Cache Coherence Protocols in Shared-Memory Multiprocessors

Xiuzhen Lian, Xiaoxi Ning, Mingren Xie, Farong Yu

Key Laboratory of Evidence of Science and Technology Research and Application，Gansu Province (Gansu Institute of Political Science and Law), Lanzhou,730070, China

Corresponding author: FarongYu. Tel: 18693932139, Email: tim9898@163.com

Keywords: Cache coherence protocols, Shared-memory, Multiprocessor

Abstract. This paper is a review of the recent research about the design of cache coherence protocols in shared-memory multiprocessors. Two important aspects of shared memory systems are memory consistency and cache coherence. Two major available protocols for cache coherence problems are snoopy coherence and directory based coherence. The snoopy cache protocol is simple and easy to implement, but relies on a low-latency, shared interconnection among the processors and the memory modules. The directory-based multiprocessors communicate with a common directory whenever the processor's action may cause an inconsistency between its cache and the other caches or memory. No broadcast is necessary in this case and therefore the network medium may be of almost any kind. However, the overhead of directory maintenance and look-up time plus the high-latency of communication networks make the directory scheme unattractive. To prevent the directory from becoming the bottleneck, directory entries can be distributed along with the memory, so that different directory accesses can go to different locations.

Introduction

In principle, a coherent cache system allows multiple copies of the same memory location to exist in the system, but they are always consistent by having processors broadcast the values of updates or invalidation. However, cache misses and memory traffic due to shared data blocks limit the performance of parallel computing in multiprocessor computers or systems. The cache coherence problem arises when parallel and distributed computing systems make local replicas of shared data for reasons of scalability and performance. The biggest problem is that parallel programs tend to use the same data at almost the same time. In both distributed shared memory systems and distributed file systems, a coherence protocol maintains agreement among the replicated copies when the underlying data are modified by programs running on the system. Therefore it is important to let other processors and primary memory know about changes as soon as possible to avoid "dirty data". Two major available protocols for cache coherence problems are snoopy coherence and directory based coherence. Snoopy schemes efficiently solve cache coherence problem on single-bus-connected multiprocessors by letting each tapped processor snoop on the operations of other processor. Directory schemes on the other hand, can be applied on any interconnection network and are relatively more scalable[1]. To solve the consistency problem of shared data blocks, it is necessary to implement the cache memory with appropriate cache coherence protocols and to minimize effective memory access time. The virtual memory architecture, by imposing a potentially many-to-one mapping of virtual to physical addresses, places constraints on how the performance of cache coherence can be achieved [2]. But the difficult is that, in different domains, the cache address is implemented as different ways. This paper is a review of the recent researches about the design of cache coherence protocols in the shared-memory multiprocessors.

Evolution of cache coherence protocols

Although very large mainframes were built with multiple processors in the 1970s, multiprocessors did not become highly successful until the 1980s. Directory-based techniques for

cache coherence were actually known before snoopy cache techniques[3]. The first cache coherence protocol which used directories was described by Tang [4]and implemented in IBM 3081.

In the late 1980s and early 1990s, the development of directory-based cache coherence protocols allowed the creation of cache-coherent shared-memory multiprocessors. The first bus-based multiprocessor with snooping caches was the Synapse N+1 described by Frank[5]. Agarwal[6] first described the idea of distributing directories with the memories to obtain a scalable implementation of cache coherence, which was served as the basis of the Standford DASH multiprocessor[7]. The commercial scalable coherent shared memory was implemented in Kendall Square Research KSR-1 in 1992[8]. Archibald and Baer[9] provided a good survey and analysis of the cache coherence protocols on multiprocessors.

Relationship between consistency models and coherence protocols

Two important aspects of shared memory systems are memory consistency and cache coherence. When multiple processors with on-chip caches are placed on a common bus sharing a common memory, then it's necessary to ensure that the caches are kept in a coherent or consistent state. A cache-coherence protocol consists of the set of possible states in the local caches, the states in the shared memory, and the state transitions caused by the messages transported through the interconnection network to keep memory coherent. Private caches in conjunction with directory-based cache coherence protocols are key to tolerate the memory-latencies in large-scale, shared-memory multiprocessors.

The major impact of cache memory models on cache protocol design is to specify when cache coherence should be enforced, when the effect of stores should be propagated, and when they should be taken into account at the receiving end. A useful way of viewing the relationship between memory consistency models and cache coherence protocols is that the coherence mechanism ensures that all of the caches see all of the writes to a specific block in the same logical order. The consistency model defines for the programmer the order of writes to different blocks as perceived by each of the processors and the rules about when the stores must propagate and the cache consistency protocol must propagate them in time for a subsequent read by a remote processor. These rules are essentially sufficient conditions for supporting the general coherence properties that are requirements for useful memory models such as the sequential Consistency, weak ordering, and release consistency in a cache-coherent shared-memory system. For example, the protocols developed in weaker memory models, the only relaxation of the preceding rules is to guarantee that a store operation finishes before the next synchronization executed by the local processor, that is, enforcing general coherence at synchronization points.

Cache coherence protocols

The essence of cache coherence problem is that multiple copies of the same data can exist in different caches simultaneously, and if processors are allowed to freely update their own copies, an inconsistent view of memory can result. Although different protocols for cache coherence have been proposed, the common schemes for cache coherence are snoopy-based and directory-based protocols.

