Marino, Mario D.
Two Techniques for Improvement the Speedup of Nautilus DSM
Computación y Sistemas, vol. 4, núm. 2, octubre-diciembre, 2000, pp. 166-177
Instituto Politécnico Nacional
Distrito Federal, México

Available in: http://www.redalyc.org/articulo.oa?id=61540209
Two Techniques for Improvement the Speedup of Nauttilus DSM

Mario D. Marino
Computing Engineering Department - Polytechnic School of the University of Sao Paulo
Av. Prof. Luciano Gualberto 158, trav 3, Cidade Universitaria
Predio de Engenharia de Electricidade, Depto. Computacao
sala C2-24, CEP 05508-900
e-mail: mario@regulus.pcs.usp.br

Article received on February 15, 2000; accepted on August 23, 2000

Abstract

Nauttilus is a home-based, page-based, multi-threaded and scope consistency-based DSM system. In this study, two techniques for improve the speedup of Nauttilus DSM are investigated: write detection and page aggregation. The write detection technique consists of maintaining the pages writable only on the home nodes and only detecting writes on the cache copies in a page-based DSM. A consequence, this technique generates a less number of page faults and page requests, consequently better speedups can be achieved. The page aggregation technique consists of considering a larger granularity unit than a page, in a page-based DSM system. In order to have a fair and homogeneous comparison, a demo version of TreadMarks and JIAJIA DSM were included in this study. The benchmarks evaluated in this study are EP (from NAS), SOR (from Rice University), LU and Water N-Squared (both from SPLASH-2).

Keywords: distributed, shared, memory, DSM.

1 Introduction

The evolution and the decrement of costs of workstations (NOWs) the most used as computer. Big projects such as Beowulf (Merkey, 1997) can be mentioned to exemplify.

The Distributed Shared Memory (DSM) (Li, 1996; Stum and Zhou, 1990), which is widely discussed for the last 9 years, is an approach of shared memory which permits viewing one of workstations as a shared memory parallelism.

Some important DSMs, Munin(Carr et al., 1995; Swanson et al., 1995; TreadMarks(Klehe, 1995), CVM(Klehe, 1995), AJIA(Eskicioglu, 1999; Hu et al., 1998), and Nauttilus(Marin and Campos, 1999; Marin and Campos, 1999b), are page-based DSM systems. Page-based DSMs have achieved good speedups for several benchmarks, but there is still available place for improvements (Iftode et al., 1999).

In order to give a speedup improvement, in this study two techniques are evaluated for Nauttilus System:

- write detection(Hu et al., August 1999; August 1999b);

Write detection is an essential mechanism to multiple-writer protocols to identify write pages. In order to implement multiple-writing protocols, only the writes that are effective (within the scope of the DSM) are detected.
write detection scheme used in Nautilus (Marino and Campos, 1999a; Marino and Campos, 1999b) is based on the scheme proposed by Hu (Hu et al., August 1999a; Hu et al., August 1999b) for home-based DSMs as JIAJIA (Eskicioglu, 1999; Hu et al., 1998a). In home nodes, a write to a shared page is detected and this page will remain to be written by the home node until it is written by another node. Thus, in this interval, the page only is written by its home node and no write detection is necessary, decreasing the number of page faults and the overhead, thus improving its speedup.

In page-based DSM systems, shared memory accesses are detected using virtual memory protection, thus one page is the unit of access detection and can be used as a unit of transfer. Depending on the memory consistency model and the situation, also the diff’s are used as an unit of transfer. For example, in home lazy release consistency (LRC), such as TreadMarks, if the node has a dirty page, diff’s are fetched from several nodes, when an invalid page is accessed. On the other hand, in JIAJIA, pages are fetched from the home nodes when a remote page fault occurs.

The unit of access detection and the unit of transfer can be increased by using a multiple of the hardware page size. In this way, if an aggregation of several pages is done, false sharing is increased. Besides, aggregation can reduce the number of messages exchanged. If a processor accesses several pages successively, a single page fault request and reply can be enough, instead of multiple exchanges, which are usually required. A secondary benefit is the reduction of the number of page-faults. On the other hand, false sharing can increase the amount of data exchanged and the number of messages (Amza et al., 1999).

