Author Retrospective for A NUCA Substrate for Flexible CMP Cache Sharing

Jaehyuk Huh
KAIST

Changkyu Kim
Google

Hazim Shafi
Samsung

Lixin Zhang
Institute of Computing Technology
Chinese Academy of Sciences

Doug Burger
Microsoft Research

Stephen W. Keckler
NVIDIA and UT-Austin

ABSTRACT

In 2005, as chip multiprocessors started to appear widely, it became possible for the on-chip cores to share the last-level cache. At the time, architects either considered the last-level cache to be divided into per-core private segments, or wholly shared. The shared cache utilized the capacity more efficiency but suffered from high, uniform latencies. This paper proposed a new direction: allowing the caches to be non-uniform, with a varying number of processors sharing each section of the cache. Sharing degree, the number of cores sharing a last-level cache, determines the level of replication in on-chip caches and also affects the capacity and latency for each shared cache. Building on our previous work that introduced non-uniform cache architectures (NUCA), this study explored the design space for shared multi-core caches, focusing on the effect of sharing degree. Our observation of a per-application optimal sharing degree led to a static NUCA design with a reconfigurable sharing degree. This work in multicore NUCA cache architectures has been influential in contemporary systems, including the level-3 cache in the IBM Power 7 and Power 8 processors.

Original paper: http://dx.doi.org/10.1145/1088149.1088154

Categories and Subject Descriptors: C.1.2 [**Processor Architectures**]: Multiple Data Stream Architectures (Multiprocessors); B.3.2 [**Memory Structures**]Design Styles – Shared memory

Keywords: NUCA design, sharing degree, multicore processors

1. BACKGROUND

In the early 2000s, as power limitations hampered single-core performance improvements, the industry shifted *en masse* to chip multiprocessors (CMPs). The on-chip integration of multiple cores and caches expanded the possible space of cache hierarchy designs. Instead of the private, per-core caches used in traditional multi-processors, CMPs enabled shared cache designs. These shared caches reduce cache misses significantly by reducing replication of cache blocks and by resolving capacity imbalances among working sets accessed by multiple cores sharing a cache.

Shared multicore caches, however, also incurred large access latencies. The increased capacity and physical distance from some of the cores led to long hit times, which could reduce or eliminate the benefit of the reduced misses. To mitigate increasing cache latencies, previous work proposed Non-Uniform Cache Architectures (NUCA) which redesigned cache structures to account for variable cache bank access latency [10]. NUCA designs consist of a set of small cache banks connected via an interconnection network to allow hit latency to vary depending on the location of accessed cache blocks. The original NUCA work focused on the last level cache for uniprocessors. The latency problem targeted by the NUCA work is typically worse for shared CMP caches, as the capacity needs to be larger to support multiple working sets.

At that time, the caching efficiency/latency tradeoff was well understood, but the consensus at the time was that caches should either be private or shared by all cores, and that caches would have a uniform access latency. Adding NUCA capabilities brought down the average latencies of large shared caches, making them faster, and thus enabling larger sharing degrees. The study in this paper focused on the tension between cache sharing and latency, using NUCA caches as a way to better balance these two constraints [8].

Other contemporaneous work took a different approach to mitigate the long latencies of large, last-level shared caches. Rather than adjusting the sharing degree, this other approach moved individual cache blocks closer to the requester, either by selectively replicating the blocks, or by migrating them to a closer portion of the cache [4, 13, 2]. This paper's approach instead asked the question of how may cores should share a cache to maximize the caching efficiency, while preserving low cache hit latency. The answer to this question determines how many replications of cache blocks can exist in a processor, how large a shared cache segment should be, and how fast a block access takes in the shared cache.

Our results showed that by adjusting the sharing degree and using a NUCA substrate, a better balance between hit latencies and misses can be struck. We also found that choosing the sharing degree on an application by applica-

ICS 25th Anniversary Volume. 2014
ACM 978-1-4503-2840-1/14/06.
http://dx.doi.org/10.1145/2591635.2591667.

tion basis, or even a cache line by line bases, can improve the CMP performance even further than a single ideal sharing degree. While our paper did not propose a mechanism to identify these ideal sharing degrees for individual applications or individual lines, our conclusion that these were promising approaches have been validated by other researchers' follow-on work.

2. SHARING DEGREE: IDENTIFYING INHERENT TENSIONS

Unlike previous shared cache studies that assumed the entire last-level cache of a CMP is composed of either per-core private caches or a single per-chip shared cache, we generalized the design space into a range of degrees of sharing, including the intermediate points where some cache banks were shared by a subset of the processors. This paper defined the term *sharing degree* to denote the number of cores sharing a given partition of the cache in a CMP design.

