Cache Coherence Protocols in NUMA Multiprocessors

NUMA 다중 프로세서에서의 캐쉬 일관성 프로토콜

Sang-Man Moh (모상만) Computer System Department (하드웨어구조연구팀 선임연구원)
Woo-Jong Hahn (한우종) Computer System Department (하드웨어구조연구팀 책임연구원, 팀장)
Suk-Han Yoon (윤석한) Computer System Department (컴퓨터시스템연구부 책임연구원, 부장)

Recently, scalable multiprocessor systems are actively developed for general-purpose computing, which are based on distributed shared memory (DSM) architecture to boost up both programmability and scalability. In this paper, we survey and analyze cache coherence protocols in non-uniform memory access (NUMA) multiprocessor systems. In particular, it has been easily inferred that specialized hardware suitable for NUMA multiprocessor systems with commodity symmetric multiprocessors (SMPs) is highly required. The cache coherence protocol combined with specialized hardware can significantly improve the performance and scalability of NUMA multiprocessor systems, providing better programmability.

I. Introduction

Parallel computing systems with multiple processors are usually classified into two large groups, namely shared-memory multiprocessors and message-passing multicomputers. The major distinction between multiprocessors and multicomputers lies in memory sharing and the mechanisms used for interprocessor communication [1].

In a shared-memory multiprocessor system, a global physical memory is equally accessible to all processors. An important advantage of such a system is the general and convenient programming model that enables simple data sharing through a uniform mechanism of reading and writing shared variables in the common memory. In a message-passing multicomputer system, multiple independent processing nodes with local memory modules communicate with each other through message passing. Message-passing multicomputer systems are claimed to be scalable, and systems with very high computing power are possible [2].

The concept of distributed shared memory (DSM) tries to combine the advantages of the two approaches [2]. A DSM system logically implements the shared-memory model in a physically distributed memory system. The ease of programming and the portability of shared-memory systems are preserved. In addition, the scalability and cost-effectiveness of underlying message-passing systems are also

inherited. The main objective of research in DSM systems is the development of general approaches that minimize the average access time to shared data, while maintaining data consistency.

Designing and programming multiprocessor systems correctly and efficiently pose complex problems. Synchronizing processes, maintaining data coherence, and ordering events in a multiprocessor are issues that must be addressed from the hardware design level up to the programming language level [3]. There are also some important issues, such as scheduling and partitioning.

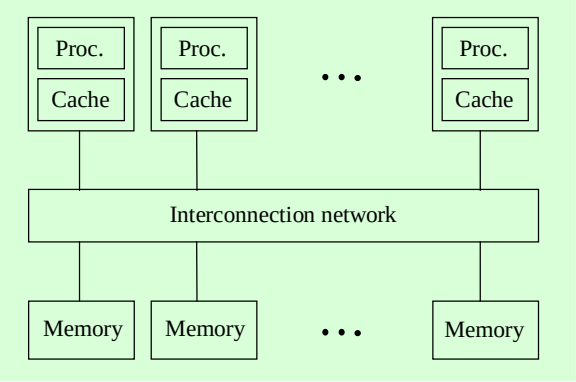

Fig. 1. A shared-memory multiprocessor system with private caches.

Cache coherence problems exist in multiprocessors with private caches and are caused by three factors: sharing of writable data, process migration, and I/O activity [3]. Figure 1 shows a shared-memory multiprocessor system with private caches, in which multiple processors and shared memory modules are interconnected through an interconnection network. A multiprocessor system with multiple caches faces the problem of making sure that all copies of a given piece of information are the same. A modification of any one of these copies should somehow be reflected in all others [4]. This is called the cache coherence problem or the cache consistency problem. A system of caches is said to be coherent if every read by any processor finds a value produced by the last previous write, no matter which processor performed it [7]. The importance of the cache coherence problem is emphasized by the fact that not only in the cache coherence solution necessary for correct program execution, but it can have a significant impact on system performance.

In Section II of this paper, the cache coherence protocols for shared-memory multiprocessors are described systematically. The NUMA multiprocessor systems and the cache coherence mechanisms for cache-coherent NUMA systems are presented in Section III. In Section IV, past related works on NUMA multiprocessor systems and cache coherence protocols are given and analyzed in detail, which include research prototypes and commercial systems. And some research issues and the concluding remarks of this paper are covered in Section V.

