Extending Magny-Cours Cache Coherence

Alberto Ros, *Member*, *IEEE*, Blas Cuesta, *Member*, *IEEE Computer Society*,
Ricardo Fernández-Pascual, María E. Gómez, *Member*, *IEEE Computer Society*,
Manuel E. Acacio, *Member*, *IEEE*, Antonio Robles, *Member*, *IEEE Computer Society*,
José M. García, *Member*, *IEEE*, and José Duato

Abstract—One cost-effective way to meet the increasing demand for larger high-performance shared-memory servers is to build clusters with off-the-shelf processors connected with low-latency point-to-point interconnections like HyperTransport. Unfortunately, HyperTransport addressing limitations prevent building systems with more than eight nodes. While the recent High-Node Count HyperTransport specification overcomes this limitation, recently launched twelve-core Magny-Cours processors have already inherited it and provide only 3 bits to encode the pointers used by the directory cache which they include to increase the scalability of their coherence protocol. In this work, we propose and develop an external device to extend the coherence domain of Magny-Cours processors beyond the 8-node limit while maintaining the advantages provided by the directory cache. Evaluation results for systems with up to 32 nodes show that the performance offered by our solution scales with the number of nodes, enhancing the directory cache effectiveness by filtering additional messages. Particularly, we reduce execution time by 47 percent in a 32-die system with respect to the 8-die Magny-Cours configuration.

Index Terms—High-performance computing, shared memory, cache coherence, directory protocol, coherence extension, scalability, traffic filtering.

1 Introduction and Motivation

In recent years, the market for servers is expanding and changing. The growing number and variety of devices connected to the Internet, the proliferation of new online services and the increasingly demanding user expectations for server responsiveness and availability require more computational power than ever. One established trend to save power, hardware, and administration costs consists in using very powerful machines to run several services on the same physical machine, usually by means of virtualization. An even more recent trend seeks to further reduce costs by outsourcing IT services to cloud computing providers which own and manage clusters of servers that are shared among customers by means of virtualization too. These trends increase the demand for servers with the largest possible computational and storage capabilities.

Until recently, many service providers were able to use clusters of relatively inexpensive PCs to fulfill their task. This kind of clusters are popular also for scientific computing. However, they usually rely on message-passing communications for remote memory accesses. Message-passing increases not only the communication latencies, but also the difficulties to develop efficient applications. The increased programming complexity is undesirable for scientific applications and is unreasonable in the server field.

At the same time, scalable point-to-point interconnect technologies are starting to be included in the server oriented processor offerings of the leading companies. AMD was the first to include such technologies in their Opteron processors with Coherent HyperTransport [1], which was followed by Intel with QuickPath [2] in their Nehalem processors. Unlike previous high-performance interconnects for clusters like InfiniBand [3], the network interface for these new interconnects is included in the same chip as the processor cores and the memory controllers, enabling glueless point-to-point communication between all the processors and memory interfaces in the system and low latency for remote memory accesses. In addition, these technologies provide support for memory coherency.

Recently, AMD has launched six-core versions of its Opteron processors, codenamed Istanbul, and a twelve-core package comprising two dies[1] with six cores each, codenamed Magny-Cours [4]. Besides the increased number of cores, the most notable difference with previous generations of Opteron processors is the inclusion of a directory cache, called *HT Assist Probe Filter* (HTA) [5], which reduces the number of off-chip messages generated by the cache coherence protocol. The Magny-Cours protocol, which is an adaptation of the protocol defined by the coherent HyperTransport (cHT) specification [1], allows to build small cache-coherent shared-memory multiprocessors (up to eight processor dies) in a single board.

Unfortunately, although the HTA reduces cache miss latency and coherence traffic, it has inherited the addressing limitations imposed by the cHT specification, which limits the coherence domain for Istanbul and Magny-Cours to eight dies at most [4]. This limitation prevents the

- *A. Ros, B. Cuesta, M.E. Gómez, A. Robles, and J. Duato are with the Departamento de Informática de Sistemas y Computadores, Universitat Politècnica de València, Escuela Técnica Superior de Informática, Campus de Vera, 46022 Valencia, Spain.*
  E-mail: {aros, blacuesa, megomez, arobles, jduato}@gap.upv.es.
- *R. Fernández-Pascual, M.E. Acacio, and J.M. García are with the Departamento de Ingeniería y Tecnología de Computadores, Facultad de Informática, Universidad de Murcia, Campus de Espinardo, 30100 Murcia, Spain. E-mail: {rfernandez, meacacio, jmgarcia}@ditec.um.es.*

Manuscript received 13 Dec. 2010; revised 18 Feb. 2011; accepted 20 Feb. 2011; published online 17 Mar. 2011.
Recommended for acceptance by R. Gupta.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2010-12-0683.
Digital Object Identifier no. 10.1109/TC.2011.65.

[1] We indistinctly refer to die as node.

development of cluster-based HPC systems able to offer large cache-coherent shared-memory address spaces, such as the SGI Ultraviolet (Altix UV) [6] machines and the 3Leaf Systems DDC-server [7].

The addressing limitation of the cHT specification is solved in the new High Node Count (HNC) HyperTransport specification [8], which extends the former by encapsulating standard cHT messages into HNC packets. However, current Opteron processors do not implement this extension and have only 3 bits in the HTA to encode the owner of a block. Thus, the coherence domain remains limited to eight dies unless additional external hardware is used.

The main advantage of extending the number of nodes in a coherence domain is that data center servers supporting virtualization solutions will be able to use system resources in a more flexible and efficient way, allowing to define larger virtual domains which better fit the requirements of some applications. Besides, it will allow to support HPC applications that currently can only be used in supercomputers and cluster-based computing platforms.

In this work, we present a device, called *bridge chip* or EMC^2 *chip* (Extended Magny-Cours Coherence), that 1) provides a way to efficiently extend the coherence domain provided by the new generation of AMD Opteron processors beyond the 8-die limit, 2) maintains the advantages provided by the HTAs, and 3) filters additional coherence traffic to enhance the HTA effectiveness and scalability [9].

The EMC^2 chip sits in a board with up to seven additional dies. It presents itself as another node to the rest of dies in the same board, while it manages the communication between dies in different boards by performing conversions between cHT and HNC packets. This way, and unlike other extensions (e.g., Horus [10], which was aimed to extend the coherence domain for previous-generation AMD Opteron processors), our proposal agrees with the new HNC standard specification. Every EMC^2 chip includes a directory cache (extended HTA or simply EHTA) that extends the functionality of the local HTAs located in the same board.

We propose three different implementations for the EMC^2 chip that cover a wide set of trade-offs between their area requirements and the amount of filtered traffic. Additionally, we also propose a coherence mechanism that decouples the number of entries of the EHTA from the number of entries of the local HTAs. Finally, to enhance the scalability of the protocol, we propose two approaches that reduce the number of replacements in the HTAs and increase the maximum number of simultaneous pending remote messages allowed in a particular board.

Unlike other multiprocessor systems, such as the SGI Origin [11] or the Cray T3E [12], whose cache coherence protocol was designed from the beginning to scale up to a large number of nodes, our proposal is based on the extension of an existing protocol limited to eight nodes. Therefore, our proposal requires no change in the functionality of the original protocol to overcome its limitations and widen its scalability.

Simulation results show that our proposal allows to build large-scale shared-memory servers based on the new-generation Opteron processors, while being able to exploit the advantages of the HTA at the overall system level. Particularly, the bridge chip named as $EMC^2\text{-}OXSX$ reduces the average execution time of the evaluated applications by 47 percent on average for a 32-die system with respect to the eight-die system allowed by Magny-Cours, while obtaining an excellent compromise between area and traffic requirements. Furthermore, thanks to the EHTA replacement mechanism proposed in the paper that allows to decouple the EHTA size from the size of local HTAs, the area of the EMC^2 chip can be significantly reduced (down to eight times) without noticeable effect on performance. Note that most concepts introduced in this paper for extending cache coherence could also be applicable to other commodity processors.

The remainder of this paper is organized as follows: Section 2 outlines the Magny-Cours cache coherence protocol. We present our proposals for extending AMD Magny-Cours cache coherence capabilities in Section 3. Section 4 discusses two approaches for improving scalability. We describe the simulation environment in Section 5. The evaluation results are presented and analyzed in Section 6. Finally, we draw conclusions in Section 7.

2 AMD Magny-Cours Cache Coherence Support

AMD Opteron processors use the cache coherence protocol defined by the cHT specification [1]. This protocol was designed to perform efficiently in a system with a small number of processors connected with tightly coupled point-to-point HyperTransport links. It can be described as a hybrid between a snoopy and a directory protocol. It is similar to snoopy protocols in the sense that all the nodes see all coherence transactions. However, like directory protocols, it does not rely on a shared bus and can, in fact, be characterized as a directory-based protocol without directory information, also known as $\mathrm{Dir}_0\mathrm{B}$ [13]. This lack of directory information reduces the memory overhead and avoids the latency of accessing it, but it does not filter messages.

On a cache miss occurrence, a node initiates a load or a store transaction by issuing a *request* for a memory block. The requests are sent to the home node (memory controller), which serializes them. On a request arrival, the home node broadcasts messages known as *Broadcast Probes* (BP) in order to invalidate or to obtain the data block from the caches of the other nodes. These nodes reply with *Probe Responses* (PR), which are directed to the requester. Once the requester receives all responses, it sends a *Source Done* (SD) message to the home node, which finalizes the request and proceeds to process the next request for the block (if any). The required BPs do not entail a serious problem in small systems. However, as the number of nodes grows, both the consumed bandwidth and the time required to receive and process all the PRs increases dramatically.

On a write-back of a dirty block, a node sends the modified block in a *VicBlk request* to its home node. This replies with a *Target Done* (TD) message to the requester indicating that the memory has been updated. Like in the previous case, the transaction ends by sending an SD message to the home.

