Coherence in the CMP ERA: Lesson learned in designing a LLC architecture

Sandro Bartolini,

Dipartimento di Ingegneria dell'Informazione
Università di Siena
Via Roma 56, Siena
ITALY
bartolini@dii.unisi.it

Pierfrancesco Foglia, Cosimo Antonio Prete, Marco Solinas
Dipartimento di Ingegneria dell'Informazione
Università di Pisa
Via Diotisalvi 2, Pisa
ITALY
{foglia, prete, solinas}@dii.unisi.it

Abstract: - Designing an efficient memory system is a big challenge for future multicore systems. In particular, multicore systems increase the number of requests towards the memory systems, so the design of efficient on-chip caches is crucial to achieve adequate level of performance. Solutions based on conventional, big sized cache may be improved due to wire delay effects, so NUCA and D-NUCA cache may represents an alternative solution, thanks to their ability to limit such effects. Another important design issue of such systems is related to coherence management: the theory of caches kept coherent via directory based coherence protocols was successful in designing high performance DSM machine, and now must consider the requirements of the new scenario: many cores on a chip, and NUCA organizations. In this paper, we face some of these aspects by presenting a NUCA based last-level cache (LLC) architecture. Such an architecture is based on a D-NUCA scheme, i.e. a banked LLC architecture with a migration mechanism to put frequently accessed data near to the requesting processor. To improve access time to shared copies limited by ping-pong effects, we adopted the copy replication, that allows the replication of shared copies that are requested by processors located on the opposite side of the cache. Finally, we have adapted a directory based, distributed, coherence protocol to a D-NUCA cache with migration and replication. Our resulting cache memory sub-system is more performing than a statically sub-banked LLC. The adoption of all such mechanisms forced us to deal with race conditions that may compromise data coherence inside the chip and the memory and, then, to modify the baseline coherence protocol. This experience demonstrated that, in the multicore era, coherence protocols still must be considered of the utmost importance by researchers and designers when facing the design of such systems.

Key-Words: - CMP systems, wire delay, NUCA, coherence, migration, replication

1 Introduction

The design of an efficient memory hierarchy for Chip MultiProcessor (CMP) [21] systems is even more critical than in the past. In fact, the increasing number of on-chip cores[1] induces an aggregated bandwidth request towards the memory hierarchy which is dramatically increased. While the bandwidth, and especially latency, performance of main memory, even if it is slowly improving, still remains far below the requirements and can potentially slow down the execution of applications. For this reasons, the on-chip cache hierarchy, through its capacity, has an increased role in translating computational potential, and thus memory requirements, into overall CMP performance.

The NUCA (Non Uniform Cache Architecture) cache design paradigm [24] is promising in this scenario because of its intrinsic scalability towards both suitable sizes and high number of access ports, which can match the increasing number of on-chip cores. However, NUCA caches, being based on a banked organization and on an interconnection network between banks, introduce additional design space and management [3], [26] directions to be explored to seek optimal performance.

In order to have an adequate scalability potential with respect to the number of cores, it is worthwhile for CMP systems to follow the principles of directory-based coherency protocols, similarly to the ones successfully adopted in the design of classical DSM systems [13] [20]. Anyway, the new Last Level (LL) cache requirements, in particular the need of addressing wire delay, *force CMP designers to carefully adapt such protocols* as traditional implementations are not suitable in this new LL

[1] Up to 16 cores in currently available in commercial products, for instance in the AMD Opteron Processors family [36].

cache scenario and risk to introduce errors or significant overheads.
In this paper we present the experiences done in such field while designing the LL cache for a high performance CMP architecture. To meet the scalability requirement of the overall design, a distributed directory-based version of the MESI coherence protocol has been adopted. To limit wire delay and bandwidth issues, we rely on a banked organization based on migration, according to a D-NUCA paradigm. In order to boost the baseline performance achieved within the D-NUCA cache, we faced the *false miss* [4] problem by proposing the implementation of a *False Miss Avoidance* (FMA) protocol that doesn't rely on any centralized structure and prevents the overheads occurring when accessing data that is migrating [3]. Then we have adopted a *replication mechanism*, which avoids the *ping-pong* issue deriving from the interaction between data migration and shared data access, in integration with the FMA protocol [19]. One crucial point is that all these proposals, in order to be able to deliver their full potential, need deep modifications to the baseline coherency protocol. The achieved performance results indicate that the resulting memory system outperforms statically last level banked cache organization (up to 15% of CPI improvement) in the considered configurations, and outperforms also conventional D-NUCA solutions, never degrading performance. Such experience highlights that in the multicore era data management policies and coherency protocols are of the utmost importance to obtain highly tuned - high performance systems, and may need to be carefully designed to address the issues introduced by nowadays technologic features and architectural implications.

2 NUCA, Migration and Replication

2.1 NUCA

When scaling semiconductor devices feature sizes, it is known that, due to physical properties, the delays of wires that interconnect modules inside a chip doesn't scale with gate delays [9], and this constitutes a big limiting factor [27] in the performance achievable by traditional single-processor designs. The computer systems design paradigm has then moved to CMP architectures [21] for what concern processors (due also to the power constraint that does not permit to increase the clock rate over a certain threshold), and to a NUCA (Non Uniform Cache Architecture) design paradigm [23][25] for what concern the Last Level Cache architecture.
NUCA caches are multi-banked based, where each bank can be accessed independently from the other, and the banks can be connected via a switched network [35]. In a NUCA cache, the access time to each bank depends on the physical distance of the cache bank from the copy requestor. As such distance can vary for different banks, it will lead to a Non-Uniform access time Cache Architecture.
Such caches outperform conventional monolithic caches (UCA, Uniform Cache Architecture)

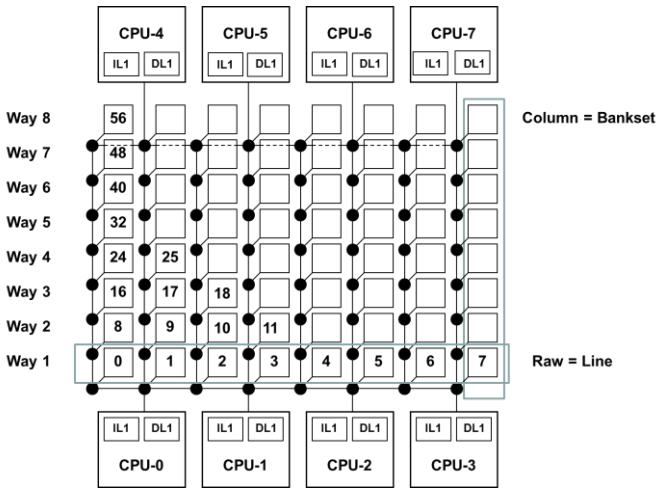

Fig. 1: The reference LLC organization. Squares are the NUCA banks, black circles are NoC switches. Bank columns are the banksets. Rows of the bank matrix are called lines, and are numbered from 1 (at the bottom) to 8. They represent the ways of a conventional set associative cache.

when wire delay effects are significant, also if UCA caches are internally spitted in arrays to optimize power or performance [25], [18]. They can be also more energy efficient [28], [22] than conventional caches.

