About the Sensitivity of the HLRC-DU Protocol on Diff and Page Sizes*  
Salvador Petit, Julio Sahuquillo, and Ana Pont  
Departamento de Informática de Sistemas y Computadores, Universidad Politécnica de Valencia (Spain)  
{spetit, jsahuqui, apont}@disca.upv.es

Abstract

Recent research on software distributed shared memory systems has focused on consistency protocols for improving performance. Home Lazy Release Consistency (HLRC) protocols have been widely adopted due to their performance advantages. Usually, these protocols invalidate pages through write notices. Variants of these protocols propose some criterion to update data of the corresponding pages instead of invalidating. In a previous paper, we proposed the HLRC-DU protocol, which is an improved version of the HLRC protocol. The HLRC-DU embeds update information in those write notices whose corresponding diff size is less than a given threshold, invalidating the remainder. The threshold trades off network bandwidth with update performance. In this paper, we study the HLRC-DU protocol’s sensitivity to page size and the threshold size selection. Our results show that while the page size slightly impacts performance, that our protocols are highly sensitive to the threshold value.

1. Introduction

Shared Virtual Memory (SVM) is an economic method to implement shared memory systems. The underlying operating software environment detects writes to shared addresses by using the virtual memory to take care of coherence maintenance. These systems have two main performance drawbacks: false sharing and communication latencies.

To alleviate these drawbacks, research has focused on multiple writer protocols. Multiple writer protocols allow several writers on a page at the same time, hiding most of the false sharing effects. In general, in order to detect the portion of a given page that was modified by a node, the protocol stores and sends page differences, also called diffs, instead of sending whole pages. Nevertheless, write notices and diffs introduce additional overhead.

In a previous work [1], we proposed the HLRC-DU memory consistency protocol based on the HLRC protocol proposed in [2]. The HLRC-DU protocol updates data through write notices when diffs are smaller than a given threshold. In this way, the write-notices update the data instead of invalidating the page, so we refer to them as write updates. The goal is to reduce the number of invalidations in order to avoid asynchronous requests to page homes.

The threshold size imposes a trade-off between bus utilization and the reduction of asynchronous communication. In this paper we study the sensitivity of the HLRC-DU to this parameter.

2. Motivation

Current SVM systems follow the Release Consistency (RC) [3] model that assumes that the parallel workload is data-race-free. These assumptions enable SVM systems to allow multiple writers in a page. To achieve this, they work as follows: initially all shared pages are labeled as read-only; as a consequence, when a process in a node (usually, a symmetric multiprocessor system) writes to a page, the corresponding processor issues an exception. Then, the operating system makes a copy (also called a twin) of the unmodified page and marks the page as read-write. When a page update is required, the operating system codifies in a diff the differences between the modified page and the twin page.

Because the parallel workload is assumed to be data-race-free, diffs from different processors on the same page cannot conflict; thus, the protocol can support multiple writers. Past research in SVM systems has incorporated these techniques to reduce both network traffic and false sharing, although they do introduce new overhead.

In home-based SVM protocols, each writer in the page creates one or more diffs, which are then resolved later in the page home. The cost to create and apply diffs grows linearly with the page size. In addition, each time a node encounters a page fault, it needs an up-to-date copy of the page, which is requested from the page home. Thus, the faulting node asynchronously interrupts the page home, which will release its workload computation to attend to the request. This may become a contention point, since there is only one home node per page.

To reduce asynchronous communication and contention at home nodes, the HLRC-DU protocol updates the diffs, by sending them together with write notices. We refer to this metadata (write notices plus associated diff) as write updates. In previous work [1], we used an empirical threshold of 128 words to alleviate network traffic. Those diffs larger than the threshold were not sent with write up-

* This work has been partially supported by research grant GV00-133-14
dates and the protocol invalidated the page.

We found that write updates improve performance unless the network becomes a bottleneck. Small write updates have a negligible impact on the network bandwidth. Because there are a lot of updates (as shown in section 6), the number of asynchronous page requests is substantially reduced, and consequently the speedup and the relative network utilization can be improved. This is not the case when using larger write notices because they are less frequent, thus there are fewer candidate asynchronous page requests to eliminate. In addition, larger request consume more network bandwidth.

