Cache Miss Characterization in Hierarchical Large-Scale Cache-Coherent Systems

Alberto Ros*, Blas Cuesta†§, María E. Gómez‡, Antonio Robles‡, José Duato‡

*Departamento de Ingeniería y Tecnología de Computadores
Universidad de Murcia, 30100 Murcia (Spain)
E-mail: aros@ditec.um.es

†Intel Labs Barcelona
E-mail: blasx.cuesta@intel.com,blacuesa@gap.upv.es

‡Department of Computer Engineering
Universitat Politècnica de València, 46021 Valencia (Spain)
E-mail: {megomez,arobles,jduato}@gap.upv.es

***Abstract*—There is a growing trend towards developing large-scale cache-coherent systems by using commodity symmetric multiprocessors, which requires to extend their coherence protocol. In such systems, cache coherence transactions issued due to cache misses traverse interconnection networks with very different topologies and latencies. In this work, we perform a cache miss characterization aimed at analyzing the benefits that can be expected for a specialized coherence controller able to locally resolve cache misses, thus saving traffic across long-latency links. Results show that there is a high potential in reducing miss latency in these systems, and that this potential reduction grows as the number of nodes in the system increases. Particularly, in a system with just two boards 40% of the cache misses do not need the expensive inter-board communication. This percentage can increase up to 67.5% for an 8-board system.**

I. Introduction

Until recently, many service providers were able to use clusters of PCs for high performance computing (HPC). This kind of clusters usually relies on message-passing communications for remote memory accesses, which not only increases the communication latencies, but also difficulties the developing of efficient applications when compared to the shared-memory programing model. These drawbacks highlight the need for large-scale cache-coherent systems.

There is a current trend towards developing such large-scale cache-coherent systems based on using existing commodity symmetric multiprocessors (SMP), which requires to extend their coherence protocol. AMD was the first to include such features in their Opteron processors. Particularly, the six- and twelve-core versions of AMD Opteron processors, codenamed Istanbul and Magny-Cours [1] respectively, can be interconnected to compound a larger system while still maintaining cache coherence thanks to the Coherent HyperTransport (cHT) technology [2]. Similarly, the Intel's QuickPath Interconnect (QPI) allows several Nehalem processors to compound a larger coherent system.

§This work was done before the author joined Intel, while being at the Universitat Politècnica de València.

In order to increase even more the number of processor cores that can be kept coherent in such systems, several proposals aimed at further extending the coherence domain have appeared recently. We can find examples of these systems either in the market (e.g., Horus [3] and SGI Altix UV [4]) or in the literature (e.g., EMC^2 [5], [6]). These hierarchical systems have very different communication latencies among processing cores depending on the distance, the interconnection technology, and its level in the coherence hierarchy, as we can see in Figure 1. The basic building block is the die, that can comprise several processor cores (currently from 4 to 12). Communication among these cores is very fast (just a few nanoseconds) and can be carried out by a shared bus. Several dies can be placed in the same board in order to compound a larger system. Communication among dies is commonly performed through a scalable point-to-point interconnect (e.g., cHT or QPI), and usually requires tens of nanoseconds [1]. Finally, several boards can be connected by an InfiniBand [7] or Ethernet switch fabric. The component responsible for managing communication between internal (intra-board) and external (inter-board) messages is the *bridge* chip (also named as HORUS chip in [3], as UV_HUB in [4], and as EMC^2 in [8]). Communication latency across the inter-board network can be higher than one microsecond [9]. Since in these systems the inter-board communication latency is extremely high when compared to the other network latencies, the avoidance of this communication becomes a fundamental goal for delivering high performance.

