1993 PVM Users' Group Meeting

Knoxville Tennessee
May 10-11, Knoxville Hilton

May 26, 1993

Sponsored by
Department of Energy and University of Tennessee

with support from	Center for Research on Parallel Computation
	Convex Computer Corporation
	Cray Research, Inc.
	Digital Equipment Corporation
	Intel Supercomputer Systems Division

ORGANIZING COMMITTEE

General Chair:	Jack Dongarra, Univ. of Tenn. and Oak Ridge Nat'l Lab.
Program Committee:	Adam Beguelin, Carnegie Mellon Univ. and Pitts. Supercomputing Ctr
	Jack Dongarra, Univ. of Tenn. and Oak Ridge Nat'l Lab.
	Al Geist, Oak Ridge National Laboratory
	Bob Manchek, University of Tennessee
	Vaidy Sunderam, Emory University

SUNDAY (May 9th)	
6:00-9:00	Conference Registration/Reception Salons D & E (second floor) Cherokee Ballroom
MONDAY (May 10th)	
7:30-8:30	Coffee and muffins (registration area)
8:30-5:00	Conference Registration
8:55-9:00	OPENING SESSION and WELCOME
9:00-10:00	SESSION 1, Chair: Jack Dongarra, University of Tennessee / Oak Ridge National Laboratory
9:00-9:40	(Invited talk) Salons A & B (second floor) Cherokee Ballroom Dennis Duke, Florida State University, "PVM Experiences at SCRI"
9:40-10:20	(Invited talk) Salons A & B (second floor) Cherokee Ballroom Bob Manchek, University of Tennessee, "PVM 3.1"
10:20-10:50	Coffee Break
10:50-12:10	PARALLEL SESSION 2 Contributed Papers
Applications	Sequoyah Room 3 (second floor) Chair: Mike Berry, University of Tennessee
10:50-11:10	Forrest B. Brown & Keith L. Derstine, Argonne National Laboratory, "Distributed Computing and Nuclear Reactor Analysis"
11:10-11:30	Bill Shelton, Oak Ridge National Laboratory, "Developing Large Scale Material Science Computer Codes by Integrating PVM and the Paragon"
11:30-11:50	John Drake, Oak Ridge National Laboratory, "A PVM3 Version of the Parallel Community Climate Model"

11:50-12:10	Rod A. Fatoohi, NASA Ames, "Performance of NAS Simulated Application Benchmark on a Cluster of SGI Workstations"
Experiences	Salons A & B (second floor) Cherokee Ballroom Chair: Adam Beguelin, Carnegie Mellon Univ. and Pitts. Supercomputing Ctr
10:50-11:10	Vaidy Sunderam, Emory University, "The PVM System: Some Recent Results & Future Directions"
11:10-11:30	Weicheng Jiang, University of Tennessee, "Using PVM on Multiprocessor Machines"
11:30-11:50	Robert Benway, General Electric Co., "A User's Experience with Increasingly Large PVM Klusters"
11:50-12:10	Gary R. Smith, Lawrence Livermore National Laboratory, "A Framework for Interactive and Flexible Parallel Computation"
12:10-1:30	Lunch Break (on your own)
1:30-2:10	SESSION 3 Chair: Al Geist, Oak Ridge National Laboratory (Invited talk) Salons A & B (second floor) Cherokee Ballroom Donna Bergmark, Cornell University, "The History and Use of PVM at the Cornell Theory Center"
2:10-3:10	PARALLEL SESSION 4 Contributed Papers
Visualization	Salons A & B (second floor) Cherokee Ballroom Chair: David Semeraro, Oak Ridge National Laboratory
2:10-2:30	Mark Bolstad, Martin Marietta Technical Services, "Scientific Visualization and PVM"
2:30-2:50	Vasudha Govindan, Michael B. Wolfe, and Mark A. Franklin, Washington University, "N-Body Simulations: Load Balancing, Visualization and Performance Evaluation "
2:50-3:10	David Robertson, Lawrence Berkeley Laboratory, "Using PVM 3.1 for Distributed Visualization of Large 3D Data Sets"

Object Oriented	Sequoyah Room 3 (second floor) Chair: Vaidy Sunderam, Emory University
2:10-2:30	Roldan Pozo and Stan Green, University of Tennessee, "An Object Oriented Stream Interface for PVM"
2:30-2:50	Raymond Namyst, Jean-Francois Roos, Luc Courtrai, and Jean-Francois Mehaut, Universite des Sciences et Technologies de Lille, France, "Implementation of Parallel Object Languages using PVM"
2:50-3:10	David Greco, Centre for Advanced Studies, Research and Development, Sardinia, "ParLib++"
3:10-3:40	Coffee Break
3:40-5:00	PARALLEL SESSION 5 Contributed Papers
Linear Algebra	Sequoyah Room 3 (second floor) Chair: Jack Dongarra, University of Tennessee/ORNL
3:40-4:00	Massimo Bernaschi, Salvatore Filippone, and Maria Loredana Sales, IBM, Italy, "Linear Algebra Operations and PVM on IBM RISC/System 6000 Clusters"
4:00-4:20	Victor Eijkhout, University of Tennessee, "Iterative Linear System Solvers Under PVM"
4:20-4:40	Steve Moulton, University of Tennessee, "PVM And Linear Algebra; A Library Level Instrumentation of LU, QR and Cholesky Factorization"
4:40-5:00	Madhurima Ghose and M. Zubair, Old Dominion University, "Cholesky Factorization of Banded Matrices using Level 3 and Level 2 BLAS on a Cluster of High Performance Workstations"
Systems	Salons A & B (second floor) Cherokee Ballroom Chair: Keith Moore, University of Tennessee
3:40-4:00	Christian Tricot, Archipel, France, (Speaker: Richard Barrett, Univ. of Tenn.),

"PVM on Transputer-Based Parallel Machines"

4:00-4:20 Richard Marciano, Demetrius Rokos, Frank Weirich,
Weeg Computing Center and Department of Computer Science and Department of Geography,
"Using PVM and Network LINDA to Obtain Supercomputer-Level Performance on a Network of Workstations"

4:20-4:40 Pat Estep, Convex Computer Corporation,
"Using PVM with Shared Memory Interconnects"

6:00-7:00 RECEPTION at Bentley's in the Hilton

7:00 CONFERENCE BANQUET Salon C (second floor) Cherokee Ballroom
After Dinner Speaker
Neil Lincoln, SSESCO,
"The Perils of Promiscuous Parallel Programming"
(subtitle) "My Six Months in Clustering Hell"

TUESDAY (May 11th)

8:00-9:00	Coffee and muffins (registration area)
9:00-10:20	SESSION 6, Chair: Adam Beguelin, Carnegie Mellon and Pitts. Super. Ctr
9:00-9:40	(Invited talk) Salons A & B (second floor) Cherokee Ballroom Peter Rigsbee, Cray Research Inc., "PVM at Cray Research"
9:40-10:20	(Invited talk) Salons A & B (second floor) Cherokee Ballroom Karl-Heinz Winkler, Los Alamos National Laboratory, "Experiences with the Los Alamos-IBM Workstation Clusters"
10:20-10:50	Coffee Break

