Available online at www.sciencedirect.com
SCIENCE DIRECT

J. Parallel Distrib. Comput. 64 (2004) 887–907

Journal of Parallel and Distributed Computing

http://www.elsevier.com/locate/jpdc

Performance analysis of methods that overcome false sharing effects in software DSMs[☆]

Manjunath Kudlur[a,*,1,2] and R. Govindarajan[b,2]

[a] *Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA*
[b] *Department of Computer Science and Automation, Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore 560 012, India*

Received 29 May 2001; revised 17 January 2004

Abstract

Page-based software DSMs experience high degrees of false sharing especially in irregular applications with fine grain sharing granularity. The overheads due to false sharing is considered to be a dominant factor limiting the performance of software DSMs. Several approaches have been proposed in the literature to reduce/eliminate false sharing. In this paper, we evaluate two of these approaches, viz., the Multiple Writer approach and the emulated fine grain sharing (EmFiGS) approach. Our evaluation strategy is two pronged. First, we use an implementation-independent analysis that uses overhead counts to compare the different approaches. Our analysis show that the benefits gained by eliminating false sharing are far outweighed by the performance penalty incurred due to the reduced exploitation of spatial locality in the EmFiGS approach. As a consequence, any implementation of the EmFiGS approach is likely to perform significantly worse than the Multiple Writer approach. Second, we use experimental evaluation to validate and complement our analysis. The experimental results match well with our analysis. Also the execution times of the application follow the same trend as in our analysis, reinforcing our conclusions. More specifically, the performance of the EmFiGS approach is significantly worse, by a factor of 1.5 to as much as 90 times, compared to the Multiple Writer approach. In many cases, the EmFiGS approach performs worse than even a single writer lazy release protocol which experiences very high overheads due to false sharing.

The performance of the EmFiGS approach remains worse than the Multiple Writer approach even after incorporating Tapeworm—a record and replay technique that fetches pages ahead of demand in an aggregated fashion—to alleviate the spatial locality effect. We next present the effect of asynchronous message handling on the performance of different methods. Finally, we investigate the inter-play between spatial locality exploitation and false sharing elimination with varying sharing granularities in the EmFiGS approach and report the tradeoffs.
© 2004 Published by Elsevier Inc.

Keywords: False sharing; Memory consistency protocols; Performance evaluation; Software DSM

1. Introduction

Software Distributed Shared Systems [4,20,21], that rely on the virtual memory mechanism provided by the Operating System for detecting accesses to shared data, support sharing granularity of page size. The large page size results is excessive *false sharing* overheads, especially in fine grain irregular applications [28]. Different methods have been proposed in literature to reduce the effects of false sharing in page-based software DSMs. Two basic approaches followed in these methods are (i) allowing concurrent write accesses to a page and (ii)

[☆] This work was partially funded by IBM's Shared University Research programme. A shorter version of this paper has appeared in the Proceedings of the 8th International Conference on High Performance Computing (HiPC-2001), Hyderabad, Dec. 2001 (http://www.hipc.org).

*Corresponding author.
E-mail addresses: kvman@umich.edu (M. Kudlur), govind@csa.iisc.ernet.in (R. Govindarajan).

[1] This work was done when M. Kudlur was at the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, 560 012, India.

[2] Also for correspondence. Tel.: +91-80-394-2794; fax: +91-80-360-2648.

0743-7315/$ - see front matter © 2004 Published by Elsevier Inc.
doi:10.1016/j.jpdc.2004.02.003

providing fine grain granularity through emulation without additional architectural support. We refer to these approaches as the Multiple Writer approach and the Emulated Fine Grain Sharing (EmFiGS) approach, respectively. We call the latter *emulation* because it provides fine grain sharing over a coarse grain sharing system and as a consequence incurs higher cost even for a fine grain coherence miss.