In this paper, we present a characterization of the cache misses that require coherence transactions among dies or boards. This characterization represents the first and fundamental step of a work in progress whose final goal is the design of a cache coherence protocol able to make the most of the hierarchical systems. In particular, we are interested

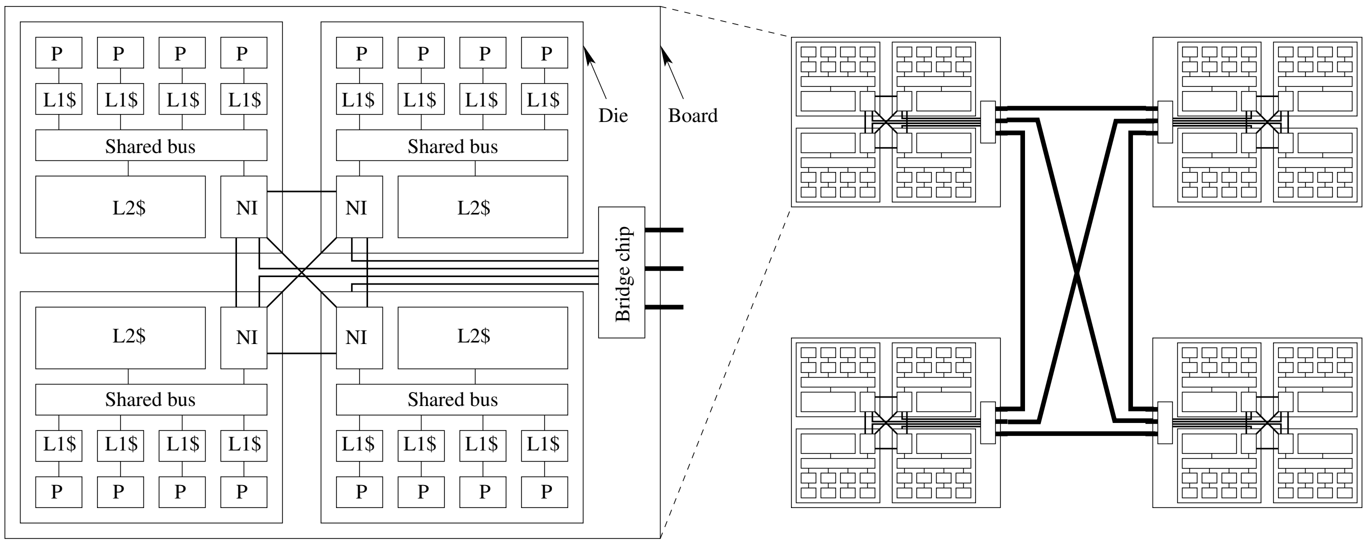

Fig. 1: Generic overview of a hierarchical system. *P* refers to the processors or cores and *NI* stands for network interface.

in a classification of cache misses according to their latency requirements. We mainly focus on the communication messages used to resolve each miss (e.g., if they require the access to another board or just to some die within the same board). Additionally, we are also interested in the misses that, requiring accesses to remote boards, could be resolved locally thereby avoiding the inter-board communication. In this paper, we briefly discuss the modifications and extra structures required in the cache coherence protocol in order to resolve cache misses without requiring remote transactions.

By means of full-system simulation we show that a high percentage of cache misses require inter-board communication. Particularly, for systems composed of 2, 4, and 8 boards, we have found that 50.0%, 75.2%, and 83.6% of cache misses (on average) need to traverse long-latency links, respectively. Additionally, this implies that as the system size grows, the latency of cache misses will also grow significantly. Our results also show that up to 85.8%, 89.8%, and 90.3% of these misses could be resolved locally for systems with 2, 4, and 8 boards, respectively, by adding some extra structures at the same time that the cache coherence protocol engine is modified to this purpose.

The rest of this paper is organized as follows. Section II offers some background about the kind of systems analyzed in this paper. Section III describes our simulation environment. The cache miss characterization is presented in Section IV. In Section V, we briefly discuss some techniques that can be employed for avoiding inter-board communication, and finally, conclusions and future work are drawn in Section VI.

II. Background

There are some systems both in the market and in the recent literature that are examples of the baseline system considered in this paper: Horus, SGI Altix UV, and EMC^2. In the following sections, we give some background about them.

