Analysis the Memory over head of Cache coherence protocols to minimize the problem of coherence in multiprocessor system

*Mamta Rani, **Er. Kavita Khatkar
*Student of M.teck(CSE), Assistant Professor deptt. of (CSE)
**JCDM college of Engineering, Sirsa
Mamtasdg78@yahoo.com

Abstract: The major problem of cache coherence occurs in shared memory multiprocessors system. In Shared memory multiprocessor each of the processors can directly access any location in the common memory address space using a single read or write instruction. Since each processor has a private data cache, a copy of the same shared memory location may be present in one or more of the cache at the same time. The private data caches introduce a coherence problem in which it is possible for the different values at the same time. To minimize the cache coherence problem in the multiprocessor system present an overview of many different types of mechanisms proposed to solve this problem. By using the Hardware based protocols to calculating the memory overhead and minimizes the problem of coherence in multiprocessor system.

1. INTRODUCTION

Cache is a small and fast storage device that holds the operands and instructions most likely to be used by the CPU. Cache Memory required between the CPU and Main Memory because the processing speed of CPU is much fast as compare to the data fetching speed of the main memory so to remove that speed miss match cache memory is used between the CPU and main memory. Cache is fast memory, as it is small in size and due to principle of locality. [1][7]. the shared memory multiprocessor is used because it is a cost effective way of performing the jobs on computer. Secondly by using multi processor the speed of computer increase or we can say that computer does the tasks very fast as compare to the single processor systems. we know that an address, which is generated by the program, has two basic properties, temporal and spatial locality. Thus to exploit these features cache used in shared memory system [8] [4].

Temporal locality: Memory location that is referenced once is likely to be referenced multiple times in near future.

Spatial locality: Memory location that is referenced once, then the program is likely to be reference a nearby memory location in near future [5].Thus we can say that the memory address generated by a program to access a data item is likely to be clustered in to a small range of address space. Thus by using private cache, exploit these memory references to reduce the average time required to access the main memory, by temporarily storing in the cache a copy of value from the main memory that is being actively reference by a program.

2. CACHE COHERENCE PROBLEM

In a shared memory multiprocessor system, all the processors share a common memory (Shared Data) or same memory address space, in addition, each processor may have a local memory, part or all of which (main memory) may be a cache. The compelling reason for having separate caches for each processor is to reduce the average access time in each processor. The same information may reside in a number of copies in some caches and main memory. The ability of the system is to execute memory operations correctly, and keep multiple copies identical.

Cache coherence problem occur only when on a shareable data item is written by a processor and at the same time other processor read the same data item and one processor does not inform to the other processor during or befor updating the data item .Thus the read only data in a shared memory system cannot create the coherence problem it is created mainly by the shared read-write data strctures.Private read–write data structure might impose a cache coherence problem if it is allowed to processes to migrate from one processor to another[2].

An Example of Cache Coherence Problem

Time	**Po**	**P1**	**P2**
T1	**Read A (55)**		
T2		**Read A (55)**	
T3	**Write A (65)**		
T4	**Read A (65)**	**Read A (55)**	**Read A (55)**

Table 1.1 Sequence of Reads and Writes by the different Processors

As show in table 1.1, a system which has three processors (Po P1, P2) and each processor have it own private cache .In the begin each cache memory and main memory have the value A=55 for variable A, as shown in the table 1.1. At time T1 and T2 processor Po and P1 reads the value of A from its private cache which is (A=55) and thus complete the reads operation successfully. But at time T3 processor Po performs the write operation on A's value and make it (A=65) it create problem for the other processors like P1 and P2 because they read old value at the same time (A=55) at T4. That is a example of cache coherence problem there are many ways to remove or avoid that problem like after modify the value processor can update the value in main memory or during the updating the memory location no read operation allowed to any processor. Thus by using such techniques we can avoid the cache coherence problem. Secondly there are many cache coherence protocols are available. These protocols are used to maintain the consistency among the private cache and main memory, which is on the same network [2].

3. Directory Based Cache Coherence Protocols

A Cache-coherence protocol that does not use broadcasts must store the locations of all cached copies of each block of shared data. This list of cached locations, whether centralized or distributed, is called a directory. A directory entry for each block of data contains a number of pointers to specify the locations of copies of the block in different caches. Directory schemes differ in how much information they maintain about blocks, where the information is shared and which invalidating or updating strategy is used to ensure coherence resulting in memory requirements and performance. Each directory entry also contains a dirty bit to specify whether or not a unique cache has permission to write the associated block of data. Before a processor can write to its cached block, an exclusive access to block is required to directory. A directory then sends, a message to all processors with a cached copy of block and forcing each processor to invalidate its copy. After receiving acknowledgement from all of these processors, the directory grants exclusive access to the writing processor. In this paper we select three directory based protocols to calculate their memory overhead and compare it with each other, by doing this we can easily find, which protocol is suitable for the particular no of processor. By using following parameters we can find the memory overhead for various protocols [3].

p=Processor used in Multiprocessor shares memory system

k=m/c

m=no of blocks in memory module

c=no of cache block in each Processor cache

b=words per block

w=bits per block.

