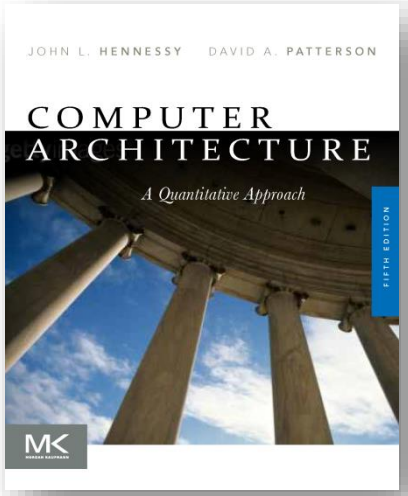


Chapter 5

Multiprocessors and Thread-Level Parallelism

Spring 2019

Soontae Kim
School of Computing, KAIST

Announcements

- Final project presentations
  - May 30
    - Teams with three members
  - June 4
    - Teams with two members and one member
  - Presentation times allowed
    - One-member teams: 6 min. Two-member teams: 8 min. Three-member teams: 10 min.
  - Contents
    - Motivation, approach, implementation, experimental methodology, experimental results, lessons & conclusion
  - Evaluation criteria
    - Time, correct implementation, results, presentation quality, details

On May 30

7. 최재영**,** 이상현**,** 안재국
8. 이준영**,** 송재윤**,** 김성엽
9. Eric, Luc, Corentin
10.Jibon, 최갑도**,** 조경재
11.김태준**,** 손선민**,** 김대화
12.권민수**,** 정인교**,** 정원준
13.이민호**,** 정용빈**,** 허철훈
10*7 = 70 min.

On June 4

1. 한윤기**,** 윤한천
2. Tanvir, Emmanuel
3. 임주원**,** 김하경
4. 박지희**,** 강다현
5. 신승철**,** 박재준
6. 이신영**,** 장현중
14.홍석빈
15.허재석
16.윤희수
17.손민석
18.배우근
48+30=78 min.

Announcements

- Final reports are due on June 17
  - Single column, 11 font size, 5~8 pages
  - Follow typical conference format
    - Introduction, related work, approach(proposed idea), experimental environment and results, conclusion
  - If you added more contents than final presentation, indicate them in your report
  - Also show the roles of all team members
- Final exam
  - From 1:00pm to 3:00pm on June 11
  - Cover all class materials after midterm exam