A. Horus

HORUS [3] is a system that scales from 8 sockets to 32 sockets the SMP capabilities of a family of AMD Opteron processors that keep coherence by means of a broadcast-based protocol [2]. Particularly, the system comprises several HORUS chips, each one connected to 4 dies (one die per socket) through internal cHT links and to other HORUS chips through an external longer-latency interconnect. In order to reduce the significant amount of external traffic generated by the broadcast-based protocol, the HORUS chip includes a directory structure that maintains both the state and a bit-vector sharing code (one bit per HORUS chip) for each local memory block that is cached remotely. In this way, the amount of external traffic can be significantly reduced.

Additionally, each HORUS chip includes a 64MB remote data cache (RDC) to avoid inter-board communication (i.e., external communication among HORUS chips). This can lead to a reduction in both cache miss latency and coherence traffic. The RDC cache can be filled in two ways. First, it keeps a copy of data sent by the memory controller to one of the 4 dies connected to an HORUS chip. Second, when a local processor evicts a dirty memory block from cache, the RDC stores the data block. Unfortunately, this RDC can only save inter-board communication on read misses and only when the data block has been previously requested by one of the processors belonging to the same board.

B. SGI Altix UV

The SGI Altix UV [4] is another example of a scalable global shared-memory system, where communication among processing cores usually goes across different networks with diverse link latencies. With up to eight cores per socket and up to 256 sockets, the SGI Altix UV can scale up to 2048 cores. A board can contain one or two sockets. The UV_HUB is the component that links the cache-coherent QPI [10] found on Intel processors with the larger cache-coherent NUMAlink environment that extends across the full system. Differently from HORUS, the UV_HUB includes neither a directory nor any cache structure to filter expensive NUMAlink traffic.

C. EMC^2

EMC^2 (Extended Magny-Cours Coherence) [5], [6] overcomes the 8-die limitation of Magny-Cours systems [1]. The basic building block of this system is the Magny-Cours die, which comprises six cores. Up to two Magny-Cours dies can be included in the same socket and up to four sockets are available per board (8 dies per board). Since the EMC^2 chip replaces one of the existing dies, the maximum number of processor dies per board is seven. These dies and the EMC^2 chip, which manages the translation between internal (cHT) and external (High Node Count or HNC [11]) messages, are connected by means of a cHT interconnect. The different EMC^2 chips are connected by an InfiniBand switch fabric, so HNC packets are encapsulated into InfiniBand packets. Again, the external network introduces longer latency than the internal one.

Differently from the HORUS chip, the EMC^2 chip has to deal with an internal directory-based protocol, which already includes directory caches (called HTA probe filter [1]) that store both the state and the owner of the cached memory blocks. When the block's owner, which is codified by using 3 bits, is any remote die (i.e., a die connected to another EMC^2 chip), it will point to its local EMC^2 chip, which contains another directory cache (called extended HTA, or EHTA) in order to extend the local HTA information to the global system. While dies (and particularly their HTAs) only have an intra-board view, EMC^2 chips (and particularly their EHTAs) have a view of all copies of local memory blocks out of its board. Although the EMC^2 system is able to filter some inter-board traffic due to the EHTA capabilities, there is still a large amount of external traffic that could be avoided, thus reducing cache miss latencies and final application's execution time, as we show in Section IV.

III. Simulation environment

We have performed the characterization presented in this paper by means of full-system simulation using Virtutech Simics [12] along with the Wisconsin GEMS toolset [13], which enables detailed simulation of multiprocessor systems. The interconnection network has been modelled using GARNET [14], a detailed network simulator included in the GEMS toolset.

