A Preliminary Speedup Comparison Between Two Scope Consistency DSM Systems: JIAJIA and Nautilus

Mario Donato Marino, Geraldo Lino de Campos*

Computer 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 speedups of some benchmarks submitted to Nautilus. To have an accurate and correct evaluation of Nautilus, it is compared with other scope consistency DSM system: JIAJIA. The benchmarks evaluated in this study are: IS (from NAS), LU (kernel 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 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.

The first generation can be exemplified with Ivy[6]. The second generation can be exemplified with Munin[2][7], TreadMarks[3] and JIAJIA[4]. This paper introduces a DSM system that belongs to the second generation: Nautilus[5].

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 other well known DSM system, 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 comparisons.

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

There are a few previous papers [3] [4] [9] comparing different DSM systems; however, most of then do not evaluate DSM systems that belong to the second generation. One of the contributions of this paper is to show the speedups of two different scope 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][9] that are using networks with different operating systems, to equalize the comparison, these two 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 a homogeneous and fair comparison.

For a meaningful evaluation of the Nautilus DSM system, one of the most important DSM systems, that have been used by several research groups in the scientific community was chosen to be compared against the Nautilus: JIAJIA. Previous and recent papers[4] only compare JIAJIA with CVM[10] (lazy release consistency) and TreadMarks with Munin[3], thus, as it was said, one contribution of this paper is to compare the speedups of two different scope consistency DSM systems.

The comparison of Nautilus speedups with JIAJIA speedups is done by applying four different benchmarks: IS (from NAS), LU (kernel from SPLASH II), Matmul (matrix multiplication) and SOR (from Rice University). Also, in order to evaluate the locality of the benchmarks, two different input parameters sizes 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 JIAJIA features

JIAJIA[4] is an important DSM system that uses scope consistency, which 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 the scope consistency model, diffs[1] are transmitted in each critical section to maintain the consistency. So, JIAJIA uses the scope consistency memory model, only sending consistency messages to the owner of the pages and invalidating pages 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, free Unix (Linux 2.x); vii) UDP protocols, minimizing network protocols overhead.

The main objective of JIAJIA[4] is to be as simple as possible 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 appropriated, allowing better speedups.

3 Nautilus features

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[10] 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: 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.

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

It is believed that the scope consistency model is a different implementation of 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.

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

- 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;

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.

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, both minimize 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.

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 rearrangements. One example of this compatibility is that IS, LU 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.

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 then.* "[4]

The **N**s used in the evaluation are **N**=**1024 and N**=**1792**.

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 two barriers are used at the beginning and the end of the program.*"[4]

The matrix **sizes** used in this experiment are **1024x1024** and **1792x1792**.

4.2.4 SOR (from Rice University)

"*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 boundary rows on a barrier*".[4]

The size of red and black matrix used are **1024x1024 and 1792x1792**. The number of iterations is **10** .

5 Result Analysis

Before presenting the results and their analysis, it is necessary to emphasize that the execution time (speedup) for number of nodes = 1 in all evaluated benchmarks is obtained from the sequential version of the benchmarks without any DSM primitive.

Only a **few results** were obtained with the current version of **JIAJIA,** because several executions have terminated abnormally (possibly an implementation problem).

5.1 IS

The figures 1 and 2 shows the both DSMs speedups for the IS benchmark. By observing the figure 1, Nautilus does not presented results for 7 and 8 nodes, because these executions finished abnormally.

The behavior of the curves showed in figures 1 and 2 is assimptotical and occurs because 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].

Looking at figure 1, for **N**=$\mathbf{2}^{21}$, JIAJIA outperforms Nautilus up to 46.27%. Looking at the figure 2, for **N**=$\mathbf{2}^{22}$, JIAJIA outperforms Nautilus about 37.8%. Although the two DSM systems use the same consistency models, there is a meaningful difference between the speedups of both DSMs. This difference occurs because, as Nautilus is a DSM which is in development, its lock/unlock implementation does not present good performance actually.

Changing N from **N**=$\mathbf{2}^{21}$to **N**=$\mathbf{2}^{22}$, this speedup difference between both DSMs decreases from 46.27% to 37.8%, because by increasing of N, the relation communication/computation decreases, so the speedups of both DSMs become more similar.

A final observation is that increasing N, the IS locality increases, as it is possible to see by the increasing of the speedups showed in the figures 1 and 2.

