Homeless and Home-based Lazy Release Consistency Protocols on Distributed Shared Memory

Byung-Hyun Yu and Zhiyi Huang

Department of Computer Science
University of Otago, New Zealand
Email: byu,hzy@cs.otago.ac.nz

Stephen Cranefield and Martin Purvis

Department of Information Science
University of Otago, New Zealand
scranefield,mpurvis@infoscience.otago.ac.nz

Abstract

This paper describes the comparison between homeless and home-based Lazy Release Consistency (LRC) protocols which are used to implement Distributed Shared Memory (DSM) in cluster computing. We present a performance evaluation of parallel applications running on homeless and home-based LRC protocols. We compared the performance between TreadMarks, which uses homeless LRC protocol, and our home-based DSM system. We found that the home-based DSM system has shown better scalability than TreadMarks in parallel applications we tested. This poor scalability in the homeless protocol is caused by a hot spot and garbage collection, but we have shown that these factors do not affect the scalability of the home-based protocol.

Keywords: Distributed Shared Memory, Lazy Release Consistency, Home-based Protocol, Performance Evaluation.

1 Introduction

Distributed shared memory (DSM) has been considered as an alternative to message passing for parallel programming. By using shared data structures, programmers can design parallel algorithms easily compared with programming with message passing. DSM also can provide easy scalability through networked computers compared to standalone multiprocessor computers such as cc-NUMA (cache coherent - Non-Uniform Memory Architecture). In spite of these advantages, performance over DSM is limited by the additional communication traffic required in order to provide memory consistency over physically separated workstations. To overcome this problem, many memory consistency models and techniques have been proposed. One of them is the Lazy Release Consistency (LRC) model (Keleher, Cox & Zwaenepoel 1992), and TreadMarks (Amza, Cox, Dwarkadas, Keleher, Lu, Rajamony, Yu & Zwaenepoel 1996) has been considered as the state of the art implementation of LRC.

More recently, the home-based protocol has been proposed and performance comparison between homeless and home-based protocols have been published (Zhou, Iftode & Li 1996, Cox, de Lara, Hu & Zwaenepoel 1999). The conclusions from those two papers showed different performance results: one found no significant difference between the two protocols (Cox et al. 1999) and the other found the home-based one significantly outperformed the homeless one (Zhou et al. 1996).

Lately, we implemented a home-based DSM system based on TreadMarks. We tested our home-based DSM system by running 7 parallel applications over 32 nodes. Our preliminary performance results showed that our home-based system outperforms the homeless one significantly in the Barnes-Hut and parallel neural network applications. However, in SOR and Gauss, the home-based one showed that proper home assignment to shared pages is critical to the performance as indicated by others (Zhou et al. 1996, Keleher 1999). Discussions of the performance evaluation are presented in Section 3.

This paper is organized as follows: Section 2 explains homeless and home-based protocols and compares the two protocols. Section 3 presents performance results and discuss strengths and weaknesses of the two protocols based on our performance results. Finally, Section 4 describes future research and conclusions.

2 Protocol Comparison

2.1 Background

When a parallel application is running over many nodes by means of DSM, each process running the application shares logically the same memory space with other processes even though each process has a physically different memory space from others. Because of this physically different location of memory, a communication protocol is needed to provide coherent and consistent memory with all the processes.

A memory consistency model plays an important role in the communication protocol. Since Lamport proposed Sequential Consistency (SC) (Lamport 1979) for multiprocessor computers, many memory consistency models such as Release Consistency (RC) (Carter, Bennett & Zwaenepoel 1991) and Lazy Release Consistency (LRC) (Keleher et al. 1992) have been proposed to improve performance of applications running on shared memory by removing unnecessary communication between processes in exchange for a more complicated programming style using synchronization variables. Even though more relaxed memory consistency models than RC and LRC, such as Entry Consistency (EC) (Bershad & Zekauskas 1991) and Scope Consistency (ScC) (Iftode, Singh & Li

This research was supported by Foundation for Research, Science and Technology in New Zealand under Grants UOOX0208.

1996), have been proposed to improve the performance further, some applications have incorrect program results under those memory models unless programmers modify the source code of the applications to guarantee correct program results.

