Snoopy Protocol for Cache Coherency in Multi Core Systems for High Performance Computation

Subrahmanya Bhat
Dept Computer Application
Srinivas Institute of Mgt Studies, Mangalore
Karnataka, India. 575001

Dr. K. R Kamath
Professor, Dept of CS
Srinivas Institute of Technology
Mangalore, Karnataka, India

ABSTRACT

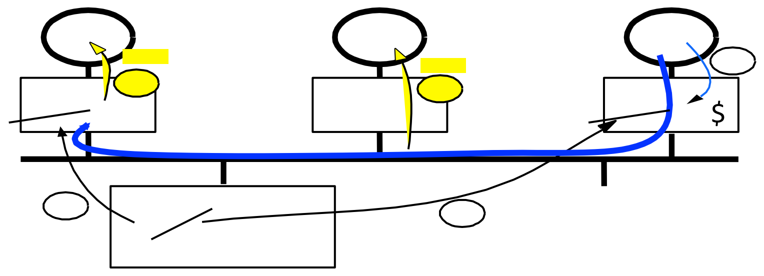
Todays systems are designed with Multi Core Architecture. The idea behind this is to achieve high system throuput. Once the Processor clock speed reached its saturation, designers opted for having multiple cores. Each Core or Processor equipped with their own private cache memory. But under Chip Multiprocessor, 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 Method. In Snoopy protocol, the Write Update will consume much memory bandwidth and also latency. Due to this reason Invalidate protocols are getting popular. Invalidation Protocols need to address the problem of providing up to date data during Read or Write miss. Invalidation with Write Through or Invalidation with Write Back are the 2 options to serve this. Invalidation with Write Through requires more memory bandwidth. On the other hand Invalidation with Write Back consume lesser memory bandwidth but may result with higher latency in case chips are located at far away.

I. 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 dependant 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.

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



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.

Snoopy Protocols

In bus based multiprocessor systems, appropriate coherence actions can be taken if coherence is detected. These are called snoopy protocols. The name snoopy comes from snoop, because each cache snoops bus transactions to watch memory transactions of other processors. Snoopy protocols require the use of a broadcast medium in the machine and hence apply only to small-scale bus-based multiprocessors. Here all the Cache Controllers will be connected to the common address bus and hence observe each memory access for both read and write by other processors. This type of protocol is most commonly used method in commercial multiprocessor. Various snoopy protocols have been proposed. The two primary categories in Snoopy Protocol are Write Update and Write Invalidate.

Write Update Protocol

In case of Write Update, the Cache Controller which Updates a shared value needs to update all other Cache Memory where the same value is copied. The advantages of this method causes no cache misses. But this Protocol requires higher memory band width and also Memory latency because it needs to send data through the data bus for every updating on shared cache. There is also a possibility where the cached memory value at other Processors might not be used for its further execution. In such case the updating will result in nothing but waste of time. So the other version of the protocol is Write Invalidate. Due to this most multiprocessor systems nowadays implement a write invalidate protocol.

Write Invalidate Protocol

Here in order to reduce the memory latency and bandwidth whenever a cache content is updated by a local processor, all another cache contents which share the same data is made invalid. Just by having one additional bit for valid/invalid the cache content will reflect whether the data is up to date or not. If the cache block is invalid, in such case the same cache reference by the processor will result with a cache miss. This requires the protocol to provide up to date data to the requesting processor either from memory or from the remote processor cache where the same data is in valid state. Again here we have 2 versions of protocol like: Write Through / Write invalidate and Write Back / Write invalidate depending on how the memory content is being updated.

Write Through / Write invalidate

When a processor writes a new value into its cache, this value is written into the memory module, and all copies in the other caches are invalidated. Again broadcasting can be used to send the invalidation requests through the system.

II. Limitations

Every write from every processor goes to shared bus and memory. Consider 200MHz, 1 CPI processor, and 15% of the instructions are 8-byte stores. Each processor generates 30M stores or 240MB data per second. 1GB/s bus can support only about 4 processors without saturating. Write-through especially is unpopular for SMPs.

Write Back / Write invalidate

In the write-back protocol, multiple copies of a cache block may exist if different processors have loaded (read) the block into their caches. If some processor wants to change this block, it must first become an exclusive owner of this block. When the ownership is granted to this processor by the memory module that is the home location of the block. All other copies, including the one in the memory module, are invalidated. Now the owner of the block may change the contents of the memory. When another processor wishes to read this block, the data are sent to this processor by the current owner. The data are also sent to the home memory module, which requires ownership and updates the block to contain the latest value.

III. Conclusion

In implementing the Cache Coherency, Update looks the simplest, most obvious and fastest, but multiple writes to the same word need only one invalidate message in Invalidation method but here it would require an update for each writes. Bus bandwidth is a precious resource in shared memory multi-processors. Experience has shown that Invalidate protocols use significantly less bandwidth. Invalidation with Write Through requires more memory bandwidth. On the other hand Invalidation with Write Back consume lesser memory bandwidth but may result with higher latency in case chips are located at far away.

References:

Effects of cache coherency in Multiprocessors, By Michel Dubois, Member- IEEE, and Faye A. Briggs, Member-IEEE

Prof. M. Shaaban's EECC 756 Lecture notes on Cache Coherence Problem in Shared Memory Multiprocessor.

Parallel Computer Architecture (PCA) BY David E. Culler and Jaswinder P. Singh (1999 edition).

http://parasol.tamu.edu/~rwerger/Courses/654/cachecoherence1.pdf

CS252 Graduate Computer Architecture. A course by David A. Patterson in CS Department of UC Berkeley.

Wikipedia.com

Google.com