Processor-Directed Cache Coherence Mechanism – A Performance Study

H. Sarojadevi, dept. of CSE
Nitte Meenakshi Institute of Technology (NMIT)
Bangalore, India
hsarojadevi@gmail.com

S. K. Nandy
CAD Lab, SERC
Indian Institute of Science
Bangalore, India
nandy@serc.iisc.ernet.in

Abstract— **Cache coherent multiprocessor architecture is widely used in the recent multi-core systems, embedded systems and massively parallel processors. With the ever increasing performance gap between processor and memory, there is a requirement for an optimal cache coherence mechanism in a cache coherent multiprocessor. The conventional directory based cache coherence scheme used in large scale multiprocessors suffers from considerable overhead. To overcome this problem we have developed a compiler assisted, processor directed cache coherence mechanism and evaluated. The approach is auto-invalidation based one that uses a hardware buffer termed Coherence Buffer (CB) and there is no need for directory. The CB method is compared in this paper with a self-invalidation based directory approach that employs a last touch predictor (LTP). Detailed architectural simulations of Distributed Shared Memory configurations with superscalar processors show that 8-entry 4-way associative CB performs better than the LTP based self-invalidation method as well as full-map 3-hop directory for five of the SPLASH-2 benchmarks under release consistency memory model. Given its performance, cost, complexity and scalability advantages, the CB approach is found to be promising approach for emerging applications in large scale multiprocessors, multi-core systems, and transaction processing systems.**

Keywords- Cache coherence, Distributed shared memory multiprocessor system, self-invalidation, Last touch predictor, Release consistency

I. INTRODUCTION

Emerging class of high performance multiprocessor architectures such as multi-core systems **Error! Reference source not found.**, large scale multiprocessors [1], and massively parallel architectures [2] require faster and more efficient cache coherence mechanism. The constraints become more stringent for complex and critical applications such as biological sequence analysis, hospital information system, multimedia, transaction processing, signal processing, vehicle automation, and many grand challenge applications. In this paper we present performance study of a cost-effective hardware-software approach to cache coherence, which we term as Coherence Buffer (CB) scheme. The CB approach is processor-directed, auto-invalidation based method that does not need a directory. Results are presented to compare with a full-map directory and a directory based self-invalidation scheme called Last Touch Predictor (LTP). LTP uses speculative self-invalidation of cache blocks.

We use architectural simulations of various DSM configurations under Release Consistency (RC) memory model. SPLASH-2 benchmark suite, which is representative of different application domains with wider sharing patterns, is used for evaluating the performance comparison of CB and directory schemes. The programs are properly labeled (PL) [5] by the compiler to aid runtime coherence mechanism. The DSM system with 8 entry, 4-way associative CB with 16 processor configuration is simulated and compared with equivalent directory system with LTP. Simulated LTP has 16 first level registers each 12 bit long and 2-way associative second level last touch trace predictors. Speedup of CB over LTP is found to be 1.07-1.32 for five of the benchmarks. Thus the CB based approach improves memory performance and is most cost-effective. By avoiding global control, CB reduces large number of coherence transactions the system, thereby making the system more power aware. With its complete local control, the CB based system becomes highly scalable.

The rest of the paper is organized as follows. Section II provides details of the CB scheme and that of LTP. Section III presents methodology and the results. Conclusions are written in Section IV.

II. BACKGROUND

CB based approach proposed in [3] and [6] imposes early coherence transactions that are directed by individual processor unlike the globally controlled coherence in a directory based scheme. Several ways to improve the directory based methods have been experimented, out of which the best performing one is Last Touch Predictor [LTP] approach. Details of the CB approach and directory approach having LTP enhancement are provided in the sub-sections below.

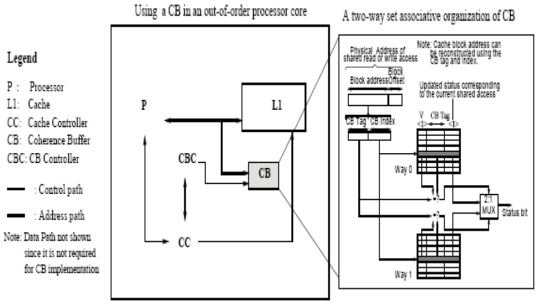

Using a CB in an out-of-order processor core
A two-way set associative organization of CB
Legend
P : Processor
L1: Cache
CC: Cache Controller
CB: Coherence Buffer
CBC: CB Controller
: Control path
: Address path
Note: Data Path not shown since it is not required for CB implementation
P
L1
CBC
CB
CC
Physical Address of shared read or write access
Note: Cache block address can be reconstructed using the CB tag and index.
Block address
Block Offset
Updated status corresponding to the current shared access
CB Tag
CB Index
V
CB Tag
<1>
<1>
Way 0
Way 1
2:1 MUX
Status bit