A NUCA can be a Static NUCA (S-NUCA), when each memory block is mapped only in one bank, or a Dynamic NUCA (D-NUCA), when a memory block is mapped (i.e. can stay) in more cache banks. Such flexible mapping can be exploited to bring the most frequently used copies near the utilizing CPU, thus minimizing the access latency to such data. For this, a migration mechanism and, therefore, a search mechanism are introduced. The migration mechanism is usually a "1hit-1hop", i.e. on a hit the data moves of 1 hop near to the utilizer. The search mechanism usually utilizes broadcast messages, i.e. a request for a data is sent to all the banks that may contain the data [25]. At the cost of increased network traffic and more sophisticated data management techniques, D-NUCA outperform S-NUCA, as evaluated in monoprocessor systems [25]. The NUCA design has been extended to optimize energy efficiency [10][30][31] and performance, it has been proposed also for CMP systems [30][23][4][6][8] also considering the effects of process variation [37]. As a matter of fact, today all the commercial CMPs exhibit a banked organization for the last level cache, that can be UCA or NUCA depending on the way in which banks are connected to the CPU, eventually leveraging NUCA features. For instance, the Power 7 CMP family utilizes the banked L3 cache as a victim for the L2 cache, putting evicted L2 data in banks "near" to the L2 cache originating the data [29].

2.2 Migration and Replication

In the design of the CMP on-chip cache hierarchy for high performance systems, that is in a wire delay dominated environment, one of the main issue is to reduce access latency to the cached data. Besides acting on the interconnection network [35] one solution is represented by reducing the distance, and so the latencies, among requesting processors and data. This can be done by introducing a copy migration mechanism, which is the essence of the D-NUCA, but also by replicating copies, so that each utilizing processor has its own copy near, especially in the case of concurrent access to shared data.

When implementing the migration of data in a CMP environment, some race conditions must be solved, otherwise there should be situations in which copies of the same data but with different values may be present in cache, or update operations may produce error conditions as in Fig. 3.

A consequence of the search and migration mechanism on the same cached copy is the *false miss* problem (Fig. 2, with reference to the system

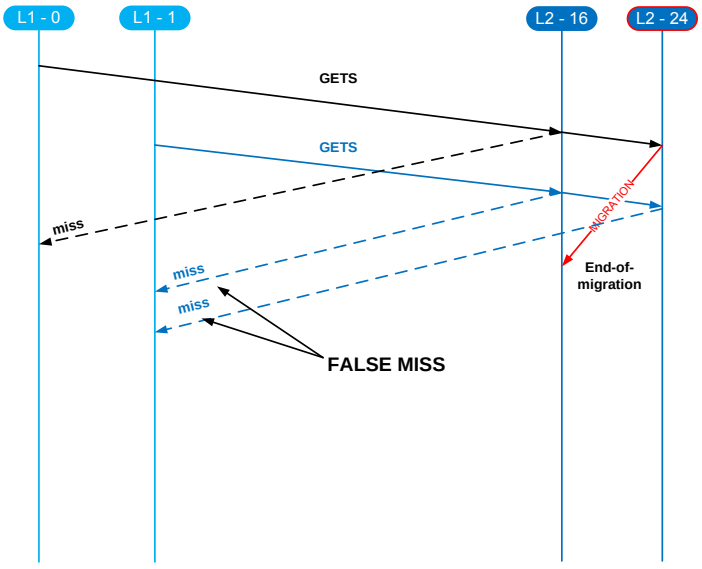

Fig. 2: The False miss problem. L1-0, with reference to the architecture of Fig. 1, generates a request that produces a migration (a data migrates from L2-24 to L2-16) and a simple migration message (MIGRATION) is used. If another cache (L1-1) generates a new request for the same data, and such request arrives at L2-16 before the end of migration, both L2-16 and L2-24 caches reply with a miss, so for L1-1 the data is not present in the cache (FALSE MISS).

of Fig. 1). The effect of such false miss is that a second copy of data is taken from the external memory, and, if the copy in cache has been modified, the two copies are different. The migration may also generate other race condition, related to the management of L1 replacements that involve private blocks in the L2 (i.e. blocks that are owned by L1 caches, according to the MESI behavior), that must be adequately solved by acting on the protocol. In such a case, when an L1 cache (L1-0 in Fig. 3) issues a GETX command to the L2 cache, such request is forwarded by the L2 cache (L2-24) to the owner (L1-1), that will receive the data. Once received the data, if L1-0 needs to replace the data, it must update the content of the L2 cache. But if the block has migrated (from L2-24 to L2-16), L1-0 is no more aware of where the block is located, and sending the put to the previous L2 bank will result in an error, as the bank no more contains the block.

From the performance point of view, the migration mechanism is less effective for those applications that exhibit a high number of accesses to blocks that are alternative accessed (fine grain sharing [42]) by two or more threads located at the opposite sides of the CPU: a thread accesses a block, and the block migrates in the next bank toward the requestor (Fig. 4 A and B). Then, another thread running on a CPU plugged at the opposite side of the D-NUCA (Fig. 1) accesses the same block, that migrates back to the previous bank (Fig. 4 C, D, E). As a consequence, frequently accessed shared blocks alternatively migrate in both directions, and they are never able to be hosted in banks close to the utilizing CPUs. For this reason, the benefits of the reduction of the average NUCA access time can be extremely limited by migration in case of applications that exhibit fine grain sharing. This phenomenon is known as ping-pong [23] or conflict hit [15].

