Application-aware Prefetch Prioritization in On-chip Networks

Nachiappan Chidambaram Nachiappan[†] Asit K. Mishra[§] Mahmut Kandemir[†]
Anand Sivasubramaniam[†] Onur Mutlu[‡] Chita R. Das[†]
[†]The Pennsylvania State University [§]Intel Corp. [‡] Carnegie Mellon University
{nachi, kandemir, anand, das}@cse.psu.edu {asit.k.mishra@intel.com} {onur@cmu.edu}

ABSTRACT

Data prefetching is an effective technique for hiding memory latency. When issued prefetches are inaccurate, performance can degrade. Prior research provided solutions to deal with inaccurate prefetches at the cache and memory levels, but not in the interconnect of a large-scale multiprocessor system. This work introduces *application-aware* prefetch prioritization techniques to mitigate the negative effects of prefetching in a network-on-chip (NoC) based multicore system. The idea is to rank prefetches from different applications based on their potential utility for the application and propensity to cause interference to other applications. Our evaluation shows that this approach provides significant performance improvements over a baseline that does not distinguish between prefetches from different applications.

Categories and Subject Descriptors: C.1.2[Computer Systems Organization] Multiprocessors; Interconnection architectures; **General Terms:** Design, Performance **Keywords:** Interconnect, Multicore, Prefetching

1. INTRODUCTION

Memory access latency is a major limiter of computer system performance. A widely-used technique to tolerate this latency is prefetching, a method that predicts the memory accesses that will be made by a processor and fetches the data before it is requested by the processor. This technique has been shown to be very effective when prefetch accuracy, coverage and timeliness are reasonably high [10]. Unfortunately, not all prefetch requests are accurate (or, useful, i.e., will be needed by the processor). An inaccurate prefetch request wastes system resources (such as memory and interconnect bandwidth, cache space) and degrades performance by delaying demand or useful prefetch requests. As recent work has shown, uncontrolled aggressive prefetching can be ineffective in bandwidth-limited and cache-space-limited multi-core systems [8, 3, 4] because it causes significant interference between cores in shared caches and main memory.

A critical shared resource in scalable multi-core systems is the on-chip interconnect/network (NoC), which is the communication substrate that connects the cores and caches, carrying private-cache-miss traffic and coherence traffic (requests and responses). Inaccurate prefetch requests waste NoC resources (bandwidth, buffers, etc) and delay other requests while traversing the NoC. In fact, we find that aggressive prefetching can cause significant congestion in the NoC and the resulting uncontrolled interference between applications significantly degrades overall system performance in some workloads.

While previous works [8, 7, 10, 3, 4] developed techniques to alleviate the negative effects of aggressive prefetching in caches and memory controllers via intelligent prioritization or prefetcher throttling mechanisms, little work considered such effects in the NoC. Our goal in this work is to examine the effects of aggressive prefetching in NoC-based multi-core systems and develop techniques to mitigate the negative effects of prefetching in the NoC. To this end, we propose an application-aware prefetch prioritization mechanism for NoC routers that differentiates between the prefetches from different applications and prioritizes prefetches from those applications where prefetching 1) is more likely to improve the performance of the application, 2) is likely to not cause significant interference to other applications.

PACT'12, September 19–23, 2012, Minneapolis, Minnesota, USA.
ACM 978-1-4503-1182-3/12/09.

2. MOTIVATION

Aggressive prefetching in an NoC-based system can cause network congestion because the prefetcher can inject a significant number of inaccurate prefetches into the network. These inaccurate prefetches can delay not only demand requests but also useful prefetch requests. To investigate the effect of aggressive prefetching in an NoC-based multicore system, we implemented a version of the aggressive, state-of-the-art stream prefetcher described in [10] on a 64-core 2D mesh based NoC.[1]

Figure 1(a), (b), (c) show respectively the prefetch accuracy, network latency increase due to prefetching, and IPC delta with the prefetcher over no prefetching across a set of multiprogrammed workloads. Each workload consists of 64 concurrently-executed copies of a SPEC 2006 benchmark. Results are averaged across all SPEC workloads, but the figure shows a subset. Overall, with an average prefetch accuracy of 43%, we see an average network latency increase of 170% due to a 45% increase in network traffic (not shown). Although the prefetcher improves average performance, we find it degrades performance by more than 10% on 8 of the 26 evaluated workloads.

