Cache Coherence Protocols in Multiprocessor System

Last Updated : 03 Jun, 2024

Prerequisite - Cache Memory

In multiprocessor system where many processes needs a copy of same memory block, the maintenance of consistency among these copies raises a problem referred to as **Cache Coherence Problem.**

This occurs mainly due to these causes:-

- Sharing of writable data.
- Process migration.
- Inconsistency due to I/O.

Basic Schemes in Enforcing Coherence Protocol

The coherence problem for multiprocessor, the I/O although similar in origin, has different characteristics that affect the appropriate solution. In a coherent multiprocessor, the caches provides both migration & replication of shared data items.

Coherent caches also provide migration, since a data item can moved to local cache and used there in a transparent fashion.

This migration reduces both the latency to access a shared data items that allocated remotely and bandwidth demand on shared memory. Multiprocessor adopt a hardware solution by introducing a protocol to maintain coherent caches. The protocol to maintain coherent for multiple processor are called coherence protocol.

There are two classes of protocols in use each of which uses different techniques to track the sharing status.

1)Directory based

2)snooping

1)Directory based

The sharing status of a particular block of physical memory is kept in one location called the directory. There are different types of directory based cache coherence.

Associated with the memory or some other single serialization point, such as the outermost cache in a multicore.

2)snooping

Rather than keeping a state of sharing in a single directory, every cache that has of copy of data from a block of physical memory could track the sharing of the status of the block.

Snooping can also be used as the coherence protocol for a multichip multiprocessor and some design support a snooping protocol on top of a directory protocol with each multicore.

Cache Coherence Protocols:

These are explained as following below:

1. MSI Protocol:

This is a basic cache coherence protocol used in multiprocessor system. The letters of protocol name identify possible states in which a cache can be. So, for MSI each block can have one of the following possible states:

- **Modified -**
  The block has been modified in cache, i.e., the data in the cache is inconsistent with the backing store (memory). So, a cache with a block in "M" state has responsibility to write the block to backing store when it is evicted.

- **Shared -**
  This block is not modified and is present in at least one cache. The cache can evict the data without writing it to backing store.
- **Invalid -**
  This block is invalid and must be fetched from memory or from another cache if it is to be stored in this cache.

2. MOSI Protocol:

This protocol is an extension of MSI protocol. It adds the following state in MSI protocol:

- **Owned -**
  It indicates that the present processor owns this block and will service requests from other processors for the block.

3. MESI Protocol -

It is the most widely used cache coherence protocol. Every cache line is marked with one the following states:

- **Modified -**
  This indicates that the cache line is present in current cache only and is dirty i.e its value is different from the main memory. The cache is required to write the data back to main memory in future, before permitting any other read of invalid main memory state.
- **Exclusive -**
  This indicates that the cache line is present in current cache only and is clean i.e its value matches the main memory value.
- **Shared -**
  It indicates that this cache line may be stored in other caches of the machine.
- **Invalid -**
  It indicates that this cache line is invalid.

4. MOESI Protocol:

This is a full cache coherence protocol that encompasses all of the possible states commonly used in other protocols. Each cache line is in one of the following states:

- **Modified -**
  A cache line in this state holds the most recent, correct copy of the data while the copy in the main memory is incorrect and no other processor holds a copy.
- **Owned -**
  A cache line in this state holds the most recent, correct copy of the data. It is similar to shared state in that other processors can hold a copy of most recent, correct data and unlike shared state however, copy in main memory can be incorrect. Only one processor can hold the data in owned state while all other processors must hold the data in shared state.
- **Exclusive -**
  A cache line in this state holds the most recent, correct copy of the data. The main memory copy is also most recent, correct copy of data while no other holds a copy of data.
- **Shared -**
  A cache line in this state holds the most recent, correct copy of the data. Other processors in system may hold copies of data in shared state as well. The main memory copy is also the most recent, correct copy of the data, if no other processor holds it in owned state.
- **Invalid -**
  A cache line in this state does not hold a valid copy of data. Valid copies of data can be either in main memory or another processor cache.