One of the main goals of this paper is to evaluate the write detection scheme for Nautilus and its influence on Nautilus’s speedup. In order to have a reference parameter of speedups, two DSMs are included in this study: TreadMarks (Keleher, 1995) and JIAJIA (Eskicioglu, 1999; Hu et al., 1998a). These two DSMs are well-known by the scientific community as optimal speedups in DSM area.

Other main goals of this paper is to evaluate the page aggregation technique (Amza et al., 1999) in Nautilus DSM system. It will be investigated what is the influence of this technique, two grain sizes are used for Nautilus and 8kB.

This study is an original contribution because a study of Amza (Amza et al., 1999) is applied with TreadMarks, a lazy release consistency home-based DSM, this technique until the present was not applied to home-based and scope consistency, multi-threaded DSMs for Unix DSM, which are Nautilus’s features. In addition, this is the first study which combines both techniques, write detection and page aggregation, and evaluated on a DSM with Nautilus features.

TreadMarks, a reference of optimal speedups in the scientific community, is included in the comparison to have a reference parameter of speedups. Unfortunately, the results from write detection technique applied to TreadMarks DSM will not be shown nor compared here because the version (1.0.3) used in this study is a demo version, therefore, the source code is not available.

The speedup results from write detection technique applied to JIAJIA DSM (version 2.1) will be shown nor compared here because, this technique showed any meaningful improvement in its speedup, probably due some implementation problem. The paper presents a comparison with JIAJIA DSM.

The evaluation comparison for write detection and page aggregation is done by applying different benchmarks: EP (from NAS), LU (kernel from SPLASH-2) (Woo et al., 1995), SOR (from Rice University Water N-Squared) (from SPLASH-2). The environment of the comparison is an eight-PC’s network connected by a fast-Ethernet shared media. The client system used in each PC is Linux (2.x). Based on the combination of write detection and page aggregation, the combinations of these techniques can be created: i) traditional virtual memory write detection with 4kB of page size; ii) traditional virtual memory write detection with 8kB of page size; iii) write coherence with 4kB of page size; iv) write detection with 8kB of page size.

In section 2 a brief description of Nautilus DSM is given. In section 3, JIAJIA is described. In section 4, TreadMarks is briefly described. In section 5, write detection mechanism for Nautilus is detailed. In section 6, aggregation of messages and page aggregation are presented.


2 Nautilus DSM

The main function of the new software DSM Nautilus is to develop a DSM with a simple consistency memory model, in order to provide good speedups, and also another one with a simpler user interface, totally compatible with TreadMarks and JIAJIA. This idea is very similar to the ideas utilized by JIAJIA, mentioned in the studies of Hu(Hu et al., 1998a) and Eskicioglu(Eskicioglu, 1999), but Nautilus makes use of some other techniques, which distinguishes it from JIAJIA. These techniques will be mentioned below. In order to be portable, it was developed as a runtime library like TreadMarks, CVM and JIAJIA, because there is no need to change the operating system kernel(Carter, 1993).

Nautilus is a page-based DSM, as TreadMarks and JIAJIA. In this scheme, pages are replicated through the several nodes of the net, allowing multiple reads and writes(Stum and Zhou, 1990), thus improving speedups. By adopting the multiple writer protocols proposed by Carter(Carter, 1993), false sharing is reduced and good speedups can be achieved. The mechanism of coherence adopted is write invalidation(Stum and Zhou, 1990), because several studies (Carter et al., 1995; Eskicioglu, 1999; Iftode et al., 1999; Keleher, 1995) show that this type of mechanism provides better speedups for general applications. Nautilus, as JIAJIA does, uses scope consistency model, which is implemented through a locked-based protocol(Hu et al., 1998b).

Nautilus is the first multi-threaded DSM system implemented on top of a free Unix platform that uses the scope consistency model because:

1) there are versions of TreadMarks implemented with threads, but it does not use the scope consistency memory model;

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

3) CVM(Keleher, 1996) is a multi-threaded DSM system, but it uses lazy release consistency and at the moment, it does not have a Linux based version.