We find two major problems contribute to the performance degradation: 1) prefetch requests are sometimes prioritized over demands in NoC routers (our baseline does not distinguish between them), 2) useless or less beneficial prefetch requests for one application are sometimes prioritized over useful or more beneficial prefetch requests for another in NoC routers.

Previous works [8, 7, 4] tackled the first problem: they developed policies to adaptively prioritize between prefetch and demand requests in memory controllers and caches such that likely-inaccurate prefetch requests do not get prioritized over demand requests. In our evaluations, we find that prioritizing demands over prefetches in the network (as done in [9]) provides better performance than treating prefetches and demands equally (shown as Demand-first in Figure 2). Adapting previous dynamic prioritization policies [8, 7, 4] to the NoC context can provide even better performance, but we leave this to future work as our goal is to focus on the second problem above.

To our knowledge, previous works did not tackle the second problem: different applications' prefetch requests have different impact on application performance and system performance.

[1]Our baseline uses the standard XY-routing algorithm, has 4 memory controllers located at the corners, and does not differentiate between prefetch and demand requests in the routers.

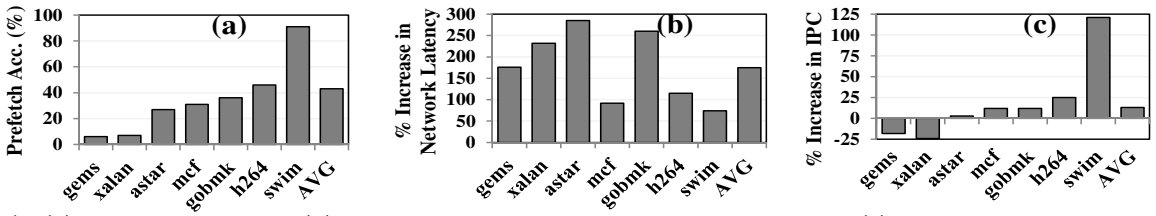

Figure 1: (a) Prefetch accuracy, (b) network latency increase due to prefetching, (c) IPC delta due to prefetching.

The major question we aim to answer is: *how should prefetch requests of different applications be prioritized in the network routers*?

3. MECHANISM

Ideally, one would like to prioritize prefetches of those applications that benefit the most from prefetching and do not cause harm to other applications due to prefetching. Again, ideally, one would like to deprioritize the prefetches of those applications that benefit the least from prefetching but cause significant performance degradation to other applications due to their prefetches. We observe that the benefit an application gains from prefetching and the harm it causes due to prefetching can be approximated by observing two metrics for the application: prefetch accuracy and prefetch count.

Our policy is based on two insights. First, an application that has high prefetch accuracy is more likely to benefit from prefetching because its prefetches are more likely to be useful. Second, an application that injects fewer prefetches into the network is less likely to harm other applications because its prefetches are less likely to interfere with other applications' requests.

Using these two insights, we develop a mechanism that categorizes applications into four categories, either statically or dynamically, based on whether the application's 1) prefetch accuracy is high or low, and 2) prefetch count is high or low. Prefetches from those applications with high prefetch accuracy but low prefetch count are given the highest priority because such applications are the most likely to benefit from prefetching and the least likely to harm others due to prefetching. Conversely, prefetches from those applications with low prefetch accuracy and high prefetch count are given the least priority. In-between these two extremes, applications with high accuracy and high prefetch count are given priority over those with low accuracy and low prefetch count. If two applications happen to be in the same category, their prefetches are prioritized using the default prioritization policy of the router (e.g., age-based or round-robin).