10:50-12:10	PARALLEL SESSION 7 Contributed Papers
Applications	Sequoyah Room 3 (second floor) Chair: Bill Shelton, Oak Ridge National Laboratory
10:50-11:10	A. Trayanov, Research Institute, North Carolina Supercomputing Center, "Application of PVM to High Energy Physics: Real-time Dynamics in Lattice Gauge Theory"

11:10-11:30	David Semeraro, Oak Ridge National Laboratory, "Parallel Panel Method Using PVM"
11:30-11:50	Stephen Fleischman, Convex Computer Corporation, "Using PVM in Computational Chemistry"
11:50-12:10	Michael Nelson, NASA Langley Research Center, "PVM Use in the Computational Aerosciences Projects"

Debugging	Salons A & B (second floor) Cherokee Ballroom Chair: Al Geist, Oak Ridge National Laboratory

10:50-11:10	Rob Gordon, Convex Computer Corp., "pvmdb : Debugging in a PVM Environment"
11:10-11:30	Adam Beguelin, Carnegie Mellon Univ. and Pitts. Supercomputing Ctr "Scalable Xab and Early Experience with Dome"
11:30-11:50	Milon Mackey, H-P Labs, "Program Replay in PVM"
11:50-12:10	Norman Wilson, University of Toronto, "Dynamic Load Balancing under PVM"

12:10-1:30	Lunch Break (on your own)

1:30-2:50	PARALLEL SESSION 8 Contributed Papers

Applications	Sequoyah Room 3 (second floor) Chair: Roldan Pozo, University of Tennessee

1:30-1:50	Andrew Lumsdaine, Mark W. Reichelt, and Jacob K. White, University of Notre Dame, The MathWorks, and MIT, "Waveform Iterative Methods for Solving Initial Value Problems"
1:50-2:10	Thomas Ericsson, Swedish Institute of Applied Math, "Boundary Integral Methods and PVM"
2:10-2:30	M. Kern and W.W. Symes, Rice University, "Shot Level Parallelization of DSO using PVM 3.0"

2:30-2:50	M.J.P. Nijmeijer and A.M. Ferrenberg, University of Georgia "A Parallel Molecular Dynamics Algorithm for the Simulation of the Growth of Optical Fibers"

Scheduling	Salons A & B (second floor) Cherokee Ballroom Chair: Robert Manchek, University of Tennessee

1:30-1:50	Jim Patterson, Boeing, "PVM applications under DQS"
1:50-2:10	Miron Livny and Jim Pruyne, University of Wisconsin, Madison, "Scheduling PVM on Workstation Clusters using Condor"
2:10-2:30	Doug Sept and Mike Berry, University of Tennessee "Task Queue Management with PVM"
2:30-2:50	Karen Barrett, Christopher Carothers, Ron Chadwick, Vernard Martin College of Computing, Ga. Institute of Technology Atlanta, GA "A Distributed Scheduler for Parallel Jobs in a Network of Workstation"

2:50-3:20	Coffee Break
3:20-4:00 Interactive Session	SESSION 9 Salons A & B (second floor) Cherokee Ballroom "Future Directions for PVM" Chair: Team PVM
4:00	End of Conference

PVM Experiences at SCRI

Dennis Duke
Supercomputer Comp. Research Inst.
Florida State University
Tallahassee FL 32306-4052
dduke@scril.scri.fsu.edu

PVM 3.1

Robert Manchek
Computer Science Department
University of Tennessee
Knoxville TN 37996-1301
manchek@cs.utk.edu

Distributed Computing and Nuclear Reactor Analysis

Forrest B. Brown & Keith L. Derstine
Reactor Analysis Division
Argonne National Laboratory
fbbrown@anl.gov klderstine@anl.gov

For over 30 years, Argonne National Laboratory has developed and applied large-scale software systems for the engineering analysis of nuclear reactor plants. Large code systems such as VIM (3D continuous energy Monte Carlo), DIF3D (3D finite-difference or nodal diffusion theory), MCC-2 (cross-section generation), SASSYS (transient analysis of reactor plants), etc., were developed and run on mainframe computers and supercomputers.

In the past two years, all of the reactor analysis codes and supporting software have been adapted to a distributed computing environment, based on a network of UNIX workstations. An integrated support environment and distributed computing methodology were established on the division network, including the following major components: (1) UNIX operating systems, with tcp/ip networking, (2) special configuration of the memory, disk, and scratch space on each workstation to accommodate both interactive and batch usage for large jobs, (3) DQS, the Distributed Queueing System from Florida State University, to manage the scheduling and load balancing for batch jobs, (4) the PVM software package from Oak Ridge National Laboratory to provide a framework for parallel processing of large applications, (5) the P4 package from Argonne to provide additional support for parallel applications, (6) numerous Bourne shell, C shell, and perl procedures to provide transition and convenience features for a diverse user group, (7) libraries of compatibility routines for I/O and environmental support, (8) source code management and version control using SCCS, (9) frequent and reliable backup services, and (10) continued development of the large production codes to improve their performance in a distributed computing environment. At present, the division workload is fully supported by a network of 15 SUN workstations, 2 IBM RS6000 workstations, and 50 NCD X-terminals. About 3,000 batch jobs are run with DQS in a typical month.

For computationally intensive tasks such as Monte Carlo and SPRT (sequential probability ratio testing), current development is focusing on parallel processing using PVM and P4. The PVM concept of a "virtual supercomputer" and the DQS approach to batch work are central to our environment. Parallel applications are intended to be scalable, so that development and small problems can be accomplished on the workstation network, and large applications can be off-loaded to Argonne's new IBM SP1 parallel computer. The use of a message passing approach ensures that such portability and scalability can be realized.

?

Bill Shelton
Mathematical Sciences Section
Oak Ridge National Laboratory
PO BOX 2008 Building 6012
Oak Ridge, Tennessee 37821-6367
shelton@msr.epm.ornl.gov

A PVM3 Version of the Parallel Community Climate Model

John Drake
Mathematical Sciences Section
Oak Ridge National Laboratory
PO BOX 2008 Building 6012
Oak Ridge, Tennessee 37821-6367
bbd@msr.epm.ornl.gov

A brief overview is presented of a parallel version of the NCAR Community Climate Model, CCM2, implemented for MIMD massively parallel computers using a message-passing programming paradigm. The parallel algorithms used for the atmospheric circulation will be described. Dynamics are calculated using a spectral transform method which requires global communication as well as with a semi-Lagrangian transport method. The parallel implementation was developed on an Intel iPSC/860 with 128 processors and on the Intel Delta with 512 processors. PVM3 is being used as a message system that allows portability across various parallel platforms and also for its parallel debugging environment on a network of workstations.