II. Cache Coherence Protocols

There are two policies for maintaining cache coherence: write-invalidate and write-update policies [5]. Figure 2 shows the conceptual behavior of write operations both for write-invalidate policy and for write-update

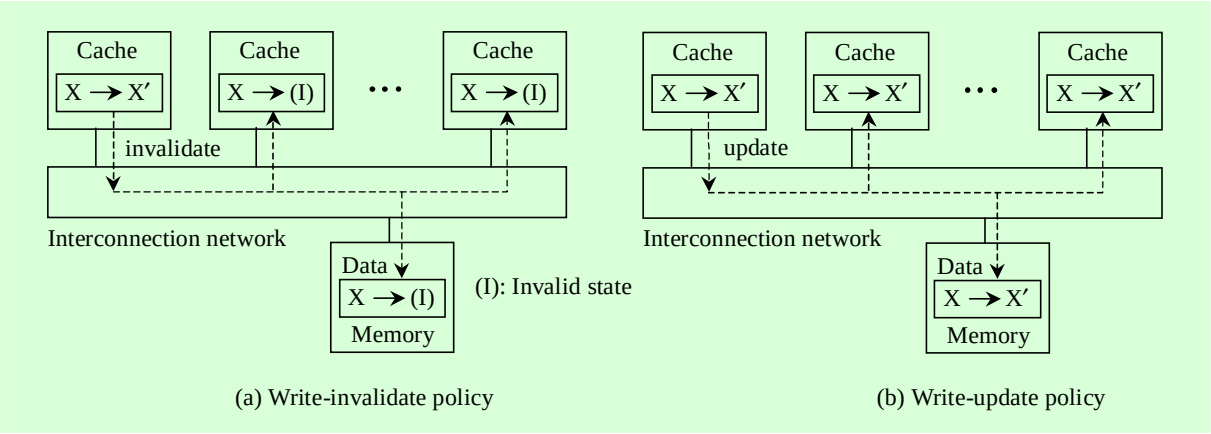

Fig. 2. Two write policies for maintaining cache coherence.

policy. The write-invalidate policy maintains coherence of multiple copies in the following way. Read requests are carried out locally if a copy of the block exists. When a processor updates a block, however, all other copies are invalidated. The write-update policy maintains coherence differently. Instead of invalidating all copies, it updates them. Whether the memory copy is updated or not depends on how this protocol is implemented.

A variety of mechanisms have been proposed for solving the cache coherence problem. The optimal solution for a given multiprocessor system depends on several factors, such as the size of the system, the anticipated usage of the system, and the desired system cost [6]. The cache coherence mechanisms are mainly classified into two classes: software-based and hardware-based solutions.

Software-based solutions generally rely on the actions of the programmer, compiler, or operating system, in dealing with the coherence problem [7]. The simplest way is to declare non-cacheable pages of shared data. More advanced methods allow the caching of shared data and accessing them only in critical sections in mutually exclusive way. Software schemes are generally less expensive than their hardware counterparts. Some disadvantages, however, are evident where inevitable inefficiencies are incurred since the compiler analysis is unable to predict the flow of program execution accurately and conservative assumptions have to be made.

Hardware-based solutions efficiently support the full range from small- to large-scale multi-processors [7]. Although they require an increased hardware complexity, their cost is well justified by the significant advantages. Hardware schemes deal with the coherence problem by dynamic recognition of inconsistency conditions for shared data entirely at run time. Being totally transparent to software, hardware protocols free the programmer and compiler from any responsibility for coherence maintenance, and impose no restrictions

on any layer of software. Technological advances have made their cost quite acceptable.

Hardware cache coherence schemes can be principally divided into two large groups: snoopy and directory protocols [7]. In snoopy cache coherence protocols, coherence maintenance is based on the actions of local cache controllers and distributed local state information. Neither centralized controller nor global state information is employed. So all the actions for the currently shared block must be announced to all other caches via broadcast capability. Local cache controllers are able to snoop on the network and to recognize the actions and conditions for coherence violation, which impose some reactions in order to preserve coherence.

Snoopy protocols are ideally suited for multiprocessors that use a shared bus as a global interconnect since the shared bus provides very inexpensive and fast broadcast [7]. They are also known for being very cost-effective and flexible schemes. However, coherence actions on the shared bus increase the bus traffic and make the bus saturation more acute. Consequently, only systems with a small to medium number of processors can be supported by snoopy protocols.

