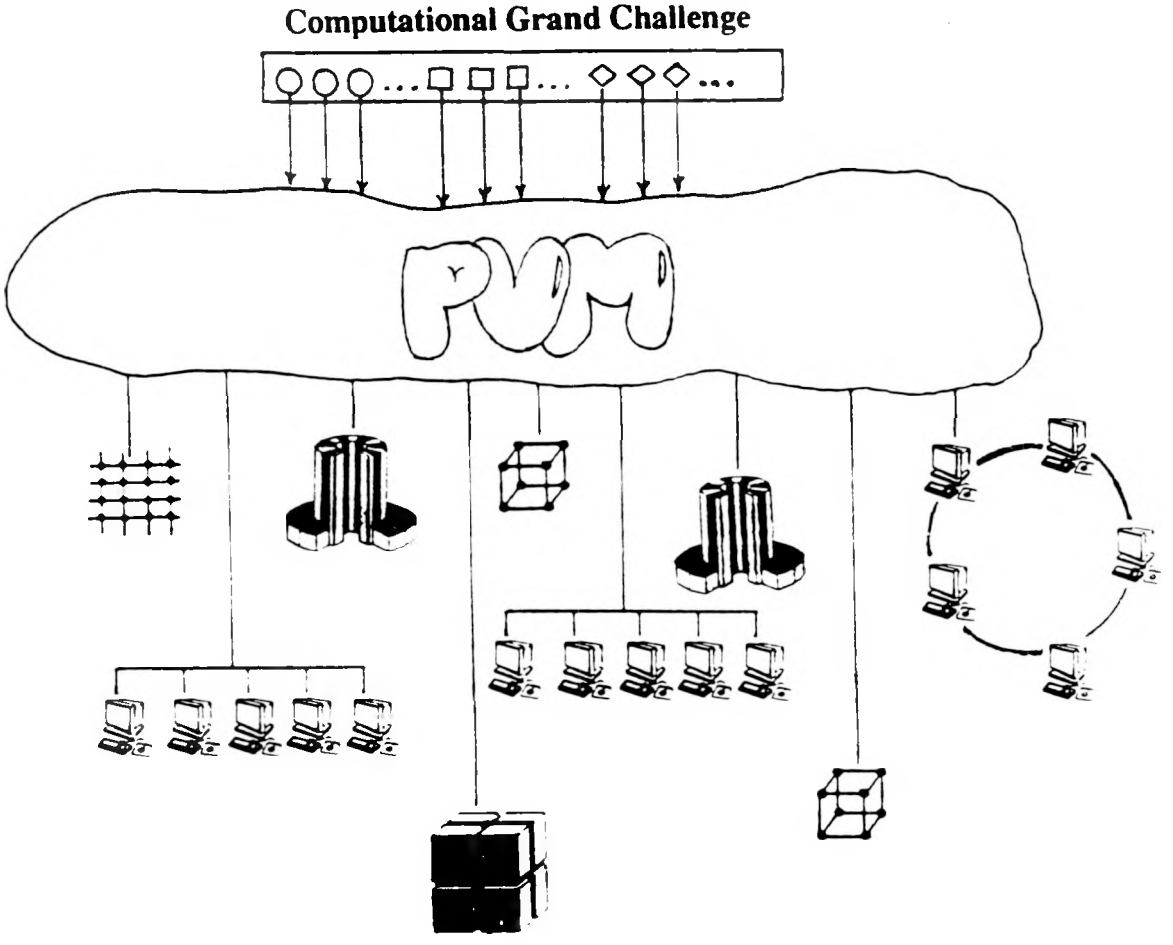


Figure 1: Architectural overview of PVM.

The subtasks of an application can be initiated on specific machines in the user's parallel virtual computer or the user can specify that particular subtasks be executed on a particular architecture. If the user does not specify anything, then PVM chooses an appropriate machine in the present virtual computer to initiate a subtask.

Application programs can be developed in C or Fortran using the message passing paradigm. In the message passing paradigm, processes communicate with each other by explicitly sending and receiving messages. The language extentions that PVM provides to manage the heterogeneous network are straightforward to use. Typically, programs that have already been developed for hypercube multiprocessors can be ported to PVM in less than a day.

The user views PVM as a distributed memory computer programmed in C or Fortran with message passing extentions. The PVM user-interface requires that all message data be strongly typed. Support for operating in a heterogeneous environment is provided by routines that selectively perform machine-dependent data conversions. All communication between PVM processes uses the external data representation standard (XDR), thus allowing machines with different integer and floating point representations to pass data. Other routines in the user interface allow initiation and termination of processes across the network as well as communication and synchronization between processes.

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. In practice, concurrent applica-

tions are usually more structured. Two typical structures are the *tree* and the *regular crowd* structures. The latter term is used to denote computations in which each process is identical and exhibits regular communication and synchronization patterns. For example, regular crowd structures are commonly used in the parallel solution of 2-D and 3-D partial differential equations.

