Exploring the Field of Cache Coherence Protocols for Server Consolidation

Antonio García-Guirado,
Ricardo Fernández-Pascual,
Alberto Ros, José M. García[1]

* *DITEC, Universidad de Murcia, Campus de Espinardo, 30100 Murcia, Spain*

ABSTRACT

It is well known that the number of cores in a chip is increasing rapidly. This will lead chip multiprocessors (CMPs) to reach a hundred of cores soon, and the trend is not expected to change in the next years. Server consolidation is one of the most popular ways to take advantage of such parallel machines by means of running several services or applications in the same server, each one executing inside a virtual machine running in a subset of the cores available. This scenario possesses some characteristics for which current coherence protocols are not optimally adapted. The purpose of this work is to give a view of the main features of such a scenario that should be addressed by coherence protocols in order to make the most out of future CMPs in consolidated environments.

KEYWORDS: Virtualization; Server Consolidation; Multicore; Chip Multiprocessors; Cache Coherence Protocols

1 Introduction

Desktop computers with several cores are common today. Experimental chip multiprocessors (CMPs) with as much as 80 cores have already been presented [Int07]. The trend of increasing the number of cores is likely to continue for at least several years. In this scenario, server consolidation has risen as one of the main ways to use the huge amount of cores of current and future CMPs. Instead of executing a single massively-parallel application, many virtual machines (VMs) are executed in subsets of the cores of the server. A lot of services like web servers, databases, decision support systems or even whole desktops for remote users are routinely executed in VMs nowadays.

[1]E-mail: {toni,r.fernandez,a.ros,jmgarcia}@ditec.um.es

Cache coherence protocols for these servers should be designed taking into account these scenarios. This would allow many optimizations that would result in improved performance, less storage overhead, reduced network requirements and less power consumption.

2 Objectives of a cache coherence protocol in a virtualized environment

We have identified six different objectives that a coherence protocol for consolidated servers should accomplish. These objectives are related to cache occupation, miss latency, performance isolation, coherence protocol complexity, coherence information overhead, and reconfiguration of the server. The next subsections give a brief view of these objectives.

2.1 Reduplication of deduplicated data

Server consolidation shows some characteristic features that a coherence protocol can take advantage from. One of them is memory deduplication [Wal02]. When several VMs are running in the same machine, it is very likely that two or more VMs use portions of memory that contain exactly the same information. For example, if two VMs execute the same operating system, they will have several memory pages used by the OS code that will have identical contents in both VMs. This information can be *deduplicated*, which means that it is present only once in memory. This deduplication is typically made at a memory page basis by the hypervisor, which is in charge of checking whether two different virtual pages have the same contents and, if so, allocate a single physical page in memory for them. This results in significant memory savings and is routinely used nowadays in most hypervisors such as Xen, VMware or even the Linux kernel.

Related to this, there has been plenty of research about the trade-offs of using shared or private caches. On the one hand, a shared cache benefits those scenarios in which the amount of data used by the cores is unbalanced. On the other hand, private caches provide faster access to the data. Many techniques have been proposed in order to gather the benefits of both approaches in a single cache hierarchy [CS06]. In the case of server consolidation, memory deduplication poses a new element to this discussion, since deduplicated data is reduplicated when using private caches. If the coherence protocol causes data reduplication, the same data is present multiples times in cache, hence increasing cache misses to other data. As virtualization density increases, the influence of deduplicated data also increases, because both the opportunities to deduplicate data and the number of copies of that data that are present in cache grow.

Therefore, a cache coherence protocol for a consolidated server should avoid the reduplication of deduplicated data.

2.2 Distance to accessed data

Each VM executes in a subset of the tiles of the server. This means that the data used by each VM is located in the tiles where it executes in the private levels of the cache hierarchy. The main drawback of these levels of cache is that they reduplicate shared data in every tile that accesses it. However, as soon as there is a shared level of cache in the CMP, the data of

every VM is spread across the whole chip. Hence, accessing to a block in such a shared level of cache requires an access to an arbitrary tile of the chip, instead of only to a tile in which that VM is running, thus increasing the latency to solve the miss and creating interference between VMs. As the number of cores and the virtualization density increase, this effect becomes more prominent. An intermediate approach is to use caches that are private to the VM, but shared between the cores of a VM. This prevents the duplication of shared data (contrary to caches private to the tile) but incurs reduplication of deduplicated data. Therefore, new mechanisms to bring the data closer to the VMs and yet avoid reduplication of deduplicated data are necessary. This reduction in distance shortens the links traversed to solve misses, therefore improving performance and reducing network usage.

