# A Survey on Cache Coherence for Tiled Many-core Processor

LIMIN HAN, JIANFENG AN, DEYUAN GAO, XIAOYA FAN, XIANGLONG REN, TAO YAO
Department of Computer Science and Engineering
Northwestern Polytechnical University
Xi'an, china
hanlm@mail.nwpu.edu.cn

***Abstract*-Due to technological parameters and constraints entailed in many-core processor with shared memory systems, it demands new solutions to the cache coherence problem. Directory-based coherence protocols have recently seemed as a possible scalable alternative for CMP designs. Unfortunately, with the number of on-chip cores increasing, many directory design strategies do not scale beyond a dozen cores due to huge energy and area costs for scaling the directories. We explored different NUCA design schemes for tiled many-core architecture, compared several conventional directory protocols, such as full-map directory protocol, sparse directory protocol, duplicate-tag-based directory protocol etc. and analyzed several novel cache protocols designed for many-core processor. At the end, we propose several design directions for scalable and adaptive cache coherence protocols for many-core processor.**

***Keywords-Many-core processor; Tiled CMP; Cache Coherence; Directory protocol***

## I. INTRODUCTION

At present, the number of cores in Chip multi-processor (CMP) is growing significantly due to continued technology advances. Tiled CMP architectures are constructed based on scalable point-to-point interconnection networks, such as a mesh or torus and on-chip networks [1]. Tiled CMP has recently emerged as a scalable alternative to current small-scale CMP designs and will be probably the architecture of choice for future many-core Processor [2, 3]. These scalable multiprocessors implement shared main memory and with private cache memory in each processing node to enable fast access to memory [4]. When multiple processors maintain locally cached copies of a unique shared memory location, any local modification of this location can result in a globally inconsistent view of memory, leading to cache coherence problems [5].

Unfortunately, many current cache protocol design strategies for a few cores CMP cannot scalable for several reasons. At first, in a cloud computing environment, it will make future workloads more heterogeneous and dynamic and these future heterogeneous workloads will need scalable and malleable cache management schemes given the hundreds of cores likely in a CMP. At second, conventional directory structures can incur significant area overheads in large-scale multi-core processor, the extra area required by directory is not scalable with the number of cores [6, 7].

Many approaches aimed at improving the scalability of directories for many-core CMP have been proposed. However, they usually reduce the directory memory overhead by compressing coherence information, which in turn results in extra unnecessary coherence messages and, therefore, wasted energy and introduce some performance degradation and do not bring perfect scalability [8, 9].

In the following sections, we present tiled many-core architecture and non-uniform cache architecture (NUCA) as well in details to illustrate the uses of distributed directories protocols firstly, then analyse several typical directory-based protocols and their implementations, and presents several cache coherence protocols aimed at many-core processor according to three main design goals: performance, on-chip network traffic and area requirements.

## II. TILED MANY-CORE ARCHITECTURE

As in Fig.1, a tile-based CMP is comprised of multiple identical tiles each with a compute core, L1 cache, shared L2 cache slice, distributed on-chip directory and network router [9]. In such architecture, directory can be distributed among all the tiles by mapping a block address to a tile called the home tile. All requests are forwarded to the home tile of a block which contains directory entry for it and a copy of the retrieved data is returned to the requesting tile [5].

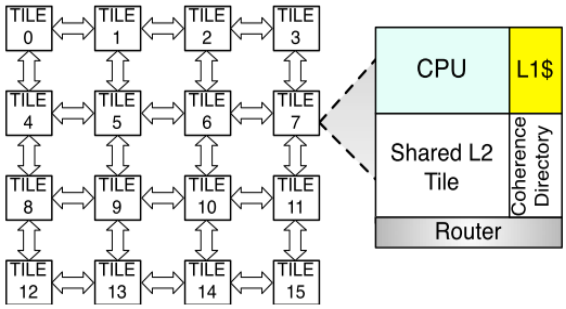


Figure.1 16-core tiled CMP with coherence directory distributed for each tile [9]

### A. NUCA structure in Tiled CMP

The tiled organization is not dramatically changed in successive processor generations, this trend implies that more tiles will lead to more aggregate L2 cache capacity, so distributed NUCA is a suitable candidate. As in Fig.1, there is a

This work is supported by Natural Science Foundation (60773223, 61003037 and 60736012), National 863 Program (2009AA01Z110) and NPU Foundation for Fundamental Research "Research of NoC Low Power Technology under Performance Constraint".