Figure 1. Processor, CB, and Cache interactions

A. Compiler assisted Coherence Buffer based approach

This approach uses an on-chip buffer called CB [6] that has cache like structure shown in Figure 1. A controller, CBC carries out all CB related operations. The CBC dynamically identifies all shared accesses between two consecutive release boundaries that are identified with the help of compiler annotations [6]. These accesses are recorded in the CB and are indexed using block address of a cache line. The CB invalidates or updates corresponding blocks in memory at data sharing boundaries (release synchronization points) thereby enforcing coherence. The cache block address is further split into CB tag and CB index to facilitate address reconstruction during CB update and flush operations. The CBC coordinates with the cache controller (CC) for tracking changes in the cache.

B. Last Touch Predictor based Directory method

The approach improves directory performance by speculative self-invalidation of cache blocks using a predictor, called LTP. An LTP (figure 2) is a two level, adaptive predictor used for predicting memory invalidations [4]. The first level, called history table, stores memory traces generated by a processor in the form of encoded trace signature. The second level is a block invalidation predictor table storing a list of previously observed trace signatures for each memory block. LTP predicts the "last touch" to a memory block which indicates that the memory block is not going to be used and hence can be invalidated. Based on the prediction, memory blocks are evicted from the local cache without waiting for the directory to send explicit invalidation request. In case of misprediction, a recovery action needs to be initiated, and this involves multiple network transactions.

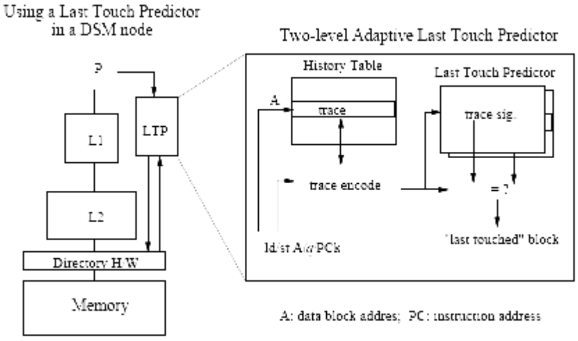


Figure 2. Using Last Touch Predictor in a DSM

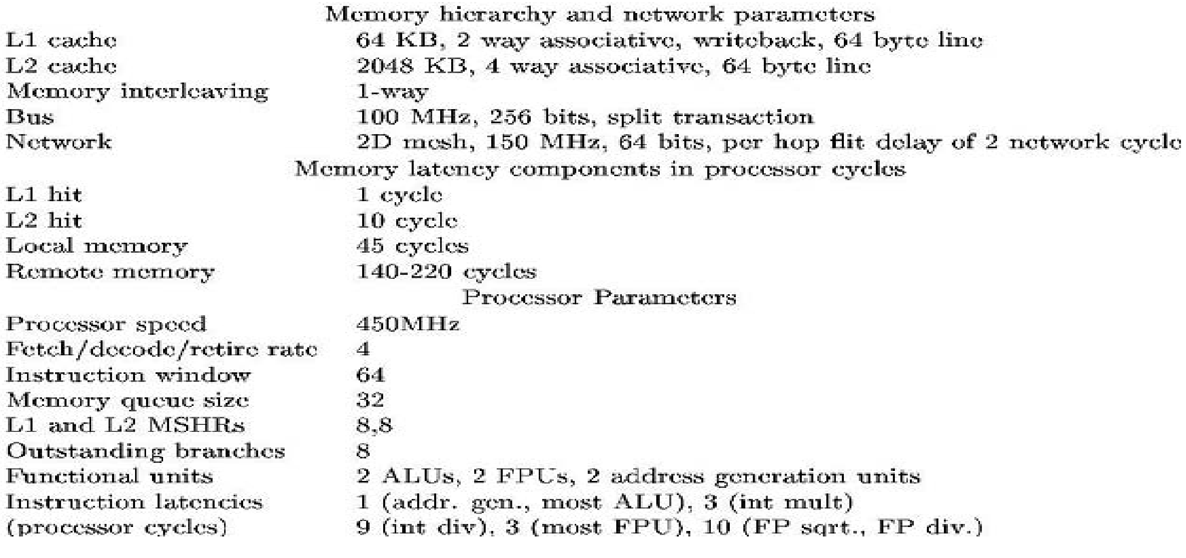
While the LTP is considered as best self-invalidation scheme, it is more complex than the directory based scheme. Besides the directory transactions for regular and misprediction situations increase the overhead as more network level transactions incur. A simulation based study to investigate this aspect and comparing CB approach is presented in the following section.

III. Methodology and Results

A. Simulation Details

RSIM simulator [7] on Solaris platform has been used for our experiments on DSM, DSM with LTP and the CB approaches. Simulator parameters are as given in table 1.

