MESI protocol for cache coherence and more

Mian Qin · August 14, 2020

Architecture

Recently I was interviewed with a design question regarding coherence. The problem statement is like this. We have a server that hold the ground truth of the data (like a database or key-value store). We also have many clients that may read the data from the sever and make local updates to the data. What kind of design guidence should we consider for such system. This is acutally a similiar problem as cache coherence for multi-core system. In this blog I will reveiw the classic MESI protocol for cache coherence.

Overview

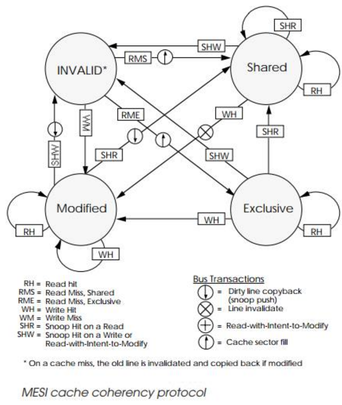

Figure 1 MESI state diagram[[1]]

The general approach to implement cache coherence is the SNOOPY based methods. The idea is to have a common bus connecting the private caches and the shared next level cache or main memory. The basic protocol is a valid/invalid protocol that only implement two states. Whenever a core miss on a block, it will invalidate all the other cores that hold the block and mark it valid. Building upon the valid/invalid protocol is the MSI protocal which differ clean and dirty state of a cache. Transfer to **M**odified state will invalidate the block from other cores. **S**hared state will also snoop on the bus but may not invalidate block if there is no M block on the other cores. However, there is a problem for the MSI protocol when a core write to the shared state block (from shared to modified). It always needs to broadcast message to other cores even it's the only core hold the block. To avoid this problem, MESI protocol is proposed as shown in Figure 1. The following section will elabrate the state transfer of the MESI protocol.

MESI State Transfer

Whether the state change from invalid to exclusive or shared depends on whether other cores hold the same block (a simple **OR** logic on the hit signal).

1. **I**nvalid to **S**hared.
   Read miss and issue read request. If there are other cores hold the block and the state is shared, transfer to shared state.
2. **I**nvalid to **E**xclusive.
   Read miss and issue read request. If there are no other cores hold the block, transfer to exclusive state.
3. **I**nvalid to **M**odified.
   Write miss and issue read request. Broadcast message to invalidate exclusive and shared state blocks. May incur dirty write back for other modified block on other core.
4. **S**hared to **M**odified.
   Write hit. Broadcast message to invalidate shared state blocks.
5. **S**hared to **I**nvalid.
   Snoop hit on a write.
6. **E**xclusive to **M**odified.
   Write hit. **No need to broadcast any messages.**
7. **E**xclusive to **I**nvalid.
   Snoop hit on a write.
8. **M**odified to **S**hared.
   Snoop hit on a read. Dirty write back and other core read the updated copy and transfer from invalid to shared.
9. **M**odified to **I**nvalid.
   Snoop hit on a write. Dirty write back and other core read the update copy and transfer from invalid to modified.

Snoopy vs Directory

As discussed above, snoop with a common bus connected with all the cores (not many) and the LLC is a typical method to implement coherence protocol. For example, lots of ARM architectures use this snoopy based coherence implementation. However, if the scale of core numbers pass a certain threshold, we may not be able to build a efficient common bus. Thus, directory based method is proposed with a better scalability. The basic idea of directory based method is a client sever model. The server (directory) hold some ground truth of the coherence metadata (such as which cores are hold the block, what are the states of those block). The client (cores) needs to consult the server first on a read/write miss and the server may coordinate efficient actions for the clients such as invalidate block from other cores with shared state (get rid of broadcasting), client to client forwarding, etc. You can design more sophisticated and efficient actions based on the interconnect. However, the caveat is the management of directory (especially on hardware).

Add Comment(cancel reply)

Name

E-mail