Chapter Seven

Summary

For many applications that target distributed memory multicomputers for their scalability and high-performance, it is desirable to provide transparent Distributed Shared Memory (DSM). For such applications, the hardware and system software must maintain coherency among the local memories. To assure sustained progress of large applications through the failures of nodes and their associated local memory, the system must also incorporate a mechanism that ensures the continued availability of the entire shared address space in spite of these failures.

Most existing techniques for providing DSM on multicomputers perform the mapping, data transfers, and ownership management at a single granularity of blocks (or pages). If large blocks are used, these techniques suffer from long latencies for remote requests, they require higher network bandwidth than is inherently needed by the application, and they lead to unnecessary transfers due to false sharing. Smaller blocks lead to a decrease in miss ratio [Dubo88], a reduction in remote access latencies, and a reduction in the required network bandwidth. However, for small blocks, management of DSM at the block level results in unacceptably high storage overhead for the various mapping tables.

We have proposed the use of multiple levels of mapping and coherency management, a new organization for the copy directory, and inverted tables as a solution to this problem. The proposed scheme dynamically distinguishes between shared and private pages in order to manage each class efficiently. Ownership is maintained at both the block and page level for shared pages while ownership is maintained only at the page level for private pages. In order to limit the storage overhead for coherency management, tables that keep track of block copies are not allowed to grow beyond a static bound. In a realistic example of a large multicomputer, less than 4% of the total system storage will be dedicated to tables needed for coherency management.

The design of correct distributed directory-based coherence protocols is one of the main tasks in DSM implementation. This design is significantly more complicated than the design of snoopy coherence protocols due to the absence of a serializing medium such as a shared bus and the non-atomicity of accesses. These factors introduce problems associated with deadlocks, race conditions, and the handling replacements.

We have designed a DSM protocol for hierarchically managed DSM as well as DSM managed at a single block level. We employ standard mechanisms, similar to those used in [Leno93], to handle deadlocks and replacement hints. Deadlocks are avoided by requiring that invalidate messages and reply messages are always accepted and processed (‘‘sunk’’) at their destinations. Update messages due to replacements are handled with additional states in the coherence protocol. We developed the use of *bilateral exchanges* as opposed to forwarding as a solution to the problem of races. Specifically, in a pairwise-FIFO network, the race problem is avoided by restricting all protocol exchanges to be bilateral. Since the use of bilateral exchanges involves a slight increase in message count, we have developed a hybrid protocol, where read accesses use forwarding and write accesses use bilateral exchanges.

Using the Proteus [Brew92b] execution-driven simulation environment, we have implemented detailed simulations of our DSM protocol for hierarchically managed DSM as well as DSM managed at a single block level. We have evaluated the performance of the hierarchical management scheme using execution-driven simualtions and trace-driven simulations. In general, the results indicate that significant performance improvement over conventional page-level schemes is possible with our two-level scheme, without incurring the storage overhead that makes uniform block-level schemes impractical in large systems. The results verify that only a small fraction of the shared address space is actually shared at any time. This validates the benefits of a scheme, such as our hierarchical scheme, where coherency tables (e.g., our shared pages table) can be kept small since they maintain information only for dynamically shared data. The results also show that our new copy directory organization is more space-efficient than the conventional organization used in most DSM systems.

Many error recovery schemes for multiprocessors and multicomputers are based on checkpointing a process whenever it communicates with another process or on the use of message logging techniques. For scalable high performance systems, used for communication-intensive applications, all of these schemes incur unacceptably high overhead.

We have developed an error recovery scheme for DSM multicomputers based on coordinated checkpointing and rollback. Our scheme is an adaptation of coordinated checkpointing and rollback for message-passing systems. This adaptation efficiently solves several potential difficulties with DSM systems which are due to the lack of explicit messages, the presence of a shared replicated address space, and the distributed coherency directory. We have developed mechanisms to identify and track read-write communication in the DSM system and to checkpoint and recover the shared replicated address space. Our scheme does not checkpoint and rollback the coherency directory. Instead, at recovery, the distributed coherency directory is ‘‘repaired’’ to restore it to a correct state. Trace-driven simulations have demonstrated the significant performance advantages of our recovery scheme. By allowing lower checkpointing frequency, leading to lower overall checkpointing traffic, our approach can result in dramatically lower overhead than alternate schemes, leading to minimal impact on the normal operation of the system.