Annual Report: 1/1/97 - 12/31/97

A Scalable Distributed Multicomputer For Massively Parallel Applications DAAH04-96-1-0327

James Griffioen, James Lumpp, Rajendra Yavatkar, Raphael Finkel Dept. of Computer Science, Univ. of Kentucky

Abstract

The Unify project is investigating the design of a large-scale geographically disperse distributed shared memory system. This report describes the advances and accomplishments made during the past year. A major milestone completed this year was the release of a beta of the Unify system with full source code. The beta release includes a long list of new features, enhancements, bug fixes, auto configuration tools, and documentation. We have also designed an implemented SEClib, a scalable distributed synchronization library that uses IP multicast to disseminate synchronization information efficiently. We also implemented MxN group communication in TMTP which provides reliable data delivery and flow/congestion control to all participating hosts. We recently began adding code that will provide distributed checkpointing of long-running applications and have been designing efficient checkpointing techniques for systems that use multicast communication models. We also designed a reactive object system and have an early implementation that shows significant improvements are possible via automatic adaptation. Finally, we designed, implemented, and evaluated a variety of new distributed applications running on top of Unify, including a variety of Treadmark applications running on our Treadmark emulation library.

Unify Project Progress Report

1 Publications

The following papers describing research supported in part by this grant were submitted or appeared this past year:

- J. E. Lumpp, Jr. and W. R. Dieter, "Checkpointing with Multicast Communication," *Proceedings of the IEEE Aerospace Conference*, February 1998, pp. to appear.
- R. Sumanasekera, J. Griffioen, J. E. Lumpp, Jr., R. Yavatkar, "SEClib: Scalable Distributed Synchronization with Eventcounts," *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*, July 1998, submitted for review.
- J. E. Lumpp, Jr., K. Sivakumar, C. Diaz, and J. Griffioen, "Xunify – A Performance Debugger for a Distributed Shared Memory System," *Proceedings of the Hawaii International Conference on Systems Sciences*, January 1998.
- J. Griffioen, B. Seales, and R. Yavatkar, "Multiple Target Tracking Across Reliable Multicast Wireless Networks," *Proceedings of the Int. Soc. for Optical Engr: Sensor Fusion and Distributed Robotic Agents*, Oct 1997.
- W. R. Dieter and J. E. Lumpp, Jr., "Fault Recovery for Distributed Shared Memory Systems," *Proceedings of the IEEE Aerospace Conference, Volume 2*, February 1997, pp. 525-540.
- T. J. Rademacher and J. E. Lumpp, Jr., "High-performance Simulation of Neural Networks," *Proceedings of the IEEE Aerospace Conference, Volume 3*, February 1997, pp. 401-414.
- A. M. Levy and R. Finkel, Measure of Efficiency on Heterogeneous Machines, UK CS Technical Report 271-97, May 1997.

2 Personnel Supported

The following Masters and Ph.D students were supported during the Fall 1997 semester or will be supported during the Spring 1998 semester.

- Chris Diaz (Ph.D. Student)
- Bill Dieter (Ph.D. Student)
- Sivakumar Kuppuswamy (Masters Student, Degree Awarded December 1997)
- Amy Levy (Ph.D. Student)
- Michael Rogers (Ph.D. Student)
- Rajitha Sumanasekera (Masters Student)

- Ivan Sumner (Ph.D. Student)
- Su Wen (Masters Student)
- Babu Yama (Masters Student)

3 Scientific Progress and Accomplishments

3.1 Project Overview

The Unify research project is investigating scalable approaches for designing distributed shared-memory (DSM) multicomputers. Unify's goal is to create a massively parallel multicomputer from geographically disperse machines interconnected by wide-area networks such as large corporate/campus networks, the Internet, or evolving networks like the vBNS or Internet II. Unlike existing software packages such as PVM, MPI and others, which impose a message-passing model on the application programmer, Unify provides a distributed shared memory abstraction that hides all communication from the application programmer. Although it is widely accepted that distributed shared memory is a more convenient programming model than message passing, existing large-scale systems have turned to message passing because the overhead needed to support DSM was considered excessive and thus will not scale to massively parallel systems.

The Unify project is investigating a multicomputer design that will support DSM with minimal overhead. To support DSM across a wide-area network, Unify (1) minimizes communication, (2) hides the high latencies of a wide-area network, and (3) introduces an efficient group communication protocol that reduces traffic and reacts quickly to network errors and congestion.

3.2 Recent Accomplishments

During the past year, we have made significant progress toward achieving the objectives and milestones defined for the project. Our contributions this past year occurred in five basic areas: *enhancements to the unify design/prototype, new communication protocols, application development and experimentation, new checkpointing and debugging techniques,* and *a beta release.* The following sections briefly describe some of our accomplishments.

3.2.1 Enhancements to the Unify Design/Prototype

Over the past year, we continued to make improvements and enhancement to the Unify prototype. Example enhancements include things such as

Support for Multi-threaded Unify Workers: To provide concurrency within a worker, we added support for multithreaded workers. Threads blocked waiting for DSM data do not prevent other threads from accomplishing useful work. In addition, threads make use of fast local synchronization when synchronizing with other local threads but use Unify synchronization with cooperating with threads in other workers. Unify threads follow posix standards. We have observed significant speedups from the ability to overlap computation and communication.

