Analysis of sharing overhead in Shared Memory Multiprocessors

Pierfrancesco Foglia, Roberto Giorgi and Cosimo Antonio Prete
Dipartimento di Ingegneria dell'Informazione
Facoltà di Ingegneria - Università di Pisa, Italy

A cache memory contributes in both hiding memory latency and reducing the traffic on the processor interconnection network of shared memory multi-processors but it causes the coherence problem. A *coherence protocol* [Tomasevic93] is required in order to guarantee the coherence of the cached copies. An adequate choice of the coherence protocol is critical for performance. In fact, when the number of nodes exceeds a critical value, the processor interconnection network reaches a saturation condition, due to both *cache misses* and *coherence operations*.

Three main classes of coherence protocols are write-update (WU), write-invalidate (WI), and hybrid. A WU protocol updates the remote copies on each write involving on a shared copy. Whereas, a WI protocol invalidates previously remote copies in order to avoid updating them. A hybrid protocol uses both WU and WI strategies to combine the best aspect of each one. The frequency and the pattern of accesses to shared copies influence coherence overhead. Since access pattern to shared data varies from application to application; neither WI nor WU is the better strategy for maintaining cache coherence in all cases. Results [Veenstra94] show that, although the hybrid protocol [Cox93, Prete90, Prete95b, Stenstrom93] does not offer any significant advantage over the best choice of pure protocols for a particular application, it may offer optimal performance over a wider range of applications than any single pure protocol.

An optimal selection for the coherence protocol can be made by considering the traffic induced by the two approaches in case of different sharing. The coherence overhead induced by a WU protocol is due to all the operations needed to update the remote copies. Whereas, a WI protocol invalidates remote copies and processors generate a miss on the access to the invalidated copy. Invalidation and block fetching (due to invalidation misses) contribute to the coherence overhead of WI protocols. By considering the cost for invalidation, block fetching and updating, the potential penalty for choosing WU incorrectly is much higher than the potential penalty for choosing WI incorrectly.

Three different sources of sharing may be observed: i) *active sharing*, which occurs when the same cached data item is referenced by processes concurrently running on different processors; ii) *false sharing* [Torrellas94, Tomasevic96], which occurs when several processors reference separate data items belonging to the same memory block; iii) *passive* [Prete90, Prete95, Prete97a] or *process-migration* [Agarwal88] *sharing*, which occurs when a memory block, though belonging to a private area of a process, is replicated in more than one cache as a consequence of owner process migration.

Active sharing can be subdivided in fine-grain and sequential sharing [Eggers89]. Fine-grain sharing denotes high contention for shared data; sequential sharing is characterized by long sequences of writes to the same memory item performed by the same processor. A WI protocol is adequate in the case of sequential sharing, whilst, in general, a WU protocol performs better than WI for programs characterized by fine-grain sharing. *Migratory sharing* is an example of sequential sharing that occurs when a block is read and written by several processors, but in long intervals the memory block is exclusively used by one processor at a time [Gupta92, Stenstrom93]. For example, n the use of data structures belonging to critical sections, processors read and modify data structures one at time. In the case of WI protocols, processors that access shared data in this way cause a cache miss followed by an invalidation request being sent to the cache belongs to the processor that most recently exited the critical section. It is possible to merge the invalidation request with the previous read-miss request and thus eliminate all explicit invalidation actions [Stenstrom93].

Passive sharing is particularly emphasized when multiprocessors are used not only to speed-up parallel applications, but also to minimize the execution time of

1060-3425/98 $10.00 (c) 1998 IEEE

general-purpose workloads consisting of both parallel and sequential applications, such as, for example, in case of network server or general-purpose high performance systems. These workloads generally produce more processes than the number of processors in the machine and, therefore, there are two factors that increase the number of misses and useless coherence overhead. First, several processes are forced to time-share the same cache, resulting in one process destroying the cache state previously built up by another. Consequently, when this process runs again, it generates a stream of misses as it rebuilds its cache state. Second, since an idle processor simply selects the highest priority runnable process, a given process often moves from one CPU to another. This frequent process migration causes passive sharing, since private data blocks of a process can become resident in multiple caches and generate useless coherence-related overhead [Prete97a].

To reduce the number of misses and the useless coherence-related overhead in these workloads, processes should reuse their cached state more. One way to encourage this is to schedule each process based on its affinity to individual cache, that is, based on the amount of state that the process has accumulated in an individual cache. The cache affinity scheduling [Squillante89] cannot be applied to all workload conditions.

