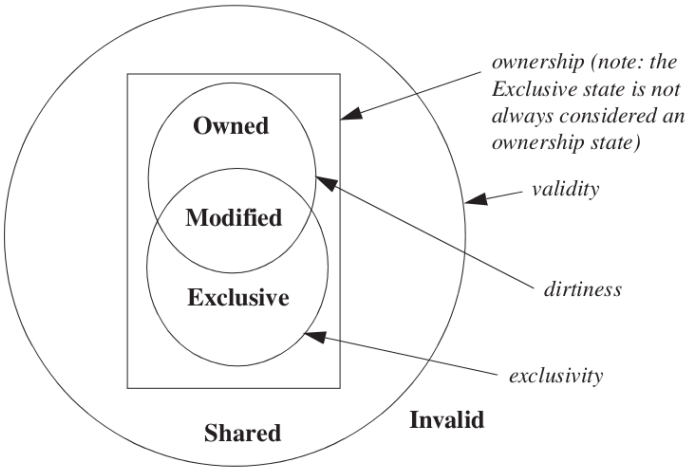
A Primer on Memory Consistency and Cache Coherence (Chapter 6-7)

by Yunqi Zhang
04/19/2013

1. Coherence Protocol
    a. each storage structure has a finite state machine called a ***coherence controller***
    b. one independent, identical finite state machine per block
    c. cache controller
        i. request from core side: loads and stores from the core, returns values back
            1. cache miss: coherence transaction start with a coherence request through the interconnection network
        ii. interface to network side: connects to the rest of the system
    d. memory controller (LLC, Memory)
        i. only network side interface: doesn't issue or receive coherence requests
    e. can be clearly described by tabular
2. Design Space
    a. state - things we want to encode
        i. validity: most up-to-date value for the block
        ii. dirtiness: most up-to-date value but differs from LLC/MEM
        iii. exclusivity: only privately cached copy in the system, can be cached in LLC
        iv. ownership: responsible for responding to coherence requests
    b. state - one commonly used **stable** states
        i. M(odified): valid, exclusive, owned, and potentially dirty - RW
        ii. S(hared): valid, ! exclusive, ! owned, ! dirty - R
        iii. I(nvalid): ! valid - NULL
        iv. O(wned): valid, ! exclusive, owned, and potentially dirty - R
        v. E(xclusive): valid, exclusive, ! dirty - R



c. state - **transient** states between the transition of two stable states
d. state - blocks in LLC/MEM
    i. cache-centric (common): aggregation of the states of that block in the caches
    ii. memory-centric: corresponds to the memory controller's permissions to that block
e. state - maintain the block state
    i. cache: few bits for each block
    ii. memory:
        1. inclusive LLC maintains a copy of every block that is cached somewhere - its state in memory is the same as in LLC
f. Transactions
    i. GetShared: obtain block in Shared (read-only) state
    ii. GetModified: obtain block in Modified (read-only) state
    iii. Upgrade: upgrade block state from read-only to read-write
    iv. PutShared: evict block in Shared state
    v. PutExclusive: evict block in Exclusive state
    vi. PutOwned: evict block in Owned state
    vii. PutModified: evict block in Modified state

3. Major Protocol Design Options
    a. Snooping VS Directory
        i. snooping: broadcast a request message to all other coherence controllers
            1. (+) simple
            2. (-) do not scale to large numbers of cores
        ii. directory: unicast a request message to the controller that is the home of that block
            1. (+) scalability
            2. (-) more time or message for many transactions
    b. Invalidate VS Update
        i. invalidate: invalidate the copies in other caches when a core wish to write
        ii. update: update the copies in other caches when a core wish to write
            1. (+) reduce the latency for a core to read a newly written block
            2. (-) consume more bandwidth
            3. (-) complicate the implementation of memory consistency model