In this paper, evaluations of two implementations of the LRC protocol, TreadMarks and a Home-based DSM system, are presented. To compare them, we briefly explain page-based DSM systems since the two implementations are page-based DSM systems. In page-based DSM systems, a page is the smallest unit of memory consistency. This means that even though part of a page is modified, the whole page is considered as modified.

Whenever a node writes something on pages of DSM, it must record information about the write. The *interval* data structure is used to contain all the information about the writes between two synchronizations, which is the vector time of its creation, a list of dirty pages and *diffs*[1] in order to update other nodes. According to the LRC model, this update does not need to be propagated to all the other nodes. Also the time of propagation can be delayed until the next synchronization time. At the synchronization time, each node gets updated by the other nodes.

In order for the other nodes to find out whether or not the memory accessed is the most up-to-date, page-based DSM systems make use of the page fault mechanism from the virtual memory system. If a node accesses a stale copy of the page, a page fault happens. The faulting node must get the most up-to-date copy of the page from another node or nodes which have the most up-to-date one. At the time of the page fault, the homeless and home-based protocols behave differently. In the homeless protocol, the faulting node requests timely ordered diffs of the page from the last writer or writers. The most up-to-date page is represented as timely ordered diffs of the page which may be distributed over the nodes. In the home-based protocol, the faulting node requests the page from the home of the page which always has the most up-to-date copy.

2.2 Homeless Protocol

Since the most up-to-date page is maintained by the timely ordered diffs and the diffs are distributed over many nodes, it is complicated and time-consuming to get the page updated in the homeless protocol. To make it worse, as the number of the nodes increases, it is common that the size of the diffs of the page can exceed the size of the page due to diff accumulation.

Since the same diffs can be stored at multiple nodes and can't remove them until all nodes have them, not only is the memory requirement for storing diffs costly but also a garbage collection process is necessary during program execution, which is another complicated and time-consuming process.

It is also likely to happen that one last writer can have all the diff requests from all the other nodes at the same time, which is called a *hot spot* (Zhou et al. 1996). A hot spot can be produced when only one node writes to the shared memory before the last barrier synchronization in the main loop. A node suffering from a hot spot delays the diff request service from all the other nodes and it can significantly degrade performance of DSM applications as presented in Section 3.3.

[1] A diff is created by word-by-word comparison of a dirty page with a twin which is copied from the initial state of the page before it became dirty.

2.3 Home-based Protocol

In the home-based protocol, a home node is assigned to a page to maintain the most up-to-date page. To get the most up-to-date page, a faulting node requests it from the home of the page. When a node writes on a page which is not owned by itself, it should update the home of the page at synchronization time by sending diffs according to LRC.

Compared to the homeless protocol, the process of getting the most up-to-date page by a faulting node is much simpler and shorter since the home-based protocol asks for the most-up-date copy of the fault page, not the timely ordered diffs of the page, and the amount of data traffic is not more than the page size. It is also easy to find out where the most up-to-date page is.

In terms of the memory requirements, the home-based protocol is a clear winner. Since non-home nodes eagerly update home nodes by sending diffs at synchronization time and discard the diffs, no memory is required to retain the diffs. Furthermore, garbage collection is not needed to remove the diff memory space, which would have involved barrier-like communication traffic (Keleher, Dwarkadas, Cox & Zwaenepoel 1994).

Another advantage of the home-based protocol is that there is no diff accumulation and no multiple packet transfer for the same diff. In the homeless protocol the same diff can be transferred to the next lock owner as many as N-1 times (where N is the number of nodes) in the worst case. In the home-based protocol, however, only one diff transfer to the home of the page is required at the end of synchronization.

Finally, a hot spot is avoided in the home-based protocol because page update requests are distributed over the nodes. In Section 3.3, we will show the hot spot effect in the parallel neural network application.

Despite all the advantages, if the home assignment to the shared pages is not well matched with an application's memory access pattern, the home-based protocol will suffer (Keleher 1999). At the current stage, our implementation of the home-based DSM system uses a fixed manner of home assignment. We will present the home assignment effect in Section 3.5 with performance results of SOR and Gauss applications.