For carrying out the miss characterization, we have considered the EMC^2 system because this is built based on current Magny-Cours dies. We assume that each board includes 4 dies. We have analyzed comprised by 2, 4, and 8 boards. The parameters for the Magny-Cours dies used in the simulations correspond to the ones described by their designers [1] and are

TABLE I: 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	2ns (L1+L2+L3)
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-board topology	Fully-connected
Inter-board topology	Hypercube
Data message size	68 or 72 bytes
Control message size	4 or 8 bytes
HyperTransport bandwidth	12.8GB/s
Inter-die link latency	2ns
Inter-socket link latency	40ns
InfiniBand bandwidth	12GB/s
Inter-board communication (one way)	1μs
Flit size	4 bytes
Link bandwidth	1 flit/cycle

shown in Table I. We do not model the intra-die coherence protocol or the intra-die cache hierarchy because in this paper we are only focused on the behaviour of the coherence protocol among dies and among boards, but not within the die. Since the detailed simulation of in-die traffic would result in a significant increase of the simulation time, we assume a fixed access latency (representing the average access time) for the whole intra-die hierarchy (L1, L2, and L3 caches). Note that these three levels are private within a die, so this does not affect the behaviour of either the inter-die or the inter-board coherence protocols, which are the target protocols for this paper. The parameters used for connecting different boards are also based on real systems [9]. These parameters are also shown in Table I.

We have performed the characterization for a wide range of scientific applications: *Barnes* (16K particles), *Cholesky* (tk16), *FFT* (64K complex doubles), *FMM* (16K particles), *Ocean* (514×514 ocean), *Radiosity* (room, -ae 5000.0 -en 0.050 -bf 0.10), *Radix* (512K keys, 1024 radix), *Raytrace* (teapot), *Volrend* (head), and *Water-Sp* (512 molecules) are from the SPLASH-2 benchmark suite [15]. *Blackscholes* (simmedium) and *Canneal* (simsmall) belong to PARSEC [16]. All the experimental results reported in this paper correspond to the parallel phase of these benchmarks. We account for the variability in multithreaded workloads [17] 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.

IV. Cache Miss Characterization

To provide some hints about the potential benefits in miss latency reduction of avoiding the inter-board communication, first of all, it is interesting to define a taxonomy of cache misses and then to obtain, for a set of parallel applications, the fraction of misses that fall into each of the miss types in the taxonomy. In this way, we can show the fraction of cache misses that imply inter-board communication, as well as the fraction of them that could avoid it, i.e., that could be resolved locally if the cache coherence protocol is modified to take this into account.

Table II shows the taxonomy defined in this paper. We first consider the type of the cache miss (read or write), since they require different coherence actions. Then, it is necessary to know if the requested block maps to a memory controller within the local die (i.e., the home memory controller is in the same board as the requester) or within a remote die. If the home memory controller is remote, the cache miss transaction will require the issue of messages through long-latency links. However if the home is local, inter-board communication will only happen depending on the following:

- For read misses, if the die having the ownership of the requested block belongs to the same board as the home memory controller (and therefore as the requester), no inter-board communication will be required. Otherwise, a coherence message must be sent to the remote board where the owner die resides.
- For write misses, if at least one copy of the block in a external die (i.e., a die within a remote board), then high-latency communication will be required. Otherwise, the write miss can be resolved locally. Note that the EMC^2 system is already able to filter board-to-board communication when there are not external sharers.

Finally, it is also interesting to analyze if, despite requiring external transactions, (1) the cache miss could be locally resolved by modifying the cache coherence engine in the bridge chip and (2) in which situations this could be done. As we can see in Table II, among the six cache miss types shown, two of them do not entail inter-board communication. Among the remaining four miss types that require inter-board communication, we have detected that three of them could avoid those high-latency links. Particularly, read misses could be resolved locally if the requested data block is present in the same board where the miss takes place. On the other hand, write misses can be resolved locally if the requested block is only stored in caches within the same board as the requester.

Once the taxonomy is clearly defined, it is interesting to study how frequent each type of miss is, and since misses entailing inter-board communication have longer latency, it is also interesting to know their impact on the overall miss latency.