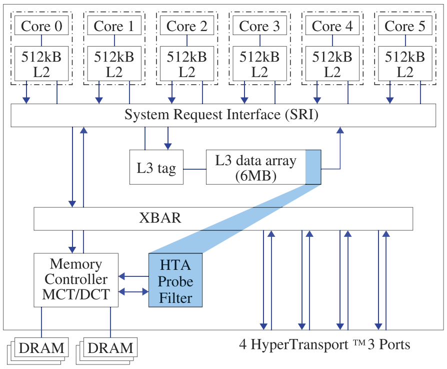


Fig. 1. Block diagram of Magny-Cours dies [4].

Finally, noncached writes, used for noncoherent transactions, are also implemented in the cHT protocol. In this case, a *WrSized request* is initiated and it forces all the memory blocks belonging to a certain memory region to be invalidated from cache and copy-backed to main memory. The requester keeps a clean copy of the data in its cache. A node sends the WrSized request to the home node, which initiates the invalidation of the cached blocks by sending a BP to the other nodes. However, these BPs require that the corresponding region is invalidated from the other nodes and also that these nodes acknowledge the invalidations to the home node instead of to the requester. When the home node has collected all the PRs, it sends a TD message to the requester, which finishes the transaction by sending an SD message back to the home node.

Fig. 1 shows the block diagram of a Magny-Cours die. As shown, Magny-Cours processors add a small on-chip directory cache [5] called *HT Assist Probe Filter* (HTA). The HTA holds an entry for every block mapped to this node cached in the system. Each entry has 4 bytes which are used to store a *tag*, a *state* (EM, O, S1, or S),[2] and a pointer to the current *owner* of the block (3 bits). This information is used to 1) filter unnecessary BPs when no copy of the data is cached and 2) to replace some BPs with unicast *Directed Probe* (DP) messages. In case of a DP, only one response, called *Directed Response* (DR), is generated. Upon a miss on the HTA, a new entry must be allocated, which may require to replace an existing one. Before performing the replacement, all the cached copies of the block identified by the replaced entry must be invalidated either by a DP (if the replaced entry is in EM or S1 state) or by a BP (if it is in O or S state). These invalidations come as a consequence of the lack of a backup memory directory.

As depicted in Fig. 1, a portion (1 MB of 6 MB available) of the L3 cache is dedicated to HTA entries to avoid adding a large overhead in uniprocessor systems. This provides enough space for 256 K entries organized in 64 K four-way sets, which are enough for tracking 16 MB (256 K entries $\times$ 64 bytes/block) of data cached in the system.

Even with the traffic filtering provided by the HTA, the scalability of Magny-Cours systems is limited to eight dies due to implementation details. First, the cHT packet format reserves only 3 bits to identify coherent nodes; and second, the pointer used in the HTA to encode the current owner of a cached block has also 3 bits only (which makes sense since it assumes that cHT will be used).

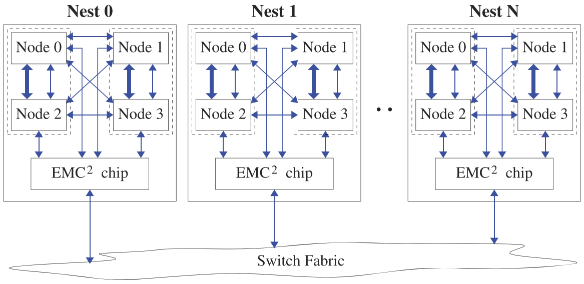


Fig. 2. Overview of the proposed system. Thick arrows inside the nests represent $\times 16$ cHT links while the narrow ones are $\times 8$ cHT links.

The HNC HyperTransport specification partially addresses the first limitation. To this end, it defines the concept of *nest* as any addressable entity (which can be anything from a single processor up to a motherboard containing several processors) and an extended packet format that can encapsulate standard cHT messages and uses a nest-based addressing scheme. However, it does not establish how packets should be handled when they move between nests. To fully overcome these two problems we propose the EMC^2 chip, which is described in the next section.

3 Extending AMD Magny-Cours Cache Coherence Capabilities

Although each nest in our system can contain up to seven processor dies, in this paper, we opt for including only four dies per nest, as illustrated in Fig. 2. This configuration allows the intranest network to be fully-connected and a straightforward mapping of memory blocks to home nodes by checking just a few address bits. Our system comprises several processor boards (referred to as nests). Each nest includes an EMC^2 bridge chip which acts 1) as a network interface controller between nests, 2) as a translator between cHT and HNC packets, and 3) as an extension of the HTAs located inside the nest. Moreover, each nest includes a continuous region of the physical memory.

3.1 Extending the Coherence Domain

To maintain coherence between nodes in different nests, we propose the use of the EMC^2 chip, whose block diagram is shown in Fig. 3. From the point of view of the other nodes, the EMC^2 chip is seen as just another node inside the nest. The EMC^2 chip and all the nodes within a nest are fully connected through a cHT interconnect. The different nests are connected by an InfiniBand switch fabric and they communicate using HNC packets encapsulated into InfiniBand packets.

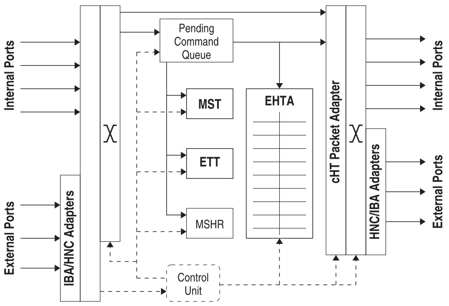


Fig. 3. Block diagram of the EMC^2 chip.

[2] Blocks are stored in caches according to the MOESI states [14].

Transactions in cHT are identified by means of three fields: the id of the node that initiated the transaction (SrcNode), the unit of the node (SrcUnit), and a tag of 5 bits generated at the node (SrcTag). Each cHT packet conveys the information of the transaction that it belongs to. However, this information is only enough within a nest, where the coherence domain is limited to eight nodes. When the coherence domain is extended to several nests, packets must be unequivocally identified out of their local nest (i.e., the nest where the node that initiated the transaction resides). Two new situations can happen: either the packet is traveling from one nest to another one or the packet is in a remote nest.

When the packet is traveling from one nest to another one, it is encapsulated into a HNC packet, which includes an additional field for its identification. This field is the id of the nest where the SrcNode of the transaction is located (SrcNest), and it is included by the EMC^2 chip when it transforms a local cHT packet into a HNC packet. This way, these packets can be globally identified.

On the other hand, packets in a remote nest use the cHT standard, so there is no SrcNest field available for identifying them. Therefore, if a packet is identified by the SrcNode, SrcUnit, and SrcTag of its corresponding transaction, a conflict with a local transaction may occur. To avoid this, EMC^2 chip changes the SrcNode and the SrcTag of the packet when it is transformed from a HNC packet into a cHT packet in a remote nest. In particular, the SrcNode becomes the id of the EMC^2 chip and a new SrcTag is assigned by the EMC^2 chip itself. This way, conflicts between packets belonging to transactions initiated in different nests are avoided.

Another task of the EMC^2 chip is the recovery of the original identifiers of the packets. When an EMC^2 chip receives a cHT packet whose SrcNode corresponds to its node id, it means that the packet is in a remote nest. When it translates the cHT packet into a HNC packet, it has to restore its original identifiers, including the SrcNest. To support this operation, the EMC^2 chip needs to keep a *matching* between the identifiers used in remote nests and the original ones. This information is stored in the *Matching Store Table* (MST) included in each EMC^2 chip. Every packet that goes into a remote nest must allocate an entry in the MST. In the MST, there is an entry for each tag available at the EMC^2 chip. Therefore, the number of entries in the MST is bounded by the maximum number of tags that can be generated by the cHT specification (i.e., 32 tags), which in turn limits the number of external transactions that can be simultaneously in progress inside a nest. Thus, when the MST is full and new entries cannot be allocated, the incoming packets are temporally stored in the *Pending Command Queue*. Possible deadlock scenarios due to the limited number of entries of the MST and their solutions are discussed later in Section 3.4.

While each packet that goes into a remote nest needs to allocate an MST entry, another structure is necessary for storing information about the packets that leave their source nest. This structure is the *Extended Tag Table* (ETT). One of the uses of this structure is to store the home nest of the transactions. This is needed because requests include the block address in the message, so a straightforward calculation can be performed to obtain the destination nest, but other packets, like *Source Done*, do not convey the block address, so the destination nest must be obtained from the ETT. In particular, ETT entries are allocated when a request leaves its source nest and deallocated when its corresponding *Source Done* packet is sent out of this nest. Since the maximum number of concurrent transactions generated by a nest is limited to 512 ($32 \text{ tags/node } \times 4 \text{ units/node } \times 4 \text{ nodes/nest}$), this table will have 512 entries. Thus, unlike the MST, it will be able to store all the transactions requesting an entry.

The EMC^2 chip also has to collect all the responses generated as a consequence of broadcast probes. These responses can be received both from the cHT interface and from the HNC one. The counting of these responses and the data block (if the responses include data), may be temporally stored by the EMC^2 chip in order to be able to generate a single response. This information is stored either in the ETT or in the MST, depending on whether the transaction which these packets belong to was generated in that nest or in another one, respectively.

3.2 Extending the HTA Functionality

To maintain and extend the functionality of the HTAs beyond the nest domain, as well as to reduce the generated coherence traffic, every EMC^2 chip includes a directory cache called *Extended HTA* (EHTA), as shown in Fig. 3. Every EHTA tracks the memory blocks whose home is located in its nest and that may be cached in a remote node (i.e., a node outside its nest). However, the EHTA is not aware of the blocks that are only cached inside its nest.

Since a HTA only knows about the existence of the nodes inside its nest, when a block's owner is a remote node, the HTA will think that the block is cached by the EMC^2 chip. To have precise information of the block's owner, the EHTA will be in charge of tracking the actual location of the owner by storing the nest (*ownerNest* field) and node (*ownerNode* field) identifiers.

In addition to the ownership information about the block, each EHTA entry also includes some information that is used to perform additional traffic filtering tasks. Depending on the quantity of information held by each entry, the filtered traffic and the area requirements will vary. Thus, in order to cover different trade-offs between area requirements and amount of filtered traffic, we propose three configurations for the EHTA entries: $EMC^2\text{-}Base$, $EMC^2\text{-}OXSX$, and $EMC^2\text{-}BitVector$.

