6. Comparative Evaluation of Fine- and Coarse-Grain Approaches for Software Distributed Shared Memory

Sandhya Dwarkadas, Kourosh Gharachorloo, Leonidas Kontothanassis,
Dan Scales, Michael Scott, and Robert Stets
Univ. of Rochester, DEC WRL and DEC CRL

Hardware cache-coherent multiprocessors offer good performance and provide a simple shared-memory model of computing for application programmers. Multicomputers, or collections of networked machines, can potentially provide more cost-effective performance, but require additional programmer effort to write message-passing programs. Software distributed shared memory (S-DSM) attempts to ease the burden of programming distributed machines by presenting the illusion of shared memory on top of distributed hardware using a software run-time layer between the application and the hardware.

Early S-DSM systems were primarily based on virtual memory. They used pages as the unit of coherence, and used page faults to trigger copy and invalidation operations. This approach can yield excellent performance for ``well-behaved'' programs. Unfortunately, it tends to be very slow for programs with any fine-grain sharing of data within a single page.

Recent systems employ relaxed consistency models that allow a page to be written by multiple processes concurrently, and limit the impact of false sharing to the points at which programs synchronize. Page-based systems may still experience overhead due to synchronization, sharing at a fine granularity, and metadata maintenance. Furthermore, their use of relaxed consistency models introduces a departure from the standard SMP shared-memory programming model that can limit portability for certain applications developed on hardware DSM.

Alternatively, some S-DSM systems rely on binary instrumentation of applications to catch access faults at a fine grain. Such systems provide the highest degree of shared memory transparency since they can efficiently run programs developed for hardware consistency models. However, the overhead of the added code can sometimes limit performance.

This paper attempts to identify the fundamental tradeoffs between these two S-DSM approaches; coarse-grain, VM-based vs.\ fine-grain, instrumentation-based. Our study compares two relatively mature but very different S-DSM systems running on identical hardware that are representative of the fine- and coarse-grain approaches: Shasta and Cashmere. Both systems run on clusters of Alpha SMPs connected by DEC's Memory Channel (MC) remote-write network.

We compare the performance of Shasta and Cashmere on thirteen applications running on a 16-processor, 4-node AlphaServer SMP cluster connected by the Memory Channel. Eight of the applications are from the SPLASH-2 application suite and have been tuned for hardware multiprocessors, while five are existing applications that have been shown in the past to perform well on software (and hardware) shared-memory multiprocessors.

We have chosen our experimental environment as representative of the way one might build an inexpensive but high-end cluster. Four-way SMP nodes provide a sweet-spot in the price/performance spectrum, while Memory Channel is a very low-latency, high-bandwidth commercial network that connects to the industry-standard PCI interface. Our environment is quite different from that used in a recent study~\cite{zhou-iftode-wood-ppopp-1997} that also examines performance tradeoffs between fine- and coarse-grain software coherence. This previous study uses custom hardware to provide fine-grain access checks with no instrumentation overhead, and uses a lower performance network (almost an order of magnitude performance difference in latency and a factor

of two in bandwidth). In addition, their study uses a cluster of uniprocessors, whereas we use a cluster of SMPs and protocols designed to take advantage of SMP nodes. Our end results are similar to the ones reported by the previous study, with a few surprising exceptions that are affected by the two-level nature of our system.

Our study clearly illustrates the tradeoffs between fine-grain and VM-based S-DSM on an aggressive hardware environment. A fine-grain system such as Shasta has more robust (and often better) performance for programs developed on a hardware DSM (H-DSM). It supports H-DSM memory models, and is better able to tolerate fine-grain synchronization. Cashmere has a performance edge for applications with coarse-grain data access and synchronization. With program modifications that take coherence granularity into account, the performance gap between the two systems can be bridged. Remaining performance differences are dependent on program structure: a high degree of false sharing at a granularity larger than a cache line favors Shasta since the smaller coherence block brings in less useless data; large amounts of mostly private data favors Cashmere, since there is no virtual memory overhead unless there is active sharing. Fine-grain false sharing also favors Cashmere due to its ability to delay and aggregate protocol operations.

One surprising result of our study has been the good performance of page-based S-DSM on certain applications known to have a high degree of page-level write-write false sharing. The clustering inherent with SMP nodes eliminates the software overhead from false sharing within nodes since coherence within nodes is managed by hardware. Moreover, for applications with regular (e.g.\ cyclic) data layout, cross-node data boundaries can end up aligned, eliminating inter-node false sharing as well.