4) Brazos(Speight and Bennet, 1997) is a multi-threaded DSM and uses the scope consistency, but it DSM: threads to minimize the switch costs of SIGIO signals(which notice the arrival of messages); v) minimization of diffs primitives compatible with TreadMarks, JIAJIA; vi) network of PCs and Linux 2.0 protocols.

Nautilus is different from other DSM ways. First, its implementation is multi-threaded, it minimizes the context switches overhead, it does not use SIGIO signals in its implementation. Second, as JIAJIA does, Nautilus uses shared memory using a home-based scheme, directory structure of all pages instead of a hierarchy of the relevant pages (cached), used. Third, a different memory organization from JIAJIA.

To improve the speedup of the application, Nautilus uses two techniques: i) multi-threaded implementation; ii) diffs of pages that were modified by the owner are not created.

The multi-threaded implementation of Nautilus's main advantages: 1) minimization of context switch; 2) minimization of use of SIGIO signals.

The major part of all DSM systems currently implemented on top of an Unix platform uses SIGIO signals to activate a handler to process the arrival of messages which come from the network. Some examples of DSMs that use the SIGIO signal to communicate with TreadMarks and JIAJIA. One of the threads which is blocked trying to read messages from the network, is blocked, it remains asleep, thus non context-sensitive. This technique decreases the overhead of the network but allows to give as much CPU time as possible to the program. Thus, Nautilus is the first scope consistency DSM system of the second generation which uses the SIGIO signal in its implementation. A multi-threaded implementation permits to minimize this overhead to take SIGIO signals and to call the respective handler, in all arrivals of messages.

2.1 Lock-based Coherence protocol

Nautilus follows the lock-based protocol used in JIAJIA(Hu et al, 1998b), because of its simplicity minimizing the overheads: the pages can be locked for coherency check and to access their respective data.
In Nautilus, the owner nodes of the page need to send the diffs to other nodes, according to the scope consistency model. So, diffs of pages owned by the owner are not created, which is more efficient than the lazy diff creation of TreadMarks.

In Unix with the mprotect() primitive, where the state of a page can be in RO, INV or RW states, thus their state can be changed easily.

2.2 Data Distribution and Concurrency Related Informations

Nautilus distributes its shared pages across the nodes and each shared pages has a home node. When remote nodes access their home pages, no page faults occur. When remote pages are accessed, page faults occur, and these pages are fetched from their home nodes and cached locally. Instead of JIAJIA, Nautilus have a replacing mechanism of cached pages. In Linux, they are replaced as memory size increases.

Nautilus uses the scope consistency model (Iftode et al., 1996), where the coherence of the pages is maintained through write-notices. A lock (lock-based). As a result from the multi-protocol technique application, diffs are sent to the home nodes. The implementation of Nautilus' mechanism is very similar to JIAJIA, believed that with less time consuming and a smaller number of messages, it becomes more efficient than the acquire/release mechanism of Nautilus as described. In order to signal the end of the section, a release message is sent to the manager, indicating that the written pages are piggy-backed on the release message. On the acquire, the processor which reads the locks sends a lock request to the manager. When the lock is acquired, the manager piggy-backs write-notices associated with this lock on the grant message. When the acquire, the processor, which is doing it, invalidates the cached pages that are notified as obsolete by the manager.
3 JIAJIA

JIAJIA (Eskicioglu, 1999; Hu et al., 1998a) is another 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. So, diffs are transmitted in each critical section to maintain the consistency. Thus, 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.

To summarize the JIAJIA features: i) scope consistency (Iftode et al., 1996) home based, minimizing the number of consistency messages through the net; ii) multiple writer techniques; iii) primitives compatible with TreadMarks; iv) UDP protocols, minimizing network protocols overhead; v) data distribution can be chosen by the user (over the network nodes).

The main objective of JIAJIA (Eskicioglu, 1999; Hu et al., 1998a) is to make the minimization of the overheads of diff creation and storage as simple as possible, thus minimizing the number of consistency messages through the net. 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, which allows for better speedups.

