A DSM Speedup Comparison: TreadMarks, JIAJIA and Nautilus

Mario Donato Marino * Geraldo Lino de Campos
Computer Engineering Department - PCS
University of Sao Paulo
Sao Paulo, SP, Brazil

Abstract: *Nautilus is a Multithreaded Distributed Shared Memory system based on scope consistency. Its multithread implementation disallows the use of SIGIO signals in order to minimize the context switch of traditional processes. This paper shows the speedups of some benchmarks submitted to Nautilus, being it compared with two other DSMs: TreadMarks and JIAJIA. The benchmarks evaluated in this study are: LU (kernel from SPLASH II) and SOR (from Rice University).*

Keywords: distributed shared memory, DSM

1 Introduction

In the last 6 years the research on *Distributed Shared Memory* (DSM)[7] area has great diffused, with the development of a large number of consistency models and DSM systems. [1] has classified the DSM evolution in two generations, the first one characterized by a big number of consistency messages and the sequential consistency; the second one, by a big reduction of the number of consistency messages and by the adoption of a release consistency model. One can exemplify the first generation with the Ivy[6] and the second generation with Munin[2], Quarks[1], TreadMarks[3], JIAJIA[4], CVM[9] and Nautilus[5], introduced in this paper.

Commonly, DSM comparisons base on simulations, rather than confronting execution results; for example, two different DSM systems over a computer network. The main goal is to evaluate Nautilus in an accurate way, confronting it with others well known DSM systems, TreadMarks and JIAJIA, executing them on the same network, machines and operating systems; once they are evaluated under the same conditions, the results of this comparison would tend towards accurate and fair comparisons.

One of the contributions of this paper is to show the speedups of three second generation DSM systems being executed on the same network of computers, because comparing executions is more accurate and more correct than comparing simulations of the DSMs. In addition, as there are papers[3, 8] that are using networks with different operating systems, to equalize the comparison, these three DSM systems are compared on a PC computer network with a free operating system. So, with an ordinary hardware, operating system, and DSMs system used throughout the academic community, it is guaranteed that the network, the computers and the operating system are the same to do an homogeneous and fair comparison. For a meaningful evaluation of the Nautilus DSM system, two of the most important DSM systems, that have been used by several research groups in the scientific community were chosen to be compared against the Nautilus. So, as a main contribution of this paper TreadMarks, JIAJIA and Nautilus speedups are compared for different benchmarks.

The comparison of Nautilus speedups with

*Supported by CNPq no.142753/97-1 .

TreadMarks and JIAJIA speedups is done by applying different benchmarks: LU (kernel from SPLASH II) and SOR (from Rice University). The environment of the comparison is a 8PC's network [1] interconnected by a fast-ethernet shared media. The operating system used is Linux (2.x).

2 TreadMarks

The consistency model used by TreadMarks is the lazy release consistency[3], so the propagation of the medifications ocurred during a critical section is delayed until the next acquire. By using multiple writer protocols and the lazy release consistency model, the speedups of TreadMarks are very known, as a consequence, it becomes one of the most used DSM systems. Let's summarize TreadMarks features: i) lazy release consistency and its variations[3], minimizing the number of consistency messages; ii) multiple writer techniques of Munin[2]; iii) IBM SP2, Sun Sparc, PCs; iv) Solaris, Linux 2.x; v) UDP protocols to minimize network protocols overhead.

The speedups of TreadMarks made it the main DSM used by the scientific community as a reference of optimal speedups. Thus, it makes sense to compare Nautilus, a new DSM system, with TreadMarks, in order to have an accurate evaluation of its performance. The efficiency of TreadMarks is mainly derived from its lazy release consistency model. The major drawback of adopting this model is the high need of memory to store the diffs[2] all over the user's application execution. So, the sizes of the benchmarks used to evaluate the speedups of the DSM system can be compromised if there is not enough memory to execute them or if the operating system does swap.

[1] Supported by Finep/Recope.