New Diffing Algorithm: To minimize bandwidth consumption, Unify only transmits modifications to a segment and uses a very compact diff list representation. However, the time required to compute or apply the compact diff list representation can be substantial and for some applications was a dominate cost. To reduce the processing costs, we designed an efficient algorithm that can merge or extract multiple segment versions in approximately linear time. The algorithm also removes all processing needed to maintain local data structures from the critical path and instead performs these operations in the background during idle cycles.

New Consistency Models: We have designed the the abstractions and semantics of shared segments that support multiple concurrent consistency specifications. This allows each worker to specify its own consistency model for a segment based on its use of the segment. Worker-specific consistency can substantially reduce the communication overhead needed to support distributed shared memory.

Reactive Object Support: We designed and have an alpha implementation of a *reactive object system* which can adapt to the state of the system/environment as well as the behavior of the application. The system automatically gathers, condenses, and provides access to performance statistics needed for adaptation. We have designed a few example reactive objects that monitor the workload and dynamically adjust the algorithm to match the current environment. We are currently evaluating the design to determine the effectiveness of this technique. Our initial test results are very encouraging.

3.2.2 New Communication Protocols

This past year, we developed a new communication protocol that provides efficient synchronization in a large distributed environment, and we implemented MxN communication in TMTP.

SEClib: We developed a scalable event count library that uses best-effort multicast communication to maintain synchronization state information. In the expected case where no errors arise, SEClib incurs no overhead at all. When errors arise, SEClib only corrects errors that will affect the synchronization of processes, all other errors are left uncorrected. Errors that are corrected are corrected locally whenever possible. We have implemented SEClib as a standalone library which we use as the synchronization method in Unify. Initial tests show speedups of as much as an order of magnitude over PVM and a factor of two over TMK.

TMTP: TMTP provides reliable multicast communication for Unify. Accomplishments this past year include improved flow and congestion control in TMTP and support for MxN communication. We have also packaged up TMTP as a standalone library and are completing the testing of the alpha release.

3.2.3 Application Development and Experimentation

Over the past year we ported various DSM applications to Unify and we wrote several new DSM applications that test various aspects of the Unify system. Example applications that we have developed based on Unify's relaxed DSM model range from a adaptive distributed SOR program to a distributed merge sort to a distributed mandelbrot generator and graphical display.

We also completed the implementation of our Treadmarks emulation library. Using the library we have been able to run a wide range of Treadmarks applications on Unify. We have just begun a detail performance comparison over high-speed wide area networks like SEPSCoR and the vBNS. Our initial results are very encouraging.

3.2.4 Debugging/Checkpointing

We completed the implementation of a graphical performance monitoring tool called XUnify. The tool helps the programmer visualize the performance of an application while it is running and can save trace information for postmortem visualization. Xunify is unique in the fact that it provides performance information at three different levels: (1) the DSM programming-model level, (2) the logical-message send and receive calls, and (3) the low-level transport protocol details. The three levels allow application writers to quickly identify inefficiencies due to inappropriate constructs or algorithms at the programming-model level, or alternatively to identify unique bottlenecks that arise from the particular network/runtime environment. Xunify is included with the standard Unify distribution and is currently used to debug both the Unify libraries and Unify applications.

We have also begun investigating the problem of checkpointing in a distributed system based on multicast communication. As a starting point for this work, we developed a simple coordinated checkpointing algorithm based on multicast and implemented it in Unify. For the actual checkpoint we are using standard freely available packages that checkpoint a process' entire address space. We also developed our own extensions to checkpoint programs that use multiple memory mappings which is needed for shared libraries and programs that use mprotect, signal handlers, timers, alarms, and pending signals (such as Unify).

3.2.5 Unify Public Release

Using the enhanced prototype systems and example applications described above, we ran a variety of tests to find lingering bugs and convince ourselves that the system was stable enough to release. We then packaged up the prototype system including the Unify libraries, the Xunify debugger, SEClib, the Treadmarks emulation library, and a variety of example applications. As part of this process we wrote installation and API documentation and also tools to aid in installation, configuration, and unify program development. The resulting beta version has been released to the general public and is available on the web at `http://www.dcs.uky.edu/~unify`. As new features or bug fixes occur, newer versions will continue to be placed on the web site. The current version runs on Solaris systems.

3.3 Research Plan for 1998

The following list some of our research and implementation goals for the next reporting period.

- continue the design and implementation of our reactive object system
- continue the design and implementation of our concurrent multiconsistency models
- examine the problem of supporting large numbers of concurrent multicast sessions in TMTP and making further improvements to TMTP's congestion avoidance algorithms.

- implement a checkpointing system that supports fault-tolerance and interactive debugging of Unify applications.
- fix bugs and improve the stability, usability, and documentation of the prototype system
- encourage and help fellow colleagues with computational intensive application to use the Unify beta release.
- complete the evaluation of the TMK emulation library and comparison with native TMK
- reimplement existing and benchmark application to use Unify's relaxed consistency abstractions and compare the performance with conventional approaches
- port the system to Linux for use on a 32 node pentium cluster we recently purchased

4 Technology Transfer

In addition to publishing our findings in conference proceedings, journals, and technical reports, we will continue to make our prototype implementations and source code available on the web. We will also continue to improve the documentation and our set of example applications, which will hopefully reduce the learning curve needed to start using Unify. We have also set up a problem-tracking system on the Web for users of Unify to report problems. We have also set up an email list for Unify users to ask questions and provide helpful insights to other Unify users.