PVM: Experiences, Current Status and Future Direction *

Adam Beguelin §, Jack Dongarra† ‡, Al Geist †,
Robert Manchek‡, Steve Otto¶, and Jon Walpole¶

‡Oak Ridge National Laboratory
Mathematical Sciences Section
P. O. Box 2008, Bldg. 6012
Oak Ridge, TN 37831-6367

†University of Tennessee
Department of Computer Science
107 Ayres Hall
Knoxville, TN 37996-1301

¶Oregon Graduate Institute
Computer Science Department
P. O. Box 9100
Portland, OR 97291-1000

§Carnegie Mellon University
School of Computer Science
5000 Forbes Avenue
Pittsburgh, PA 15213-3890

The computing requirements of many current and future applications, ranging from scientific computational problems in the material and physical sciences, to simulation, engineering design, and circuit analysis, are best served by concurrent processing. While hardware multiprocessors can frequently address the computational requirements of these high-performance applications, there are a number of integration aspects to concurrent computing that are not adequately addressed when conventional parallel processors are used to solve these problems.

The PVM (Parallel Virtual Machine) software package provides the software infrastructure for programming heterogeneous networks [4, 1, 3]. PVM provides mechanisms for configuring a virtual machine on a network, initializing processes on this network and communicating among these processes. PVM is a lightweight package intended for user installation. Nearly any Unix or Unix-like machine can be used as a processor in a virtual machine as long as the user has an account on the machine and it is accessible over a network. Several existing concurrent applications have been ported to execute on the PVM system, with encouraging results. For example, PVM execution speeds for molecular dynamics simulations (an application with a high volume of communication) using IBM RS/6000 workstations averaged only 30 percent slower than an iPSC/860 hypercube with a comparable number of processors. Another application calculates the electronic structure of a high-temperature superconductor at a rate of approximately 250 Mflops under the PVM system.

Applications that use a combination of shared memory machines, hypercubes, and scalar processors have been run under the PVM system, with corresponding increases in performance over any one multiprocessor. Preliminary experiments have shown promise in this type of computing environment. PVM virtualizes such a heterogeneous collection of computers into a distributed-memory, message-passing, parallel computer. PVM is becoming a spanning software technology for network computers, cluster computers, and tightly-coupled massively-parallel processors (MPPs), and significantly, vendors, including IBM, Convex, and Cray, are implementing versions of PVM on their platforms.

We are working to enhance significantly the functionality of PVM. Extensions to PVM along a number of dimensions are proposed. First, we are planning to add support for parallel and distributed processing in multi-user and multi-owner environments. This support will include the ability to capture idle cycles available on shared networks, to sensibly schedule multiple parallel jobs, and to integrate MPPs with networked personal workstations. Secondly, we will add a collection of tools for debugging, profiling and monitoring concurrent applications. Source level debugging will be supported using existing debugging systems that will be adapted for concurrent environ-

*This project was supported in part by the Defense Advanced Research Projects Agency under contract DAAL03-91-C-0047, administered by the Army Research Office, the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract DE-AC05-84OR21400, and by the National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615.

ments [2]. Profiling will concentrate on visual animation and display of program behavior, with emphasis on inter-component communication and synchronization. In addition, we will also develop tools for monitoring overall system load, resource availability, and network traffic. These facilities will permit reconfiguration and relocation of application components, and will provide administrative interfaces for general resource management. Finally, we will extend the PVM message passing system to support communication of multimedia data, including digital audio and video. The idea here is to give a simple and very portable interface for multimedia applications.

This new version of PVM will form the "kernel" of a Concurrent Processing Environment (CPE). Rather than build all the above capabilities directly into the kernel, we will take a modular approach. Interfaces are defined that allow the modules to be used independently in other environments, and allow the environment to coexist with related software systems. A distributed scheduler, for example, will be a separate module with a well-defined interface to the CPE kernel. This "open-systems" approach addresses heterogeneity at the system software level as well as at the operating system and hardware levels.

By hiding the complexity of concurrent processing systems and by presenting a uniform programming model we will encourage the development of parallel applications and system services. The CPE will also provide a runtime environment and target (virtual) machine for compilers such as HPF.

Despite our focus on heterogeneous multiuser networks, the project will also contribute to the process of integrating massively parallel processors (MPPs) with more common-place workstation networks. A prominent trend in MPP design has been the move towards cluster-oriented architectures. Examples of such systems are the Intel Paragon and TMC CM5. These systems are conceptually similar to workstation networks in that they are multi-user, network-based architectures with an operating system running on each node. In fact, such systems constitute a simpler (because they are homogeneous and do not support such complex notions of ownership) subset of the architectures considered in the proposed research. Furthermore, MPP vendors are already interested in ports of PVM to their platforms. Consequently, we expect our environment to become a spanning technology that will assist in the seamless integration of MPPs into heterogeneous networks.

Future users of our system will be able to run batch and interactive parallel applications unobtrusively on multi-user heterogeneous networks. These applications will be able to transfer audio and video data as well as conventional data, will enjoy fault-tolerant execution, and will have access to external visualization and performance monitoring packages. The system will support automatic redistribution of work, through dynamic process migration, in order to cope with reclaimed and newly available processors. This dynamic reconfiguration will be transparent to application programmers, who will program in terms of virtual, rather than physical, processors. Application programmers will not need to worry about the degree of physical parallelism, the location of processors, process migration, resource allocation or scheduling. Hence, the programming model based on PVM message passing, will be kept simple.

Not only will the application programmer's interface to our system be simple and transparent, but the execution of the system will also be transparent to other users of the machines on the heterogeneous network. A key requirement of a cycle harvesting system is that it be unobtrusive to the users of the computers it is using. Our system will place high importance on this characteristic, if necessary trading efficiency for unobtrusiveness. In this way, we hope to gain access to many more processor cycles than would otherwise be possible.

Acknowledgements

We would like to thank Jim Patterson of BCS for helpful discussions.

References

[1] A. L. Beguelin, J. J. Dongarra, A. Geist, R. J. Manchek, and V. S. Sunderam. Heterogeneous network computing. In *Sixth SIAM Conference on Parallel Processing*, 1993.

[2] D. Cheng. Proposal for a standard debugger server protocol. Technical report. To appear at the SuperComputing '93 workshop, "Debuggers for High Performance Computers."

[3] Jack Dongarra, Al Geist, Robert Manchek, and Vaidy Sunderam. Integrated PVM framework supports heterogeneous network computing. *Computers in Physics*, April 1993.

[4] A. Geist and V. S. Sunderam. Network-based concurrent computing on the PVM system. *Concurrency: Practice and Experience*, 4(4):293–311, June 1992.