2.3 Related Works

NUCA caches have been initially proposed for monoprocessor systems [25] to limit wire delay effects. The NUCA design has been then extended to optimize energy efficiency [31], [30], [24] and performance [38], and has been proposed also for CMP systems [23][4] [6] [8], [30].

In DSM systems, data migration and replication are solutions utilized to speed-up performance [40], [41]. Their implementation has been explored also in the context of NUCA cache in the CMP environment [3][4][5][6].

Migration for NUCA CMP systems has been proposed in [4], [23]. Beckmann and Wood in [4] propose an 8-cpus CMP system based on a huge shared L2 NUCA cache. They face the false miss problem by i) relaying on an idealized centralized

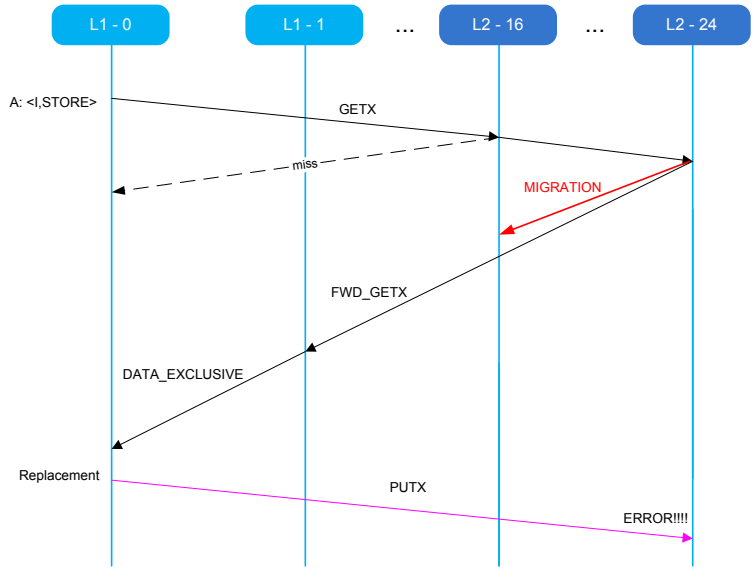

Fig. 3: Race condition in case of replacement of a block owned by a L1 cache. The block is owned by L1-1. As a consequence of a STORE operation, L1-0 issues a GETX command, that hits in L2-24. After the hit, the block migrates from L2-24 to L2-16. Anyway, as the block is owned by L1-1, L2-24 forwards the request to L1-1, which must furnish the data to L1-0. If now L1-0 needs to replace the data, L1-0 must perform a PUTX operation to update the content of L2, so it issues the PUTX command towards L2-24, generating an error condition as L2-24 no more holds the data, with potential loss of the updated data.

off-chip directory embedded in the controller that knows which blocks are actually on-chip, ii) stopping migration (and solving related race conditions) when the directory receives a request for a block that believes being in the cache (false miss detection) and sequential searching for an existing valid copy of the data, iii) delaying block migration of thousand cycles in order to reduce the false miss probability (lazy migration). A centralized directory is adopted also by Huh et al. in [23] for a CMP system in which 16 processors share an L2 NUCA cache, with partial TAGs optimization. Both [4] and [23] observe that for some applications a statically-partitioned shared cache performs better than the corresponding banked cache with migration, due to the conflict hit access pattern.

The replication has been considered in [6], [7], [8], [11]. Chishti et al. [6], propose an hybrid design that takes advantage of both shared and private configuration for the LL cache, and they exploit distance locality by decoupling the data and tag arrays. However their CMP implementation requires a non-scalable atomic bus for maintaining coherence. Cooperative Caching [7] is based on a private cache design with a centralized directory scheme, and considers the aggregate private caches as shared cache space by introducing optimizations such as replication-aware replacement and global replacement of inactive data. ESP-NUCA [8] adopts a dynamic private-shared cache partitioning, with replication and victim management. The ASR mechanism [11] is proposed to optimize CMP performance by controlling the cache replication of memory blocks to minimize potential miss rate increase induced by replication. As a summary, such replication-based schemes adopt a mechanism that controls the number of replicas, as blind replication may affect the LL cache hit-rate, but none of them considers the combined effects of migration and replication.

Finally, Hardavellas et al. [5] propose a tiled architecture in which the aggregation of all L2 banks is seen as a shared NUCA, and adopt a hybrid hard-soft mechanism to control block placement, migration and replication among slices. This, together with others [33], [14], [39] operates at software or at integrated hardware and software level and not at hardware level only.

3 The adopted D-NUCA scheme

The adopted LL cache architecture (L2 in our case) relies on a banked shared cache with migration and replication. In particular, the overall system configuration has a dance-hall architecture [35], where CPUs with private L1 caches are distributed on two opposite sides of the shared L2 NUCA cache (Fig. 1).

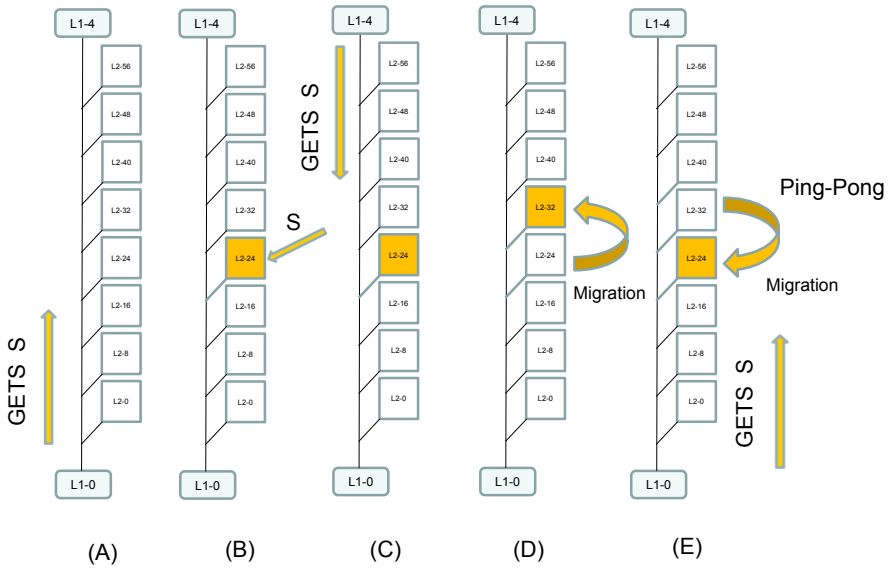

Fig. 4: Conflict hits and the ping-pong phenomenon. L1-0 requests data S (A), that is taken from main memory and put in the L2-24 block (B). Then L1-4, located at the opposite side of the cache with respect to L1-0, requests the same data (C). The data now hits in the cache, and migrates towards L1-4 (the requestor) in the bank L2-32 (D). If L1-0 requests again the data S, it migrates towards L1-0, but it does not improve the access latency with respect to (B), nullifying the benefit of migration.

