Distance-Aware Round-Robin Mapping for Large NUCA Caches

Alberto Ros*, Marcelo Cintra†, Manuel E. Acacio* and José M. García*
* Departamento de Ingeniería y Tecnología de Computadores, Universidad de Murcia, 30100 Murcia (Spain)
Email: {a.ros,meacacio,jmgarcia}@ditec.um.es
† School of Informatics, University of Edinburgh
Email: mc@inf.ed.ac.uk

***Abstract*—In many-core architectures, memory blocks are commonly assigned to the banks of a NUCA cache by following a physical mapping. This mapping assigns blocks to cache banks in a round-robin fashion, thus neglecting the distance between the cores that most frequently access every block and the corresponding NUCA bank for the block. This issue impacts both cache access latency and the amount of on-chip network traffic generated. On the other hand, first-touch mapping policies, which take into account distance, can lead to an unbalanced utilization of cache banks, and consequently, to an increased number of expensive off-chip accesses. In this work, we propose the *distance-aware round-robin* mapping policy, an OS-managed policy which addresses the trade-off between cache access latency and number of off-chip accesses. Our policy tries to map the pages accessed by a core to its closest (local) bank, like in a first-touch policy. However, our policy also introduces an upper bound on the deviation of the distribution of memory pages among cache banks, which lessens the number of off-chip accesses. This trade-off is addressed without requiring any extra hardware structure. We also show that the private cache indexing commonly used in many-core architectures is not the most appropriate for OS-managed distance-aware mapping policies, and propose to employ different bits for such indexing. Using GEMS simulator we show that our proposal obtains average improvements of 11% for parallel applications and 14% for multi-programmed workloads in terms of execution time, and significant reductions in network traffic, over a traditional physical mapping. Moreover, when compared to a first-touch mapping policy, our proposal improves average execution time by 5% for parallel applications and 6% for multi-programmed workloads, slightly increasing on-chip network traffic.**

I. Introduction

Nowadays, the most efficient way of organizing the increasing number of transistors per chip is to integrate multiple processor cores in the same chip. Recent examples of these chip multiprocessors (CMP) are, among others, the 2-core IBM Power6 [1] and the 8-core Sun T2 [2]. CMP architectures that integrate tens of processor cores (usually known as many-core CMPs) are expected for the near future, after Intel unveiled recently the 80-core Polaris prototype [3]. Particularly, tiled CMP architectures, which are designed as arrays of identical or close-to-identical building blocks (tiles), are a scalable alternative to current small-scale CMP designs and will help in keeping complexity manageable. In these architectures, each tile is comprised by a core, one or several levels of caches, and a network interface that connects all tiles through a point-to-point network.

One important decision when designing tiled CMPs is how to organize the last-level on-chip cache, since cache misses at this level result in long-latency off-chip accesses. The two common ways of organizing this cache level, e.g. the L2 cache, are *private* to the local core or *shared* among all cores. The private L2 cache organization, implemented, for example, in the AMD Athlon™ Dual-Core processor, offers a fast access to the L2 cache. However, it has two main drawbacks that lead to an inefficient use of the aggregate L2 cache capacity. First, local L2 banks keep a copy of the blocks requested by the corresponding core, potentially replicating blocks in multiple L2 cache banks. Second, load balancing problems appear when the working set accessed by all the cores is heterogeneous, i.e., some banks may be over-utilized while others are under-utilized. Since these drawbacks can result in more off-chip accesses, which are very expensive, it seems that the trend is to implement a shared cache organization [1], [2], [4].

The shared L2 cache organization, also called non-uniform cache architecture (NUCA) [5], achieves more efficient use of the L2 cache by storing only one copy of each block and by distributing the copies across the different banks. The main downside of this organization in many-core CMPs is the long L2 access latency, since it depends on the bank wherein the block is allocated, i.e., the *home* bank or tile. This issue is addressed in this work.

The most straightforward way of distributing blocks among the different tiles is by using a physical mapping policy in which a set of bits in the block address defines the home bank for every block. Some recent proposals [6], [7] and commercial CMPs [1], [2] choose the less significant bits[1] for selecting the home bank. In this way, blocks are assigned to banks in a round-robin fashion with block-size granularity. This distribution of blocks does not take into account the distance between the requesting core and the home bank on a L1 cache miss. Moreover, the average distance between two tiles significantly increases with the size of the CMP, which can become a performance problem for many-core CMPs.

On the other hand, page-size granularity seems to be a better choice than block-size granularity for future tiled CMPs because (1) it is more appropriate for new technologies aimed

[1]In this paper, when we refer to the less significant bits of an address we are not considering the block offset.

978-1-4244-4921-7/09/$25.00 ©2009 IEEE

to reduce off-chip latencies, like 3D stacking memory architectures [8], and (2) it provides flexibility to the OS for implementing more efficient mapping policies [9], such as *first-touch*, which has been widely used in NUMA architectures to achieve more locality in the memory accesses. The behavior of a first-touch policy is similar to a private cache organization but without replication. One nice aspect of this policy is that it is dynamic in the sense that pages are mapped to cache banks depending on the particular memory access pattern. However, this policy can increase off-chip accesses when the working set of the application is not well-balanced among cores.

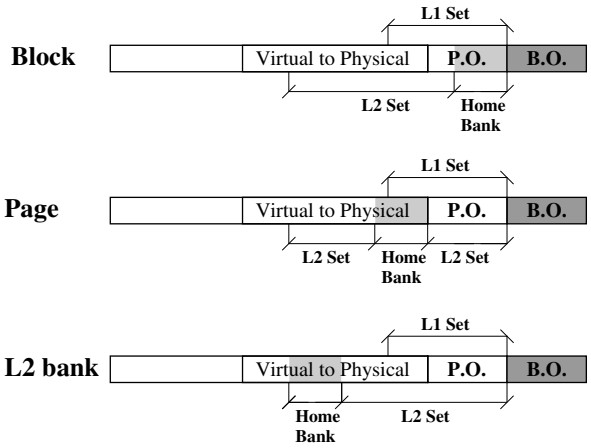
Additionally, many-core architectures are very suitable for throughput computing [10] and, therefore, they constitute a highly attractive choice for commercial servers in which several programs are running at the same time using different subsets of the cores available on chip. The use of these architectures as commercial servers emphasize the need of efficient mapping policies because (1) data is shared by cores that are placed in a small region of the chip, but with a round-robin policy they could map to any bank in the chip, and (2) more working set imbalance can occur in these systems since the applications running on them could have very different memory requirements.

In this work, we propose the *distance-aware round-robin* mapping policy, an OS-managed policy which does not require extra hardware structures. This policy tries to map the pages to the local bank of the first requesting core, like a first-touch policy, but also introduces an upper bound on the deviation of the distribution of memory pages among cache banks, which lessens the number of off-chip accesses.