3 Performance Evaluation

To make a comparison between the homeless and home-based protocols, we chose 7 DSM applications which are typically used to evaluate DSM systems. The problem sizes and sequential execution times for them are presented in Table 1. The sequential execution times are measured using TreadMarks. We found that the sequential execution times for the home-based implementation are slightly longer than for those using TreadMarks because of the different initial page state of node 0. The initial state of pages in node 0 is Read-Write in TreadMarks and Read-Only in the home-based system. We ran those applications over our network which has 32 nodes and is connected by 100 Mbit switched Ethernet. Each node has a 350 MHz Pentium II CPU and 192 MB of memory. All nodes are running Red Hat Linux 7.2 (gcc 2.96).

Application	Problem Size, Iterations	Sequential Execution Time (secs)
NN	44,000,235	710.07
Barnes-Hut	64k Bodies,3	85.74
IS	2^{24}x2^{15},20	72.07
3D FFT	64x64x64,50	44.44
TSP	19 cities	33.07
SOR	2000x1000,50	6.21
Gauss	1024x1024,1023	15.15

Table 1: Problem Sizes and Sequential Execution Times

3.1 Applications

3.1.1 Parallel Neural Network

We implemented a parallel version of a neural network application which is trained by a two pass algorithm: forward propagation and backward propagation (Werstein, Pethick & Huang 2003). We used the shuttle data set obtained from the University of California, Irvine machine learning repository (Blake & Merz 1998). Each item contains nine numerical attributes, with 44,000 items in the set. To train the network for the data set, 235 epochs were taken. In the main loop for training the network, two barriers and two lock synchronizations were used.

3.1.2 Barnes-Hut

This application is a simulation of gravitational forces using the Barnes-Hut N-Body algorithm (Woo, Ohara, Torrie, Singh & Gupta 1995). In the main loop, three barriers are used to synchronize program execution between nodes.

3.1.3 IS

Integer Sort (IS) ranks numbers represented as an array of keys by using a counting or bucket sort (Bailey, Barszcz, Barton, Browning, Carter, Dagum, Fatoohi, Frederickson, Lasinski, Schreiber, Simon, Venkatakrishnan & Weeratunga 1991). Each key[i] is ranked by its size and sorted into rank[i]. rank[i] means the rank of key[i]. In the parallel version of DSM, three barriers are used to synchronize the program execution.

3.1.4 3-D FFT

The 3-D Fast Fourier Transform (FFT) benchmark is included in the TreadMarks application suite. This solves a partial differential equation using forward and inverse FFTs. In the main loop only one barrier synchronization happens at the end of each loop. The memory access pattern of each loop is regular.

3.1.5 TSP

The Travelling Salesman Problem (TSP) application finds the cheapest way of visiting a sequence of cities and returning to the starting point, given a finite number of cities along with the cost of travel between each pair of them. We used the implementation in the TreadMarks application suite, which uses a branch-and-bound algorithm. It uses only lock synchronization to protect the shared data to find out the minimum tour length and path.

3.1.6 SOR

We ran the Successive Over-Relaxation (SOR) program included in the TreadMarks application suite. The shared matrix is divided into N blocks of rows on which N nodes work. To minimize false sharing it allocates two matrices named 'red' and 'black'. The main loop has two barrier synchronizations, one for the black matrix and the other for the red matrix. Each node always accesses the same pages in the main loop.

3.1.7 Gaussian Elimination

Gaussian Elimination is a method for solving matrix equations of the form $Ax = b$. The application is included in the TreadMarks application suite. In the main loop, only one node finds a pivot element. After finding the pivot element all the nodes run Gaussian elimination. Each loop has one barrier synchronization before finding the pivot element.

3.2 Performance Results

Overall speed-up of the applications are presented in Table 2. Neural Network, Barnes-Hut and IS showed much better performance over the home-based protocol especially over more than 16 nodes. On the other hand, SOR and Gauss have a better speed-up over the homeless protocol. The other two applications, 3D-FFT and TSP have a slightly better speed-up over the homeless protocol. The analysis of the performance results will be discussed in Sections 3.3, 3.4 and 3.5.