The topology of the interconnection network among CPUs and cache banks is derived from a 2D mesh by employing only a subset of the links of a full 2D mesh (partial 2D mesh [35], [32]). With such topology, there is only one path that links a CPU (that is, a L1 cache) with a L2 bank, and this simplify in-order point-to-point communication. More complex interconnection and switches infrastructures does not produce performances improvements, and instead cost more in terms of area. Coherence of the private L1 caches is maintained via a directory MESI protocol. By assuming an inclusive cache hierarchy, we avoid the need of a centralized directory, so that the directory is distributed [20], with directory information held together with the TAG field of the block stored in the LLC banks.

As a consequence of migration, a block can be moved to *any* of the *bankset*'s banks [25] (that is, the banks included in a column of banks in Fig. 1), so we assume that, to locate the copy, requests coming from L1 are broadcasted to all the banks of the *bankset*. A block, belonging to a specific section of the memory space, may be stored in any bank (or way) of the bankset. The address of the memory block addresses the specific memory set. The adopted *migration mechanism* considers *per-hit* block migration: a block migration is triggered whenever a request from L1 hits. Data can migrate only along a column of banks, and migration happens towards the requesting L1 cache. This solution constitutes a good trade-off between performance and complexity (search mechanism and coherence management). A smart replication mechanism is adopted to minimize the ping-pong effects produced by applications having fine-grain sharing [34].

3.1 False Miss Avoidance Protocol

In order to face the *false miss* race condition, our solution consists in adding to the baseline MESI protocol a block migration mechanism, called *False Miss Avoidance* or *FMA* [3] protocol, which is implemented completely on chip. In particular, the FMA protocol is able to prevent false misses by guaranteeing that at any time, if a copy exists in the L2 cache, at least *one* bank knows where it is and, therefore, it is able to signal the hit condition and forward the request. For this reason, the FMA lets the two banks involved in the migration transaction communicate via message exchange. In this way, even if a request arrives when the block is *on-the-fly*, it won't result in a miss.
The FMA protocol doesn't deny the other LLC-1 caches to issue new requests for a migrating block, as described in [23], nor relies on any centralized structure and *lazy migration* as in [4]. Fig. 5 shows the main FMA actions: in case of hit in an L2 bank (L2-24 in Fig. 2), the block is provided to the L1 Requestor (L1-0), and a MIGRATION_START message, containing both the block and the directory

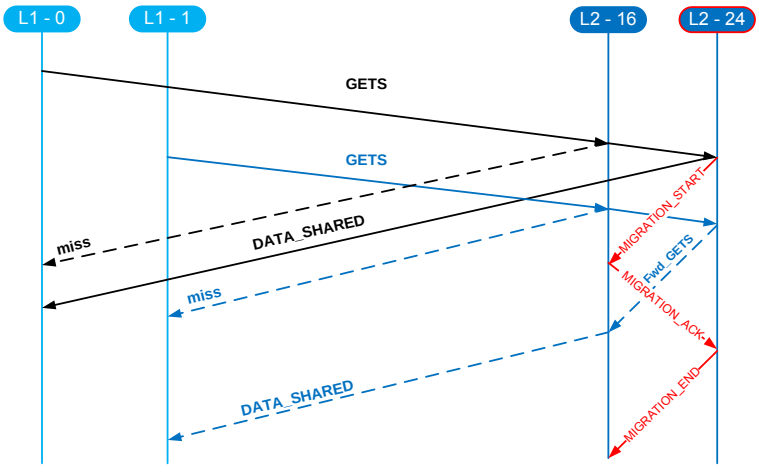

Fig. 5: Sequence diagram representing the block migration process of the FMA protocol. In case of hit in an L2 bank (L2-24), the copy is sent to the L1 Requestor (L1-0), and a MIGRATION_START message, containing both the block value and the directory information, is sent to the L2 Receiver (L2-16). When the L2 Receiver gets the message, allocates a cache line for the block, and replies with a MIGRATION_ACK to the L2 Sender (L2-24), which will conclude the migration process with a MIGRATION_END message. If a new request is received by the L2 Sender while waiting for the MIGRATION_ACK, the request is forwarded to the L2 Receiver, which will serve it.

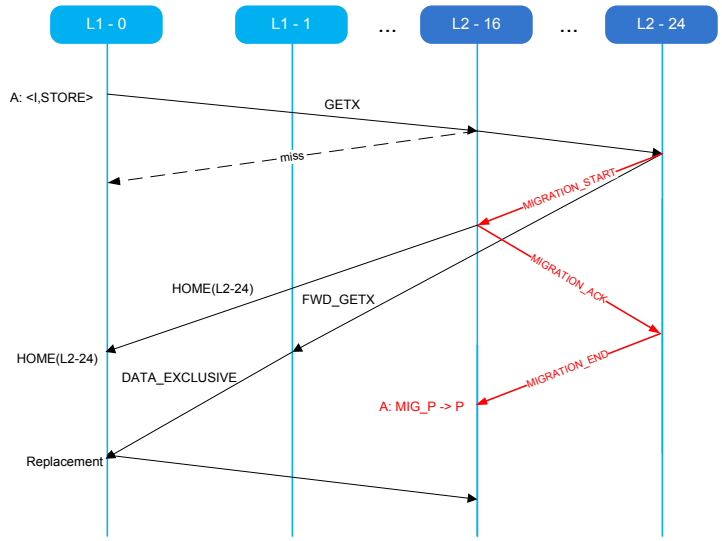

Fig. 6: Sequence diagram representing the protocol behavior in case of replacement of a block owned by an L1 cache. After receiving the MIGRATION_START message, L2-16 keeps L1-0 informed of the new home node, so that L1-0 can send, on replacement, the data (via PUTX) to the right bank.

information, is sent to the destination L2 bank (L2-16). When L2-16 receives the migrating block, it allocates a cache line for the block, and replies with a MIGRATION_ACK to L2-24, which will conclude the migration process with a MIGRATION_END message. If a new request is received by L2-24 while waiting for the MIGRATION_ACK (that is, a potentially false miss condition), the request is forwarded to L2-16, which will serve it. In conclusion, L2-16 replies with a miss until the reception of the MIGRATION_START message; after that it replies with a hit. L2-24 replies with an hit and send a MIGRATION_START message; then it forwards the GETS request until the reception of the MIGRATION_ACK message, with which the block in L2-24 became invalid, so that subsequent requests generate a miss.