Performance of NAS Simulated Application Benchmark on a Cluster of SGI Workstations

Rod Fatoohi and Sisira Weeratunga
Computer Sciences Corp.
NAS Applied Research Branch
Mail Stop T045-1
NASA Ames Research Center
Moffett Field, CA 94035

We present early results of an experiment to study the performance of a cluster of Silicon Graphics, Inc. (SGI) workstations at NASA Ames Research Center using the three simulated CFD applications of NAS Parallel Benchmark suite. These workstations are mainly 4D320s and 4DRPCs (Indigo) connected via Ethernet, and some of them are connected via Ultranet. These machines have a clock cycle of at least 33 MHz. The simulated applications employ the numerical solution of a synthetic system of five nonlinear PDEs using iterative techniques similar to those found in CFD applications. These techniques are time implicit integration techniques which require a significant amount of communication, both short and long, on parallel machines. The parallel versions of these codes were first developed on the 128 processor Intel iPSC/860 machine at NASA Ames Research Center. Recently, these codes were converted to a cluster of SGI workstations using PVM version 2.4. Several routines were written to convert Intel message passing calls to PVM 2.4 calls. Different partitioning schemes of the computational domain are considered and the results are reported for both environments. Comparisons are drawn between the tightly coupled parallel machines environment, exemplified by the iPSC/860, and the loosely coupled parallel machines environment, exemplified by the cluster of SGI workstations.

The PVM System: Some Recent Results & Future Directions

Vaidy S.Sunderam
Dept. of Math and Computer Science
Emory University
Atlanta, GA 30322
vss@mathcs.emory.edu

The two most important aspects of systems such as PVM are performance and functionality. Performance is primarily characterized by communications speed and overheads; in this talk, we present recent results on both Ethernet and FDDI networks, and briefly describe some experimental enhancements for improving performance. The latter part of the talk will discuss the "Generalized Distributed Computing Layer", an auxiliary subsystem for PVM aimed at widening the applicability of the system and making it functionally comprehensive.

Using PVM on Multiprocessor Machines

Weicheng Jiang
Computer Science Department
University of Tennessee
Knoxville TN 37996-1301
wjiang@cs.utk.edu

PVM's view of the world. How the multiprocessor environment differs from that of a workstation cluster. Writing portable PVM applications. A few words on implementation.

A User's Experience with Increasingly Large PVM Klusters

Robert W. Benway
General Electric Co.
Corporate Res. & Dev. Ctr.
Bldg. KWC-114
P. O. Box 8
Schenedtady, NY 12301
rwb@c3.crd.ge.com

Two GE components, Corporate Research & Development and Power Generation Engineering have been using PVM (Parallel Virtual Machine, from Oak Ridge National Laboratory), separately and jointly, in research efforts involving large scale problems. In the course of this usage, numbers of systems employed in problem solution have increased from a single machine to klusters of over 30 systems. These klusters have been both homogeneous and heterogeneous in nature. As kluster size increases, various factors have appeared which have resulted in modifications to problem approaches. Discussion will feature the general nature of two types of problems subjected to PVM solution as well as technical aspects of employment of large klusters of systems.

A Framework for Interactive and Flexible Parallel Computation

Gary R. Smith
gsmith@llnl.gov
L-630, P.O. Box 808
Lawrence Livermore National Laboratory, Livermore, CA 94551

PVM has been used to implement a parallel-computation framework called PROTOPAR that is strongly oriented towards enhancing the productivity of scientists who are building and debugging programs that may run on a variety of computational resources. Programs built within the PROTOPAR framework are highly interactive and have great flexibility. Scientists may use such programs like general-purpose calculators (with plotting capabilities) augmented by function keys that invoke routines relevant to the scientist's specialty. Interactivity and flexibility are provided by means of the Basis system developed at Lawrence Livermore National Laboratory. Features of Basis include portability to a wide variety of Unix environments; macro expansion, compiling, and loading on multiple architectures; run-time memory management, setting of variables, display of documentation, sizes, and values of variables, looping, and execution of user-defined or compiled (Fortran or C) coding. The PROTOPAR framework retains all of these features even though variables may be distributed in different processor memories and compiled coding may be run concurrently on many processors. Macros are used to localize details of message-passing library routines. PROTOPAR programs have a master-and-slaves structure. Domain decomposition can be used to partition work among slaves.

The History and Use of PVM at the Cornell Theory Center

Donna Bergmark
745 E&TC Building
Hoy Road, Cornell
Ithaca, NY 14853
bergmark@tc.cornell.edu

The emergence of PVM in 1991 as a popular message passing library coincided with the Cornell Theory Center's foray into clustered and distributed computing. Since that time our cluster of RS/6000's has grown and so has our use of PVM. This talk will cover various aspects of PVM at Cornell, including PVM tools, applications, training, and plans for the future.

Scientific Visualization and PVM

Mark Bolstad
Martin Marietta Technical Services
8700 Walkelin Court
Raleigh, NC 27615
bolstad@vislab.rtpnc.epa.gov

Scientific Visualization uses computer graphics to gain insight into scientific data, whether it be produced by numerical simulations or collected in the field. Many of the algorithms used in visualization can be parallelized, but few implementations exploit it.

This presentation will show how scientific visualization in a heterogenous computing enviroment can benefit from a wide-range of types of parallelism. Using PVM, visualization systems can exploit fine-grain parallelism within the algorithms as well as coarse-grain parallelism by distributing entire "modules" or algorithms onto different machines.

Results are presented for the parallelization of two visualization algorithms, particle tracing, and a ray-traced volume renderer. Comparisons of the relative performances of the scalar and parallel algorithms along with the effects of changing the number of processors as well as the composition of the parallel machine are shown.

N-Body Simulations: Load Balancing, Visualization and Performance Evaluation

Vasudha Govindan (vasu@wuccrc.wustl.edu)
Michael B. Wolfe (mbw1@wuccrc.wustl.edu)
Mark A. Franklin (jbf@random.wustl.edu)
Computer and Communications Research Center
Washington University
Campus Box 1115
St. Louis, MO 63139

The work presented here is part of an ongoing effort directed towards the design, implementation and performance evaluation of parallel algorithms on distributed computing systems. The N-body simulation algorithm was implemented on a network of SUN workstations under the PVM environment. Here, we discuss the communication costs and the performance of N-body simulations on the workstation network and present two load balancing algorithms to improve performance.

Classical N-body problems study the evolution of a system of bodies under forces exerted on each body by the entire ensemble. For n bodies, a direct evaluation of forces results in a time complexity of $O(n^2)$. However, hierarchical tree-based methods reduce the complexity to $O(nlogn)$. Our implementation is based on the hierarchical Barnes-Hut algorithm to study the evolution of a system of particles under gravitational forces. The particles are distributed among the available processors. The simulation proceeds over a number of timesteps, at each timestep computing the net force on every particle and updating its position and other attributes.

The dynamic nature of the simulation causes load imbalance resulting in significant performance degradation. We have developed and implemented two load balancing strategies - domain redistribution and dynamic load balancing. A speedup factor of 4.4 was obtained on eight processors with dynamic load balancing.

We have modified the ParaGraph visualization tool to help us visualize various aspects of the algorithm. The tool displays the evolution of the system of particles and the performance of the load balancing techniques.

The communication costs of the system largely limit the performance of the N-body algorithm and the load balancing techniques. We have developed a detailed model of the communication costs under the PVM system and a performance model of the N-body algorithm to identify performance bottlenecks and evaluate system and algorithm enhancements.