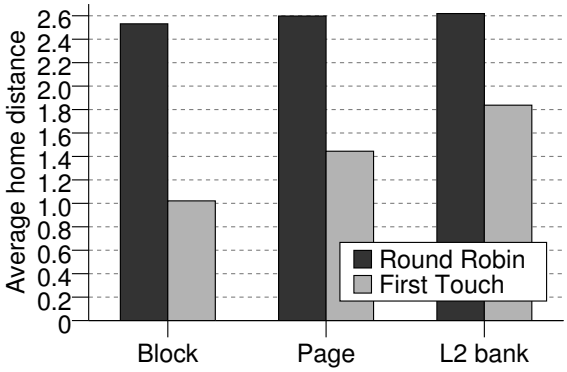
We also observe that OS-managed distance-aware mapping policies can hurt in some cases the L1 cache hit rate. This happens when the same bits that define the home bank are used for indexing the private L1 caches. In these cases, some sets in the cache are overloaded while others remain almost unused. This imbalance increases conflict misses. Hence, we propose to avoid the home bank bits for indexing the L1 caches when distance-aware mapping policies are employed.

Our proposal obtains average improvements of 11% for parallel applications and 14% for multi-programmed workloads over a round-robin policy. In terms of network traffic, our proposal obtains average reductions of 39% for parallel applications and 65% for multi-programmed workloads. When compared to a first-touch policy average improvements of 5% for parallel applications and 6% for multi-programmed workloads are obtained, slightly increasing on-chip network traffic.

The rest of the paper is organized as follows. A background on mapping policies for NUCA caches is given in Section II. Section III describes the distance-aware round-robin mapping policy. The impact of distance-aware mapping policies on private cache miss rate is discussed in Section IV. Section V introduces the methodology employed in the evaluation. Section VI shows the performance results. Section VII presents a review of the related work and, finally, Section VIII concludes the paper.



(a) Different granularities of interleaving (P.O.=Page offset, B.O.=Block offset).



(b) Impact on average home distance for the SPLASH-2 benchmark suite and 16 cores.

Fig. 1. Granularity of L2 cache interleaving and its impact on average home distance.

II. Background on mapping policies in NUCA caches

Non-uniform cache architecture (NUCA) caches [5] are a set of cache banks distributed across the chip and connected through a point-to-point network. Although cache banks are physically distributed, they constitute a logically shared cache (the L2 cache level in this work). Therefore, the mapping of memory blocks to cache entries is not only defined by the cache set, but also by the cache bank. The cache bank where a particular block maps is called the *home* bank for that block.

Most CMP architectures that implement NUCA caches map memory blocks to cache banks by taking some fixed bits of the physical address of the block [1], [2]. This physical mapping uniformly spreads blocks among cache banks, resulting in optimal utilization of the cache storage. Commonly, the bits taken to select the cache bank for a particular block are the less significant ones, leading to a block-grained interleaving (*Block* diagram in Figure 1(a)). One of the advantages of this interleaving is that it offers less contention at the home tile by distributing contiguous memory blocks across different cache banks.

Another option is to use an interleaving with a granularity of at least the size of a page (e.g., *Page* or *L2 bank* diagram in Figure 1(a)). As shown in Figure 1(b), when a physical mapping, or round-robin, policy is considered the granularity of the interleaving does not significantly affect the average distance to the home bank. However, this interleaving becomes an important decision when either 3D stacked memory or OS-managed mapping techniques are considered.

A 3D stacked memory design can offer latency reductions for off-chip accesses when a coarse-grained interleaving (at least of page size) is employed. In tiled CMPs with 3D stacking memory, each tile includes a memory controller for the memory bank that it handles [8]. Low-latency, high-bandwidth and very dense *vertical* links [11] interconnect the on-chip controller with the off-chip memory. These vertical links provide fast access to main memory. On a L2 cache miss, it is necessary to reach the memory controller of the memory bank where the block is stored. If the memory controller is placed in a different tile than the home L2 bank, a *horizontal* on-chip communication is entailed. Since blocks in memory are handled at page-size granularity, it is not possible to assign the same mapping for the L2 cache if a block-size granularity is considered. Differently, with a granularity of at least the size of a page the same mapping can be assigned to both memories, thus avoiding the horizontal latency.

The other advantage of a coarse-grained interleaving is that it allows the OS to manage the cache mapping without requiring extra hardware support [9]. The OS maps a page to a particular bank the first time the page is referenced, i.e, a memory miss. At that moment, the OS assigns a physical address to the virtual address of the page. Therefore, some bits in the address of the page are going to change (*Virtual to Physical* field in figure 1(a)). Then, the OS can control the cache mapping by assigning to this page a physical address that maps to the desired bank. For example, a first-touch policy can be easily implemented by assigning an address that physically maps to the tile wherein the core that is accessing the page resides. The OS only needs to keep in software a list of available physical addresses for each memory bank. With a first-touch mapping policy, finer granularity offers shorter average distance between the missing L1 cache and the home L2 bank, as shown in Figure 1(b). Therefore, it is preferable to use a grain size as fine as possible. Since block granularity is not suitable for OS-managed mapping, the finest granularity possible is achieved by taking the less significant bits of the *Virtual to Physical* field, i.e., a page-grained interleaving.

The drawback of a first-touch policy is that applications with a working set not balanced among cores do not make optimal use of the total L2 capacity. This happens more frequently in commercial servers where different applications with different memory requirements run on the same system, or when some applications are running in a set of cores while the other cores remain idle. To avoid this situation, policies like *cache pressure* [9] can be implemented. Cache pressure uses bloom filters to collect cache accesses in order to determine the *pressure* of the different data mapping to cache banks. In this way, newly accessed pages are not mapped to the most pressured caches. However, this approach has several drawbacks. First, it requires extra hardware, (e.g., bloom filters that have to be reset after a timeout period). Second, an efficient function to detect the pressured cache banks can be difficult to implement. Third, this mechanism only considers neighbouring banks, i.e., banks at 1-hop distance. Finally, as far as we know, neither parallel nor multi-programmed workloads have been evaluated using this technique.

III. Distance-Aware Round-Robin Mapping

In this work, we propose distance-aware round-robin mapping, a simple OS-managed mapping policy for many-core CMPs that assigns memory pages to NUCA cache banks. This policy minimizes the total number of off-chip accesses as happens with a round-robin mapping, and reduces the access latency to a NUCA cache (the L2 cache level) as a first-touch policy does. Moreover, this policy addresses this trade-off without requiring any extra hardware support.

In the proposed mechanism, the OS starts assigning physical addresses to the requested pages according to a first-touch policy, i.e, the physical address chosen by the OS maps to the tile of the core that is requesting the page. The OS stores a counter for each cache bank which is increased whenever a new physical page is assigned to this bank. In this way, banks with more physical pages assigned to them will have higher value for the counter.