The forwarding technique, on which the FMA protocol is based, solves also other race conditions related to the management of L1 replacements that involve private blocks in the L2 presented in the previous section. With the FMA protocol, L2-24 (Fig. 6) forwards the request to the owner and starts the migration with the MIGRATION_START message. L2-16, on receiving such a message, will send an HOME message to the L1 requestor, so that it is informed of the home node change, and can then send the consequent PUTX to the right block.

3.2 Shared Blocks Migration Issues

In order to avoid the *ping-pong* in accessing to shared blocks, a block replication mechanism is also adopted [19]. By considering a compromise between protocol complexity and performance gain, only two copies of a data are permitted in the architecture, each one devoted to serve the accesses of the nearer CPU. In particular, a replica is created when a replica does not exists and a block receives requests coming from a CPU in the opposite site, i.e. only in case of potential conflict hits (Fig. 7). In this way, each copy is able to reach the respective low latency way, and the benefits of migration are always effective. In case L1 needs the exclusive copy of a replicated block, the LLC copy farthest to the requesting CPU will be invalidated.

We called this mechanism *limited replication [34]*, as it allows at most two copies of the same block to be stored in the shared LLC, in contrast to a private caches solution in which more than 2 copies of the same block can be present (up to n, where n is the number of CPUs). As for the *false miss* solution, once again the introduction of the replication mechanism requires to update the coherence protocol for replicas management and coherence. In the following, we describe the main changes introduced by the mechanism, while a complete description of the protocol can be found in [19].

The replication mechanism is implemented via a messages exchange between the bank that receives a request that hits the block (sender), and the bank that will store the replica (receiver). Banks are able to distinguish between near requests (coming from L1s placed at the closest NUCA side) and far requests (coming from L1s placed at the farthest NUCA side), Fig. 7. Both Read-Only and Read-Write data can be replicated.

The sender bank starts the replication process when the request that hits the block is a Read requests (GETS) and the request is recognized as far.
Fig. 8 shows the messages exchanged by the banks involved in the replication. The choice of which L2 bank will host the replica is fixed for each bankset. For example, referring to the bankset 0 in Fig. 1, banks 0, 8, 16 and 24 will send the REPLICATION_START message to bank 40, while banks 32, 40, 48 and 56 will send the message to bank 24. At the end of the replication, two copies of the same block exist. In order to let a copy be aware that it is a replica, an isReplica bit is added to the TAG field of the block. The isReplica bit is set whenever a replication process ends. Besides, both the copies are able to migrate toward the faster way of the respective target side. The migration process is started for a replica whenever a GETS near is received; GETS far are treated as miss.
As it is allowed the replication of both Read-Only and Read-Write blocks, it is possible that a replicated block receives a Read-With-Intent-To-Modify requests (GETX). In this case, in order to guarantee the correctness of memory operations as there are more copies of the same data, an invalidation protocol is introduced that involves both the banks that hold a copy of the block. In particular, the proposed schema invalidates the farthest replica with respect to the L1 requestor and updates the state of the remaining copy, according to the coherence protocol.

4 Performance evaluation

Performance of the systems were evaluated via full-system simulation using Simics [16] and GEMS [17].

Our reference architecture assumes an in-order 8-cpu UltraSparc II ISA-like CMP system.

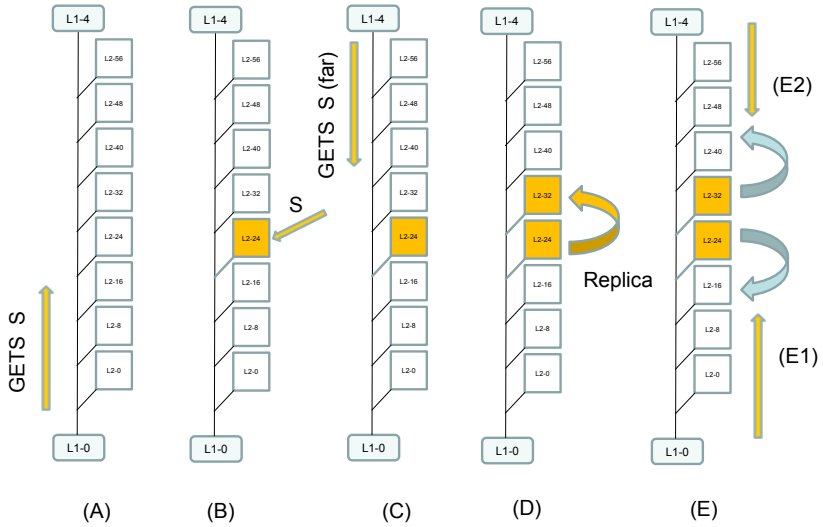

Fig.7: The replication scheme. On a request for a data S from an L1 cache, L1-0 in this case (A), the data is put in the cache from the external memory (B). On a subsequent request from a CPU located at the opposite side with respect to the previous request (L1-4) (C), called a far request, a replica of the data is created (D). The two replicas now can migrate, each one on requests coming from the nearer CPUs (E1 and E2).

