Computing and Informatics, Vol. 30, 2011, 1037–1060

BP-NUCA: CACHE PRESSURE-AWARE MIGRATION FOR HIGH-PERFORMANCE CACHING IN CMPS

Xiaomin Jia, Jiang Jiang, Yongwen Wang, Shubo Qi
Tianlei Zhao, Guitao Fu, Minxuan Zhang

Department of Microelectronics
School of Computer
National University of Defense Technology
Changsha 410073, China
e-mail: {mxzhang, xmjia}@nudt.edu.cn

Communicated by Andrzej Goscinski

Abstract. As the momentum behind Chip Multi-Processors (CMPs) continues to grow, Last Level Cache (LLC) management becomes a crucial issue to CMPs because off-chip accesses often involve a big latency. Private cache design is distinguished by smaller local access latency, good performance isolation and easy scalability, thus is becoming an attractive design alternative for LLC of CMPs. This paper proposes Balanced Private Non-Uniform Cache Architecture (BP-NUCA), a new LLC architecture that starts from private cache design for smaller local access latency and good performance isolation, then introduces a low cost mechanism to dynamically migrate private blocks among peer private caches of LLC to improve the overall space utilization. BP-NUCA achieves this by measuring the cache access pressure level that each cache set experiences at runtime and then using the information to guide block migration among different private caches of LLC. A heavily accessed set, namely a set with high access pressure level, is allowed to migrate its evicted blocks to peer private caches, replacing blocks of sets which are with the same index and have low access pressure level. By migrating blocks from heavily accessed cache sets to less accessed cache sets, BP-NUCA effectively balances space utilization of LLC among different cores. Experimental results using a full system CMP simulator show that BP-NUCA improves the overall throughput by as much as 20.3 %, 12.4 %, 14.5 % and 18.0 % (on average 7.7 %, 4.4 %, 4.0 % and 6.1 %) over private cache, shared cache, shared cache management scheme UCP and private cache organization CC respectively on a 4-core CMP for SPEC CPU2006 benchmarks.

Keywords: Chip multi-processors (CMPs), last-level cache (LLC), block migration, non-uniform cache architecture (NUCA)

1 INTRODUCTION

Chip Multi-Processors (CMPs) have emerged as the mainstream microprocessor architecture of choice in both marketplace and academia. Compared to single core design, multi-core is a more cost-effective, power efficient design alternative to explore both intra-application and inter-application thread level parallelism (TLP). Multi-threaded workloads and multiprogrammed workloads are the two fundamental ways to take advantage of the rich TLP enabled by CMPs [1]. This paper focuses on the latter and investigates the scenarios in which multiple disparate applications are running simultaneously on CMPs.

The increasing transistor budget advocates the usage of large on-chip caches to compensate for the relatively limited off-chip bandwidth on CMP platforms. Last Level Cache (LLC) [2] is an extremely important part of on-chip cache hierarchy because it is the last line before an access goes off-chip.

Traditionally, LLC can be structured as shared or private [2, 3]. In shared LLC, cache space is shared among all cores, serving misses from above cache level or bypassed requests of multiple applications running on all cores on demand basis. Whereas private LLC is composed of multiple private caches, each one private to and closely coupled to a different core both logically and physically, serving only local misses and requests. This paper treats all the private caches of private LLC as a whole and each one is called a LLC slice.

Shared caches usually have higher hit rates and work better under unbalanced workloads than private caches because shared data are not replicated and applications can use the entire cache space, as opposed to just the local portion in private caches. However, the above merits of shared caches could be jeopardized by two facts. One is that for some simultaneously running applications, absence of proper control may lead to destructive interferences [4]. The other is that the access latency of unified shared caches is bounded by the cache block farthest to the requesting core [5]. Due to poor wire scaling [5], the actual access latency to different parts of a unified shared cache differs significantly. Non-Uniform Cache Architecture (NUCA) [5] is proposed to address this problem by partitioning a cache to several latency regions, each one having a different access latency.

Private caches are distinguished by smaller access latency [3]. This is because private cache organization is inherently a NUCA [5] organization in that it has non-uniform access latencies[6] to different private caches that belong to different cores, and cache blocks are placed near to their owner cores. Private cache organization also has the advantages of good performance isolation and easy scalability. These merits make private cache organization an attractive design alternative for large on-chip LLC of CMPs. However, in a private organization, cache resources are statically partitioned among cores without regard to the diversity of application mixes running on it. This often leads to undesirable low utilization of the precious on-chip cache resources.

This paper puts forwards a new private LLC architecture, namely Balanced Private Non-Uniform Cache Architecture (BP-NUCA). BP-NUCA chooses private

LLC as the base organization of BP-NUCA for the smaller local access latency and good performance isolation, and then introduces set-level block migration to improve the space utilization of private LLC through the usage of a low cost hardware mechanism Saturation Counter Table (SCT) [7]. SCT associates each set with a Saturation Counter (SC), which is used to estimate the cache pressure level of each cache set at runtime. BP-NUCA then uses the cache pressure level information to guide the set-level block migration among different private caches of LLC. A heavily accessed set, namely a set with high access pressure level, is allowed to migrate its evicted blocks to peer private caches, replacing blocks of sets which are with the same index and have low access pressure level.

By migrating blocks from heavily accessed cache sets to less saturated cache sets, BP-NUCA not only balances the cache utilization among different cores effectively, but also avoids unnecessary interferences. Experimental results using a full system CMP simulator show that BP-NUCA improves the overall throughput by as much as 20.3 %, 12.4 %, 14.5 % and 18.0 % (on average 7.7 %, 4.4 %, 4.0 % and 6.1 %) over private cache, shared cache, a mainstream shared cache management scheme Utility-based Cache Partitioning (UCP) [8] and a mainstream private cache organization Cooperative Caching (CC) [9] respectively on a 4-core CMP for SPEC CPU2006 benchmarks.

Although BP-NUCA is equally applicable to three on-chip cache levels, we assume each core in CMPs executes one application and L2 is the LLC for brevity throughout this paper. The rest of the paper is organized as follows. Section 2 details the general framework, the basic ideas and hardware support of BP-NUCA. Section 3 describes experimental methodology used and evaluation results are shown and analyzed in Section 4. Section 5 elaborates the related work. Finally, Section 6 concludes the work.

2 BP-NUCA ARCHITECTURE

2.1 General Framework

BP-NUCA is based on Least Recently Used (LRU) replacement policy managed private cache organization. Figure 1 shows the basic framework of BP-NUCA for a 4-core CMPs. P0, P1, P2, P3 stand for the four processor cores, and L1D, L1I stand for Level 1 data cache, Level 1 instruction cache, respectively. L2 consists of four private L2 slices, i.e. L2 A, L2 B, L2 C and L2 D, with each L2 slice closely coupled to a different processor core. L2 slices are connected with on-chip network. Actually, various on-chip network topologies can be applied to on-chip interconnection of BP-NUCA. We choose the commonly used mesh in this study only for brevity. As can be seen in Figure 1, private LLC organization is a NUCA in its nature due to the fact that the access latency to a block is determined by the relative distance between the requesting core and the block. BP-NUCA takes this non-uniformity into account in its policies, which will be discussed later.