In directory-based cache coherence protocols, the global, system-wide state information relevant for coherence maintenance is stored in some kind of a directory [7]. The responsibility of coherence is predominantly delegated to a centralized directory controller that is usually a part of the main memory controller. Upon the individual requests of the local cache controllers, the directory controller checks the directory and issues necessary commands for data transfer between memory and caches, or between caches themselves. It is also responsible for keeping state information up-to-date, so every local action that can affect the global state of the block must be reported to the central directory controller. Besides the global directory maintained by the central controller, private caches store some local state information about cached blocks. Directory-based cache coherence protocols are primarily suitable for multiprocessors with general interconnection networks [7].

Agarwal *et al.* [8] introduced one useful classification of directory schemes denoting them as Dir_iX, where i is the number of pointers, and X is B or NB, for broadcast and non-broadcast schemes, respectively.

The different flavors of directory protocols fall under three primary categories: full-map directories, limited directories, and chained directories [9]. Figure 3 shows the three types of directory organizations. Full-map directories store enough state associated with each block in global memory so that every cache in the system can simultaneously store a copy of any block of data. That is, each directory entry contains N pointers, where N is the number of processors in the system. Limited directories differ from full-map directories in that they have a fixed number of pointers per entry, regardless of the number of processors in the system. Limited directory protocols are designed to solve the directory size problem. Restricting the number of simultaneously cached

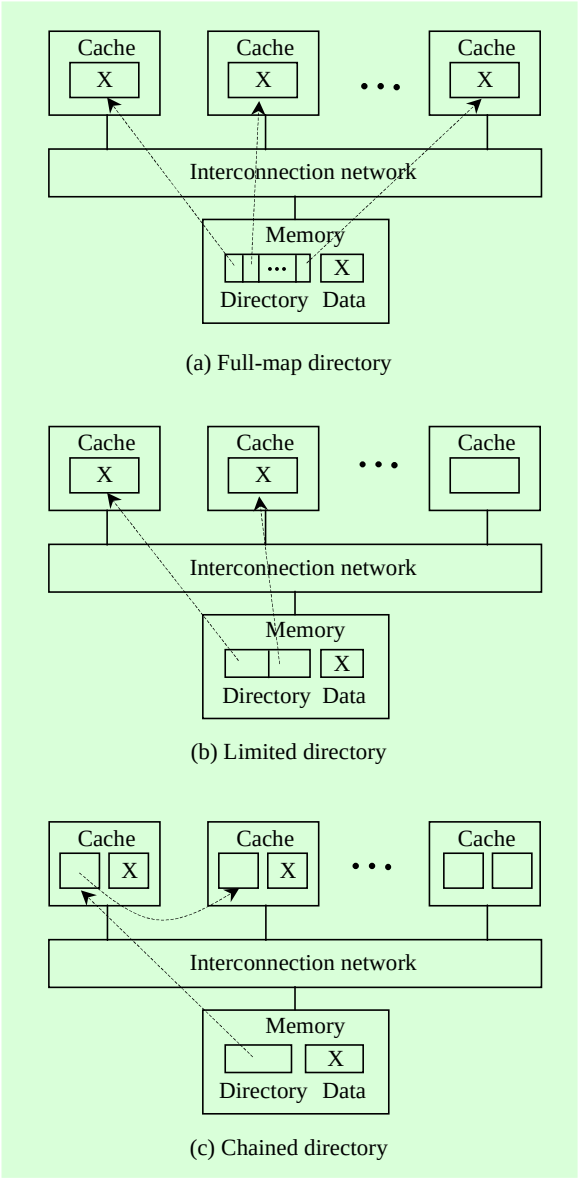

Fig. 3. Three types of directory organizations.

copies of any particular block of data limits the growth of the directory to a constant factor. Chained directories emulate the full-map schemes by distributing the directory among the caches. Chained directories keep track of shared copies of data by maintaining a chain of directory pointers, and they do not untilize a broadcast mechanisms. These schemes realize the scalability of limited directories without restricting the number of shared copies of data blocks.

III. NUMA Systems and Cache Coherence

Multiprocessor systems are suitable for general-purpose multiuser applications where programmability is the major concern. Shared-memory multiprocessors are roughly categorized into two large models: the uniform memory access (UMA) model and non-uniform memory access (NUMA) model [1]. In a UMA multiprocessor model, the physical memory is uniformly shared by all the processors. All processors have equal access time to all memory words, which is why it is called uniform memory access.

