Scalability Challenge of Snooping Protocols

Snooping protocols require communication with all caches on every cache miss, including writes of potentially shared data. While simple and inexpensive, this becomes their Achilles' heel for scalability.

Bandwidth Requirements

A system of four 4-core multicores at 4 GHz could require between 4 GB/sec to 170 GB/sec of bus bandwidth - far beyond the capability of any bus-based system.

Multicore Impact

The development of multicore processors has forced designers to shift to distributed memory systems to support the bandwidth demands of individual processors.

Even with distributed memory (which separates local from remote traffic), we gain little unless we eliminate the need for the coherence protocol to broadcast on every cache miss.

Directory-Based Coherence

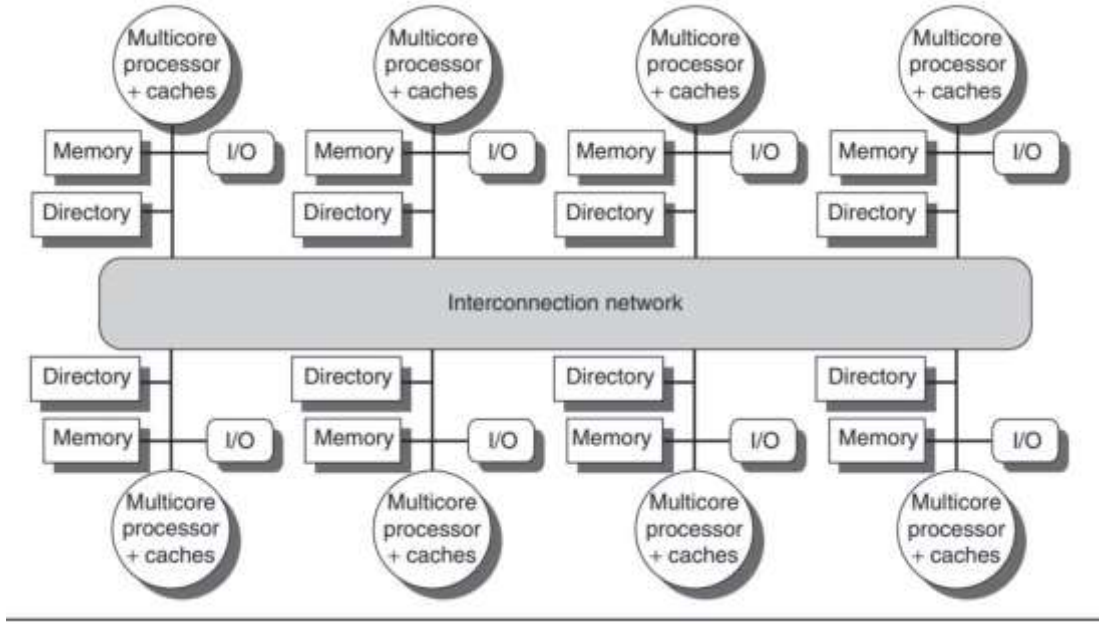


Figure: A directory is added to each node to implement cache coherence in a distributed-memory multiprocessor.

A directory keeps the state of every block that may be cached, including which caches have copies and whether blocks are dirty. This eliminates the need for broadcast messages.

Within a multicore with shared outermost cache (L3), implementation is simple: keep a bit vector equal to the number of cores for each L3 block, indicating which private caches may have copies. Invalidations are only sent to those caches.

For scalability across multiple chips, the directory must be distributed along with memory, allowing different coherence requests to go to different directories.

Directory Structure and Tracking

Directory Entry States

- Shared - One or more nodes have the block cached, memory is up to date
- Uncached - No node has a copy of the cache block
- Modified - Exactly one node has a copy and has written to it; memory is out of date

Tracking Mechanism

The simplest approach uses a bit vector for each memory block. When shared, each bit indicates whether the corresponding processor has a copy. When exclusive, it identifies the owner.