Our current and future work is directed towards developing advanced strategies for partitioning, load balancing and synchronization, investigating techniques for overcoming communication latencies, and developing accurate performance models and advanced visualization tools.

Using PVM 3.1 for Distributed Visualization of Large 3D Data Sets

David W. Robertson, Brian L. Tierney, and William E. Johnston
Imaging Technologies Group
Lawrence Berkeley Laboratory
dwrobertson@lbl.gov

As technology improves, larger and larger data sets are being produced by 3D imaging devices such as MRI. A number of data parallel algorithms have been developed for visualizing such data. Often a massively parallel processor has been used for this purpose, but a distributed environment can be used as well. The combined processing and especially memory capability of multiple workstations can exceed that of massively parallel machines.

In this approach the dividing cubes algorithm [1] is used to generate a surface from a 3D volume for rendering. The algorithm is suitable for parallelizing at a variety of granularities: with PVM we use a relatively coarse level, which ensures a high ratio of computation to communication. The task is divided among a user interface module, and a master module which parcels out data to worker modules.

Dividing cubes is a local algorithm, and calculations for portions of the data can be undertaken independently by each processor. The 3D data set is divided into the same number of sub-cubes as there are worker processors. Calculations require padding each sub-cube by two in each direction to allow correct computation at the edges of sub-cubes. The output is a list of 3D points with associated surface normals for lighting calculations.

Each sub-cube's set of points is rendered, with hidden surface removal performed by the painter's algorithm. The resulting sub-images are then sent to the master for compositing in the correct order into the final image, which is sent to the user interface module for display.

We have used this approach with processors on a Cray C90, and on groups of Sun workstations. It has successfully been used on very large data sets, including a 45 megavoxel (248x496x368) MRI data set obtained by enlarging an existing data set.

[1] H.E. Cline, W.E. Lorensen, S. Ludke, C.R. Crawford, and B.C. Teeter, Two Algorithms for the

Three-Dimensional Reconstruction of Tomograms, Medical Physics, May 1988.

An Object Oriented Stream Interface for PVM

Roldan Pozo and Stan Green
Computer Science Department
University of Tennessee
Knoxville TN 37996-1301
pozo@cs.utk.edu and sgreen@cs.utk.edu

We present an object-oriented interface in C++ for message passing in PVM 3.0. By abstracting the notion of interprocessor communication to resemble byte stream input/output, we can extend the familiar notion of C++ streams to include interprocessor communication. This new approach simplifies the message passing syntax and allows for runtime type-checking of messages.

This interface supports no send() or recv() calls; instead communication is accomplished by inheritance of basic message stream classes and the overloading of the standard stream insertion operator “$\gg$” and the extraction operator “$\ll$” for input/output (I/O) in C++. We implement this interface as a simple library in C++ which can built upon PVM. This allows to create an extendable heterogeneous programming interface without changing the basic layouts of existing distributed-memory programs or sacrificing run-time performance.

Various programming examples are presented and we discuss the relative strengths and weaknesses of this approach.

Implementation of Parallel Object Languages using PVM

Raymond Namyst
Jean-François Roos   Luc Courtrai   Jean-François Méhaut
Laboratoire d'Informatique Fondamentale de Lille - URA CNRS 369
Université des Sciences et Technologies de Lille
Bâtiment M3 59655 Villeneuve d'Ascq Cedex - France
Email: {namyst,roos,courtrai,mehaut}@lilf.fr

Our research team works on the design and the implementation of Parallel Object Environments for Distributed Architectures. In order to support such an ambitious project, we have first designed a medium level tool which purpose was to hide architecture specific details to the programmers by offering them a uniform view of the system. Therefore, we proposed to modelize a unique structuring entity (called **CAC**) which every distributed application should be composed of.

A CAC (**C**ommunicating **A**ctive **C**omponent) can be compared to an Actor, in the way that it has a thread of control (that's running the CAC's behavior), a mailbox and a local data area. Behavioral functions and the associated created CAC/s are partitioned in **modules**, which may be duplicated onto several nodes. A global CAC naming mechanism is provided, allowing direct communication between CAC/s, even if they belong to different modules.

The present platform runs on a set of cooperating workstations. In terms of implementation, modules are carried out by UNIX processes and CAC/s by lightweight ones (using user-level threads packages).

Thus, C$_A$C/s provide a way to express fine-grain parallelism inside of modules which are a tool for supporting their distribution. This approach is especially suitable for language implementations which provide *active objects* model and/or which allow intra-object parallelism.

The current version of the platform is entirely based on the *PVM* layer, which is complemented by various lightweight process libraries (like *POSIX* threads). The integration of lightweight processes inside of PVM ones, as well as the associated inter-lightwei ght process communication mechanism, constitutes an interresting innovation. Furthermore, *PVM*, associated with *POSIX* threads packages, guarantees our runtime to become increasingly portable.

A multi-paradigm parallel Object Oriented Language, called *BOX*, has already been developped on top of the C$_A$C runtime, making great benefits of *PVM* operability across heterogeneous architectures. This shows that PVM, often seen as a tool for parallel scientific programming, is also an excellent support for the design and the implementation of Parallel Languages.

ParLib++

David Greco
Parallel Computing Group
Centre for Advanced Studies, Research and Development in Sardinia I-09123 Cagliari Italy
e-mail greco@crs4.it

ParLib++ is a C++ library to distribute arrays transparently onto parallel processes. ParLib++ is built on top of the message passing library PVM (Parallel Virtual Machine). PVM offers a simple mechanism of communication (send and receive) among different processes running on different machines. Only send and receive constructs are utilized, therefore it is easy to port this library onto additional architectures. Using this library, a user can write a program in a Fortran 90 style without worrying about the distribution of arrays on the processes involved in the computation.

ParLib++ is based on the SPMD (Single Program Multiple Data) paradigm, which means that there is a single program which is duplicated onto the all processors. Each copy of the program manages only a part of an array while global data, for example scalar variables or small arrays, are duplicated on all the processes.

There are four classes of arrays corresponding to different types of data: `ParallelFloatArray`, `ParallelIntArray`, `ParallelDoubleArray`, `ParallelLogicalArray`. On each class it is possible to denote the basic four operations by using the ordinary operators (+, -, *, /). Moreover there are global and reducing operations such as `cshift`, `global sum/or/and`, `max`, `min` .

Linear Algebra Operations and PVM on IBM RISC/System 6000 Clusters

Massimo Bernaschi, Salvatore Filippone, and Maria Loredana Sales
IBM ECSEC
Viale Oceano Pacifico 173
00144 Roma, Italy
bernasch@romvmnic.vnet.ibm.com
filippon@vnet.ibm.com
sales@vnet.ibm.com

We report our experience in porting PVM and linear algebra codes on clusters of IBM RISC System/6000 workstations.

We describe the design and implementation issues we encountered optimizing the PVM perfomance on RISC/6000 clusters connected by means of high speed networks like FDDI and the SOCC[1]. A complete new implementantion of PVM, fully compatible with the public domain PVM 2.4.2, is presented along with the low level communication protocols for the different types of network supported. The new PVM not only has a higher raw communication perfomance but also offers a better support for large (i.e. memory demanding) applications. Further improvements will be possible by exploiting the new concepts introduced in PVM 3.0. We also discuss the interacton between the application structure and the PVM structure affecting the application performance.