4 TreadMarks

The consistency model used by TreadMarks is the lazy release consistency (Keleher, 1995), so the propagation of the modifications which occurred during a critical section are delayed until the next acquire. By using multiple writer protocols and the lazy release consistency model, the speedups of TreadMarks are well-known, making it one of the most used DSM systems.

To summarize TreadMarks features: i) lazy release consistency and its variations (Keleher, 1995), minimizing the number of consistency messages through the network.

The speedups of TreadMarks made it to be used by the scientific community as a reference for DSM speedups. Thus, it makes sense to compare other DSMs in order to have an accurate estimation of its performance.

The efficiency of TreadMarks is mainly due to its lazy release consistency model. The major drawback of adopting this model is the need to store the diffs all over the memory to store the diffs all over the system execution. Thus, the size of the memory used to evaluate the speedups of TreadMarks can be compromised if there is not enough memory to execute the program or if the open file does swap. If it cannot use enough size and benchmarks, the computation versus system overhead becomes unfavorable for using a DSM system.

5 Write detection

Nautilus has several common features with JIAJIA, i.e., by observing the item 3, where JIAJIA were mentioned, from feature i) to iv) Nautilus is similar to JIAJIA, which permits Nautilus to adopt JIAJIA’s write detection scheme.

As other DSMs like TreadMarks (Keleher, 1995), JIAJIA (Eskicioglu, 1999; Hu et al., 1998a), Nautilus uses virtual memory page faults to detect write to shared pages. Shared pages are protected at the beginning of an interval (several critical sections). The first write to a shared page occurs when the signal is delivered, and in this moment, the page must be written without protection. At the end of the critical section, as JIAJIA does, Nautilus sends notices related about the shared pages.

Several studies (Amza et al., 1999; Beamer et al., 1993; Carter, 1993; Eskicioglu, 1999; Hu et al., 1998a; Iftode et al., 1999; Keleher, 1995; Keleher, 1995) show that the detection of writes to shared pages has significant overheads. Other studies show that applications with large shared data set and good data distribution, the writes hit the home. The above studies enable to conclude that for applications with large shared data set and good data distribution, if the Nautilus is written, it can be used with the write detection scheme.
the home node writes to this page. Concluding, if the home page is written in some interval, several mprotect() handlers calls are saved, improving the DSM's speedup. If the home page is not written by its home in the interval, some unnecessary invalidations of remote cached pages can occur, thus more remote accesses.

The study of Amza (Amza et al., 1999) showed that in many applications, single-writer constitutes the dominant part of the sharing behavior and shared pages are normally written by the home node (owner) for a certain interval. Nautilus implements its write detection scheme which recognizes automatically a single write to a shared page by its home node, presuming that the page will continue to be written by its home node until the page is written by remote nodes.

6 Page Aggregation

In terms of implementation, following the other DSMs directions, in Nautilus there is a handler responsible for request a page from a remote node when a segmentation fault occurs. Following the Figure 1 diagram, when a page is accessed and it is in the INV state a SIGSEV signal is generated and the respective handler, as it was said before, requests the page from the home node. When the page arrives the primitive mprotect() changes the state from INV to RO.

When the page is written, another SIGSEV signal is generated and the primitive mprotect() changes the state of the page from RO to RW, as can be seen in Figure 1. As is shown in this figure, after the generation of the diffs, also with the mprotect() primitive, pages go to RO state again. And, when the write-notices arrive, indicating the pages are modified by other nodes, pages go to INV state (again with the use of mprotect() primitive). The primitive mprotect() permits to consider a granularity multiple of a page, thus giving the same permission for a region multiple of a page. Thus, this fact gives the condition to modify more than one page at the same time, which is named page aggregation technique.

The study (Amza et al., 1999) says that if

which are usually required. The study (Amza et al., 1999) also shows that there is a reduction of the amount of page fault, but false sharing can increase the number of data exchanged and the number of messages.

By changing the page size default (4kB) for example, 8kB using mprotect() primitive in Nautilus is possible to evaluate the effects of the increase in page fault reduction in the speedups.

7 Experimental Platform and Applications

The results reported here are collected on a PC's network. Each node (PC) is equipped with a 2.33 MHz (AMD) processor, 64 MB of memory, a fast Ethernet card (100 Mbit/s). The network is interconnected with a hub. In order to measure speedups, the network above was completely isolated from any other external networks. Each PC is running Red Hat 6.0. The experiments are executed by the other user process.