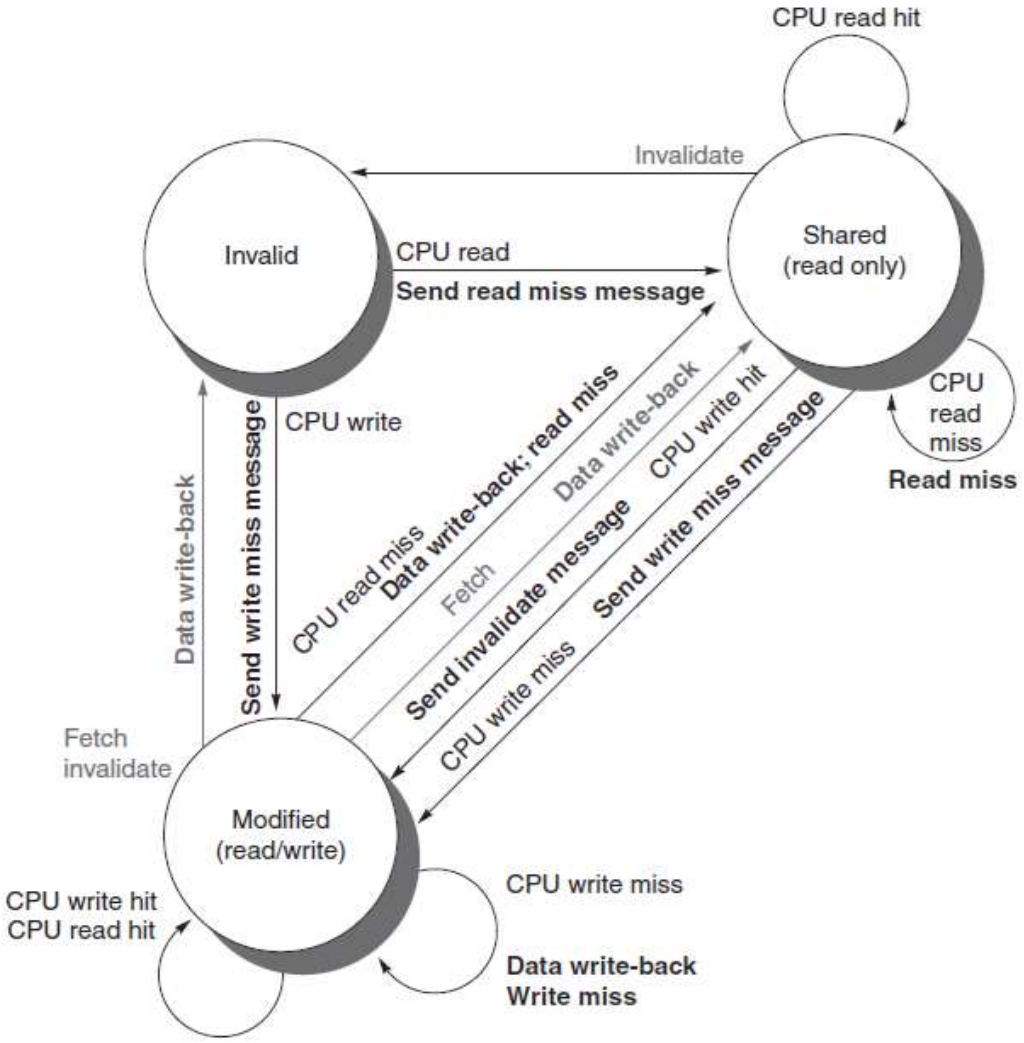
The directory size is proportional to the product of memory blocks times the number of nodes.

This overhead is tolerable for multiprocessors with less than a few hundred processors. Larger systems require more efficient scaling methods.