As a reference test case we shall use the LINPACK benchmark, implemented with the use of BLADE, a distributed memory linear algebra library developed in conjunction with the PVM porting. The discussion will include a closer look at the effect of the communication bandwidth and latency on this type of applications.

Iterative Linear System Solvers Under PVM

Victor Eijkhout
Computer Science Department
University of Tennessee
Knoxville TN 37996-1301
eijkhout@cs.utk.edu

As part of the ScaLapack project, a library of iterative solution methods for linear systems is being implemented. Two distributed memory platforms are targeted at the moment: Picl and Pvm. The Pvm implementation makes use of such features as multiple buffers and multicast. In this talk, the global structure of the library and some implementational details will be discussed.

[1] Risc System/6000 and SOCC are trademarks of IBM Co.

PVM And Linear Algebra; A Library Level Instrumentation of LU, QR and Cholesky Factorization

Steve Moulton
Computer Science Department
University of Tennessee
Knoxville TN 37996-1301
moulton@cs.utk.edu

In this talk, we discuss the PVM Implementation of the BLACS (Basic Linear Algebra Communication Subprograms), and some performance testing on a virtual machine consisting of IBM RS6000/320h and 340 workstations.

Cholesky Factorization of Banded Matrices using Level 3 & Level 2 BLAS on A Cluster of High Performance Workstations

M. Ghose and M. Zubair
Department.of Computer Science
Old Dominion University
Norfolk, VA 23529
ghose@cs.odu.edu

In the recent past, clusters of high performance workstations on a local area network have emerged as a cost effective complement to high cost parallel machines for a number of large scale scientific applications. PVM provides a convenient programming environment for concurrent computing on such systems. The performance of algorithms on clusters is limited by high latency and significant communication costs. In general, problems that can be divided into subproblems with coarse granularity and with high computation to communication ratios tend to perform well on such systems. The effect of latency for such cases is minimal and the available communication bandwidth is effectively utilized. Solution of dense linear systems are examples of such problems and recently good performances for them have been reported. Many of the matrices arising from large scientific applications are not dense and a number of such matrices have a banded structure. Banded solvers, as opposed to dense solvers, may not inherently be well suited to such systems because of their lower computation to communication ratios. Therefore, algorithms for such problems need to be designed carefully to reduce communication overheads. Also, for a scheme to perform well on any parallel machine, it is important to have a good single node performance. In this paper, we present a scheme for computing the Cholesky factorization of large banded matrices that aims at improving the single node performance and also minimizes the communication overheads. The computation kernels on each node has been designed in terms of level 2 and level 3 BLAS. We try to minimize the number of messages and consequently the effect of latency and also reduce the total volume of communication. We study the effects of varying bandwidth, block sizes and number of processors on the performance of our scheme. We tested our scheme on matrices arising from real applications, for example, matrices arising from High Speed Civil Transport plane model (obtained from NASA) and from acoustic scattering problem (obtained from NRL). We used PVM as our parallel programming environment. All our experiments were conducted both on a cluster of Sun 4 workstations and on a cluster of RS/6000 Model 550 workstations with upto 8 nodes. We present below our preliminary unoptimized results on one of the matrices obtained from acoustic scattering problem.

Table 1. Performance results for matrix of size 2112 and bandwidth 134.

No. of procs	Wall clock time (in secs) for clusters of SUN 4
1	25.86
2	24.48
3	21.79
4	20.78

PVM on Transputer-based Parallel Machines

Christian Tricot
Archipel
Speaker: Richard Barrett
University of Tennessee

Archipel, French vendor of distributed-memory multiprocessors, has ported PVM to their VOLVOX machines based on Transputers. PVM is available on Archipel's VOLVOX machines. They run Volvix (parallel UNIX) and PVM takes advantage of these Unix facilities.

The aim of this talk is to present the way PVM is implemented over Volvix. PVM uses Unix features to be able to efficiently communicate between the external world of the parallel machine and the parallel machine itself, and uses the features of the machine itself to run inside the machine (to provide efficient communications between the different nodes). In a first part of the talk, Volvix will be presented, and in a second part the implementation of PVM over Volvix will be detailed.

The parallel machine is viewed as a compute resource. It is managed by a daemon that dynamically balance the load among the processing nodes. Once a task is sent to the VOLVOX parallel machine, the daemon handles it and sends it to a node whose identity is chosen depending on the value of a criterion (number of pending tasks, available resources...). The daemon is responsible for transmitting the messages coming from the external world to the right task on the parallel machine, and transmitting messages coming from a task in the parallel machine to the external world. It uses the Unix sockets. But inside the parallel machine, another communication protocol is used that is very efficient since it takes advantage of the hardware of the parallel machine itself. However, it is totally transparent to the user. In other words, any PVM application written using PVM procedures, will run efficiently, because at compile time, a version of the programs is prepared so that they can run efficiently on the parallel machine. This step is done through calls to a communication library, thus translating the calls to PVM communication procedures into calls to VOLVOX communication procedures.

Using PVM and Network LINDA to Obtain Supercomputer-Level Performance on a Network of Workstations

Richard Marciano (1), Demetrius Rokos (2), and Frank Weirich (2)
(1) Weeg Computing Center and Department of Computer Science
(2) Department of Geograph
The University of Iowa
marciano@umaxa.weeg.uiowa.edu

Experiments at the Cornell National Supercomputing Center on a network of 7 IBM RS/6000 Model 550 POWERstations show surprising supercomputer-level performance. The message-based PVM parallel programming environment was used to model a Lyapunov chaos computation problem. The timing results obtained are compared with the best serial and parallel implementations we could reach on a wide range of architectures: PCs, workstations, mainframes and supercomputers comprising the IBM3090-600J (using all 6 processors), KSR-1 (using 64 processors), iPSC/860 (using 64 processors), CRAY Y-MP (using 8 processors), CM-2 (using 32K processors), CM-5 (using 128 processors), MasPar MP-1 (using 16K processors), and MasPar MP-2 (using 4K processors).

The PVM-based implementation beats the Cray Y-MP (1 processor), IBM3090-600J (6 processors), CM-5 (128 processors), and comes very close to the KSR1, MasPar and CM-2 runs.

We also contrast the PVM and Network Linda approaches and report timing results for a Network Linda implementation of the Lyapunov chaos problem running on a heterogeneous network of 4 IBM RS/6000 workstations (models 530, 320h, 320, and 220l) at the "Geography Parallel Processing Lab" at The University of Iowa.

Using PVM with Shared Memory Interconnects

Pat Estep
Convex Computer Corporation
Richardson, TX
pestep@rigel.convex.com

PVM was modified to take advantage of the Shared Memory Interconnect (SMI) on a Convex Meta Series cluster.

SMI is a low-latency, high-bandwidth, interconnect designed for those that plan to use their Meta Series as a parallel, time-to-solution engine, or want to start developing applications for the Convex Scalable Parallel Processor today.

