International Journal of Current Research and Modern Education (IJCRME)
ISSN (Online): 2455 - 5428
(www.rdmodernresearch.com) Volume I, Issue I, 2016

DIRECTORY BASED CACHE COHERENCY PROTCOL IN MULTI-CORE SYSTEM FOR HIGH PERFORMANCE COMPUTATION

Subrahmanya Bhat* & Dr. K. R. Kamath**
*Department Computer Application, Srinivas Institute of Management Studies, Mangalore, Karnataka
**Professor, Department of Computer Science, Srinivas Institute of Technology, Mangalore, Karnataka

Abstract:

Today's systems are designed with Multi Core Architecture. The idea behind this is to achieve high system throughput. Once the Processor clock speed reached its saturation, designers opted for having multiple cores. Each Core or Processor equipped with their own private cache memory. But under Chip Multiprocessor, where all the processor have access to shared memory, having respective cache memory will result with Cache Coherency Problem. Cache Coherency Problem is nothing but maintaining data consistency in spite of allowing multiple processor to have a access to common memory. This problem is addressed by software or hardware. The later method is usually preferred than the former, so as to get rid of programming issues. In hardware method 2 options are usually adopted like Snoopy Protocol or Directory Protocol. In Directory Protocol, for each block of data there is a directory entry that contains a number of pointers. The purpose of this number is to mention the locations of block copies. Each directory entry also contains a dirty bit to specify whether a unique cache has a permission or not to write the associated block of data. There are three primary categories of directory-based protocol: Centralized Directories, Flat Directories, and Hierarchical Directories. The important advantage of directory based protocols is that they scale much better than snoopy protocols. In addition to this it has the advantage of ability to exploit arbitrary point-to-point interconnects. But mean time it also has the overhead in terms of the storage and manipulation of directory state. This paper explains different Directory Based implementation and its advantages over Snoopy Protocol.

1. Introduction:

Multi-processor systems use two or more central processing units that communicate with each other through a bus or general interconnection network.in order to gain high performance by increasing the number of transistors and clock frequencies. Various design constraints such as high power consumption, heat dissipation, etc., restricts the designers from increasing frequency of the clock beyond certain limit. This limitation led to the development of embedding multiple processing cores onto a single chip. Such multiprocessors are called as Chip Multi-Processors (CMPs). CMPs increase throughput and efficiency of the system by utilizing multiple simple cores to perform parallel processing on a larger task with less power and heat dissipation. In CMP each processor core has its own cache memory that is not shared with any other processor cores. This cache memory available with each core enables fast data access by reducing disk access latency in case of a cache hit. The efficiency of the CMPs depends on type of cache mechanism employed. These protocols can impact the performance of a multiprocessor system and it is hard to estimate. The performance of a system is directly proportional to the latency of microprocessor accesses on memory. The latency of an access is dependent on congestion in the system which is directly related to the amount of communication traffic involved in Coherency Protocols. Hence improving the latency of accesses and reducing the traffic can thus reduce the cost of the system by reducing the bandwidth requirements at large. This paper will address the issues related to Snoopy Protocols and its different versions with their pros and cons.

2. Coherency Problem:

A typical shared memory multiprocessor contains multiple levels of caches in the memory hierarchy. Each processor may read data and store it in its cache. This results in copies of the same data being present in different caches at the same time. The problem occurs when a processor performs a write to data. If only the value in the writing processor's cache is modified, no other processor will see the change. If some action is not taken, other processors will read a stale copy of the data. Intuitively, a read by another processor should return the last value written. To avoid the problem of reading stale data, all processors with copies of the data must be notified of the changes

3. Coherency Protocols:

Cache coherence protocols are classified based on the technique by which they implement as Snooping and Directory based protocols. In Snooping based protocols, address lines of shared bus are monitored by cache for every memory access by remote processors. The action is taken when locally saved data is changed by the transaction started by the remote processor. In Directory based protocols, a main directory is maintained containing information on shared data across processor caches. The directory works as a look-up table for each processor to identify coherence and consistency of data which is currently being updated. A directory-based protocol is a smart way of implementing cache consistency on an arbitrary interconnection network. This Directory Protocols are bit complex, but they have the advantage of scalability factor. As the number of processor increase, the snoopy based protocol suffer with band width limitations and hence in such case going for Directory based Protocol is the alternative.

