1 A PRELIMINARY DSM SPEEDUP COMPARISON: JIAJIA X NAUTILUS

Mario Donato Marino
and Geraldo Lino de Campos

Computing Engineering Department - University of Sao Paulo
[mario, geraldo]@regulus.pcs.usp.br

Abstract: Nautilus is a Multithreaded Distributed Shared Memory system based on scope consistency. The multithread implementation permits no use of SIGIO signals and to minimize the context switch of traditional processes. We show and compare the speedup of a new DSM system, Nautilus, with an important DSM used by the academic community: JIAJIA. This comparison is done submitting four different benchmarks to both DSM systems. The benchmarks evaluated in this study are: EP (from NAS), LU (kernel from SPLASH II), Matmul (matrix multiplication) and SOR (from Rice University).

keywords: distributed shared memory; DSM; distributed computing.

1.1 INTRODUCTION

In the last 6 years the *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.

After some time, nowadays 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, techniques applied 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 another DSM system that belongs to the third generation: Nautilus.

DSM comparisons are commonly based 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 DSM system, 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; 10) comparing different DSM systems; however, they 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. Taking into account these last considerations, it's intended to

compare them on the same network of PCs interconnected by a common media. In addition, as there are papers (3; 10) that are using networks with different operating systems to equalize these comparison, these 2 DSM systems are compared on a PC network with a free operating system. Therefore, with an ordinary hardware, operating system and a DSM (JIAJIA) 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 JIAJIA speedups is done by applying four different benchmarks: EP (from NAS), LU (kernel from SPLASH II, Matmul (matrix multiplication) and SOR (from Rice University).

As it was said before, the environment of the comparison is a 8 PC's network interconnected by a fast-ethernet shared media and the operating system used by the PCs is Linux (free Unix).

1.2 JIAJIA

1.2.1 JIAJIA features

JIAJIA (4) is an important DSM system that 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. The main aim of JIAJIA(4) is to be as simple as possible in order to minimize overheads of diff[1] creation and diff storage and also to minimize the number of consistency messages through the net.

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 most interesting feature of JIAJIA is its simple ideas trying to make it a fast DSM system.

The consistency model used by JIAJIA is the scope consistency model, only sending consistency messages to the owner of the pages and invalidating pages in the acquire primitive, minimizing the number of the messages through the net.

1.3 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 both process and multithreaded versions of TreadMarks that can be executed on top of free Unix, but it does not use scope consistency;

2. JIAJIA is a DSM system based on scope consistency, but it is not implemented using threads.

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 in order 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) Linux 2.x; ix)UDP protocols.

Nautilus is the first multithreaded DSM system implemented on top of a free unix platform that uses the scope consistency model.

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.

As JIAJIA, Nautilus is a multi-home DSM system, which means that the owners of the pages are fixed, so the diffs do not need to be sent to different nodes to maintain the consistency.

To improve the speedup of the applications submitted, Nautilus introduces two news:

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

The multithreaded implementation of Nautilus permits:

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

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

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 will minimize the overheads of the system.

On the same way that TreadMarks and JIAJIA do, Nautilus also is worried about network protocols. So, it also uses UDP protocol to minimize overheads.

Nautilus cares about compatibility of primitives. Its primitives are simple and totally compatible with TreadMarks, JIAJIA and Quarks; as a result, there isn't any need of code rearranjements. One example of this compatibility is that EP, 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.

1.4 EXPERIMENTAL EVALUATION

1.4.1 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 network.

The operating system used was Linux Red Hat 5.0.

The applications were executed and the speedups measured using *Nautilus* running on up to **8 nodes .**

1.4.2 Evaluation programs

1.4.2.1 EP (from NAS). "*The Embarrassingly Parallel (EP) program from the NAS benchmark suite generates pairs of Gaussian random deviates with a scheme that is well suited for parallel computation and tabulates the number of pairs successively. The only communication and synchronization in this program is summing up a ten-integer list in a critical section at the end of program*" (4).

The parameter used in EP program is $\mathbf{M{=}2^{24}}$.

1.4.2.2 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 used is the evaluation is $\mathbf{N = 1024}$.