- The $EMC^2\text{-}Base$ chip includes an EHTA whose entries encode the ownership of the block and the same states as the HTAs: EM, O, S1, and S (2 bits).
- The $EMC^2\text{-}OXSX$ chip includes an EHTA that encodes two additional states: OX and SX (3 bits). These new states are intended to be able to turn Broadcast Probes into Directed Probes when all the remote copies of a certain block are located in the same nest. Notice that on the arrival to the remote nest, these Directed Probes will be turned again into Broadcast Probes to be able to invalidate more than one copy.
- The $EMC^2\text{-}BitVector$ chip includes an EHTA with the same states as the EMC^2-Base chip, but its entry also holds a bit-vector. This bit-vector includes one bit per every remote nest in the system, indicating

TABLE 1
EHTA States of the EMC^2-$OXSX$ Chip

State	Description
EM	Only the owner's copy is cached outside the home nest. Other copies may be cached inside the home nest.
OX	The owner's copy is cached outside the home nest. Other copies may be cached either in the home nest or in the owner nest.
O	The owner's copy is cached outside the home nest. Other copies may be cached in any nest.
S1	At most one shared copy is cached outside the home nest.
SX	Only shared copies cached outside the home nest, all of them located in the same nest.
S	Only shared copies cached outside the home nest. They can be located in any nest.
I	No valid copy of the block cached outside the home nest.

which of the nests may have a cached copy of the block. This information allows to replace the Broadcast Probes with Multicast Probes. Although this is the most effective configuration in terms of filtered traffic, it is the most area-demanding approach.

Since there are not huge implementation differences among the three proposed configurations, from now on we will just focus on the EMC^2-$OXSX$ chip, which achieves a good traffic-area trade-off (as shown in Section 6.4). Table 1 shows a detailed description of each possible state for the EHTA entries assuming this configuration. Notice that this state only considers copies of the block cached in a remote nest.

Depending on the state in both the HTA and the EHTA, different scenarios can come up, such as Table 2 depicts. For each combination, the table shows a short description of how and where the block is cached and the actions performed by the EMC^2 chip (if any) under load and store transactions. The three possible reactions to a transaction are: 1) no action (the probe is simply forwarded), 2) turning a Broadcast Probe into a Directed Probe, and 3) filtering a Broadcast Probe. The actions in bold are those that entail a reduction in coherence traffic.

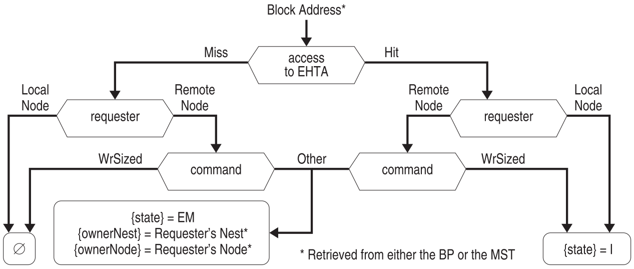


Fig. 4. Updating the EHTA by BPs or PRs for store transactions.

The information in the EHTA must be updated when the caching of the blocks changes. This updating is only performed when the EMC^2 chip receives a packet generated as a consequence of an action performed by some local HTA. The following four sections describe how this information is updated depending on the packet received. Since the EHTA is a cache indexed by the block address and some of the received packets do not carry such information, the MST must be also in charge of storing the address of the block involved in the transaction.

3.2.1 Broadcast Probes and Probe Responses

To update the EHTA while avoiding races, the EMC^2 chip uses the last packet received among the Broadcast Probe and Probe Responses generated as a result of a store or a WrSized transaction. Upon the receipt of this last message, the EMC^2 chip carries out the actions shown in Fig. 4. As depicted, if there is no valid entry for that block in the EHTA (EHTA miss) and a copy is going to be sent outside the home nest (the requester is a remote node) and the message belongs to a store transaction, a new entry is allocated, the state is set to EM, and the block's owner (ownerNest and ownerNode) is set to the requester node. If there is an EHTA miss and the message belongs to a WrSized transaction, the EHTA is not modified since the block will not be cached after the WrSized. If the EHTA already contains an entry for the block and the message

TABLE 2
Scenarios Depending on the HTA State (Rows) and the EHTA State (Columns)

	EM	OX	O	S1	SX	S	I
EM	owner out no copy in/out ld / st:DP→DP	-	-	-	-	-	owner in no copy out ld / st: -
O	owner out no copy out copies in ld:DP→DP **st:BP→DP**	owner out copies in owner nest copies in ld:DP→DP **st:BP→DP***	owner out copies out copies in ld:DP→DP st:BP→BP	owner in 1 copy out copies in ld: - **st:BP→DP**	owner in copies out (1 nest) copies in ld: - **st:BP→DP***	owner in copies out copies in ld: - st:BP→BP	owner in no copy out copies in ld: - **st:BP→Filtered**
S1	-	-	-	owner in memory 1 copy out no copies in ld: - / st:DP→DP	-	-	owner in memory no copy out 1 copy in ld / st: -
S	-	-	-	owner in memory 1 copy out copies in ld: - / **st:BP→DP**	owner in memory copies out (1 nest) copies in ld:- / **st:BP→DP***	owner in memory copies out copies in ld: - / st:BP→BP	owner in memory no copy out copies in ld:- / **st:BP→Filtered**
I	-	-	-	-	-	-	owner in memory no copy in/out ld / st: -

In/out refers to inside/outside the home nest, and ld/st to load/store. DP^* *means that the BP turns into a DP, but only while the DP is transmitted between nests. However, when the DP reaches a nest, the DP is turned into a BP (only inside that nest).*

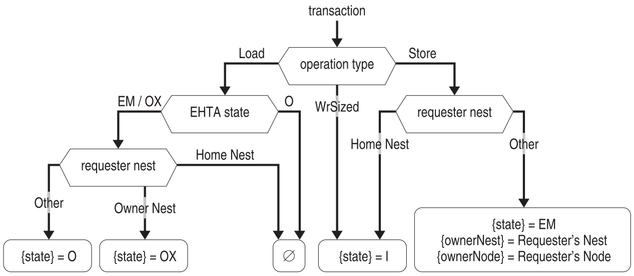


Fig. 5. Updating the EHTA by DPs.

belongs to a store transaction, the EMC^2 chip updates the existing entry accordingly. Finally, when the requester is in the home nest or the message belongs to a WrSized transaction, the EHTA entry is set to invalid because all the external copies will be invalidated.

3.2.2 Directed Probes

Fig. 5 shows how the EHTA is updated on a Directed Probe (DP) arrival. When an EMC^2 chip receives a DP (from inside its nest) due to a load transaction, the owner node must be outside the home nest and, therefore, the EHTA state can only be EM, OX, or O. If the requester is local to the home nest, the coherence information is not modified. If the requester is located in the owner nest and the state field is either EM or OX, all the external copies must be in the same nest and, consequently, the state field is set to OX. If the requester is neither in the home nest nor in the owner nest, the state transitions to O. When the state is O, DPs do not change it.

In case of a store transaction, the EMC^2 chip will receive a DP only when a single external copy of the block exists, which is at the owner node (EM state). In such a case, if the requester node is not in the home nest, the state field transitions to EM and the owner field is set to the requester node (ownerNest and ownerNode). Otherwise, if the requester is in the home nest, the EHTA entry is set to I state because the external copy will be invalidated and forwarded to the requester. Finally, when a DP is received due to a WrSized transaction, the EHTA entry is set to invalid because all copies are going to be invalidated.

3.2.3 Directed Responses

Fig. 6 shows how the EHTA is updated on a Directed Response (DR) receipt. In this case, the owner is located inside the home nest while the requester is outside. If the DR conveys an exclusive copy of a memory block (indicated by the *shared* bit conveyed by DRs), the state transitions to EM and the owner field is set to the requester node. In case the DR carries a shared copy, several actions can take place depending on the state in the EHTA. If the state is I or an EHTA miss occurs, a new entry is allocated setting the state to S1 and storing the requester information in the *ownerNest* and *ownerNode* fields. On an EHTA hit, if the EHTA state is S1 or SX and the requester nest matches the *ownerNest* field, the state is set to SX. If the state is S1 or SX and the requester nest does not matches the *ownerNest* field, the state is updated to S. Finally, if the state is S, it is not changed.

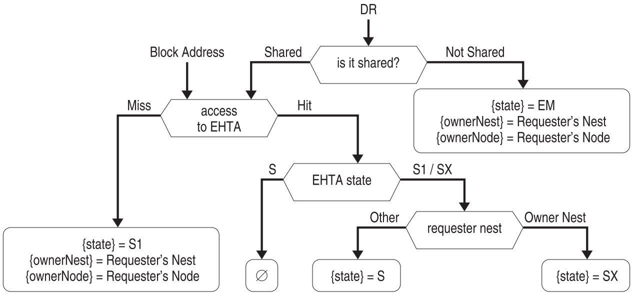


Fig. 6. Updating the EHTA by DRs.

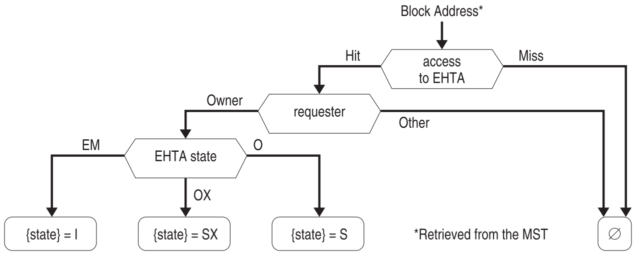


Fig. 7. Updating the EHTA by local TDs.

