The Speedup of Nautilus, a new DSM System, Compared to TreadMarks

Mario Donato Marino, Geraldo Lino de Campos*
Computing Engineering Department- Polytechnic School of University of 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, which permits to minimize the context switch of traditional processes, thus improving the speedups. 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), LU (from SPLASH-II), Matmul (matrix multiplication) and SOR (from Rice University).

1 Introduction

In the last years the research on *Distributed Shared Memory* (DSM) [8] area has greatly 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 Carter[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 and the adoption of the sequential consistency model; one can exemplify this with the Ivy [6];
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; the examples are Munin [2] and Quarks[7];
3. several efficient consistency models and a minimal number of messages to maintain the consistency; the examples are TreadMarks[3], JIAJIA[4] and Nautilus[5].

This paper introduces a DSM system that belongs to the third generation: Nautilus.

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 systems; once they are evaluated under the same conditions, the results of this comparison would tend towards accurate comparisons.

There are a few previous papers [3, 7] 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

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

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[3, 7] 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, 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 a homogeneous and fair comparison.

The comparison of Nautilus speedups with TreadMarks speedups is done by applying four different benchmarks: IS (from NAS), LU (from SPLASH-II), Matmul (matrix multiplication) and SOR (from Rice University). Also, different input parameters size are evaluated for each benchmark.

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 speedup comparable [3] to a shared memory machine for the SOR benchmark (from Rice University): basically comparing an ATM network of DEC machines with a SGI shared memory machine.

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:

- lazy release consistency and its variations [3], minimizing the number of consistency messages through the net;
- multiple writer techniques of Munin [1];
- primitives compatible with m4;
- IBM SP2, Sun Sparc, PCs;
- AIX, Solaris, free Unix (Linux 2.x);
- UDP protocols to minimize network protocols overhead;
- first DSM to have a speedup compatible to a shared memory machine[3].

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[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.

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

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:

- 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 ;
- multiple writer techniques;
- multithreaded DSM: threads to minimize the switch context;
- no use of SIGIO signals(which notice the arrival of a network message);
- minimization of diffs creation;
- primitives compatible with TreadMarks, Quarks and JIAJIA;
- network of PCs;
- operating under Linux 2.x;
- UDP protocols.

The following considerations are valid to Nautilus:

1. it uses the scope consistency model, which is simple and efficient[4];
2. if the scope consistency model will be interpreted as a release consistency model, which it is believed until to be efficient if some techniques will be used[7];
3. the scope consistency model minimizes the memory use to maintain the consistency, instead of the lazy release consistency model 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. So, it is possible to view Nautilus as a multi-home and multithreaded DSM system based on release consistency.

According to the scope consistency model, in Nautilus the owner nodes of the pages do not need to send the diffs to other nodes. Thus, in this model, diffs of pages written by the owner are not created, what is believed to be more efficient than the lazy diff creation of TreadMarks.

To improve the speedup of the applications submitted, Nautilus uses:

- multithreaded implementation;
- diffs of pages that were written by the owner are not created.

The multithreaded implementation of Nautilus permits:

- minimization of context switch;
- no use of SIGIO signals;

Most DSM systems created as of now are 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. A thread is maintained sleeping in a blocked reading of a socket, while it waits for the arrival of the message. At its arrival, the thread wakes up, read it and wakes up a decoder thread in order to decode the header of the message. This decoder thread, which can be one or more threads, does the respective action contained in the messages which already have arrived, following the rpc model. Avoiding the use of SIGIO signal and the handler system call minimizes the overheads of the system. Nautilus is the first third generation DSM system which has eliminated completely the use of SIGIO signal in its implementation.

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.

Nautilus also cares about compatibility of primitives. Its primitives are simple and totally compatible with TreadMarks, JIAJIA and Quarks; 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.