3.3 Hot Spot

The hot spot phenomenon in DSM applications can severely degrade performance. The homeless protocol is likely to have a hot spot more frequently than the home-based protocol, as can be seen in our neural network application. To know why the hot spot occurs in the neural network application, we should understand the parallel algorithm to train the neural network. The algorithm is:

1. Check if the job_flag is not 'done'.
2. Each node trains the network on its part of the training set.
3. The changes obtained from nodes are combined.
4. Barrier synchronization.
5. Node 0 applies the changes and calculates a final error. If the error is smaller than the specified error value, set the job_flag to 'done'.
6. Barrier synchronization
7. Go to Step 1.

During the parallel computation in Step 2, each node get access to the pages of the shared global weights which were invalidated in the prior Step 6. Then all the nodes except node 0 will have a page fault. In response to the page fault, the two protocols behave differently.

In the homeless protocol, all the nodes except node 0 will request node 0 to send diffs since node 0 is the last writer because of the prior Step 5. These simultaneous multiple diff requests from all the other nodes except one node can be a significant bottleneck as the number of the nodes increases. Even worse, if

Application	2 nodes		4 nodes		8 nodes		16 nodes		32 nodes	
	LRC	HLRC	LRC	HLRC	LRC	HLRC	LRC	HLRC	LRC	HLRC
NN	2.0	2.0	3.9	4.1	6.5	7.4	2.2	10.3	0.4	8.3
Barnes-Hut	1.6	1.5	2.0	2.0	2.0	2.4	1.4	2.8	0.2	3.4
IS	1.9	1.9	3.3	3.6	3.8	5.9	0.9	8.0	0.4	4.5
3D FFT	0.9	0.7	1.2	0.9	1.76	1.3	2.3	1.8	1.1	1.5
TSP	1.9	2.0	3.8	3.6	6.5	4.1	5.7	3.1	2.9	2.3
SOR	2.02	0.9	3.7	1.2	5.9	1.8	6.5	2.3	2.9	1.1
Gauss	1.6	0.2	1.8	0.3	1.4	0.3	0.8	0.35	0.4	0.2

Table 2: Comparison of Overall Speed-ups between Homeless and Home-based DSM Implementations

diffs from the same page are accumulated and become larger, this diff request handling time will take more time. We confirmed this hot spot phenomenon by measuring the time breakdown of the main loop. We measured the time duration between:

- Step 1 and Step 3 which is where the main computation occurs in all the nodes.
- Step 4 which is the first barrier.
- Step 5 which is the computation only in node 0.
- Step 6 which is the last barrier.

Figure 1 presents the data obtained from node 0, which shows an abnormal barrier time increase starting from 16 nodes over TreadMarks but a minor barrier time increase over the home-based system. We also measured per-node time breakdown on 32 nodes over TreadMarks and the home-based system. As shown in Figure 2, the first barrier time in node 0 took more than 98% of the total execution time due to a hot spot. This data means that even though node 0 gets into the first barrier first, it can not get out of the barrier until all the faulting nodes are satisfied with diffs sent from node 0.

In the home-based protocol, in response to the page fault, each node requests the pages from the homes of the pages. The page requests are likely to be distributed over multiple nodes. Compared to the homeless protocol in which one node takes all the responsibility for diff request handling, the page request handling is distributed over multiple nodes. Figure 3 shows that node 0 does not suffer from a hot spot. This resulted in good performance. In the neural network application, the page request handling is distributed over the first 5 nodes. Also page request handling time is much shorter because sending a page can use a much smaller message than sending accumulated diffs in the homeless protocol.

Another proof that the hot spot affects the performance can be found by comparing the number of resent request packets. In the implementation of the two protocols, if a request is not satisfied in one second, the request is resent. As shown in Figure 4, the number of resent request packets steeply increases with the number of nodes after 16 nodes in the neural network application over the homeless protocol, but there are none in the home-based protocol.

3.4 Garbage Collection Effect

Another negative effect that the homeless protocol can have is the need for continuous garbage collection. In the Barnes-Hut application, we found that the garbage collection affected the entire performance significantly, in particular, over more than 16 nodes.