Snoopy Coherence Protocols

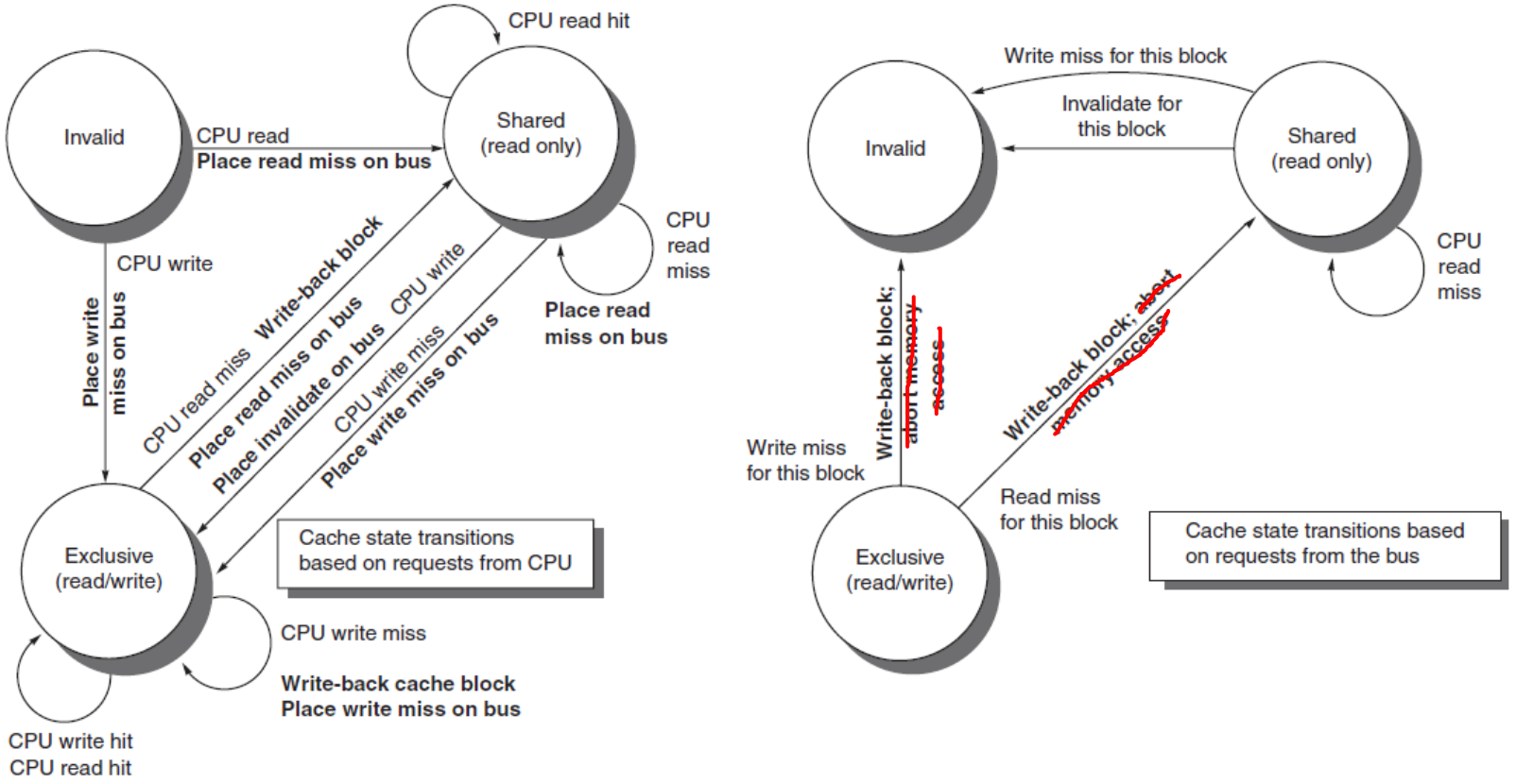


Figure 5.6 **A write invalidate, cache coherence protocol for a private write-back cache showing the states and state transitions for each block in the cache.** The cache states are shown in circles, with any access permitted by the block in the shared state generates an invalidate. Whenever a bus transaction occurs, all private caches that contain the cache block specified in the bus transaction take the action dictated by the right half of the diagram. The protocol assumes that memory (or a shared cache) provides data on a read miss for a block that is clean in all local caches. In actual implementations, these two sets of state diagrams are combined. In practice, there are many subtle variations on invalidate protocols, including the introduction of the exclusive unmodified state, as to whether a processor or memory provides data on a miss. In a multicore chip, the shared cache (usually L3, but sometimes L2) acts as the equivalent of memory, and the bus is the bus between the private caches of each core and the shared cache, which in turn interfaces to the memory.