In developing the high performance interconnect for the Meta Series, Convex looked at a variety of possible options, including HIPPI and UltraNet. Upon close examination, it was decided that none of these products offered the right balance of low-latency and high-bandwidth that is so important to cluster computing, especially when used in a parallel environment. Therefore, a custom interconnect was chosen as the best solution.

After Dinner Speaker

The Perils of Promiscuous Parallel Programming (subtitle) My Six Months in Clustering Hell

Neil Lincoln
Supercomputer Systems Engr & Service Co.
1200 Washington Ave.
Minneapolis MN 55415-1258
nrl@nips.ssesco.com

PVM at Cray Research

Peter Rigsbee
Cray Research, Inc.
655 F Lone Oak Drive
Eagan, MN 55121
par@teak.cray.com

Experiences with the Los Alamos-IBM Workstation Clusters

Karl-Heinz Winkler
X-DOT
Los Alamos National Lab.
Los Alamos, NM 87545
khw@lanl.gov

Application of PVM to High Energy Physics. Real-time Dynamics in Lattice Gauge Theory

A. Trayanov
Research Institute, North Carolina Supercomputing Center,
a division of MCNC
PO Box 12889
3021 Cornwallis Road
Research Triangle Park, NC 27709-2889

In the last two decades lattice gauge theory simulations have become a very successful tool to study processes in high energy physics, and in particular, in the quark-gluon plasma. Due to the huge demands for CPU and memory resources those simulations are limited to small lattice sizes and to processes close to equilibrium. Recently we have developed a new approach to study the gauge fields based on Hamiltonian evolution in the semi-classical approximation. The field dynamics is simulated by numerical integration of the equations of motion. We show that the field dynamics is chaotic, which leads to very short thermalization times. The scalar version of the code is limited to moderate lattice sizes (up to 28^3) and relatively short simulation runs.

Many interesting phenomena that include dynamics of baryon number violation, topological defects, pion physics, etc., require simulations on bigger lattices and/or longer runs. Here we report the results of our first attempt to apply PVM to a relatively coarse-grain parallelization of the lattice gauge theory code. The lattice is split in several sub-cubes and the field dynamics in each of them is mapped to a grid of PVM nodes. Each of the PVM processors integrates the equations of motion in its region of the space with no periodic boundary conditions. The only additional information is the value of the magnetic field at the sub-cube boundaries and it is delivered to the neighboring nodes using the PVM send/receive protocol.

The results from the simulations performed on a cluster of 8 SUN SPARC stations are very favorable and demonstrate that PVM can be a very useful tool to study computationally bounded problems. The gauge dynamics parallel code scales with number of processors and lattice sizes. The speedup curve is almost linear with a speedup factor of 7 on 8 processors.

Parallel Panel Method Using PVM

David Semeraro
Mathematical Sciences Section
Oak Ridge National Laboratory
PO BOX 2008 Building 6012
semeraro@msr.epm.ornl.gov

Some experiences with a parallel panel method for solving potential flow problems are discussed. In particular, issues related to the solution of three dimensional potential flow problems in aerodynamics, and the decomposition of the problem for solution on a network of workstations using PVM are described.

The panel method for solving potential flow problems has three major parts. Those parts are 1) the formation of a dense matrix of influence coefficients, 2) the solution of a linear system of equations, and 3) the determination of the aerodynamic characteristics of the geometry under investigation.

Each aspect of the solution process can be parallelized to some degree. We discuss the strategy for parallelization of each aspect and the use of PVM for the implementation of the method on a network of workstations.

Using PVM in Computational Chemistry

Stephen Fleischman
Convex Computer Corporation
Richardson, TX
sfleisch@hydra.convex.com

PVM [1] was used to parallelize three types computational chemistry applications: molecular dynamic, neural networks and genetics algorithms, for use on workstation clusters.

With the program CHARMM [2], 3.4X performance on a 4 node HP730 cluster was obtained using FDDI. The results will be presented for the faster HP735 cluster and the higher bandwidth VME based shared memory interface (SMI) as well as the faster FDDI available on the HP735. The writing of efficient combination and broadcast routines will be discussed.

Feed-forward back-propagation neural networks [3] have been used in a number of areas of computational chemistry as well as in the interpretation and control of experimental chemistry. They have been used to predict protein secondary structure based on sequence, to predict properties such as NMR chemical shifts and ESR g-factors. A Fortran based neural network program has been written for cluster computing using PVM. It uses a novel "molecular-dynamics-like" algorithm to speed convergence. It's application to protein secondary structure prediction will be described.

Genetics algorithms [3] have been used in conformational analysis[3] and other global minimum search problems. An example of a parallel version will be presented.

References

1. A. Beguelin, J. J. Dongarra, G. A. Geist, R. Manchek and V. S. Sunderam, ORNL-TM-11826, Oak Ridge National Laboratory (1991).

2. B. R. Brooks, R. E. Bruccolerie, B. D. Olafson, D. J. States, S. Swaminathan and M. Karplus, J. Comp. Chem. 4, 187, (1983).

3. D. E. Rumelhart, J. L. MaClelland, et al., Parallel Distributed Processing, Vol. 1., MIT Press, Cambridge, Ma, 1986.

4. J. H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor.

PVM Use in the Computational Aerosciences Project

Michael L Nelson
NASA
Langley Research Center
mln@blearg.larc.nasa.gov

The Computational Aerosciences (CAS) Project is one of the two Grand Challenge areas selected by NASA of the High Performance Computing and Communications Program (HPCCP). CAS is defined as the integrated, multidisciplinary simulations and design optimizations of aerospace vehicles throughout their mission profiles. The goal of the CAS project is to develop and/or accelerate the massively parallel algorithm and architecture technology necessary for the simulation and design optimization of aerospace vehicles, and to transfer this technology to the aerospace and computer industries.

At Langley Research Center, the Framework for Interdisciplinary Design Optimization (FIDO) Project is implementing a computational system to allow multi-disciplinary codes to communicate over a heterogeneous distributed network of computers. This requires the integration of codes from such disciplines as Aerodynamics, Structures, Performance, and Propulsion. To achieve this coupling, a high-level communications library is provided by the FIDO system. Because the strengths of the Parallel Virtual Machine (PVM) message passing system include heterogeneous support, robustness, high performance, large installed user base and ease of use, the FIDO communications library is built on top of PVM. The current status of FIDO and its future directions in relation to the use of PVM will be discussed.

pvmdb : Debugging in a PVM Environment

Rob Gordon
Convex Computer Corporation
Richardson, TX
rgordon@convex.com

We will present a debugger front-end integrated with PVM that manages debugging of PVM components across multiple platforms of diverse architectures. The current status of pvmdb along with future directions will be discussed.

pvmdb provides a single point of interaction for a PVM developer when debugging multiple PVM components in a heterogeneous environment. pvmdb manages multiple sessions of line-oriented debuggers native to the architecture on which each PVM component is executing. Through tight integration with the PVM daemon, pvmdb allows the user complete control over the execution and debugging of a PVM application. The use of pvmdb requires no modification to user codes and does not require the ability to 'attach' to an executing process on the part of the native debugger.

pvmdb is configurable to allow the user to select a subset of nodes for which debugging is enabled. An extension to the PVM host file enables debugging on a per node basis. A pvmdb initialization file also allows the user to select a native debugger by host name or architecture class. Since pvmdb contains no debugger specific knowledge, any line oriented debugger may be specified.