Related to page aggregation, two sizes are considered for page size: 4kB, which is the default of the hardware) and 8kB, which is multiple of 4kB related to write detection, a version with this size will be compared to a version without this technique.

Based on the combination of write detection and page aggregation techniques, as it has already been mentioned, four versions of Nautilus DSM were evaluated and compared to TreadMarks and JIA: two are with Nautilus using: 1. traditional virtual memory write detection with 4kB of page size; 2. virtual memory write detection with 8kB of page size; 3. write detection with 4kB of page size; 4. write detection with 8kB of page size.

The test suite includes some programs: NAS, LU (from SPLASH-2 (Woo et al., 1995) from Rice University) and Water N-Square (from SPLASH-2). SPLASH-2 is a collection of parallel applications implemented to evaluate and design new algorithms.
nization in this program is summing up a ten-integer list in a critical section at the end of program (Hu et al., 1998a). The parameter used in EP program is $M=2^{24}$.

The LU kernel from SPLASH-2 factors a dense matrix into the product of a lower triangular and upper triangular matrix. The $N \times N$ matrix is divided into an $n \times n$ array of $b \times b$ 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. 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 (Hu et al., 1998a).

SOR from Rice University solves partial differential equations (Laplace equations) with an Over-Relaxation method. There are two arrays, black and red array allocated in shared memory. Each element from the red array is computed as an arithmetic mean from the black array and each element from the black array is computed as an arithmetic mean from the red array. Communication occurs across the boundary rows on a barrier. The SOR from Rice University solves Laplace partial equations. For a number of iterations it has two barriers each iteration and communication occurs across boundary rows on a barrier. The communication does not increase with the number of processors and the relation communication/computation reduces as the size of problem increases (Hu et al., 1998a).

Water is an N-body molecular simulation program that evaluates forces and potentials in a system of water molecules in the liquid state using a brute force method with a cutoff radius. Water simulates the state of the molecules in steps. Both intra- and inter-molecular potentials are computed in each step. The most computation- and communication-intensive part of the program is the evaluation of the force vector.

<table>
<thead>
<tr>
<th>Application</th>
<th>EP</th>
<th>LU</th>
<th>Water</th>
</tr>
</thead>
<tbody>
<tr>
<td>t(1)</td>
<td>121.80</td>
<td>350.90</td>
<td>2983.00</td>
</tr>
<tr>
<td>t(8).Tmk</td>
<td>15.62</td>
<td>54.45</td>
<td>403.20</td>
</tr>
<tr>
<td>t(8).JIA</td>
<td>15.60</td>
<td>65.44</td>
<td>429.54</td>
</tr>
<tr>
<td>t(8).NautV4k</td>
<td>15.65</td>
<td>54.32</td>
<td>426.40</td>
</tr>
<tr>
<td>t(8).NautWD4k</td>
<td>15.67</td>
<td>49.60</td>
<td>422.07</td>
</tr>
<tr>
<td>t(8).NautV8k</td>
<td>15.66</td>
<td>55.52</td>
<td>426.69</td>
</tr>
<tr>
<td>t(8).NautWD8k</td>
<td>15.67</td>
<td>49.28</td>
<td>427.55</td>
</tr>
<tr>
<td>Sp.Tmk</td>
<td>7.80</td>
<td>6.44</td>
<td>7.39</td>
</tr>
<tr>
<td>Sp.JIA</td>
<td>7.81</td>
<td>5.36</td>
<td>6.94</td>
</tr>
<tr>
<td>Sp.NautV4k</td>
<td>7.78</td>
<td>6.46</td>
<td>7.00</td>
</tr>
<tr>
<td>Sp.NautWD4k</td>
<td>7.77</td>
<td>7.07</td>
<td>7.07</td>
</tr>
<tr>
<td>Sp.NautV8k</td>
<td>7.77</td>
<td>6.45</td>
<td>6.99</td>
</tr>
<tr>
<td>Sp.NautWD8k</td>
<td>7.78</td>
<td>7.12</td>
<td>6.98</td>
</tr>
<tr>
<td>SG.JIA</td>
<td>1</td>
<td>5882</td>
<td>106</td>
</tr>
<tr>
<td>SG.NautV4k</td>
<td>2</td>
<td>7980</td>
<td>10210</td>
</tr>
<tr>
<td>SG.NautWD4k</td>
<td>2</td>
<td>440</td>
<td>824</td>
</tr>
<tr>
<td>SG.NautV8k</td>
<td>1</td>
<td>5029</td>
<td>855</td>
</tr>
<tr>
<td>SG.NautWD8k</td>
<td>1</td>
<td>245</td>
<td>671</td>
</tr>
<tr>
<td>gp.JIA</td>
<td>1</td>
<td>3110</td>
<td>1924</td>
</tr>
<tr>
<td>gp.NautV4k</td>
<td>1</td>
<td>1528</td>
<td>505</td>
</tr>
<tr>
<td>gp.NautWD4k</td>
<td>1</td>
<td>340</td>
<td>448</td>
</tr>
<tr>
<td>gp.NautV8k</td>
<td>1</td>
<td>1232</td>
<td>440</td>
</tr>
<tr>
<td>gp.NautWD8k</td>
<td>1</td>
<td>195</td>
<td>331</td>
</tr>
</tbody>
</table>