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

4 Experimental Evaluation

4.1 Environment

The evaluation programs of this study are executed on top of *Nautilus* on the following network of PCs:

- nodes: K6 - 233 MHz (AMD), 64 MB of memory and 2 GB IDE disk;
- 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:

- the applications were executed and the speedups measured using *Nautilus* running on up to **8 nodes;**
- **bigger input sizes** for the benchmarks Matmul and SOR, **were not possible** to be evaluated.

It is possible with JIAJIA and Nautilus to choose different data distribution, and if the data access pattern was known, it is possible to optimize the speedup. Thus, in order to have **a fair and homogeneous comparison**, in this paper the data distribution used by all DSMs is the default provided by their primitives.

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**=$\mathbf{2}^{21}$ and **N**=$\mathbf{2}^{22}$.

4.2.2 LU (blocked LU: kernel from SPLASH II)

"*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 Ns used in the evaluation are from **N** = **1024**, **N**=**1408** and **N** = **1792**, with step of 128.

4.2.3 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 **sizes** used in this experiment are **1024x1024**, **1408x1408** and **1792x1792**.

4.2.4 SOR (from Rice University)

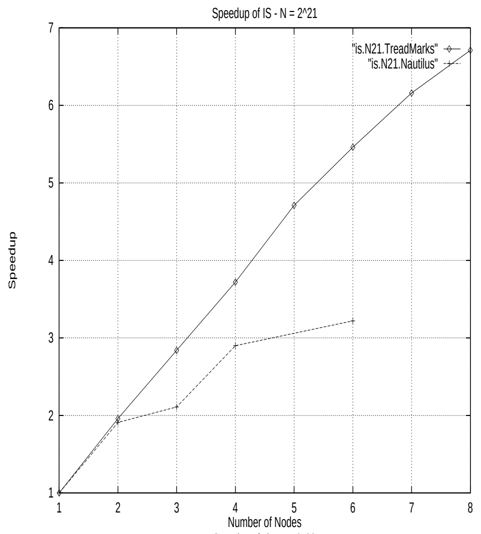
"*SOR from Rice University is solves partial differential equations (Laplace equations) with a Over-Relaxation method. There are two arrays, black and red array allocated in shared memory. Each element from red array is computed as an aritmethic mean from black array and each element from black array 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 are **1024x1024**, **1408x1408** and **1792x1792**. The number of iterations is **10** .

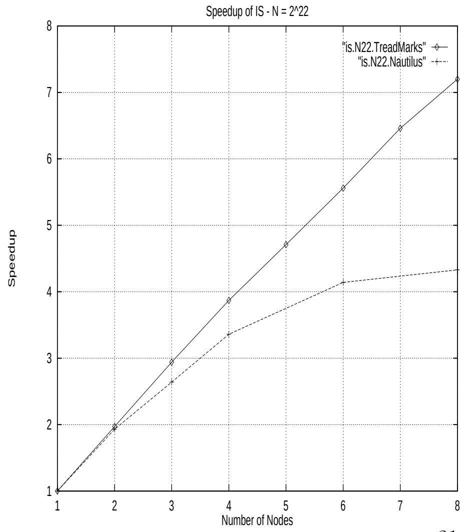
5 Result Analysis

5.1 IS

The figures 1a and 1b show that both DSMs speedups scale with the problem size. This is because the computation to communication ratio of IS increases proportionally 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].

Looking at the figures 1a and 1b, TreadMarks outperforms Nautilus until 70%, specially for large number of nodes and for small N. With the increasement of N, this speedup difference between both DSMs decreases. By increasing 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 actually.





figures 1a, 1b: speedups of IS with $\mathbf{N=2^{21}}$ and $\mathbf{N=2^{22}}$respectively

5.2 LU

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].

Looking at the figures 2a, 2b and 2c, the speedups of the three DSMs can be seen under different input sizes: N=1024, N=1408 and N=1792.