When all processors have equal access to all peripheral devices, the system is called a symmetric multiprocessor (SMP). In this case, all the processors are usually capable of running the execution programs, such as the OS kernal and I/O service routines.

A NUMA multiprocessors is a shared-memory system in which the access time varies with the location of the memory word. The shared memory is physically distributed to all processors, called local memories. The collection of all local memories forms a global address space accessible by all processors. It is faster to access a local memory with a local

processor. The access of remote memory attached to other processors takes longer due to the added delay through the interconnection network.

As a variation of NUMA model for multiprocessors, a cache-coherent non-uniform memory access (CC-NUMA) model can be specified with distributed shared memory and cache coherence mechanism [1]. A major concern of CC-NUMA multiprocessor systems is the cache-coherent memory access transparent to all the processors. Latency tolerance for remote memory access is also a major limitation of CC-NUMA systems.

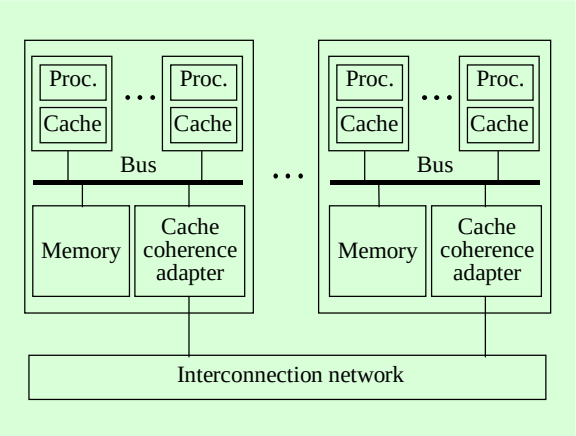

Fig. 4. Typical example of an SMP-based CC-NUMA multiprocessor system.

As mentioned in the above, a major shortcoming of multiprocessors is the lack of scalability. It is rather difficult to build large-scale machines using centralized shared-memory model. As a result, DSM systems have been studied and developed for general-purpose computing. SMP-based CC-NUMA systems are hierarchically structured multiprocessor machines, which have two layers of architecture: SMP and CC-NUMA models. In an SMP-based CC-NUMA multiprocessor system, SMP nodes are interconnected via the interconnection network based on the cache-coherent non-uniform memory access model. All processors belonging to the same SMP node are allowed to uniformly access the node memory modules. All SMP nodes have access to all physically distributed memories. However, the access time to remote memories is longer than that to the local shared memory. Figure 4 shows a typical example of an SMP-based CC-NUMA multiprocessor system, where the bus-based SMP nodes are interconnected through an interconnection network. The availability of cost-effective commodity SMPs makes them an attractive choice for CC-NUMA designers.

In SMP-based CC-NUMA multiprocessor systems, multilevel cache hierarchy seems to be unavoidable [7]. Lower levels of such a hierarchy are smaller, but faster, caches. Their task is to reduce miss penalty. Upper level caches are slower but much larger in order to attain higher hit ratio and reduce the traffic on the interconnection network.

There are two alternatives in implementing a cache coherence controller for multiprocessor systems, in particular, for SMP-based CC-NUMA multiprocessor systems: a custom hardware coherence controller and a coherence controller based on commodity protocol processors [10]. Custom hardware coherence controllers can be efficiently implemented for spe-

cific target systems, and run faster. Coherence controllers based on protocol processors allow the coherence protocol to be tailored to specific application needs.

The cache coherence schemes that combine the principles and advantages of both snoopy and directory protocols on different levels may be highly effective for large-scale shared-memory multiprocessors [7]. In most of SMP-based CC-NUMA multiprocessor systems, directory protocols are used for internode cache coherence since the systems have many processors and general interconnection networks.

IV. Related Works

Some research prototypes and commercial systems of NUMA multiprocessors based on SMP nodes have been developed. Typical examples of such systems are DASH [11]-[13], ASURA [14], NUMAchine [15], and NUMA-Q [16]. New machines based on SMP-based CC-NUMA multiprocessors are being developed or shipped by some companies today. These kinds of multiprocessors typically use the latest commodity microprocessors available at the time they are shipped. Moreover, because of cost-effectiveness and time-to-market requirements, off-the-shelf commodity SMPs are widely used as computing nodes in developing the cache-coherent NUMA multiprocessor systems.