To minimize the amount of off-chip accesses we define an upper bound on the deviation of the distribution of pages among cache banks. This upper bound can be controlled by the OS through a threshold value. In this way, in case that the counter of the bank where a page should map following a first-touch policy has reached the threshold value, the page is assigned to another bank. The algorithm starts checking the counters of the banks at one hop from the initial placement. The bank with smaller value is chosen. Otherwise, if all banks at one hop have reached the threshold value, then the banks at a distance of two hops are checked. This algorithm iterates until a bank whose value is under the threshold is found. The policy ensures that at least one of the banks has always a value smaller than the threshold value by decreasing by one unit all counters when all of them have values different than zero.

Figure 2 shows, from left to right, the behavior of this mapping policy for a 2×2 tiled CMP with a threshold value of two. First, processor P_0 accesses a block within page $0x00$ which faults in memory (1). Therefore, a physical address that maps to the bank 0 is chosen for the address translation of the page, and the value for the bank 0 is increased. Then, processor P_1 perform the same operation for page $0x01$ (2). When processor P_1 accesses page $0x00$ no action is required for our policy because there is a hit in the page table (3). The next access of processor P_0 is for a new page, which is also stored in bank 0, which reaches the threshold value (4). Then, if processor P_0 accesses a new page again, this page must be allocated in another bank (5). The closer bank with a smaller value is bank 2. Finally, when processor P_3 accesses a new

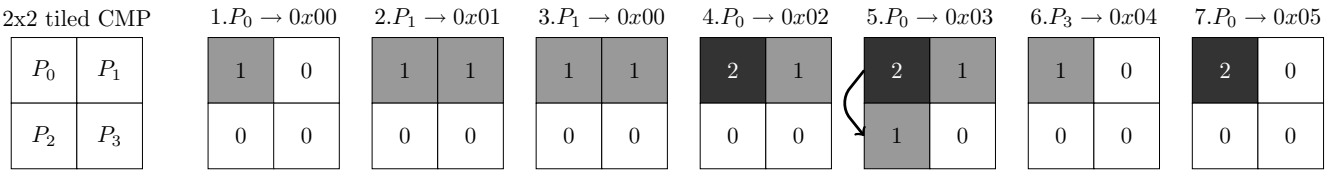


Fig. 2. Behavior of the distance-aware round-robin mapping policy.

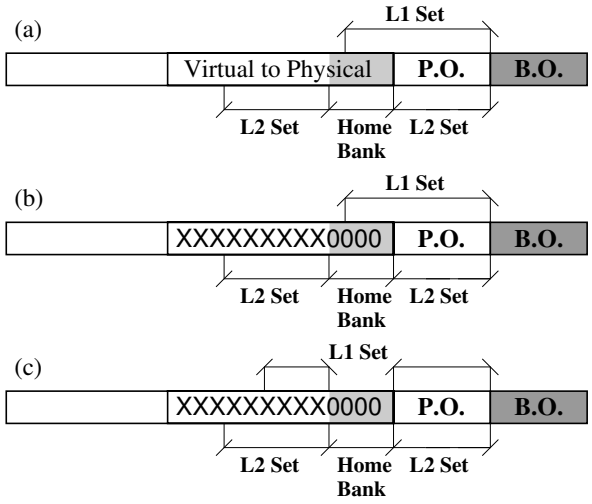


Fig. 3. Changes in the L1 cache indexing policy.

page, the page is assigned to its local bank and all counters are decreased (6), allowing bank 0 to map a new page again (7).

The threshold defines the behavior of our policy. A threshold value of zero denotes a round-robin policy in which a uniform distribution of pages is guaranteed, while an unlimited threshold implies a first-touch policy. Therefore, with a small threshold value, our policy reduces the number of off-chip accesses. Otherwise, if the threshold value is high, our policy reduces the average latency of the accesses to the NUCA cache. Note that the threshold value serves as a proxy approximation for the cache pressure since the actual pressure does not directly depend on the uniform distribution of pages, but on the utilization of blocks within pages. However, pages are distributed among all cache banks, thus performing an efficient use of the shared cache. Although, the OS could choose different thresholds depending on the workload, we have found that values between 64 and 256 work well for the workloads considered in this work.

IV. First-Touch Mapping and Private Cache Indexing

In this section, we study how OS-managed mapping can hurt the hit rate of private L1 caches, mainly when a first-touch policy is implemented. Figure 3(a) shows the cache mapping and indexing used in an OS-managed mapping policy. As mentioned in Section II, it is important to choose the smallest granularity (the less significant bits of the virtual to physical field), to achieve shorter average distance to the home bank. On the other hand, the bits used to index the private caches, i.e, to select the set for a particular block, are commonly the less significant bits of the block address. When the number of bits used to index the L1 cache is greater than the number of bits of the page offset two main issues appear. First, no virtual indexing can be used to accelerate the L1 cache access [12]. Second, the L1 hit rate can be reduced as consequence of the changes in the assignment of physical addresses.

A first-touch mapping policy tries to map blocks frequently requested by a processor to its closest (local) cache bank. This is carried out by the OS when it assigns the physical address to the requested page (e.g., Figure 3(b) represents a physical address that maps to the bank 0). Therefore, most of the blocks that the processor's private L1 cache holds have these bits with the same value. If some of the bits used for selecting the home tile are also used for indexing private L1 caches (Figure 3(b)), most of the blocks will map to a specific range of the L1 cache sets, while other sets will remain under-utilized. This factor increases the number of conflict misses in the L1 cache.

This problem also arise with the mapping policy proposed in this work. The closer our policy is to a first-touch policy (high threshold value) the more set imbalance will occur in private caches. Therefore, we propose to avoid the bits used to define the home tile when indexing private caches, as shown in Figure 3(c). This change allows for better utilization of the private L1 caches, which in turn results in higher L1 cache hit rates, as we show in Section VI-A.

V. Simulation Environment

We have evaluated our proposals using the Simics full-system multiprocessor simulator [13] extended with GEMS 1.3 [14] and SiCoSys [15]. GEMS provides a detailed cache coherent memory system timing model. SiCoSys simulates a detailed interconnection network that allows one to take into account most of the VLSI implementation details with high precision but with much lower computational effort than hardware-level simulators.