978-1-4673-2193-8/12/$31.00 ©2012 IEEE

typical NUCA structure with one slice of the shared L2 cache by mapping of a physical memory address to banks in the L2 cache for each tile. In terms for data placement policy, there are two kinds of organization structure for NUCA.

One is Static NUCA: Data are statically mapped into banks. One frequently-used data mapping scheme is that the low-order bits determining to which bank it will be mapped. There are two main advantages to the use of static NUCA. First, the banks closer to the controller have better average access time than the banks farther to the controller. Second, accesses to different banks may proceed in parallel, reducing contention [10]. It is easy for static NUCA structure to determine "home core" for directory cache protocol. However, given the presence of dynamically changing fine-grain access patterns, such naive data placement can result in high remote-cache access rate and introduce many indirect coherence messages, for it is unlikely that any simple mapping schemes can ensure that most cache lines are mapped close to or at the accessing nodes.

Another is Dynamic NUCA structure: by dynamic mapping of data into any banks, frequently used data could be stored in closer, faster banks than data that are used less frequently. For example, J. Zebchuk developed a compile-time framework for data locality optimization via data layout transformation for Dynamic NUCA structure [6]. In this way, D-NUCA promises big benefits by leveraging locality of data in the NUCA cache [10]. A dynamic NUCA model can be characterized by four policies that are involved in its behavior: the bank placement policy, the bank access policy, the bank migration policy and the bank replacement policy [11]. It needs OS-assisted data placement algorithm [12]. Proximity-Aware directory is proposed for CMP based on Dynamic NUCA substrate [13]. However, Prediction mechanism is needed to determine "home core" to maintain cache coherence, this will hurt performance by introducing long access latency.

### *B. Disadvatage of Directory Coherence protocol*

A directory-based protocol could take advantage of spatial and temporal locality by making a copy of the block in the local cache. In a conventional directory-based protocol, cache lines are distributed among the nodes in a straightforward interleaved fashion, such an interleaving allows easy determination of the home node, but creates overheads in coherence activities. Other than create overheads in coherence, directory-based coherence is not appropriate for many-core processor for the following reasons [14]:

*1) The automatic data replication of shared data results in one address being stored in many core-local caches, reducing the cache capacity left for other data, thereby have an worsening impaction on cache miss rates;*

*2) The directory causes an indirection which leads to an increase to the cache miss access latency. In a typical directory-based protocol, consulting the directory of a line on the home node is on the critical path for a number of situations such as transferring a line from the last writer to the next reader;*

*3) A write to shared data or an eviction of on-chip directory cache requires invalidation of all shared copies of the data, resulting in higher cache miss rates and protocol latencies;*

*4) The memory overhead introduced by the directory structure may not scale gracefully with the number of cores.*

## III. SEVERAL DIRECTORY DESIGN SCHEMES

A directory is a list storing the location and sharing status of all shared copies of each shared blocks. In particular, the directory information consists in a bit-vector sharing code that is employed for keeping track of the sharers. The directory is consulted first to identify the CPUs that share the data. This sharing code allows the protocol to send invalidation messages just to the caches currently sharing the block without broadcasting, thus removing unnecessary coherence messages. So implementing directory can be one of the key issues for the system performance. We discuss the implementation of several directory structures for tiled CMP as follows:

*1) Full-map directories*

In full-map directories, it stores enough states associated with each line in main memory by keeping a bit-map with one bit corresponding to each cache in the system, so that every cache in the system can simultaneously store a copy of any block of data [5]. First, an update of main memory can cause updates to be forwarded to every cache whose bit is set in that line's map, so this scheme led to heavy communication traffic. Second, the scheme is too simple to explore application data locality, so redundancy information introduce too much area costs.