Table 1. Configuration Parameters

CPUs	8 cpus (Ultra Sparc II), in-order
Clock Freq.	~4 GHz
L1 cache	Private 16 Kbytes I + 16 Kbytes D, 2 way s.a., one cycle TAG, 2 cycles TAG+Data
L2 cache	16 Mbytes, 64 banks (256 Kbyte banks, 4 way s.a., 5 cycles TAG, 8 cycles TAG+Data)
Block Size	64 bytes
NoC configuration	Partial 2D Mesh Network; NoC switch latency: one cycle; NoC link latency: one cycle; flit size: 32 bits
Main Memory	2 GByte, 240 cycles latency

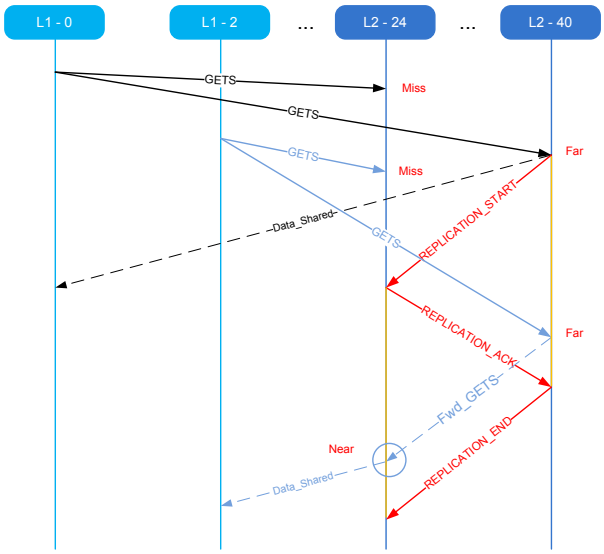

Fig. 8: Sequence diagram of the messages exchanged during block replication. In case of a hit far in an L2 bank (L2-40), the block is sent to the L1 Requestor (L1-0), and a REPLICATION_START message is sent to the L2 Receiver (L2-24). When L2-24 gets the message, it replies with a REPLICATION_ACK to the L2 Sender (L2-40), which will conclude the migration process with a REPLICATION_END message. If a new request is received by L2-40 while waiting for the REPLICATION_ACK (in the figure L1-2 sends a request to L2-40, that without the mechanism would recognize it as far), the request is forwarded to L2-24, which will serve it (L2-24 recognizes the forwarded request as near, and no action is taken by the replication mechanism).

The shared NUCA LL cache is composed by 64 banks organized as in Fig. 1, (each of 256 KB, 4 ways set associative), for a total storage capacity of 16 MB, backed up by a 240-cycles latency main memory. Cache latencies have been obtained by CACTI [18]. The NoC is organized as a partial 2D mesh network, with 64 wormhole switches; NoC link latency has been calculated using the Berkeley Predictive Model [1]. The simulated system runs the Sun Solaris 10 operating system. We run applications from the SPLASH-2 benchmark suite [2], and from the PARSEC 2.0 suite [12]. All the applications were compiled with the *gcc* provided with the Sun Studio 10 suite. Table 1 reports the main configuration parameters.

In the first set of experiments, we compare the performance of a D-NUCA implementing the FMA protocol against a baseline S-NUCA cache. Fig. 9 shows the Normalized CPI with respect to the S-NUCA for the considered benchmarks. In line with other works [4][23], the adoption of a DNUCA is effective, but not for every workload. In particular, we notice different behaviors: a good CPI improvement with the D-NUCA (6% or more for raytrace, blackscholes and swaptions); a little improvement (about 2-4% for barnes, ocean, bodytrack and canneal); and in some cases a performance degradation (radix, radiosity, and streamcluster). To explain such behavior, we can observe that performance improvements are good for those applications that succeed in bringing most frequently accessed blocks in lines 1 and 8 (i.e. in the faster ways), as Fig. 10 demonstrates for raytrace, blackscholes and swaptions, indicating that when migration is not limited by conflict hits, it is successful in boosting performance. For all the other applications, performance improvement are limited by the *ping-pong* phenomenon, that prevents most used data to reach the faster ways, so that the accesses are equally distributed among all the cache lines.

To limit *ping-pong* effects on a D-NUCA, the solution consisted in adopting the replication of blocks. Fig. 11 shows the CPI of the limited replication scheme (R-NUCA), compared to the one achieved by the D-NUCA schemes. Performance of the solution adopting replication are similar to the D- NUCA one for Ocean and bodytrack. For all the other applications, we observe an improvements, which varies from the 3% of Streamcluster up to the 15% of bodytrack. Adding the replication greatly improves the distribution of hits (Fig. 12): for Ocean and Canneal almost 20% of the hits are to ways other than Way8 and Way1. In all the other cases, less than 5% of the hits are to slower ways. Besides, in no case, the replication introduces performance degradation with respect to a D-NUCA: when conflict hit are not discovered, the replication protocol behaves exactly like a D-NUCA.

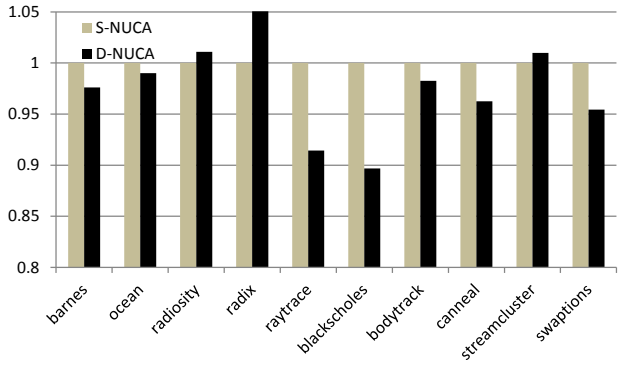

Fig. 9: CPI for S-NUCA and D-NUCA. Data are normalized to the S-NUCA system.