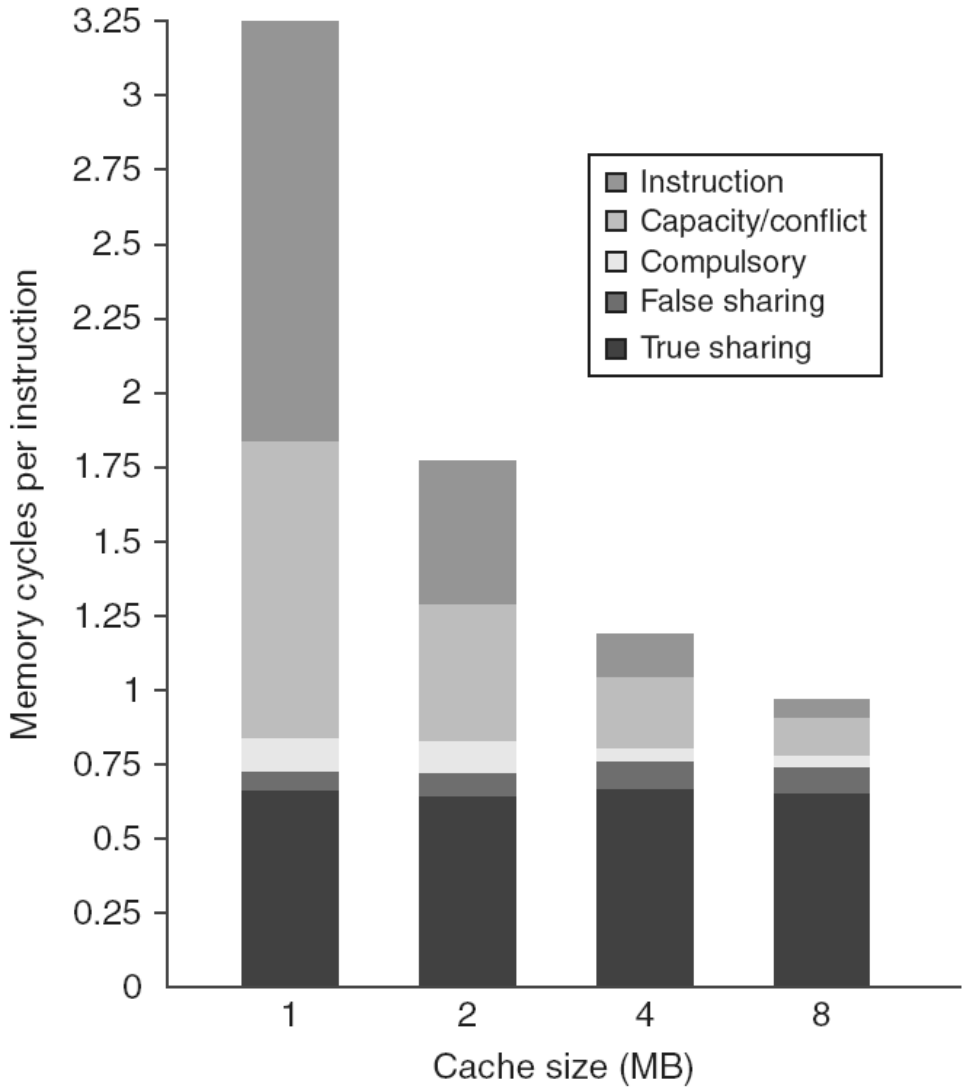
Snoopy Coherence Protocols

- Complications for the basic MSI protocol:
  - Operations are not atomic
    - E.g. detect miss, acquire bus, receive a response
    - Creates possibility of deadlock and races
    - One solution: processor that sends invalidate can hold bus until other processors receive the invalidate
- Extensions:
  - Add exclusive state to indicate clean block in only one cache (MESI protocol)
    - Prevents needing to write invalidate on a write
  - Owned state (MOESI)
    - Associated block is owned by that cache and out-of-date in memory
    - The cache is responsible for replacement and providing data on a miss