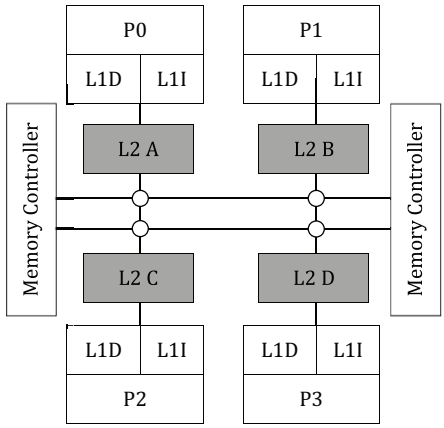


Fig. 1. The framework of BP-NUCA

As discussed above, private cache organization has the problem of low space utilization under unbalanced workloads. We call sets of all LLC slices which are of the same index address peer sets. One fact about private cache organization is that peer sets often experience very different levels of access demand. This is the case not only for unbalanced workloads but also for balanced workloads, because even the accesses of a same application to different sets are often non-uniformly distributed [7]. Based on the key insight, BP-NUCA addresses the low utilization problem by introducing set-level block migration. Set-level block migration moves blocks among peer sets. BP-NUCA performs two complementary set-level block migration types, namely *downward migration* and *upward migration*.

Downward migration migrates private blocks of a private LLC slice to neighboring peer LLC slices to "steal" cache space. A heavily accessed set, namely a set with high access pressure level, is allowed to migrate its evicted blocks to peer sets which have low access pressure level.

When an access misses in the local LLC slice but results in a hit in a remote LLC slice, the block is brought to the local LLC slice in case there is any next access to the block. This kind of migration is called *upward migration*.

To sum up, downward migration migrates blocks away from the owner cores to available space of remote cores to circumvent the low utilization problem of private caches and to increase on-chip hit rates, while upward migration brings remote hit blocks to the requesting cores to reduce the hit latency of potential further accesses to the same blocks.

2.2 Basic Ideas

BP-NUCA seeks to provide well managed cache space sharing over a private LLC. Downward migration allows sets with high access pressure to "steal" the space of

neighboring LLC slices, thus is the leading component of BP-NUCA architecture. It is intuitive that downward migration should be performed with proper control, otherwise it may end up causing more interferences rather than reducing miss rates.

Downward migration reduces the pressure on the cache sets which are unable to hold all blocks of their working set, by displacing some of those blocks to peer sets which are underutilized. Since for shared data blocks there are already replicates at peer caches, downward migration is only performed on private blocks.

Downward migration requires in the first place a mechanism to measure the cache access pressure level each cache set experiences at runtime. BP-NUCA introduces a low cost hardware mechanism Saturation Counter Table (SCT) [7] for this purpose. Rolan et al. [7] propose Saturation Counter (SC) and show that the cache demand level on each set can be effectively measured with SCs. BP-NUCA associates an SC to each set of each LLC slice to catch the cache pressure level estimation. All SCs of a LLC slice make up an Saturation Counter Table (SCT). Figure 2 illustrates the SCT structure. As depicted by Figure 2, SC [7] is a counter with saturating arithmetics, and is modified each time the set is accessed. It is incremented if the access results in a miss, otherwise it is decremented.

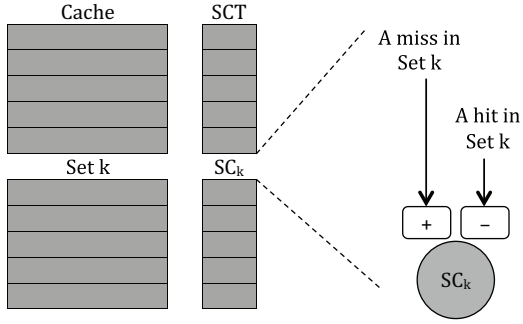


Fig. 2. SCT structure

BP-NUCA then guides the downward migration dynamically using the cache pressure level estimation. The decision whether a set should migrate its evicted blocks to peer sets, receive migrated blocks or just do nothing is called downward migration decision. Figure 3 illustrates our SCT structure-based downward migration decision algorithm.

Three policy parameters, namely (SAT, Th_M, Th_R) are used in the algorithm, where SAT is the saturation value of SCs, and Th_M, Th_R are the two thresholds used to distinguish the cache pressure levels of cache sets. As shown by Figure 3, when the SC value of a set is larger than Th_M, it is reckoned to be under high access pressure, while when the value is smaller than Th_R, it is reckoned as an underutilized set. Since the access pressure of each set is highly dependent on the set associativity of the cache itself, the proper values for saturation value SAT

```
Algorithm  DownwardMigrationDecision
Input:  Set number k and its SC value SC_k
Output:  Downward migration decision of set k Dec_k
if  SC_k ≥ Th_M
    // Set k is under high access pressure
    // Set k can migrate to peer sets
     Dec_k ← Downward Migrate
else if  SC_k < Th_R
    // Set k is under low access pressure
    // Set k can receive migrated blocks of peer sets
     Dec_k ← Receive
else
    // Set k is under moderate access pressure
    // Do nothing
    Dec_k ← None
return  Dec_k
```

Fig. 3. SCT-based downward migration decision algorithm

and the two thresholds Th_M, Th_R should reflect this correlation. In light of this, we set the values of SAT, Th_M, Th_R in the form of $\alpha A + \beta$, with A being the set associativity. The optimal values for the three parameters are $(3A-1, 2A-1, 3A/2)$, respectively, which are obtained experimentally.

Appending each set with a SC enables BP-NUCA to make migration decision at the set level rather than at the slice level. This fine control of migration would be of great value due to the fact that even different sets of a same private cache can experience very different levels of demand, which has already been discussed at length in the bibliography [7, 10, 11]. Note that although in BP-NUCA, different sets of a same slice may act differently at the same time, e.g. some sets act as block providers while others act as block receivers, each set only acts as one identity at one time, either as a provider or as a receiver but not both, as guaranteed by the downward migration algorithm. Therefore, there will not be the undesirable case that a set seeks to store its blocks to peer caches while giving away its own space to other cores [6] at the same time in BP-NUCA.

To improve cache utilization, we need not only to send blocks away to "steal" space, but also need to be able to take them back and use them. As the complement of downward migration, upward migration checks whether there are remote copies in peer sets on a miss and brings remote hit block to the requesting core. In fact, although downward migration determines whether and where to displace the evicted blocks of highly accessed sets and therefore determines whether cache space utilization can be improved, the actual utilization of the downward migrated blocks can only be performed by upward migration.

2.3 Enforcement of Migration

An enforcement mechanism is also required to actually perform migration in BP-NUCA, apart from the cache pressure estimation. The very timing for perspective downward as well as upward migration is whenever a miss happens.