Table 1: Table comparing TreadMarks, JIAJIA, and Nautilus (virtual and write detection, with 4kB and 8kB)

8 Result Analysis

Before presenting the results and their analysis, it is necessary to emphasize that the execution number of nodes=1 in all evaluated benchmarks obtained from the sequential version of the code without any DSM primitive. So, the primitive allocate memory to obtain the sequential number of nodes=1 is `malloc()`, default programming.

In order to have an accurate, homogeneous comparison, the same programs are executed on TreadMarks (version 1.0.3), JIAJIA (version 0.0.1), and Nautilus (version 0.0.1).
ii) bigger input sizes: the shared memory size is limited in this version;
iii) the source code was not available: only time and speedups can be obtained from this version, thus it was not possible to obtain number of page faults and SIGSEGV signals.

The data input size N used in the LU and SOR evaluation is 1792x1792; the number of iterations for the SOR benchmark is 10. The number of steps used in Water is 25 and the number of molecules is 1728. For EP, M = 2.4

Table 1 shows some features and results of the benchmarks: sequential time (t(1)), eight-processor parallel run time (8), speedup (Sp), remote get page request counts (gp) and number of local SIGSEGV of Nautilus(SG). The sequential time t(1) was obtained from the sequential program without DSM primitives and malloc() primitive. In order to evaluate the write detection speedup, remote get page request counts and the number of local SIGSEGVs of Nautilus are taken. For Table 1, Tmkn means TreadMarks, JIA means JIAJIA, NautV means Nautilus with the traditional virtual memory write detection and NautWD means Nautilus with the write detection. NautWD assumes that a page will only be written by its home in the future barrier interval, keeping its home page writable if only the home writes to it. When 4k is mentioned, it means that Nautilus is using grain size of 4kB and 8k means Nautilus using 8kB of grain size. Thus, for Nautilus using: i) Nautilus versions with 8kB of page size: Naut8k; ii) Nautilus versions with 4kB: Naut4k; iii) traditional virtual memory write detection with 4kB of page size, the notation is NautV4k; iv) traditional virtual memory write detection with 8kB of page size: NautV8k; v) traditional virtual memory write detection versions: NautV; vi) write detection with 4kB of page size: NautWD4k; vii) write detection with 8kB of page size: NautWD8k; viii) write detection versions: NautWD.

Except for EP benchmark, some general conclusions can be taken from Table 1: the number of SIGSEGVs and the number of page requests decreased when both mechanisms, write detection and page aggregation, not write it, the assumption causes unnecessary validation of remote cached pages and consequently other page requests. This justifies the decrease in the number of SIGSEGVs and page requests. To aggregation, it was said before that if the page requests increases, the number of page requests and SIGSEGV decreases, which justifies the observed behavior.

Now, for each benchmark, the behavior of techniques is analyzed.

