Leveraging On-Chip Networks for Efficient Prediction on Multicore Coherence

Libo Huang
State Key Laboratory of High Performance Computing and School of Computer
National University of Defense Technology, Changsha 400073, China
Email: libohuang@nudt.edu.cn

***Abstract*—Coherent data prediction is introduced as a promising architectural technique for reducing cache-to-cache accesses in directory protocol. However, limited on-chip resources cause the accuracy of current prediction to be generally low. Low accuracy would result in a large number of unnecessary or incorrect predictions, which would consequently generate excessive network traffic. This leads to large power and performance overhead for coherent memory access. This paper proposes an early abort mechanism (EBT) that leverages NoC design to reduce the negative effect of wrong prediction operations, thus facilitating overall performance improvement and traffic reduction. Using detailed full-system simulations, we conclude that EBT provides a cost-effective solution for designing efficient multicore processors. To the best of our knowledge, this study is the first to leverage on-chip network for the prediction optimization on multicore coherence.**

I. Introduction

One of the biggest challenges facing multicore processors is the maintenance of cache coherence across all of cores to support the shared memory programming paradigm. In current large scale network-on-chip (NoC)-based multicore architectures, directory-based coherence protocols are preferred over snoop based protocols. However, directory protocols achieve scalability at the cost of placing the directory access in the critical path of coherence cache misses. Furthermore, the exponential increase in the number of cores will result in a commensurate increase in the hop count of remote data access, thus restricting overall performance severely.

Coherent data prediction (DP) is introduced as an effective method to reduce or eliminate cache-to-cache delay. This process speculatively fetches data to the local buffer (or cache), which is believed to be consumed shortly thereafter. Thus, DP can convert 3-hop misses into 0-hop misses when the prediction is correct. A large number of solutions have been proposed in multiprocessor systems, which includes producer-initiated predictor [1], Cosmos coherence message predictor [14], pattern-based memory sharing prediction scheme [13], and the SORDS predictor [16]. Axiras et al. [11] proposed instruction-based predictors as an alternative to address-based predictors to move the shared data close to the consumers as soon as possible.

Prior prediction methods that require a large amount of resources are unsuitable for on-chip usage. A number of recent works have integrated a coherence predictor on multicores such as data prediction [12], and owner prediction [2], [10]. All these solutions focus on the cost-effective implementation of existing predictors without on-chip network optimization. Since the on-chip hardware resource is rather limited for predictors used in multiprocessor systems, the accuracy of current simple on-chip predictors is generally low. Therefore, such predictors would make a large amount of unnecessary or incorrect predictions, generating excessive network traffic. This condition results in large power and performance overhead for coherent memory access. To mitigate the negative effect of this prediction problem, we attempt to optimize the coherence prediction through on-chip network optimization. There are already some works for the study of network optimization for cache coherence. Enright-Jerger et al. proposed Virtual Tree Coherence [7], Eisley et al. proposed an in-network coherence [6], and our previous work introduced an NoC design for a hierarchical cache coherence using traffic locality [8].

This paper presents early abort mechanism (EBT), a novel cache coherence design that advocates a practical in-network optimization for efficient DP. EBT attempts to use the network to abort fetching mispredicted data as early as possible. That is, the NoC router would abort the wrong data prediction as long as they travel the same path. Using detailed full system simulations on a 16-core processor, we find that EBT is capable of achieving approximately 15% traffic reduction and approximately 14% energy-delay product (EDP) improvement on average compared with the conventional DP design when normalized to baseline directory protocol.

II. Motivation

DP can convert 3-hop misses to local misses, effectively eliminating the impact of remote memory latency. To integrate prediction techniques into NoC-based multicore architectures, the proposed scheme would ideally satisfy the goals of high performance and low cost. The two goals are possibly contradicting, and simultaneously achieving both would be challenging. A wrong prediction would not only affect the performance, but also increase the network traffic. Such possibility increase of network and buffer contention would result in performance degradation. However, more predicted data can improve the DP hit ratio. Thus, this paper attempts to use the in-network optimization technique to abort fetching such data as early as possible. Figure 1(a) illustrates the problem of wrong DP. Given that a time gap exists between the producer (P3) sending the predicted data and the consumer (P1) finding the wrong prediction, the data will continue to be sent out

This work is supported in part by NSFC (No. 61070037, No. 61025009, No. 61103016, and No. 61170045), HNNSF (No. 12JJ4070), RFDP (No. 20114307120010), and CHB (No. 2013ZX01028-001-002).