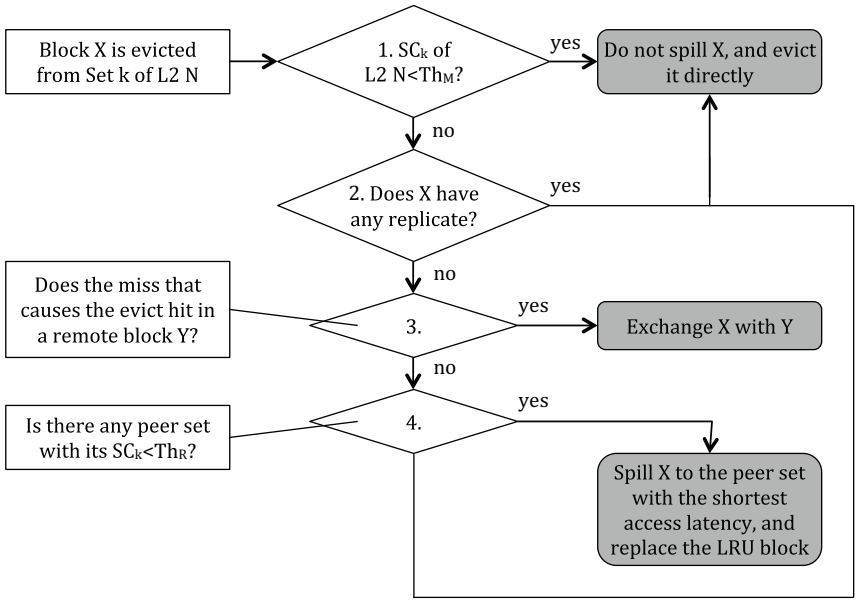


Fig. 4. Migration process of BP-NUCA

When an access misses in the local private LLC slice, a victim in the accessed set is evicted and all other LLC slices are snooped. The latter operation is also required in the baseline private cache for coherence [6]. When the SC value of the accessed set suggests it could perform downward migration, if a copy of the access is found at a remote LLC slice, then the remote copy is exchanged with the evicted block. That is, a downward migration of the evicted block and an upward migration of the remote hit block are performed at the same time. Otherwise, if no copy of the access is found, then the block is fetched from off-chip memory and in the meantime the evicted block is migrated to one of the peer sets which can receive blocks according to the decision algorithm. If there is no such peer set, the downward migration of the evicted block is aborted. The destination peer set is chosen according to the access latency. As discussed above, private cache organization is a NUCA in nature. BP-NUCA tries to migrate an evicted block to the nearest peer LLC slice to minimize the hit latency of potential future access to the block. Figure 4 illustrates the downward migration process extensively.

Considering the replacement policy for downwards migrated blocks at the destination set, we choose the basic LRU rather than other complex policies. The basic philosophy behind this choice is that the use of SCT already ensures destination sets of downwards migrated blocks are underutilized, and the migration source set is under high access pressure level. Therefore, we can reasonably assume that treating migrated blocks just like local blocks at destination sets won't cause unacceptable interferences to the local application. Otherwise, if we insert the downwards migrated blocks at the LRU position of the destination set, the blocks will be evicted soon after one more local miss to the set. A relatively aggressive replacement policy is required for downward migration to provide considerable performance boost.

When a downwards migrated block is evicted by its destination set, it is possible that the block will be further migrated to other peer sets if without proper control. This kind of migration is called *secondary downward migration*. Secondary downward migration is not allowed in BP-NUCA. In fact, the case that a downwards migrated block is evicted by its destination set generally indicates that either the block will not be reused or the reuse distance is quite large. Under either circumstance, it is likely to have little to no performance benefit to continue holding the block in LLC. Moreover, secondary downward migrations may cause deadlocks. To identify secondary downward migration, a migration bit (called m bit) is added to each block to distinguish downwards migrated blocks from local blocks.

From the above discussion, we can see that BP-NUCA requires rather little change to coherence protocols and nearly no change to replacement policies. It should be mentioned that the downward and upward migration processes are done concurrently with the fill operation of the miss, thus it is not in the critical path of memory access.

2.4 Discussion

Figure 5 shows a simple example of the operation of a BP-NUCA with four 2-way associative sets for each L2 slice. The optimal values for SAT, Th_M, Th_R are therefore 5, 3, 3, respectively. 8-bit addressing is used for simplicity, with the lower two bits being the set index and the upper 6 bits the tag. In the two tags of each set, the one above is the Most Recently Used (MRU) and the one below is the LRU.

The first reference (Ref 1) is mapped to set 0 of L2 A, where $SC = 4$, and results in a miss, thus SC is incremented. Coherence status indicates that there is no copy of the miss in peer L2 slices. The SC value of set 0 of L2 A ($SC = 5$) suggests downward migration is allowed, therefore the LRU block of set 0 of L2 A, the block with tag 010100, is migrated to the MRU position of set 0 of L2 C, where $SC < Th_R$. The corresponding m bit of set 0 of L2 C is set as well.

The second reference (Ref 2) is mapped to set 3 of L2 D, where $SC = 5$. Ref 2 results in a miss and there is no copy of the miss in peer L2 slices. Therefore, a downward migration of the victim (with tag 110000) to set 3 of L2 A is performed. The corresponding m bit of set 3 of L2 A is set as well. This example clearly demonstrates that the fine level migration control of BP-NUCA allows different sets

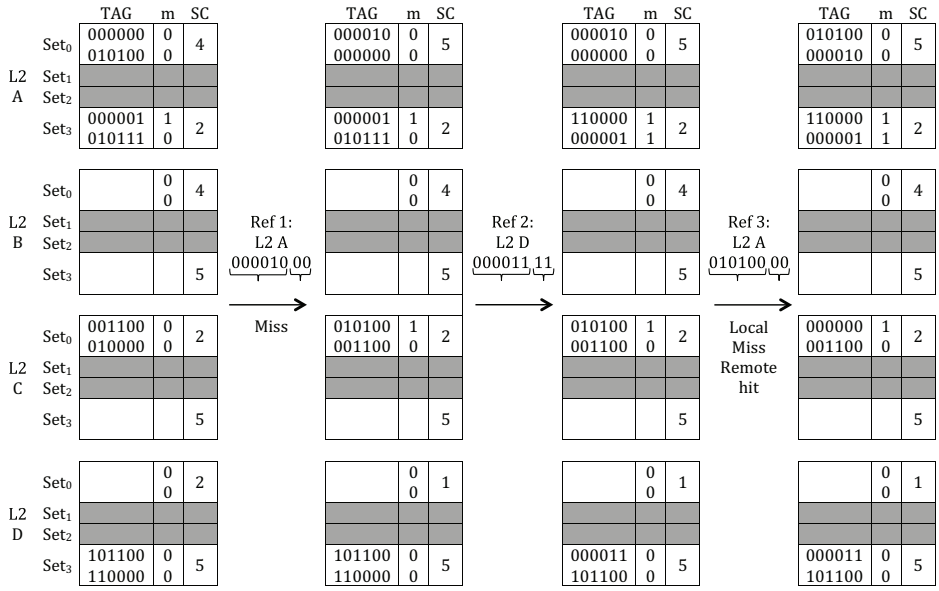


Fig. 5. An example of BP-NUCA references

of a same slice to act differently at the same time. In this case, set 0 of L2 A works as a downward migration source, meanwhile set 3 of L2 A works as a downward migration destination.

The third reference (Ref 3) is mapped to set 0 of L2 A, where $SC = 5$, which results in a miss. Coherence status suggests there is a remote copy at set 0 of L2 C. According to Figure 4, the LRU block of set 0 of L2 A is exchanged with the remote copy at set 0 of L2 C.

The adoption of SCT [7] renders BP-NUCA great adaptability in that SCs could follow and reflect the dynamic changes of cache access pressure status of each set real-timely. When downward migration starts to exert great pressure on a destination peer set, the corresponding SC catches this pressure and its value increases correspondingly. By the time the value starts to go beyond Th_R, further downward migration to the set is then prevented by the SC value. Therefore, with proper parameter values, the SCT-based downward migration algorithm itself can ensure that the interferences to destinations caused by downward migrations are under control.

3 EXPERIMENTAL METHODOLOGY

3.1 Configurations

We use g-cache of Virtutech Simics [12], a full system simulator for our performance studies. Evaluation is performed on a 4-core CMP with configuration given in

Table 1. An in-order core model is used so that we can evaluate our proposal within a reasonable time.

