Adaptive Cooperative Caching for Many-cores systems

Safae Dahmani, Loïc Cudennec, Guy Gogniat

Adaptive Cooperative Caching for Many-cores systems

Safae Dahmani[*,†,1], Loïc Cudennec[*,1],
Guy Gogniat[†,2]

[*] *CEA, LIST, Embedded Real Time Systems Laboratory, 91191 Gif-sur-Yvette, France*
[†] *Lab-STICC Laboratory, University of Bretagne Sud, Lorient, France*

ABSTRACT

Nowadays, many-core processors emerge as a serious alternative to regular processors by offering high computing performances while controlling power consumption. As for every complex parallel and distributed systems, data sharing and data consistency is of major importance. In this paper we propose a short overview of one particular coherency protocol, the data sliding mechanism, based on cooperative caching. We also present some perspectives to this work, and its global integration within a complete system.

KEYWORDS: Many-cores, Cooperative Caching, Data Sliding, Adaptive Protocol.

1 Introduction

New emerging classes of applications are known to need very high performance computing systems. As it becomes difficult to increase core's performance due to complex design and energy consumption, embedding thousands of cores on a single chip is expected to be very prevalent over the next few years. This leads to a new generation of massively parallel systems called many-cores. However, the on-chip memory capacity is still an issue for such systems because of several technological and physical constraints. Thus, high speed cache memories are mainly used to improve on-chip data storage and reduce access latency. Despite the increasing number of embedded cores and the overall caching capacity on the chip, cache size per core is quite limited in many-cores comparing to multi-core processors. Besides, the way cached data are handled directly affects application performance. In such a context, data consistency is one of the big issues that has always interested system designers. Many cache coherency models, are proposed for massively parallel architectures. In the *ALMOS* [Alm11] operating system, data is managed according to transactional memory model, a technique that is also tested in some *AMD* multi-core platforms [CCD$^+$10]. Other works proposed the extensions of the *MESI* protocol; such as the *CoCCA* [MLC$^+$12] and the Data Sliding mechanism [DCG13] that we present in this paper.

[1] E-mail: {firstname.name}@cea.fr
[2] E-mail: {guy.gogniat}@univ-ubs.fr

Data sliding is based on the cooperative caching concept. Multiple distributed caches are aggregated to form virtual shared areas. Cooperating cores are allowed to use each others free cache space in order to balance workloads over the chip. In this short paper, we present our current and future works on cache management techniques. We first describe some relevant works from the state of art. Afterwords, we present the Data Sliding mechanism for cooperative caching in many-cores and some related results. Finally, we discuss extension ideas from this mechanism for a better memory scalability.

2 Hierarchical cache structure

A wide range of processors with tens to hundreds cores are available such as Intel Xeon-phi(60 cores), Tilera TileGx(63 cores), *STHORM* platform(69 cores), and Kalray MPPA chip (256 cores). They are all based on a hierarchical cache organization that consists on up to 3 levels, where memory cost gradually decreases while both access latency and storage capacity increase. The first cache level is private, while lower levels could be private or shared between several cores. Unlike a wholly private hierarchy, shared low-level cache reduces on-chip data replication, which remarkably increases the effective per-core memory size. However, concurrent accesses lead to contention and data consistency problems.

The cooperative caching policy [CS07] has been proposed to get advantages from both private and shared caching, while relying on systems that do not offer physical shared memory. The strategy consists in taking benefits of some unused memory blocks in the neighborhood's private caches. In main energy aware areas, such as wireless networking [TC07] and storage file systems [SLX12], cooperative caching demonstrated to be a relevant approach. In such a context, one of our contributions focuses on important workloads conditions, by proposing new policies for cache entries replacement and cache partitioning.

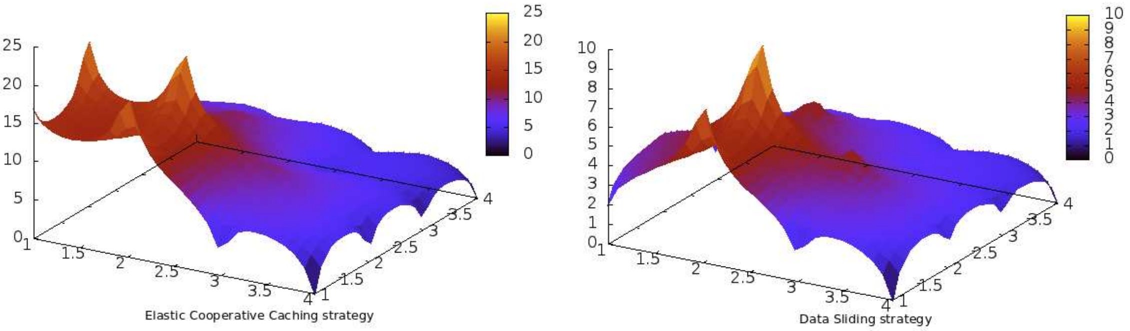
Many static and dynamic strategies have been proposed to manage the cache limits between private and shared zones in order to avoid unused space. The Elastic Cooperative Caching [HGC10] has been presented as an adaptive memory hierarchy, dynamically adjusting local and shared areas. It is based on the data reuse amount information available in each side. Another adaptive cache management strategy is the Adaptive Set-Granular Cooperative caching [RFD12]. It proposes techniques that measure the stress level of each set in a set-associative cache, thereafter used by the system to decide which set to be replaced.