3.2.4 Target Done Messages

A Target Done (TD) packet can only cause the EHTA to be updated when it has been generated as a result of a cache replacement (VicBlk transaction). Notice that only the owner node can initiate a replacement because the shared copies are evicted silently. Upon the arrival of a TD message, if the owner in the EHTA matches the requester of the VicBlk transaction, the state field is checked. If the state is EM, it transitions to I because the single external copy has been invalidated. If the state is OX, it transitions to SX, and if the state is O, it transitions to S. On the contrary, if the owner in the EHTA does not coincide with the requester, the EHTA is not modified because this can only occur if a race condition happened and the EHTA has already been correctly updated.[3] These operations are depicted in Fig. 7.

3.3 Handling EHTA Replacements

In Magny-Cours, each HTA holds 256 K entries. Therefore, a maximum of 256 K blocks from the same home memory can be cached in the system at the same time. Given that we consider four HTAs per nest, a maximum of 1 M blocks from the same nest could be simultaneously cached. If all those blocks were cached outside the home nest, the EHTA would have to track all of them. To be able to do it without needing evictions, each EHTA would require 1 M entries and an associativity equal to the aggregate associativity of the four local HTAs (i.e., 64 K 16-way sets), assuming it has the same mapping as the HTAs. In order to reduce the EHTA size, and therefore, its access latency, we propose a mechanism for handling EHTA replacements. This mechanism allows the EMC^2 chip to have lower memory requirements.

The eviction of an EHTA entry will entail the invalidation of all the external copies of the block associated to such an entry. However, there are two facts that make the eviction of EHTA entries a bit complicated. First, Magny-Cours dies are

[3] Alternatively, the EHTA state could be updated upon the recepcion of the SD message, which includes information about the success of the VicBlk transaction.

only able to process the coherence messages defined by the cHT protocol [5] and, consequently, new coherence messages cannot be introduced. As the cHT protocol does not include any specific command for performing EHTA evictions, they should be performed by using some of the commands already defined by the cHT protocol. Second, EHTA evictions could introduce complex race conditions if they are not serialized by the home node.

In order to adjust to these two facts, we employ *WrSized* requests to perform EHTA evictions, which are already supported by the cHT protocol. WrSized requests force all memory blocks belonging to a certain memory region to be evicted from cache and copy-backed to main memory. Additionally, the requester keeps a clean copy of the written data in its cache. In case of an EHTA eviction, the memory region indicated by the *WrSized* request is the block whose EHTA entry is going to be replaced. Since EMC^2 chips send *WrSized* requests to the home nodes, this mechanism resolves the serialization problem.

In particular, EHTA replacements are handled as follows. When the EMC^2 chip receives a packet that requires the allocation of a new EHTA entry, as described in the previous section, and the EHTA set for that block is full, the LRU entry of that set must be replaced. To avoid delaying the incoming packet, the evicted EHTA entry is temporally stored in the *Miss Status Hold Register* (MSHR) structure located in the EMC^2 chip, where the information regarding the ongoing transactions is stored. This way, the EMC^2 chip can store the required information in the EHTA and process the incoming packet. If the MSHR is full the incoming packet is stalled.

Then, the EMC^2 chip begins a WrSized transaction. Since this transaction requires a unique identifier, the EMC^2 chip has to assign a new SrcTag to it. This tag cannot be used by any packet in this nest belonging to an external transaction, because in this case two transactions would have the same identifier. Therefore, this tag must be obtained from the free tags in the MST. Since both external transactions and WrSized transaction due to EHTA replacements allocate entries in the MST, deadlock situations could occur if proper care is not taken. We discuss this issue in more detail in Section 3.4.

WrSized transactions are sent to the home node, which is one of the dies within the nest where the EHTA replacement took place. When the home node receives a WrSized request, it issues Broadcast Probes in case the HTA entry for that block is valid. These probes are transmitted by the EMC^2 chip to the remote nests and nodes as previously described. Nodes reply to these probes with the corresponding responses. When the EMC^2 chip, first, and the home memory controller, later, collect all the associated responses, the home node sends a Target Done message to the requester of the WrSized request (i.e., the local EMC^2 chip). At this moment, the EMC^2 chip is allowed to free the MSHR entry and the corresponding MST entries (i.e., the tag is freed). Finally, the WrSized transaction completes by sending a Source Done message to the home node. Now, the block is not stored in any cache. However, the HTA state for the evicted entry is S1, since it assumes that a clean copy is stored in the EMC^2 chip. In order to avoid future broadcast probes as a consequence of the state S1 in the HTA (e.g., upon a write transaction), the EMC^2 chip initiates a clean VicBlk transaction once the WrSized transaction completes. The VicBlk transaction will cause the invalidation of the HTA entry if is found in S1 state.

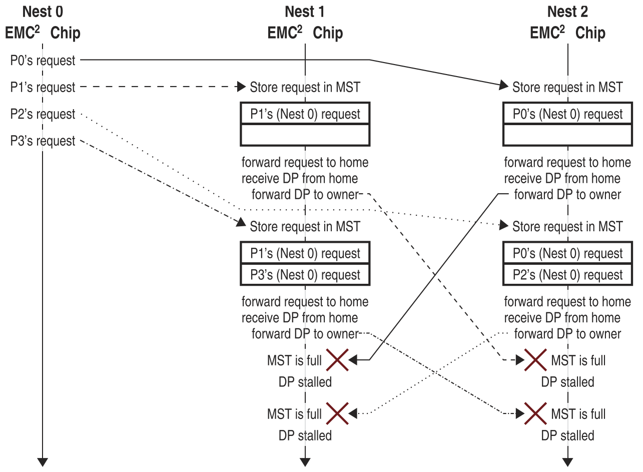


Fig. 8. Probes stalled due to requests stored in the MST. The home node for P0's and P2's requests is placed within Nest 2, whereas the home node for P1's and P3's requests are in Nest 1. Besides, the owner for P0's and P2's requests is in Nest 1, whereas the owner for P1's and P3's requests is in Nest 2.

WrSized transactions force the invalidation of all the copies of the block from cache, even those copies held by nodes within the home nest. However, the invalidation of these internal copies is not strictly necessary, since we only need to invalidate the external copies. The invalidation of the internal copies is a side effect of using transactions already defined by the cHT protocol. Therefore, the proposed mechanism may unnecessarily increase the cache miss rate. Fortunately, we have found that most EHTA evictions correspond to blocks that have no internal copies. As a result, the collateral damage caused by the use of WrSized requests for the EHTA evictions is negligible.

3.4 Deadlock avoidance

The MST is used for assigning an internal tag to any external transaction received by the EMC^2 chip. These transactions can be either a remote request or a remote probe. Like them, the WrSized requests issued by the EMC^2 chip upon an EHTA replacement also need an internal tag, which is also taken from the available ones of the EMC^2 chip (i.e., from the available entries in the its MST). Due to the limited number of MST entries, deadlock situations could arise.

Fig. 8 shows a deadlock scenario where several probes are locked waiting for a free MST entry. Each probe that arrives to a remote nest needs a new SrcTag and, since the MST is full, the probe is stalled. However, the requests that allocated the MST entries cannot progress because their corresponding probes are also stalled. As a result, each request is stalled waiting for another request (which is also stalled) to finish.

Fig. 9 depicts another deadlock scenario where some EHTA replacements are locked because the corresponding WrSized request cannot be issued due to the lack of free MST entries. Again, no MST entry will ever be released since they have been allocated by the requests causing the evictions, and they cannot progress until the eviction is performed.

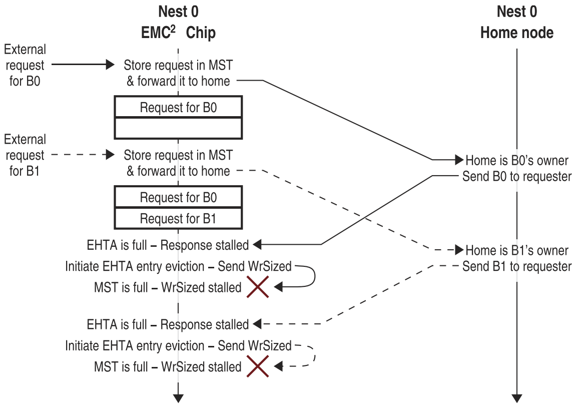

Nest 0
EMC^2 Chip
Nest 0
Home node
External request for B0
Store request in MST & forward it to home
Request for B0
External request for B1
Store request in MST & forward it to home
Request for B0
Request for B1
Home is B0's owner
Send B0 to requester
Home is B1's owner
Send B1 to requester
EHTA is full – Response stalled
Initiate EHTA entry eviction – Send WrSized
MST is full – WrSized stalled
EHTA is full – Response stalled
Initiate EHTA entry eviction – Send WrSized
MST is full – WrSized stalled

Fig. 9. EHTA replacements stalled due to requests stored at the MST. Both B0's and B1's requesters are located outside Nest 0. The home of B0 and B1 is also the owner.

To solve these deadlock scenarios, we first discuss each type of transaction that can allocate an entry in the MST and their mutual dependencies.

- *Broadcast and Directed Probes*: The MST entries allocated by them are valid until the associated responses go back to the EMC^2 chip. They do not depend on the assignation of any tag for a subsequent message.
- *Requests*: The entries allocated by them remain in the MST until the arrival of the corresponding Source Done message. The requests can issue probes to other nests, which could require the assignation of a new SrcTag. Additionally, it can be necessary to evict an entry from the EHTA, which always requires the occupation of a new tag in the same MST as the request.
- *EHTA Replacements*: The entries created by the WrSized requests issued as a consequence of EHTA replacements are released when the corresponding Target Done message is received. A WrSized request never incurs in the replacement of another EHTA entry. However, it may be necessary to send probes to remote nests, which may require in turn the occupation of a MST entry.

According to this, probes do not depend on any other transaction, EHTA replacements only depend on probes, and requests depend on both probes and EHTA replacements. Therefore, by assigning higher priority to the requests with less dependencies and by ensuring at least one MST entry for the requests with higher priority we can avoid the deadlock. Additionally, for a good utilization of the limited number of MST entries, we allow a transaction to occupy any MST entry as long as its priority is higher or equal than the priority of the MST entry. In our particular implementation, we assign four entries to probes (priority 2), four entries to replacements (priority 1), and the remaining entries to requests (priority 0) from the 32 entries available in the MST. This way, replacements can occupy any entry assigned to requests meanwhile probes can occupy any entry.