Different directory schemes are

- Full-Map
- Limited
- Chained Directory

3.1 FULL-MAP DIRECTORY

Full–Map directory schemes, uses directory entries to represent the data which is in the caches, for these entries two bits are used, one bit for processor which show the status of the block in the corresponding processor's cache (Absent or present). Second bit is dirty bit, which is used to show the write persimmon, if the dirty bits set, it means that the processor has permission to write in to the block. Thus cache maintains the two bits per block to show state of the block [3].

3.2 LIMITED DIRECTORY

Limited directory protocols are designed to solve the directory size problem. Restricting the number of simultaneously cached copies of any particular block of data, limits the growth of the directory to a constant factor**.** A limited directory protocol is based on pointer replacement policy. A limited directory protocol that uses i<n pointers is denoted as DiriNB. Where "i" is stands for the number of pointers and "n" stands for " number of processor in the system.A directory protocol can be classified as DiriX ., and X is NB for a scheme with no broadcast and B for one with broadcast. A full-map scheme without broadcast is represented as DirnNB [3].

3.3 CHAINED DIRECTORY

Chained directories, the third option for cache-coherence schemes that do not utilize a broadcast mechanism, based on the scalability of limited directories without restricting the number of shared copies of data blocks. This type of cache-coherence scheme is called a chained scheme because it keeps track of shared copies of data by maintaining a chain of directory pointers. The chained directory is simple to implement [3].

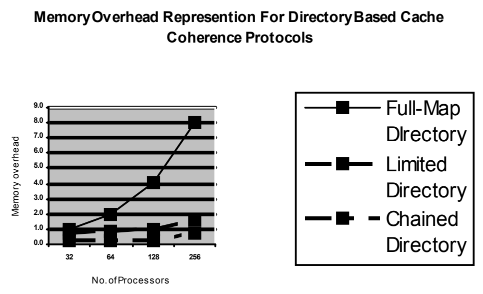
4. COMPARISION OF D.B CACHE COHERENCE PROTOCOLS

As we know to minimize the cache coherence problem different type of directory based cache protocol are available, but each protocol has its drawbacks in the form of memory overhead requirements. Thus we have studied four directory-based protocols, but for memory overhead calculations we just use three of them (full map, chained, limited directory) and after this calculation we compare them on the basis of memory overhead to find out the most suitable protocol according to the size of shared memory system. Ultimately we find that as the no of processors are varies the overhead also changes, and for the same no of processors different protocols has different memory overhead requirements. Thus by comparing their overhead we can easily find out which protocol is suitable for a shared memory system.

er of Processors				
Scheme	**P=32**	**P=64**	**P=128**	**P=256**
Full-Map Directory	1.027	2.024	4.016	8.0002
Limited Directory	0.68556	0.80958	0.93409	1.0586
Chained Directory	0.2188	0.25512	0.28137	0.31262

Table1.1 The memory overhead for various schemes for given number of processors

The table 1.1 describes the values of memory overhead for various protocols, and we find out for the same no of processors Full-map, Limited, Chained protocols have different values. Thus it can be easily find out which protocol is best for a no of processor by comparing their overhead values.



5. CONCLUSION

As we know today thousands of computers are connected through networks, in the form of shared memory multiprocessor system .The basic idea behind the shared memory multiprocessor system is to done the tasks or jobs at very rapid speed. Thus to fulfill the user's request quickly cache memories are being used in multiprocessor systems, but These cache memories impose the cache coherence problem in shared memory system and create the inconsistency among the data. Inconsistence may be occurring between the shared main memory and private caches or between the private caches, which are also on same network. Thus the local caching of data not only increases the data access speed but also introduces the problem of cache coherence. Early shared-memory machines left it to the programmer to deal with the cache coherence problem, and consequently these machines were considered difficult to program. Today's multiprocessors solve the cache coherence problem by hardware and software cache coherence protocol. The hardware protocols can be further divided into two categories directory based and snoopy base. These protocols have their advantages and disadvantages in the form of memory overhead and cache miss (read/write). Thus on the basis of these advantages and drawbacks we must chose the appropriate protocol for a particular network. After analysis and comparing the memory overhead for various schemes show how well these schemes scale, as the no of processors grows full memory-directory scheme has a sharp increase in a memory overhead hence it is poorly scalable. Limited and chained protocols have similar values. Limited directory has the disadvantages of, only small set of processor can share a data block so it is not preferable .Chained directory have lower memory overhead and it scales gracefully with additional processors. Chained directory can be candidate choice in term of lower memory utilization.

REFERENCES

M.Morris Mano; Computer System Architecture 3rd Edition; Dorling Kindersley (India); 2008.

David j.Lilja ; Cache Coherence in large –Scale Shared Memory Multiprocessor :Issues and Comparisons., ACM Computing Surveys, Vol. 25, No. 3, September 1993, pp. 303-338.

David Chaiken, Craig Fields et.al.; Directory Based cache coherence in large scale Multiprocessors; IEEE Computer Society Press; Volume 23; 1990; PP 49-58.

http://cs.colgate.edu/faculty/nevison/cs201web/ Sampriya Chandra.ppt

www.cmpe.boun.edu.tr/courses/cmpe511/fall2004/Mehmet%20Senvar%20-%20Cache%20Coherence%20Protocols.doc

http://cs.colgate.edu/faculty/nevison/cs201web/MehmetSenvar–CacheCoherence Protocols.ppt.