Static vs. Dynamic: An application's prefetch accuracy and prefetch count category can be estimated statically or dynamically. In the static version of our policy, a profile run determines in which of the four categories an application falls. In the dynamic version, hardware collects these metrics periodically and at the end of a period (i.e., time quantum) assigns each application to one of the four categories. For the next quantum, the network routers use the assigned category of an application to appropriately prioritize prefetches from that application.

Results: Figure 2 shows the performance improvement of our static and dynamic application-aware prefetch prioritization mechanisms (AppAware-S and AppAware-D, respectively) over the baseline system that uses round-robin prioritization in routers, as described in [1]. In our mechanisms, demands are always prioritized over prefetches. For comparison, we also show the performance improvement with two different policies: 1) one that only prioritizes demands over prefetches but employs round-robin prioritization across different prefetches (Demand-first), and 2) an unimplementable policy in which prefetch requests never contend with each other or with demands, which provides an upper bound on what is achievable with our techniques. The results are averaged across 12 heterogeneous multiprogrammed workloads, where each workload consists of a combination of different SPEC benchmarks running together. We show five representative workloads in the figure.

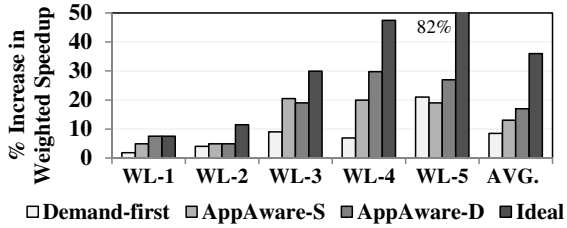

Figure 2: Performance of app-aware prefetch prioritization.

Figure 2 leads to three major conclusions: 1) Application-aware prioritization among prefetches, static or dynamic, provides significant performance improvement over policies that do not distinguish between prefetches, 2) Our dynamic policy performs better than the static one as it can adapt to changes in application and prefetcher phase behavior (improving performance by 9% over Demand-first), 3) Although our mechanism bridges more than one third of the gap between Demand-first and Ideal, there is still significant performance to be gained by eliminating the negative effects of prefetching in the NoC.

4. CONCLUSIONS

We showed differentially prioritizing between prefetches from different applications in an NoC based system significantly improves system performance. Our future work includes analyzing our techniques and combining them with other application-aware prioritization techniques in NoC routers (e.g., [1, 5, 2, 6]).

References

[1] R. Das et al. Application-Aware Prioritization Mechanisms for On-Chip Networks. In *MICRO-42*, 2009.

[2] R. Das et al. Aergia: Exploiting packet latency slack in on-chip networks. In *ISCA-37*, 2010.

[3] E. Ebrahimi et al. Coordinated control of multiple prefetchers in multi-core systems. In *MICRO-42*, 2009.

[4] E. Ebrahimi et al. Prefetch-aware shared-resource management for multi-core systems. In *ISCA-38*, 2011.

[5] B. Grot et al. Preemptive Virtual Clock: A Flexible, Efficient, and Cost-effective QOS Scheme for Networks-on-a-Chip. In *MICRO-42*, 2009.

[6] B. Grot et al. A QoS-Enabled On-Die Interconnect Fabric for Kilo-Node Chips. *IEEE Micro*, May/June 2012.

[7] C. J. Lee et al. Improving memory bank-level parallelism in the presence of prefetching. In *MICRO-42*, 2009.

[8] C. J. Lee et al. Prefetch-aware memory controllers. *IEEE Transactions on Computers*, 2011.

[9] J. Lee et al. Exploiting mutual awareness between prefetchers and on-chip networks in multi-cores. In *Poster Session, PACT*, 2011.

[10] S. Srinath et al. Feedback directed prefetching: Improving the performance and bandwidth-efficiency of hardware prefetchers. In *HPCA-13*, 2007.