8.1 EP

EP (NAS) has a small amount of communication, a small number of messages transmitted through the network, as can be observed from Table 1. By looking at Figure 2, all Nautilus's versions, JIAJIA and Marks have similar speedups, only differing by 3%. Due to the small number of messages generated by this applicative, it was possible neither to observe the difference when the write detection nor the page aggregation techniques, when both were applied.
improve Nautilus speedup.

Analyzing the write detection technique in LU, matrices are distributed across processors in a way that each processor writes to its home part of the matrices in the computing. (Becker and Merkey, 1997) Since the computation of an iteration is synchronized with barriers and passing a barrier causes all shared pages to be write-protected in traditional virtual memory, page faults occur for writing all home pages in an iteration. The method does not write protect shared pages on a barrier, and writing to home pages of a processor can process without any SIGSEGV.

From Table 1, for eight nodes, it can be noticed that for Naut4k (page size of 4kB), the write method improved the speedup up to 9.78% and for Naut8k (page size of 8kB), the method improved the speedup of Nautilus up to 0.7%. The increasing of the speedups can be justified by observing the number of SIGSEGVs from Table 1 an order of magnitude lower for the technique versions compared to traditional versions. The number of page requests (gp rows) were reduced an order of magnitude when this method was applied.

Analyzing the page aggregation technique, for eight nodes, for NautV, this method decreased the speedup up to 0.1% and, for NautWD, this method improved the speedup up to 0.7%. The justification of the decrease of speedup is increasing of false sharing. The number of SIGSEGVs was reduced by 4.98% for NautV and by 44.32% for NautWD; in addition the number of page requests was reduced by 19.37% for NautV and by 42.64% for NautWD.

With both methods applied, write detection and page aggregation, the speedup of Nautilus was improved up to 10.21%, for eight nodes.

Comparing TreadMarks generically with NautV4k, both have similar behavior, some times TreadMarks outperforming NautV4k (six nodes for example) and some times NautV4k outperforming TreadMarks (for five and seven nodes). This alternate behavior is due to choice of data distribution and the needing of different storage of TreadMarks, which can cause in some cases the swapping of the operating system (Marino and Campos, 1999b). With the adoption of the mechanisms (write detection and aggregation), Nautilus outperformed TreadMarks in many cases.

![Figure 3: speedups of LU: N=17](image)

distribution (choice of the page owners) improves locality and gives a lower cold start up to distribute shared data are factors which also contribute to the better speedups of TreadMarks and Nautilus over JIAJIA. In addition, the elimination of signals minimizes the overheads of Nautilus over JIAJIA. In terms of number of SIGSEGVs, NautV has 26.29% lower than Naut4k, the later has 14.51% lower than the former; for the versions, Nautilus is of magnitude lower than JIAJIA. In terms of requests, NautV has lower number of them (60.38%) and for NautWD, it is one order of magnitude lower than JIAJIA.

8.3 Water

By looking at Figure 4, the speedups of W are seen. It can be noticed from this figure that the write detection technique improved Nautilus and the page aggregation technique did not improve Nautilus speedup.

From Table 1, for eight nodes, it can be seen that for NautV4k the write method improved the speedup up to 9.78% and for Naut8k the method improved the speedup of Nautilus up to 0.7%. The increasing of the speedups can be justified by observing the number of SIGSEGVs from Table 1 an order of magnitude lower for the technique versions compared to traditional versions. The number of page requests (gp rows) were reduced an order of magnitude when this method was applied.
order of magnitude lower, and NautWD8k has 21.52% lower SIGSEGVs. In the same way, the number of page requests (gp rows) were reduced: 11.28% for 4kB's page size and 24.77% for 8kB's page size.

Analyzing the page aggregation technique, for eight nodes, for NautV, this method decreased the speedup by 0.15% and, for NautWD versions, decreased the speedup by 1.30%, both speedup reductions justified because of the increase of the false sharing effect. With this technique, the number of SIGSEGVs was reduced an order of magnitude for NautV and by 18.57% for NautWD; in addition the number of page requests was reduced by 12.87% for NautV and by 26.11% for NautWD.