978-3-9815370-2-4/DATE14/©2014 EDAA

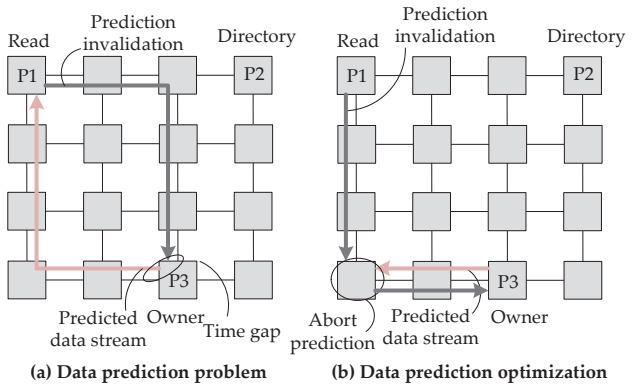

Fig. 1. Data prediction optimization

until the prediction abort message has arrived at the producer. Furthermore, the useless data, which have been sent out, will continue to be sent to the consumer.

Figure 1(b) illustrates the in-network optimization for wrong DP, where the prediction abort message is sent through the same path of prediction data stream. When any router has found the wrong prediction data stream, this router will abort such predicted data transmission. Moreover, the router will continue to abort other predicted data transmissions along the path until the data stream reaches the producer. Given that the size of the predicted data message is evidently larger than that of the coherence control message, the former can save network traffic significantly.

III. EBT Approach

A. Overview

Figure 2 provides an overview of the EBT approach. The main function of EBT is to perform in-network optimization to reduce the negative effect of data misprediction. The implementation of EBT can be divided into two parts. The first part is related to the message generation of DP abort. We place it into an L1 controller. The second part is related to the message abort of data fetching, the logic of which can be implemented in the NoC router for detection and abort.

We adopt a simple, low-cost, stride-based data predictor. Previous work has shown that stride-based prediction can be effective for data stream sharing [15]. This method is also simple because it involves few entries of address history buffer to be used for prediction. This predictor implements stream fetching with non-unit both in ascending and descending streams strides. For instance, after observing misses for addresses A, A+4, A+8, the predictor issues prefetch for address A+12, A+16 and so on. To avoid unnecessary fetches, the address prediction only fetches the stream that already has two elements for unit stride and three elements for nonunit stride.

B. EBT Abort Generator

The EBT abort generator is used to detect useless DPs and send the abort requests to the owner core. Accurate detection of useless DPs is difficult because we do not know whether data could be accessed at a later time. This paper proposes an pessimistic method for implementing the EBT abort generator.

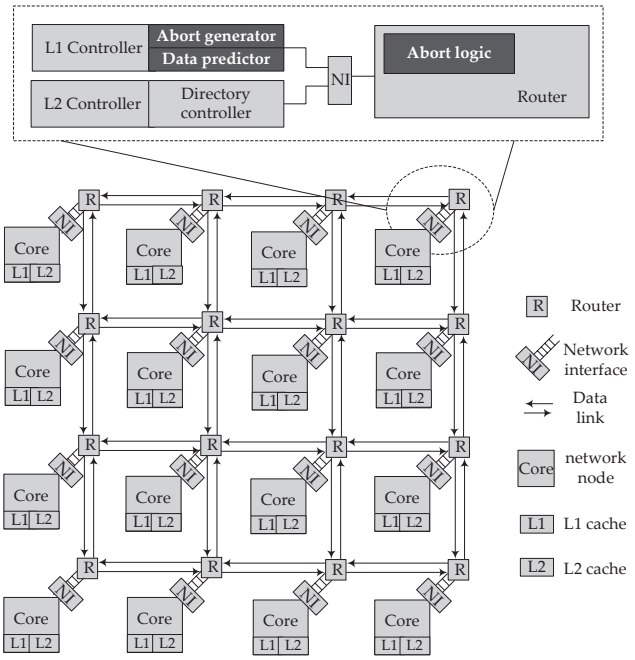

Fig. 2. EBT architecture overview for 16-core configuration

On one hand, the DP is built on the data locality and access pattern of remote data accesses. If the subsequent remote accesses violate these two conditions, we can regard the data predicted on the access pattern as useless data. On the other hand, the transmission overhead of the abort messages is small. Thus, significant benefit can be obtained if the fetching data is really useless.