System	**4-core CMP**
Processor Core	UltraSPARC-III, in-order
OS	Linux2.6.18
L1 I/D (Private)	32 KB, 64 B-block, 2-way, LRU repl., 1-cycle hits
L2 (Private)	512 KB, 64 B-block, 8-way, LRU repl., 10-cycle local hits, two peers with 38-cycle remote hits, one peer with 46-cycle remote hits
L2 (Shared)	2 MB, 64 B-block, 32-way, LRU repl., 19-cycle hits
CC Protocol	Snoop based MESI protocol
Memory	350-cycle access latency

Table 1. Simulation configuration

To fully evaluate BP-NUCA, we extend the simulator with four other schemes for comparison, i.e. Private, Shared, UCP [8] and CC [9]. Private and Shared stand for the basic private and shared LLC organization, respectively, with no further optimization. As BP-NUCA starts from private organization, we choose Private as the baseline for comparison. UCP [8] is a mainstream shared cache partitioning scheme to tackle the destructive inter-thread interference problem of shared cache organization. UCP seeks to minimize the overall miss rate of shared cache by giving a cache way to the application that benefits most from it. CC [9] is a mainstream private cache optimization scheme for CMPs, and is the first to explicitly introduce "cache stealing" to improve space utilization in private cache.

Each core is coupled with a 512 KB L2 for all the schemes considered. For private schemes, i.e. Private, CC and BP-NUCA, this means a 512 KB private L2 per core, with a 10-cycle hit latency to the local L2, a 38-cycle hit latency to two peer L2s and a 46-cycle hit latency to one peer L2. As for shared schemes, i.e. Shared and UCP, this results in a unified 2 MB L2 with a global 19-cycle hit latency. We use relatively small cache configuration so that the cache will be under higher access pressure, with more obvious contentions and interferences. The basic philosophy behind is that a large fraction of the current real-world applications already have working sets much larger than those of the selected benchmarks.

Mesh network is used for intra-chip data transfers, modeling non-uniform access latency of private organization. A unified organization for shared schemes is used, with access latency bounded by the farthest bank. We model on-chip network and cache latencies with CACTI 6.0 [13]. All caches are of a uniform block size of 64 B and use LRU as baseline replacement policy. Off-chip memory access latency is 350 cycles.

3.2 Workloads

For our study, we use 23 SPEC CPU2006 benchmarks to create 16 4-benchmark multiprogrammed workloads, as listed in Table 2. Benchmarks for each workload are randomly selected. All workloads are simulated till each benchmark in the workload executes at-least 250 M instructions. When a benchmark reaches 250 M instructions, its statistics are "frozen", but it continues to execute so that it still competes for cache resources.

Name	Benchmarks in Workloads
MIX00	perlbench.c mcf.i milc.s dealII.r
MIX01	bwaves.b cactusADM.r dealII.r libquantum.r
MIX02	dealII.r soplex.r povray.s lbm.r
MIX03	milc.s dealII.r namd.r xalancbmk.r
MIX04	bzip2.ip bwaves.b leslie3d.r omnetpp.r
MIX05	gamess.c leslie3d.r dealII.r libquantum.r
MIX06	cactusADM.r soplex.r omnetpp.r astar.b
MIX07	mcf.i zeusmp.r libquantum.r astar.b
MIX08	hmmer.r sjeng.r libquantum.r astar.b
MIX09	bzip2.ip gcc.1 gobmk.1 lbm.r
MIX10	bwaves.b milc.s zeusmp.r libquantum.r
MIX11	bwaves.b soplex.r libquantum.r astar.b
MIX12	bzip2.ip milc.s mcf.i hmmer.r
MIX13	zeusmp.r gobmk.1 libquantum.r astar.b
MIX14	milc.s gobmk.1 lbm.r specrand.r
MIX15	bzip2.ip milc.s povray.s lbm.r

Table 2. Workloads (reference input sets)

3.3 Metrics

Throughput, Weighted Speedup (WS) and Harmonic Mean (Hmean) [14] are the three metrics commonly used to quantify the aggregate performance of a system with multiple applications (threads) running concurrently. Let IPC_i^{MP} and IPC_i^{SP} be the number of useful instructions executed per cycle of application i under multiple-application and single-application execution respectively, for N-application (or N-thread) workloads, throughput, WS and Hmean are defined as follows [14]:

$$throughput = \sum_{0}^{N-1} IPC_i^{MP} \tag{1}$$

$$WS = \sum_{0}^{N-1} \frac{IPC_i^{MP}}{IPC_i^{SP}} \tag{2}$$

$$Hmean = \frac{N}{\sum_0^{N-1} \frac{IPC_i^{SP}}{IPC_i^{MP}}}. \quad (3)$$

Throughput reflects the overall performance boost but may favor high IPC applications too much. WS weights the relative speedups of all applications evenly and indicates the improvement on execution time. Hmean is a fairness metrics and balances both fairness and performance. We use all the three metrics for performance comparisons. Besides, since BP-NUCA is based on private cache organization and the number of accesses to each slice of BP-NUCA is different, the commonly used miss rate metric is meaningless. For the purpose of memory access analysis, we adopt the L1 misses breakdown metric used in CC [9] to show the changes to LLC accesses.

4 RESULTS AND ANALYSIS

We compare BP-NUCA to four other schemes: Private, Shared, UCP and CC. (SAT, Th_M, Th_R) are policy parameters used in BP-NUCA. Their values not only impact the accuracy of the cache pressure level estimation, but also tune the aggressiveness of BP-NUCA. We conduct a parameter sensitivity study to pinpoint the optimal parameter configuration. Simulation for more than 20 combinations of (SAT, Th_M, Th_R) are examined and ($3A-1$, $2A-1$, $3A/2$) are found to be the optimal values for the three parameters.

In addition, we examine the sensitivity of BP-NUCA performance to cache size and associativity as well as its scalability to larger CMP systems with more cores.

4.1 Performance on Throughput Metric

Figure 6 shows the throughput of Shared, UCP, CC and BP-NUCA, normalized to Private. Geomean is the geometric mean of all 16 workloads.

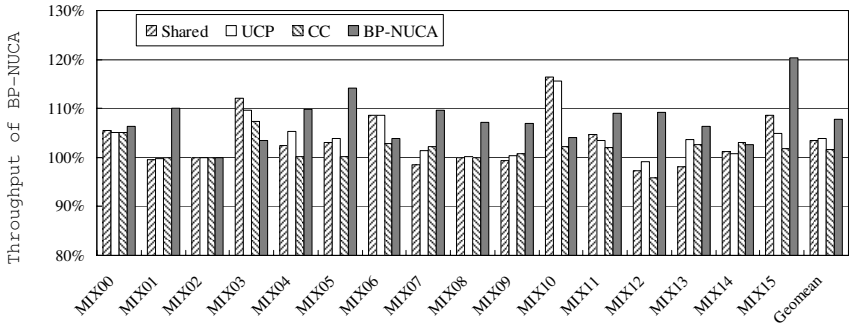


Fig. 6. Throughput of Shared, UCP, CC and BP-NUCA (normalized to that of Private)

We expect BP-NUCA to outperform the other four schemes due to several reasons:

1. first and foremost, we adopt saturation counters (SCs) [7] to measure the access pressure level of each cache set;
2. BP-NUCA proposes a SCT-based downward migration, which enables adaptive control of cache space "stealing";
3. as we discussed in Section 2.2, the SCT-based downward migration is performed at set level, and this fine control style promises more performance boost due to the fact that different sets of a same application can experience very different levels of demand [7];
4. the private organization of BP-NUCA allows a smaller hit latency for most of the cache accesses over shared organization, the hit latency of which is bounded by the farthest bank and which may sometimes store local blocks of a core far away due to its global addressing style.