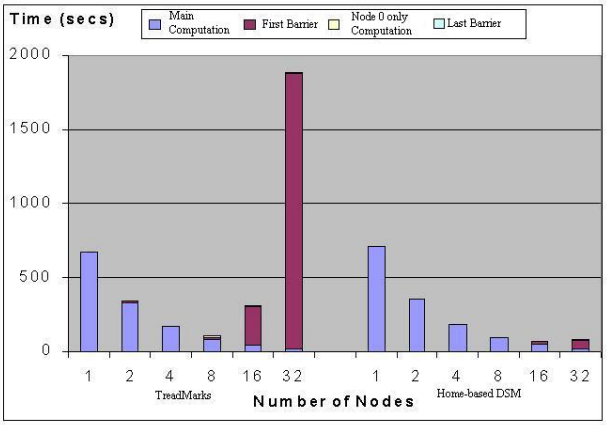

Figure 1: Time Breakdown for Neural Network in Node 0

To see how the garbage collection affects the performance negatively we measured garbage collection time over 8, 16 and 32 nodes.

In the implementation of TreadMarks, the process of garbage collection happens inside the barrier synchronization implementation if memory space for diffs and intervals exceeds some limitation. There are 9 barrier synchronizations during the main computation in Barnes-Hut. Out of the 9 barriers, garbage collection was run for 8, 16 and 32 nodes 3, 3 and 5 times respectively over TreadMarks. Because garbage collection requires barrier-like communication, it would be very costly as the number of nodes increase as shown in Table 3. For 32 nodes, the barrier time took more than 377 seconds out of the total execution time (411 seconds) over LRC. Out of 377.6 seconds, garbage collection time took 274.38 seconds, which was almost 70% of the execution time. On the other hand, Barnes-Hut over the home-based protocol showed no garbage collection and much better performance.

3.5 Home Assignment Effect

One of the weaknesses of the home-based protocol can be unnecessary data communication if the home assignment to the pages is not matched well with the memory access pattern of applications. Our performance results for SOR and Gauss showed poor performance over the home-based protocol because our fixed home assignment was not well matched with the memory access patterns. The fixed home assignment means the home assignments to the pages were *page's home = page number % number of node* where "%" is the modulus operator.

SOR and Gauss are typical matrix problems that

	8 nodes		16 nodes		32 nodes	
	LRC	HLRC	LRC	HLRC	LRC	HLRC
Execution Time (secs)	43.13	36.18	63.13	31.62	411.08	22.17
Barrier Time (secs)	18.55	13.91	36.36	14.58	377.6	8.45
Number of Garbage Collection	3	0	3	0	5	0
Garbage Collection Time (secs)	7.74	0	14.6	0	274.38	0

Table 3: Garbage Collection Effect

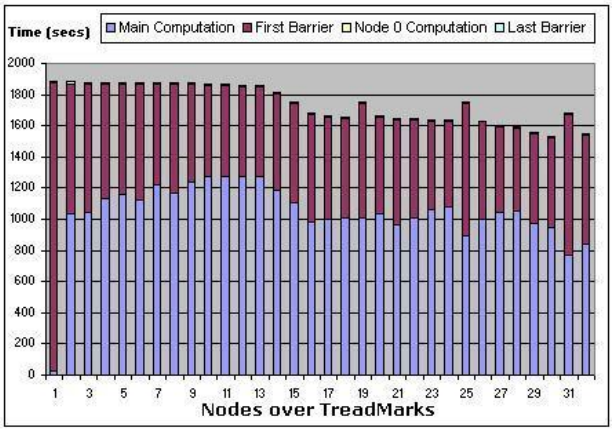

Figure 2: Time Breakdown for Neural Network in 32 nodes over TreadMarks

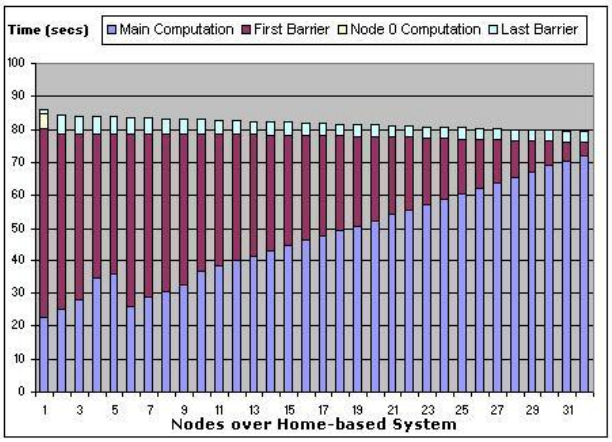

Figure 3: Time Breakdown for Neural Network in 32 nodes over the Home-based System