[2] diffs: codification of the modification suffered by a page during a critical section

3 JIAJIA

JIAJIA[4] is another important DSM system, which uses scope consistency, that can be interpreted as an intermediary consistency model between release consistency and lazy release consistency or also be interpreted as a kind of implementation of release consistency. According to this model, diffs are transmitted in each critical section to maintain the consistency. So, JIAJIA, which uses scope consistency model, only sends consistency messages to the owner of the pages, invalidating them in the acquire primitive.

Let's summarize JIAJIA features: i) scope consistency[4] home based, minimizing the number of consistency messages through the net; ii) multiple writer techniques; iii) data distribution possible to be chosen by the user: the user can choose where the shared data is located (over the network nodes); iv) primitives compatible with TreadMarks; v) IBM SP2, Sun Sparc, PCs; vi) AIX, Solaris, Linux 2.x; vii) UDP protocols, minimizing network protocols overhead.

The main objective of JIAJIA[4] is to be as simple as possible in order to minimize overheads of diff creation and diff storage and also minimize the number of consistency messages through the net. Concluding, the most interesting feature of JIAJIA is its simple ideas: home based, so the diffs are transmitted only to the owner of the pages and not to several nodes, minimizing the number of messages through the net; the user knowing the behavior of his program, chooses a data distribution which is more apropriated, allowing better speedups.

4 Nautilus

Nautilus is the first multithreaded DSM system implemented on top of a free unix platform that uses the scope consistency model, because: 1) As of now, there are no multithreaded versions of Treadmarks that can be executed on Linux 2.x, but only a process-based version; 2) JIAJIA is a DSM system based on scope consistency, but it is not im-

plemented using threads; 3) CVM[9] is a multithreaded DSM system, but uses lazy release consistency and as of now, it does not have a linux based version.

Let's summarize Nautilus features: i) scope consistency (possibly interpreted as a kind of release consistency implementation) only sending consistency messages to the page owners and invalidating pages in the acquire primitive; ii) multiple writer techniques; iii) multithreaded DSM: threads to minimize the switch context; iv) no use of SIGIO signals(which notice the arrival of a network message); v) minimization of diffs creation; vi) primitives compatible with TreadMarks, Quarks and JIAJIA; vii) network of PCs; viii) operating under Linux 2.x; ix) UDP protocols.

To improve the speedup of the applications submitted, Nautilus introduces two news: i) multithreaded implementation; ii) lower overhead of receiving messages when they arrive. The multithreaded implementation of Nautilus permits: i) minimization of context switch; ii) no use of SIGIO signals. Nautilus is based in the following idea: the owner nodes of the pages do not need to send the diffs to other nodes, according to the scope consistency model. So, diffs of pages written by the owner are not created, what it's believed to be more efficient than the lazy diff creation of TreadMarks. The major part of all DSM systems created until today implemented on top of an Unix platform uses SIGIO signals to activate a handler to take care of the arrival of messages which come from the network. Some examples of DSMs that use the SIGIO signal are TreadMarks and JIAJIA. The use of a multithreaded implementation permits to eliminate this overhead to take SIGIO signals and activate its respective handler, in all arrivals of messages. Avoiding the use of SIGIO signal and the handler system call minimizes the overheads of the system.

On the same way that TreadMarks and JIAJIA do, also Nautilus is worried about network protocols. So, it also uses UDP protocol to minimize overheads. As TreadMarks and JIAJIA do, Nautilus also is worried about synchronization messages. To minimize the number of messages, the synchronization messages would also carry a consistency information.

5 Experimental Evaluation

The evaluation programs of this study are executed on top of *Nautilus* on the following network of PCs: i) nodes: K6 - 233 MHz (AMD), 64 MB of memory and 2 GB IDE disk; ii) interconnection: a hub and fast ethernet cards (100 Mbits/s). The network above was completely isolated from any other external networks. The operating system used was Linux Red Hat 5.0.

Due to the limitation of the TreadMarks version used:

1) the applications were executed and the speedups measured using *Nautilus* running on up to *8 nodes*;

2) bigger input sizes for both benchmarks LU and SOR, *were not possible* to be evaluated.

Only *a few results* were obtained with the current *JIAJIA* version (possibly an implementation problem).