As can be seen in Figure 6, BP-NUCA outperforms both Shared and UCP for 11 of 16 workloads, with an average improvement of 4.4 %, 4.0 %, respectively, which confirms our expectation. In contrast, another private scheme CC degrades throughput when compared to Shared or UCP for 10 out of the 16 workloads. This suggests that BP-NUCA can effectively simulate a shared cache organization and explore the capacity sharing potential with SCT-based downward migration, improving cache resource utilization of private cache organization.

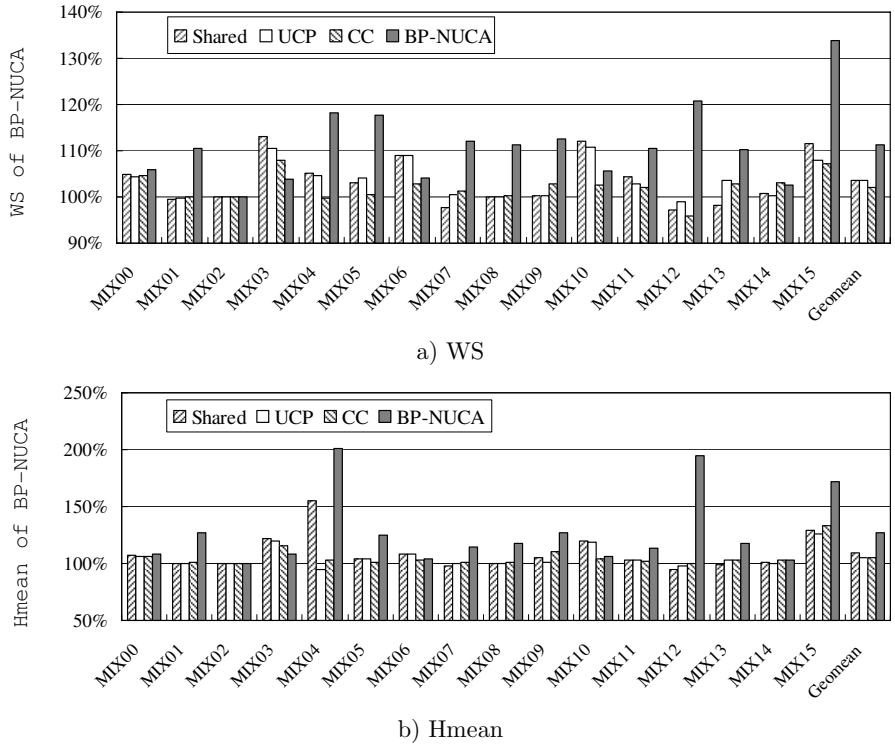
Shared and UCP work better than BP-NUCA for 5 of the 16 workloads. It is easy to understand since BP-NUCA is based on private organization, which is totally different from shared schemes, i.e. Shared and UCP, due to the fact that they can adequately share the cache space, especially under unbalanced workloads. We perceive BP-NUCA fails to achieve comparable performance to Shared and UCP for the 5 workloads is because of its sort of conservative "stealing". However, it is difficult to tune the aggressiveness of BP-NUCA, since the performance boost is highly dependent on the inherent characteristics of the workload mixes. Dynamically tuning the parameters may be profitable.

BP-NUCA shows stable and consistent performance boost over Private, outperforming Private for all 16 workloads simulated. As for another private scheme CC, BP-NUCA significantly outperforms it for 13 out of the 16 workloads, with an average performance boost of 6.1 %.

The use of cache pressure level information collected with SCT to guide cache space "stealing", including both when to migrate and when to stop, contributes to BP-NUCA's benefit over CC. In general, BP-NUCA improves throughput by as much as 20.3 %, 12.4 %, 14.5 % and 18.0 % (on average 7.7 %, 4.4 %, 4.0 % and 6.1 %) over Private, Shared, UCP and CC, respectively.

4.2 Performance on Weighted Speedup and Hmean

Figure 7 a) shows the WS of Shared, UCP, CC and BP-NUCA, normalized to Private. The WS with BP-NUCA is higher than those with Shared and UCP for 13 out of the 16 workloads. BP-NUCA achieves better WS than Private for all 16 workloads, and than CC for 13 out of the 16 workloads. The results of WS are consistent with those of throughput. Generally, BP-NUCA improves WS by as much as 33.8 %, 24.3 %, 23.9 % and 25.9 % (on average 11.2 %, 7.7 %, 6.8 % and 9.0 %) over Private, Shared, UCP and CC, respectively. This indicates that BP-NUCA can bring a considerable reduction in overall execution time.



a) WS

b) Hmean

Fig. 7. WS and Hmean of Shared, UCP, CC and BP-NUCA (normalized to that of Private)

Although BP-NUCA improves the overall throughput and WS significantly, it is important that this does not come at the expense of fairness of the system. Figure 7 b) shows the Hmean fairness of Shared, CC, UCP and BP-NUCA, normalized to Private. BP-NUCA has better Hmean value over Private for all 16 workloads, and better Hmean value over CC for 13 out of the 16 workloads. BP-NUCA also has better Hmean value over Shared and UCP for most workloads. The results are consistent with those for throughput and WS. To sum up, BP-NUCA improves Hmean

fairness by 27.5 %, 17.3 %, 15.5 % and 21.1 % on average over Private, Shared, UCP and CC, respectively. Thus, BP-NUCA not only improves performance but also balances fairness well.

4.3 Memory Access Breakdowns

Figure 8 shows L1 miss breakdowns [9] for Private, Shared, UCP, CC and BP-NUCA schemes. We break down the L1 misses into three parts, namely local L2 access rate, remote L2 access rate and off-chip access rate respectively. BP-NUCA can effectively reduce the amount of off-chip accesses and increase the on-chip L2 hit rates when compared to Private for most of the workloads. For several workloads, the on-chip hit rates (local combined with remote) of BP-NUCA are even higher than that of Shared or UCP. This suggests that the downwards migrated blocks truly promise more hits for most applications.

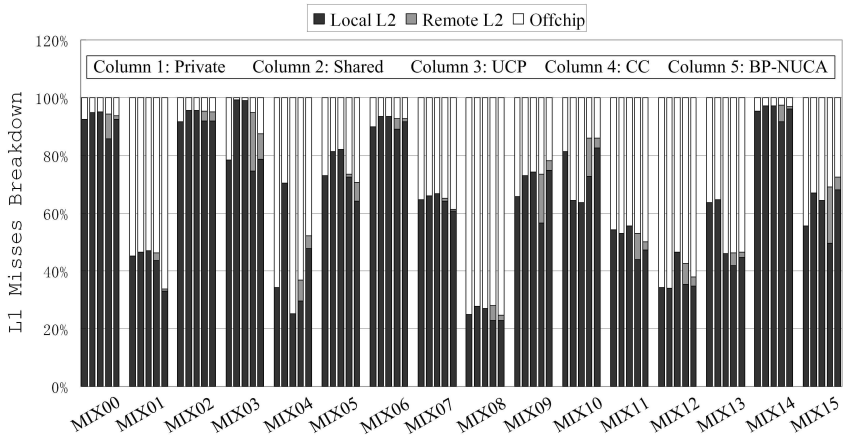


Fig. 8. L1 misses breakdown (left to right in each group: Private, Shared, UCP, CC and BP-NUCA)

However, for some workloads, the miss rates of BP-NUCA even increase over other schemes, although the corresponding throughput, WS and Hmean are improved. Comparing Figure 6 and Figure 8, we see that lower total miss rates do not necessarily result in higher performance. This is because the contribution of an access to performance is highly dependent on the importance of the access to the application and on the inherent behavior characteristics of the application itself.