In this work, we study the impact of the threshold and page sizes in the HLRC-DU protocol in order to find the value that maximizes performance.

3. Related work

The Hybrid LRC [4] was one of the first proposed LRC protocols that update data during acquisition. It differs from HLRC-DU in the update policy used (history based) and the LRC baseline protocol employed. LRC does not have the same problems as HLRC (e.g., the centralization of asynchronous communication). In addition, it is known that HLRC protocols offer better performance than LRC protocols. For the HLRC protocol, some proposals were introduced in [5][6] to reduce asynchronous communication by using specific hardware. Our proposal reduces asynchronous communication without additional hardware cost. In this sense, Stets et al. [7] propose an HLRC based protocol that updates shared data accessed by multiple consumers. The system updates the whole page from the home when there are multiple requests from clients to reduce asynchronous communication. We send diffs instead of the whole page, and since diffs are smaller, bus utilization is reduced. The additional diff application costs do not grow excessively because of their size.

4. Protocol descriptions

The baseline protocol is based on the HLRC protocol from [2]. In our implementation, each page has a home node to accumulate diffs generated by the writer nodes. When a node receives a write notice for a page, it marks its local copy as invalid and asks the home for an up-to-date version of the page, whenever any local process tries to access it. A page’s home is determined by means of a modulo function which uses the most significant bits of their page address. Each write notice contains the identification of the writer process, the timestamp of the write, and the page address. There is no garbage collection of globally known write notices. Write notices are only sent to a given process if it acquires a semaphore, or to all processes when they reach a barrier.

Once a process releases a semaphore or a barrier, and sends the write notices to the acquirer, it immediately sends to the homes those diffs produced by its previous writes in order to keep the page homes up-to-date. The acquirer will invalidate the pages corresponding to the addresses in the received write notices. Then, if the acquirer accesses the invalidated page, a page fault will occur. Consequently, the protocol asks the home for an updated page. If the home is not up-to-date, the diff will arrive immediately because it was sent just after the write notice. Figure 1 shows how modifications at the writer node (node A) on page P arrive at the home node of the page (node B) before the invalidated node (node C) can ask for an updated copy from the home node (node B).

![Figure 1. Baseline protocol](image1)

**Figure 1. Baseline protocol**

4.1. The HLRC-DU protocol

The HLRC-DU protocol detects diffs smaller than a threshold size and injects them as updates via write notices. Larger diffs are only sent to the home nodes like in Figure 1. If we did not use a threshold, network traffic would rise, potentially reducing performance; thus, we use this threshold to control the load injected in the network by write updates.

![Figure 2. HLRC-DU protocol](image2)

**Figure 2. HLRC-DU protocol**

Figure 2 shows how modifications of the writer node (node A) on page P arrive directly to node C when it enters the semaphore. In this way, the home node is not interrupted from workload computation. The page fault (on READ X) in node C is also saved. If during the acquisition of a semaphore, a process receives both an update and an invalidation for the same page, the page becomes invalid. In such a case, the protocol invalidates the node; as a consequence, the node will need an up-to-date copy from the home, which will contain all the modifications. Thus, there is a need to update the home (node B) as in base HLRC protocol.

As the threshold approaches zero, the HLRC-DU protocol performs more invalidation actions and its behavior is similar to the HLRC behavior. On the other hand, it behaves like a pure update protocol when there are no threshold restrictions.
5. Simulation environment

We use the LIvE execution driven simulator [8] to evaluate the described protocols. The compiler has been modified to trap the memory accesses from the parallel workloads, which enables LIvE to simulate them. The benchmarks were compiled with the gcc v2.6.3, using the O2 optimization flag.

To carry out our experiments we use a subset of the benchmarks from the SPLASH-2 benchmark suite [9]. We have selected FFT, Ocean, Radix, Barnes and LU, with a problem size of 64K points, 66x66 ocean, 256K integers 1024 radix, 512 particles, 256x256 points, respectively. Six different threshold values ranging from 16 to 512 words have been considered, as well as a seventh option with infinite threshold or no threshold restrictions. Two different page sizes (4 and 8KB) have been used to check the sensitivity of our model to this parameter.

