Enhancing Functionality and Performance in the PVM Network Computing System

U.S. Department of Energy Grant DE-FG05-91ER25105 (A003)
Period 1 Progress Report & Request for Continuation

Vaidy Sunderam
Department of Mathematics and Computer Science
Emory University, Atlanta, GA 30322, USA
vss@mathcs.emory.edu

1 Overview

The research funded by this grant is part of an ongoing research project in heterogeneous distributed computing with the PVM system, at Emory as well as at Oak Ridge Labs and the University of Tennessee. This grant primarily supports research at Emory that continues to evolve new concepts and systems in distributed computing, but it also includes the PI's ongoing interaction with the other groups in terms of collaborative research as well as software systems development and maintenance. The research effort at Emory has, in this first project period of the renewal (Septmeber 1994 - June 1995), focused on (a) I/O frameworks for supporting data management in PVM; (b) evolution of a multithreaded concurrent computing model; and (c) responsive and portable graphical profiling tools for PVM.

2 Overall Progress

Progress has been made in each of the three areas of focus, in the initial project period and is described below:

2.1 Parallel I/O with PIOUS

Work on the PIOUS project has progressed very well, with significant results in the deployment of software systems and in attaining high levels of scalable I/O performance over

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

networks. The PIOUS architecture was refined during the early stages of the project, and the PVM implementation was completed. In order to provide PVM applications with a uniform filesystem interface, PIOUS employs data declustering to exploit the combined file I/O and buffer cache capacities of networked computing resources, and transaction-based concurrency control, to guarantee access consistency without explicit synchronization. PIOUS consists of a set of data servers, a service coordinator, and library routines linked with client processes. An underlying transport mechanism is assumed to carry messages between client processes and components of the PIOUS architecture. PIOUS data servers are assumed to access permanent storage via a native file system. A PIOUS Data Server (PDS) resides on each machine over which files are declustered. Each PDS provides transaction-based access to the local files that represent a portion of a declustered file. A single PIOUS Service Coordinator (PSC) initiates activities within the system. Finally, each PIOUS client process is linked with library routines that translate file operations into PDS/PSC service requests. Global file structure and access policies are implemented at this level, facilitating experimentation in user interface design without requiring re-implementation of the PDS or PSC. A prototype of the PIOUS system has been developed and has been distributed to the PVM community — at the time of writing, over 150 users have retrieved, and are using, this software.

2.2 Threads-based computing with TPVM

In an attempt to improve the functionality of the PVM system for applications that do not fit the standard process model, and to explore the potential for overlapping computation and communication, we have begun a new initiative in heterogeneous concurrent computing based on lightweight threads. The TPVM (Threads-oriented PVM) system is an experimental auxiliary subsystem for the PVM distributed system, which supports the use of lightweight processes or "threads" as the basic unit of parallelism and scheduling. TPVM provides a library interface which presents both a traditional, task based, explicit message passing model, as well as a data-driven scheduling model that enables straightforward specification of computation based on data dependencies. The TPVM system comprises three basic modules: a library interface that provides access to thread-based distributed concurrent computing facilities, a portable thread interface module which abstracts the required threads-related services, and a thread server module which performs scheduling and system data management. Our design is still under development, but a prototype implementation has allowed us to perform a number of preliminary experiments. These have provided strong evidence that TPVM can offer improved performance, processor utilization, and load balance, to several application categories. Through our experiments we have also determined that the current TPVM design is not very well suited to certain types of applications, most notably highly synchronous, SPMD-style algorithms.

2.3 PGPVM

While a number of graphical tools and interfaces for PVM have been developed, both within our research group and outside, we have encountered a pressing need for one that is fast and responsive, widely portable, and relatively robust. To this end, we developed PGPVM, an enhancement package for PVM 3.3 that produces trace files for use with standard ParaGraph — a very widely used, stable, portable, and acclaimed profiling tool. PGPVM attempts to give an accurate portrayal of applications by minimizing the perturbation inherent with this type of monitoring. PGPVM does not utilize standard PVM tracing but instead its own buffered tracing techniques to provide more accurate monitoring information with much less overhead. Further, PGPVM provides a shellscript that performs some postprocessing and produces a `<file>.trf`, i.e. a standard ParaGraph trace event file. Tachyon removal and clock synchronization are performed during postprocessing when necessary. In addition to this standard visualization facility, PGPVM also incorporates an online tracing feature for gross, system-related, events. The PGPVM tools cause only minimal perturbation, produce accurate visualizations, and elucidate the usefulness of visualization in cluster environments. We have implemented these designs in usable systems, and these have been freely distributed to the PVM community — at the time of writing, nearly 150 users have retrieved, and are using, this software.

3 Research Output

3.1 Invited Talks and Professional Activities

1. “Heterogeneous Parallel and Distributed Computing with PVM”, invited tutorial, Advanced School on Computing and Imaging, Netherlands, May 1995.
2. “Heterogeneous Concurrent Computing” program chair’s address, Heterogeneous Computing Workshop, Intl. Parallel Processing Symposium, Santa Barbara, April 1995.
3. Discussions on Heterogeneous Computing With PVM, invited participant, Workshop on Cluster Computing, Caltech, Pasadena, March 1995.
4. Discussions on Heterogeneous Computing With PVM, invited participant, Second Pasadena Workshop on Systems Software and Tools for High Performance Computing, Caltech, Pasadena, January 1995.
5. “New Initiatives in Heterogeneous Parallel Computing”, keynote address, Parallel Computing and Transputers Conference, Wollongong, Australia, November 1994.
6. “Threads, I/O, and Distributed Computing with PVM”, invited talk, European PVM User’s Group Meeting, Rome, Italy, October 1994.