4.4 Memory Configuration Sensitivity

We now evaluate the performance robustness of BP-NUCA. Cache size and cache associativity are the two aspects of memory configuration that impact cache access

behaviors greatly. When cache size increases, more data in working set can be held in cache, reducing cache access pressure. On the other hand, larger associativity can reduce conflict misses, and that is why a lot of modern microprocessors adopt highly associative non-first level caches. The main idea of the memory configuration sensitivity study is to examine the benefit of BP-NUCA across a spectrum of memory configurations, including different size and different ways.

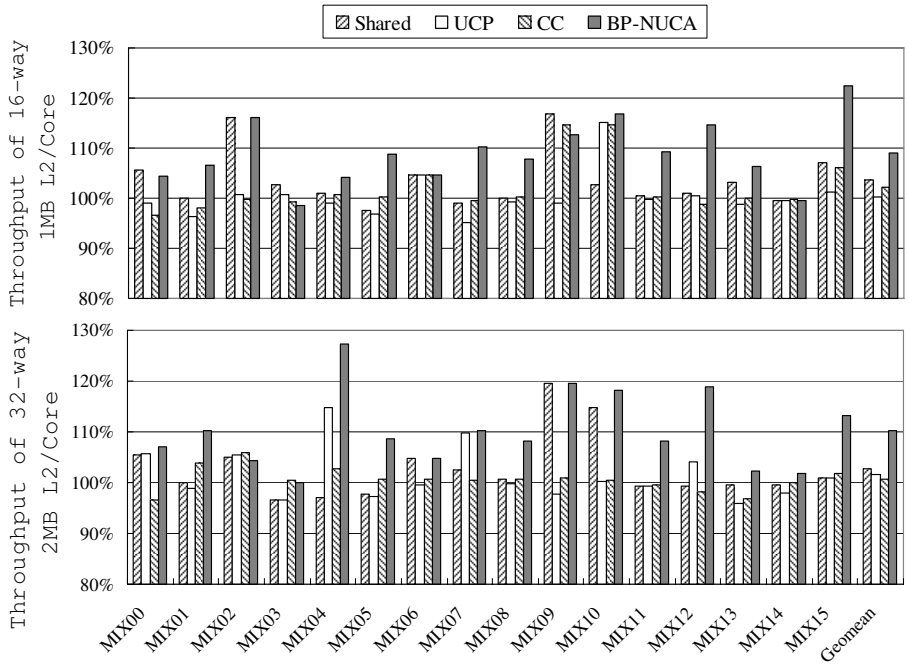


Fig. 9. Throughput sensitivity of BP-NUCA to cache size (from top to bottom: 16-way 1 MB L2 per core, 32-way 2 MB L2 per core, normalized to Private)

The benefit of BP-NUCA is impacted by the aggregate cache capacity. We vary the cache size by keeping the number of sets constant while changing the associativity. Figure 9 shows the throughput of Shared, UCP, CC and BP-NUCA with larger cache size.

As the aggregate cache size increases, the advantage of BP-NUCA over Private also increases gradually. In contrast, as the aggregate cache size increases, the benefit of Shared, UCP and CC is diminishing in general, as shown by Figure 9. Therefore, the performance of BP-NUCA shows better scalability with increasing cache size. With larger cache size, BP-NUCA is able to explore more cache access pressure imbalance among different cores for more performance benefit. With the increasing problem size and number of applications, access pressure on future large

on-chip caches and the access pressure gap between different sets are more likely to increase rather than decrease, which makes BP-NUCA more favorable.

In general, BP-NUCA improves the average throughput by 8.9 %, 5.3 %, 8.7 % and 6.8 % over Private, Shared, UCP and CC for 16-way 1 MB L2 per core respectively, by 10.2 %, 7.5 %, 8.6 % and 9.5 % for 32-way 2 MB L2 per core respectively.

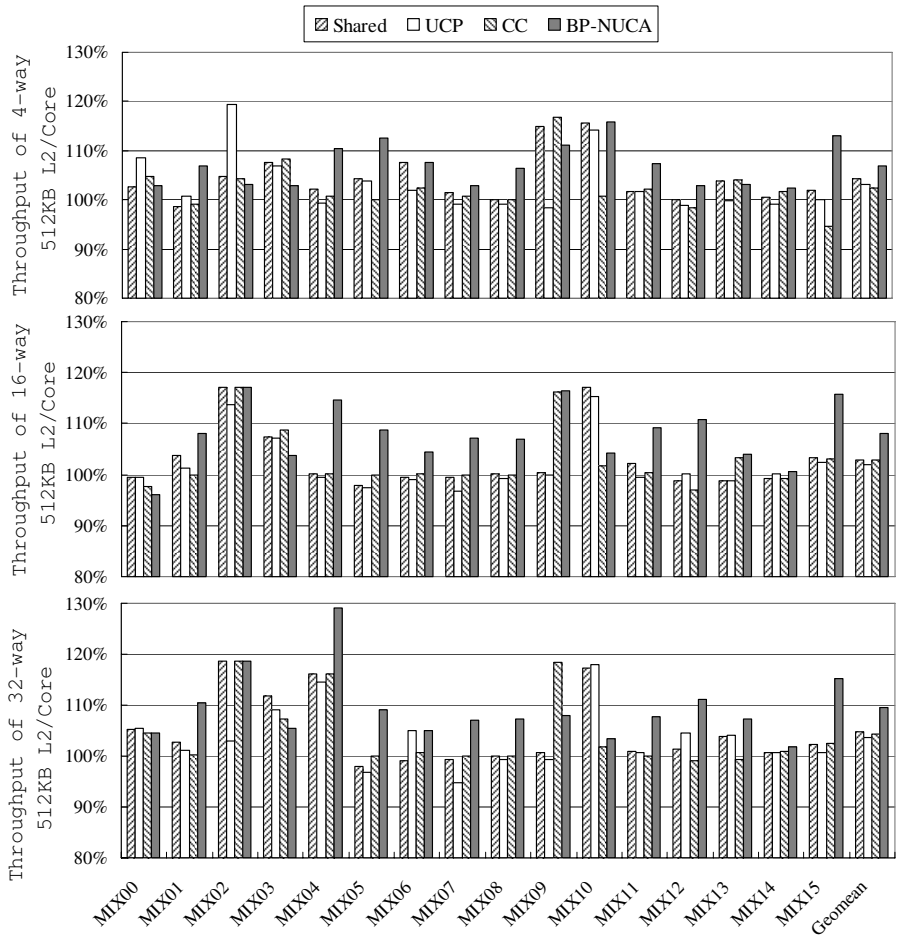


Fig. 10. Throughput sensitivity of BP-NUCA to associativity (from top to bottom: 4-way, 16-way and 32-way 512 KB L2 per core, normalized to Private)

Figure 10 compares the throughput of Shared, UCP, CC and BP-NUCA with different associativity (4-way, 16-way or 32-way 512 KB L2 per core, respectively). Combining Figure 6 and Figure 10, we can see that BP-NUCA achieves noticeable average performance boost for all 4 associativity configurations over Private,

though the performance benefit of individual workload varies slightly across configurations. It can also be observed that with associativity increasing under constant cache size, the performance benefit of Shared, UCP and CC changes unpredictably. In contrast, the performance boost of BP-NUCA over Private increases slightly with increasing ways. The stability of BP-NUCA makes it more suitable for caches with high associativity, which is the case for large on-chip LLCs in many modern CMP platforms.

