Communication Efficient Shared Memory

Miguel Castro *

M.I.T. Laboratory for Computer Science
Cambridge, MA 02139
email: castro@lcs.mit.edu

1 Introduction

In message passing systems all communication is explicit and therefore the programmer can tune applications to reduce the number of messages and the amount of data exchanged. On the other hand, distributed shared memory systems [4] provide programmers with little or no control over communication but offer a convenient shared memory programming model. This model makes it easier to develop a correct program but makes it difficult to improve the program's performance.

DiSOM [3] is a new software based distributed shared memory system, which supports intra and inter-application sharing in heterogeneous networks of multiprocessor workstations. The system offers a simple shared memory programming model where concurrent threads synchronize explicitly and communicate by executing operations on shared objects. However, the programmer can still control *when* to communicate, *which* information to send and *where* to send the information. Therefore, programmers can tune communication to improve application performance.

2 Efficiency and Flexibility

The programmer can choose when to communicate because the system implements an update protocol which ensures that communication only occurs at synchronization interactions. The combined use of a variant of the *entry consistency* [1] memory model and object-oriented language techniques allows the programmer to select which information to send. For example, each class can redefine the operations that are executed by the system to marshall and unmarshall its instances. The lower layers of the system are structured as a toolkit that ensures programmers can customize system actions with a minimum programming effort.

The set of synchronization primitives offered by the system lets the programmer choose where to send the data. The implementation of read-write locks *pulls* information from the releaser to the acquirer of the lock, a variant of binary semaphores can be used to *push* information to a set of threads and barriers allow every thread to broadcast information to all others.

Finally, the programmer can choose between the data-shipping mechanism offered by the synchronization primitives and a function-shipping mechanism offered by a transparent remote object invocation scheme. The remote object invocation scheme is consistently integrated with the data-shipping memory coherence protocol.

The entry consistency memory model was introduced by Midway [1] but no significant results were ever published showing the advantages of this model. Unlike Midway, DiSOM uses semaphores, remote object invocation and object-oriented language techniques to provide programmers with more control over communication. We defined the first formalization of an entry consistency memory model and presented the first comparative study between an entry consistency based system and a lazy release consistency based system [2]. The study compares the performance of DiSOM and TreadMarks [4], a state-of-the-art distributed shared memory system that implements lazy release consistency. The study is based on four shared memory parallel applications, matrix multiply, successive over-relaxation (SOR), traveling salesperson and Water, ported from TreadMarks.

The performance results presented in [2] show that DiSOM sends between 2 and 26 times less messages than TreadMarks and a comparable amount of data with 8 processors. This is achieved at the expense of a slight increase in programming complexity. Furthermore, the results show that without the extensions introduced by DiSOM an update based implementation of entry consistency, such as Midway, can perform significantly worse than TreadMarks.

DiSOM also offers the possibility of sharing objects between different applications, it supports sharing between different architectures and it exploits shared memory multiprocessors efficiently by allowing multiple threads per-process. We are unaware of another distributed shared memory system that combines this functionality.

*Faculty advisor: Paulo Guedes, INESC, Rua. Alves Redol n.9, 1000 Lisboa, Portugal, email: Paulo.Guedes@inesc.pt. The author is supported by a PRAXIS fellowship.