Figure 2 shows the fraction of misses that fall into each category of the taxonomy for a system with two boards (8 dies), four boards (16 dies), and eight boards (32 dies), and for the different applications described in Section III. The fraction of misses represented as white or striped corresponds to the misses that, not requiring inter-board communication, could be resolved locally. We can see that this fraction of misses increases with the number of boards. For a 2-board system comprised of 8 dies these misses represent 40.0% (on average) of the total, but for a 4-board system this percentage goes up to 61.5% on average, and up to 67.5% on average for an 8-board system.

Additionally, the misses with inter-board communication are the ones that take longer to resolve. Therefore, the avoidance of the inter-board communication for them can be very helpful to reduce the applications' execution time. Figure 3 shows the fraction of time required to resolve the misses falling into each category. Again, we can see that the fraction of time spent for misses that could avoid the inter-board communication increases with the number of boards in the system, being of about 69.8% for a 2-board configuration and about 89.2% for an 8-board configuration (on average). This increase is due to both the larger number of misses within these categories and the increase in the number of network hops required to reach a remote board.

Although we have shown that the latency of many cache misses can be shortened, current implementations of hierarchical systems do not take full advantage of this. For example, neither *SGI Altix UV* nor *EMC*^{2} employ any technique for reducing the inter-board communication. Differently, *HORUS* includes the remote data cache, but this cache is only able to avoid the inter-board communication for some *Read_Remote* misses, which only account for 18% and 36% of total misses for a 2-board and an 8-board configuration, respectively. We have shown that a larger fraction of read misses (from 27.4% when considering 2 boards to 44.9% for 8 boards) and a significant fraction of write misses (from 12.5% –and 37% w.r.t the number of write misses– for 2 boards to 22.1% –and 61% w.r.t the number of write misses– for 8 boards) can severely reduce their latency.

V. Avoiding High-Latency Communication

In the previous section we have shown that cache miss latency can be considerably reduced if the cache coherence protocol was modified by having the presented characterization into account. Since these results are very promising, we think that it is worthy to pay more attention to the development of specific cache coherence protocols for hierarchical systems. Therefore, this section points out some of the techniques that can be employed to improve performance. A complete implementation and evaluation of such a protocol is out of the scope of the paper and is kept as future work.

We have noticed that the avoidance of remote read misses can be carried out by adding extra data structures (caches) to

TABLE II: Taxonomy of cache misses.

Miss type	Home	Inter-board communication	Can it be avoided?
Read	Local	No (local owner)	Not necessary
		Yes (remote owner)	Yes, if there are local copies
	Remote	Yes	Yes, if there are local copies
Write	Local	No (no external copies)	Not necessary
		Yes (external copies)	No
	Remote	Yes	Yes, if there are only internal copies