*2) Duplicate-tag-based directories*

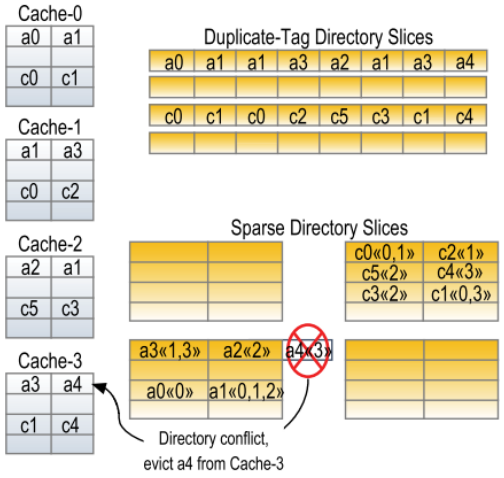


Figure.2 organizations for duplicate- tag directory and sparse directory[15]

As in Fig.2, The Duplicate-Tag organization mirrors the organization of the private-cache tags, ensuring that there is always sufficient space in the directory to track all cached blocks [15]. Associativity for the Duplicate-Tag-based or shadow tags directory must equal the product of the cache associativity and the number of caches, this result in designs with large associative directories. For example, shadow tags are used in Niagara2, It requires highly associative structures that grow with core count [16]. An associative search of the

shadow tags is used to generate the sharer vector on the fly, precluding scalability due to prohibitive power consumption [17].

*3) Sparse directory*

The goal of sparse directory is reducing the space of directory storage. Sparse directories overcome the power barrier of Duplicate-tag-based directories by reducing directory associativity make use the fact that actually directory access has locality because total number of cache blocks in all processors is far less than total number of memory blocks. As in Fig.2, When sparse directory space overflows, sparse directories experience set conflicts and forcing evictions of cached blocks that cannot be simultaneously tracked by the directory. Sparse directories with full-map vectors use a set-associative directory structure where each entry contains a block tag and a vector with one bit per core identifying the cores caching the block [18, 19]. To minimize the effect of these placement restrictions, each entry usually contains at least one entry for each cached block. The area of each entry grows as O ($P^2$) making it impractical for large Processor.

*4) In-cache directory*

The in-cache directory organization or the static cache bank directory extends an inclusive shared cache's tags with sharers information and distribute directory among the tiles. It implicitly saves directory tag storage, but grossly over-provision the sharer storage, because the number of tags in the lower-level cache greatly exceeds the number of tracked blocks in the private caches [15, 20, 21]. One example of real architecture implementing in-cache directory is the SGI Origin2000 multiprocessor system [4]. It uses each cache block state field keeping the coherence state of the block. Another architecture implementing in-cache directory is the Tilera Tile64 architecture. It is a 64-core processor consisting of an 8*8 grid of tiles [22]. In-cache directory integrates directory state with the cache tags thereby avoiding a separate directory either in DRAM or on-chip SRAM. However the tag overhead can become substantial in a many-core CMP with a bit for each sharer. When beyond 128 cores, in-cache directories lose their advantages and become dominated by bit-vector storage, limiting applicability to larger systems. At 256 cores, the aggregate vector-based L1 directory could consume more than 256MB of on-chip storage, exceeding the capacity of the L2 caches [23]. On the other side, the invalidations and writebacks for in-cache directory required to victimize a block with active directory state can hurt performance and it is not applicable to the private-L2 configuration, as inclusion of private L2s in other private L2s is not possible.

## IV. CACHE COHERENCE PROTOCOL FOR MANY-CORE PROCESSOR

So far, there are several scalable and energy-efficient cache coherence design strategies catering to many-core processor as follows:

*1) Coherence protocol aimded at avoiding indirection*

By changing the distribution of the roles involved in cache coherence maintenance, DiCo-CMP (direct coherence) avoids indirection for most cache misses [2]. In this way, for most cases, indirection via the home can be avoided. The task of storing up-to-date sharing information and ensuring ordered accesses for every memory block is assigned to the owner cache. DiCo-CMP does not rely on broadcasting but just send requests to the potential owner cache. It depends on prediction mechanism to find the current owner cache by employing a per-core address-based, 1K entry 8-way associative predictor of the current keeper or sharer for 16-way CMP. In addition, they handle data accesses differently by classifying data into private, shared read-only and shared read-write. When data is shared, unnecessary communication and indirection via the directory can be avoided by tailoring the coherence protocol according to the access pattern.

*2) Techniques to reduce directory area overhead*

Many proposals exist to reduce directory overheads by reducing the directory entry size. One example of such design is Tag-less Coherence Directory (TL) for many-core systems with private-L2 cache. It is a scalable directory structure using an implicit, conservative representation for sharing information. As in Fig.3, TL is a grid of Bloom Filters with one column for each CMP core and one row for each cache set. Each Bloom Filter tracks the blocks of one cache set of one core [6]. According to their analytical models, it shows that TL will scale well for processor with increasing numbers of cores in terms of dynamic energy, directory area and bandwidth.