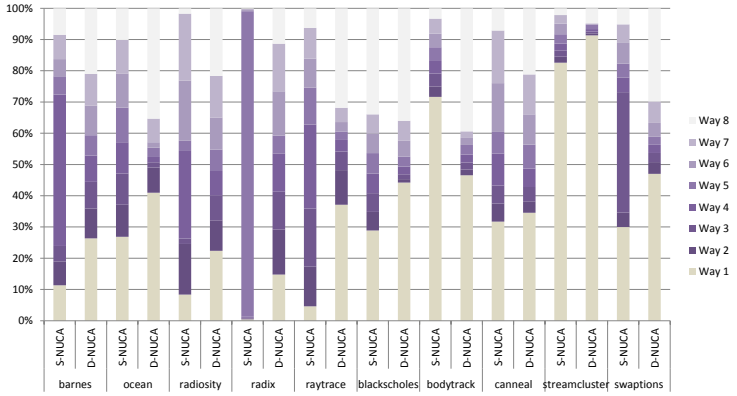

Fig. 10: Hit distribution for S-NUCA and D-NUCA. Lines are numbered according to Fig. 1

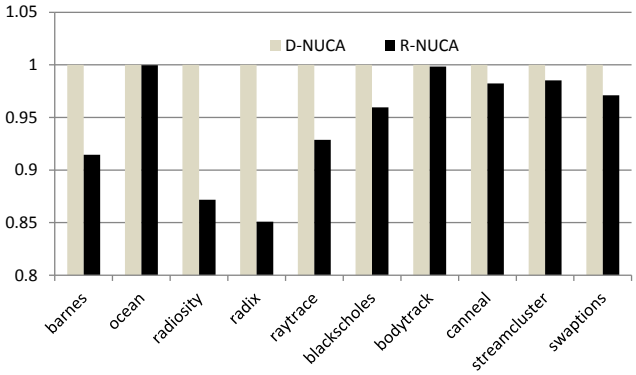

Fig. 11: Normalized CPI for D-NUCA without (D-NUCA) and with (R-NUCA) block replication.

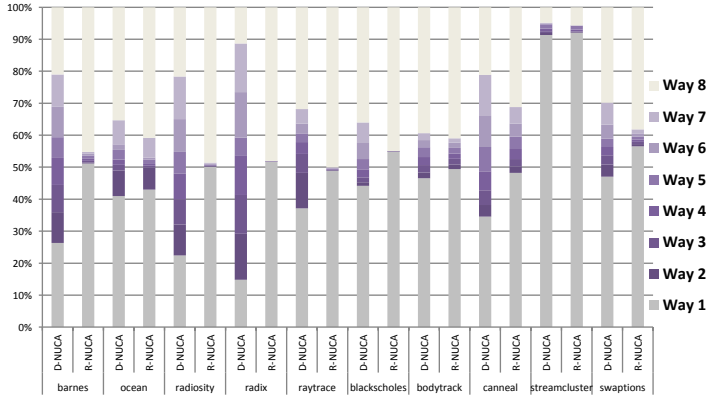

Fig. 12: Hit Distribution for D-NUCA, without (D-NUCA) and with (Re-NUCA) block replication.

5 Conclusions

Technology trends have imposed the advent of CMPs and the proposal of new architectures for the on-chip cache memory subsystems. Such proposals may introduce new race conditions in the coherence protocol, that must be adequately managed to guarantee performance and correctness. In this paper we have shown how a D-NUCA solution, based on migration and replication, can significantly improve the performance with respect to a conventional banked cache, and we have highlighted both the negative effect of migration, both how they can be avoided via replication. Such results have been obtained mainly acting on the coherence protocol. In fact, to deal with race conditions imposed by the new architecture and data management policies, specific actions have been added to the baseline MESI solution (the FMA protocol). To speed-up the performance of the resulting system, again there was the need to modify the coherence protocol, by introducing replication and the states needed to manage it. The resulting system is able to achieve up to 15% improvement of CPI, and, differently from other proposals, never performs worse than a S-NUCA based solution. Such results highlight that data management and coherence protocol can be a significant source of performance improvement also in the multicore era, so they still must be considered of the utmost importance by researchers and designers of such systems.

References:

[1] Predictive Technology Model (PTM), http://www.eas.asu.edu/~ptm/

[2] Woo, S. C., Ohara, M., Torrie, E., Singh, J. P., Gupta, A., The SPLASH-2 programs: characterization and methodological considerations. 22th ISCA, S. Mar. Ligure, Italy, pp.24-36, June 1995.

[3] P. Foglia, F. Panicucci, C.A. Prete and M. Solinas, Analysis of performance dependencies in NUCA-based CMP systems. 21st Int. Symp. on Computer Architecture and High Performance Computing, Sao Paulo, Brazil, pp.49-56, Oct. 2009.

[4] B.M. Beckmann and D.A. Wood, "Managing Wire Delay in Large Chip-Multiprocessor Caches" 37th Int. Symp. on Microarchitecture, Portland, OR, USA, pp. 319-330, Dec. 2004.

[5] N. Hardavellas, M. Ferdman, B. Falsafi, A. Ailamaki, "Reactive NUCA: Near-Optimal Block Placement and Replication in Distributed Caches". 36th Int. Symp. on Comp. Arch., Austin, TX, USA, June 2009, pp. 184-195

[6] Z. Chishti, M.D. Powell and T.N. Vijaykumar, "Optimizing Replication, Communication, and Capacity Allocation in CMPs" 32nd ISCA, Madison, WI, USA, pp. 357-368, June 2005.

[7] J. Chang and G.S. Sohi, "Cooperative Caching for Chip Multiprocessors" 33rd Int. Symp. on Comp. Arch., Boston, MA, USA, June 2006.

[8] J. Merino, V. Puente, J.A. Gregorio, "ESP-NUCA: A Low-cost Adaptive Non-Uniform Cache Architecture". 16th Symp. on High-Perf. Comp. Arch., Bangalore, India, Jan. 2010.

[9] R. Ho, K.W. Mai and M.A. Horowitz, "The future of wires" Proc. of the IEEE, 89(4), pp. 490-504, April 2001.

[10] P. Foglia, M. Comparetti, "A Workload Independent Energy Reduction Strategy for D-NUCA Caches, Journal of Supercomputing, 10.1007/s11227-013-1033-5, October 2013.

[11] B. M. Beckmann, M. R. Marty, D. A. Wood, "ASR: Adaptive Selective Replication for CMP Caches" 39th Int. Symp. on Microarchitecture, Orlando, FL, pp. 443-454, Dec. 2006.

[12] C. Bienia, S. Kumar, J. Pal Singh and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications" 17th PACT. Toronto, Canada, pp. 72-81, Oct. 2008.

[13] K. Gharachorloo, M. Sharma, S. Steely and S. Van Doren, "Architecture and Design of AlphaServer GS320". 9th ASPLOS, Cambridge, MA, USA, pp. 13-24, Nov. 2000.

[14] Q. Lu, et al. "A compile-time data locality optimization framework for NUCA chip multiprocessors". OSU-CISRC-6/08-TR29

[15] A. Bardine, M. Comparetti, P. Foglia, G. Gabrielli and C.A. Prete, "A power-efficient migration mechanism for D-NUCA caches". Design, Automation and Test in Europe, Nice, France, pp. 598-601, Apr. 2009.