4. Directory Based Protocol:

Directory Based Protocol is based on tracking which processor cache contain a memory line, to send the number of necessary messages, and avoid broadcasts. Sharing information is kept in an auxiliary data structure called a directory. Directory information can be distributed to multiple directory engines to avoid the performance bottleneck of a single, monolithic directory. Each node or group of nodes is associated with a directory corresponding to the locations in that node's group local memory. The directory consists of a collection of directory entries, one for each memory block in the node's local memory. In its simplest form, a directory entry contains two fields - a state indication and a presence bit vector. In invalidation-based protocols the state indication specifies whether the memory line associated with the directory entry is held shared (i.e., read-only) in one or more caches or whether it is held exclusive (i.e., with read/write permission) in a single processor's cache.

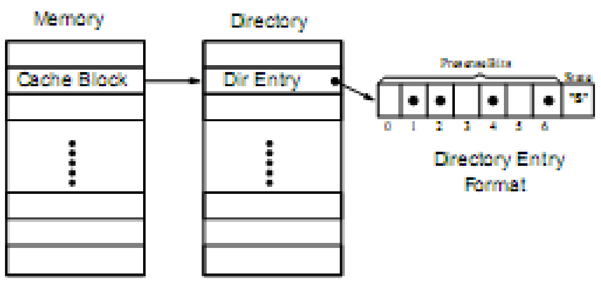

Figure 1: Directory Entry Structure

The presence bit vector indicates which processors are caching the memory line; if the memory line is held exclusive, only one presence bit may be set. The directory entry depicted in Figure shows a case in which the corresponding memory line is held shared, indicated symbolically by the S in the state field, and is present in the caches of processors 1, 2, 4, and 6, indicated by the presence bit vector. When a memory request arrives at a processing node, the controller of the node then retrieves the corresponding directory entry to determine what ad ditional actions are required to service the request. For example, as shown in Figure, if processor 3 requested exclusive access to the memory line, the memory line must be removed, or invalidated, from all processor caches currently holding it. In a distributed system, the controller of the node must consult the presence bit vector to determine that explicit invalidation messages need to be sent to processors 1, 2, 4, and 6. In a bus-based system, this invalidation's would be performed automatically when processor 3's exclusive request was issued on the bus.

5. Directory Organizations:

Directory-based cache coherence protocols have been used for long in shared memory multiprocessors. These protocols introduce directory memory overhead due to the need of keeping the sharing status of a memory block in a directory structure. In the past, this structure would provide an entry for every block of main memory and, because of its size, was kept in DRAM. The directory information represent memory overhead as it adds state information either for each cached or also for each non-cached memory block in the system, depending on the directory organization. However, this overhead could become very high depending on both the sharing code and the number of cores that comprise the multiprocessor system, and even be in large systems prohibitive. In this section, we study a directory organization for CMPs that addresses the problem discussed above. Then it reviews the main alternatives for storing the directory information and offers a proposal to optimize look-up time for the directory organization used in this work. Moreover, the straightforward way of tracking sharers of a block is by using a full-map sharing code where each bit represents a core in the system, which is set when that cache holds a copy of the block. The size of this directory structure scales with the number of cores (P) in the system. In particular, the order of its size is (P X M), where M is the number of memory entries and P is the number of cores in the system. Based on the location of directory, directories schemes are the the centralized and the distributed schemes, where the memory is distributed and multiple directories are responsible for a portion of the address space. As shown in Figure 2, the two alternatives for finding the source of the directory information for a block are known as flat directory schemes and hierarchical schemes. The taxonomy that is showed, also divides Flat schemes into two categories based on the way they use in order to locate the copies of the memory blocks.