1.4.2.3 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.*"(8)

The matrix **size** of the matrix used in this experiment is **1024x1024**.

1.4.2.4 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**. The number of iterations is **10** .

1.5 ANALYSIS OF THE RESULTS

Before presenting the results and their analysis, it's necessary to emphasize some considerations:

1. So, whenever a - symbol is found on the table, the execution DSM being evaluated terminated abnormally, possibly due to implementation or specific benchmark problems.
2. the execution time for number of nodes = 1 in all evaluated benchmarks is obtained from the sequential version of the benchmarks without any DSM primitive. So, the primitive used to allocate memory is **malloc()** default primitive of C programming;
3. the times obtained are measured in seconds, using unix primitives to measure the elapsed time (**gettimeofday()**);

The metrics used to obtain the speedups of the DSMs evaluated in this study are:
$DSM.s(i) = \frac{t(1)}{DSM.t(i)}$, where

- (i) means the number of nodes *where* the DSM is being executed;
- t(1) = execution time in seconds of the sequential program (see the consideration 3 above);
- DSM.t(i)are the times of the DSM with i nodes ;
- DSM.s(i) is the speedup of the DSM with i nodes ;

1.5.1 EP

EP is a benchmark that has a small amount of communication and a small number of messages transmitted through the net. As we have said, it's a class A problem in the NAS benchmarks.

By looking at table1.1, a time reduction with the increase of the number of nodes can be noticed for both DSM systems.

Table 1.1 execution time (in seconds) and speedups for the EP benchmark (from NAS)

(i)	*JIAJIA.t(i)*	*JIAJIA.s(i)*	*Nautilus.t(i)*	*Nautilus.s(i)*
1	121.80	1.00	121.80	1.00
2	63.70	1.91	61.80	1.97
3	-	-	41.82	2.82
4	30.78	3.96	31.51	3.87
5	-	-	24.79	4.91
6	-	-	20.76	5.87
7	-	-	18.00	6.77
8	15.62	7.80	15.89	7.67

Another observation that can be taken from table1.1 is the similarity of the execution times for both DSM systems, differing by 2-3%. JIAJIA is faster than Nautilus, and exactly as the times are very similar, so are the speedups. As a consequence, the speedups of both DSM systems are good and linear for this problem size and this number of processors: the speedups increase according to the increasement of the number of nodes.

By plotting table 1.1, figure1.1 is obtained.

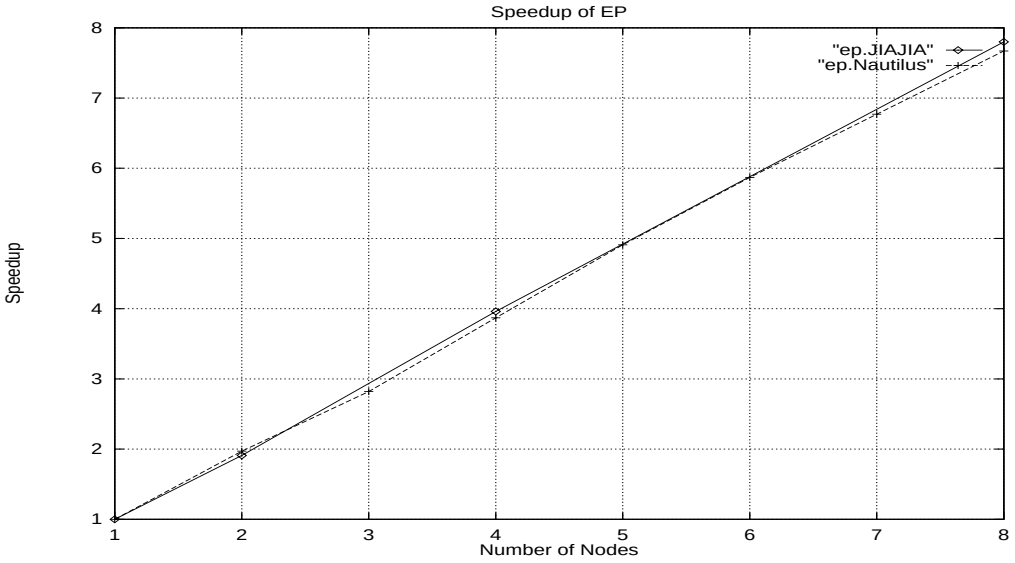
It is believed that JIAJIA has better speedups because of the semaphore implementation JIAJIA against the implementation of Nautilus semaphores.