Besides the policy already provided by GEMS, a physical mapping with block-grained interleaving that we call *Block-RoundRobin*, we have implemented the other three OS-managed policies evaluated in this work. The first one, named as *Page-RoundRobin*, is an OS-managed policy that assigns physical pages in a round-robin fashion to guarantee the uniform distribution of pages among cache banks. Therefore, this policy does not take into consideration the distance to the

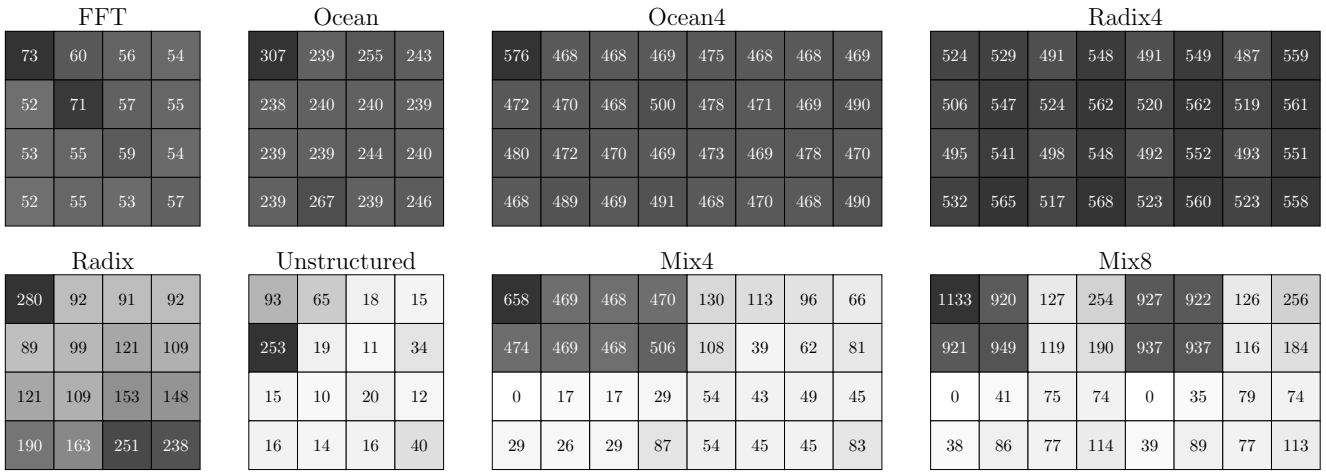


Fig. 4. Number of pages mapped to each cache bank in a first-touch policy for the workloads evaluated in this work.

TABLE I
SYSTEM PARAMETERS.

Memory Parameters: GEMS (4GHz)	
Cache block size	64 bytes
Split L1 I & D caches	64KB, 4-way
L1 cache hit time	3 cycles
Shared unified L2 cache	512KB/tile, 16-way
L2 cache hit time	6 cycles
Memory access time	300 cycles
Page size	4KB
Network Parameters: SICOSYS (2GHz)	
Topology	2-dimensional mesh
Switching technique	Wormhole
Routing technique	Deterministic X-Y
Data and control message size	4 flits and 1 flit
Routing time	1 cycle
Switch time	1 cycle
Link latency (one hop)	2 cycles
Link bandwidth	1 flit/cycle

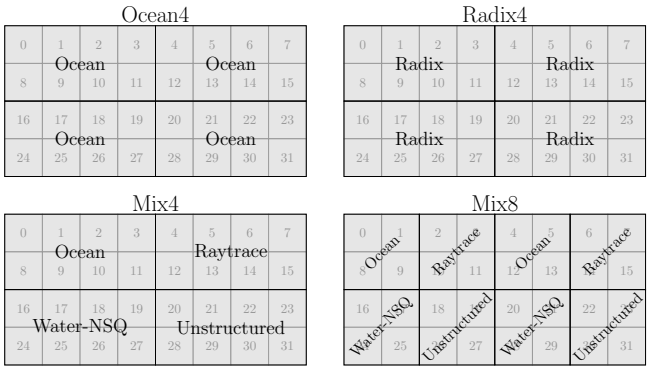


Fig. 5. Multi-programmed workloads evaluated in this work.

home bank. The second one, named as *Page-FirstTouch*, maps memory pages to the local cache bank of the first processor that requested the page. Although this policy is distance-aware, it is not concerned about the pressure on some cache banks. Finally, we also implement the policy proposed in this work. We simulate our proposal with threshold values ranging from 2^0 to 2^{10}. We call our policy *Page-DARR-T*, where T is the threshold value. In addition, we have implemented a cache indexing scheme that skips the bits employed for identifying the home bank, as explained in Section IV.

The simulated system is a tiled CMP in which each tile contains an in-order processor core since a large number of simple cores can offer better performance/Watt ratio than a small number of complex cores. Moreover, a memory controller connected to a 3D-stacked memory bank is placed in each tile. Table I shows the values for the main parameters of the system evaluated in this work, where cache latencies have been calculated using the CACTI tool [16] for 65nm technology and a processor frequency of 4GHz. Memory blocks stored in the private L1 caches are kept coherent by means of a directory-based cache coherence protocol that uses MESI states.

A. Benchmarking

We have evaluated our proposal with parallel and multi-programmed workloads. Multi-programmed workloads consist of several program instances running at the same time in the system. We classify workloads as either *homogeneous* or *heterogeneous*. Homogeneous workloads uniformly distribute memory pages among cache banks when a first-touch policy is employed. In contrast, in heterogeneous workloads a few banks allocate more pages than the others considering a first-touch policy.

For evaluating the parallel applications we have chosen two homogeneous and two heterogeneous scientific benchmarks. *FFT* (256K complex doubles), with a small working set, and *Ocean* (258x258 ocean), with a larger working set, represent the homogeneous workloads. *Unstructured* (Mesh.2K, 5 time steps), with small working set, and *Radix* (1M keys, 1024 radix), with a larger working set, constitute the heterogeneous workloads. *FFT*, *Ocean* and *Radix* belong to the SPLASH-2 benchmark suite [17] while *Unstructured* is a computational fluid dynamics application [18]. Since, in general, the working set of the scientific benchmarks is small we have shrunk the simulation parameters to 32KB 2-way L1 caches, 128KB 4-

way L2 caches and 16 cores. However, the access latencies are kept unchanged.

Since multi-programmed workloads have bigger working sets, we can fairly simulate a 32-core CMP with the cache sizes shown in Table I. We have simulated the configurations shown in Figure 5, again, two homogeneous and two heterogeneous workloads. *Ocean4* and *Radix4* consist of four instances of the *Ocean* and *Radix* applications, respectively, with eight threads each one, representing homogeneous workloads. *Mix4* and *Mix8* run *Ocean*, *Raytrace* (teapot), *Water-NSQ* (512 molecules, 4 time steps) and *Unstructured*. In *Mix4* each application has eight threads. In *Mix8* two instances of each application are run with four threads each. These two workloads represent the heterogeneous and more common multi-programmed workloads. Figure 4 shows for all the workloads evaluated in this work the number of pages mapped to each bank for a first-touch policy.

We account for the variability in multithreaded workloads [19] by doing multiple simulation runs for each benchmark in each configuration and injecting random perturbations in memory systems timing for each run.

VI. Evaluation Results

This section firstly evaluates the impact of the change in the bits used for indexing private L1 caches, as described in Section IV. On the other hand, to understand the improvements obtained by the distance-aware round-robin mapping policy, we study the average distance to the home cache bank and the number of off-chip accesses, and how a good trade-off in those metrics can reduce the applications execution time. Finally, we study the network traffic required by our proposal since it has serious impact on the energy consumed in the interconnection network.