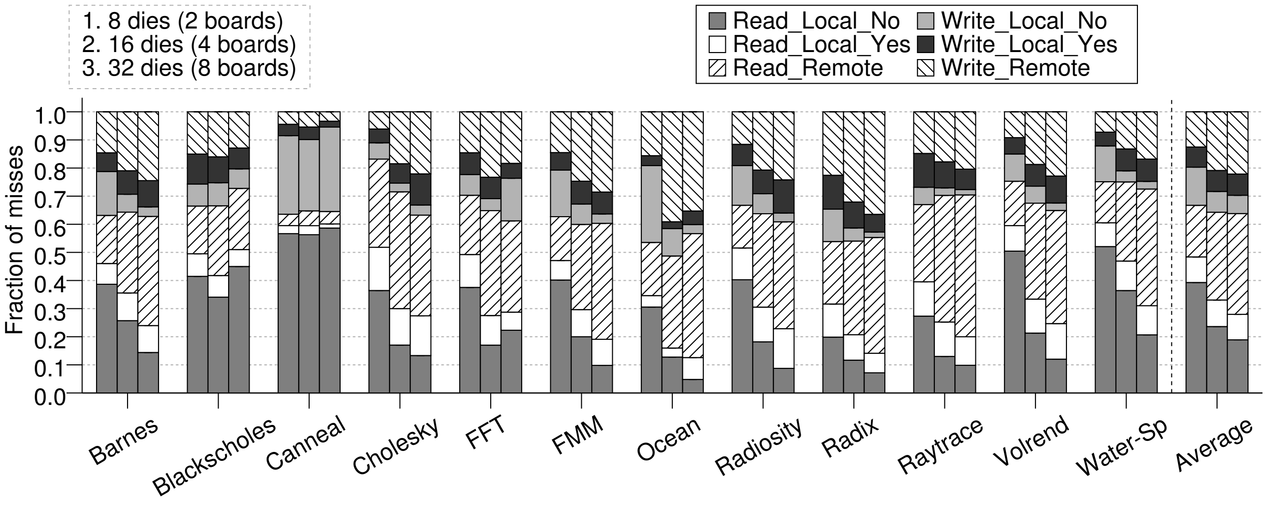

Fig. 2: Fraction of misses that falls into each category.

the component responsible for the inter-board communication (i.e., the bridge chip).

Read misses can be locally solved if the requested data are found in the local board. To get this, we can force the data to be found in the local board in two ways. When a die in the same board requests some data, this can be stored either in a *local data cache* located in the bridge chip (similar to the RDC cache in Horus) or in a *local directory cache* pointing to the die holding the data block. The decision will depend on the state of the block in the local cache. If the block is clean, evictions will take place in a silent way, so it is important to have the data in the bridge chip. If the block is dirty, write-backs will inform about evictions, and the identity of the die holding the data is enough to locally provide the requested data.

On the other hand, the avoidance of remote write misses can be achieved by adding in the local directory cache extra information about if there exist external copies for the requested block. In case only internal copies exist, the miss can be resolved locally by broadcasting invalidation messages to every die within the local board.

Note that, although these two structures would require a significant amount of extra storage, the current size of the bridge chip is extremely small (e.g., less than 30mm^2 in [6]) considering that it will be allocated instead of a processor die of several hundreds of mm^2, so extra data caches can be added to its design without compromising its area.

VI. Conclusions and future work

This paper constitutes a preliminary work that studies the potential of avoiding long latency communication for large-scale cache coherent systems comprised of interconnection networks with different latencies. Particularly, this work focuses on the avoidance of the inter-board communication, which is the most expensive one in hierarchical systems. We have noticed that a large percentage of cache misses can be resolved locally, i.e., without leaving the board where the miss takes place, if some additional extra structures were used by the cache coherence protocol. This percentage increases from 40% in a 2-board system to 67.5% in an 8-board system, and it is expected to increase even more for larger configurations. Additionally, we have shown that the reduction that can be