Snoopy-based cache coherence protocols

In the bus-based shared-memory multiprocessor architecture, the bus is as a broadcast medium to main coherency, which is called snooping cache coherence protocol[10]. With respect to a single memory location, one and only one write operation is allowed to propagate at a time. In a coherent multiprocessor, the caches can provide both migration and replication of shared data items. The main difference between a snoopy cache and an uniprocessor cache is the cache controller, which stored the information of the cache, and the bus controller, which is a finite-state machine that implements the cache coherence protocol according to the state transition[11]. The snoopy cache protocol relies on a low-latency, shared interconnection among the processors and the memory

modules, such as a common bus that allows each processor to monitor all transactions to the shared memory. This protocol is simple and easy to implement. Many commercial, bus-based multiprocessors have used this protocol such as Sequent Computer Systems' Symmetry multiprocessor and Alliant Computer systems' Alliant FX which use write-invalidate policies to maintain cache consistency[10].

With the increase of multiprocessors, the shared bus can be a severe performance bottleneck. Also, buses have not enough bandwidth to support a large number of processors, resulting in that the bus cycle time is restricted by the signal transmission times in multiple environments and must be long enough to allow the bus to ring out. Even we can use additional buses to increase the bandwidth between the processors and the shared memory, their performance ultimately will be limited by the bus contention when there are too many processors and by the difficulty of physically constructing these long, high-speed buses. In addition, snoopy cache coherence protocols do not suit general interconnection networks, mainly because a snooping protocol requires a communication with all caches on every cache miss, including writes of potential shared data.

Directory-based cache coherence protocols

In the directory-based multiprocessors, no broadcast is necessary in this case and therefore the network medium may be of almost any kind (should be fast enough to maintain consistency though). Each processor communicates with a common directory whenever the processor's action may cause an inconsistency between its cache and the other caches or memory. The directory maintains the information about which processor has the copy of which a b oak since several processors may have a copy of the same block cached at the same time. To prevent the directory from becoming the bottleneck, directory entries can be distributed along with the memory, so that different directory accesses can go to different locations. The basic concept of this protocol is that a processor must ask for permission to load an entry from the primary memory to its cache. It asks a directory which has information about which caches contain which entries. When an entry is changed, the directory must be notified either before the change is initiated or when it is complete, and other caches with that entry also must be updated or invalidated.

Most directory-based cache coherence protocols use the write-back policy, which is based on the states of directory and cache. The basic directory states include:

a. Shared: one or more processors have the block cached, and the value in memory is up to date.

b. Uncached: no processor has a copy of the cache block.

c. Exclusive: exactly one processor has a copy of the cache block and it has written the block, so the memory copy is out of date.

While the cache states include：

a. Exclusive: the line in the cache is the same as that in main memory and is not present in any other cache.

b. Shared: the line in the cache is the same as that in main memory and may be present in another cache.

c. Invalid: the line in the cache does not contain valid data.

However, the overhead of directory maintenance and look-up time plus the high-latency of communication networks make the directory scheme unattractive. The major drawbacks of the protocol with directory-based mechanisms are the high coherence traffic due to all requests to the directory and the great need for memory. If precision of block-sharing information is high, cache block size is small and there are a large amount of processors in the system, large proportions of primary memory will be dedicated to the directory service.

The development of cache design becomes more and more impacted complex by utilizing different domains, such as super-pipelining, super-scaling, multithreading, prediction, parallelization, etc. Although the existing improvement of cache coherence protocols is substantial, the inherently slow DRAM access still presents a significant gap with respect to the speed of the processor. In practice the cache coherence protocols are notoriously difficult to implement, debug, and maintain; the details of the protocols depend on the requirements of the system under

consideration and are highly varied. Because the trade-off exists between network traffic and directory size, no commercial implementation yet uses directory schemes.

Acknowledgements

The research work was supported by Gansu Province Groups of Basic Research Innovation Projects No.: 145RJIA333;Gansu Province Science and Technology Support Projects No.: 1304FKCA082.

References

[1] I.Tartalja, V. Milutinovic, The Cache Coherence Problem in Shared-Memory Multiprocessors: Software Solutions. IEEE Computer Society Press, 2003.

[2] J.K. Peir, W.W. Hsu, A.J. Smith, Functional implementation techniques for CPU cache memories. IEEE Trans. Comput. 48 (1999) 100-110.

[3] L.M. Censier, P.A. Feautrier, New solution to coherence problems in multicache system. IEEE Trans. Comput. 27 (1988) 1112-1118.

[4] C.K. Tang, Cache design in the tightly coupled multiprocessor system. Proc. AFIPS Nat. Comput. Conf., 1976, pp. 749-753.

[5] S.J. Frank, Tightly coupled multiprocessor systems speed memory access times. Electronics 57 (1984) 164-169.

[6] A. Agarwal, An evaluation of directory schemes for cache coherence. Proc. 15th Int'l Syrup. Comput. Archit., 1990, pp. 280-289.

[7] S. Thakkar, M. Dubois, A.L. Laundrie, Scalble shared-memory multiprocessor architecture. Computer 23 (1988) 71-74.

[8] W. Stallings, Computer Organization and Architecture: Design for Performance. Prentice Hall, 1996.

[9] J. Archibald J.L. Baer, Cache coherence protocols: evaluation using a multiprocessor simulation model. ACM Trans. Comput.Syst. 4 (1986) 273-298.

[I0] M. Thapar, B. Delagi, Standford distributed-directory protocol. Computer 23 (1990) 78-80.

[11] E.A. Stenstrom, Survey of cache coherence schemes for multiprocessors. Computer 23(1990) 12-24.