4 Improving Scalability

In this section, we discuss two possible scalability bottlenecks that could appear when the Magny-Cours coherence protocol is extended to a large number of nodes. These two bottlenecks come as a consequence of constraints present in Magny-Cours: 1) the limited size of the HTA structure which affects the HTA coverage ratio, and 2) the limited number of tags available to identify transactions, which restricts the number of external transaction that can be translated into internal transactions at the same time to only 32. Next sections discuss these issues and propose two alternatives to prevent them from being a bottleneck in large-scale configurations.

4.1 Lessening Worst-Case HTA Coverage Ratio

Each HTA is comprised of 256 K entries for keeping coherence information for its local blocks, i.e., the blocks mapped to its memory controller. On the other hand, the cache hierarchy of each die has 128 K entries (5 MB L3 and 3 MB L2). This means that if all cached blocks were uniformly distributed among the home memory controllers, the coverage ratio of the HTAs would be ×2. This coverage ratio is named as *typical* in [4].

However, Magny-Cours does not assume that the memory is interleaved among the different dies. Therefore, some memory controllers may hold more cached blocks than others. The worst-case scenario appears when all the cached blocks map to the same memory controller. Fortunately, since Magny-Cours systems are comprised of only up to eight dies, the coverage ratio in this case only decreases down to $\times 0.25$, which could be acceptable. Nevertheless, when we extend the coherence mechanism to a larger number of dies, this worst-case coverage ratio falls drastically (down to $\times 0.062$ for a 32-die system), which could result in a significant number of cache invalidations as a consequence of replacements in the HTA. These invalidations may impact negatively on the L3 cache miss rate, which may lead to a significant performance degradation.

A proper interleaving of memory blocks or memory pages would alleviate this issue. However, if we perform a full memory interleaving (i.e., considering all the nodes in the system), sequential applications would have to access memory controllers belonging to remote nests very frequently. Since the internest communication is much slower than intranest communication, these applications would be severely slowed down. A solution for this problem is to perform a hybrid interleaving, where memory is interleaved inside each nest (i.e., among the dies belonging to the same nest), but it is not interleaved among nests. This way, the intranest interleaving lessens the impact of the worst-case coverage ratio by homogeneously distributing blocks among HTAs within the same nest, while the internest contiguous memory addresses avoids accesses to memory controllers in remote nests. Therefore, this approach offers a very good trade-off between coverage ratio and access latency. Note that currently the address mapping functionality of Magny-Cours is not sufficiently flexible to support the proposed hybrid mapping scheme, so it would require an extension of the mapping functionality.

4.2 Increasing the Number of MST Tags

Magny-Cours uses a five-bit field in order to assign ids (SrcTag) to transactions, so there are only 32 tags available per die. Therefore, our MST only has 32 entries, i.e., each

EMC^2 chip can support only 32 internal transactions at the same time. When all the entries of the MST are occupied, the EMC^2 chip cannot issue another message into its nest. Again, as either the system size grows or the EHTA becomes smaller more MST entries are needed because of the larger number of required internal transactions. Therefore, its limited number may become a bottleneck, and consequently, may degrade applications' performance.

We propose to increase the number of available tags by employing the unused die identifiers in the nest. Note that our system configuration has four dies per nest plus one bridge chip, and therefore, there are three die identifiers that are not used in each nest. The utilization of these identifiers would allow us to assign up to 96 additional tags to the MST (128 tags in total). This way, the number of internal coherence transactions that can be generated by the EMC^2 chip at the same time also increases, thus alleviating this possible bottleneck. Obviously, the reduction of this bottleneck comes as consequence of an increase in the size of the MST. However, we will see in Section 6.4 that the area required by the MST is marginal compared to the area required by the EHTA.

TABLE 3
System Parameters

Memory Parameters	
Processor frequency	3.2 GHz
Cache block size	64 bytes
Aggregate L1+L2 caches	3MB, 4-way
L3 cache	5MB, 16-way
Average cache access latency (L1+L2+L3)	2ns
HT assist (probe filter)	1MB, 4-way
HT assist access latency	4ns
EMC^2 chip processing latency	16ns
Memory access latency (local bank)	100ns
Network Parameters	
Intra-nest topology	Fully-connected
Inter-nest topology	Hypercube
Data message size	68 or 72 bytes
Control message size	4 or 8 bytes
HyperTransport bandwidth (16 bits, 6.4GT/s)	12.8GB/s
Inter-die link latency	2ns
Inter-socket link latency	20ns
InfiniBand bandwidth (12x, 10Gb/s)	12GB/s
Inter-nest communication (one way)	150ns
Flit size	4 bytes
Link bandwidth	1 flit/cycle

5 Simulation Environment

We evaluate the proposed extended cache coherence protocol with full-system simulation using Virtutech Simics [15] along with the Wisconsin GEMS toolset [16], which enables detailed simulation of multiprocessor systems. The interconnection network has been modeled using GARNET [17], a detailed network simulator included in the GEMS toolset. Additionally, we have also employed the CACTI 5.3 tool [18], assuming a 45 nm process technology, to measure the area requirements of the different configurations of our proposal.

In order to carry out the evaluation of our proposal, we have first implemented the Magny-Cours cache coherence protocol, which represents the base protocol against which we compare our proposal. Then we have implemented the three different EMC^2 chips explained in Section 3. We have also provided the simulator with the functionality of having several cores per die sharing the same L3 cache. The intradie coherence (L1 and L2) has not been modeled since 1) it is out of the scope of our work and 2) the simulation time would increase considerably. We have run simulations from 8 to 32 dies and with 1 and 2 cores per die. For the Magny-Cours (*MC*) system we only simulate one nest with eight dies, which corresponds to the base case. For the EMC^2 systems, we simulate 4 dies per nest (plus the EMC^2 chip). The parameters assumed for the systems evaluated in this work are shown in Table 3. Since we do not model the intra die protocol or the cache hierarchy, we assume a fixed access latency (representing the average access time) for the whole hierarchy (L1, L2, and L3 caches).

We have evaluated our proposal by using six scientific workloads from the SPLASH-2 benchmark suite [19]: *Barnes* (16 K particles), *Cholesky* (tk16), *FMM* (16 K particles), *Ocean* (514×514 ocean) *Raytrace* (teapot) and *Water-Sp* (512 molecules). All the experimental results reported in this work correspond to the parallel phase of these benchmarks. We account for the variability in multithreaded workloads [20] by doing multiple simulation runs for each benchmark in each configuration and injecting small random perturbations in the timing of the memory system for each run.

6 Evaluation Results

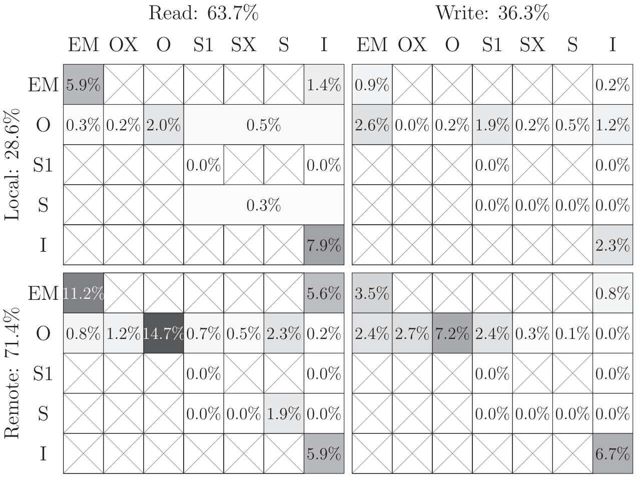
In this section, we show how our proposals are able to support more than eight dies while scaling in terms of execution time. To this end, we compare the three bridge chips proposed in this paper (i.e., *EMC*^{2}-Base, EMC^2-$OXSX$, and EMC^2-$BitVector$) for systems from 8 to 32 dies with a base Magny-Cours system comprised of eight dies. Particularly, we evaluate them in terms of network traffic, cache miss latency, execution time, and area requirements.

Additionally, we study the impact that the hybrid interleaving scheme has on the HTA coverage ratio. We also perform a sensitivity study of the size of the EHTA, demonstrating how it can be significantly reduced without affecting the execution time seriously. Finally, we evaluate the advantages of employing the unused die identifiers in the nest to increase the number of available MST tags.

6.1 Cache Miss Characterization

First of all, it is important to characterize the applications in order to know the fraction of cache misses that can take advantage of the EHTA filtering capabilities. Fig. 10 shows this characterization for a 32-die system that includes the EMC^2-$OXSX$ chip, as a representative example. In this characterization, we show the percentage of misses that fall into each one of the possible combinations of states for the EMC^2-$OXSX$ chip (see Table 2).

The EMC^2 chip can reduce network traffic only when a write miss takes place for a block in O or S states in the HTA (i.e., when a Broadcast Probe is received). This happens for 21.7 percent of cache misses (on average) for the considered applications and a 32-die configuration. Depending on the state in the EHTA, the EMC^2 chip can either completely



		Read: 63.7%							Write: 36.3%						
		EM	OX	O	S1	SX	S	I	EM	OX	O	S1	SX	S	I
Local: 28.6%	EM	5.9%	☒	☒	☒	☒	☒	1.4%	0.9%	☒	☒	☒	☒	☒	0.2%
	O	0.3%	0.2%	2.0%	0.5%				2.6%	0.0%	0.2%	1.9%	0.2%	0.5%	1.2%
	S1	☒	☒	☒	0.0%	☒	☒	0.0%	☒	☒	☒	0.0%	☒	☒	0.0%
	S	☒	☒	☒	0.3%				☒	☒	☒	0.0%	0.0%	0.0%	0.0%
	I	☒	☒	☒	☒	☒	☒	7.9%	☒	☒	☒	☒	☒	☒	2.3%
Remote: 71.4%	EM	11.2%	☒	☒	☒	☒	☒	5.6%	3.5%	☒	☒	☒	☒	☒	0.8%
	O	0.8%	1.2%	14.7%	0.7%	0.5%	2.3%	0.2%	2.4%	2.7%	7.2%	2.4%	0.3%	0.1%	0.0%
	S1	☒	☒	☒	0.0%	☒	☒	0.0%	☒	☒	☒	0.0%	☒	☒	0.0%
	S	☒	☒	☒	0.0%	0.0%	1.9%	0.0%	☒	☒	☒	0.0%	0.0%	0.0%	0.0%
	I	☒	☒	☒	☒	☒	☒	5.9%	☒	☒	☒	☒	☒	☒	6.7%

Fig. 10. Characterization of cache misses according to the HTA (vertical) and EHTA (horizontal) states, read/write misses, and local/remote misses. Results show the average of all the evaluated benchmarks. Crossed cells represent impossible combinations of states. The darker the color of a cell is, the higher the miss percentage is. Multiple cells represent the case where the EHTA has not been reached, and therefore, the EHTA state can be any one of those covered by the cell.

filter the Broadcast Probe or convert it into a single Directed Probe. Note that for the remaining misses, the HTA already filters the unnecessary probes.