TABLE 1. SYSTEM PARAMETERS



Memory hierarchy and network parameters	
L1 cache	64 KB, 2 way associative, writeback, 64 byte line
L2 cache	2048 KB, 4 way associative, 64 byte line
Memory interleaving	1-way
Bus	100 MHz, 256 bits, split transaction
Network	2D mesh, 150 MHz, 64 bits, per hop flit delay of 2 network cycle
Memory latency components in processor cycles	
L1 hit	1 cycle
L2 hit	10 cycle
Local memory	45 cycles
Remote memory	140-220 cycles
Processor Parameters	
Processor speed	450MHz
Fetch/decode/retire rate	4
Instruction window	64
Memory queue size	32
L1 and L2 MSHRs	8,8
Outstanding branches	8
Functional units	2 ALUs, 2 FPUs, 2 address generation units
Instruction latencies (processor cycles)	1 (addr. gen., most ALU), 3 (int mult) 9 (int div), 3 (most FPU), 10 (FP sqrt., FP div.)

RSIM is modified to incorporate the LTP into the directory scheme, this version is DSMLTP. The CB based coherence mechanism is built into RSIM after removing the directory part, retaining the processor and memory hierarchy structure - DSMCB. DSMDir uses full map three hop MSI invalidate based approach, forming the base configuration.

SPLAH-2 benchmark suite of applications is used for the performance evaluation. The applications include - MP3D(50000 particles), Water(512 molecules), LU(512x512 matrix) with block size of 16, optimized LU i.e, LUOPT, FFT(256K points) and its optimized version FFTOPT, Radix(512K keys) and Quicksort(32K integers) ; these applications are compiler-annotated for proper labeling [7].

B. Invalidation Prediction with LTP

Predicting invalidations correctly is important for the performance of LTP based coherence mechanism. The results in figure 3 show fraction of invalidations, giving a split up of number of correct predictions obtained by the LTP approach. It can be observed that major fraction of the invalidations is mispredicted in all the benchmarks.

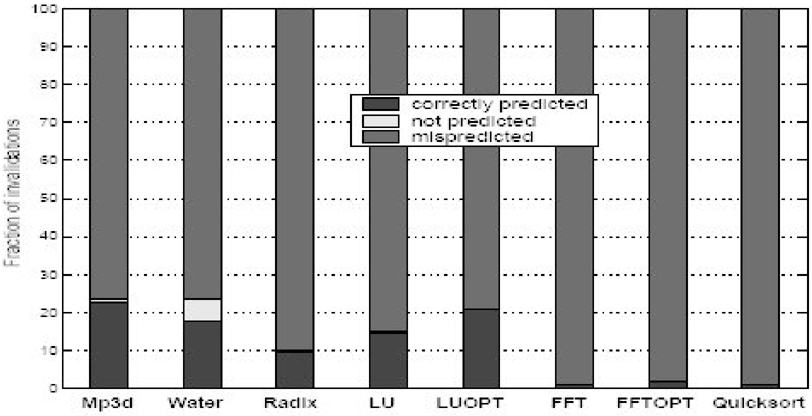
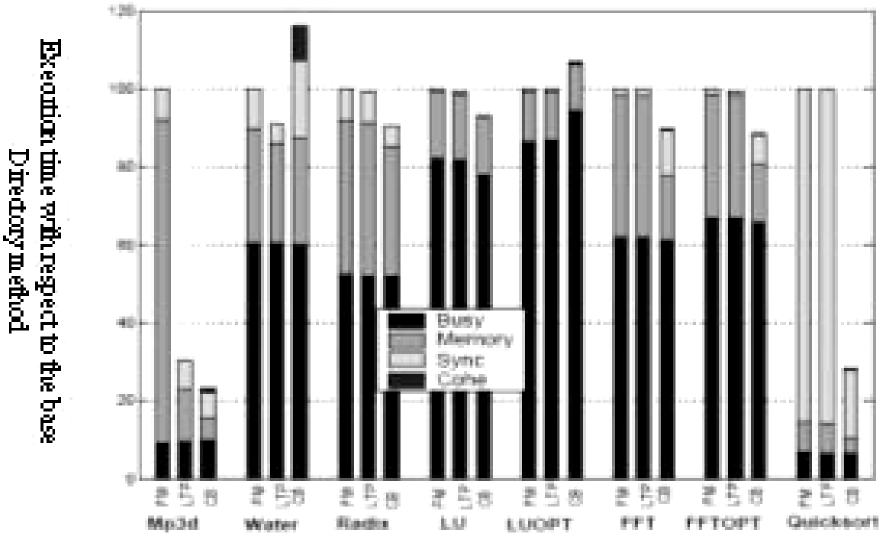


Figure 3. Invalidations with the LTP (12 bit wide 16 history registers, 1K entry two-way associative second level table)

C. Execution time Performance Comparison of various Coherence Mechanisms



.