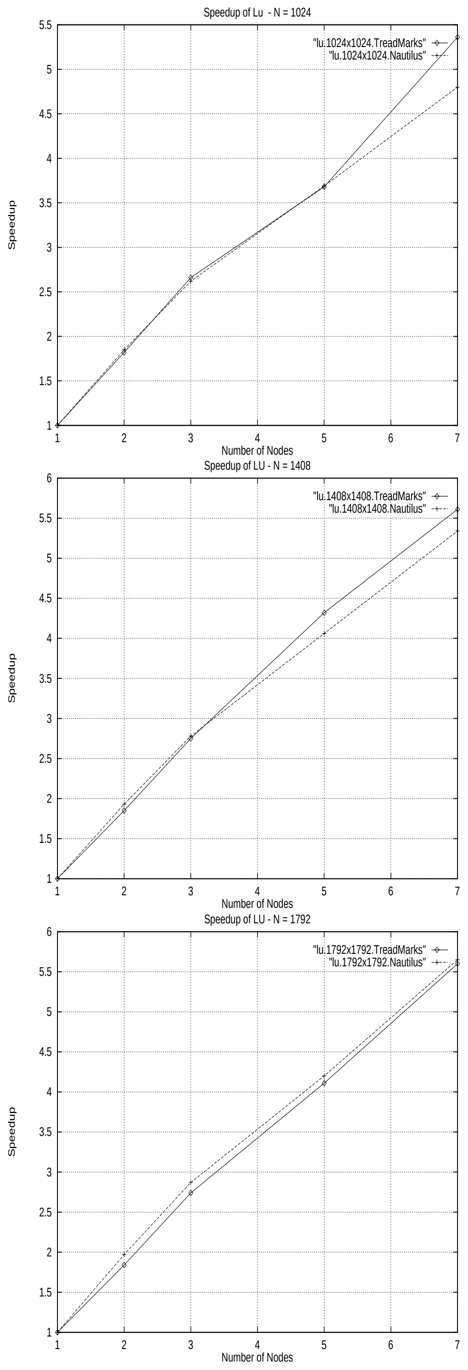
It is possible to notice from the figures 2a, 2b and 2c, the speedups of both DSMs increase with the number of the nodes N. Also the speedups increase with the increasement of N, except with the TreadMarks speedup from N=1408 to N=1792, as it will be explained bellow.

With a number of nodes less than 3 nodes, for N=1024 and N=1408, the speedups of both DSM systems are very similar. For more than 5 nodes, taking advantage of its data distribution and minimizing the diffs to be sent by using the lazy release consistency, for N=1024 and for N=1408, TreadMarks is faster than Nautilus. But, for N=1792, Nautilus is faster than TreadMarks. For N=1792, the best speedups of Nautilus occur because with this N value, due to lazy release consistency model adopted by TreadMarks, too much diffs are stored, which is a feature of lazy release consistency model, causing the swapping of the operating system.

In terms of percentage, for N=1024 and for 7 nodes, TreadMarks outperforms Nautilus about 11.67%. For N=1408, TreadMarks is up to 6.04%. For N=1792, Nautilus outperforms TreadMarks about 3% .

Concluding, as it was said, for N=1024 and N=1408, its data distribution and the lazy diffing (minimizing the transmission of diffs) used by TreadMarks, minimizes the number of messages through the net and gives it better speedups than Nautilus. For N=1792, due to the diffs stored by TreadMarks, which cause the swapping of the operating system, Nautilus has the best speedup, up to 3% faster than TreadMarks. Due to the diffs stored, as it was said cause the swapping of the operating system, also cause the speedup decreasement with the increasement of N (from N=1408 to N=1792).

So, the two release consistency models, lazy release consistency model (adopted by TreadMarks) and the scope consistency model (adopted by Nautilus) have a good and similar behavior, providing good speedups.



figures 2a, 2b and 2c: speedups of LU: **N=1024, N=1408 and N=1792**

5.3 Matmul

Looking at the figures 3a¸3b and 3c, the speedups of both DSMs can be seen under three different input size: N=1024, N=1408 and N=1792.

