NWO-P: Parallel Simulation of the Alewife Machine

Kirk Johnson and David Chaiken*
MIT Laboratory for Computer Science
545 Technology Square, Room NE43-635
Cambridge, Massachusetts 02139

Alan Mainwaring†
Thinking Machines Corporation
Cambridge, Massachusetts 02142

1 Introduction

Detailed simulation of a large-scale multiprocessor is a computationally intensive task. In fact, the computational requirements of such simulations are often large enough such that only moderate machine sizes (perhaps up to 64 processors) with a small amount of memory per node (one or two megabytes) can be tractably simulated on today's fastest desktop workstations. Fortunately, the systems used for these types of simulation are often amenable to parallelization for execution on available multiprocessor systems; doing so can yield substantial performance improvements.

We have ported a version of our detailed Alewife machine simulator to Thinking Machines' CM-5 family of multiprocessors. This paper provides a brief overview of that effort, sample results indicating the performance of the current implementation, and a few comments about future work. The CM-5 port of our simulator has been operational since June 1992 and has proved invaluable, especially for running simulations of large Alewife systems (64 to 512 nodes).

2 Alewife Overview

Alewife is an experimental distributed-memory multiprocessor under construction at the MIT Laboratory for Computer Science [1]. As shown in Figure 1, each Alewife node consists of a Sparcle processor [2], a floating-point coprocessor, several megabytes of DRAM, a 64-kilobyte unified instruction/data cache, the local portion of a two-dimensional mesh interconnect, and a Communications and Memory-Management Unit (CMMU) that serves as both memory and network interface.

We have implemented a detailed sequential simulator for Alewife that runs on SPARC- and MIPS-based workstations; this simulator is called NWO [3]. In addition to providing cycle-by-cycle simulation of the processing elements and flit-by-flit modeling of the network interfaces, NWO strives to model internal workings of the actual Alewife CMMU [4] as closely as possible.[1]

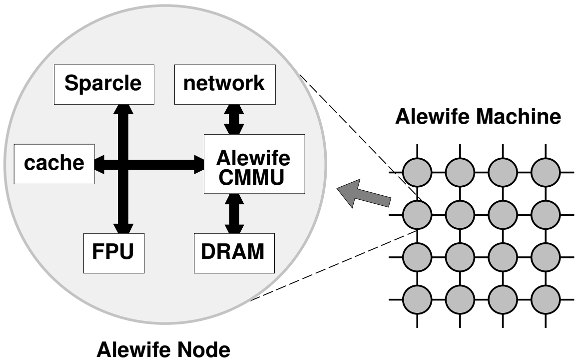

Figure 1: Alewife hardware organization.

Because of the level of detail simulated, NWO runs quite slowly—on the order of a few thousand node-cycles per second on our fastest desktop workstations (*e.g.*, SPARCstation 10/30s or DECstation 5000/240s equipped with at least 64 megabytes of physical memory).

NWO is implemented as a single process that simulates all nodes in lock step, alternating between (i) looping through all simulated nodes allowing each one cycle of forward progress and (ii) simulating one cycle of the interconnection network's behavior.

3 The Parallel Simulator

The fact that our detailed Alewife simulator is intended to be a vehicle for functional debugging and verification of the Alewife CMMU design imposes two constraints on our parallel simulator implementation:

- The parallel simulator must behave deterministically.
- The parallel simulator must have the same semantics as NWO—given the same input, both simulators should produce exactly the same outputs at exactly the same time.

These constraints dictate a parallel simulator implementation that, like NWO, simulates all nodes in lock step.

* E-mail: {tuna, chaiken}@lcs.mit.edu. The research reported on herein was supervised by Professor Anant Agarwal. It was funded in part by NSF grant # MIP-9012773, in part by DARPA contract # N00014-87-K-0825, and in part by a NSF Presidential Young Investigator Award.

† E-mail: amm@think.com

[1] We simulate at this level of detail in order to verify CMMU functionality as much as possible without resorting to *extremely* slow switch-level simulation of the actual CMMU design. See "A Hybrid Simulation System for Multiprocessor Hardware Verification and Software Development" by Chaiken and Kubiatowicz elsewhere in this proceedings.

