Using PVM 3.0 to Run Grand Challenge Applications on a Heterogeneous Network of Parallel Computers*

Jack Dongarra
Oak Ridge National Laboratory and University of Tennessee
Al Geist

Oak Ridge National Laboratory
Robert Manchek

University of Tennessee
Weicheng Jiang

Dartmouth University

December 23, 1992

Abstract. This paper describes some recent research on PVM (Parallel Virtual Machine). One of the new features added in PVM 3.0 is multiprocessor integration. This is the ability to run PVM applications on the nodes of several different distributed memory multiprocessors as though they constitute one large parallel computer. We describe how multiprocessor integration is accomplished in PVM 3.0 and illustrate its use with examples using some of the parallel computers in ORNL's Center for Computational Science.

Several computational Grand Challenge problems are being addressed at Oak Ridge National Laboratory. Two examples are the calculation of the electronic structure of solids from first principles and groundwater transport. We will report on the use of PVM in the solution of these problems.

1 Introduction

The face of scientific computing is changing. Today's infrastructure of networks and powerful workstations allows researchers to collaborate and compute remotely in an efficient manner. The latest trend in scientific computing is to exploit the aggregate power and memory of clusters of computers sitting on local area networks. This trend is being driven

*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.

by three factors: familiar development environment of local workstations, robustness and existing system support for the machines, and most importantly low cost. By utilizing unused cycles of existing hardware, supercomputer performance can be generated with no additional hardware expense.

One of the initial problems encountered in this computing environment is heterogeneity, which is getting machines with different architectures, operating systems, and data formats to cooperate. PVM (Parallel Virtual Machine) is one of the first software systems specifically designed to allow a heterogeneous collection of computers hooked together by a network to be used as a single large parallel computer [1]. PVM is an ongoing research project involving Vaidy Sunderam at Emory University, Al Geist at ORNL, Bob Manchek at the University of Tennessee (UT), Adam Beguelin at Carnegie Mellon University, and Jack Dongarra at ORNL and UT. It is a basic research effort aimed at studying and advancing the state of the art in heterogeneous distributed computing. Available since 1989, PVM now enjoys widespread use throughout the world in solving important scientific, industrial, and medical problems.

The first uses of PVM were to combine a set of workstations into a single computational resource. This later grew to include a mixture of parallel supercomputers and workstations. The next research step in heterogeneous distributed computing is to integrate the individual processors of a multiprocessor into a seamless environment with other computers on a network some of which may also be multiprocessors. The key distinction and difficulty of this research is that the individual processors often have limited operating system support and no access to the network.

In section 2 we describe how PVM allows multiple parallel computers to be logically combined into a single large system. In section 3 we present some preliminary experiments using PVM 3.0 to connect several parallel computers in ORNL's Center for Computational Science. In section 4 we describe the methods and amount of effort required to convert two Grand Challenge application codes to PVM.

2 PVM 3.0

PVM 3.0 is a software system that permits a heterogeneous collection of Unix computers hooked together by a network to be used as a single large parallel computer. Thus large computational problems can be solved by using the aggregate power and memory of many computers.

In many ways PVM 3.0 is vastly improved over the previous versions of PVM. The PVM core has been completely redesigned for improved performance, scalability and fault tolerance while maintaining PVM's high standards of portability and robustness. Several new features are available in PVM 3.0 including:

- multiprocessor integration,
- dynamic configuration,
- dynamic process groups,
- multiple message buffers,
- process signalling, and
- user definable receive contexts.

In this paper we will focus on only one of the new features, multiprocessor integration.

2.1 Multiprocessor Integration

PVM was originally developed to join machines connected by a network into a single logical machine. Distributed memory multiprocessors such as the Intel iPSC/860 often have only a host or a special node that is actually connected to the network. All the other processors can only communicate over the internal communication paths. To exploit such machines with PVM 2.4 it is necessary to write a PVM program for the *host* processor that receives messages coming from the network, and sends them to individual nodes using the vendor's message-passing routines. Similarly for messages coming from a node destined for a machine out on the network, it is necessary for an intermediate process to intercept the native messages, convert them to PVM format, and route them.

With PVM 3.0 the dependence on UNIX sockets and TCP/IP software is relaxed. For example, programs written in PVM 3.0 can run on a network of SUN's, on a group of nodes on an Intel Paragon, on multiple Paragons connected by a network, or a heterogeneous combination of multiprocessor computers distributed around the world. All this without having to write any vendor specific message-passing code. (See Figure 1) PVM 3.0 is designed so that native multiprocessor calls can be compiled into the source. This allows the optimized message-passing of a particular system to be realized by the PVM application. Messages between two nodes of a multiprocessor go directly between them while messages destined for a machine out on the network go to the user's single PVM daemon on the multiprocessor for further routing. On shared-memory systems the data movement can be implemented by memory copies or by passing of pointers and careful use of locks.

PVM 3.0 contains a reference port to the Intel iPSC/860. Integration into Intel's Paragon and Thinking Machine Corporation's CM-5 multiprocessors was not complete at the time of this writing because of the immaturity of these machine's Operating Systems. Cray, Convex, SGI, and IBM are also supplying PVM 3.0 compatibility with their respective multiprocessors when they become available. More multiprocessor machines will be added to subsequent PVM 3.x releases.