Future pvmdb development will be discussed. A Motif interface will extend the capabilities of pvmdb to include generic debugging commands across different architectures. The most frequently used debugging commands will be made available from buttons and applied either selectively or globally. The pvmdb initialization file will be used to map generic button commands to architecture specific syntax.

pvmdb is currently supported on ConvexOS and HP-UX. This does not preclude it from being used in an environment that includes other architectures. The effort to port to other UNIX platforms will be addressed.

Scalable Xab and Early Experience with Dome

Adam Beguelin
Carnegie Mellon University
School of Computer Science, and
Pittsburgh Supercomputing Center
Adam_Beguelin@N2.SP.CS.CMU.EDU

Xab (X-window Analysis and deBugging) is a tool for run time monitoring of PVM (Parallel Virtual Machine) programs. In this talk I will briefly describe Xab and discuss the current plan for making Xab a more scalable tool.

Early experience using Dome (Distributed Object Multiprogramming Environment) will also be presented. The Dome project provides tools which address several of the complicated issues of parallel programming of shared heterogeneous networks. Dome provides sets of distributed objects which are easy to use for parallel programming. Each Dome object can distribute itself, perform specific communication tasks, adjust its distribution for load balancing, checkpoint itself, and migrate portions of itself to different processors.

Program Replay in PVM

Milon Mackey
Hewlett-Packard Laboratories
1501 Page Mill Road, 3U-5
P. O. Box 10490
Palo Alto, CA 94303
mackey@cello.hpl.hp.com

The Research on Environments and Applications for Parallel Processing (REAPP) project in HP Laboratories is currently developing parallel programming environments and tools for clusters of workstations. The first tool we are developing, IVD, is a visualizing debugger for parallel programs. One of our first goals is to have IVD work with PVM programs.

As part of this effort we have been implementing "program replay" inside PVM 3.0. Program replay is one of the ways to make cyclic debugging work for parallel programs. Most parallel programs produce deterministic output. If they are given the same input, they will produce the same output. But if one looks inside the program at the actual computations performed during different runs of the program on the same input data, the computations can be wildly different. This non-deterministic internal behavior is caused by race conditions in the program that get resolved differently in different runs of the program even if the input data is the same during each run. This non-deterministic internal behavior is bad for debugging. It may not be possible to reproduce the exact resolution of the races that resulted in a defect when the program is re-executed using a debugger.

Program replay solves this problem through a two step process. First, the program is executed and a trace is made recording how all of the races were resolved. The program is then re-executed using the trace. The information in the trace is used to ensure that each race is resolved as it was during the first execution when the trace was made. This way the internal behavior of a parallel program becomes deterministic and the cyclic debugging techniques used for sequential programs can be used to debug parallel programs. To isolate any defect, the only requirement is that a trace was taken when the defect occurred.

A first implementation of this facility in PVM 3.0 is nearing completion. This paper describes the user interface to program replay and outlines the high level details of its implementation in PVM 3.0.

Dynamic Load Balancing under PVM

Norman Wilson
High Performance and Research Computing
University of Toronto (UTIRC)
Toronto, Ontario, Canada M5S 1A1
norman@utirc.utoronto.ca

We have modified PVM to consult Utopia, a locally-written load balancing system, when deciding on which node to create a new process; less-loaded nodes are picked first, at most one process of a type is permitted per node, and overloaded nodes are avoided entirely. The idea is to allow a carefully-written parallel application to coexist with higher-priority uses of the nodes of a computer cluster. The application can arrange to run only on lightly-loaded systems, moving dynamically as the load changes.

The load-balancing PVM has been used to run an astrophysics Monte Carlo code on a cluster of 60 workstations used for undergraduate instruction, with reasonable efficiency and without undue interference with interactive use.

This paper describes the changes to PVM and the structure of the astrophysics application, and presents some measurements showing how well the application ran and how busy the machines were while it was running.

Waveform Iterative Methods for Solving Initial Value Problems

Andrew Lumsdaine, Mark W. Reichelt (The MathWorks, MIT), and Jacob K. White (MIT)
Dept. Comp. Sci. & Engr.
353 Fitzpatrick Hall
University of Notre Dame
Notre Dame, IN 46556
Andrew.Lumsdaine@nd.edu

The traditional approach for solving large systems of equations in ordinary differential or differential-algebraic equations typically includes discretization in time with an implicit integration method which necessitates the solution of a linear system of equations at every timestep. This approach can be disadvantageous for parallel implementations on machines with high synchronization and communication costs, such as a cluster of workstations, because the processors must communicate and synchronize at every timestep.

Waveform methods provide an attractive alternative to standard solution methods. With this approach, the system of equations is first decomposed and distributed among the processors of the parallel machine. Each processor thus solves its own subsystem over the entire interval of interest using previous iterates from other processors as inputs. Synchronization and communication take place infrequently, and communication consists of large packets of information – entire waveforms. Because of this computation and communication structure, waveform methods are especially well suited to implementation on loosely-coupled parallel machines.

One criticism of classical waveform relaxation is that it can exhibit slow convergence. However, the application of acceleration schemes such as convolution SOR and Krylov-subspace techniques can dramatically improve the convergence rate of waveform relaxation in certain application areas.

In this paper, we review recent developments in accelerated waveform methods and present results using these methods with PVM on a cluster of workstations to solve the time-dependent semiconductor drift-diffusion equations. The experimental results demonstrate that waveform techniques are superior to standard methods in certain parallel implementations.

Boundary Integral Methods and PVM

Thomas Ericsson
ITM (Swedish Institute of Applied Math.)
Chalmers Teknikpark
S-412 88 Gothenburg
Sweden
thomas@cs.chalmers.se

Boundary integral or panel methods are standard tools for potential flow computations, e.g. in the aircraft industry. Such a problem consist of two parts. In the first one, which is embarrassingly parallel, a large number of integral-computations is necessary. The integrals are elements a_{ij} of a large, square, unsymmetric and dense matrix A of order N. The second part consists of solving a system of linear equations A x = b. On a single processor machine, a standard iterative solver like SOR is very efficient since A is strongly diagonally dominant.

An industrial code of this type, used for everyday calculations of subsonic aircraft flow on a CRAY-X/MP, has been augmented with calls to PVM 2.4 routines to run on a cluster of Sun-ELC workstations. An iterative Arnoldi-type solver was used to improve the parallelism.

A problem of N=10000 panels, giving a dense system matrix of a 100 million elements, has been solved in 14 minutes wall clock time on 83 Sparc stations. This compares very favourably with the corresponding CPU-time requirements on the CRAY.

Shot Level Parallelization of DSO using PVM 3.0

M. Kern and W.W. Symes
Department of Computational and Applied Mathematics
Rice University
Houston, TX 77251

Seismic inversion attempts to recover the structure of the earth form measurements performed (usually) on the earth surface. DSO is a code for seismic inversion, based on a variant of least-squares inversion, that tries to exploit the redundancy in seismic data. The implementation is generic in that all the optimization software is independent on the particular model being used to simulate the physics of wave propagation. It can thus be viewed as a template for optimization of multiexperiments.