[16] Virtutec Simics, http://www.virtutech.com

[17] Winsconsin Multifacet GEMS Simulator, http://www.cs.wisc.edu/gems/

[18] Muralimanohar, N., Rajeev B., and Norman P. Jouppi."CACTI 6.0: A tool to model large caches." HP Laboratories (2009).

[19] P. Foglia, M. Solinas, Exploiting Replication to Improve Performances of NUCA-Based CMP Systems, ACM Transactions on Embedded Computing Systems, Vol. 13, No. 3s, doi: 10.1145/2566568, March 2014.

[20] D. Lenoski, J. Laudon, K. Gharachorloo, A. Gupta and J. Hennessy, "The directory-based cache coherence protocol for the DASH

multiprocessor". 17th ISCA, Seattle, WA, USA, pp. 148-159, May 1997.
[21] K. Olukotun, B.A. Nayfeh, L. Hammond, K. Wilson and K. Chang, "The case for a single-chip Multiprocessor" 7th ASPLOS, Cambridge, MA, USA, pp. 241-251, Oct. 1996.
[22] A.Bardine, P. Foglia, F. Panicucci, J. Sahuquillo, M. Solinas, "Energy Behaviors of NUCA caches in CMPs". 14th EUROMICRO Conference on Digital System Design, Architectures, Methods and Tools, OULU, Finland, pp: 746-753, Aug 31 -Sept. 2, 2011.
[23] J. Hu, C. Kim, l. Zang, D. Burger, S. W. Keckler, "A NUCA substrate for flexible CMP cache sharing", 19th Int. Conf. Supercomputing, Camb. MA, USA, June 2005.
[24] A. Bardine, M. Comparetti, P. Foglia, G. Gabrielli, C. A. Prete, P. Stenstrom, "Leveraging Data Promotion for Low Power D-NUCA Caches". 11th Euromicro Conference on Digital System Design, Parma, Italy, pp. 307-316, Sept. 3-5 2008.
[25] C. Kim, D. Burger and S. W. Keckler, "An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches" 10th ASPLOS, San Jose, CA, USA, Oct 2002.
[26] P. Foglia, F. Panicucci, C. A. Prete, M. Solinas, An Evaluation of Behaviors of S-NUCA CMPs running scientific workload. 12th Euromicro Conf. on Digital System Design, Architectures, Methods and Tools, Aug.t 27-29, Patras, Greece, pp. 26-33, 2009.
[27] Agarwal, V.; Hrishikesh, M. S.; Keckler, S.W.; Burger, D., "Clock rate versus IPC: the end of the road for conventional microarchitectures". 27th Intern. Symp. on Com. Arch, June 2000
[28] A. Bardine. P. Foglia, G. Gabrielli, C. A. Prete, Analysis of Static and Dynamic Energy Consumption in NUCA Caches: Initial Results. ACM MEDEA 2007 Worskhop, Brasov, Romania, pp. 113-120., Sept. 2007.
[29] B. Sinharoy et al: IBM Power7 Multicore Processor, IBM J. RES. & DEV. VOL. 55 NO. 3 PAPER 1, MAY/JUNE 2011.
[30] Z. Chisti, M.D. Powell, T.N. Vijaykumar: "Distance Associativity for High-Performance Energy-Efficient Non-Uniform Cache Architectures" 36th Int. Symp. On Microarchitecture, pp.55-66, Dec. 2003.
[31] P. Foglia, D. Mangano, C. A. Prete, A NUCA Model for Embedded Systems Cache Design, IEEE 2005 Work. on Embedded Systems for Real-Time Multimedia, New York Metropolitan Area, Usa, pp. 41-46, Sept. 2005.
[32] A. Bardine, M. Comparetti, P. Foglia, G. Gabrielli, C. A. Prete, Impact of On-Chip Network Parameters on NUCA Cache Performance. IET Computers & Digital Techniques, Vol. 3(5), pp. 501-512, Aug. 2009.
[33] S. Bartolini, P. Foglia, C.A. Prete, M. Solinas, "Feedback Driven Restructuring of Multi-Threaded Applications for NUCA Cache Performance in CMPs". 22nd Intern. Symp. on Computer Architecture and High Perf. Comp., Oct. 27-30, Petropolis, Brazil, pp. 87-94, 2010.
[34] P. Foglia, C.A. Prete, M. Solinas, G. Monni, "Re-NUCA: Boosting CMP Performance Through Block Replication" 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, 2010, pp.199- 206, 1-3 Sept. 2010.
[35] J. Duato, S. Yalamanchili and L. Ni, Interconnection Networks: an Engineering Approach. Morgan Kauffmann, 2003.
[36] http://www.amd.com/us/products/server/processors/6000-series-platform/6300/Pages/6300-series-processors.aspx#2
[37] A. Bardine, M. Comparetti, P. Foglia, C. A. Prete, Evaluation of Leakage Reduction Alternatives for Deep Submicron Dynamic Nonuniform Cache Architecture Caches, IEEE Transactions on Very Large Scale Integration Systems, vol.22(1), pp.185-190, Jan. 2014.
[38] Z. Chishti, M. Powell, and T. Vijaykumar. Distance Associativity for High-Performance Energy-Efficient Non-Uniform Cache Architectures. 36th Intern. Symp. on Microarchitecture, Wash., DC, USA, 2003.
[39] S. Cho and L. Jin, "Managing Distributed, Shared L2 Caches through OS-Level Page Allocation". 39th M, Orlando, FL, USA, 2006.
[40] L. Noordergraaf, R van der Pas. 1999. Performance experiences on Sun's Wildfire prototype. 1999 Conf. on Supercomputing. ACM, New York, NY, USA, Article 38, 1999
[41] R. Chandra, et al., Scheduling and page migration for multiprocessor compute servers, 6th ASPLOS, Oct. 1994, pp. 12–24.
[42] P. Foglia, R. Giorgi, C. A. Prete, Reducing Coherence Overhead and Boosting Performance of High-End SMP Multiprocessors Running a DSS Workload, Journal of Parallel and Distributed Computing, Vol. 65(3), pp. 289-306, March 2005.