In general, BP-NUCA improves the average throughput by 7.0 %, 2.8 %, 3.9 % and 4.7 % over Private, Shared, UCP and CC for 4-way 512 KB L2 per core respectively, by 8.0 %, 5.4 %, 6.3 % and 5.2 % for 16-way 512 KB L2 per core respectively, by 9.5 %, 4.6 %, 5.9 % and 5.1 % for 32-way 512 KB L2 per core respectively.

4.5 Scalability of BP-NUCA to Larger CMP Systems

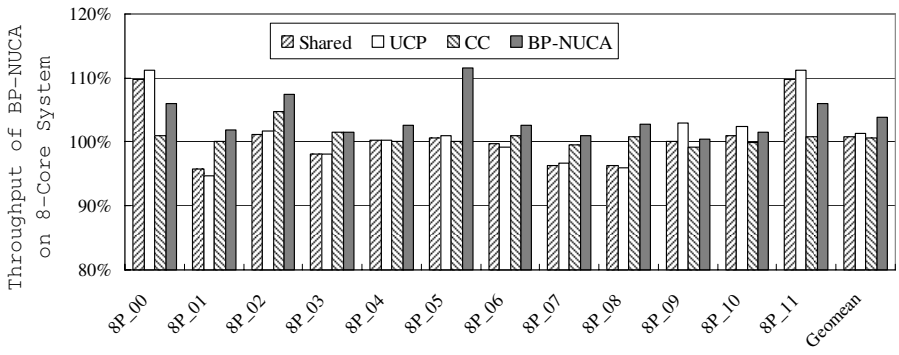


Fig. 11. Throughput of Shared, UCP, CC and BP-NUCA on 8-core CMPs (normalized to Private)

To examine the scalability of BP-NUCA performance to larger CMP systems with more cores, we evaluate BP-NUCA for 8-core system with 12 8-benchmark workloads formed by randomly combining from 20 SPEC CPU2006 benchmarks. We use the same 8-way 512 KB per core basic configuration. Figure 11 shows the throughput of Shared, UCP, CC and BP-NUCA, also normalized to Private. BP-NUCA outperforms Private and CC for all 8-core workloads except for 8P_03. BP-NUCA also achieves considerably better performance over Shared and UCP for 8 out of the 12 workloads considered. In general, BP-NUCA improves throughput by up to 11.6 %, 10.9 %, 10.6 % and 11.4 % (on average 4.1 %, 3.2 %, 2.7 % and 3.1 %) over Private, Shared, UCP and CC, respectively.

4.6 Hardware Overhead

The storage overhead of BP-NUCA is composed of three parts, i.e. SCT, m bits and threshold registers. Assuming $SAT = 3A - 1$, let S, A, L and T be the number of sets, associativity, block size and tag size respectively; total storage overhead of BP-NUCA can be calculated by Equation (4).

$$\frac{A + \log_2 A + 2}{A \times (L + T)} + \frac{3 \times \log_2 A + 6}{S \times A \times (L + T)} \tag{4}$$

As can be seen, the total storage overhead of BP-NUCA is mainly dependent on terms associativity, block size and tag size. Assuming a 40-bit physical address space, for the baseline 8-way 512 KB private LLC per core with 64 B cache block, BP-NUCA requires not more than 0.3 % of the LLC storage overhead, incurring fairly small silicon area cost; and this value decreases with increasing associativity. Note that none of the structures or operations required by BP-NUCA is in the critical path, resource-intensive, complex, or power hungry.

5 RELATED WORK

The related work on LLC of CMPs generally falls into two categories, namely shared cache based and private cache based. Most [8, 15, 16, 17, 18] of the studies are built on top of a physically shared cache for its good space utilization and low complexity. The main challenge for a physically shared design is to tackle the inter-thread interferences. Cache partitioning [8, 15, 17] is the most studied and relatively effective solution to the challenge, with UCP [8] being the prevailing scheme. However, while a large shared LLC provides good utilization of cache space, latency and power consumption worsens, particularly with aggressive porting or banking due to poor wire scaling [5, 19], which limits its scalability with growing cores.

Several other designs [2, 19, 20, 21] seek to optimize distributed shared caches. Distributed shared cache is a more scalable design, and the local access latency is also much smaller. However, it faces the challenge of bringing a block close to the requestor, which, if not neatly realized, may lead to big hit latency.

Since compared to shared cache, private cache organization has the advantages of smaller access latency, good performance isolation, easy scalability and non-uniform access latency [5] in its nature [6], it is becoming an attractive design alternative for large on-chip LLCs of CMPs. However, in a private organization, the statically partitioned fashion of cache resources may lead to undesirable low utilization problem.

As a private cache sharing scheme for LLC of CMPs, CC [9] is the first to explicitly introduce “cache stealing” for cache capacity utilization improvement. When a block is evicted from a private LLC, CC [9] decides whether to store it to peer LLCs randomly with a pre-set probability. However, one problem with CC is that the migrated blocks can only “steal” the space of replicates (blocks that have

more than one copy) of peer caches, which limits the potential performance benefit. Another problem with CC is that CC [9] decides whether to migrate an evicted block randomly with a pre-set probability. Neither the control with a probability nor the probability value selection takes into account the cache access pressure of each application.

CMP-NuRAPID [22] suggests stealing capacity of neighboring caches when there is not enough capacity for private data in local cache. However, the "stealing" in CMP-NuRAPID is simply demand-based, with no proper control. BP-NUCA differs from CC [9] and CMP-NuRAPID [22] in that it introduces SCT [7] to measure the access pressure on each set, which is then used for downward migration control.

In Dynamic Spill-Receive (DSR) [6], each private cache learns whether it should act as a downward migration source (or spiller in their words) or a destination (receiver) using set dueling monitors. DSR determines downward migration source or destination at the granularity of the whole LLC slice, thus could waste some opportunity for improvement due to the uneven distribution of application cache access across different sets whereas BP-NUCA performs downward migration at set level, providing a much finer control to account for the uneven access distribution across sets.

To the best of our knowledge, existing studies on private LLCs either migrate evicted blocks only into invalid or replicate blocks of peer caches [6, 9] or without proper control [22]. In contrast, BP-NUCA allows evicted blocks to use the same LRU replacement policy at the destination to derive more benefit, and to use the SCT structure to adaptively avoid the undesirable interferences to local applications.

6 CONCLUSIONS

Private LLCs of CMPs suffer from the inability to share cache capacity among cores, especially when the cache requirement of each core is unbalanced [6]. This paper presents a new private LLC architecture BP-NUCA, which introduces a low cost mechanism SCT [7] to help smartly perform the block migration among peer LLC slices to effectively balance the cache utilization. BP-NUCA enables fine level control of private LLC management and great adaptability to the dynamical changes of workload behaviors. Experimental results using a full system CMP simulator show that BP-NUCA improves the overall throughput by 7.7 %, 4.4 %, 4.0 % and 6.1 % on average over private cache, shared cache, UCP [8] and CC [9], respectively on a 4-core CMP for SPEC CPU2006 benchmarks. Moreover, BP-NUCA requires a total storage overhead of no more than 0.3% the current LLC storage, and does not require changes to the current cache structure.