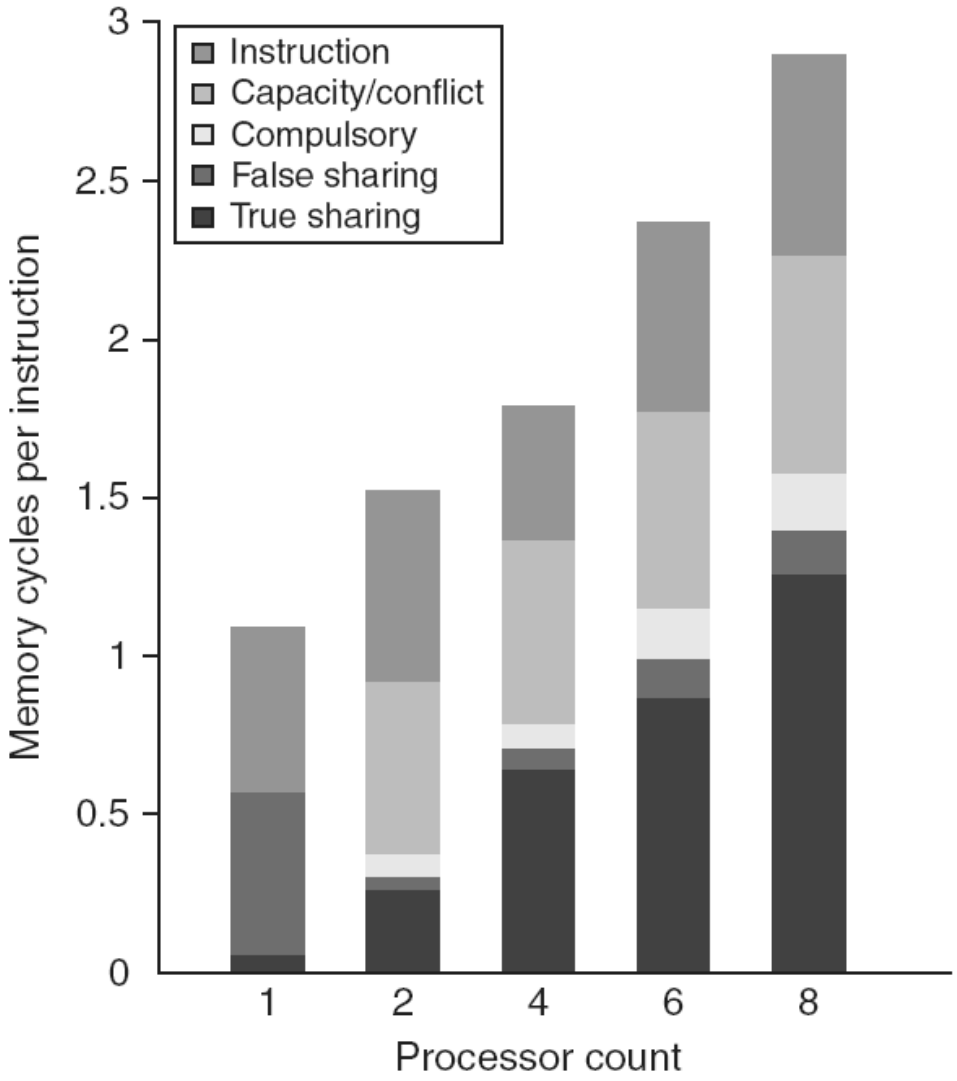
Performance

- Coherence influences cache miss rate
  - Coherence misses
    - True sharing misses
      - Write a word to shared block (transmission of invalidation) and read the modified word in the invalidated block
    - False sharing misses
      - Read an unmodified word in the invalidated block
      - Can reduce by using one-word blocks