The modeled architecture consists of a single cluster with 16 uniprocessor nodes connected through a detailed overclocked (1 Gb. per sec.) Ethernet network (contention is also modeled). Both protocols do not use the broadcast capability of Ethernet. The processor speed is 1 Ghz. The load in each node consists of the parallel application plus the operating system overhead introduced by the memory consistency model. A memory data access on a local page takes just one cycle. Caches are not simulated because of their limit on the simulation time, due to the long OS and network latencies. When a page fault or a remote request occurs, the operating system takes 100 µsec. to change the context. Before returning to the parallel application, a check is made to see if there is any request pending from a remote processor. If so, those requests have higher priority than the local requests, and each takes 10 µsec. before issuing a response. Both diff creation and its application take 4 cycles per word: 2 loads, 1 comparison, and 1 store. This overhead is not present when copying a single page because the model assumes that a DMA device performs this task.

6. Experimental Results

All our experiments were performed for 4KB and 8KB pages, however, due to space restrictions only results for 8KB page are shown. The results differ slightly because the SPLASH-2 benchmark suite uses a page size parameter to distribute data among nodes [9]. So, this affects the data distribution algorithm, producing similar results when varying the page size.

Asynchronous communication is one of the main drawbacks in current SVM systems [5][7], so the goal of our protocol is to reduce the number of asynchronous requests to the page homes caused by write notices. This is accomplished by receiving a write update instead of a write notice. In this manner, the page copy remains valid and the asynchronous request, and its associated network traffic, is saved.

Write notices are received when a process acquires a semaphore or a barrier. We instrument the baseline protocol to obtain the size of the diff produced by the write notice when it is received. Table 1 represents the distribution per processor of those sizes. As can be seen, most diffs are relatively small (about the 82% contain less than 512 words). This allows the protocol to embed those diffs in the write updates while slightly increasing the network traffic and diff application cost.

<table>
<thead>
<tr>
<th>Diff size range</th>
<th>Radix</th>
<th>Barnes</th>
<th>Ocean</th>
<th>LU</th>
<th>FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,16)</td>
<td>116</td>
<td>0</td>
<td>595</td>
<td>9284</td>
<td>40</td>
</tr>
<tr>
<td>[16,32]</td>
<td>1632</td>
<td>512</td>
<td>162</td>
<td>2409</td>
<td>1</td>
</tr>
<tr>
<td>[32,64)</td>
<td>64,128</td>
<td>950</td>
<td>151</td>
<td>869</td>
<td>0</td>
</tr>
<tr>
<td>[64,128)</td>
<td>128,256</td>
<td>2442</td>
<td>181</td>
<td>3579</td>
<td>0</td>
</tr>
<tr>
<td>[256,512]</td>
<td>256,512</td>
<td>47</td>
<td>0</td>
<td>4517</td>
<td>127</td>
</tr>
<tr>
<td>[512,1024)</td>
<td>1024,2048</td>
<td>42</td>
<td>0</td>
<td>791</td>
<td>259</td>
</tr>
</tbody>
</table>

Table 1. Distribution of diff sizes

The advantage of a write update with respect to a write notice is that it can avoid some page requests to the home. So, the numbers in Table 1 are an upper bound on the number of requests that can be avoided per processor. These results indicate that the benchmarks are able to avoid more requests in Barnes, Ocean and Radix.