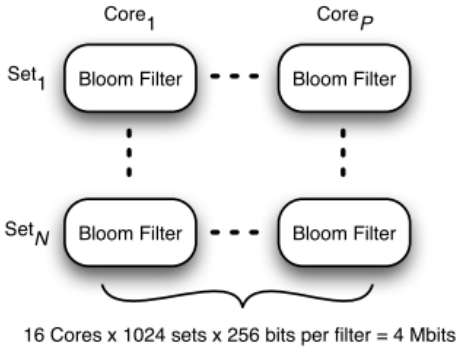


Figure.3 Tagless Coherence Directory [6]

*3) Software and hardware controlled coherence protocol*

The Cohesion is a hybrid memory model for heterogeneous many-core processor. It achieves consistency across hardware-managed and software-managed coherence domains at fine-grained granularity. Cohesion offers reduced message traffic and does not require an on-chip directory when software coherence is used [24]. Cohesion achieves scalability by exploiting scalable parallel applications which lack of fine-grained sharing. When fine-grained sharing is advantageous, it applies hardware coherence techniques.

*4) Techniques for reduced coherence traffic*

Pugsley et al. proposed a directory-less coherence protocol SWEL [25]. The SWEL protocol replaces the directory with a much smaller bookkeeping structure that tracks states for private, read-only and shared read/write cache lines. The read-only data is allowed to be replicated in the L1 caches and the read/write data is only allowed to be present in the L2 cache that is shared by all cores. Their results show that SWEL can improve performance over the directory-based counterpart by

reducing the number of coherence operations when an application has frequent read-write data sharing.

*5) Hierarchical directory*

M.E. Acacio proposed a two level directory for highly scalable cc-NUMA multiprocessors [26]. Milo M. K. Martin et al. proposed a hierarchical directory design [27]. It is designed as a hierarchy of inclusive caches with embedded coherence state whose tracking information is kept precise with explicit cache replacement messages. It reduces directory overheads by reducing the directory information storage, e.g., 512 cores can be supported with 5% extra cache area with two cache levels or 2% with three levels.

*6) Data placement policy exploiting data locality*

The managing schemes for on-chip last-level cache of multicore processors is even difficult to design when on-chip communication delay increases and workloads working sets of commercial and scientific larges. There are papers with a significant interest on NUCA schemes for chip multiprocessors; these works are about cache coherence protocols and duplication of shared cache blocks to reduce access latencies for several cores [28, 29 and 21]. For one example, N Hardavellas propose a design cooperates with the operating system to support intelligent data placement, migration, and replication without the overhead of an explicit coherence mechanism for the on-chip last-level cache [7]. Another example is CloudCache architecture proposed by Pugsley et al. aimed at expediting on chip communication [30]. As in Fig.4, it is a distributed L2 cache substrate for many-core Processor. Cloudcache tackles the large NUCA effect on a switched network by distance-aware data placement, which place private data to the thread's host core and neighbor cores. In addition, it quickly locates nearby data on a local cache miss by a limited target broadcast protocol (LTBP) for private data while process shared data by a conventional directory protocol.

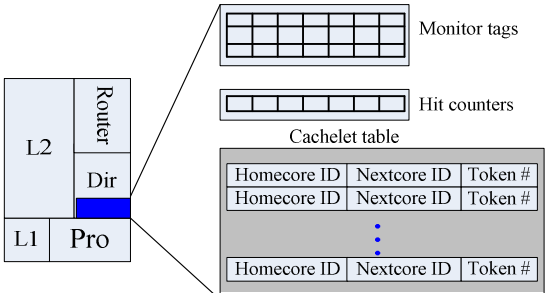


Figure.4 Cloudcache architecture [18]

## V. SUMMARY

The technological parameters and constraints entailed in many-core Processor demanding new solutions to the cache coherence problem. This makes that conventional hardware cache coherence protocol is not suitable for many-core CMP. Because hardware cache coherence remains important for on chip multi-processors, so we need a scalable and adaptive cache coherence protocols which can be low-energy and low access latency and be scalable with processor core number. In other words, we need energy-effective, communication localization, and directory storage effective cache coherence protocol. In conclusion, we can design such a coherence protocol following four directions:

1) Detecting specific sharing patterns to optimize coherence actions. We can exploit application-level asymmetrical behaviors, such as application's access patterns at application-level granularity to boost system performance by adapted adaptability coherence mechanisms.