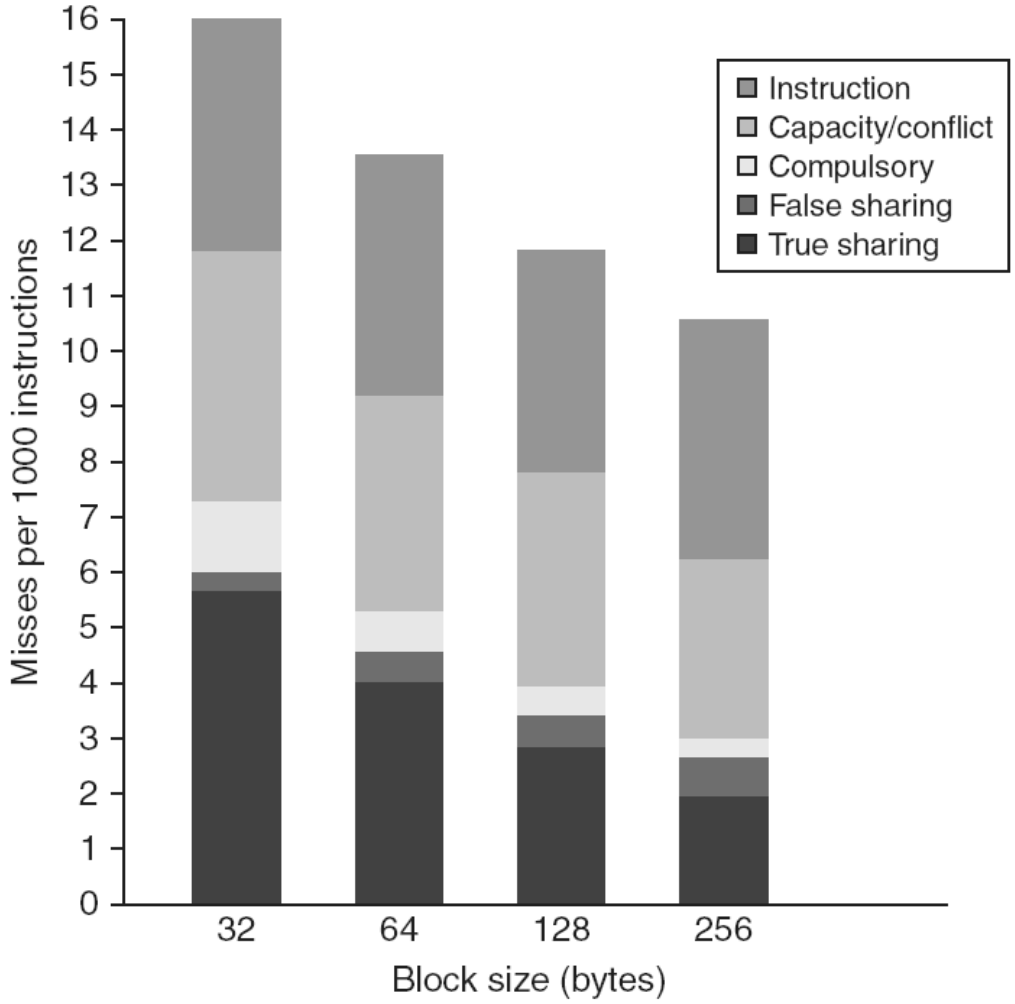
Performance Study: Commercial Workload



Performance Study: Commercial Workload

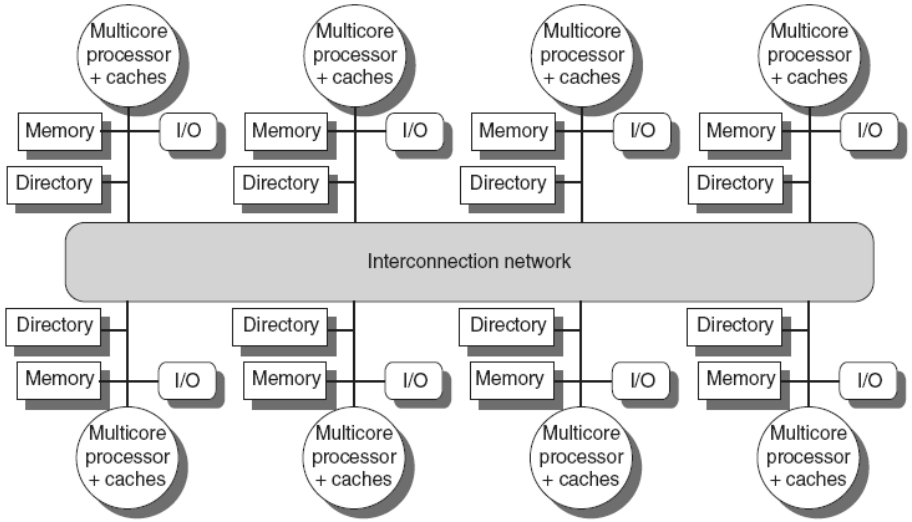


Performance Study: Commercial Workload



Directory Protocols

- Directory keeps track of every block
  - Which caches have each block
  - Dirty status of each block
- Implemented in shared L3 cache
  - Keep bit vector of size = # cores for each block in L3
  - Not scalable beyond shared L3
- Implement in a distributed fashion:



Directory Protocols II

- The directory must be distributed, but the distribution must be done in a way that the coherence protocol knows where to find the directory information for any cached block of memory
  - Distribute the directory along with the memory
  - Sharing status of a block is always in a single known location to avoid broadcast
  - The simplest directory implementations associate an entry in the directory with each memory block
    - Amount of information is proportional to the product of the number of memory blocks times the number of nodes

Directory Protocols III

- For each block, maintain state:
  - Shared
    - One or more nodes have the block cached, value in memory is up-to-date
    - Set of node IDs
  - Uncached
  - Modified
    - Exactly one node has a copy of the cache block, value in memory is out-of-date
    - Owner node ID
- Directory maintains block states and sends invalidation messages

Terms

- The *local node* is the node where a request originates
- *Home node* is the node where the memory location and the directory entry of an address reside
  - physical address space is statically distributed, so the node that contains the memory and directory for a given physical address is known.
- Remote node is the node that has a copy of a cache block
- Sharer of memory block