Message Types in Directory Protocols

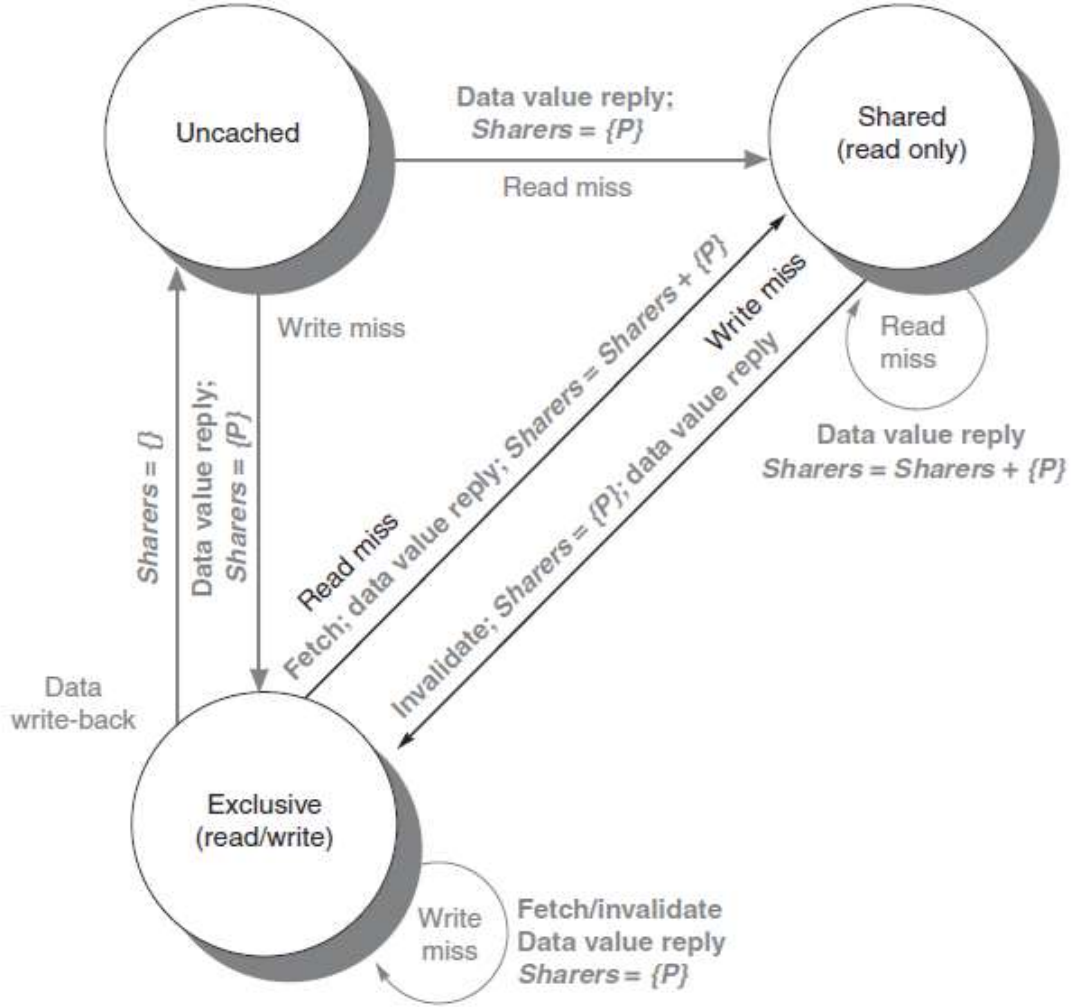
Message type	Source	Destination	Contents	Function
Read miss	Local cache	Home directory	P, A	Node P has a read miss at address A; request data and make P a read sharer
Write miss	Local cache	Home directory	P, A	Node P has a write miss at address A; request data and make P the exclusive owner
Invalidate	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