The goal of the sharing degree concept was to better balance the tensions inherent in shared cache designs. With a low sharing degree, cache blocks are replicated in more places, reducing caching efficiency and potentially increasing coherence overheads. This additional replication, however, can result in lower latencies, especially when the working sets fit in the local cache partitions and the data are not often written. With a high sharing degree, conversely, the replication is reduced, thus using the cache space more effectively. Also, as more cores share a cache, the chance to share the capacity increases, balancing dynamic shifts in the required capacity among cores. However, the access latencies of shared caches increase with higher sharing degrees.

To empirically identify the tensions caused by the degree of replication, we started from a start-of-the-art cache design for reducing cache hit latencies. Based on NUCA caches, we simulated and evaluated our target processors with several different sharing degrees. Our initial goal was to understand the behaviors of parallel applications with different sharing degrees and hit latencies determined by cache capacity. The performance of each application was evaluated in several different designs with a static sharing degree. The study included three different NUCA designs from static mapping to fully dynamic mapping policies to understand the effect of sharing degree, when dynamic migration within a shared cache is enabled or not.

With static NUCA, a cache block can be located in a fixed position in a shared cache, but in a dynamic NUCA policy, a cache block can be migrated toward a frequently accessing core by block migration mechanism. In single-core NUCA designs, the dynamic NUCA policies exhibited better performance by reducing hit latencies. In our study, we explored both static and dynamic NUCA cache policies to understand their effect on optimal sharing degree.

Our evaluation led to somewhat unexpected results, although the results were (of course) obvious after the tension of sharing degree become understood. We found that each application may respond differently to various sharing degrees. Some applications benefit from reduced cache misses due to higher sharing degrees, while others suffer from increased hit latencies without much benefit from miss reduction. The range of optimal sharing degrees across applications led to the next contribution of this paper: a cache architecture that could flexible support different sharing degrees at different times.

3. FLEXIBLE SHARING DEGREE

Our observation of per-application optimal sharing degree led to a cache design supporting flexible sharing degree. The design borrows its base structure from the single-core NUCA design with an interconnected array of cache banks in a multi-core processor. The NUCA substrate supports a reconfigurable sharing degree, to support a different optimal sharing degree for each application. A key component is a central flexible directory and sharing bits embedded in each cache tag. The central directory provides coherence among different cache partitions, sharing bits in the tags to support coherence among L1 caches within a shared cache domain.

The main contribution of this paper (flexible sharing degree) was a very different result than we set out to demonstrate. Initially, we were trying to provide a single large cache that leveraged D-NUCA techniques to permit the sharing efficiency and use migration to address the latency issue. With this goal in mind, we explored the design space with three NUCA substrates: static NUCA, 1D dynamic NUCA, and 2D dynamic NUCA. We were surprised to find that the performance benefit of the dynamic NUCA organizations over static NUCA was relatively small compared to the benefit of dynamic NUCA in uniprocessor caches. The flexible sharing degree ended up being a more important design factor than dynamic data migration within a shared cache region. If an application has a low optimal sharing degree, cache latencies are already relatively short. If an application has a high optimal sharing degree, there may be highly shared data accessed by many cores. For such highly shared data, dynamic migration of blocks was not very effective as most of the cores attempt to migrate the shared blocks closer to them. We concluded that a static NUCA architecture combined with flexible sharing degree provides an effective design with low complexity. This study showed that the proposed dynamic NUCA designs are not effective, despite their performance potential measured in the earlier paper. The limited gains for multi-cores made them clearly not worth the complexity.

Our paper was limited in that it applied the same sharing degree to all cache lines in an application. However, even within an application, different memory regions may require different optimal sharing degrees. In an extension of this paper, we briefly showed the potential of such sharing degree per cache line or memory region of an application [9]. Although the analysis showed potential performance improvements, we were not able to propose a viable mechanism to support per-line or per-region sharing degree. Hardavellas et al. later proposed to use two sharing degrees, *private and shared*, for different memory regions of the same application [7]. That paper proposed to use the TLB and paging mechanism to separate private and shared pages, and to map them differently to the NUCA substrate.

Another limitation of this study is that it focused primarily on the identification of the sharing degree problem and the NUCA mechanism to change the sharing degree, and did not include a deep study of how to find the optimal sharing degree for each application. A range of techniques that drive profiling, dynamic phase detection and reconfiguration can be applied to this problem.

4. IMPACT

This work has had significant influence on the memory system design for some of industry's contemporary multiprocessor architectures. The L3 cache of the IBM POWER 7 processor is a NUCA design for 8 cores with private L1 and L2 caches [11]. The 32MB eDRAM-based L3 consists of eight L3 regions that are shareable across the cores. In POWER 7, non-uniformity appears at two levels of the L3 organization. First, the access latency to an L3 region depends on the distance between it and the requesting processor core. Second, each L3 region is composed of four banks; access latencies to each bank within a region also varies by the distance to a core. The POWER 7 L3 cache has a mechanism to control sharing degree dynamically. The L3 changes the broadcast scope among L3 regions; if the same copy of a cache block is found in another L3 region in the same broadcast scope, the duplicate copies are merged. By changing the broadcast scope across L3 regions, the L3 can adjust the sharing degree. The subsequent POWER 8 processor extended the NUCA design to a 96MB L3 cache [12]. For twelve 12 cores, the L3 is decomposed into twelve 8MB regions, and each region consists of 8 banks with non-uniform access times even within a region.