1.5.2 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).

By looking at table 1.2, a time reduction with the increasement of the number of nodes for all three DSM systems.

Figure 1.1 speedups of EP



Taking advantage of the its data distribution and its data locality, JIAJIA is faster than Nautilus, having better execution times and, as a consequence, better speedups (table 1.2).

By plotting table 1.2, figure 1.2 is obtained with the speedups of both DSMs.

Table 1.2 execution time (in seconds) and speedups for the LU benchmark (from SPLASH II)

(i)	*JIAJIA.t(i)*	*JIAJIA.s(i)*	*Nautilus.t(i)*	*Nautilus.s(i)*
1	72.50	1.00	72.50	1.00
2	35.9	2.02	39.09	1.85
3	-	-	27.72	2.62
4	19.23	3.77	-	-
5	-	-	20.2	3.59
6	-	-	-	-
7	-	-	15.53	4.67
8	6.48	6.48	-	-

Although there have been some problems to obtain results for JIAJIA (a few results), it is supposed (the speedup curve obtained is approximated) to be about 8.4%(with 2 nodes) to 19.5% (with 7 nodes). Generally, the scope consistency model (adopted by JIAJIA and Nautilus) gives good speedups.

Although Nautilus uses the same memory consistency model of JIAJIA (scope consistency), the worst speedup obtained by Nautilus for this benchmark is due to its data distribution and, as a consequence, several page faults occur, slowing down the speedup.

1.5.3 Matmul

By looking at table 1.3 for both DSM systems it can be seen the times reduction with the increasement of the number of nodes.

As we can see in table 1.3, Nautilus is about 25% to 45% faster than JIAJIA, that is a good reference of the DSM area.

By plotting the speedups showed on table 1.3, we obtain figure 1.3. Still looking at table 1.3 or figure 1.3, the speedups of the DSMs increase with the number of nodes available.

From the few points obtained for JIAJIA, it is noticed that it does not scale well and has a poor speedup (observations 1 and 2 item Analysis). Nautilus has an excellent speedup. This happens because of the its data distribution (improving the data locality: multiplicand and result matrices are local giving a lower number of page