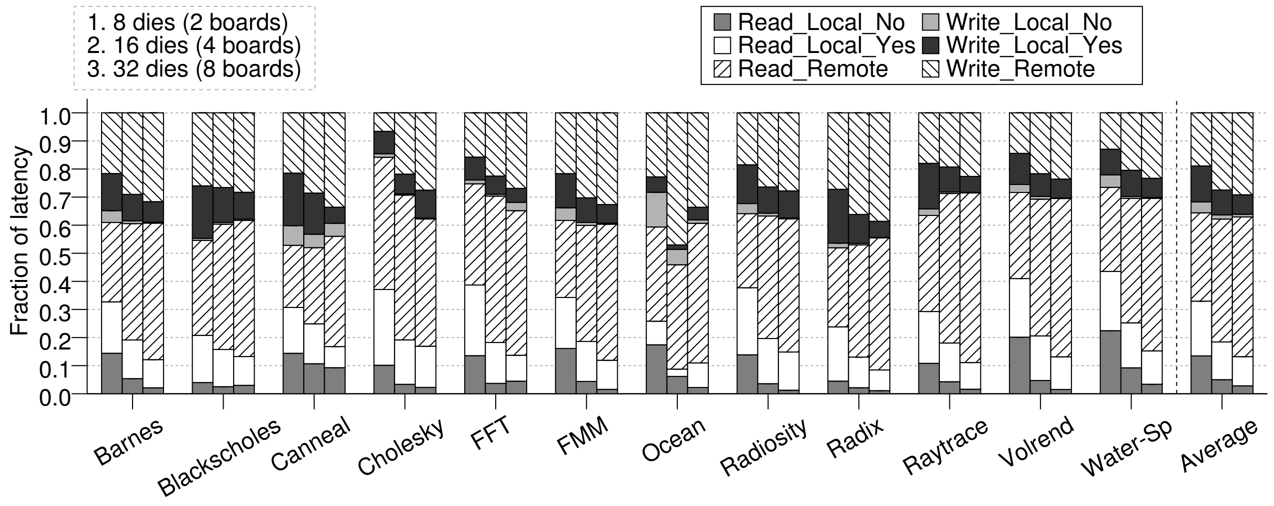

Fig. 3: Fraction of latency for misses into each category.

obtained in the cache miss latency is expected to be significant and also higher for larger systems. These promising results suggest that more attention must be paid in the design of cache coherence protocols for hierarchical systems.

Therefore, the next step in this project consists on modifying the cache coherence protocol of such systems in order to be able to avoid long-latency communication for the cases that we have detected in this paper. To this end, we plan to add some hardware structures to the bridge chip, as well as to change the behaviour of the coherence protocol to support locally resolved cache misses.

Acknowledgment

This research was supported by the Spanish MEC and MICINN, as well as European Commission FEDER funds, under Grants CSD2006-00046 and TIN2009-14475-C04, and PROMETEO from Generalitat Valenciana (GVA) under Grant PROMETEO/2008/060.

References

[1] 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.

[2] 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, Jun. 2006.

[3] R. Kota and R. Oehler, "Horus: Large-scale symmetric multiprocessing for opteron systems," *IEEE Micro*, vol. 25, no. 2, pp. 30–40, Mar. 2005.

[4] SGI, "Technical advances in the SGI Altix UV architecture," whitepaper, 2009. [Online]. Available: http://www.sgi.com/pdfs/4192.pdf

[5] 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," in *17th Int'l Conference on High Performance Computing (HiPC)*, Dec. 2010, pp. 1–10.

[6] A. Ros, B. Cuesta, R. Fernández-Pascual, M. E. Gómez, M. E. Acacio, A. Robles, J. M. García, and J. Duato, "Extending magny-cours cache coherence," *IEEE Transactions on Computers*, Apr. 2011.

[7] *InfiniBand Architecture specification release 1.2*, InfiniBand Trade Association^{TM}, Oct. 2004. [Online]. Available: http://www.InfiniBandta.com

[8] A. Ros, M. E. Acacio, and J. M. García, "A direct coherence protocol for many-core chip multiprocessors," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 21, no. 12, pp. 1779–1792, Dec. 2010.

[9] H. Montaner, F. Silla, H. Fröning, and J. Duato, "A new degree of freedom for memory allocation in clusters," *Cluster Computing*, pp. 1–23, 2011.

[10] Intel, "An introduction to the Intel QuickPath interconnect," whitepaper, Jan. 2009. [Online]. Available: http://www.intel.com/technology/quickpath/introduction.pdf

[11] J. Duato, F. Silla, S. Yalamanchili, B. Holden, P. Miranda, J. Underhill, M. Cavalli, and U. Brüning, "Extending HyperTransport protocol for improved scalability," in *1st Int'l Workshop on HyperTransport Research and Applications (WHTRA)*, Feb. 2009, pp. 46–53.

[12] 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.

[13] 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, Sep. 2005.

[14] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," in *IEEE Int'l Symp. on Performance Analysis of Systems and Software (ISPASS)*, Apr. 2009, pp. 33–42.

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

[16] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *17th Int'l Conference on Parallel Architectures and Compilation Techniques (PACT)*, Oct. 2008, pp. 72–81.

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