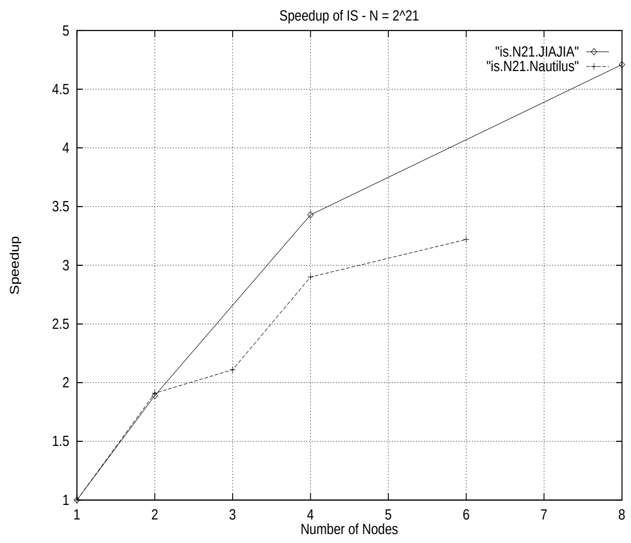

figure 1: speedups of IS with $N=2^{21}$

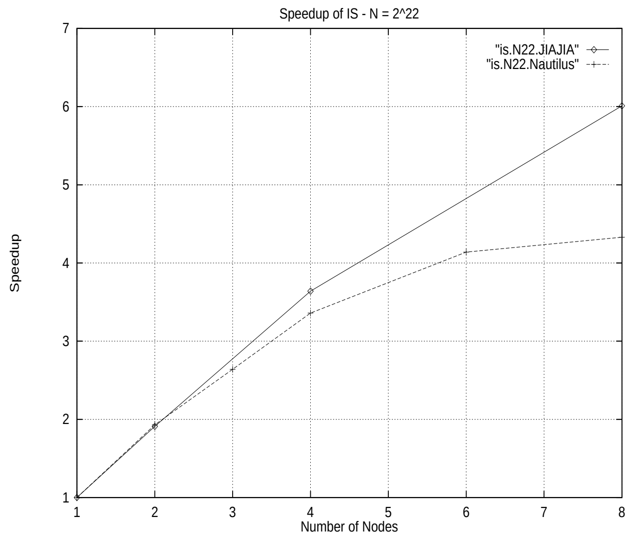

figure 2: speedups of IS with $N=2^{22}$

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 3 and 4, the speedups of both DSMs for LU benchmark can be seen for N=1024 and N=1792. The first observation from these figures is that the increasement of N improves the locality of LU, when it is submitted to both DSMs, thus improving its speedups.

With a number of nodes less than 3 nodes, the speedups of both DSM systems are good and very similar. With four nodes, JIAJIA is faster than Nautilus up to 7.10%, for N=1024, and up to 6.50%, for N=1792. For more than 4 nodes, Nautilus is faster than JIAJIA up to 2.10%, for N=1024, and up to 6.60%, for N=1792. It can be also observed that the speedup difference between the DSMs increases with the increasement of N. This occurs because, 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.