Lazy Multiple Writer Protocol (LMW) [16,19] implemented in most state-of-the-art software DSMs [2,17,27] is a typical example of the Multiple Writer approach. It allows concurrent writes to a page by providing mechanisms to merge the modifications at synchronization points. LMW is considered heavy-weight because of the Twin/Diff creation overheads incurred to maintain and update the modifications. Writer Owns Protocol [8] improves upon LMW by performing run time re-mapping of subpages such that all subpages in a page are written by the same process. The sharing is still performed at the page level, but by remapping parts of the page, false sharing is eliminated. Millipage/Multiview [11,12] follows the EmFiGS approach. It decreases the sharing granularity to less than a page by allowing smaller size pages, called *millipages*. However, to avoid wastage of physical memory, multiple millipages are mapped to a single physical page (of larger size). Millipage is implemented with single-writer protocol for efficient operation. Compile-time methods [9,14] have also been proposed, following the EmFiGS approach, wherein data structures susceptible to false sharing are identified at compile-time and various transformations are applied to them to place them in different pages, thus eliminating false sharing. While the Multiple Writer approach *tolerates* false sharing, the EmFiGS approach *eliminates* false sharing.

While it is true that these approaches are successful in reducing false sharing, they do not sufficiently address the following questions: (i) what are the additional costs incurred, if any, in reducing false sharing? and (ii) is the reduction in false sharing overheads significantly higher than the additional costs, thereby leading to overall performance improvement of the application? Also, while there have been performance evaluation of individual implementations of these approaches, there has been no complete comparative analysis of these approaches which is required to understand the interplay of overheads. This motivates our performance analysis work on methods that overcome false sharing in software DSMs.

Our performance evaluation strategy is two pronged. First, we present an implementation independent analysis to obtain the counts of various overheads incurred under different protocols. In our analysis, we have taken into account all the overheads incurred in most page-based DSMs, including the round trip message overhead, synchronization overhead, and the page fault kernel overhead incurred in entering the OS kernel to transfer control to SEGV handler. The overhead counts provide a basis for comparison of the different methods. Second, we augment this comparison with execution time results of the benchmarks under an actual implementation of the methods. The experimental evaluation complements the manual analysis by reporting the contributions of different overheads to execution time. It also helps to validate the manual analysis and to attribute costs to the overhead components. Last, the time incurred by certain overheads such as synchronization overheads depend on the actual runtime conditions and the application behavior. In these cases, the performance results obtained from our experimental evaluation presents a better picture. The following discussion presents a more specific account of the performance study.

We have considered those protocols for study that sufficiently represent the dominant approaches to overcome false sharing, viz., the Multiple Writer approach and the EmFiGS approach. In addition, we have considered the Lazy Single Writer Protocol (LSW) as the base case, since it can incur high false sharing overheads. Our application suite consists of three SPLASH2 benchmarks [29], viz., Barnes, Water-Spatial and Radix, all exhibiting high false sharing behavior. First, our manual analysis reveals that, despite eliminating false sharing completely, the EmFiGS approach still incurs significantly higher number of page faults and messages compared to the Multiple Writer approach. This is because, with a smaller sharing granularity in the EmFiGS approach, the amount of spatial locality is reduced, more true sharing faults occur. Our analysis indicates that the overheads incurred by the EmFiGS approach are significantly higher by a factor of 4 or more as compared to LMW. As our analysis is independent of any specific implementation and captures the overheads that are intrinsic to the protocol, we conclude that any implementation of the EmFiGS approach is likely to incur more true sharing overheads than the savings in false sharing.

Second, we use experimental evaluation to validate the manual analysis. Our experimental evaluation also augments the analysis by attributing costs to the overheads counts. Our experimental platform is IBM's Scalable Parallel (SP) architecture, running open source CVM [17], a software DSM system. The experimental results on overheads counts match closely with those obtained from manual analysis and reinforces our conclusions. The decreased exploitation of spatial locality manifesting as increased page faults and messages in the EmFiGS approach, results in a degradation of 1.5 to as much as 90 times compared to the Multiple Writer approach. Further, contrary to the popular belief about the heavy-weightedness of Multiple Writer protocols, the overheads of Twin/Diff