Supporting Multiple Coherence Protocols in Programmable Shared Memory Controllers

Ravindra Kuramkote, John Carter, and Chen-Chi Kuo

Department of Computer Science
University of Utah
Salt Lake City, UT 84112

Hardware distributed shared memory (DSM) systems are primarily targeted for parallel applications with fine-grained communication requirements. In this paper we show that one can reduce the number of remote memory misses, thereby improving performance, by using a DSM controller that supports multiple DSM protocols and managing each data object in the application with a protocol well-suited to how it is used.

Many researchers have espoused the benefits of using commodity workstations and networks in building the hardware DSM systems [2, 12, 4, 9] instead of using proprietary components [11, 3, 1]. Designs that use commodity components allow for faster prototype development and are more portable across existing hardware platforms. However, the latency of remote memory access in such systems tends to be higher since designers are unable to specialize the components. Hence, in such systems it is paramount to reduce the number of misses to remote memory to achieve high performance.

Remote memory misses fall into two types: *essential* and *non-essential* [5]. Essential misses are inherent in the parallel application and cannot be avoided. Non-essential misses are caused by: (i) inefficient use of the cache and memory on each node and (ii) use of sub-optimal coherency protocols for consistency management. Various memory models have been proposed to reduce the first type of non-essential misses [10, 7, 15]. We focus on reducing the non-essential misses induced by sub-optimal consistency management.

Several recent architectures employ programmable DSM controllers [9, 8, 6]. The programmability of such controllers can be exploited to reduce the number of non-essential consistency misses by supporting multiple coherence protocols. In such a system, the programmer or compiler can specify the protocol to use when allocating memory for a data object. In our design, this protocol information is managed on a per-page basis by the DSM controller in a manner similar to how remote page information is tracked in Simple-COMA architectures [13]. On a cache miss, the DSM controller executes the appropriate protocol (e.g., write-invalidate, write-update, etc.).

To evaluate this idea, we first modeled an ideal system in which all non-essential misses (i.e., all misses after the first cold miss) could be satisfied by a zero-latency coherence protocol [14]. We then measured the performance of a set of programs from the Splash-2 and Wisconsin Wind Tunnel benchmark suite. We found that architectures employing a single release-consistent (RC) write-invalidate protocol for all data objects incurred a performance loss of up to 79% compared to the ideal system. We then modified our simulator to support both an RC write-invalidate and an RC write-update protocol. Using just these two protocols, we were able to reduce the run-time of applications by 3–56% and the network bandwidth consumption by 18–75% compared to an architecture with just the write-invalidate protocol. Hence, we believe that a mechanism to map data objects to coherence protocols should be supported by machines with programmable DSM controllers.

References

[1] A. Agarwal and D. Chaiken et al. The MIT Alewife Machine: A large-scale distributed-memory multiprocessor. Technical Report Technical Memp 454, MIT/LCS, 1991.

[2] M.A. Blumrich, K. Li, R. Alpert, C. Dubnicki, E.W. Felten, and J. Sandberg. Virtual memory mapped network interface for the SHRIMP multicomputer. In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, pages 142–153, April 1994.

[3] H. Burkhardt, S. Frank, B. Knobe, and J. Rothnie. Overview of the KSR-1 computer system. Technical Report KSR-TR-9002001, Kendall Square Research, February 1992.

[4] A. Davis, M.R. Swanson, E. Brunvand, J.B. Carter, L.B. Stoller, and R. Kuramkote. Avalanche: Communication and memory architectures for scalable parallel computing. Technical report, University of Utah Department of Computer Science, 1996.

[5] M. Dubois, J. Skeppstedt, L. Ricciulli, and P. Stenström. The detection and elimination of useless misses in multiprocessors. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pages 88–97, May 1993.

[6] B. Falsafi, A.R. Lebeck, S.K. Reinhardt, I. Schoinas, M.D. Hill, J.R. Larus, A. Rogers, and D.A. Wood. Application-specific protocols for user-level shared memory. In *Proceedings of Supercomputing'94*, pages 380–389, November 1994.

[7] B. Falsafi and D.A. Wood. Reactive NUMA: A design for unifying S-COMA and CC-NUMA. In *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pages 229–240, June 1997.

[8] R. Kuramkote, J.B. Carter, A. Davis, C.-C. Kuo, L.B. Stoller, and M.R. Swanson. Analysis of avalanche's shared memory architecture. Technical Report UUCS-97-008, University of Utah Department of Computer Science, July 1997.

[9] J. Kuskin and D. Ofelt et al. The Stanford FLASH multiprocessor. In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, pages 302–313, May 1994.

[10] A. Moga and M. Dubois. The effectiveness of SRAM network caches in clustered DSMs. In *Proceedings of the Fourth Annual Symposium on High Performance Computer Architecture*, 1998.

[11] A. Nowatzyk, G. Aybay, M. Browne, E. Kelly, M. Parkin, B. Radke, and S. Vishin. The S3.mp scalable shared memory multiprocessor. In *Proceedings of the 1995 International Conference on Parallel Processing*, 1995.

[12] S.K. Reinhardt, J.R. Larus, and D.A. Wood. Tempest and Typhoon: User-level shared memory. In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, pages 325–336, April 1994.

[13] A. Saulsbury, T. Wilkinson, J.B. Carter, and A. Landin. An argument for Simple COMA. In *Proceedings of the First Annual Symposium on High Performance Computer Architecture*, pages 276–285, January 1995.

[14] G. Shah, A. Singla, and U. Ramachandran. The quest for a zero overhead shared memory parallel machine. In *Proceedings of the 1995 International Conference on Parallel Processing*, August 1995.

[15] Z. Zhang and J. Torrellas. Reducing remote conflict misses: Numa with remote cache versus coma. In *Proceedings of the Third Annual Symposium on High Performance Computer Architecture*, February 1997.