The DASH prototype system was an outgrowth of research into scalable shared-memory multiprocessing at Stanford University. The primary goal of building the machine was a better understanding of the design issues and feasibility of this class of machines [11]-[13]. The system is limited to a 4×4 configuration with 16 nodes, due to the constraints in memory addressing of the base node. In the DASH prototype, SMP nodes are interconnected via a pair of 2-D mesh networks. The global shared memory is distributed among the SMP nodes called processing nodes. Each node contains four R3000 processors with a 256-Kbyte write-back cache per processor, a portion of global memory, and local I/O devices. The size of each node's memory partition is only 16 Mbytes. The two mesh networks are called request network and reply network, respectively, each of which guarantees point-to-point delivery of messages without deadlocks.

A bus-based snoopy scheme is used to keep caches coherent within an SMP node, while internode cache coherence is maintained using a distributed directory-based coherence protocol. The DASH cache coherence protocol is an invalidation-based ownership protocol. A memory block can be in one of three states as indicated by the associated full-map directory entry: uncached, shared-remote, and dirty-remote. The remote access cache is direct-mapped, write-back cache with 16-byte lines, which contains only 128 Kbytes of remote memory data. The remote access cache of 128 Kbytes is very small, and its line size of 16 bytes is also relatively short compared to the processor cache line size of 16 bytes. Due to the small size and short

lines of the remote access cache, the performance of the cache coherence protocol can be degraded. Slow lookup of the DRAM directory against node bus transactions is also a shortcoming of the DASH implementation.

ASURA [14] is a large, SMP-based, distributed shared-memory multiprocessor system developed at Kyoto University and Kubota Corporation. It consists of up to 128 nodes interconnected through an internode network. The ASURA node is configured as a bus-based SMP with up to eight R4000MC processors, a 256-Mbyte local shared memory, a 4-Gbyte global shared memory, and I/O devices. Each processor has a 4-Mbyte write-back cache. While the local shared memory is shared only by up to eight processors in the node, the global shared memory is shared by all nodes. Several candidate topologies for the internode network were considered, from which various configurations for the network could be chosen to obtain a transfer rate of 400 Mbytes per second for each link.

While intranode cache coherence is maintained by a snoopy protocol, internode cache coherence is done by a full-map directory protocol. The SMP node has a remote access cache, called global cache, which is a 4-way set-associative, 32-Mbyte write-back cache with 1024-byte lines. In the directory-based internode cache coherence protocol, a cached memory block can be in one of three states: invalid, clean, and dirty. The 1024-byte line of the remote access cache is too long, which can make false sharing problem become more serious. The memory overhead of the directory is a significant constraint since the system connects up to 128 nodes using the full-map directory scheme. Such constraints may restrict the scalability and degrade the performance of the ASURA system.

NUMAchine [15] is a cache-coherent shared-memory multiprocessor designed at University of Toronto. It is an SMP-based CC-NUMA machine designed to connect up to 16 nodes via a hierarchy of unidirectional bit-parallel rings. Each bus-based SMP node consists of four R4400 processors, a 2-Gbyte memory, and I/O devices. The R4400 processor internally contains a 1-Mbyte write-back cache. The internode rings are divided into two levels of hierarchy: a global ring and local rings.

The NUMAchine cache coherence protocol employs a write-back and write-invalidate scheme. A snoopy coherence protocol is used for four processors in each SMP node. To maintain internode cache coherence, a hierarchical, two-level full-map directory exists for global and local rings. Each SMP node contains a remote access cache, called network cache, which is a write-back 8-Mbyte cache. Four basic states are defined for a cache line in a memory module or a remote access cache: local valid, local invalid, remote valid, and remote invalid. The two-level full-map directory inevitably includes the complicated control of the cache coherence protocol which may be a hurdle in maintaining the cache coherence efficiently. The hierarchical ring architecture also has the latency problem since multiple hops are serialized to hand over data.

NUMA-Q [16], initially named STiNG, is a CC-NUMA multiprocessor system designed and built by Sequent Computer Systems, Inc. A key enabler of the NUMA-Q architecture is the off-the-shelf 4-way SMP building block. It combines up to 16 4-processor SMPs called Quads, using a scalable coherent interface (SCI) interconnect. The SMP nodes are based on the Intel Pentium Pro processor and external bus it defines. In addition to four processors, each node may contain 4 Gbytes of system memory and I/O devices. The Pentium Pro processor has a 4-way set-associative, 512-Kbyte, write-back cache with 32-byte lines. SCI-based dual rings are used for the internode interconnection network.