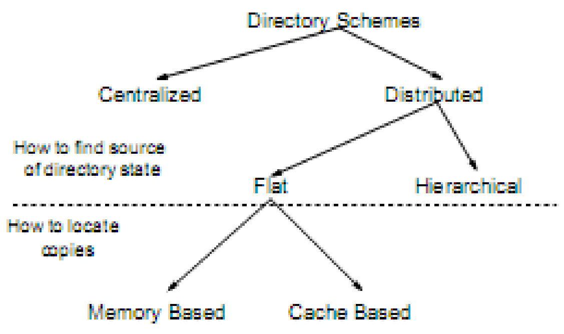

Figure 2: Directory Organization

6. Flat Schemes:

Flat schemes are more popular than hierarchical, and they can be classified into two categories: memory-based schemes and cache-based schemes. Memory based schemes store the directory information about all main memory blocks, or only cached copies, at the home node of each block. In cache-based schemes (also known as chained directory schemes), the information about cached copies is not all contained at the home but is distributed among the copies themselves. The home node contains only a pointer to the first sharer in a distributed double linked-list organization with forward and backward pointers. The locations of the copies are therefore determined by traversing the list via network transactions. The most important advantage of cache-based directory schemes is their ability to significantly reduce directory memory overhead, since the number of forward and backward pointers is proportional to the number of cache entries, which is much smaller than the number of memory entries. The problem of the directory memory overhead in memory-based schemes is usually managed from two separate points of view: reducing directory width and reducing directory height. The width of the directory structure is given by the directory entries and it mainly depends on the number of bits used by the sharing code. The height of the directory structure is given by the number of entries that comprise the directory.

7. Hierarchical Scheme:

Hierarchical memory schemes treat the processing cores as the leaves of a logical tree, with main memory distributed along with the processing nodes. Every block is assigned to a home node (leaf) in which it is allocated, but this does not mean that the directory information is maintained or rooted there. The internal nodes of the tree are not processing cores and only hold directory information. Each such directory node keeps track of all memory blocks that are being cached or recorded by its sub-trees and it uses a presence vector per block to tell which of its sub-trees have copies of the block and a bit to tell whether one of them has it dirty. It also records information about local memory blocks that are being cached by processing nodes outside its sub-tree. This information is used then to decide when requests originating within the sub-tree should be propagated further up the hierarchy. In general, the advantages of hierarchical schemes are tightly related to the amount of locality shown by memory accesses, as the delay is high if all the buses/levels that need to be traversed to serve a high percentage of the memory accesses. The main drawback of such schemes is the latency problem, because the number of network transactions sent up and down the hierarchy to satisfy a request tends to be larger than in a flat memory-based scheme. Even though these transactions may be more localized in the network, each one is a network transaction that also requires either looking up or modifying the directory at its (intermediate) destination node. This increased endpoint overhead at the nodes along the critical path tends to prevail any reduction in the total number of network hops traversed and hence network delay, especially given characteristics of modern networks.

8. Conclusion:

A typical multiprocessor systems contains multiple processors with levels of caches in the memory hierarchy to improve the system performance. Such multiprocessor systems where all the processor have access to shared memory, having respective cache will result with Cache Coherency Problem. Cache coherency Problems are typically addressed by Snoopy based and Directory based protocols. As the number of processor increase, the snoopy based protocol suffer with band width limitations and hence in such case going for Directory based Protocol is the alternative. With Directory based, Centralized scheme is very straight forward but has the bottleneck and single point failure. With Flat Directory Scheme one cane have the option of implementing memory-based schemes and cache-based schemes. The advantage of cache-based directory schemes is their ability to significantly reduce directory memory overhead. The advantage with hierarchical schemes are tightly related to the amount of locality shown by memory accesses, as the delay is high if all the buses/levels that need to be traversed to serve a high percentage of the memory accesses.

9. References:

1. Design and Implementation of a Directory based Cache Coherence Protocol by Dimitris Tsaliagos Technical Report FORTH-ICS/TR-418 May 2011
2. Effects of cache coherency in Multiprocessors, By Michel Dubois, Member- IEEE, and Faye A. Briggs, Member-IEEE
3. Prof. M. Shaaban's EECC 756 Lecture notes on Cache Coherence Problem in Shared Memory Multiprocessor.
4. Parallel Computer Architecture (PCA) BY David E. Culler and Jaswinder P. Singh (1999 edition).
5. http://parasol.tamu.edu/~rwerger/Courses/654/cachecoherence1.pdf
6. CS252 Graduate Computer Architecture. A course by David A. Patterson in CS Department of UC Berkeley.
7. Wikipedia.com