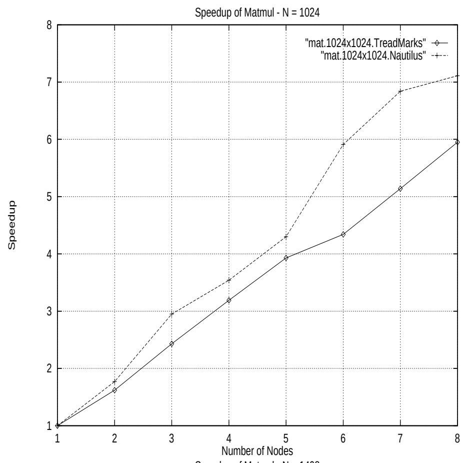
The first observation is that the speedups of both DSMs increase with the number of nodes available for a fixed N and, also the speedups increase according to the increasement of N.

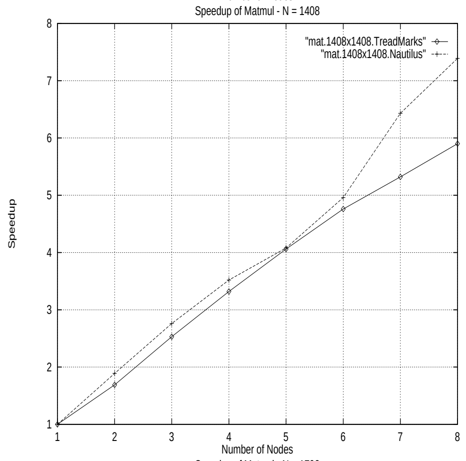
Still looking at figures 3a, 3b and 3c, for N=1024, Nautilus is between 10% to 26.56% faster than TreadMarks, the best reference of DSM area. For N=1408, Nautilus outperforms TreadMarks about 20.16% . And for N=1792, Nautilus is about 5.70 to 15.20% faster than TreadMarks except with 5 nodes, when TreadMarks outperforms Nautilus about 5.22%.

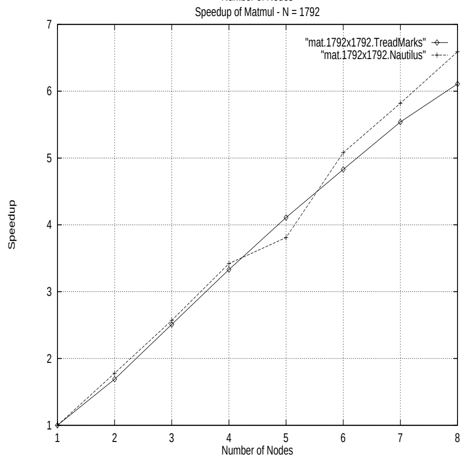
The best speedup of Nautilus happens due to its data distribution (improving the data locality: factors and result matrices are local giving a lower number of page faults and also a lower cold start up time to distribute shared data) and the lower overhead of the scope consistency model. As there is 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 the threads used by Nautilus help improving the overall perfomance.

5.4 SOR

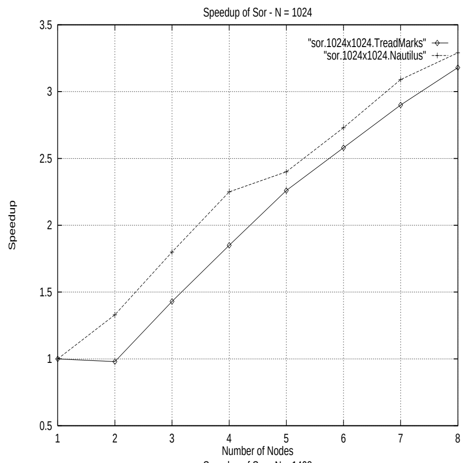
The SOR from Rice University solves Laplace partial equations. For a specyfied number of iterations it has two barriers each iteration and communication occurs across boundary rows on a barrier. The communication does not increase with the number of processors and the relation communication/computation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