6.2 Impact on Network Traffic

In Fig. 11, we show the average number of Broadcast/Directed Probes that arrive to the dies for each Broadcast Probe issued by the home memory controller. This number is plotted for the three EMC^2 chips proposed (for systems with 8, 16, and 32 dies) and the base Magny-Cours system comprised of eight dies. Note that without any filtering this number should be 8, 16, and 32 for 8-, 16-, and 32-die systems, respectively.

Since we only consider Broadcast Probes, the average number of probes arriving to a die in Magny-Cours is always eight. However, for the same system size our protocols reduce this number by filtering some probes. Obviously, when we consider eight dies (i.e., two nests), there is only one remote nest, so all EMC^2 chips behave in the same way. For larger systems, we can see that the more coherence information the HTA stores, the more traffic it filters. Particularly, for a 32-die system, we can see that the average number of received probes is reduced by 23.6 percent (24.4/32), 49.7 percent (16.1/32), and 61.6 percent (12.3/32) for $EMC^2\text{-}Base$, $EMC^2\text{-}OXSX$, and $EMC^2\text{-}BitVector$, respectively.

This reduction in the number of probes received by the dies has two consequences: 1) the number of generated probe responses is also reduced, and 2) the network congestion and the coherence controller utilization decreases. They lead to less time waiting for Probe Responses, and therefore, shorter cache miss latency, which will finally translate into improvements in execution time.

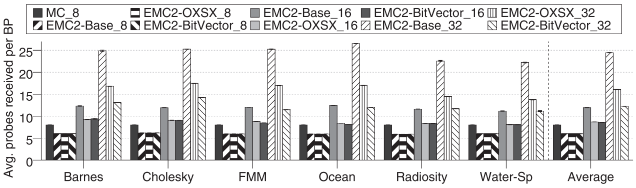


Fig. 11. Number of probes received for each broadcast probe sent by the home die.

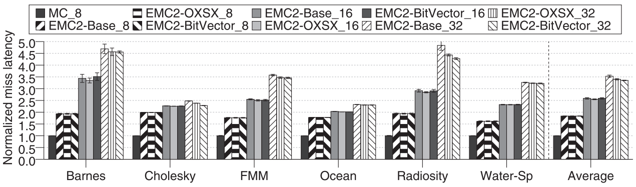


Fig. 12. Normalized miss latency.

6.3 Impact on Execution Time

As we can see in Fig. 12, average miss latency of EMC^2 increases with respect to *MC* for an 8-die system. This is because the latency for transmitting messages between nests is higher than between dies. Remember that in *MC* we have the eight dies in the same nest, whereas in EMC^2 there are only four dies per nest.

On the other hand, when we consider a larger system, the cache miss latency increases due to the growth in the internest communication. Nevertheless, we reduce the final execution time because the applications can be distributed among more dies, which considerably lessens the workload of each die. Finally, we can appreciate a reduction in average miss latency for some EMC^2 chips and the 32-die configuration. Compared to $EMC^2\text{-}Base$, $EMC^2\text{-}OXSX$ reduces the average miss latency by 3.7 percent, and $EMC^2\text{-}BitVector$ by 5.0 percent. The obtained reductions are expected to increase for larger configurations. These reductions in cache miss latency in turn lead to improvements in execution time.

Fig. 13 shows the normalized execution time when we scale up the size of the system. We can see that, although for the 8-die configuration our proposals behave worse than *MC_8* (due to the larger internest latency), when we extend the coherence domain through the bridge chip and allow a higher number of nodes in the system, the execution time of the applications is significantly reduced. Particularly, $EMC^2\text{-}OXSX\_32$ and $EMC^2\text{-}BitVector\_32$ improve the base Magny-Cours system (*MC*) by 47 percent on average. Finally, comparing our three proposals for a 32-die system, $EMC^2\text{-}OXSX$, and $EMC^2\text{-}BitVector$ obtain similar execution time and slightly improve $EMC^2\text{-}Base$ (≈4%).

6.4 Area Requirements

The different EMC^2 chips cover a wide trade-of between memory requirements and filtered traffic. This section studies these tradeoffs for a 32-die configuration.

The three chips differ only in the size of the EHTA entries. Their sizes and those of the ETT and MST are

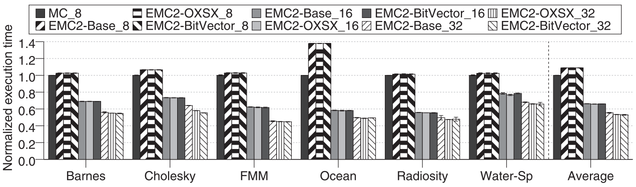


Fig. 13. Normalized execution time.

TABLE 4
Size of the Different EMC^2 Chips for 32-Die Systems (Eight Nests)

	Structure	Entries	Assoc	Entry size (bits)	Area (mm^2)
	ETT	128	1	540	0.64
	MST	32	1	607	0.23
EMC^2-Base	EHTA	1M	16	tag + 8	25.72
EMC^2-OXSX	EHTA	1M	16	tag + 9	25.97
EMC^2-BitVector	EHTA	1M	16	tag + 15	33.38

shown in Table 4. The EHTA of the EMC^2-*Base* is the one that less bits needs per entry (the tag plus 8 bits that include the state, the id of the owner die, and the id of the owner nest). The EHTA of the EMC^2-$OXSX$ needs an extra bit for codifying the two additional states. Finally, the EHTA of the EMC^2-$BitVector$ needs seven extra bits for storing the presence vector for the remote nests.

Fig. 14 plots the trade-off of these three chips in terms of network traffic and area requirements. The total area of each chip has been calculated by adding the areas (in mm^2) of the three main data structures presented in the chip. The normalized network traffic corresponds to the average number of flits transmitted by each switch in the whole system for the six benchmarks evaluated in this work, and normalized to EMC^2-*Base*. We can observe that, EMC^2-$OXSX$ reduces the traffic by 10.6 percent compared to EMC^2-*Base*, while EMC^2-$BitVector$ reduces the traffic by 15 percent. Moreover, the area of EMC^2-$OXSX$ is very close to the area of EMC^2-*Base*. Therefore, we can conclude that EMC^2-$OXSX$ achieves a good compromise between network traffic and area requirements. Note that reductions in network traffic will lead to reductions in power consumption.

6.5 HTA Coverage Ratio

As discussed in Section 4, the coverage ratio of the HTA structure can become an scalability issue for large systems when the worst-case scenario appears, i.e., when cached blocks are not uniformly distributed among memory controllers. In order to emphasize the negative effect of the worst-case coverage ratio, we use different simulation parameters for this study. Particularly, we consider system comprised of two cores per die instead of just one (to stress the caches more), four dies per nest, and two nests (due to simulation time constraints). Additionally, since the working set of the SPLASH-2 benchmarks is very small compared to the cache sizes, we have halved the size of the caches (both data and HTA). Since both caches are halved, both the typical and the worst-case coverage ratio remain constant.

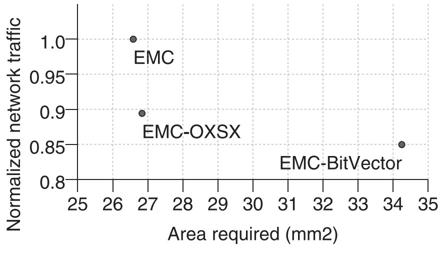


Fig. 14. Traffic-area trade-off for a 32-die system.

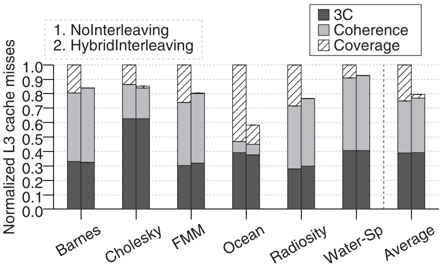


Fig. 15. Classification of cache misses for the no-interleaving policy and the hybrid interleaving policy.

In order to achieve a better understanding of the impact that the interleaving policy has on the coverage ratio, we have split the misses suffered by data caches into a new *5C* classification: the traditional 3C classification (*Cold* misses, *Capacity* misses and *Conflict* misses), *Coherence* misses, and *Coverage* misses. While coherence misses are caused by previous invalidations or loss of write permission due to requests issued by other nodes, coverage misses are caused by prior invalidations performed as a consequence of replacements in the HTA.

As we can see in Fig. 15, when no interleaving is performed, the replacements in the HTA can cause up to 50 percent of cache misses, as happens in *Ocean*, and 25 percent on average. However, by using the hybrid interleaving policy described in Section 4, a more uniform distribution of memory blocks is achieved, thereby reducing the percentage of coverage misses significantly (it only represents a three percent of cache misses). We also can observe that the total number of cache misses is reduced by 20 percent, on average, which will impact positively on execution time.

