A Preliminary Speedup Comparison between TreadMarks and Nautilus DSM Systems

Mario Donato Marino, Geraldo Lino de Campos*

Computing Engineering Department- Polytechnic School of University of São Paulo

Abstract

Nautilus is a Multithreaded Distributed Shared Memory system based on scope consistency. The multithread implementation disallows the use of SIGIO signals in order to minimize the context switch of traditional processes. This paper shows the results of some benchmarks submitted to Nautilus. To have an accurate and correct evaluation of Nautilus, it is compared with the main important DSM system: TreadMarks. The benchmarks evaluated in this study are: IS (from NAS), Matmul (matrix multiplication) and SOR (from Rice University).

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. Carter[1] has classified the DSM evolution in two generations, the first one characterized by a big number of consistency messages and the sequential consistency and the second one, by a big reduction of the number of consistency messages and by the adoption of a release consistency model.

Nowadays, it is believed that it is more appropriated to use a new DSM classification. It is possible to extend the DSM evolution classification suggested by [1], by introducing a third generation, represented by a new generation of DSM systems. By adapting the definition from Carter[1], these three generations would be characterized by: 1) a big number of consistency messages by the adoption of the sequential consistency model; Ivy[6] is an example; 2) a drastic reduction of the number of consistency messages by the adoption of the release consistency model, applying techniques to reduce the false sharing; Munin[2] is an example; 3) several efficient consistency models and a minimal number of messages to maintain the consistency; TreadMarks[3], JIAJIA[4] and Nautilus[5] are examples. This paper introduces a DSM system that belongs to the third generation: Nautilus.

*{mario, geraldo}@regulus.pcs.usp.br

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

There are a few previous papers [2, 3] comparing different DSM systems; however, most of them do not evaluate DSM systems that belong to the third generation. One of the contributions of this paper is to show the speedups of two different third 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[2, 3] that are using networks with different operating systems, to equalize the comparison, these two third generation DSM systems are compared on a PC computer network with a free operating system. So, with an ordinary hardware, a free operating system, and a DSM 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.

The comparison of Nautilus speedups with TreadMarks speedups is done by applying three different benchmarks: IS (from NAS), Matmul (matrix multiplication) and SOR (from Rice University).

The environment of the comparison is a 8PC's network interconnected by a fast-ethernet shared media. The operating system used in each PC is Linux (2.x).

2 TreadMarks

2.1 TreadMarks features

TreadMarks [3] is one of the most important DSM systems. It was the first DSM to have its speedup comparable [3] to a shared memory machine. 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 consistency model used by TreadMarks is lazy release consistency[3], so the propagation of the modifications 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, making it 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 through the net; ii) multiple writer techniques of Munin [1]; iii) primitives compatible with m4; iv) AIX, Solaris, free Unix (Linux 2.x); v) UDP protocols.

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[1] all over the user's application execution. So, the size of the benchmarks used to evaluate the speedups of the DSM system can be compromised if there is not enough memory to execute the program or if the operating system does swap. If it cannot use enough size to run the benchmarks, the relation computation versus synchronization becomes unfavorable to use a DSM system.

3 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 implemented using threads; 3) CVM[8] 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 owner of the pages and invalidating pages in the acquire primitive ; ii) multiple writer techniques; iii) multithreaded DSM 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 and JIAJIA; vii) operating under Linux 2.x; viii) UDP protocols.

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. It is believed that the scope consistency model is a different implementation of the release consistency model that can produce good speedups if well applied.

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

As TreadMarks and JIAJIA do, Nautilus also worries about synchronization messages. To minimize the number of messages, the synchronization messages would carry consistency information, minimizing the emission of the last ones.

To improve the speedup of the applications submitted, Nautilus introduces two news: i) multithreaded implementation; ii) diffs of pages that were written by the owner are not created. The multithreaded implementation of Nautilus permits: i) minimization of context switch; ii) no use of SIGIO signals. Most 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, also Nautilus is worried about network protocols. So, it uses UDP protocol to minimize overheads.

Nautilus also cares about compatibility of primitives. Its primitives are simple and totally compatible with TreadMarks and JIAJIA; as a result, there is not any need of code rearranjements. One example of this compatibility is that IS and Matmul are converted from JIAJIA and SOR from TreadMarks, basically changing the name of the primitives.