The evaluation programs are LU (SPLASH-II) and SOR (from Rice University):

1)"*The LU kernel from SPLASH II factors a dense matrix into the product of a lower triangular and upper triangular matrix. The NxN matrix is divided into an nxn array of bxb blocks (N = n*b) to exploit temporal locality on submatrix elements. The matrix is factored as an array of blocks, allowing blocks to be allocated contiguously and entirely in the local memory of processors that own them.* "[4] The N used in the evaluation is $N = 1792$.

2) "*SOR from Rice University solves partial differential equations (Laplace equations) with a Over-Relaxation method. There are two arrays, black and red array allocated in shared memory. Each element from red array is computed as an aritmethic mean from black array and each element from black array is computed as an aritmethic mean from red array. Communication occurs across the bound-*

ary rows on a barrier".[4] The sizes of red and black matrix used are *1792*. The number of iterations is *10* .

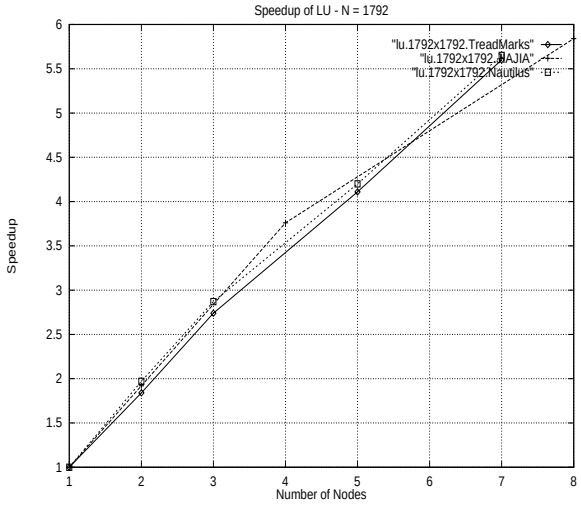

Figure 1: speedup of SOR(1792)

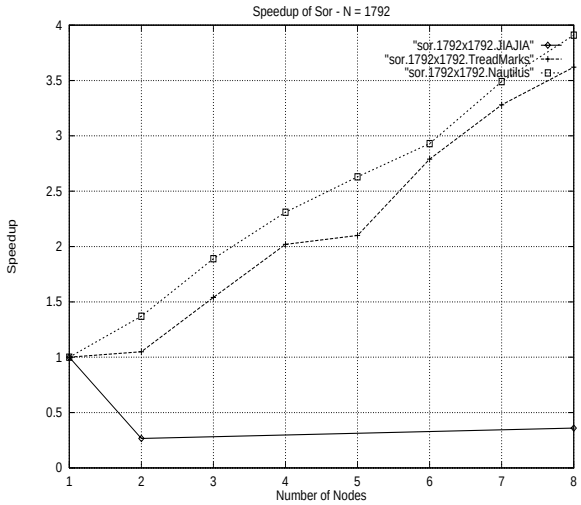

Figure 2: speedup of LU (N=1792)

5.1 LU - Result Analysis

"LU is a kernel from SPLASH2 benchmarks that has a rate computation/communication $O(N^3)/O(N^2)$, which increases with the problem size N. The nodes frequently synchronize in each step of computation and none of the phases are fully parallelized ".[4]

The figure 1 shows good LU speedups for the three DSMs. With a number of nodes less than 3 nodes, the speedups of the three DSM systems are very similar. Nautilus is faster than TreadMarks and JIAJIA, except for 4 and 5 nodes, when JIAJIA is 6.50% faster than the others. Percentually, Nautilus outperforms TreadMarks about 3% and TreadMarks outperforms JIAJIA around 5.67%. The best speedup of Nautilus occurs because with this N value, due to lazy release consistency model adopted by TreadMarks, too much diffs are stored causing the swapping of the operating system. And, although JIAJIA and Nautilus use the same memory consistency model, the use of multithreading and the avoidance of SIGIO signals makes Nautilus faster than JIAJIA.