Within an SMP node, cache coherence is maintained using a snoopy cache coherence protocol called MESI protocol. Each node contains a bridge board called IQ-Link, initially named Lynx, that plugs into the local SMP bus. Included in the board is a 4-way set-associative, 32-Mbyte, write-back remote access cache with 64-byte lines. A chained directory implements a directory-based internode cache coherence protocol, which contains forward and backward pointers. That is, a doubly-linked list directory structure is used to maintain the internode cache coherence over the SCI-based dual rings. The directory state bits indicate whether each line is: home, fresh, or gone. Inherently, the chained directory can not allow any direct access to one of chained cache data. Moreover, simultaneous or out-of-order invalidations are also impossible since the architecture does not have any broadcast or multicast capability. Although the doubly-linked list directory structure improves the performance limitation, such restrictions may be drawbacks of scalable performance.

Origin[18], [19] is also a CC-NUMA multiprocessor system designed and manufactured by Silicon Graphics, Inc. However, every node is not an SMP architecture in the Origin system. So we simply look through key features of the system. Now the Origin system consists of up to 64 nodes interconnected by a scalable Craylink network. Each node consists of one or two R10000 processors, up to 4 Gbytes of coherent memory, and an I/O subsystem. Internode cache coherence is maintained by an invalidation-based, full-map directory protocol, which is similar to the DASH protocol.

V. Issues and Concluding Remarks

Although some works have been carried out to build and improve cache coherence protocols in designing NUMA multiprocessor systems, there are still many research issues remained. The issues are mainly divided into two large categories: correctness and performance. The correctness issue includes the ordering and consistency of memory accesses, deadlock, livelock, starvation, and the verification of correctness. The performance issue includes data access latency, bandwidth, and protocol overhead. The latency and bandwidth of data accesses are the major concerns

of cache coherence protocol design, which significantly influence system performance. The protocol overhead is tightly related to both latency and bandwidth, and restricts the efficiency of cache coherence protocols.

We believe there are many ways to improve performance characteristics such as latency, bandwidth and protocol overhead in designing a cache coherence protocol for NUMA multiprocessors. In particular, more aggressive and specialized hardware seems to be a useful solution for high-performance implementation.

Recently, off-the-shelf commodity SMPs like Intel standard high-volume (SHV) servers [20] are more popular in the commercial marketplace. Moreover, well-developed system area interconnects based on packet-switched interconnection networks provide higher performance with scalability. In such an SMP-based CC-NUMA multiprocessor system, a cache coherence protocol is significantly important in that the scalability of system performance is determined by the performance of the cache coherence protocol. Specialized hardware is highly required to improve the performance of the cache coherence protocol since it can provide the cache-coherent interconnection of SMP nodes, with minimal degradation of scalability. An efficient design of specialized hardware also minimizes protocol overhead resulting in the improvement of internode latency and bandwidth.

As we have described so far, some research and development works on cache coherence protocols for NUMA multiprocessor systems have been carried out for a decade. However, even though some innovative works have not only triggered the research and development of such architectures but also enabled commercially successful systems, there still exist many problems and issues to be digged as mentioned.

In this paper, we have presented cache coherence protocols in NUMA multiprocessor systems. In particular, it has been easily inferred that specialized hardware suitable for CC-NUMA multiprocessor systems with off-the-shelf commodity SMPs is highly required. The specialized hardware can make a cache coherence protocol enable high-performance system-wide data accesses with better programmability.

There exist many research topics to be digged in the future, including the verification of protocol correctness, performance evaluation and comparison, the minimization of protocol overhead, directory size problems, and cost-to-performance tradeoffs.

References

[1] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability,* McGraw-Hill, Inc., New York, 1993, pp. 3–32.

[2] J. Protic, M. Tomasevic, and V. Milutinovic, "An Overview of Distributed Shared Memory," *Distributed Shared Memory: Concepts and Systems,* J. Protic, M. Tomasevic, and V. Milutinovic, Ed. IEEE Computer Society Press, Los Alamitos, California, 1998, pp. 12–41.

[3] F.A. Briggs, "Synchronization, Coherence, and Event Ordering in Multiprocessors," *IEEE Computer,* Vol. 21, No. 2, Feb. 1988, pp. 9–21.

[4] A.J. Smith, "Cache Memories," *ACM Computing Surveys,* Vol. 14, No. 3, Sep. 1982, pp. 473–530.