Acknowledgments

We thank the anonymous reviewers for their constructive comments and suggestions. This work was partially supported by the National Natural Science Foundation of China under Grant No. 60970036, No. 60873016 and the National High

Technology Development 863 Program of China under Grant No. 2009AA01Z124 and No. 2009AA01Z102.

REFERENCES

[1] Zhao, L.—Makineni, S.: Cachescouts: Fine-Grain Monitoring of Shared Caches in Cmp Platforms. Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques, Brasov, Romania, Acm 2007, pp. 339–352.

[2] Liu, C.—Kandemir, M.: Organizing the Last Line of Defense Before Hitting the Memory Wall for Cmps. Proc. Int. Symposium on High Performance Computer Architecture, Madrid, Spain, Ieee Computer Society 2004, pp. 176–185.

[3] Merino, J.—Gregorio, J. A.: Esp-Nuca: A Low-Cost Adaptive Non-Uniform Cache Architecture. Proceedings of the P^{th} International Symposium on High-Performance Computer Architecture, Bangalore, India, Ieee 2010, pp. 200–209.

[4] Jia, X.—Zhang, M.: Towards Online Application Cache Behaviors Identification in Cmps. the 12th Ieee International Conference on High Performance Computing and Communications, Melbourne, Australia, Ieee Computer Society 2010, pp. 1–8.

[5] Kim, C.—Keckler, S.: An Adaptive, Non-uniform Cache Structure for Wire-dominated On-chip Caches. Proc Int. Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, California, ACM 2002, pp. 211–222.

[6] Qureshi, M. K.: Adaptive Spill-Receive for Robust High-Performance Caching in CMPs. Proc. Int. Symposium on High Performance Computer Architecture, Raleigh, North Carolina, USA, IEEE Computer Society 2009, pp. 45–54.

[7] Rolán, D.—Doallo, R.: Adaptive Line Placement with the Set Balancing Cache. Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, New York, NY, ACM 2009, pp. 529–540.

[8] Qureshi, M. K.—Patt, Y. N.: Utility-based Cache Partitioning: A Low-overhead, High-Performance, Runtime Mechanism to Partition Shared Caches. Proc. Int. Symposium on Microarchitecture, Orlando, Florida, USA, IEEE Computer Society 2006, pp. 423–432.

[9] Chang, J.—Sohi, G. S.: Cooperative Caching for Chip Multiprocessors. Proc. Int. Symposium on Computer Architecture, Boston, MS, USA, IEEE Computer Society 2006, pp. 264–276.

[10] Peir, J.—Hsu, W. W.: Capturing Dynamic Memory Reference Behavior with Adaptative Cache Topology. ASPLOS, San José, California, USA, ACM 1998, pp. 240–250.

[11] Qureshi, M. K.—PATT, Y. N.: The V-Way Cache: Demand Based Associativity via Global Replacement. SIGARCH Comput. Archit. News, Vol. 33, 2005, No. 2, pp. 544–555.

[12] Magnusson, P. S.—Werner, B.: Simics: A Full System Simulation Platform. Computer, Vol. 35, 2002, No. 2, pp. 50–58.

[13] Muralimanohar, N.—Jouppi, N.: Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0. Proc. Int. Symposium on Microarchitecture, Chicago, Illinois, USA, IEEE Computer Society 2007, pp. 3–14.

[14] Eyerman, S.—Eeckhout, L.: System Level Performance Metrics for Multiprogram Workloads. Ieee Micro, May-June 2008, Vol. 28, No. 3, pp. 42–53.

[15] Suh, G. E.—Rudolph, L.: A New Memory Monitoring Scheme for Memory-Aware Scheduling and Partitioning. Proc. Int. Symposium on High Performance Computer Architecture, Boston, Ma, Usa, Ieee Computer Society, 2002, pp. 117–128.

[16] Kim, S.—Solihin, Y.: Fair Cache Sharing and Partitioning in a Chip Multiprocessor Architecture. Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques, Antibes, Juan-les-Pins, France, IEEE Computer Society 2004, pp. 111–122.

[17] Rafique, N.—Thottethodi, M.: Architectural Support for Operating System-driven CMP Cache Management. Proceedings of the 15th International Conference on Parallel Architectures and Compilation Techniques, Seattle, Washington, USA, ACM 2006, pp. 2–12.

[18] Jaleel, A.—Emer, J.: Adaptive Insertion Policies for Managing Shared Caches on Cmps. Proc. Int. Conference on Parallel Architectures and Compilation Techniques, Toronto, Canada, Acm 2008, pp. 208–219.

[19] Yeh, T. Y.—Reinman, G.: Fast and Fair: Data-Stream Quality of Service. Proc. Int. Conference on Compilers, Architectures and Synthesis for Embedded Systems, San Francisco, California, Usa, Acm 2005, pp. 237–248.

[20] Zhang, M.—Asanovic, K.: Victim Replication: Maximizing Capacity While Hiding Wire Delay in Tiled Chip Multiprocessors. Proceedings of the 32nd Annual IEEE/ACM International Symposium on Computer Architecture, Madison, Wisconsin, USA, IEEE Computer Society 2005, pp. 336–345.

[21] Beckmann, B. M.—Wood, D. A.: ASR: Adaptive Selective Replication for CMP Caches. Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, Orlando, Florida, USA, IEEE Computer Society 2006, pp. 443–454.

[22] Chishti, Z.—Vijaykumar, T.: Optimizing Replication, Communication, and Capacity Allocation in CMPs. Proc. Int. Symp. on Computer Architecture, Madison, Wisconsin, USA, IEEE 2005, pp. 357–368.

Xiaomin Jia received her B. Sc. degree, M. Sc. degree and Ph. D. degree in Computer Science from the National University of Defense Technology in China in 2004, 2006 and 2011, respectively. Her current research interests cover on-chip memory hierarchies and interconnections.

Jiang Jiang received his B. Sc. degree, M. Sc. degree and Ph. D. degree in Computer Science from the National University of Defense Technology in China. He is now a Professor of Computer Science at the National University of Defense Technology. His research interests cover microprocessor architectures and on-chip interconnections.

Yongwen Wang received his B. Sc. degree, M. Sc. degree and Ph. D. degree in Computer Science from the National University of Defense Technology in China. He is now an Associate Professor of computer science at the National University of Defense Technology. His research interests cover microprocessor architectures, compilation techniques and on-chip interconnections.

Shubo Qi received his B. Sc. degree from Xi'an University of Electronic Technology in 2004, and his M. Sc. degree from the National University of Defense Technology in 2006 in China. Since 2007, he has been a PhD student in Computer Science at the National University of Defense Technology. His research interests cover microprocessor architectures and on-chip interconnections.

Tianlei Zhao received his B. Sc. degree and his M. Sc. degree in Computer Science from the National University of Defense Technology in China in 2004 and 2006, respectively. Since 2007, he has been a Ph. D. student in Computer Science at the National University of Defense Technology. His research interests cover microprocessor architectures and compilation techniques.

Guitao Fu received his B. Sc. degree and M. Sc. degree in computer science from the National University of Defense Technology in China in 2004 and 2007, respectively. Since 2008, he has been a PhD student in Computer Science at the National University of Defense Technology. His current research interests include cache architecture and coherence protocols of Chip Multi-Processors.

Minxuan Zhang is a Professor of Computer Science at the National University of Defense Technology. His research interests cover computer architectures, computer systems, novel microprocessor architectures and VLSI design.