can be parallelized by dividing the matrix. Each divided sub-matrix is processed on each node and the sub-results are collected at the barrier time. The two applications show a regular memory access pattern meaning that each node always works on the same sub-matrix. This kind of memory access pattern should not generate much data traffic between nodes other than the synchronization for the boundary of each sub-matrix. However this is not the case for the fixed manner of home assignment. According to the home-based protocol, each node still needs to update the homes of the pages that have been changed. This will generate unnecessary data traffic which could be avoided if the pages were assigned to the home nodes that exclusively access the pages.

We measured this home effect by manually assigning the right home nodes to the pages. As expected

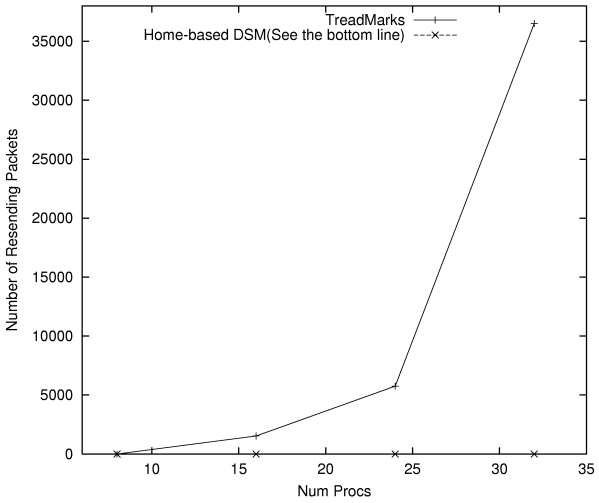

Figure 4: Comparison of the Number of Resent Request Packets in the Neural Network Application

and shown in Figures 5 and 6, the performances of SOR and Gauss with the manipulated home assignment improved significantly.

3.6 Data Communication Traffic

In Table 4, the number of messages and the amount of data traffic can be compared between the two protocols. In Barnes-Hut we see that even though the amount of data transferred over the home-based protocol is more than for the homeless protocol, the performance over the home-based protocol is better than over the homeless protocol, in particular, over 32 nodes. Similarily, though the number of messages and the amount of data traffic in Neural Network are not much different, the performance over the home-based protocol is much better than over the homeless protocol. From the speed-up and data traffic of Neural Network and Barnes-Hut, we can conclude that burst data traffic over a short time, for example at the barrier synchronization time, and burst request service to one node can be more critical to overall performance than the amount of the overall data traffic as explained in Section 3.3.

Nevertheless, in general, the greater the amount of data and the number of messages, the worse the performance as clearly shown in SOR and Gauss over the fixed home assignment. When the home nodes were properly assigned by us according to the memory access pattern of the applications, the number of messages and the amount of data traffic were significantly reduced as shown in Table 5.

Application	8 nodes				16 nodes				32 nodes			
	Number of Messages		Amount of Traffic(MB)		Number of Messages		Amount of Traffic(MB)		Number of Messages		Amount of Traffic(MB)	
	LRC	HLRC	LRC	HLRC	LRC	HLRC	LRC	HLRC	LRC	HLRC	LRC	HLRC
NN	57993	61848	78.8	88.8	121208	127294	177.9	180	272508	259956	403.0	378.3
Barnes-Hut	889912	218745	330.7	498.8	2803645	410022	621.8	891.8	8696047	435632	1212.3	924.8
IS	21390	22746	141.9	51.2	62730	55960	604.4	119.7	197502	139179	2326.9	285.2
3D FFT	107079	99463	205.6	545.3	134505	127483	223.1	612.3	320947	307382	453.9	837.1
TSP	18134	17533	6.0	27.9	25622	23791	13.4	393.3	34315	28938	28.5	493.3
SOR	7135	12630	8.0	63.8	11537	37000	10.3	77.5	20129	123208	17.2	99.0
Gauss	55545	95676	56.4	1322.7	120705	205016	124.4	1501.1	250454	569828	259.2	1749.7