Fig. 16 plots the reduction in execution time when both the base interleaving policy and the hybrid interleaving policy are employed. We can observe that the execution time can be reduced up to 30 percent (as happens for *Ocean*) and by 7.8 percent on average with the hybrid interleaving policy.

6.6 EHTA Size Analysis

In this section, we analyze the size and associativity of EHTA structure and its impact in execution time and area requirements. The smaller the size of the EHTA is, the more EHTA replacements take place. According to our implementation, these replacements imply the invalidation of all

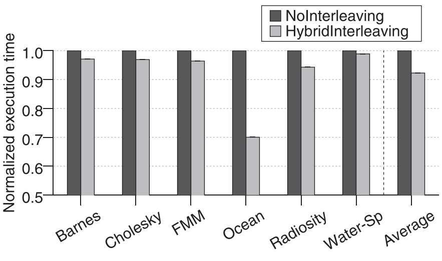


Fig. 16. Normalized execution time for the no-interleaving policy and the hybrid interleaving policy.

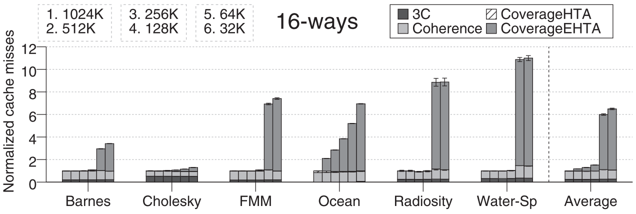


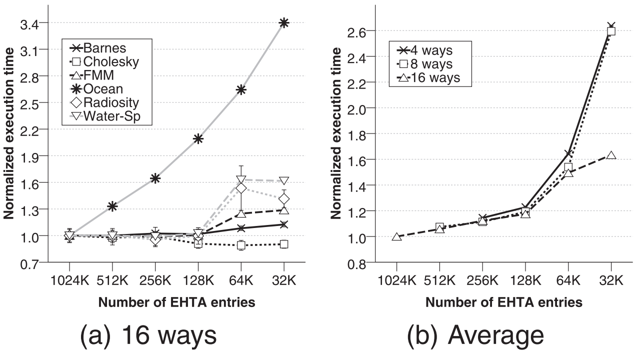
Fig. 17. Normalized number of cache misses when the EHTA size is reduced from 1,024 K entries to 32 K entries.

the cached copies in the system of the block whose EHTA entry is being evicted. These invalidations can increase the cache miss rate, and consequently the applications' execution time.

For this study, we employ the hybrid interleaving scheme, which spreads the directory entries among the HTAs within a nest, thus preventing them from being the bottleneck in the simulations. Moreover, we employ the simulation parameters described in Section 5. Again, we focus on the EMC^2-$OXSX$ chip and on a 32-die configuration.

First, in Fig. 17 we plot the increase in the amount of cache misses when the number of entries of the EHTA is reduced from 1,024 K (representing the case where there are no replacements) to 32 K, and the associativity does not vary (16 ways). For this study we split again the cache misses according to a new *6C* classification. This classification is the same as the one described in the previous section, but it divides the coverage misses into *CoverageHTA* misses or *CoverageEHTA* misses, which represent the misses that arise as a consequence of replacements in the HTA or in the EHTA, respectively. We can observe that the number of EHTA misses increases as the number of entries of the EHTA becomes smaller. However, for all the applications except for *Ocean*, the size of the EHTA can be reduced up to eight times without significantly increasing the data cache miss rate.

Fig. 18 shows impact on the execution time of reducing the EHTA. In Fig. 18a we do not modify the associativity of the EHTA and therefore shows the consequences in execution time of Fig. 17. We can see that *Ocean* is the most affected by the reduction of the EHTA size , whereas the other applications are not affected up to reductions about eight times. Particularly, *Cholesky* slightly improves its execution time when we move from a 256K-entry configuration to a 128K-entry configuration. This effect is due to premature invalidations, i.e., the probes generated as a consequence of EHTA replacements invalidate cache blocks that are not going to be used which lessens the load of the caches.



(a) 16 ways (b) Average

Fig. 18. Execution time when the EHTA size is reduced from 1,024K entries to 32K entries.

TABLE 5
Size of the Different EMC^2 Chips with Different EHTA Configurations for 32-Die Systems (Eight Nests)

EHTA	EHTA	Total area requirements (mm^2)		
entries	assoc	EMC^2-Base	EMC^2-OXSX	EMC^2-BitVector
1024K	16	26.58	26.83	34.25
512K	16	13.76	13.88	17.61
256K	16	7.35	8.12	9.67
128K	16	4.64	4.68	5.28
64K	16	2.79	2.80	3.46
32K	16	1.88	1.89	2.25
512K	8	13.59	13.72	17.40
256K	8	7.27	7.33	9.54
128K	8	4.22	4.63	5.22
64K	8	2.74	2.76	3.20
32K	8	1.86	1.87	2.22
16K	8	1.40	1.41	1.54
256K	4	7.19	7.26	9.08
128K	4	4.18	4.21	5.17
64K	4	2.53	2.73	3.18
32K	4	1.86	1.86	2.02
16K	4	1.39	1.40	1.54
8K	4	1.10	1.10	1.16

On the other hand, Fig. 18b shows the average execution time for the six applications considered in this work, varying both the number of entries of the EHTA and its associativity (from 16 to 4 ways). We can see that a 4-way or a 8-way configuration behave similarly to a 16-way configuration down to 64 K entries. For smaller sizes the impact on execution time of reducing the associativity is not admissible.

Finally, we show in Table 5 the area requirements of each EMC^2 chip variant containing each EHTA configuration and assuming 32 dies. To summarize the results we plot a trade-off between execution time and area requirements for the EMC^2-$OXSX$ chip and considering 32 dies in Fig. 19. We can see that an EHTA structure comprised by 128 K entries and four ways obtains a good trade-off between execution time and area requirements.

6.7 Impact of Providing More MST Tags

Finally, we study the impact of increasing the number of tags in the MST by employing the identifiers of the three unused dies in the nest. In particular, Fig. 20 presents the improvements in terms of execution time when the number of tags in the MST is multiplied by four (i.e., the EMC^2 chip uses a 128-tag MST instead of a 32-tag MST). Again, we

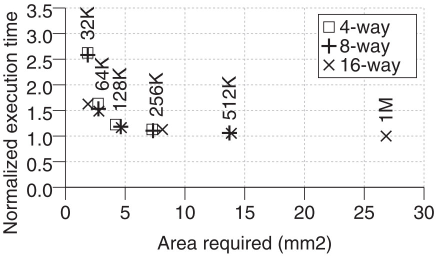
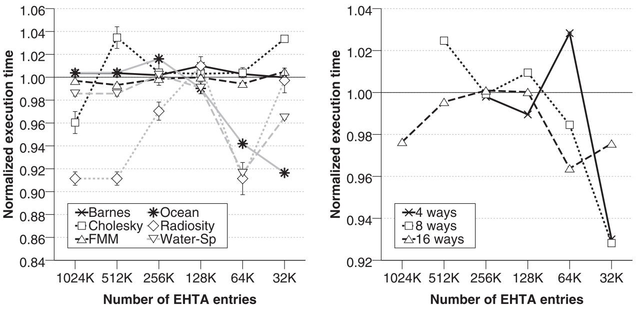


Fig. 19. Execution time-area trade-off varying the EHTA size.



(a) 16 ways (b) Average

Fig. 20. Improvements in execution time by increasing the number of tags available in the MST up to 128 tags. Results are normalized with respect to 32 tags.

assume a system with 32 dies, eight nests, and 8 EMC^2 chips. Since the occupation of tags depends on the number of EHTA replacements, several EHTA sizes have been considered in this study.

Fig. 20a shows the variations for the six benchmarks evaluated and a 16-way EHTA. For some applications the execution time is improved by up to nine percent depending on the configuration. Additionally, in Ocean the higher improvements are obtained for smaller EHTAs. This partially compensates the slow downs shown in the previous section for this application.

On the other hand, Fig. 20b shows the average of the six applications varying the associativity. As we can observe, the trend is to obtain better improvements for smaller EHTA configurations, because they will suffer more evictions. These improvements can reach up to seven percent for an EHTA with 32 K entries.

7 Conclusions

In this paper, we have extended, by means of an external logic (the EMC^2 chip), the coherence domain of the AMD Magny-Cours processors beyond the 8-die limit imposed by both the cHT specification and the size of the owner field of the HTA. The proposed chip not only maintains the HTA capability to filter the coherence traffic over the entire system, but also filters additional traffic, providing the scalability required to build large-scale servers. Evaluation results for a 32-node system show how the runtime of the applications scales with the number of nodes, reducing the application runtime by 47 percent on average when compared to the 8-die Magny-Cours system.

We have analyzed three EMC^2 chip variants which provide different trade-offs between filtered network traffic and required silicon area. Particularly in a 32-die system, $EMC^2\text{-}OXSX$ achieves a good compromise between network traffic (10.6 percent of traffic reduction compared to $EMC^2\text{-}Base$) and reducing area requirements (22.2 percent of area reduction compared to $EMC^2\text{-}BitVector$).

In addition, we have also addressed two potential scalability problems that could degrade the performance of large systems. First, the HTA coverage ratio problem can be palliated by using a hybrid interleaving policy, reducing execution time by 7.8 percent. Second, taking advantage of the unused die identifiers to allow the EMC^2 chip to manage more external transactions simultaneously can reduce the execution time by seven percent on average.

Acknowledgments

This work was supported by the Spanish MICINN, Consolider Programme and Plan E funds, as well as European Commission FEDER funds, under Grants CSD2006-00046 and TIN2009-14475-C04-01/03. It was also partly supported by (PROMETEO from Generalitat Valenciana (GVA) under Grant PROMETEO/2008/060).

References

[1] J.M. Owen, M.D. Hummel, D.R. Meyer, and J.B. Keller, "System and Method of Maintaining Coherency in a Distributed Communication System," U.S. Patent 7069361, June 2006.

[2] Intel, "An Introduction to the Intel QuickPath Interconnect," Whitepaper, http://www.intel.com/technology/quickpath/introduction.pdf, Jan. 2009.