2.3 Isolation of virtual machines

A misbehaving VM can affect the performance of the others. For example, a faulty VM whose working set grows out of control could pollute the shared levels of cache, thus harming other VMs' performance. Therefore, mechanisms at the coherence protocol level should be developed to isolate the performance of the VMs. One simple way to achieve this is by giving each VM its own cache at every level, thus preventing one VM from polluting the cache space reserved to another VM. However, as noted before, this would reduplicate deduplicated data in every level of cache. Additionally, mechanisms at the network level can be used to prevent the traffic from one VM to interfere with the traffic from another one.

2.4 Protocol simplicity

A cache coherence protocol should be as simple as possible. Complex designs are too hard to prove correct with formal methods. Hierarchical protocols show a noticeable complexity, especially if the boundary between the different levels of the protocol is not clear, and therefore the protocol cannot be easily divided in simpler protocols to prove their correctness separately, and then extrapolate the correctness to the original protocol by means of some recently developed methods to do so [Che08]. Our belief is that one single level in the protocol is desirable in order for the protocol to be actually implementable in a real machine.

2.5 Reduction of the overhead due to the coherence protocol

A major problem for current proposals to keep cache coherence is that they do not scale well with the number of cores. The overhead introduced by directory based protocols in terms of space in the cache grows linearly. In addition to their traffic requirements, token based protocols show scalability problems due to the tables needed by the mechanism used to perform persistent requests.

Apart from deduplicated data, all data used by VMs is private data, and one VM is restricted to a set of cores. In such a case it is a waste of space trying to keep coherence information about sharers for the whole chip. Therefore, mechanisms to track the sharers in a single VM can significantly reduce the space overhead of the coherence protocol.

2.6 Dynamic reconfiguration of the server

The arrangement of VMs running in the server cannot be known beforehand. The same server might execute two VMs or twenty. Additionally, the transitions between two different scenarios can happen at any time.

To avoid wasting resources, dynamic reconfiguration of the server should be possible. Resources should be dynamically reallocated in order to match the current distribution of VMs running in the server. For example, if VMs are given private caches and a change of arrangement of VMs takes place, the private caches should match the new arrangement of VMs. On the other hand, mechanisms to reduce the coherence protocol overhead can negatively affect the ability to rearrange resources. Therefore, a trade-off or new mechanisms are needed to design the best possible coherence protocol that achieves both objectives.

3 Conclusion

We have pointed out six different objectives that a coherence protocol for consolidated servers should accomplish. These are: avoiding the reduplication of deduplicated data, bringing the data closer to the cores that use it, isolating virtual machines, keeping the protocol simple, reducing coherence information to track VMs' sharers and allowing for dynamic reallocation of resources. Necessarily, trade-offs between all the desirable features must be explored, since some of these objectives face opposite directions. Nevertheless, mechanisms can be developed to go around this problem and integrate some of them together in not yet explored ways. These objectives would lead to improved performance, less coherence overhead, less network requirements, less power consumption and shorter design time. Already, our research has produced some protocols that successfully fulfill some of the requirements exposed in this document, and the results will soon be published.

References

[Che08] Xiaofang Chen. *Verification of Hierarchical Cache Coherence Protocols for Futuristic Processors*. PhD in Computer Science, University of Utah, 2008.

[CS06] Jichuan Chang and Gurindar S. Sohi. Cooperative Caching for Chip Multiprocessors. In *Proceedings of the 33rd Annual International Symposium on Computer Architecture (ISCA)*, pages 264–276, 2006.

[Wal02] Carl A. Waldspurger. Memory Resource Management in VMware ESX Server. In *5th Symposium on Operating System Design and Implementation (OSDI)*, pages 181–194, 2002.

[web07] Intel's Teraflop Research Chip. `http://techresearch.intel.com/articles/Tera-Scale/1449.htm`, 2007.