Table 4: Comparison of Data Communication Traffic between Homeless and Home-based Protocols

	8 nodes		16 nodes		32 nodes	
	Number of Messages	Amount of Traffic(MB)	Number of Messages	Amount of Traffic(MB)	Number of Messages	Amount of Traffic(MB)
SOR	4316	6.3	9244	14.4	19100	33.2
Gauss	42155	66.9	91171	155	188915	356.2

Table 5: Data Communication Traffic over the Manipulated Home Assignment

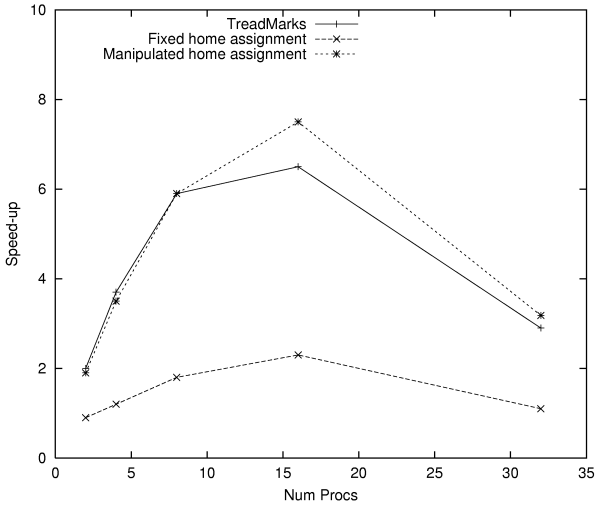

Figure 5: Home Effect in SOR

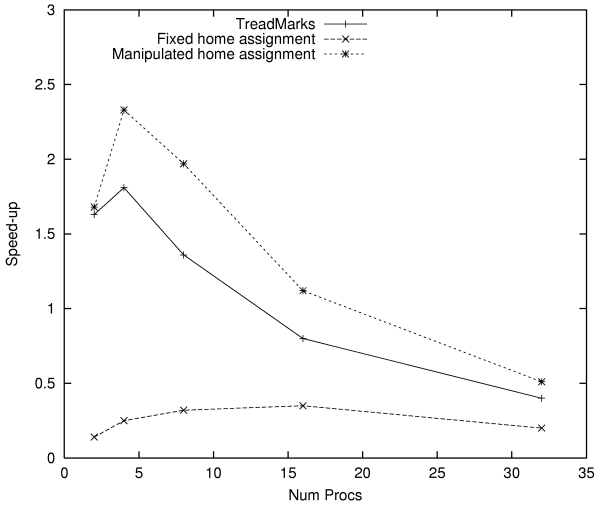

Figure 6: Home Effect in Gauss

4 Conclusions and Future Research

In this paper we presented a performance evaluation of homeless and home-based LRC protocols. For this, we compared the performance of 7 applications over TreadMarks, which is the state of the art of homeless LRC protocol implementation, and our home-based DSM system.

The performance results showed that the homeless protocol is less scalable compared to the home-based protocol. The neural network and Barnes-Hut applications showed abnormal super degradation of the performance over the homeless protocol for 16 nodes and more. This super degradation is caused by a hot spot and garbage collection.

We have concluded that the homeless protocol is likely to get a hot spot more frequently than the home-based protocol. Also, garbage collection which is necessary in the homeless protocol can reduce performance of DSM applications. On the other hand, the home-based protocol is not affected by these factors.

However, performance over the home-based protocol is very sensitive to the home assignment to the pages as shown in SOR and Gauss. If the home is assigned to the pages that are exclusively accessed, it will significantly reduce data traffic and the number of page faults.

In conclusion, if the homes of the pages are well assigned according to the memory access pattern of DSM applications, the home-based protocol will be likely to be superior to the homeless protocol. In particular, since the number of nodes over DSM are likely to become larger in the future, the home-based protocol will be the best answer for scalable DSM.