A. Private cache indexing and miss rate

As discussed in Section IV, an OS-managed mapping policy that tries to reduce the distance to the home bank can increase the miss rate of private L1 caches. In this section, we study this issue and compare the traditional indexing method with the proposed one. Figure 6 shows the L1 miss rate for the workloads evaluated in this work and two indexing methods: the traditional method, that we call *less significant bits*, and the proposed one, named as *skip home bits*. Moreover, the miss rate is shown for a range of threshold values for our policy, from 0, i.e., a round-robin (*RR*) policy, to unlimited threshold, i.e., a first-touch (*FT*) policy.

In general, we can see that the *less significant bits* indexing scheme has worse hit rate than the *skip home bits* indexing scheme when the distance-aware mechanism is more aggressive. However, when the policy tries to guarantee a uniform distribution of pages, the hit rate of the indexing scheme only depends on the locality in the memory accesses of each workload.

Therefore, for the rest of the evaluation we use the *less significant bits* indexing scheme for the round-robin policies and the *skip home bits* indexing scheme for the first-touch and the proposed policy, which are the best schemes for each configuration.

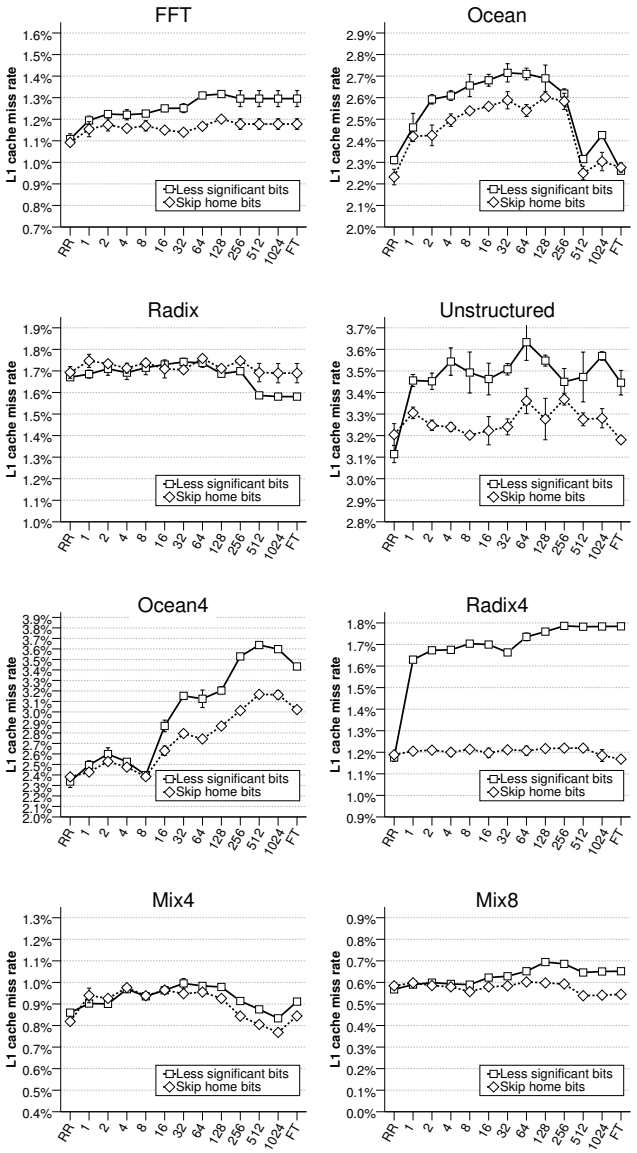


Fig. 6. Impact of the changes in the indexing of the private L1 caches for the workloads evaluated in this work.

B. Average distance to the home banks

Figure 7 plots the average distance in terms of network hops between the tile where the miss takes place and the tile where the home L2 bank is placed. As discussed, a round-robin policy does not care about this distance and, therefore, the average distance for these policies matches up with the average number of hops in a two-dimensional mesh (2.5 for a 4×4 mesh and 3.875 in a 4×8 mesh). On the other hand, the first-touch policy is the one that requires less hops to solve a miss (1.58 for parallel applications and 0.82 for multi-programmed workloads). As can be observed, the results obtained by our

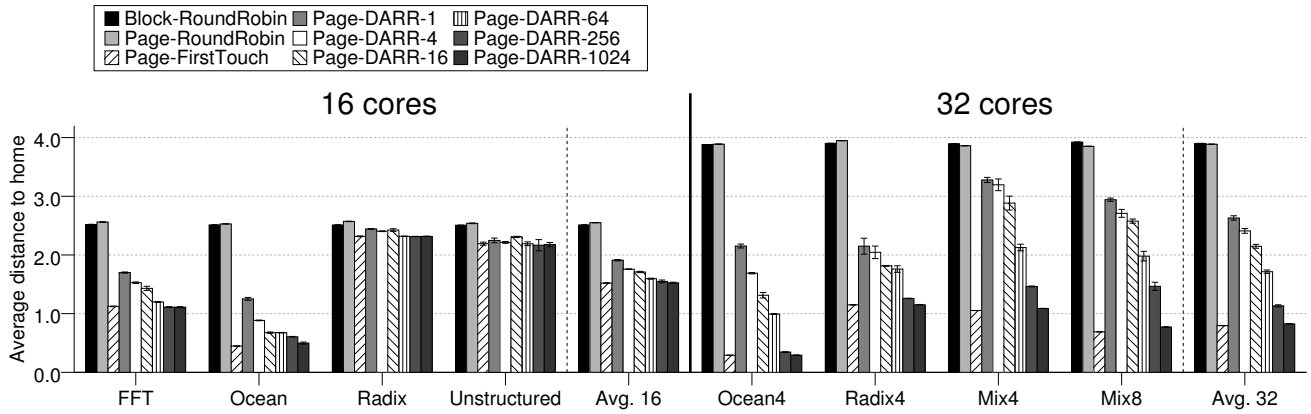


Fig. 7. Average distance between requestor and home tile for the workloads evaluated in this work.

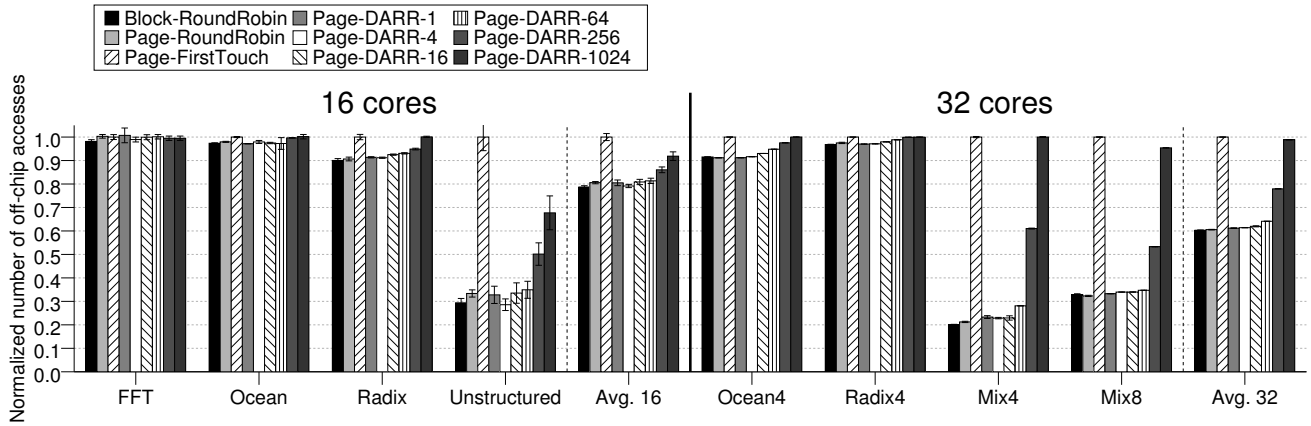


Fig. 8. Normalized number of off-chip accesses for the workloads evaluated in this work.

policy always lie between those of the round-robin and first-touch schemes.