2) Accommodating workload heterogeneity to coherence protocol design. We can adapt system software OS to guide coherence protocol actions and takes transition between different protocols.

3) Adapting area-effective for directory design to ensure on-chip directory storage is scalable. For example, we can take hierarchical design and tag-less design options for these purposes.

4) Accommodating optimized data placement policy in many-core cache system to decrease remote cache accesses by making private data closer to "home core" and limiting data migrate operations in coherence protocol.

## REFERENCES

[1] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao and B. Edwards, C. Ramey, M. Mattina, C.C. Miao, J.F. Brown, A. Agarwal, "On-Chip Interconnection Architecture of the Tile Processor, " Micro, IEEE, vol.27, no.5, Sept.-Oct. 2007, pp.15-31

[2] Alberto Ros, Manuel E. Acacio and José M. García, " DiCo-CMP: Efficient Cache Coherency in Tiled CMP Architectures, " Proc. of IEEE International Symposium onParallel and Distributed Processing (IPDPS 2008), pp.1-11

[3] James Balfour and William James Dally, "Design tradeoffs for tiled CMP on-chip networks," Proc. of the 20th annual international conference on Supercomputing (ICS '06), ACM New York, NY, USA, 2006, pp.187-198

[4] H. Lee, S. Cho and B.R. Childers, "RFECTORY: A Fault-Tolerant Directory Memory Architecture, " Computers, IEEE Transactions on, vol.59, no.5, May 2010, p.638-650

[5] D. Chaiken, C. Fields, K. Kurihara and A. Agarwal, "directory-based cache coherence in large-scale multiprocessors, " Computer, vol.23, no.6, Jun 1990, pp.49-58

[6] J. Zebchuk, V. Srinivasan and M.K. Qureshi, and A. Moshovos, "A Tagless Coherence Directory," Proc. of 42st International Symposium on Microarchitecture (MICRO '09), New York, NY, 2009. pp.423-434

[7] Nikos Hardavellas, Michael Ferdman, Babak Falsafi and Anastasia Ailamaki, "R-NUCA: Data Placement in Distributed Shared Caches, " Proc. of 36th Annual International Symposium on Computer Architecture (ISCA 2009), New York, NY, 2009.pp.184-195

[8] Sun Microsystems, Inc, "Opensparc T2 system-on-chip (SoC) mi croarchitecture specification," http://www.opensparc.net/opensparc-t2/index.html, May 2008.

[9] Hongzhou Zhao, Shriraman, A. Dwarkadas, S and Srinivasan, V, "SPATL: Honey, I Shrunk the Coherence Directory," Proc. of International Conference on Parallel Architectures and Compilation Techniques (PACT 2011), 10-14 Oct ,2011, pp.33-44

[10] C. Kim, D. Burger and S.W. Keckler, "An Adaptative, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches, " Proc. 10th international conference on Architectural support for programming languages and operating systems (ASPLOS 2002), San Jose, CA,2002, pp.1-12

[11] J. Lira, C. Molina and A. Gonzalez, "Analysis of Non-Uniform Cache Architecture Policies for Chip-Multiprocessors Using the Parsec Benchmark Suite, " Proc of The 2nd Workshop on Managed Multi-Core Systems (MMCS'09), Washington DC: [s,n], 2009:1-8

[12] S. Cho and L. Jin, "Managing Distributed, Shared L2 Caches through OS-Level Page Allocation," Proc. of 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-39), Dec. 2006, pp.455-468

[13] J.A. Brown, R. Kumar and D. Tullsen, "Proximity-Aware Directory-based Coherence for Multi-core Processor Architectures, " Proc. of the nineteenth annual ACM symposium on Parallel algorithms and architectures(SPAA'07), June 2007, ACM New York, NY, USA, pp. 126- 134

[14] O Khan, H Hoffmann, M Lis, F Hijaz, A Agarwal and S Devadas,"ARCc: A Case for an Architecturally Redundant Cache-coherence Architecture for Large Multicores, "Proc. of the 2011 IEEE 29th International Conference on Computer Design(ICCD 2011), IEEE Computer Society Washington, DC, USA, pp.411-418

[15] M Ferdman, P Lotfi-Kamran, K Balet and B Falsafi, "Cuckoo Directory: A Scalable Directory for Many-Core Systems," Proc. of 17th International Symposium on High Performance Computer Architecture(HPCA,2011), pp. 1-12.