5.2 SOR - Result Analysis

The SOR (Rice University) solves Laplace partial equations. For a number of iterations, it has two barriers for each iteration, and communication occurs across boundary rows on a barrier. The communication does not increase with the number of processors and the relation communication/computation reduces in the same proportion as the size of problem increases.

The figure 2 shows good speedups for TreadMarks and Nautilus. The speedups of JIAJIA are very unusual. Therefore, any related speedups are not considered for SOR analysis. For 2 to 8 nodes Nautilus outperforms TreadMarks. In terms of percentage, the speedup difference reaches up to 25.23%. Justifying the best speedups of Nautilus and the speedup difference compared to TreadMarks: with a low number of processors, the speedup difference between Nautilus and TreadMarks is higher, because of the higher cost of lazy release consistency to maintain the directory on a page fault. It is needless to alloc twins and diffs when a page is written in its owner (node), which decreases the overhead, and also the data distribution, multithreading and the avoidance of SIGIO signals, improve the Nautilus speedup.

6 Conclusion

This paper confronts the speedups of TreadMarks, JIAJIA and Nautilus, used at present and compared on the same environment, with the same computers, network and operating system. It was possible to notice the behavior of different benchmarks which have several features under three different DSMs and their respectives consistency memory models. For all the applicatives tested in this experiment, JIAJIA has the worse speedups. For all the applicatives Nautilus has the best speedups. For LU applicative, Nautilus is up to 5.67% faster than TreadMarks. For the SOR applicative, is 25.23% faster than TreadMarks. As shown, Nautilus has good speedups, comparable to other well-known DSMs, surpassing some of them depending on the application. The use of multithreading, good data distribution (choice of the page owners, minimizing the diffs sending) and the avoidance of SIGIO signals also help to improve Nautilus speedup.

The lazy release consistency model (TreadMarks) and the scope consistency model (JIAJIA and Nautilus) presented good speedups for the benchmarks evaluated in this paper. Depending on the size of the input parameters and the application evaluated, it was possible to verify how the application performs under the different DSMs and how the consistency models used by these DSMs influence their speedups.

Since only a few results were obtained with the current version of JIAJIA, it will be possible to compare the speedups of an improved version of this DSM with TreadMarks and Nautilus. And if the computers' memory had been reduced, very probably TreadMarks would have presented worst results in all benchmarks, because of its lazy release consistency model and its need to store the diffs.

References

[1] Carter J. B., Khandekar D., Kamb L., *Distributed Shared Memory: Where We are and Where we Should Headed*, Computer Systems Laboratory, University of Utah, 1995.

[2] Carter J. B., *Efficient Distributed Shared Memory Based on Multi-protocol Release Consistency*, PHD Thesis, Rice University, Houston, Texas, September, 1993.

[3] Keleher P. , *Lazy Release Consistency for Distributed Shared Memory*, PHD Thesis, University of Rochester, Texas, Houston, January 1995.

[4] Hu W., Shi W., Tang Z., *JIAJIA: An SVM System Based on a new Cache Coherence Protocol,* technical report no. 980001, Center of High Performance Computing , Institute of Computing Technology, Chinese Academy of Sciences, January, 1998.

[5] Marino M. D.; Campos G. L.; *Evaluation of The Traffic on the Nautilus DSM System Using Updates: Ratio Among the Number of Messages and the Mean Size of the Consistency Messages*, XXIV Latin American Conference of Informatics (CLEI'98), Memories, October 1998, Vol. 1, pp. 325-333.

[6] Li K, *Shared Virtual Memory on Loosely Coupled Multiprocessors*, PHD Thesis, Yale University, 1986.

[7] Stum M. , Zhou S. , *Algorithms Implementing Distributed Shared Memory*, University of Toronto, IEEE Computer v.23 , n.5 , pp.54-64 , May 1990.

[8] Bershad B. N. , Zekauskas M. J. , SawDon W. A. , *The Midway Distributed Shared Memory System* , COMPCOM 1993.

[9] Keleher P., *The Relative Importance of Concurrent Writers and Weak Consistency Models*, in Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS-16), pp. 91-98, May 1996.