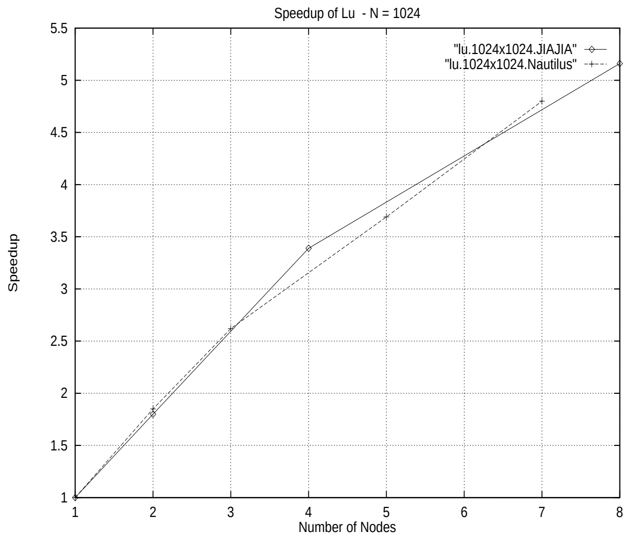

figure 3 : speedups of LU for N=1024

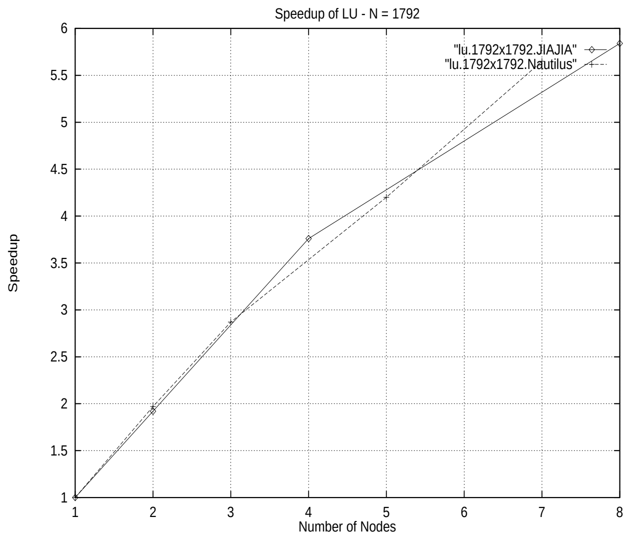

figure 4 : speedups of LU for N=1792

5.3 Matmul

Observing the figures 5 and 6, the speedups of both DSMs for Matmul benchmark can be seen for N=1024 and N=1792. By analyzing these figures, Nautilus is faster up to 32.30%, for N=1024, and up to 25.05%, for N=1792, than JIAJIA. Both DSMs addopt the same consistency memory model, but the best speedup of Nautilus happens due to the good data distribution (improving the data locality: the matrix factors and the matrix result are local, i.e., the pages are modified by their owners, 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.

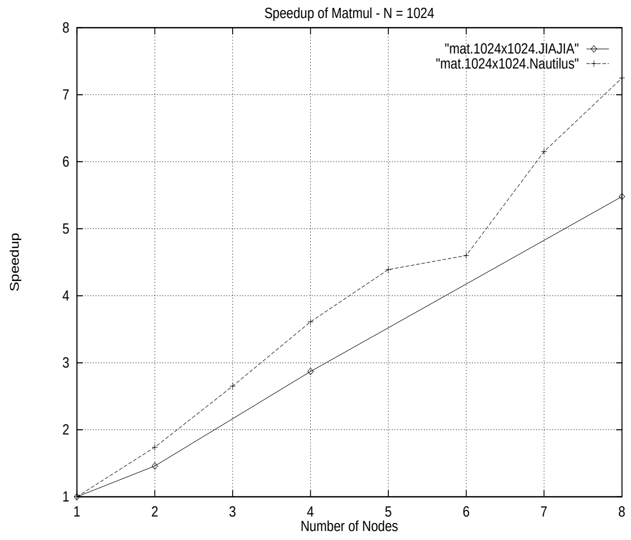

figure 5: speedups of Matmul N=1024

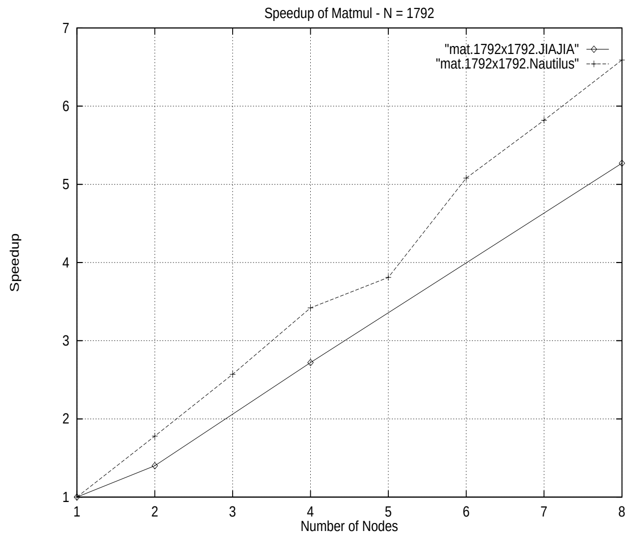

figure 6: speedups of Matmul N=1792

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 to improve the overall performance.

By increasing N from 1024 to 1792, it is believed that the speedup difference decreases from 32.30% to 25.05% because of the increasement of the Matmul locality for JIAJIA is bigger than its increasement under Nautilus.

5.4 SOR

The SOR benchmark 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.