Messages

Message type	Source	Destination	Message contents	Function of this message
Read miss	Local cache	Home directory	P, A	Node P has a read miss at address A; request data and make P a read sharer.
Write miss	Local cache	Home directory	P, A	Node P has a write miss at address A; request data and make P the exclusive owner.
Invalidate	Local cache	Home directory	A	Request to send invalidates to all remote caches that are caching the block at address A.
Invalidate	Home directory	Remote cache	A	Invalidate a shared copy of data at address A.
Fetch	Home directory	Remote cache	A	Fetch the block at address A and send it to its home directory; change the state of A in the remote cache to shared.
Fetch/invalidate	Home directory	Remote cache	A	Fetch the block at address A and send it to its home directory; invalidate the block in the cache.
Data value reply	Home directory	Local cache	D	Return a data value from the home memory.
Data write-back	Remote cache	Home directory	A, D	Write-back a data value for address A.

Figure 5.21 The possible messages sent among nodes to maintain coherence, along with the source and destination node, the contents (where P = requesting node number, A = requested address, and D = data contents), and the function of the message. The first three messages are requests sent by the local node to the home. The fourth through sixth messages are messages sent to a remote node by the home when the home needs the data to satisfy a read or write miss request. Data value replies are used to send a value from the home node back to the requesting node. Data value write-backs occur for two reasons: when a block is replaced in a cache and must be written back to its home memory, and also in reply to fetch or fetch/invalidate messages from the home. Writing back the data value whenever the block becomes shared simplifies the number of states in the protocol, since any dirty block must be exclusive and any shared block is always available in the home memory.