4 Experimental Evaluation

4.1 Environment

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). In order to measure the speedups, 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: i) the applications were executed and the speedups measured using *Nautilus* running on up to **8 nodes; ii) bigger input sizes** than 1792 for the benchmarks Matmul and SOR, **were not possible** to be evaluated.

4.2 Evaluation programs

4.2.1 IS (from NAS)

"*The IS problem from NAS Benchmarks ranks an unsorted sequence of keys using bucket sort. It is unique in that floating point operations are not involved. The parallel version of IS divides up the keys among processors. There is a shared bucket for all processors* *and each processor has a private bucket. First, each processor counts its keys in the private array of buckets. These values in private buckets are summed up into the shared bucket in a critical section which is protected by a lock. Finally, each processor reads the sum and ranks their keys.*"[4] The parameters used in the evaluation are **NUMREPS=10**, **MAXKEY** = $\mathbf{2}^7$ and the following **N**(number of keys)**:** **N=$\mathbf{2}^{22}$**.

4.2.2 Matmul

"*Matmul is a simple matrix multiplication program with inner product algorithm. All arrays are allocated in shared memory. To achieve a good data locality, the multiplier is transposed before multiplication. This program requires no synchronization in the multiplication process, so only three barriers are used at the beginning and the end of the program.*"[4] The matrix **size** used in this experiment is **1408x1408**.

4.2.3 SOR (from Rice University)

"*SOR from Rice University is 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 boundary rows on a barrier*".[4] The size of red and black matrix used is **1408x1408**. The number of iterations is **10** .

5 Result Analysis

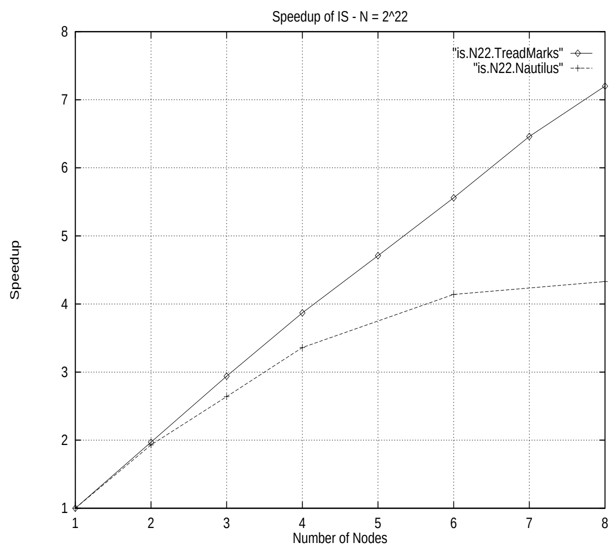

figure 1: speedups of IS - N=$\mathbf{2}^{22}$

5.1 IS

The computation to communication ratio of IS increases linearly with the problem size. "*In IS, most time-consuming computation is for each processor to count its local part of keys. Summing the counting results up in the critical section constitutes the communication work*"[4].

The figure 1 shows that both DSMs speedups scale with the number of nodes and TreadMarks outperforms Nautilus until 66.28%, specially for large number of nodes and for small N. With the increasement of N, this speedup difference between both DSMs decreases. With the increasement of N, the relation communication/computation decreases, so the speedups of both DSMs become similar. In Nautilus, diffs have to be sent to its home nodes before the release message is sent to the lock, while in TreadMarks, diffs are kept locally. As a result, when summing up values, Nautilus takes mor time for each processor to enter and leave the critical section. Also, the lock/unlock implementation of Nautilus is under development, does not presenting good performance nowadays.

5.2 Matmul

Looking at the figure 2, the speedups of both DSMs can be seen under the input size N=1408.