We formalize the following two conditions to trigger abort mechanism: (1) spatial locality violation: none of the subsequent N remote access are in the range of DP locations. To decrease the number of prefeching data, N should be small. In our experiments, we set N as two. That is, only one remote data are allowed to violate current recognized access patterns. (2) temporal locality violation: all subsequent remote accesses are mispredicted before the configured time. In our experiments, we set it as the time that the next predicted cache line arrives at the local core. That is, if the predicated data can not be consumed timely, we just abort the subsequent data fetching.

C. EBT NoC Router

Figure 3 illustrates the EBT abort logic along with the conventional NoC design. The prediction abort logic is responsible for receiving abort requests and subsequently performing the actual abortion operation. When a packet arrives at the input channel, such packet would be decoded to verify whether it is a prediction abort packet. This process is achieved by verifying the prediction abort bit (PA). If the prediction abort operation is confirmed, the logic would verify whether the subsequent packet is the target DP stream. The logic performs the corresponding abortion operation, which simply discards the operations. The logic then generates the invalidation packet to the directory to maintain the coherence protocol. Until the prediction abort packet is sent to the next router, the logic continues this abortion operation.

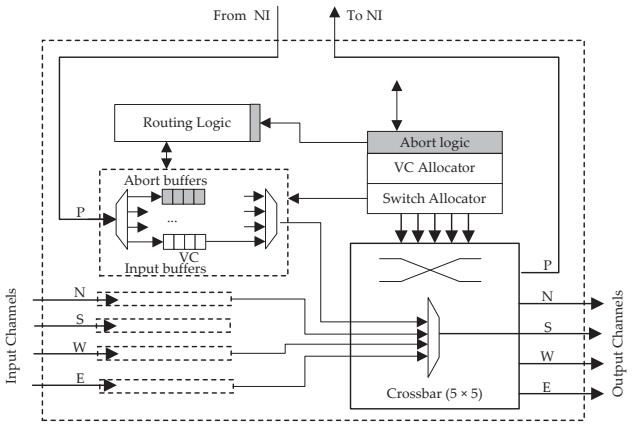

Fig. 3. Block diagram of EBT router

An important function of in-network abortion is the transmission of the prediction abort packet through the reversed path of the data prediction stream transmission, as shown in Figure 1(a). This process requires some modifications in network routing designs. For the deterministic and non-deterministic routing, we have different different schemes, which will be described in the following text.

1) Abort for deterministic routing: For deterministic routing, it is relatively simple to abort the prediction data transmission since such deterministic routing can be calculated directly. Taking dimension-ordered XY routing as an example, the prediction abort packet and corresponding prediction stream packet would not transferred through the same path in an opposite direction. Thus, we employ a method in which the prediction abort packet is routed in a YX ordered manner. This method requires the prediction abort logic to inform the routing logic to select the XY or YX routing. In the experiment, we consider XY routing as the target routing.

2) Abort for non-deterministic routing: For non-deterministic routing or other topology, a general abort approach is to record the direction of each hop, and then the abort message would be transmitted in the opposite direction according to the record. For one hundred cores, only 18-bit flit data may be added to record such information. This method can be applied to the most routing situations when the transmission hops is smaller than the bit length used to record the path.

D. Coherence Protocol

Compared with the baseline directory protocol, the main difference of the EBT protocol is that it updates the directory information when DP or fetching abort is completed. Predicted fetching data would be regarded as any other sharer to take the same coherence operations such as the set corresponding to the directory sharing bits. When any other core wants to update the predicted fetching data, such core should invalidate the fetched data. If the useless predicted data are evicted from buffer or aborted from the NoC, the directory sharing bit has to be cleared. If the predicted data is read by the L1 cache, the core becomes the "real" sharer of the data.

To verify the proposed coherence protocol, we have performed extensive simulation on the protocol. We obtained correct results for all tested benchmarks. To enhance confidence in our protocol, we use the Murphi Model Checker [5], a well-known checker, to conduct an explicit-state model checking under a small configuration size with four cores. We performed the exhaustive reachability analysis and found no coherence violation in the experiments.