[3] *InfiniBand Architecture specification release 1.2,* InfiniBand Trade Assoc., http://www.InfiniBandta.com, Oct. 2004.

[4] P. Conway, N. Kalyanasundharam, G. Donley, K. Lepak, and B. Hughes, "Cache Hierarchy and Memory Subsystem of the AMD Opteron Processor," *IEEE Micro,* vol. 30, no. 2, pp. 16-29, Apr. 2010.

[5] P. Conway, "Computer System with Integrated Directory and Processor Cache," U.S. Patent 6868485, Mar. 2005.

[6] SGI, "Technical Advances in the SGI Altix UV Architecture," Whitepaper, http://www.sgi.com/pdfs/4192.pdf, 2009.

[7] 3Leaf Systems "Next Generation Hybrid Systems for HPC," Whitepaper, http://www.3leafsystems.com/download/3leaf_wt_ paper_Next_Gen_Hybrid_Systems_for_HPC.pdf, 2009.

[8] J. Duato, F. Silla, S. Yalamanchili, B. Holden, P. Miranda, J. Underhill, M. Cavalli, and U. Brüning, "Extending Hyper Transport Protocol for Improved Scalability," *Proc. First Int'l Workshop Hyper Transport Research and Applications (WHTRA),* pp. 46-53, Feb. 2009.

[9] A. Ros, B. Cuesta, R. Fernández-Pascual, M.E. Gómez, M.E. Acacio, A. Robles, J.M. García, and J. Duato, "EMC^2: Extending Magny-Cours Coherence for Large-Scale Servers," *Proc. 17th Int'l Conf. High Performance Computing (HiPC),* pp. 1-11, Dec. 2010.

[10] R. Kota and R. Oehler, "Horus: Large-Scale Symmetric Multiprocessing for Opteron Systems," *IEEE Micro,* vol. 25, no. 2, pp. 30-40, Mar. 2005.

[11] J. Laudon and D. Lenoski, "The SGI Origin: A cc-NUMA Highly Scalable Server," *Proc. 24th Int'l Symp. Computer Architecture (ISCA),* pp. 241-251, June 1997.

[12] J. Brooks, C. Grassl, and S. Scott, "Performance of the CRAY T3E Multiprocessor," *Proc. ACM/IEEE Conf. Supercomputing (SC '97),* pp. 1-17, Nov. 1997.

[13] A. Agarwal, R. Simoni, J.L. Hennessy, and M.A. Horowitz, "An Evaluation of Directory Schemes for Cache Coherence," *Proc. 15th Int'l Symp. Computer Architecture (ISCA),* pp. 280-289, May 1988.

[14] P. Sweazey and A.J. Smith, "A Class of Compatible Cache Consistency Protocols and Their Support by the IEEE Futurebus," *Proc. 13th Int'l Symp. Computer Architecture (ISCA),* pp. 414-423, June 1986.

[15] P.S. Magnusson, M. Christensson, and J. Eskilson et al., "Simics: A Full System Simulation Platform," *IEEE Computer,* vol. 35, no. 2, pp. 50-58, Feb. 2002.

[16] M.M. Martin, D.J. Sorin, and B.M. Beckmann et al., "Multifacet's General Execution-Driven Multiprocessor Simulator (GEMS) Toolset," *Computer Architecture News,* vol. 33, no. 4, pp. 92-99, Sept. 2005.

[17] N. Agarwal, T. Krishna, L.-S. Peh, and N.K. Jha, "GARNET: A Detailed On-Chip Network Model Inside a Full-System Simulator," *Proc. IEEE Int'l Symp. Performance Analysis of Systems and Software (ISPASS),* pp. 33-42, Apr. 2009.

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

[19] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," *Proc. 22nd Int'l Symp. Computer Architecture (ISCA),* pp. 24-36, June 1995.

[20] A.R. Alameldeen and D.A. Wood, "Variability in Architectural Simulations of Multi-Threaded Workloads," *Proc. Nine Int'l Symp. High-Performance Computer Architecture (HPCA),* pp. 7-18, Feb. 2003.

Alberto Ros received the MS and PhD degrees in computer science from the Universidad de Murcia, Spain, in 2004 and 2009, respectively. In 2005, he joined the Computer Engineering Department at the same university as a PhD student with a fellowship from the Spanish government. Since 2009, he has been working as a researcher at the Parallel Architecture Group (GAP) of the Universitat Politècnica de València. Currently, he is working toward the designing and evaluating scalable cache coherence protocols for shared-memory multiprocessors. His research interests include cache coherence protocols, memory hierarchy designs, and scalable multiprocessor architectures. He is a member of the IEEE.

Blas Cuesta received the MS degree in computer science from the Universitat Politècnica de València, Spain, in 2002. In 2005, he joined the Parallel Architecture Group (GAP) in the Department of Computer Engineering at the same university as a PhD student with a fellowship from the Spanish government, receiving the PhD degree in computer science in 2009. Currently, he is working toward the designing and evaluating scalable coherence protocols for shared-memory multiprocessors. His research interests include cache coherence protocols, memory hierarchy designs, scalable cc-NUMA and chip multiprocessor architectures, and interconnection networks. He is a member of the IEEE Computer Society.

Ricardo Fernández-Pascual received the MS and PhD degrees in computer science from the Universidad de Murcia, Spain, in 2004 and 2009, respectively. In 2004, he joined the Computer Engineering Department as a PhD student with a fellowship from the regional government. Since 2006, he is an assistant professor in the Universidad de Murcia. His research interests include general computer architecture, fault tolerance, chip multiprocessors and performance simulation.

María E. Gómez received the MS and PhD degrees in computer science from the Universitat Politècnica de València, Spain, in 1996 and 2000, respectively. She joined the Department of Computer Engineering (DISCA) at Universitat Politècnica de València in 1996 where she is currently an associate professor of computer architecture and technology. Her research interests includes the field of interconnection networks, networks-on-chips and cache coherence protocols. He is a member of the IEEE Computer Society.

Manuel E. Acacio received the MS degree in computer scienceis, after that he joined the Computer Engineering Department (DiTEC) in 1998. He is an associate professor of computer architecture and technology at the University of Murcia, Spain. He started as a teaching assistant, at the time he began his work on the PhD thesis, which he successfully defended in March 2003. Before, in the summer of 2002, he worked as a summer intern at IBM TJ Watson, Yorktown Heights New york (NY). After that, he became an assistant professor in 2004, and subsequently, an associate professor in 2008. Currently, he leads the Computer Architecture and Parallel Systems (CAPS) research group at the University of Murcia, which is part of the ACCA group. He has published several papers in top conferences such as HPCA, IPDPS, ICS, DSN, PACT, or SC, and renown journals such as*IEEE TPDS and IEEE TC*. He has served as a committee member of important conferences, ICPP, and IPDPS among others. His research interests are focused on the architecture of multiprocessor systems. More specifically, he is actively working on prediction and speculation in multiprocessor memory systems, synchronization in CMPs, power-aware cache-coherence protocols for CMPs, fault tolerance, and hardware transactional memory systems. He is a member of the IEEE.

Antonio Robles received the MS degree in physics (electricity and electronics) from the Universitat de València, Spain, in 1984 and the PhD degree in computer engineering from the Universitat Politècnica de València in 1995. Currently, he is working as a full professor in the Department of Computer Engineering at the Universitat Politècnica de València, Spain. He has taught several courses on computer organization and architecture. His research interests include high-performance interconnection networks for multiprocessor systems and clusters and scalable cache coherence protocols for SMP and CMP. He has published more than 70 refereed conference and journal papers. He has served on program committees for several major conferences. He is a member of the IEEE Computer Society.

José M. García received the MS degree in electrical engineering and the PhD degree in computer engineering both from the Technical University of Valencia in 1987 and 1991, respectively. He is professor of computer architecture at the Department of Computer Engineering, and also head of the Parallel Computer Architecture Research Group. Currently, he is serving as Dean of the School of Computer Science at the University of Murcia, Spain. He has developed several courses on computer structure, peripheral devices, computer architecture, parallel computer architecture, and multicomputer design. He specializes in computer architecture, parallel processing and interconnection networks. His current research interests includes high-performance power-efficiency coherence protocols for Chip Multiprocessors (CMPs) and shared-memory multiprocessor systems, high-speed interconnection networks, and the use of GPUs for general-purpose applications such as bioinformatics and biomedical applications. He has published more than 150 refereed papers in different journals and conferences in these fields. He is member of HiPEAC, the European Network of Excellence on High Performance and Embedded Architecture and Compilation. He is also member of several international associations such as the IEEE and the ACM.

José Duato received the MS and PhD degrees in electrical engineering from the Universitat Politècnica de València, Spain, in 1981 and 1985, respectively. Currently, he is working as a professor in the Department of Computer Engineering at the Universitat Politècnica de València. He was an adjunct professor in the Department of Computer and Information Science at the Ohio State University, Columbus. His research interests include interconnection networks and multiprocessor architectures. He has published more than 380 refereed papers. He proposed a powerful theory of deadlock-free adaptive routing for wormhole networks. Versions of this theory have been used in the design of the routing algorithms for the MIT Reliable Router, the Cray T3E supercomputer, the internal router of the Alpha 21364 microprocessor, and the IBM BlueGene/L supercomputer. He is the first author of the *Interconnection Networks: An Engineering Approach* (Morgan Kaufmann, 2002). He was a member of the editorial boards of *the IEEE Transactions on Parallel and Distributed Systems*, *the IEEE Transactions on Computers*, and *the IEEE Computer Architecture Letters*. He was a cochair, member of the steering committee, vice chair, or member of the program committee in more than 55 conferences, including the most prestigious conferences in his area of interest: HPCA, ISCA, IPPS/SPDP, IPDPS, ICPP, ICDCS, EuroPar, and HiPC. He has been awarded with the National Research Prize *Julio Rey Pastor 2009*, in the area of Mathematics and Information and Communications Technology and the *Rei Jaume I Award on New Technologies 2006*.