The key to efficient multiprocessor integration is the PVM task IDs. In PVM 3.0 all tasks are identified by a PVM supplied task ID (tid) that has the location of the task encoded in it. This 32 bit integer contains three fields. The first is the host address, the second is the CPU number (if the host is a multiprocessor), the third is the process ID on this cpu (for the case where the CPU is multitasking).

We will illustrate the use of the tid with two examples. In the first example assume a PVM task is running on a node of an Intel Paragon and calls the PVM send routine to send data to another PVM task that happens to be running on another node of the same Paragon. PVM compares the first field of the destination tid with the sender's tid and discovers that the message will stay within the same host. From the second and third fields of the destination tid a vendor optimized communication routine is called by PVM to move the message directly to the destination task. In the second example assume the PVM process now sends a message to a task that is running on another (possibly parallel) computer out on Internet. PVM determines the message will not stay within the host so it uses the vendor optimized communication routine to move the message to the PVM daemon process running on the service node. The service node of a Paragon has access to the network. The daemon routes the message to the daemon running on the destination host. (There is only one PVM daemon per user per Internet address). This remote daemon then delivers the message to the destination task using the second and third fields of the destination tid. In this way a PVM application can have tasks scattered across several heterogeneous parallel computers and still send messages between all the tasks as though they were all on one large parallel computer.

The PVM daemon that runs on a multiprocessor listens for requests and messages coming from nodes of the multiprocessor as well as the network. The efficiency that this can be implemented is very dependent on the underlying operating system and message-passing software.

When confined to using only workstations, a PVM application seldom uses more than 60 cpus. But given PVM 3.0's integration into multiprocessors, it is possible to join together just a few hosts with an aggregate of several thousand cpus. This is the type of power necessary to solve computational Grand Challenge problems in materials design and environmental cleanup.

3 Experiments

`[nothing to say here yet]`

4 PVM use in Grand Challenges at ORNL

ORNL material scientists and their collaborators 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: high-strength 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 KKR-CPA method extends the usual implementation of 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 as input only the atomic numbers of the species making up the solid.

The KKR-CPA algorithm contains several locations where parallelism can be exploited. These locations correspond to integrations in the KKR-CPA algorithm. Evaluating integrals typically involves the independent evaluation of a function at different locations and the merging of these data into a final value. The integration over energy was parallelized. The parallel implementation is based on a master/slave paradigm to reduce memory requirements and synchronization overhead. In the implementation one processor is responsible for reading the main input file, which contains the number of nodes to be used on each multiprocessor as well as the number and type of workstations to include, the problem description, and the location of relevant data files. This master processor also manages dynamic load balancing of the tasks through a simple pool-of-tasks scheme. The development of the parallel KKRCPA code using PVM required about two months of effort.

A number of mixed heterogeneous multiprocessor tests have been run using the KKRCPA code, but more important is its use as a research code to solve important materials science problems. Since its development the KKRCPA code has been used to compare the electronic structure of two high temperature superconductors, $Ba(Bi_{.3}Pb_{.7})O_3$ and $(Ba_{.6}K_{.4})BiO_3$, to explain anomalous experimental results from a high strength alloy, NiAl, and to study the effect of magnetic multilayers in CrV and CrMo alloys for their possible use in magnetic storage devices.

The goal of the groundwater modeling group is to develop state of the art parallel models for today's high performance parallel computers, which will enable researchers to model flow with higher resolution and greater accuracy than ever before. As a first step researchers at ORNL have developed a parallel 3-D finite element code called PFEM that models water flow through saturated-unsaturated media. PFEM solves the system of equations

$$F \frac{\partial h}{\partial t} = \nabla \cdot [K_s K_r (\nabla h + \nabla z)] + q,$$

where h is the pressure head, t is time, K_s is the saturated hydraulic conductivity tensor, K_r is the relative hydraulic conductivity or relative permeability, z is the potential head, q is the source/sink and F is the water capacity ($F = d\theta/dh$, with θ the moisture content) after neglecting the compressibility of the water and of the media.

Parallelization was accomplished by partitioning the physical domain and statically assigning subdomains to tasks. The present version uses only static load-balancing and relies on the user to define the partitioning. In each step of the solution the boundary region of each subdomain is exchanged with its neighboring regions.

Originally developed on an Intel iPSC/860 multiprocessor, a PVM version has now been developed which opens the possibilities of running across several multiprocessors. Presently, the PVM version of PFEM has been delivered to several members of the groundwater modeling group for testing using networks of workstations while they await the availabilitiy of parallel supercomputers.

These experiments are the initial steps in utilizing several parallel supercomputers configured into a virtual super-parallel computer in order to achieve the computational power necessary to solve Computational Grand Challenge Problems.

The PVM software and Users Guide are available on netlib. To find out how to obtain this material send e-mail to `netlib@ornl.gov` with the message: `send index from pvm`. An automatic mail handler will reply with detailed directions.

References

[1] A. Geist, A. Beguelin, J. Dongarra, R. Manchek, V. Sunderam, PVM 3.0 User's Guide and Reference Manual. Technical Report ORNL/TM-12187, Oak Ridge National Laboratory, February 1993.

[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).