The main task in the code is the repeated simulation of a basic exper- iment, under different conditions. An example is the solution of the wave equation with several right hand sides, corresponding to different seismic source positions. Because of their physical origin each of these experiments is called a shot. As they are all independent, it is quite natural to simulate them in parallel, and this leads to a coarse grain parallelism, of SPMD type.

This can be naturally implemented by a master-slave structure, in which the master dispenses the data for each shot to the slaves in turn, and then collects the results back. The master is completely model-independent, as is the slave's driver. All the model dependent code is in the lower levels of the slave-code. This can then be optimized on a per processor basis.

An immediate advantage is that parallelization is model independent, and thus the inversion software can again drive different physical models.

We present a first implementation using PVM 3.0 on a network if SUN workstations. We will review the difficulties encountered in the development process, and address some of the remaining open questions.

A Parallel Molecular Dynamics Algorithm for the Simulation of the Growth of Optical Fibers

M.J.P. Nijmeijer and A.M. Ferrenberg
Center for Simulational Physics
Department of Physics and Astronomy
University of Georgia
Athens, GA 30602

We carry out molecular dynamics simulations of a growth technique for optical fibers. The need to simulate large, three-dimensional fibers is the impetus for developing a parallel version of the algorithm. It currently runs with PVM 3 at a cluster of IBM RISC/6000 workstations. The parallelization is of the geometric type and performed in one dimension, as is most suited for the simulation of our rod-shaped fibers. The algorithm employs up to date techniques such as the use of a pair list with a linked list presorting scheme and dynamic list updating. We assess the efficiency of the parallel algorithm by accurate measurements of CPU and elapse times and compare these with the timing results for the serial program.

PVM applications under DQS

Jim Patterson
Boeing Computer Services
P.O. Box 24346, MS 7L-21
Seattle, WA 98124-0346
pattersn@espresso.rt.cs.boeing.com

This talk will address the issues involved in running PVM applications on a network of workstations under DQS (Distributed Queuing System), developed at SCRI (SuperComputing Research Institute). These issues will be introduced in the context of a Boeing project which was devised to demonstrate the feasibility and cost effectiveness of doing production computational fluid dynamics calculations on a heterogeneous cluster of workstations. The talk will focus on the support DQS presently provides for PVM applications, the enhancements made to DQS at Boeing to further support parallel applications, and the important problems still to be resolved in order for DQS to provide baseline support of parallel applications.

Scheduling PVM on Workstation Clusters using Condor

Miron Livny and Jim Pruyne
University of Wisconsin, Madison
miron@cs.wisc.edu and pruyne@cs.wisc.edu

PVM provides users a rich environment for parallel programming on heterogeneous clusters of workstations, but it does not address the problem of scheduling the available resources. The user must choose the machines to be used for the PVM job. If the user chooses a heavily loaded machine it may become a bottleneck for the entire job. Condor provides scheduling of workstations for serial jobs by monitoring all available machines, and assigning jobs to those not in use. We are developing a system which uses Condor's scheduling facility to start PVM jobs on clusters of idle workstations. The integration of scheduling with PVM has raised a number of issues regarding interaction between the parallel programming environment and the scheduler. First, an interface must be provided which allows a user to easily describe job requirements while still providing suffcient information to the scheduler. Second, extensions must be made to the programming environment so that a program can adapt to changes in resource allocation at run-time. Changes in resource allocation include both availability of new hosts and loss of hosts already in the PVM pool. We have built a prototype system which uses Condor's scheduler to select a cluster for a PVM job from a large pool of workstations, brings up PVM daemon processes on each of these machines, and starts processes on each of the workstations as requested by the user. The current system supports all PVM version 3.0 calls.

The Design and Implementation of a Queue Manager for PVM

Doug Sept and Mike Berry
Computer Science Department
University of Tennessee
Knoxville TN 37996-1301
sept@cs.utk.edu and berry@cs.utk.edu

The PVM Queue Manager (QM) application addresses some of the load balancing problems associated with the heterogenous, multi-user, computing environments for which PVM was designed. In such environments, PVM is not only confronted with the difficulties of distributing tasks among machines of variable loads, it must also contend with machines of varying performance levels in the same virtual machine. The QM addresses both of these problems using two different load balancing techniques, one static, the other dynamic. In its simplest (static) mode, the QM will initiate PVM processes for the user on demand, taking into account information such as the peak megaflop/s and actual load of each machine. In addition to the initiation of processes, the QM will also accept tasks to be completed by a specified PVM process type. These tasks are shipped to the QM where they are kept in a FIFO queue. Worker processes in the virtual machine send idle messages to the QM when they are ready for a task, and the QM ships a task to the process if there is one (of a type matching the process) in the queue. The QM also maintains a list of idle processes and will choose the *best* one for the task, should one arrive when several processes are idle. Since faster machines typically send more idle messages (and receive more tasks) than slower ones, this provides a level of dynamic load balancing for the system. Three applications have already been implemented using the QM within PVM: a Mandelbrot image generator, a conjugate-gradient algorithm, and a map analysis program used in landscape ecology applications. Benchmarks of elapsed wall-clock time comparing standard PVM versions with the QM-based versions demonstrate substantial performance gains for both methods of load balancing. When processing a 1000×1000 image, for example, the QM-based Mandelbrot application averaged 63.92 seconds, compared to 139.62 seconds for the standard PVM version in a heterogenous network of five workstations (comprised of Sun4's and an IBM RS/6000).

A Heterogeneous Environment for Distributed Computing

Karen Barrett, Christopher Carothers, Ron Chadwick, and Vernard Martin
College of Computing, Ga. Institute of Technology Atlanta, GA
High Performance Computing Group
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 0280-30332
vernard@cc.gatech.edu

This paper focuses on the design and implementation of a Heterogenous Parallel Computing environment. The environment's main goals is to to provide a consistant interface for programming distributed applications using threads that will allow the use to exploit the capabilities of each individual node of the distributed network. The hardware architecture is kept hidden from the user whenever possible. We combined the existing PVM system with the Ga Tech version of the CMU Cthreads package as the basis for our environment.

The PVM package version 2.4.2 already allows for a group of networked machines (possibly of different architectures) to send and receive information transparent of their architectures. The system uses sockets for communication across nodes even when the two communicating processes were on the same node. We modified the system to use a shared memory buffer instead to improve performance.

By using the Ga Tech Cthreads package we also allow user level multi-threaded applications to exist on each of the nodes. This allows each of the nodes to exploit parallel capabilities of that node if they are present. Each thread can send and receive messages normally although messages to threads that are not on the same node must be routed. Each node has one master thread which routes for that node. This routing is done transparently to the user.

Both the Ga Tech Cthreads package and the PVM package are portable to multiple architectures and they work on single processor and multiple processor machines. It is possible to develop code to with both single and multiple processor nodes on just a single node machine and port the code to the actual multiple processor machine with only recompilation The measurements of speedup were taken using a network of SparcStation 2's, and a Sequent S27 all connected by standard 10 Mb ethernet.