<table>
<thead>
<tr>
<th>Bench.</th>
<th># requests</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>∞</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radix</td>
<td>4837</td>
<td>0.6%</td>
<td>0.6%</td>
<td>9.1%</td>
<td>10.7%</td>
<td>12.6%</td>
<td>12.6%</td>
<td>16.1%</td>
</tr>
<tr>
<td>Barnes</td>
<td>9853</td>
<td>59.0%</td>
<td>77.7%</td>
<td>80.0%</td>
<td>83.7%</td>
<td>88.8%</td>
<td>91.8%</td>
<td>91.8%</td>
</tr>
<tr>
<td>Ocean</td>
<td>55318</td>
<td>10.3%</td>
<td>11.3%</td>
<td>26.9%</td>
<td>31.0%</td>
<td>61.1%</td>
<td>61.7%</td>
<td>97.7%</td>
</tr>
<tr>
<td>LU</td>
<td>747</td>
<td>1.6%</td>
<td>1.6%</td>
<td>1.9%</td>
<td>2.0%</td>
<td>2.0%</td>
<td>6.0%</td>
<td>36.6%</td>
</tr>
<tr>
<td>FFT</td>
<td>22790</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>26.0%</td>
</tr>
</tbody>
</table>

Table 2. Percentage of saved home page requests

Table 2 shows the percentage of home page requests saved by write updates considering different values for the threshold in HLRC-DU. Values from the last column of Table 2 are lower than 100% due to the initial home page requests. Results for our benchmarks fall into categories: 1) benchmarks that are highly sensitive to the threshold value (e.g., Barnes), show savings from 59% (with threshold 16) to 91.8% (with threshold 512), 2) benchmarks that are slightly less sensitive (e.g., Ocean and Radix), which is consistent with the results shown in Table 1 as explained above, and 3) benchmarks that insensitive to the threshold value (saving less than a 4% of the home page requests) LU and FFT due to low values shown in Table 1. Consequently, the performance of these benchmarks in the HLRC-DU protocol will not differ much from those found for the baseline protocol when using small thresholds, i.e., 256 words.

Comparing Tables 1 and 2, it can be seen that larger
write updates tend to save more home page requests across the benchmarks. This is because larger diffs present less false sharing, i.e., they leave fewer places in the page that could be invalidated. Although larger diffs hugely reduce the number of requests, we must trade off this reduction in network traffic, to improve the system performance (as discussed below).

Figure 3 shows the network utilization relative to the baseline HLRC model for each benchmark used, varying the threshold size. Relative utilization considers the measurement time for the HLRC baseline model; thus, results flow the same trend as the traffic. It can be seen that our protocol reduces network traffic in some applications for any threshold value (e.g., Barnes and Ocean), because in these benchmarks the write updates save more traffic than they introduce. In some cases, under a large threshold value, the injection of write updates considerably increases the network traffic (e.g., Radix, LU and FFT). It is remarkable that no benchmark increases the network traffic when using a small threshold value.

![Figure 3 – Relative network utilization](image)

While it is unclear from the Figure 3 when the network reaches saturation, this can be observed clearly in Figure 4. This graph summarizes the speedup results for HLRC-DU over the baseline protocol, while varying the threshold size value. When the speedup increases, it is obvious that there is no contention point (e.g. Ocean, Barnes, and Radix). However, in Radix, the network becomes saturated when using a threshold of 256 words. As discussed above, in FFT and LU, our protocol behaves close to the baseline protocol.

![Figure 4 – Speedup](image)

In general, our experiments show that the threshold values that achieve the best speedup in the range from 64 to 256 words (depending on the workload used), while not decreasing the performance of any benchmark when using a threshold value lower than 512 words.

7. Conclusions

The HLRC-DU consistency protocol updates those diffs smaller than a given threshold using write notices. Asynchronous communication and network utilization are strongly dependent on the threshold value used. In this paper, we have focused on the estimation of the optimal threshold that maximizes system performance.

Our results show that the HLRC-DU protocol can save about the 90% of request petitions to the page homes in some of the benchmarks, when working under their optimal threshold. This reduction has a doubly positive impact on performance. For small threshold values, we reduce the network congestion and increase the speedup, which in some cases (8KB pages) reaches 80% versus the HLRC base protocol. The performance obtained using an 8KB page size is similar than that obtained with 4KB pages, due to the data distribution algorithm used in the SPLASH-2 benchmarks implementation. Although the optimal threshold value is workload dependent, it ranges from 64 words to 256 words, a relatively small value that avoids the network becoming a bottleneck.

References