In addition the industrial influence, this paper also had significant academic influence. Many follow-on papers have continued to explore the design space of hybrid private and shared cache architectures. Chang and Sohi proposed a hybrid shared cache architecture based on private caches cooperating to hold victim cache blocks from other caches [3]. Such cooperative caching improves caching efficiency of private caches by selectively sharing capacity. Reactive NUCA [7] uses two distinct mapping policies, private and shared, for different memory pages of an applications. It uses private mapping for private memory regions to minimize access latencies, while shared data are mapped differently to avoid replication. Beckmann et al. proposed an adaptive replication mechanism for NUCA-based multi-core caches, which can control per-line sharing degree dynamically [1]. Dybdahl et al. investigated cache partitioning schemes for NUCA-based shared caches [6]. Given that dynamic NUCA showed only marginal improvements for shared CMP caches, Cho et al. investigated a static shared NUCA that used the operating system to optimize the placement of pages by OS-level page mapping [5]. Instead of using complex and expensive hardware-based dynamic migration, the OS page allocation policy places pages in caches close to requesting cores. That paper adapted some of our earlier work that used OS-page placement for on-chip uniprocessor memory for multiprocessor caches [14]. Our earlier paper hypothesized in the conclusions that multicore gains would be higher, but missed the opportunity to apply this technique to caches, which was an inspired leap by Cho et al.

5. REFERENCES

[1] B. M. Beckmann, M. R. Marty, and D. A. Wood. ASR: Adaptive Selective Replication for CMP Caches. In *International Symposium on Microarchitecture (MICRO)*, pages 443–454, December 2007.

[2] B. M. Beckmann and D. A. Wood. Managing wire delay in large chip-multiprocessor caches. In *37th International Symposium on Microarchitecture (MICRO)*, December 2004.

[3] J. Chang and G. S. Sohi. Cooperative Caching for Chip Multiprocessors. In *International Symposium on Computer Architecture (ISCA)*, pages 264–276, June 2006.

[4] Z. Chishti, M. D. Powell, and T. N. Vijaykumar. Optimizing replication, communication, and capacity allocation in cmps. In *Proceedings of the 32nd annual international symposium on Computer Architecture*, 2005.

[5] S. Cho and L. Jin. Managing Distributed, Shared L2 Caches Through OS-Level Page Allocation. In *International Symposium on Microarchitecture (MICRO)*, pages 455–468, December 2006.

[6] H. Dybdahl and P. Stenstrom. An Adaptive Shared/Private NUCA Cache Partitioning Scheme for Chip Multiprocessors. In *International Symposium on High-Performance Computer Architecture (HPCA)*, pages 2–12, February 2008.

[7] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Reactive NUCA: Near-optimal Block Placement and Replication in Distributed Caches. In *International Symposium on Computer Architecture (ISCA)*, pages 184–195, June 2009.

[8] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler. A NUCA Substrate for Flexible CMP Cache Sharing. In *Proceedings of International Conference on Supercomputing (ICS)*, pages 31–40, June 2005.

[9] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler. A NUCA Aubstrate for Flexible CMP Cache Sharing. *IEEE Transactions on Parallel and Distributed System*, 18(8):1028–1040, August 2007.

[10] C. Kim, D. Burger, and S. W. Keckler. An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 211–222, October 2002.

[11] B. Sinharoy, R. Kalla, W. J. Starke, H. Q. Le, R. Cargnoni, J. A. Van Norstrand, B. J. Ronchetti, J. Stuecheli, J. Leenstra, G. L. Guthrie, D. Q. Nguyen, B. Blaner, C. F. Marino, E. Retter, and P. Williams. IBM POWER7 Multicore Server Processor. *IBM Journal of Research and Development*, 55(3):1:1–1:29, May 2011.

[12] J. Stuecheli. POWER8. *Hot Chips 25: A Symposium of High Performance Chips*, August 2013.

[13] M. Zhang and K. Asanovic. Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors. In *Proceedings of the 32nd annual international symposium on Computer Architecture*, 2005.

[14] R. Desikan, C.R. Lefurgy, S.W. Keckler, and D. Burger. On-Chip MRAM as a High-Bandwidth, Low-Latency Replacement for DRAM Physical Memories. *Department of Computer Sciences Technical Report TR-02-47*, The University of Texas at Austin, 2002.