3 Data Cooperative Sliding Mechanism

In a context of short global space storage, where all neighbors are saturated, none of the described strategies allows to efficiently balance cache stress over the chip, while reducing data off-chip eviction rate. In a such highly stressed neighborhood, the Data Sliding strategy allows the migration of private blocks between neighbors, even if there is no free space. When a core sends a storage request to his neighbor, the latter pushes a local block to his closer neighborhood in order to release cache space for the received data. Each block is only 1-chance forwarded, which allows each core to keep its local blocks 1-hop close. This process is repeated until the propagation reaches a non saturated area.

The data-sliding mechanism relies on two policies: the priority-based replacement policy and the best-neighbor selector policy. Both are based on stress measurement of local accesses to private data and remote accesses to hosted data. The implementation consists of a single *Local Hit Counter* (for private data), and 4 *Neighbor Hit Counters*, each one related to a direct neighbor (for hosted data, in the case of a 4-neighbor mesh network). This implementation does not require a global view of the system, which is relevant for scaling up. The replacement policy chooses the least recently used block from the least accessed set of data, either it is private or shared. The best neighbor policy basically chooses the least stressed one, which corresponds to the lowest Neighbor Hit Counter.

First performance evaluations show that thanks to data sliding policies, we reduce by half the global on-chip protocol-related traffic rate compared to the Elastic Cooperative strategy. This is mainly due to neighbor cache cooperation, which promotes neighbor-to-neighbor communications and spread fairly cache workloads across the chip (figure 1). Using the Data Sliding mechanism allows each node to keep their most frequently accessed blocks 1-hop close, instead of evicting them off-chip.



Elastic Cooperative Caching strategy

Data Sliding strategy

Figure 1: Traffic evaluation, using the Elastic protocol (left) and the Data Sliding protocol (right)

4 Extending the Data Sliding Strategy

From the original proposition, we can investigate several directions, either it is a direct extension of the protocol, or a global positioning within a complete system. Such a complete system include low level data management over the chip, but also a higher view, up to the programming language and its compilation toolchain.

In the data sliding mechanism, we allow each block to migrate once to the direct neighbor. As the number of cores grows up, off-chip memory accesses become costly. As a counterpart, a larger cooperative area can be exploited. One current direction is to study the extension of the data migration radius from the closest neighborhood (1-Chance Forwarding) to a larger sliding area (N-Chance Forwarding). Extended data migration is expected to promote on-chip storage, and lead to a balanced spread of data across the chip. We expect to go further by enhancing the global on-chip miss rate and therefore reducing the data access cost. In order to handle efficiently block migrations, we are currently studying a new approach based on the physical model that describes a mass connected to a spring. This approach constrains data migration according to local difference of potential on the chip and the spring constant.

Another work in progress is based on the fact that cooperative nodes are not allowed to access hosted data. They still have to contact the node in charge of managing these data to get the proper rights. Allowing such accesses would bypass the cache protocol and break the consistency model. However, we think that given some particular conditions, these kind of access would benefit to the application performances. We are currently studying these conditions, as well as the mechanisms needed to detect friendly cases.

From these examples we can argue that there wont probably be one data consistency model and one protocol that fit to every combination of application and many-cores architectures. We think that some decisions regarding data consistency can be made at compile time or even later, either based on code implicit indications, or on the applications behavior analysis. This should enhance the application performance, mainly when the choice of an appropriate protocol is dynamically adapted to the deployed workloads and the targeted architecture.

References

[Alm11] Ghassan Almaless. Almos : un système d'exploitation pour manycores en mémoire partagée cohérente. In *8ème Conférence Française sur les Systèmes d'Exploitation (CFSE'11), Chapitre français de l'ACM-SIGOPS, GDR ARP*, 2011.

[CCD+10] Dave Christie, Jae-Woong Chung, Stephan Diestelhorst, Michael Hohmuth, Martin Pohlack, Christof Fetzer, Martin Nowack, Torvald Riegel, Pascal Felber, Patrick Marlier, and Etienne Rivière. Evaluation of amd's advanced synchronization facility within a complete transactional memory stack. In *Proceedings of the 5th European conference on Computer systems*, 2010.

[CS07] J. Chang and G.S. Sohi. Cooperative cache partitioning for chip multiprocessors. In *Proceedings of the 21st annual international conference on Supercomputing*, pages 242–252. ACM, 2007.

[DCG13] Safae Dahmani, Loïc Cudennec, and Guy Gogniat. Introducing a data sliding mechanism for cooperative caching in manycore architecture. In *18TH International Workshop on High-Level Parallel Programming Models and Supportvie Eenvironments (HIPS 2013)*, 2013.

[HGC10] E. Herrero, J. González, and R. Canal. Elastic cooperative caching: an autonomous dynamically adaptive memory hierarchy for chip multiprocessors. *ACM SIGARCH Computer Architecture News*, 2010.

[MLC+12] Jussara Marandola, Stéphane Louise, Loïc Cudennec, Jean-Thomas Acquaviva, and David Bader. Enhancing cache coherent architectures with access patterns for embedded manycore systems. In *In International Symposium on System-on-Chip 2012 (SoC 2012)*, 2012.

[RFD12] D. Rolan, B.B. Fraguela, and R. Doallo. Adaptive set-granular cooperative caching. In *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pages 1–12. IEEE, 2012.

[SLX12] L. Shi, Z. Liu, and L. Xu. Bwcc: A fs-cache based cooperative caching system for network storage system. In *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, pages 546–550. IEEE, 2012.

[TC07] Y.W. Ting and Y.K. Chang. A novel cooperative caching scheme for wireless ad hoc networks: Groupcaching. In *Networking, Architecture, and Storage, 2007. NAS 2007. International Conference on*, pages 62–68. IEEE, 2007.