[16] R Golla, "Niagara2: A Highly Threaded Server-on-a-Chip," Sun Microsystems, October 10, 2006, unpublished www.opensparc.net/pubs/preszo/06/04-Sun-Golla.pdf

[17] Hongzhou Zhao, Arrvindh Shriraman and Sandhya Dwarkadas, "SPACE: Sharing Pattern-based Directory Coherence for Multicore Scalability," Proc. of 19th International Conference on Parallel Architecture and Compilation Techniques (PACT 2010), Sep. 11-15, 2010. ACM 2010, pp.135-146

[18] C. Fensch and M. Cintra, "An OS-Based Alternative to Full Hardware Coherence on Tiled Processor," Proc. of 14th International Symposium on High Performance Computer Architecture (HPCA'08), Salt Lake City, UT, 2008.

[19] P. Mahoney, Y. Savaria, G. Bois, and P. Plante, "Parallel Hashing Memories: an Alternative to Content Addressable Memories," Proc. of The 3rd International IEEE-NEWCAS Conference (NEWCAS '05), 2005.pp.223-226

[20] R. Singhal, "Inside Intel® Next Generation Nehalem Microarchitecture," Hot Chips 20, Stanford, CA, 2008.

[21] M. Zhang and K. Asanovic, "Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors, " Proc. of the 32nd Annual International Symposium on Computer Architecture(ISCA 2005), June 2005. IEEE Computer Society Washington, DC, USA, pp.336-345

[22] Edwards, B., Amann, J. ,Conlin, R., Joyce, K., Leung, V., MacKay, J.,Reif, M. , Liewei Bao, Brown, J., Mattina, M. , Chyi-Chang Miao,Ramey, C. , Wentzlaff, D. , Anderson, W. , Berger, E. ,Fairbanks, N., Khan, D. , Montenegro, F. , Stickney, J. and Zook, J, "TILE64-Processor: A 64-Core SoC with Mesh Interconnect Tilera Corporation", IEEE International Solid-State Circuits Conference( ISSCC 2008), pp. 88-598

[23 ] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Reactive NUCA: Near-Optimal Block Placement and Replication in Distributed Caches," Proc. of 36th Annual International Symposium on Computer Architecture (ISCA'09), New York, NY, 2009. pp. 184-195

[24] J. H. Kelm, D. R. Johnson, W. Tuohy, S. S. Lumetta, and S. J. Patel, "Cohesion: A hybrid memory model for accelerators," Proc. of International Conference on Computer Architectures(ISCA 2010), ACM New York, NY, USA, 2010, pp.429-440

[25] S. H. Pugsley, J. B. Spjut, D. W. Nellans, and R. Balasubramonian, "Swel: Hardware cache coherence protocols to map shared data onto shared caches," Proc. of International Conference on Parallel Architectures and Compilation Techniques (PACT 2010), ACM New York, NY, USA , 2010, pp.465-476

[26] M.E. Acacio, J. Gonzalez, J.M. Garcia and J. Duato, "two-level directory architecture for highly scalable cc-NUMA multiprocessors, " Parallel and Distributed Systems, IEEE Transactions on, vol.16, no.1, Jan. 2005, pp. 67-79

[27] Milo M. K. Martin, Mark D. Hill and Daniel J. Sorin, "Why on-chip cache coherence is here to stay," Communications of the ACM, July 2012 (in press)

[28] Haakon Dybdahl and Per Stenstrom, "An Adaptive Shared/Private NUCA Cache Partitioning Scheme for Chip Multiprocessors, " Proc. of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture (HPCA '07), IEEE Computer Society Washington, DC, USA , 2007 pp.2-12

[29] J. Chang and G. S. Sohi, "Cooperative caching for chip multi-processors," Proc. of the 33rd annual international symposium on Computer Architecture (ISCA 2006), IEEE Computer Society Washington, DC, USA, pp.264-276

[30] H Lee, S Cho, and B. R. Childers, "CloudCache: Expanding and Shrinking Private Caches," Proc. of 17th International Symposium on High Performance Computer Architecture (HPCA 2011), IEEE Computer Society Washington, DC, USA, pp. 219-230