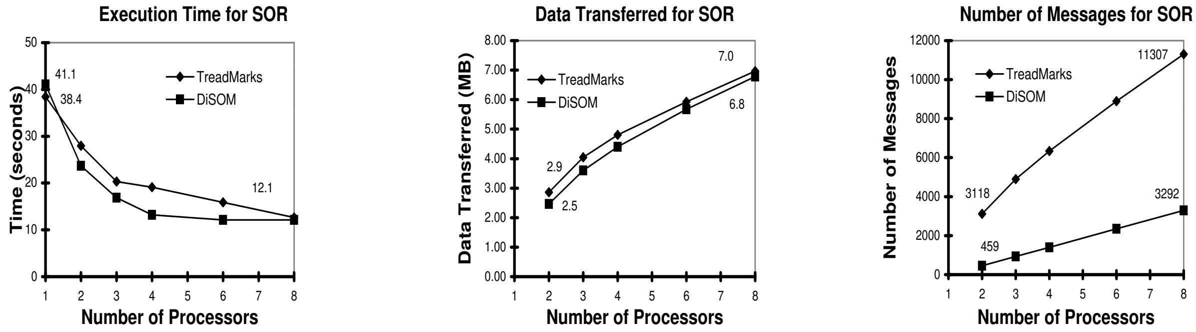

Figure 1: Performance of SOR with a 1024×512 floating point matrix and 106 iterations. The measurements were performed on a cluster of eight SPARCstations 10/30 connected by an otherwise idle 10 Mbps Ethernet network. All the values presented are averages of three or more runs.

3 Fault tolerance

In a network of workstations, the probability of a failure during an application's execution increases with the execution time and the number of workstations used. If no provision is made for handling failures, it is unlikely that long running applications will terminate successfully. Therefore, we have defined an efficient checkpoint protocol that allows transparent recovery from single node failures and, in some cases, from multiple node failures [5]. The protocol keeps a distributed log of shared data accesses in the volatile memory of the processes, taking advantage of the independent failure characteristics of workstation clusters. Periodically, or whenever the log reaches a high-water mark, each process checkpoints its state, independently from the others. The protocol needs no extra messages during the failure-free period, since all checkpoint control information is piggybacked on the memory coherence protocol messages.

4 Evaluation

Figure 1 compares the performance of a red-black SOR application running on DiSOM and TreadMarks. The interior elements of the matrix are initialized to zero and each iteration computes a new value for every matrix element which is the average of its neighbors. In the first iterations the values of interior elements remain zero, therefore TreadMarks page diffing mechanism does not send updates for these elements. In spite of this, DiSOM still sends less data than TreadMarks due to the false sharing problem. With a large number of iterations this problem would make TreadMarks send between 2 and 4 times more data than DiSOM depending on the data layout. Our system sends approximately 3 times less messages than TreadMarks because it uses an update protocol whereas TreadMarks uses an invalidate protocol. Semaphores allow DiSOM to use an update protocol and send the minimum number of messages per-iteration without sending more data than required.

5 Conclusions

We presented a distributed shared memory system that provides programmers with the control over communication of message passing while retaining a convenient shared memory programming model. This is achieved by using an update based implementation of entry consistency, efficient semaphores, remote object invocation and allowing programmers to customize system actions on a per object basis. Together these techniques provide a consistent framework for the development of efficient parallel and distributed applications.

The system is currently running on a network composed of several SPARCstation 10, i486 PC, DEC Alpha and a shared-memory multiprocessor Sun SPARCcenter 2000 with 10 processors. It supports sharing of objects across all machines.

References

[1] B. N. Bershad, M. J. Zekauskas, and W. A. Sawdon. The Midway Distributed Shared Memory System. In *Proceedings of the 93 COMPCON Conference*, pages 528–537, February 1993.

[2] Miguel Castro, Paulo Guedes, Manuel Costa, and Manuel Sequeira. Efficient and flexible object sharing. Technical report, INESC, Lisboa, Portugal, March 1995. Submitted for publication.

[3] Paulo Guedes and Miguel Castro. Distributed Shared Object Memory. In *Proceedings of the Fourth Workshop on Workstation Operating Systems (WWOS-IV)*, Napa, CA., October 14-15 1993. IEEE Computer Society Press.

[4] P. Keleher, A. Cox, S. Dwarkadas, and W. Zwaenepoel. Treadmarks: Distributed Shared Memory on Standard Workstations and Operating Systems. In *Proceedings of the 1994 Winter USENIX Conference*, January 1994.

[5] Nuno Neves, Miguel Castro, and Paulo Guedes. A Checkpoint Protocol for an Entry Consistent Shared Memory System. In *Thirteenth ACM Symposium on Principles of Distributed Computing (PODC)*, August 1994.