[5] P. Stenstrom, "A Survey of Cache Coherence Schemes for Multiprocessors," *IEEE Computer,* Vol. 23, No. 6, June 1990, pp. 12−24.

[6] D.J. Lilja, "Cache Coherence in Large-Scale Shared-Memory Multiprocessors: Issues and Comparisons," *ACM Computing Surveys,* Vol. 25, No. 3, Sep. 1993, pp. 303−338.

[7] M. Tomasevic and V. Milutinovic, "Hardware Solutions for Cache Coherence in Shared-Memory Multiprocessor Systems," *The Cache Coherence Problem in Shared-Memory Multiprocessors: Hardware Solutions,* M. Tomasevic and V. Milutinovic, Ed., IEEE Computer Society Press, Los Alamitos, California, 1993, pp. 57−67.

[8] A. Agarwal, R. Simoni, J. Hennessy, and M. Horowitz, "An Evaluation of Directory Schemes for Cache Coherence," *Proc. 15th Int. Symp. on Computer Architecture,* Honolulu, Hawaii, May 30−June 2, 1988, pp. 280−289.

[9] D. Chaiken, C. Fields, K. Kurihara, and A. Agarwal, "Directory-Based Cache Coherence in Large-Scale Multiprocessors," *IEEE Computer,* Vol. 23, No. 6, June 1990, pp. 49−58.

[10] M.M. Michael, A.K. Nanda, B.-H. Lim, and M.L. Scott, "Coherence Controller Architectures for SMP-Based CC-NUMA Multiprocessors," *Proc. of 24th Int. Symp. on Computer Architecture,* Denver, Colorado, June 2-4, 1997, pp. 219−228.

[11] D. Lenoski, J. Laudon, K. Gharachorloo, A. Gupta, and J. Hennessy, "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor," *Proc. of 17th Int. Symp. on Computer Architecture,* Seattle, Washington, May 28-31, 1990, pp. 148−159.

[12] D. Lenoski, J. Laudon, K. Gharachorloo, W.-D. Weber, A. Gupta, J. Hennessy, M. Horowitz, and S. Lam, "The Stanford Dash Multiprocessor," *IEEE Computer,* Vol. 25, No. 3, Mar. 1992, pp. 63−79.

[13] D. Lenoski and W.-D. Weber, *Scalable Shared-Memory Multiprocessing,* Morgan Kaufmann Publishers, San Francisco, California, 1995, pp. 173−203.

[14] S. Mori, H. Saito, M. Goshima, M. Yanagihara, T. Tanaka, D. Fraser, K. Joe, H. Nitta, and S. Tomita, "A Distributed Shared memory Multiprocessor: ASURA," *Proc. of Int. Conf. on Supercomputing,* Portland, Oregon, Nov. 1993, pp. 740−749.

[15] Z. Vranesic, S. Brown, M. Stumm, S. Caranci, A. Grbic, R. Grindley, M. Gusat, O. Krieger, G. Lemieux, K. Loveless, N. Manjikian, Z. Zilic, T. Abdelrahman, B. Gamsa, P. Pereira, K. Sevcik, A. Elkateeb, and S. Srbljic, *The NUMAchine Multiprocessor,* Technical Report CSRI-324, Computer Systems Research Institute, University of Toronto, Canada, 1995.

[16] T. Lovett and R. Clapp, "STiNG: A CC-NUMA Computer System for the Commercial Marketplace," *Proc. of 23th Int. Symp. on Computer Architecture,* Philadelphia, Pennsylvania, May 22−24, 1996, pp. 308−317.

[17] T.D. Lovett, R.M. Clapp, and R.J. Safranet, NUMA-Q: An SCI-Based Enterprise Server, Sequent Computer Systems, Inc., http://www.sequent.com/news/papers/qnumasci/ (current Mar. 1998).

[18] J. Laudon and D. Lenoski, "The SGI Origin: A ccNUMA Highly Scalable Server," *Proc. of 24th Int. Symp. on Computer Architecture,* Denver, Colorado, June 2−4, 1997, pp. 241−251.

[19] Silicon Graphics, Inc., *Origin Technology,* http://www.sgi.com/origin/technology.html (current Mar. 1998).

[20] Intel Corporation, *Standard High-Volume Servers: Changing the Rules for Business Computing,* http://www.intel.com/shv/servers/ (current Mar. 1998).