Some parallel applications, like *Radix* and *Unstructured* do not obtain representative reductions in the average distance, even when a first-touch policy is considered. This is because the blocks that frequently cause a cache miss are widely shared. Other applications in which most of the misses are for blocks with a small number of sharers, like *Ocean*, obtain significant reductions in the average distance with a first-touch policy. On the other hand, we can observe that the multi-programmed workloads always achieve important reductions in the average distance. Even when all the applications running in the system are instances of *Radix*, which does not offer reductions in the parallel case, as happens in *Radix4*. This is because data is only shared in the region of the chip running each instance.

Finally, it is important to note that a threshold value of one for our policy reduces the average distance compared to round-robin (by 25% for parallel and 32% for multi-programmed workloads), and also guarantees a uniform distribution of pages. The higher threshold value is employed, the more reductions in the average number of hops are achieved by our proposal.

C. Number of off-chip accesses

The main issue of the first-touch policy is that it incurs in more off-chip accesses specially for workloads that have unbalanced working sets. Figure 8 shows the number of off-chip accesses for the policies evaluated in this work normalized with respect to the first-touch policy. We can observe that for homogeneous workloads the difference in the number of off-chip accesses is minimal. On the other hand, when the working set is not well balanced among the processors, the first-touch policy severely increases the number of off-chip accesses. This increment happens mainly in *Unstructured* ($\approx \times 3$), *Mix4* ($\approx \times 5$) and *Mix8* ($\approx \times 3$). Note that servers usually run a heterogeneous set of applications, like *Mix4* and *Mix8*. Although *Radix* has also a heterogeneous distribution of pages the first-touch policy does not significantly increase the number of off-chip accesses compared to round-robin. This is because its working set is larger than the aggregate L2 cache and, therefore, even when a round-robin policy is used, the number of off-chip misses is high.

Regarding the threshold value of our policy, we can observe that with a value smaller than 256 the number of off-chip accesses is kept very close to the round-robin policy. Finally, when the value is very high, the behavior is close to the first-

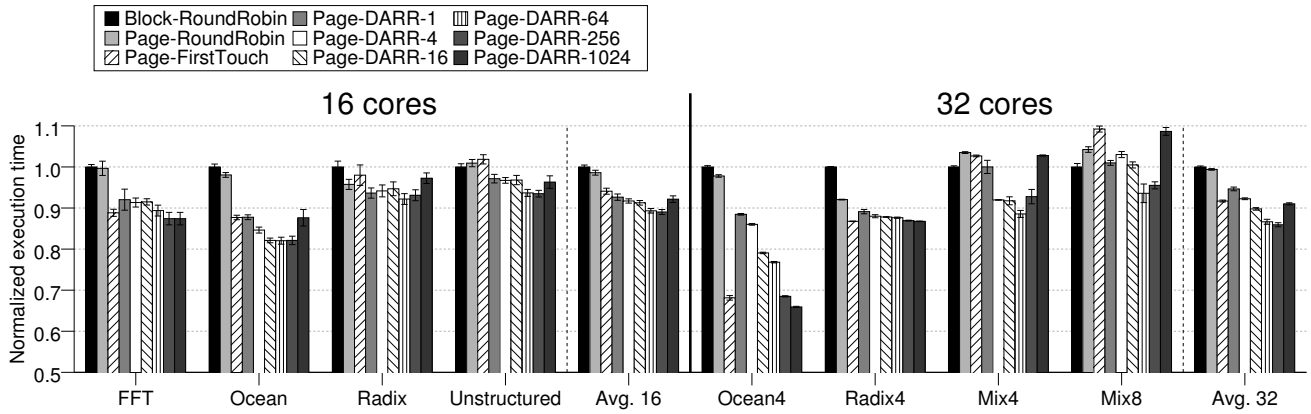


Fig. 9. Normalized execution time for the workloads evaluated in this work.

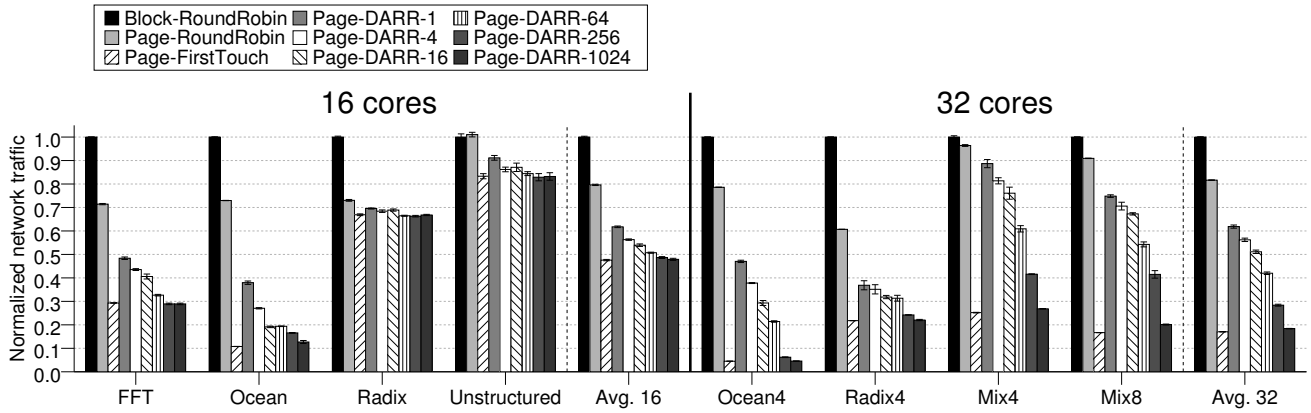


Fig. 10. Normalized network traffic for the workloads evaluated in this work.

touch policy and the number of off-chip accesses becomes prohibitive.

D. Execution time

The results discussed in the previous subsections show that our distance-aware round-robin mapping policy is able to achieve a good trade-off between short distance to the home bank and balanced mapping of memory pages. This achievement results in improvements in execution time as Figure 9 shows.