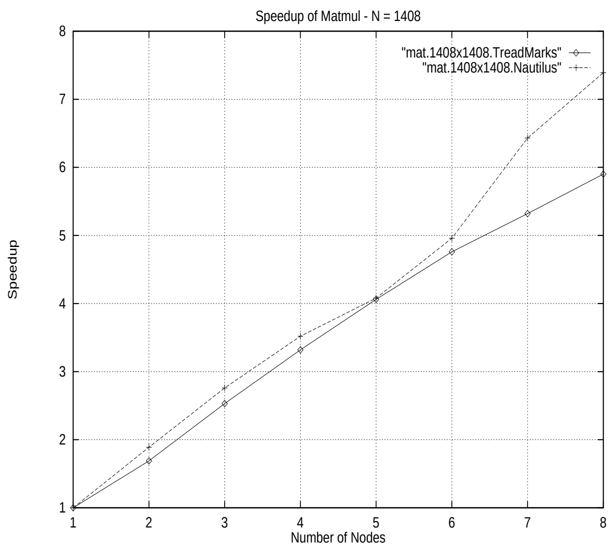

figure 2: speedups of Matmul - 1408x1408

The first observation is that the speedups of both DSMs increase with the number of nodes available for N = 1408. Still looking at figure 2, Nautilus is about 10% to 20.16% faster than TreadMarks, the best reference of DSM area. The best speedup of Nautilus happens due to the good data distribution (improving the data locality: multiplicand and result matrices are local giving a lower number of page faults and a lower cold start up time to distribute shared data) for Matmul benchmark and the lower overhead of the scope consistency model. As there is no need to allocate twins and to create diffs when a page is written in its owner node, the avoidance of SIGIO signals and the lower overhead of Nautilus multithreading helps to improve the overall perfomance.

5.3 SOR

The SOR from Rice University solves Laplace partial equations. For a number of iterations it has two barriers 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 as the size of problem increases.

Looking at the figure 3, the speedups of both DSMs can be seen under the iput size N=1408 and 10 iterations.

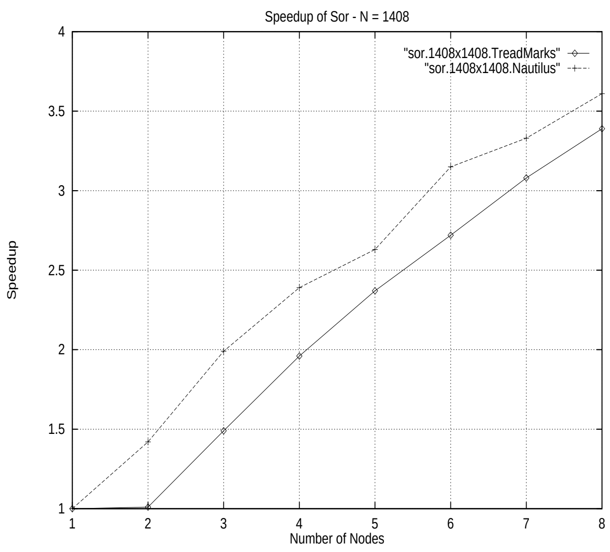

figure 3: speedups of SOR - 1408x1408

For 2 to 8 nodes Nautilus outperforms TreadMarks. Looking at the figure 3, the speedup difference reaches up to 33.56%. The excellent speedup of Nautilus happens because of the better data distribution adopted by itself improving the matrix data locality (minimizing the number of messages through the net) and giving a lower cold start up time to distribute shared data. The lazy release protocol of TreadMarks has additional overhead of looking up and maintaining directory on a page fault.

Justifying the best speedups of Nautilus: 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. As it was said in Matmul evaluation, 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

In this paper a new DSM systems classification was proposed based on their evolution, extending the classification initially proposed by [1].

This paper confronts the speedups of TreadMarks and Nautilus, used at present and compared on the same environment, with the same computers, network and operating system. For the IS applicative, Nautilus has worse speedup than TreadMarks because of the lock/unlock implementation and diffs sending (to the owner nodes) of Nautilus. For the applicative Matmul, Nautilus is from 10% to 20% faster than TreadMarks (N=1408). Finally, for the applicative SOR, Nautilus is until 33.56% faster than TreadMarks (10 iterations and N=1408).

As shown, Nautilus has good speedups, comparable to other more well-known DSM, TreadMarks, surpassing its speedup depending on the application. The use of multithreading and the avoidance of SIGIO signals also help to improve Nautilus speedup. Other data distribution and other lock/unlock implementation are undergoing study, in order to further improve Nautilus speedup. The lazy release consistency model (TreadMarks) and the scope consistency model (Nautilus) presented good speedups for the benchmarks evaluated in this paper. It is intended to evaluate a TreadMarks version which does not have limitations as the shared memory size and limited number of nodes.

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 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] 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.