Figure 1.2 speedups of LU

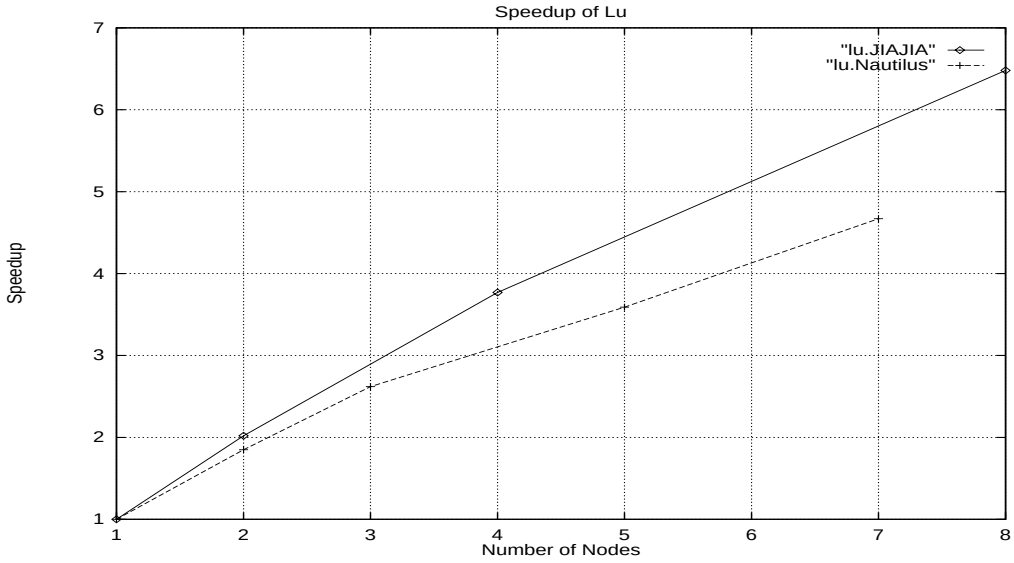
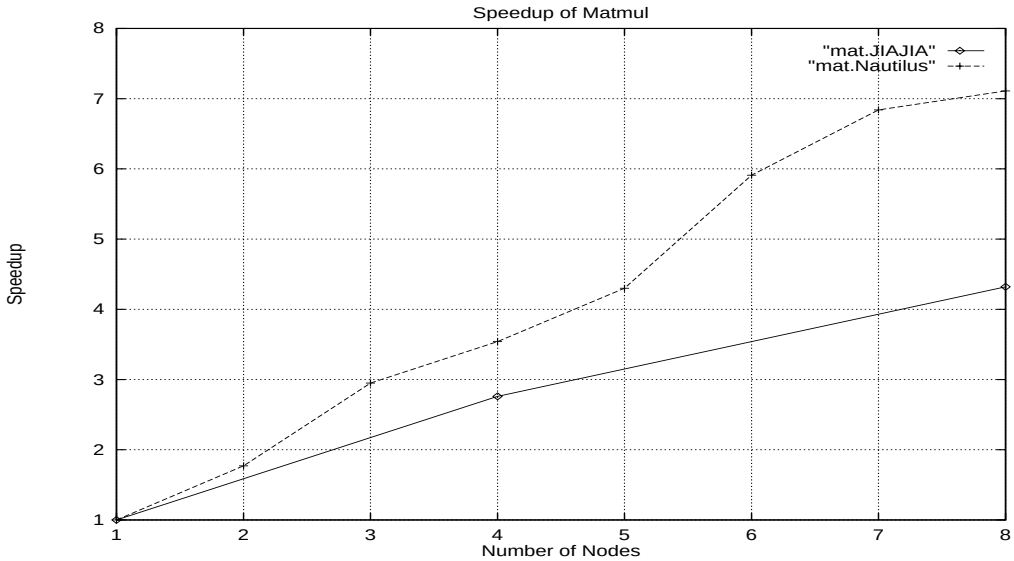


Table 1.3 execution time (in seconds) and speedups for the Matmul benchmark

(i)	*JIAJIA.t(i)*	*JIAJIA.s(i)*	*Nautilus.t(i)*	*Nautilus.s(i)*
1	186.96	1.00	186.96	1.00
2	-	-	105.86	1.77
3	-	-	63.54	2.95
4	67.69	2.76	52.80	3.54
5	-	-	43.48	4.30
6	-	-	31.61	5.91
7	-	-	27.33	6.84
8	43.18	4.32	26.30	7.11

Figure 1.3 speedups of Matmul



faults and 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, and also the lower overhead of the threads used by Nautilus help to improve the perfomance.

1.5.4 SOR

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

Table 1.4 execution time (in seconds) and speedups for the SOR benchmark (Rice University)

