The following paper was originally published in the
Proceedings of the USENIX Windows NT Workshop
Seattle, Washington, August 1997

Millipede: a User-Level NT-Based Distributed Shared Memory System

with Thread Migration and Dynamic Run-Time Optimization of Memory References

Ayal Itzkovitz, Assaf Schuster, Lea Shalev
Computer Science Department
Technion, Haifa

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

Millipede: a User-Level NT–Based Distributed Shared Memory System with Thread Migration and Dynamic Run-Time Optimization of Memory References

http://www.cs.technion.ac.il/Labs/Millipede

Ayal Itzkovitz Assaf Schuster Lea Shalev
Computer Science Department, Technion, Haifa

Abstract

MILLIPEDE is an all user mode, no kernel-patches, "add on" software tool for standard corporate environments, that takes advantage of idle system resources and efficiently utilizes idle processor time in available distributed environments of personal workstations. MILLIPEDE presents to the user a powerful virtual parallel machine which abstracts away the underlying hardware configuration. In this way MILLIPEDE supports mapping of the applications to dynamically varying levels of parallelism according to both changes in the underlying hardware and changes in the application requirements.

MILLIPEDE is multi-threaded, thus taking full advantage of SMPs. MILLIPEDE provides a true distributed shared memory with several coherence protocols (and a flexible mechanism for easy inclusion of new ones) and dynamic thread migration [3]. MILLIPEDE studies the memory access pattern and optimizes the locality of memory references by adapting the thread distribution accordingly [5].

MILLIPEDE supports several of the more liberal parallel programming paradigms [2]. Currently we support PARC (which allows for example nested parallelism and barriers among sibling activities), PARC++ (which is as flexible as C++ except for additional parallelizing constructs), the SPLASH macros (which we had to adapt to Windows-NT and to the multi-threaded concept), and of course for best speedups one can use directly the MILLIPEDE job manager library. We are working on the implementation of PARFORTRAN90 and JAVA (for which we came up with a new definition of the JVM memory behavior, and with an efficient algorithm for distributed garbage collection [4]).

In order to support many different programming languages on top of a single virtual parallel machine we developed MILLIPEDE Job Event Control (MJEC) [2]. MJEC can be used to efficiently implement a variety of synchronization and communication protocols (e.g., PARC in about 250 lines of code).

MILLIPEDE is fully implemented at the Technion, Haifa, using the Windows-NT operating system (previous version on MACH [1]). We refer the reader to our WWW site (see above) for online versions of the papers, and for downloading a distribution of MILLIPEDE.

References

[1] R. Friedman, M. Goldin, A. Itzkovitz, and A. Schuster. Millipede: Easy Parallel Programming in Available Distributed Environments. *Software: Practice & Experience*, 1997. (To Appear). Also Technion/LPCR/TR, #9506, November 1995.

[2] A. Itzkovitz, A. Schuster, and L. Shalev. Millipede: Supporting Multiple Programming Paradigms on Top of a Single Virtual Parallel Machine. In *Proc. HIPS Workshop*, Geneve, April 1997.

[3] A. Itzkovitz, A. Schuster, and L. Shalev. Thread Migration and its Applications in Distributed Shared Memory Systems. *The Journal of Systems and Software*, 1997. To appear (See also Technion TR LPCR-#9603).

[4] D. Kogan and A. Schuster. Collecting Garbage Pages with Reduced Memory and Communication Overhead. In *Proc. European Symposium on Algorithms*, Graz, September 1997.

[5] A. Schuster and L. Shalev. Access Histories: How to Use the Principle of Locality in Distributed Shared Memory Systems. Technical Report #9701, Technion/LPCR, Jan 1997. Submitted for Publication.