7. "Heterogeneous Computing with Exportable Services", invited talk, Second CNRS-NSF Workshop on Environments and Tools, Townsend, TN, May 1994.
8. "PVM: Recent Developments and New Initiatives", invited talk, 2nd PVM User's Group Meeting, Oak Ridge, TN, May 1994.

3.2 Publications:

1. A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. S. Sunderam, "Recent Enhancements to PVM", *International Journal of Supercomputer Applications*, Vol. 9, No. 2, to appear, summer 1995.
2. S. Moyer and V. S. Sunderam, "Parallel I/O as a Parallel Application", *International Journal of Supercomputer Applications*, Vol. 9, No. 2, to appear, summer 1995.
3. H. Nakanishi, V. Rego, and V. S. Sunderam, "On the Effectiveness of Superconcurrent Computation on Heterogeneous Networks", *Journal of Parallel and Distributed Computing*, Vol. 24, No. 2, pp. 177-190, February 1995.
4. B. Topol, J. Stasko, and V. Sunderam, "Integrating Visualization Support Into Distributed Computing Systems", *Proceedings — 15th International Conference on Distributed Computing Systems*, Vancouver, BC, pp. 19-26, May 1995.
5. V. S. Sunderam, "Heterogeneous Concurrent Computing with Exportable Services", *Environments and Tools for Parallel Scientific Computing*, Eds. J. J. Dongarra and B. Tourancheau, pp. 142-151, SIAM Press, May 1994.
6. A. Ferrari and V. S. Sunderam, "TPVM: Distributed Concurrent Computing with Lightweight Processes", accepted *High Performance Distributed Computing Symposium HPDC-4*, August 1995.
7. B. Topol and V. S. Sunderam, "Visualization in Cluster Environments with PGPVM", *3rd PVM Users Group Meeting*, February 1995.
8. S. Moyer and V. S. Sunderam, "Light-weight Concurrency Control in Parallel File Systems", Workshop on I/O in Parallel & Distributed Systems, *Intl. Parallel Processing Symposium*, February 1995.

4 Forthcoming Project Period Plans

During the second project period, we intend to continue to develop fully the I/O subsystem and continue further investigations and developments on the threads framework, TPVM. We will also investigate distributed object facilities for PVM that are facilitated by the existence of a threads subsystem.

4.1 The PIOUS I/O Subsystem

In the second phase of this project project, we propose to further enhance the PIOUS infrastructure by designing effective file system primitives that will enable the efficient implementation of high level file access semantics. Providing such interface primitives is critical, if the full potential of parallel filesystems are to be realized. We will begin by refining and completing the PIOUS interface architecture and data access framework; we propose an extensible and flexible, multi-layered model, that is substantially language and transport independent. Our focus in this work will be on the top three layers of the I/O interface stack: file model and access methods, caching and pre-fetching, and record definition. The first step will be to develop an appropriate file model for parallel file systems that is consistent with sequential file semantics, but provides the flexibility to impose structure semantics such as dimensionality and logical views on parallel files. Based on the file model developed, a set of access interfaces with in-built optimization capabilities will be formulated. We will subsequently implement these access methods as filesystem primitives within PIOUS. After completion, probably in the third project period, we will also incorporate caching and pre-fetching algorithms into these primitives, but in a distinct, lower level layer.

4.2 TPVM Development and Testing

In continuing to enhance the TPVM system, we will refine the current process-like semantics of the TPVM interface and also provide a service-based interface where processes "export" threads that are spawned and executed in a self-scheduled manner when the data that they depend upon become available. The first considers an application to be a collection of sub-algorithms, each of which is manifested as a collection of cooperating threads. In this model, PVM application processes register with the PVM system and export entry points into modularized functions that will execute as threads. These may be initiated either by special spawn calls or by using an RPC like model for service invocation. We will specify in detail the syntax and semantics of these primitives and implement them under PVM. After the TPVM system has been completed, the next effort along this direction will be to port the NAS parallel benchmark codes to use the multithreading facilities provided. This will involve considerable effort, since many of the codes are not currently written in a re-entrant manner. Further, since threads systems are, by definition, machine dependent, porting of TPVM to some architectures will also be necessary. Once the porting is complete,

the revised NPB codes will be executed in our standard test environments (Sun SS1's on Ethernet, IBM RS6000's on FDDI, and SGI R4400 on GigaSwitch), and measurements performed. Based upon these experiments, further tuning or refinement may be needed. After an iterative refinement and tuning process, the new results will be published and the software packaged for release.

4.3 Distributed Object Threads

In conjunction with the threads-based architecture described above, we also intend to investigate the viability of an object-based programming methodology. Object-oriented development has proved to be an effective technique for software construction; we propose the design of object toolkits for PVM that would permit distributed object-oriented programming. Essentially, we propose the construction of a set of base classes in the C++ language for common objects that are utilized in high performance computing, particularly computational fluid dynamics, such as vectors, matrices, other types of grid representations etc. These base classes would contain methods for communication, synchronization, and possibly "object migration". Applications programs would be written in terms of algorithm-specific objects that inherited attributes of one or more built-in base classes. Public defining operators in these domain-specific objects only need to invoke the appropriate member function from the built-in class in order to perform data movement and message exchange. In addition, when combined with the threads-based model, first-class objects may be manifested at run-time as threads, thereby encapsulating data as well as operations into a single object that may potentially be migrated from one host process to another. This is a natural way of programming many application classes, e.g. "particle-in-cell" codes, and if efficient implementations are possible, will contribute significantly to both the software engineering aspect as well as performance aspect of heterogeneous concurrent computing. We intend to explore and evolve this methodology and toolkit as part of this project.

*budget removed
ds*

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.