Considering the two techniques, the write detection and page aggregation, the method proportioned better speedup for Nautilus, up to 1.00%, for eight nodes. The problem with this application is its high synchronization, which is the dominant feature. Also, the false sharing effect increases with the increase of the page size, which contributes to decrease the speedup when the aggregation technique is applied.

Confronting TreadMarks generically with NautV4k, TreadMarks outperforms it up to 5.7%, due to its better data distribution and the semaphore implementation of Nautilus is until in developing. Confronting with JIAJIA, for eight nodes, TreadMarks outperforms it up to 6.48% and NautV4k outperforms it up to 1.87%. In terms of number of SIGSEGVs, JIAJIA has an order of magnitude lower than NautV4k, and 70-80% lower for the other versions, mainly due its better data distribution than Nautilus.

8.4 SOR

By looking at Figure 5, the speedups of SOR can be seen. It can be noticed from this figure that mainly the write detection technique improved Nautilus's speedup and the page aggregation technique practically did not improve Nautilus speedup.

In SOR, as the same way in LU, matrices are distributed across processors in a way that each processor writes to its home part of the matrices in the computing. (Becker and Merkey, 1997) Since the computation of an iteration is synchronized with barriers and barriers take more cycles than other operations, a barrier, and writing to home pages of a process on any SIGSEGV.

From Table 1, for eight nodes, it can be noticed for Naut4k, the write method improved the speedup up to 75.26% and for Naut8k, up to 57.98%. Decrease of the speedups can be justified by increasing the number of SIGSEGVs from Table 1, an order of magnitude lower for the NautWD versions compared to NautV versions. The number of page requests was reduced too.

Analyzing the page aggregation technique, for eight nodes, for NautV version, this method improved speedup up to 17.11% and, for NautWD version, 5.56%. In this technique, the number of SIGSEGV was reduced by 36.32% for NautV version and 50.59% for NautWD version; also the number of page requests was reduced by 38.98% for NautV and 55.41% for the NautWD.

With both methods applied, write detection and page aggregation, the speedup of Nautilus was improved up to 85.00%, for eight nodes.

Confronting TreadMarks generically with Nautilus, it outperforms TreadMarks up to 13.10%. The
time to distribute shared data. In addition, the avoidance of SIGIO signals and the multi-threading help to improve the speedup of SOR. In addition, with all techniques proposed and applied in this paper for Nautilus, it outperforms TreadMarks up to 109.22%.

For this benchmark, it seems that JIAJIA probably has an implementation problem for this benchmark, so it is not considered for speedup analysis. For number of SIGSEGVs, JIAJIA outperforms NautV4k and NautV8k by one order of magnitude and for the NautWD versions, they outperform JIAJIA by 21.24% and 61.09%. For the number of page requests, JIAJIA is 86.20% higher than NautV4k and has an order of magnitude higher page requests than other Nautilus's versions.

9 Conclusion

The contribution of this study is an evaluation of two techniques, write detection and page aggregation, on a DSM with Nautilus features. In order to have a fair and homogeneous comparison, two well known DSMs are used: TreadMarks and JIAJIA. In addition, these three DSMs are compared with respect to speedup, data and single-writer behavior, the write detection technique can improve its speedup. For SOR, the increasement of 85.00% for Nautilus was observed. In other applications with high synchronization occurrence, both techniques do not contribute to improvement in speedup. The number of SIGSEGVs and the number of request page faults have reduced by one order of magnitude in several cases, mainly when the detection technique was applied.

In future works, other benchmarks will be used in this comparison. Also, an improved version of JIAJIA will be evaluated and a complete tool with TreadMarks to measure the number of pages and SIGSEGVs.

References


Carter J. B., Khandekar D., Efficient Distributed Shared Memory: Where We Are and Where we Should Headed, Computer Science Laboratory, University of Utah, 1995.


Hu W., Shi W., Tang Z., Write detection in Home-based Software DSMs, to appear in Proceedings of Euro-Par'99, August 31-September 2, Tolouse, France.


Marino M. D.; Campos G. L.; A Preliminary DSM Speedup Comparison: JIAJIA x Nautilus, to be published at HPCS99.