## 3 Applications

The PVM system has been used for a variety of application codes on different networks, each with a unique mix of processing elements. Example applications that have been executed under PVM include Cholesky factorization, stochastic simulation of toroidal networks, statistical modeling, electronic structures calculations for disordered materials, and classical molecular dynamics calculations.

The electronic structures code is an implementation of the Korringa, Kohn, and Rostocker coherent potential approximation (KKR-CPA) method for calculating the properties of substitutionally disordered alloys [3]. The KKR-CPA method 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. Our implementation consists of 19,000 lines of Fortran 77.

The implementation has a low communication to computation ratio due to the parallelization scheme employed. The parallel implementation also includes dynamic load balancing. Several materials have been studied with this code including: NiAl, which is a new high strength alloy, MnO, which is a transition metal oxide that is close to the metal/insulator transition, and the perovskite superconductor $(Ba_6K_4)BiO_3$. The computational rates shown in Table 1 were measured during calculations of the electronic structure of this high temperature superconductor. In the 13 RS/6000 experiment, which executed at 261 Mflops, 4 model 320's and 7 model 530's were physically on the same Ethernet, while the remaining 530 and 550 were geographically distant and accessed via a T1 link.

| Model 320 | | Model 530 | |
|---|---|---|---|
| nproc | Mflops | nproc | Mflops |
| serial | 18.2 | serial | 24.4 |
| 2 | 31.3 | 2 | 45.9 |
| 4 | 63.1 | 4 | 92.2 |
| | | 7 | 161.9 |
| 6(530's) + 4(320's) | | | 206.5 |
| 7(530's) + 4(320's) | | | 226.0 |
| 1(550) + 8(530's) + 4(320's) | | | 261.0 |

Table 1: Performance of KKR-CPA code on several virtual computers.

Another scientific computing application in which very high levels of performance have been achieved using PVM on a network of workstations is a classical molecular dynamics problem. Molecular dynamics is used to calculate the dynamic properties of liquid and solid state systems [2]. Our particular implementation treats each of the $N$ atoms (or molecules) as a point mass, and Newton's equations are integrated to move each atom forward in time. Atoms are allowed to diffuse, i.e. each atom's neighbors change as the simulation progresses and only short range forces are considered significant.

The parallel algorithm assigns a fixed region of space to each processor, which updates the positions of all atoms within its box in a given timestep. Atom velocities and force values are exchanged with nearest neighbors between each time step. The results of the molecular dynamics application for a range of processors and problem sizes are given in Table 2.

| Molecular Dynamics Simulation | | | |
|---|---|---|---|
| PVM | Problem Size | | |
| RS/6000 procs | 5X5X5 | 8X8X8 | 12X12X12 |
| 1 | 23 | 146 | 1030 |
| 2 | 15 | 91 | 622 |
| 4 | 12 | 62 | 340 |
| 8 | 6 | 34 | 184 |
| iPSC/860 procs | | | |
| 1 | 42 | 202 | 992 |
| 2 | 22 | 102 | 500 |
| 4 | 11 | 52 | 252 |
| 8 | 6 | 27 | 129 |

Table 2: Times in seconds for MD simulations on PVM and iPSC/860

The table compares the execution times of PVM using a network of IBM RS/6000's and the iPSC/860 hypercube. For small numbers of processors, PVM using Ethernet is quite competitive with the hypercube with dedicated channels. Load imbalances became worse on PVM as workstations with different computational rates were added to the virtual computer. Nevertheless, it is encouraging to note that the PVM system performs well for this application that has a high communication to computation ratio.

## 4 Current Status and Availability

PVM was originally developed at Oat Ridge National Laboratory two years ago and was made publically available in March of this year. The current version, Version 2.2, is available through *netlib*. For details on how to obtain the PVM User's Guide or the source code, send e-mail to `netlib@ornl.gov` with the message: `send index from pvm`.

A graphical interface called HeNCE is being developed on top of PVM. A prototype version of HeNCE is expected to be available by the end of summer.

## References

[1] G. A. Geist and V. S. Sunderam, "Network Based Concurrent Computing on the PVM Sysyem,"

Oak Ridge National Laboratory Tech. Report (ORNL/TM-11760), January 1991.

[2] S. J. Plimpton, "Molecular Dynamics Simulations of Short-Range Force Systems on 1024-node Hypercubes," *Proc. Fifth Distributed Memory Computing Conference*, ed. D. Walker and Q. Stout, IEEE Computer Society Press, pp.478-483, 1990.

[3] G. M. Stocks *et. al.*, "Complete Solution of the Korringa-Kohn-Rostoker Coherent Potential Approximation: Cu-Ni Alloys," *Phys. Rev. Letters*, Vol. 41, 339, 1978.