Our future research will be concentrated on finding a better memory consistency model for the home-based protocol. Though the LRC model used in TreadMarks and our home-based DSM system provides correct and efficient memory consistency for DSM, Scope Consistency and our View-based Consistency (VC) (Huang, Sun, Cranefield & Purvis 2001) models can improve DSM performance by more selectively and accurately invalidating or updating pages.

Acknowledgment

The authors would like to thank Mark Pethick who implements the parallel neural network application.

References

Amza, C., Cox, A., Dwarkadas, S., Keleher, P., Lu, H., Rajamony, R., Yu, W. & Zwaenepoel, W. (1996), 'Treadmarks: Shared memory computing on networks of workstations', *IEEE Computer* **29**(2), 18–28.

Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Dagum, D., Fatoohi, R. A., Frederickson, P. O., Lasinski, T. A., Schreiber, R. S., Simon, H. D., Venkatakrishnan, V. & Weeratunga, S. K. (1991), 'The NAS Parallel Benchmarks', *The International Journal of Supercomputer Applications* **5**(3), 63–73.

Bershad, B. N. & Zekauskas, M. J. (1991), Midway: Shared memory parallel programming with entry consistency for distributed memory multiprocessors, Technical Report CMU-CS-91-170, Carnegie Mellon University, Pittsburgh, PA (USA).

Blake, C. & Merz, C. (1998), 'UCI repository of machine learning databases'. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Carter, J. B., Bennett, J. K. & Zwaenepoel, W. (1991), Implementation and performance of Munin, *in* 'Proc. of the 13th ACM Symp. on Operating Systems Principles (SOSP-13)', pp. 152–164.

Cox, A. L., de Lara, E., Hu, C. Y. C. & Zwaenepoel, W. (1999), A performance comparison of homeless and home-based lazy release consistency protocols for software shared memory, *in* 'Proc. of the 5th IEEE Symp. on High-Performance Computer Architecture (HPCA-5)'.

Huang, Z., Sun, C., Cranefield, S. & Purvis, M. (2001), View-based consistency and its implementation, *in* 'Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001)', pp. 74–81.

Iftode, L., Singh, J. P. & Li, K. (1996), Scope consistency: A bridge between release consistency and entry consistency, *in* 'Proc. of the 8th ACM Annual Symp. on Parallel Algorithms and Architectures (SPAA'96)', pp. 277–287.

Keleher, P., Cox, A. L. & Zwaenepoel, W. (1992), Lazy release consistency for software distributed shared memory, *in* 'Proc. of the 19th Annual Int'l Symp. on Computer Architecture (ISCA'92)', pp. 13–21.

Keleher, P., Dwarkadas, S., Cox, A. L. & Zwaenepoel, W. (1994), Treadmarks: Distributed shared memory on standard workstations and operating systems, *in* 'Proc. of the Winter 1994 USENIX Conference', pp. 115–131.

Keleher, P. J. (1999), Symmetry and performance in consistency protocols, *in* 'International Conference on Supercomputing', pp. 43–50.

Lamport, L. (1979), 'How to make a multiprocessor computer that correctly execute multiprocess programs', *IEEE Transactions on Computers* **C-28**(9), 690–691.

Werstein, P., Pethick, M. & Huang, Z. (2003), A performance comparison of dsm, pvm, and mpi, *in* 'Proceedings of the 4th International Conference on Parallel and Distributed Computing, Applications and Technologies', SW Jiaotong University, Chengdu, China, pp. 476–482.

Woo, S. C., Ohara, M., Torrie, E., Singh, J. P. & Gupta, A. (1995), The SPLASH-2 programs: Characterization and methodological considerations, *in* 'Proceedings of the 22th International Symposium on Computer Architecture', Santa Margherita Ligure, Italy, pp. 24–36.

Zhou, Y., Iftode, L. & Li, K. (1996), Performance evaluation of two home-based lazy release consistency protocols for shared memory virtual memory systems, *in* 'Proc. of the 2nd Symp. on Operating Systems Design and Implementation (OSDI'96)', pp. 75–88.