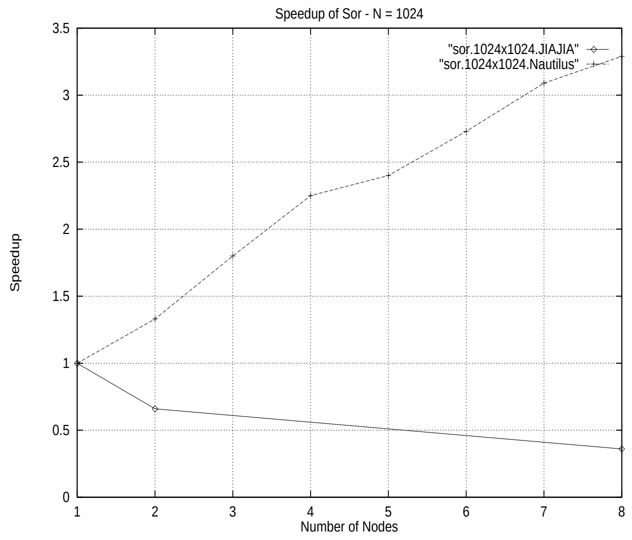

figure 7: speedups of SOR for N=1024

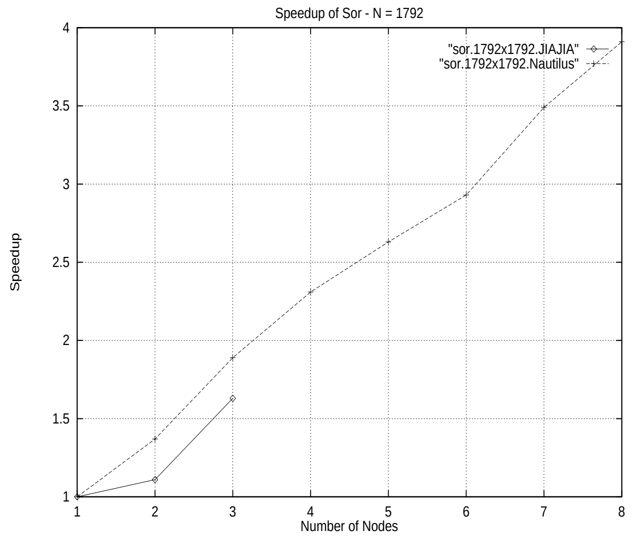

figure 8: speedups of SOR for N=1792

The speedups of JIAJIA are very unusual and only too few to conclude about them. Therefore, any related speedups for JIAJIA are not considered for SOR analysis. The figures 7 and 8 shows the speedups of Nautilus with N=1024 and N=1792. The first observation of these figures is that by changing N from 1024 to 1792, the speedup of SOR under Nautilus improves, due to the increasing of the locality of this benchmark.

The excellent speedup of Nautilus happens because of the better data distribution (good choice of the page owners) addopted 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. Also the avoidance of SIGIO signals and the multithreading help to improve the SOR speedup.

Concluding, 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 (good choice of the page owners), multithreading and the avoidance of SIGIO signals, which improve the Nautilus speedup.

6 Conclusion

This paper confronts the speedups of two scope consistency DSM systems 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 several different benchmarks which have several different features under two DSMs, with their own aspects and implementations.

The scope consistency model addopted by JIAJIA and Nautilus presented good speedups for all benchmarks evaluated in this paper.

Summarizing the results for this experiment, for the IS benchmark, JIAJIA is faster than Nautilus. For the LU, Matmul and SOR Nautilus is faster than JIAJIA.

For the IS applicative, JIAJIA is up to 46.27% faster than Nautilus, for N=1024, and up to 37.80%, for N=1792. For the LU applicative, Nautilus has the best speedup, until 6.60% faster than JIAJIA. For the Matmul applicative, Nautilus has the best speedup, reaching up to 32.30% more speedup than JIAJIA, for N=1024, and up to 25.05%, for N=1792. For the SOR applicative, Nautilus also has a good speedup, since JIAJIA has very unusual speedups.

As it was shown, Nautilus has good speedups, comparable to other well-known DSM, JIAJIA, surpassing it depending on the application. The use of multithreading, the avoidance of SIGIO signals and a good data distribution (good choice of the page owners) also help to improve Nautilus speedup. The lock/unlock implementation of Nautilus is under development in order to improve Nautilus speedup for, for example, IS benchmark. Also, other data distribution are undergoing study, in order to further improvement of Nautilus speedup.

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 other DSMs, for example, TreadMarks and Nautilus.

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

[10] 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.