Figure 4. Performance Comparison of Full-Map Directory (FM), LTP and CB based approaches (LTP configuration: 12 bit wide 16 history registers, 1K entry two-way associative second level table)

Overall execution time performance is shown in figure 4, for the three coherence mechanisms - full map directory denoted by FM, LTP based directory method denoted by LTP and CB based method denoted by CB - in a DSM. The execution times reported are relative to that of the full map directory execution time. The CB structure is of a width of one word plus status bits, and is 8 entry 4-way associative.

The results in figure 4 show that the LTP improves performance in certain applications, which is an evidence to say that self-invalidation is certainly a step to achieve improved performance. However the cost-benefit of the LTP approach seems not high. Bursty nature and long queuing of self-invalidations can offset the performance achievable by the LTP, resulting in poor performance.

The CB approach performs significantly better than the other two methods in general. The performance benefit is mainly because of the early and local coherence transactions that are processor-directed. Water and LUOPT cases are different. Due to its O (n^2) complexity, Water encounters heavy number of shared boundaries, for which larger CB size is required. The sharing pattern of LUOPT requires a CB configuration different from the present one to improve the performance. Both these results for Water and LUOPT were found in separate experiments reported in the thesis [8].

IV. CONCLUSION

With the emerging class of multiprocessors that include multi-core and massively parallel processor systems, cache coherence is a major system mechanism that contributes to performance and power. Cost-effective approaches such as Coherence Buffer (CB) based processor-directed auto-invalidation approach [3][6] are a must for the current day parallel systems that invariably use shared memory units deployed in distributed architecture. In this paper performance study of a directory based approach with self-invalidation using a last touch predictor (LTP) is presented and compared with full map 3 hop conventional directory method, as well as the CB approach. While the processor-directed CB approach shows significant performance benefit, the cost-benefit of self-invalidation based LTP approach does not seem to be good. The CB approach, which is a promising cost-effective approach for cache coherence, can be extended easily for providing cache coherence is multi-core systems and other bus based multiprocessor systems.

ACKNOWLEDGMENT

This work was carried out as part of the post doctoral research work by the first author at CAD Lab, SERC, IISc. The author is grateful for the support provided by the Institute and the research guide. The author is thankful to the present institution of employment, i.e., NMIT, for the encouragement and support received.

REFERENCES

[1] J. Hennessy and D. Patterson, “Computer Architecture: A Quantitative Approach”, 4th Edition, M.K. Publishers, 2006.

[2] Culler and Singh, “Parallel Computer Architecture: A Hardware Software Approach”, M.K. Publishers, 1998.

[3] H. Sarojadevi, S. K. Nandy, and S. Balakrishnan, “Enforcing Cache Coherence at Data Sharing Boundaries without Global Control: A Hardware-Software Approach”, EuroPar 2002.

[4] A.C. Lai and B. Falsafi, “Selective, Accurate, and Timely self-invalidation Using Last-Touch Prediction”, ISCA 2000.

[5] S. Adve, V. Pai and P. Ranganathan, “Recent Advances in Memory Consistency Models for Hardware Shared Memory Systems”, Proceedings of the IEEE, 1999, 87(3): pp. 445-455.

[6] H. Sarojadevi, S. K. Nandy and S. Balakrishnan, “On the Correctness of Program Execution when Cache Coherence is maintained Locally at Data-Sharing Boundaries in Distributed Shared Memory Multiprocessors”, International Journal of Parallel Programming, 2004, vol.32(no.5): pp. 415-446.

[7] C.J. Hughes, V. Pai, P. Ranganathan and S. Adve, “RSIM: Simulating Shared Memory Multiprocessors with ILP Processors”, IEEE Computer, 35(2): 40-49, 2002

[8] H. Sarojadevi, "Extending Architectural Support for Imposing Early and Local Cache Coherence in Distributed Shared-Memory Multiprocessors", PhD thesis / Technical report (CAD Lab), IISc, 2002

AUTHORS PROFILE

Dr. H. Sarojadevi (corresonding author) is presently working as Professor in the department of Computer Science & Engineering, Nitte Meenakshi Institute of Technology (NMIT), Bangalore. She is a PhD graduate from the Indian Institute of Science. She has more than 18 publications most of them are at the International level. She has 14 years of teaching experience, 16 years of research experience, 4 years of industry experience. She is guiding 6 PhD candidates at present, and produced several Master level projects as well as BE Projects.

Dr. S. K. Nandy is Professor in the Dept. of Super Computer Education and Research Centre, IISc, Bangalore. He is also convenor of the CAD Lab. He has authored several papers and carries out funded projects from Internationally reputed organizations. He has produced more than five PhDs and several MSc graduates from IISc.