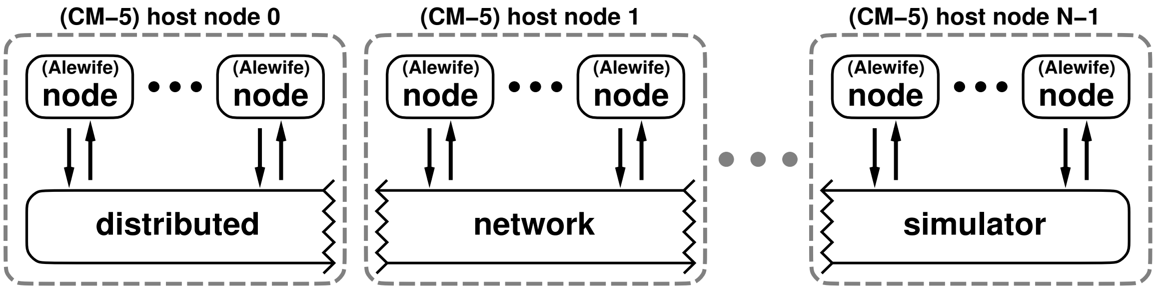

Figure 2: Parallel simulator (NWO-P) organization; each dashed box represents a process run on one of the host nodes (see Section 3).

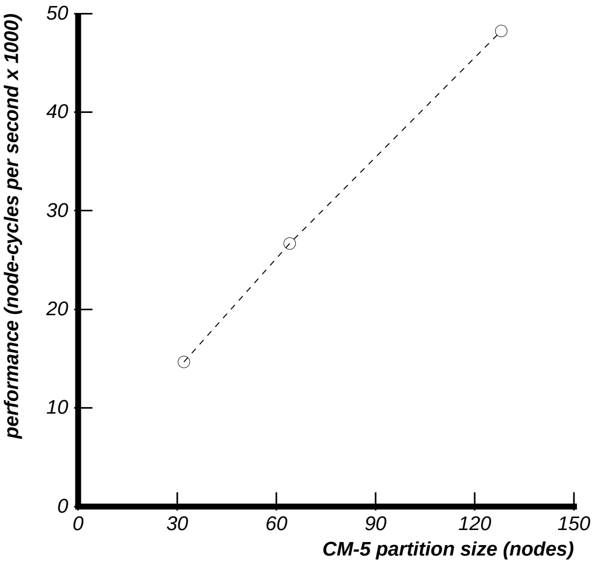

Figure 3: NWO-P performance on the CM-5.

The parallel simulator, called NWO-P (for NWO-P*arallel*), is implemented in a manner that allows the vast majority of the NWO code to be reused. As depicted in Figure 2, NWO-P statically partitions the simulated (Alewife) nodes amongst the host (CM-5) nodes. In effect, each host node runs a single process that simulates a small Alewife machine using the NWO code, except the network simulator has been modified to keep the node simulations in synchrony and effect any necessary inter-host-node communication using the CM-5 communication and synchronization facilities.

Figure 3 provides an indication of the performance of the current NWO-P implementation. Each data point represents the performance of NWO-P (measured in simulated node-cycles per second) averaged over four small simulations when each CM-5 processor is simulating a single Alewife node. The performance shown for a 32-node CM-5 is roughly five to six times that of NWO. On 128 nodes, NWO-P runs between 16 and 20 times faster than NWO. Increasing the ratio of simulated processors to host processors can improve performance somewhat; for the same set of simulations mentioned above, simulating two Alewife nodes per CM-5 node improves performance an average of 15 percent.

4 Future Work

Because the CM-5 message-passing libraries that were available when we started this effort (CMMD 2.x) simply did not provide much of the functionality we needed, the current NWO-P implementation utilizes the CMNA library, which provides register-level access to the CM-5 network interface. As such, it can be quite difficult to maintain and debug. The most recent release of the message-passing libraries (CMMD 3.0) appears to address many of the problems of earlier versions. We expect to reimplement NWO-P on top of CMMD 3.0 sometime this summer; doing so should improve code maintainability and extensibility substantially.

While the increases in simulation speed afforded by the current NWO-P implementation have proved invaluable, speedups of only 16 to 20 on 128 nodes are a bit disappointing; we would like to gain a better understanding of what factors are limiting performance to this level.

5 Acknowledgments

We would like to thank Dave Douglas and Thinking Machines Corporation for their generosity and assistance in porting our simulation system to the CM-5.

References

[1] Anant Agarwal, David Chaiken, Kirk Johnson, David Kranz, John Kubiatowicz, Kiyoshi Kurihara, Beng-Hong Lim, Gino Maa, and Dan Nussbaum. The MIT Alewife Machine: A Large-Scale Distributed-Memory Multiprocessor. Technical Report MIT/LCS/TM-454, MIT Laboratory for Computer Science, June 1991.

[2] Anant Agarwal, John Kubiatowicz, David Kranz, Beng-Hong Lim, Donald Yeung, Godfrey D'Souza, and Mike Parkin. Sparcle: An Evolutionary Processor Design for Multiprocessors. *IEEE Micro*, June 1993. (to appear).

[3] David Chaiken. NWO User's Manual. Alewife Memo No. 36, MIT Laboratory for Computer Science, June 1993.

[4] John Kubiatowicz. User's Manual for the A-1000 Communications and Memory Management Unit. Alewife Memo No. 19, MIT Laboratory for Computer Science, January 1991.