TABLE I. SIMULATION ENVIRONMENT CONFIGURATIONS

Designs	Parameter	Measure
Basic router design	Topology	2D Mesh 4x4
	Basic routing	Dimension-ordered XY
	Number of ports	5 ports
	VCs per port	4 VCs
	Buffers per VC	16 buffers
	Channel width/flit size	128 bits
EBT	Packet size	8 flits
	prediction fetch size	$Sp = 4$
	address history buffer	8 entries
	prediction buffer	16 entries

IV. EVALUATION

A. Methodology

To evaluate the proposed EBT design, we use the M5 architecture-level simulator [3] as the baseline simulator for the performance modeling of our target multicore design. We model our target multicore systems with the Alpha instruction set architecture (ISA), which is the most stable ISA supported in M5. Each core is modeled with a 2-way 16KB L1 instruction cache and 2-way 32-KB L1 data cache, and all cores shares a 16-way 1MB L2 cache. Given that the original M5 simulator only possesses shared bus communication implementation, we integrate a cycle-accurate NoC simulator to support communication over NoC design. We have implemented the hardware architecture and the associated coherence protocol, which models a detailed pipeline structure for NoC router. The address space is statically divided among all nodes, using the least significant bits of the tag, to determine which node is the home (directory) node for a given address. We also integrate Orion [9] to estimate NoC energy and Cacti [4] to estimate cache energy, thus obtaining the EDP metric for cache-NoC system.

The benchmarks [Barnes (128K bodies), Cholesky(tk29.O), FFT(256K pts), FMM(32K particles), LU-con(1024×1024), Watersp(4K molecules), Water-nsq(4K molecules), Radix(8M Keys), Volrend (head), and Raytrace(car)] used in our simulations are chosen from the SPLASH-2 benchmark suite [17], which represents a variety of important scientific and graphic computations with different communication requirements.

B. Results

1) Network Traffic Reduction: In this experiment, we identify the network traffic as the number of bytes transmitted through the each NoC router. Figure 4 illustrates the network traffic reduction compared with the baseline directory and conventional DP protocols. Each bar plots the number of bytes transmitted through the interconnection network normalized with respect to the basic directory protocol. We can see that the conventional DP (as listed in the figure) design would increase the network traffic by approximately 22% increment on average. This condition is caused by the added unnecessary

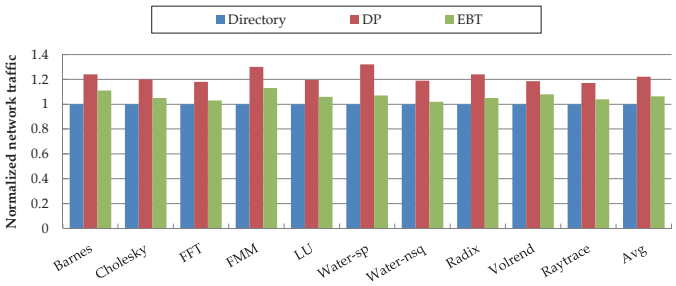

Fig. 4. Network traffic analysis results

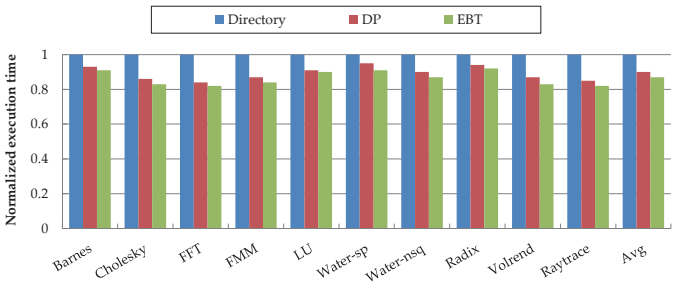

Fig. 5. Execution time analysis results

data fetching. Such added network traffic of DP not only increases power consumption, but also the network congestion, thus reducing performance benefits. The EBT design reduces the negative effect of this problem because it significantly decreases the misprediction traffic, thus resulting in overall traffic reduction only an approximately 7% increase on average compared with baseline directory protocol.

2) Performance Improvement: Figure 5 shows the performance results of different coherence designs in terms of execution time. The primary effect of prediction in cache coherence is the reduction of the cache-tocache transfer latency of coherence miss, which would result in application speedup. Compared with the baseline directory protocol, DP reduces execution time by approximately 10% on average because of its remote access elimination. EBT also has a slight performance advantage over DP design of approximately 3% execution time reduction on average. These results successfully demonstrate that with the same prediction technique, EBT would result in performance improvements.