Directory

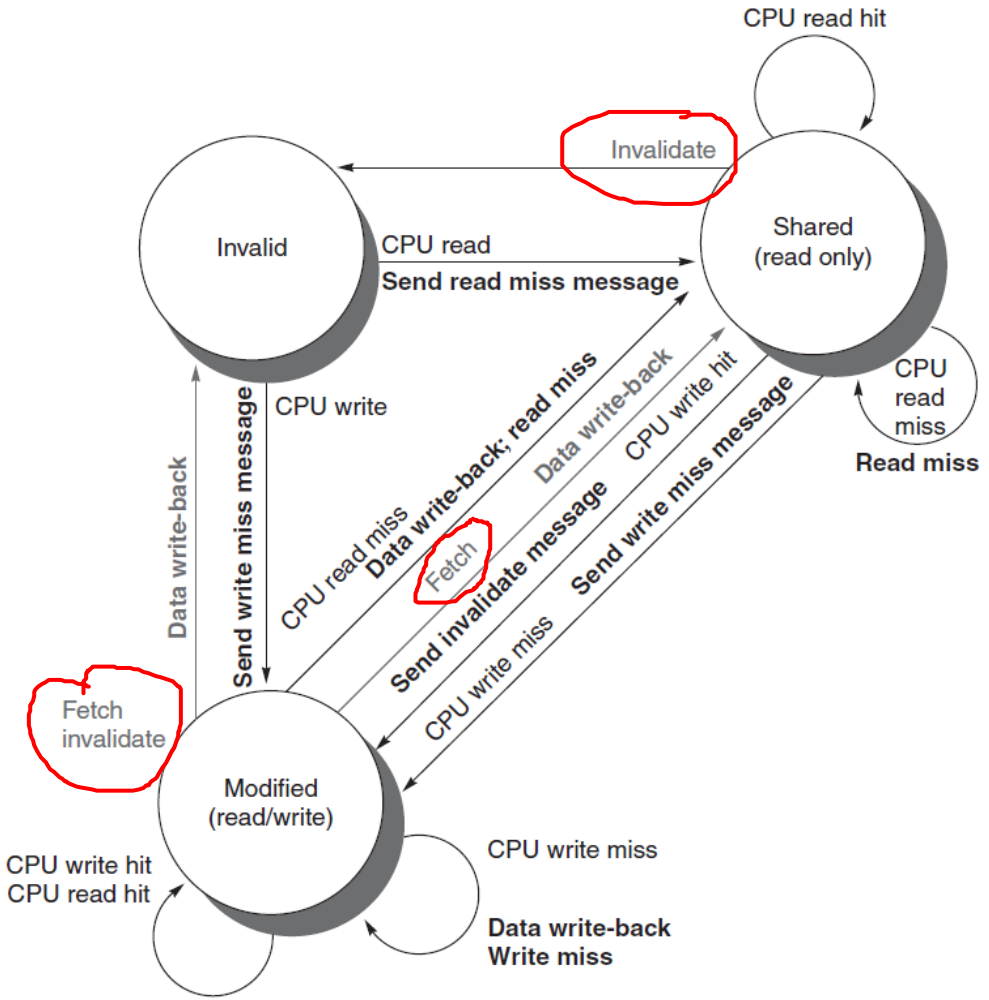


Figure 5.22 State transition diagram for an individual cache block in a directory-based system. Requests by the local processor are shown in black, and those from the home directory are shown in gray. The states are identical to those in the snooping case, and the transactions are very similar, with explicit invalidate and write-back requests replacing the write misses that were formerly broadcast on the bus. As we did for the snooping controller, we assume that an attempt to write a shared cache block is treated as a miss; in practice, such a transaction can be treated as an ownership request or upgrade request and can deliver ownership without requiring that the cache block be fetched.

Directory Protocols II

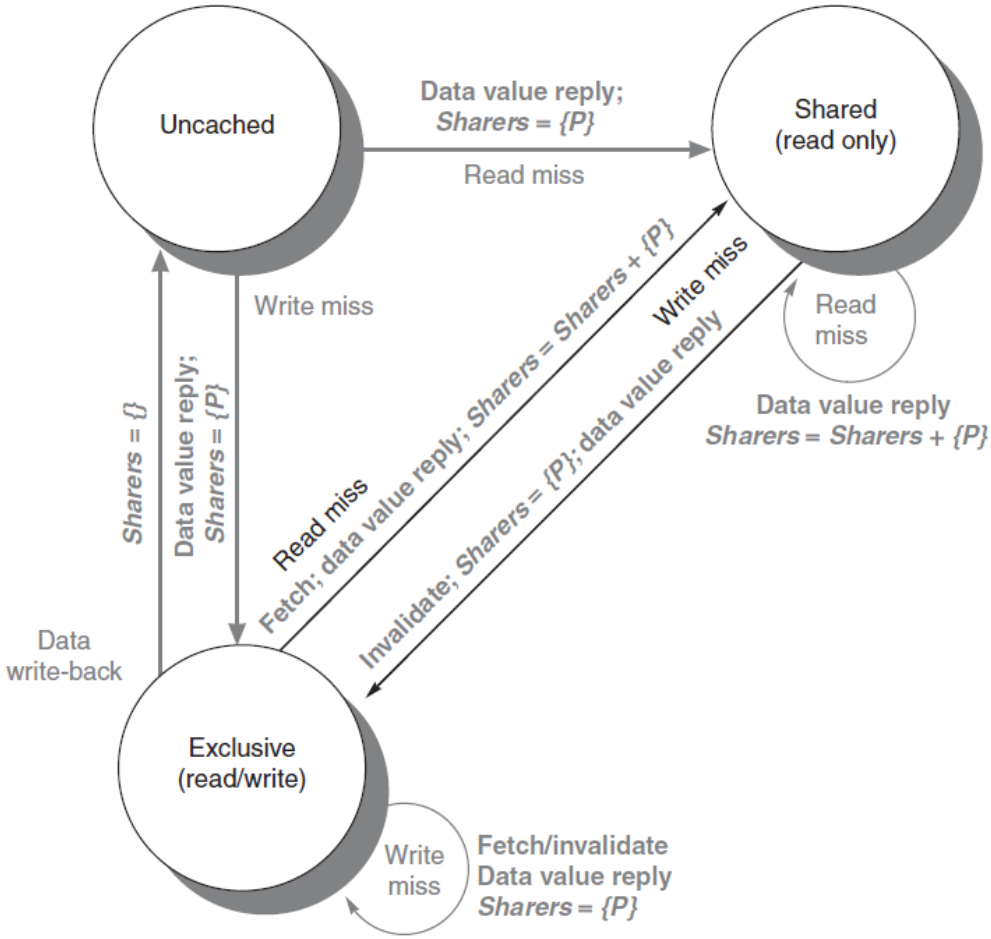


Figure 5.23 The state transition diagram for the directory has the same states and structure as the transition diagram for an individual cache. All actions are in gray