As we can observe, the first-touch policy achieves important improvements compared to a round-robin policy when the working set accessed by the different cores is homogeneous, as happens in *FFT*, *Ocean*, *Ocean4* and *Radix4*. In contrast, when the distribution of pages accessed by each core is heterogeneous, as occurs in *Radix*, *Unstructured*, *Mix4* and *Mix8*, the first-touch policy incurs in more off-chip accesses, thus degrading performance. In contrast, our proposal achieves the best of a round-robin policy and a first-touch policy with a threshold value between 64 and 256. In this way, we obtain improvements of 11% on average for parallel applications and of 14% on average for multi-programmed workloads compared to a round-robin policy with page-sized granularity. When compared to a first-touch policy we obtain improvements of 5% for parallel applications and 6% for multi-programmed workloads, but additionally avoiding the performance degradation incurred by the first-touch policy in some cases.

E. Network traffic

Figure 10 compares the network traffic generated by the policies considered in this work. In particular, each bar plots the number of bytes transmitted through the interconnection network (the total number of bytes transmitted by all the switches of the interconnect) normalized with respect to the *Block-RoundRobin* policy. We can see that the round-robin policies lead to the highest traffic levels because the distance to the home bank is not taken into consideration. However, the *Page-RoundRobin* policy leads to less traffic than the *Block-RoundRobin* policy. This reduction comes as consequence of using a 3D-stacked memory design, in which the horizontal traffic generated by L2 cache misses disappears when pages are equally mapped to the L2 cache and the memory.

On the other hand, network traffic can be tremendously reduced when a first-touch policy is implemented. In parallel applications, network traffic is reduced by 40% on average. For multi-programmed workloads the savings are greater (72% on average), since most of the blocks are only accessed by cores

placed in a small region of the chip. The distance-aware round-robin policy proposed in this work always obtains reductions in network traffic compared to the round-robin policy, even when the threshold value is just one. When the threshold value increases, the memory-demanding cores can allocate more pages in its local bank and, therefore, less network traffic is generated. As discussed in the previous subsection, a threshold value between 64 and 256 achieves an optimal compromise between round-robin and first-touch. Now, we can see that with a threshold of 256 the network traffic generated by our proposal is reduced by 39% for parallel applications and 65% for multi-programmed workloads. Obviously, the first-touch policy introduces less traffic than our proposal (3% on average for parallel applications and 31% on average for multi-programmed workloads), at the cost of increasing the number of off-chip accesses.

VII. Related Work

There are several ways of reducing cache access latency in NUCA caches. The most relevant ways are data migration, data replication or to perform an intelligent data mapping to cache banks. Next, we comment on the most important works for these approaches.

Kim *et al.*[5] presented non-uniform cache architecture (NUCA) caches. They studied both a static mapping of blocks to caches and a dynamic mapping based on *spread sets*. In such dynamic mapping, a block can only be allocated in a particular *bank set*, but this bank set can be comprised of several cache banks that act as *ways* of the bank set. In this way, a memory block can migrate from a bank far from the processor to another bank closer if the block is expected to be accessed frequently. Chishti *et al.* [20] achieved more flexibility than the original dynamic NUCA approach by decoupling tag and data arrays, and by adding some pointers from tags to data, and vice versa. The tag array is centralized and accessed before the data array, which is logically organized as distance-groups. Again, memory blocks can reside in different banks within the same bank set. Differently from the last two proposals, Beckmann and Wood [21], considered block migration in multiprocessor systems. They proposed a new distribution of the components in the die, where the processing cores are placed around the perimeter of a NUCA L2 cache. Migration is also performed among cache banks belonging to the same bank set. The block search is performed in two phases, both requiring broadcasting the requests. Unfortunately, these proposals have two main drawbacks. First, there are data placement restrictions because data can only be allocated in a particular bank set and, second, data access requires checking multiple cache banks, which increases network traffic and power consumption.

Zhang and Asanovic [7] proposed victim replication, a technique that allows some blocks evicted from an L1 cache to be stored in the local L2 bank. In this way, the next cache miss for this block will find it at the local tile, thus reducing miss latency. Therefore, all L1 cache misses must look for the block at the local L2 bank before the request is sent to the home bank. This scheme also has two main drawbacks. First, replication reduces the total L2 cache capacity. Second, forwarding and invalidation requests must also check the L2 tags in addition to the L1 tags. Later on, in [22], they proposed victim migration as an optimization that removes some blocks from the L2 home bank when they are frequently requested by a remote core. Now, the drawback is that an extra structure is required to keep the tags of migrated blocks. Moreover, in both proposals, write misses are not accelerated because they have to access the home tile since coherence information does not migrate along with the data blocks.

Differently from all the previous approaches, and closer to ours, Cho and Jin [9] proposed using a page-size granularity (instead of block-size). In this way, the OS can manage the mapping policy, e.g, a first-touch mapping policy can be implemented. In order to deal with the unbalanced utilization of the cache banks, they propose using bloom filters that collect cache access statistics. If a cache bank is *pressured*, the neighbouring banks can be used to allocate new pages. As discussed in Section II, this proposal has several implementation issues (e.g., it is difficult to find an accurate metric to decide whether a cache is pressured or not) and they do not evaluate the cache pressure mechanism with neither parallel nor multi-programmed workloads. In addition, they only distribute pages among neighbouring banks, i.e., at one-hop distance. In contrast, in our proposal pages are distributed among all banks, if necessary, in an easy way and without requiring any extra hardware. On the other hand, they do not care about the issue of the private cache indexing since they use 16KB 4-way L1 caches, in which the number of bits used to index them is smaller than the number of bits of the offset of the 8KB pages considered in that work, and they can use virtually indexed L1 caches. Lin *et al.* [23] applied Cho and Jin's proposal to a real system. They studied the dynamic migration of pages and the high overheads that it causes. Recently, Awasthi *et al.* [24] and Chaudhuri [25] proposed several mechanisms for page migration that reduce the overhead of migration at the cost of requiring extra hardware structures. Unfortunately, since migration of pages entails an inherent cost (e.g., flushing caches or TLBs), this mechanism cannot be performed frequently. Although migration can be used along with our proposal, this work focuses on the initial mapping of pages to cache banks. Finally, Awasthi *et al.* do not consider the private cache indexing issue because they use small caches that can be virtually indexed, and Chaudhuri do not take care about the indexing bits despite one bit matches with the home offset bits.

VIII. Conclusions

In CMP architectures, memory blocks are commonly assigned to the banks of a NUCA cache by following a physical mapping policy in which the home tile of a block is given by a set of bits in the block address. This mapping assigns blocks to cache banks in a round-robin fashion, thus neglecting the distance between the requesting cores and the home NUCA bank for the requested blocks. This issue impacts both cache access latency and the amount of on-chip network traffic

generated, and can become a performance problem for large-scale tiled CMPs. On the other hand, first-touch mapping policies, which take into account distance, can lead to an unbalanced utilization of cache banks, and consequently, to an increased number of expensive off-chip accesses.