3) EDP Improvement: An important metric in current processor design evaluation is the EDP. As it was for ILP architectures, power efficiency is currently the key factor for any novel computer architecture proposal. This case is also true when prediction is used for unnecessary prediction. Figure 6 shows the EDP results for different coherence designs. We can see that conventional DP design even increase the EDP by an approximately 7% on average. This result makes DP less appealing for utilization in an efficient processor design. EBT changes this situation to achieve the best EDP results (approximately 7% reduction on average compared with directory). These results successfully demonstrate that with the same prediction technique, EBT optimizations would result in performance and EDP improvements.

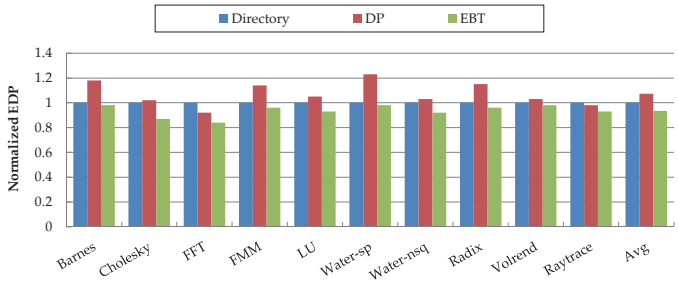

Fig. 6. EDP analysis results

V. Conclusion

This paper proposes EBT, a novel multicore coherence design that supports an early abort mechanism for mispredicted data fetching. Extensive evaluation results demonstrates that EBT can reduce the network traffic of DP design by an average of 15% and improve the EDP of DP by an average of 14% for 16-core CMP when normalized to the baseline protocol. For the future, studies can focus on further on-chip network optimization for improving prediction accuracy.

References

[1] H. Abdel-Shafi, J. Hall, S. V. Adve, and V. S. Adve, "An evaluation of fine-grain producer-initiated communication in cache-coherent multiprocessors," ser. HPCA '97, 1997.

[2] E. Atoofian and A. Baniasadi, "A power-aware prediction-based cache coherence protocol for chip multiprocessors," in *IPDPS*, 2007, pp. 1–8.

[3] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The m5 simulator: Modeling networked systems," *IEEE Micro*, vol. 26, pp. 52–60, July 2006.

[4] CACTI, "http://www.cs.utah.edu/∼rajeev/cacti6/."

[5] D. L. Dill, "The murphi verification system," ser. CAV '96. London, UK, UK: Springer-Verlag, 1996, pp. 390–393.

[6] N. Eisley, L.-S. Peh, and L. Shang, "In-network cache coherence," ser. MICRO 39, 2006, pp. 321–332.

[7] N. D. Enright Jerger, L.-S. Peh, and M. H. Lipasti, "Virtual tree coherence: Leveraging regions and in-network multicast trees for scalable cache coherence," ser. MICRO 41, 2008, pp. 35–46.

[8] L. Huang, Z. Wang, and N. Xiao, "An optimized multicore cache coherence design for exploiting communication locality," ser. GLSVLSI '12. ACM, 2012, pp. 59–62.

[9] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "Orion 2.0: a fast and accurate noc power and area model for early-stage design space exploration," ser. DATE '09, 2009, pp. 423–428.

[10] S. Kaxiras and G. Keramidas, "Sarc coherence: Scaling directory cache coherence in performance and power," *IEEE Micro*, vol. 30, pp. 54–65, September 2010.

[11] S. Kaxiras and C. Young, "Coherence communication prediction in shared-memory multiprocessors," *HPCA '00*, p. 156, 2000.

[12] A. Kayi and T. El-Ghazawi, "An adaptive cache coherence protocol for chip multiprocessors," ser. IFMT '10. ACM, 2010, pp. 4:1–4:10.

[13] A.-C. Lai and B. Falsafi, "Memory sharing predictor: the key to a speculative coherent dsm," ser. ISCA '99, 1999, pp. 172–183.

[14] S. S. Mukherjee and M. D. Hill, "Using prediction to accelerate coherence protocols," ser. ISCA '98, 1998, pp. 179–190.

[15] S. Palacharla and R. E. Kessler, "Evaluating stream buffers as a secondary cache replacement," ser. ISCA '94, 1994, pp. 24–33.

[16] T. F. Wenisch, S. Somogyi, N. Hardavellas, J. Kim, C. Gniady, A. Ailamaki, and B. Falsafi, "Store-ordered streaming of shared memory," ser. PACT '05, 2005, pp. 75–86.

[17] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: Characterization and methodological considerations," ser. ISCA '95, 1995, pp. 24–36.