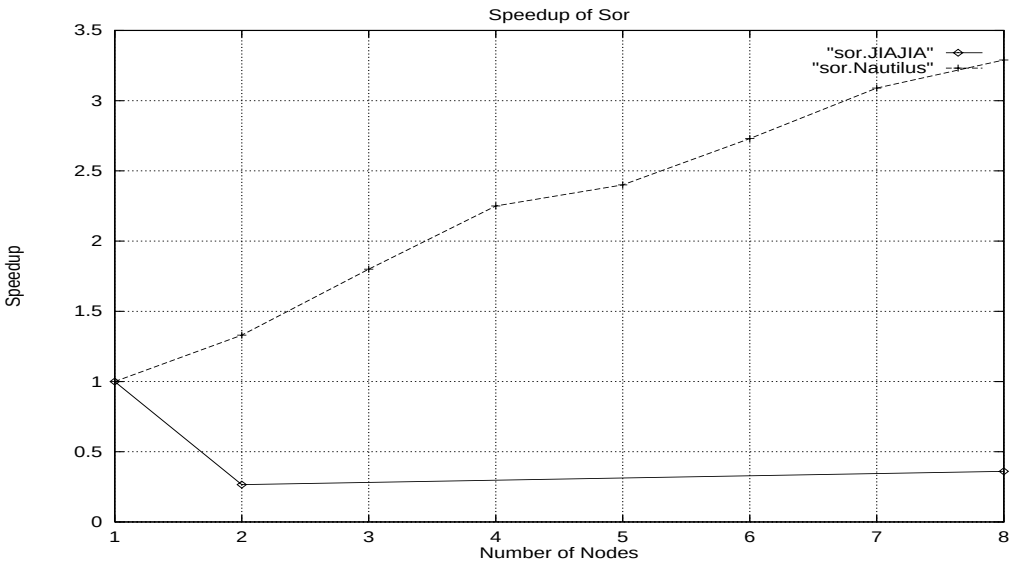
(i)	*JIAJIA.t(i)*	*JIAJIA.s(i)*	*Nautilus.t(i)*	*Nautilus.s(i)*
1	10.04	1.00	10.04	1.00
2	37.79	0.266	7.84	1.33
3	-	-	5.57	1.80
4	-	-	4.47	2.25
5	-	-	4.18	2.40
6	-	-	3.68	2.73
7	-	-	3.25	3.09
8	27.96	0.36	3.05	3.29

By observing table 1.4, for both DSM systems the times reduce according to the increasement of the number of nodes.

Still observing table 1.4 or figure 1.4, we notice that the times obtained using JIAJIA DSM are very unusual (observations 1 and 2 from item 5). Therefore, any related speedups and times are not considered for SOR analysis.

As we have said before, it's not possible to compare the speedups of the DSMs. So, instead of comparing Nautilus with JIAJIA, we can mention the study (13), where Nautilus is compared with TreadMarks. In this mentioned study, for the SOR benchmark, Nautilus is 20% faster than TreadMarks. "*We believe this happens mainly because of the better distribution of Nautilus improoving the matrix data locality and giving a lower cold start up time to distribute shared data, as a consequence improving the SOR speedup .*

Figure 1.4 speedups of SOR



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 an page fault. As we have said in Matmul evaluation, it is needless to alloc twins and diffs when a page is written in its owner, which decreases the overhead, improving the speedup of Nautilus."

1.6 CONCLUSION AND FUTURE WORKS

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

Also Nautilus, a new DSM system was evaluated and its speedup was compared with another important DSM system used by the scientific community: JIAJIA. For the applicative EP, Nautilus has an equivalent speedup to JIAJIA. For the LU applicative, Nautilus has a worse speedup of about 8.4% to 19.5% when compared with JIAJIA because of its data distribution. For the applicative Matmul, Nautilus is from 25 to 45% faster than JIAJIA. Finally, for the applicative SOR, Nautilus is until 20% faster than JIAJIA.

So, Nautilus has good speedups, comparable to other DSMs and for some of them, until better depending on the applicative. Other data distribution and techniques as suggested in (12) are undergoing study, in order to further improve Nautilus speedup.

It is possible that a more reliable version of JIAJIA for Linux be evaluated to better compare its speedup with Nautilus, since only a few results were obtained with the current version.

We intend to evaluate the DSMs chosen in this study with different input parameters to see what is the behavior of the applicative and also to evaluate other applicatives.

Also, we intend to evaluate the speedup of other DSMs against Nautilus, comparing not only their speedups, but also their number of consistency and synchronization messages and number of diffs generated.

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.

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

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

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.

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.

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

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

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

Sterling T., Becker D., Savarese D., et al. *BEOWULF: A Parallel Workstation for Scientific Computation*, Proceedings of the 1995 International Conference on Parallel Processing (ICPP), August 1995, Vol. 1, pp. 11-14.

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

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.

Chiola, G., *Gamma Project: Genoa Active Message Machine*, http:/www.disi.unige.it/project/gamma.

Marino M.D.; Campos G. L., Relative Speedup of Third Generation DSM Systems, paper submitted to ICS'99.