The aim of this work is to analyze [Prete95a, Prete97b] the overhead caused by general-purpose workloads in managing shared cached copies and investigate the relations among workload features, process scheduling policies, and some architecture features, such as, processor and cache organization, and coherence protocol.

References

[Agarwal88] **A. Agarwal** and **A. Gupta**, "Memory reference characteristics of multiprocessor applications under Mach", *Proc. ACM Sigmetrics*, Santa Fe, NM, pp. 215-225, May 1988.

[Cox93] **A.L. Cox** and **R.J. Fowler**, "Adaptive cache coherency for detecting migratory shared data", *Proc. 20th Int. Symp. Comput. Arch.*, pp. 98-108, 1993.

[Eggers89] **S.J. Eggers**, "Simulation analysis of data sharing in shared memory multiprocessors", Ph.D. dissertation, Univ. of California, Berkeley, April 1989.

[Gupta92] **A. Gupta** and **W. D. Weber**, "Cache Invalidation Patterns in Shared-Memory Multiprocessors", *IEEE Transactions on Computers*, vol. C-41, n. 7, pp.794-810, July 1992.

[Prete90] **C.A. Prete**, "A new solution of coherence protocol for tightly coupled multiprocessor systems", *Microprocessing and Microprogramming*, vol. 30, n. 1-5, Amsterdam, pp.207-214, 1990.

[Prete95a] **C.A. Prete**, **G. Prina**, and **L. Ricciardi**, "A trace-driven simulator for performance evaluation of cache-based multiprocessor systems", *IEEE Trans. Parall. Distr. Syst.*, vol.6, n. 9, pp. 915-929, Sept. 1995.

[Prete95b] **C.A. Prete**, **G. Prina**, and **L. Ricciardi**, "A selective invalidation strategy for cache coherence", *IEICE Trans. on Information and Systems*, vol. E78-D, n. 10, pp. 1316-1320, Oct. 1995.

[Prete95c] **C.A. Prete**, **G. Prina**, and **L. Ricciardi**, "Reducing coherence-related overhead in multiprocessor systems", *Proc. 3rd Euromicro Workshop on Par. and Distr. Processing*, Sanremo, IEEE Computer Society Press, pp.444-451, Jan. 1995.

[Prete97a]. **A. Prete**, **G. Prina**, **R. Giorgi** and **L. Ricciardi**, "Some Considerations About Passive Sharing in Shared-Memory Multiprocessors", *IEEE TCCA Newsletter*, pp.34-40, March 1997.

[Prete97b] **R. Giorgi**, **C.A. Prete**, **L. Ricciardi**, **G. Prina**, "Trace Factory: a Workload Generation Environment for Trace-Driven Simulation of Shared-Bus Multiprocessors", to appear on *IEEE Concurrency*.

[Squillante93] **M.S. Squillante** and **D.E. Lazowska**, "Using processor-cache affinity information in shared-memory multiprocessor scheduling", *IEEE Trans. Parall. Distr. Syst.*, vol. 4, n. 2, pp. 131-143, Feb. 1993.

[Stenstrom93] **P. Stenström**, **M. Brorsson**, and **L. Sandberg**, "An adaptive cache coherence protocol optimized for migratory sharing", *Proc. 20th Int. Symp. Comput. Arch.*, pp. 109-118, 1993.

[Tomasevic93] **M. Tomasevic**, **V. Milutinovic**, eds., *The cache coherence problem in shared-memory multiprocessors: hardware solutions*, IEEE Computer Society Press, Los Alamitos, CA, April 1993.

[Tomasevic96] **M. Tomasevic**, **V. Milutinovic**, "The word-invalidate cache coherence protocol", *Microprocessors and Microsystems*, vol. 20, pp. 3-16, 1996.

[Torrellas94] **J. Torrellas**, **M.S. Lam**, and **J.L. Hennessy**, "False sharing and spatial locality in multiprocessor caches", *IEEE Trans. Comput.*, vol. 43, n. 6, pp. 651-663, June 1994.

[Veenstra94] **Jack E. Veenstra** and **Robert J. Fowler**, "The Prospects for On-Line Hybrid Coherency Protocols on Bus-Based Multiprocessors", University of Rochester, Computer Science Department", Technical Report TR 490, March 1994.