In this work, we propose the *distance-aware round-robin* mapping policy, an OS-managed policy which addresses the trade-off between cache access latency and number of off-chip accesses. Our policy tries to map the pages accessed by a core to its closest bank, like in a first-touch policy. However, we also introduce an upper bound on the deviation of the distribution of memory pages among cache banks, which lessens the number of off-chip accesses. This upper bound can be controlled by a threshold. We have observed that our proposal achieves a good compromise between a round-robin and a first-touch policy with a threshold between 64 and 256.

We also show that the private cache indexing commonly used in CMP architectures is not the most appropriate for OS-managed distance-aware mapping policies like a first-touch policy or our policy. When the bits used for selecting the home bank are also used for indexing the private L1 cache, the miss rate of private caches can increase significantly. Therefore, we propose to reduce the miss rate by skipping these bits when private L1 caches are indexed.

Our proposal obtains average improvements of 11% for parallel applications and of 14% for multi-programmed workloads compared to a round-robin policy with page granularity (better improvements are obtained compared to a policy that uses block granularity). In terms of network traffic, our proposal obtains average improvements of 39% for parallel applications and 65% for multi-programmed workloads. When compared to a first-touch policy we obtain average improvements of 5% for parallel applications and 6% for multi-programmed workloads, slightly increasing on-chip network traffic. Finally, one of the main assets of our proposal is its simplicity, because it does not require any extra hardware structure, differently from other previously proposed mechanisms.

ACKNOWLEDGMENT

This work has been supported by European Comission funds under HiPEAC Network of Excellence and under grant "Consolider Ingenio-2010 CSD2006-00046". Alberto Ros is supported by a research grant from Spanish MEC under the FPU national plan (AP2004-3735).

REFERENCES

[1] H. Q. Le, W. J. Starke, J. S. Fields, F. P. O'Connell, D. Q. Nguyen, B. J. Ronchetti, W. M. Sauer, E. M. Schwarz, and M. T. Vaden, "IBM POWER6 microarchitecture," *IBM Journal of Research and Development*, vol. 51, no. 6, pp. 639–662, Nov. 2007.

[2] M. Shah, et al, "UltraSPARC T2: A highly-threaded, power-efficient, SPARC SoC," in *IEEE Asian Solid-State Circuits Conference*, Nov. 2007, pp. 22–25.

[3] M. Azimi, et al, "Integration challenges and tradeoffs for tera-scale architectures," *Intel Technology Journal*, vol. 11, no. 3, pp. 173–184, Aug. 2007.

[4] N. Sakran, M. Uffe, M. Mehelel, J. Dowweck, E. Knoll, and A. Kovacks, "The implementation of the 65nm dual-core 64b merom processor," in *IEEE Int'l Solid-State Circuits Conference (ISSCC)*, Feb. 2007, pp. 106–590.

[5] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," in *10th Int. Conf. on Architectural Support for Programming Language and Operating Systems (ASPLOS)*, Oct. 2002, pp. 211–222.

[6] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler, "A NUCA substrate for flexible CMP cache sharing," in *19th Int'l Conference on Supercomputing (ICS)*, Jun. 2005, pp. 31–40.

[7] M. Zhang and K. Asanovic, "Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors," in *32nd Int'l Symp. on Computer Architecture (ISCA)*, Jun. 2005, pp. 336–345.

[8] G. H. Loh, "3d-stacked memory architectures for multi-core processors," in *35th Int'l Symp. on Computer Architecture (ISCA)*, Jun. 2008, pp. 453–464.

[9] S. Cho and L. Jin, "Managing distributed, shared L2 caches through OS-level page allocation," in *39th IEEE/ACM Int'l Symp. on Microarchitecture (MICRO)*, Dec. 2006, pp. 455–465.

[10] S. Chaudhry, P. Caprioli, S. Yip, and M. Tremblay, "High-performance throughput computing," *IEEE Micro*, vol. 25, no. 3, pp. 32–45, May 2005.

[11] S. Das, A. Fan, K.-N. Chen, C. S. Tan, N. Checka, and R. Reif, "Technology, performance, and computer-aided design of three-dimensional integrated circuits," in *Int'l Symposium on Physical Design*, Apr. 2004, pp. 108–115.

[12] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 4th ed. Morgan Kaufmann Publishers, Inc., 2007.

[13] P. S. Magnusson, et al, "Simics: A full system simulation platform," *IEEE Computer*, vol. 35, no. 2, pp. 50–58, Feb. 2002.

[14] M. M. Martin, et al, "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," *Computer Architecture News*, vol. 33, no. 4, pp. 92–99, Sep. 2005.

[15] V. Puente, J. A. Gregorio, and R. Beivide, "SICOSYS: An integrated framework for studying interconnection network in multiprocessor systems," in *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, Jan. 2002, pp. 15–22.

[16] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, "Cacti 5.1," HP Labs, Tech. Rep. HPL-2008-20, Apr. 2008.

[17] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," in *22nd Int'l Symp. on Computer Architecture (ISCA)*, Jun. 1995, pp. 24–36.

[18] S. S. Mukherjee, S. D. Sharma, M. D. Hill, J. R. Larus, A. Rogers, and J. Saltz, "Efficient support for irregular applications on distributed-memory machines," in *5th Int'l Symp. on Principles & Practice of Parallel Programming (PPoPP)*, Jul. 1995, pp. 68–79.

[19] A. R. Alameldeen and D. A. Wood, "Variability in architectural simulations of multi-threaded workloads," in *9th Int'l Symp. on High-Performance Computer Architecture (HPCA)*, Feb. 2003, pp. 7–18.

[20] Z. Chishti, M. D. Powell, and T. N. Vijaykumar, "Distance associativity for high-performance energy-efficient non-uniform cache architectures," in *36th IEEE/ACM Int'l Symp. on Microarchitecture (MICRO)*, Dec. 2003, pp. 55–66.

[21] B. M. Beckmann and D. A. Wood, "Managing wire delay in large chip-multiprocessor caches," in *37th IEEE/ACM Int'l Symp. on Microarchitecture (MICRO)*, Dec. 2004, pp. 319–330.

[22] M. Zhang and K. Asanovic, "Victim migration: Dynamically adapting between private and shared CMP caches," Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, Tech. Rep., Oct. 2005.

[23] J. Lin, Q. Lu, X. Ding, Z. Zhang, X. Zhang, and P. Sadayappan, "Gaining insights into multicore cache partitioning: Bridging the gap between simulation and real systems," in *14th Int'l Symp. on High-Performance Computer Architecture (HPCA)*, Feb. 2008, pp. 367–378.

[24] M. Awasthi, K. Sudan, R. Balasubramonian, and J. Carter, "Dynamic hardware-assisted software-controlled page placement to manage capacity allocation and sharing within large caches," in *15th Int'l Symp. on High-Performance Computer Architecture (HPCA)*, Feb. 2009, pp. 250–261.

[25] M. Chaudhuri, "PageNUCA: Selected policies for page-grain locality management in large shared chip-multiprocessor caches," in *15th Int'l Symp. on High-Performance Computer Architecture (HPCA)*, Feb. 2009, pp. 227–238.