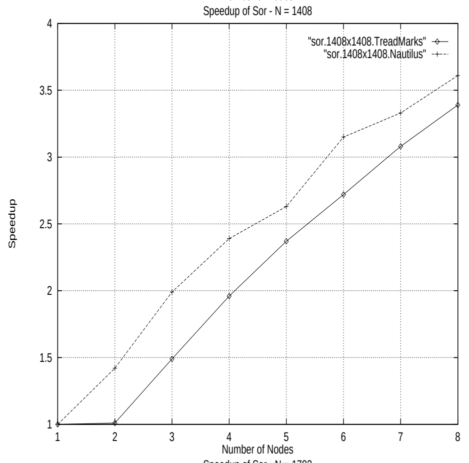


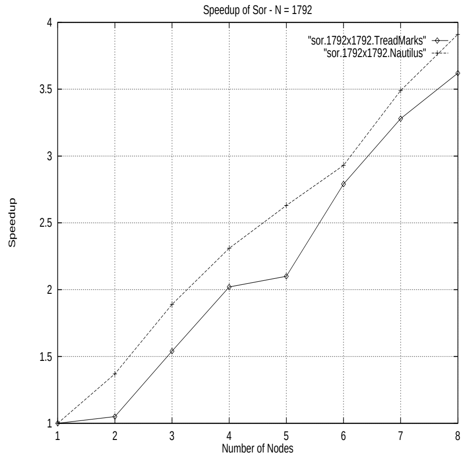




figures 3a, 3b and 3c: speedups of Matmul with sizes **1024x1024**, **1408x1408** and **1792x1792**







figures 4a, 4b and 4c: speedups of SOR with sizes **1024x1024**, **1408x1408** and **1792x1792**

Looking at the figures 4a, 4b and 4c, the speedups of both DSMs can be seen under three different input sizes: N=1024, N=1408 and N=1792.

For both N=1024, N=1408 and N=1792, and for 2 to 8 nodes Nautilus outperforms TreadMarks. For N=1024 and N=1408 the speedup difference reaches up to 26.31%. For N=1408, the speedup difference is up to 28.87%. And, for N=1792, the speedup difference reaches up to 23.36%. 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. The avoidance of SIGIO signals and the multithreading of Nautilus help improving the SOR speedup.

Increasing N from 1024 to 1408 and from 1408 to 1792, the locality of the SOR improves, also improving the speedup.

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, improving the Nautilus speedups.

6 Conclusion

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

This paper introduces Nautilus, a new DSM system, based on scope consistency, which brings the ideas of multithreading and eliminating the SIGIO signals. Nautilus is the first third generation DSM system which has eliminated completely the use of SIGIO signal in its implementation. Also, 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 speedups than TreadMarks because of the lock/unlock implementation and diffs sending (to the owner nodes) of Nautilus. For the applicative LU, in terms of percentage, for N=1024 , TreadMarks outperforms Nautilus about 11.67%, and for N=1408, TreadMarks outperforms Nautilus up to 6.04%, but for N=1792, Nautilus outperforms TreadMarks about 3%. For the applicative Matmul, Nautilus is between 10% to 20% faster than TreadMarks for N=1024 and until 15.20% faster for N=1792. Finally, for the applicative SOR, Nautilus is until 20% faster than TreadMarks for N=1024, 28.87% for N=1408 and until 23.36% for N=1792.

As it was 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.

Evaluate other problem size: one of the problems of this study is the Demo version of TreadMarks that was evaluated. This version, that has the same efficiency of a normal version of the software, has two problem limitations: the size of the shared memory and a limited number of nodes. A full version of the software to evaluate it better is being acquired.

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.; A Preliminary Speedup Comparison between TreadMarks and Nautilus DSM Systems, to be published in ERSADS99, Madeira Island, April, 1999.

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

[7] Swanson M., Stoller L., Carter J., *Making Distributed Shared Memory Simple, Yet Efficient*, Computer Systems Laboratory, University of Utah, technical report , 1998.

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

[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.