Cache State Transitions



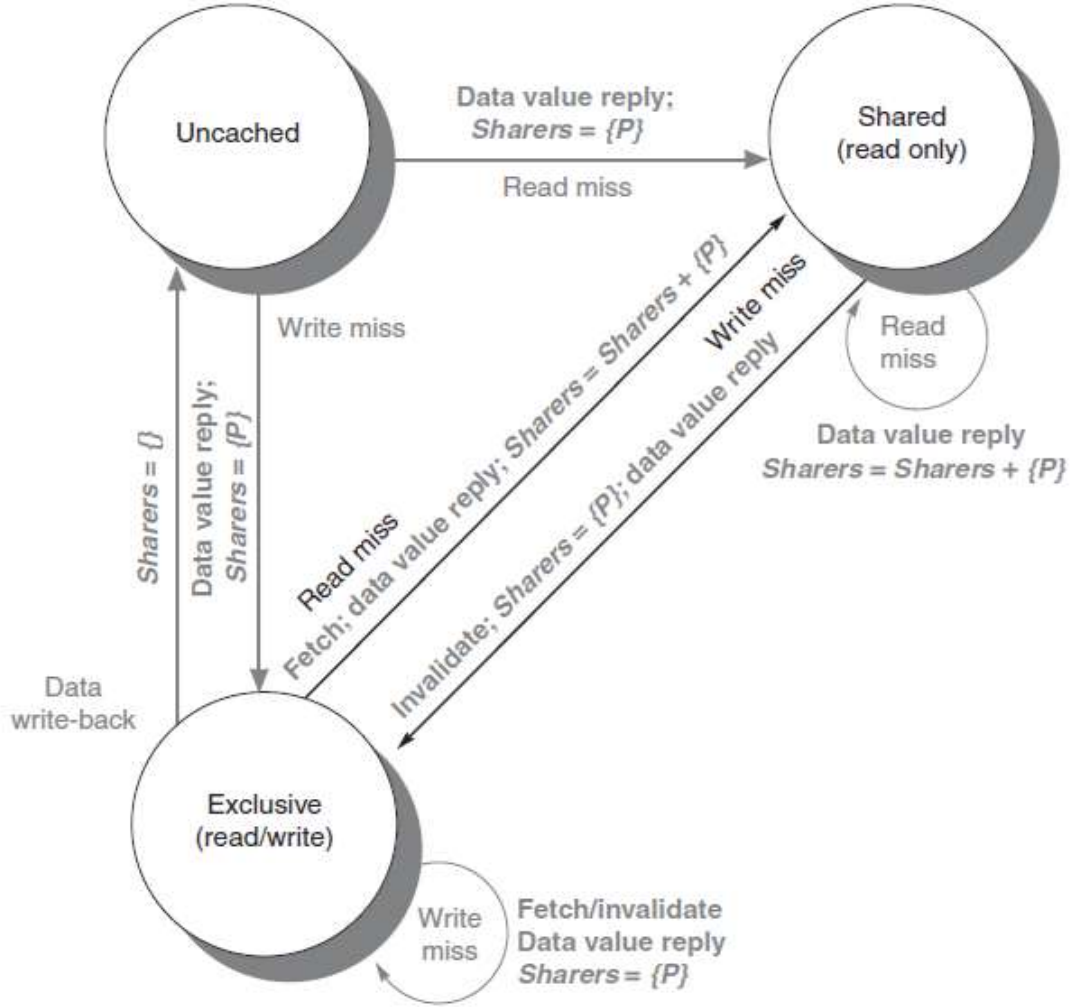
- The states for individual cache blocks are identical to those in snooping protocols: Invalid, Shared, and Exclusive/Modified. However, the transactions differ:
  - Local processor requests are shown in black
  - Directory requests are shown in gray
  - Broadcast write misses are replaced by targeted invalidate and fetch operations
- As with snooping, any cache block must be in exclusive state when written, and any shared block must be up to date in memory.

Directory State Transitions



- In a directory-based protocol, the directory implements the other half of the coherence protocol
- The directory receives three different requests: read miss, write miss, and data write-back

Directory Protocol Operations

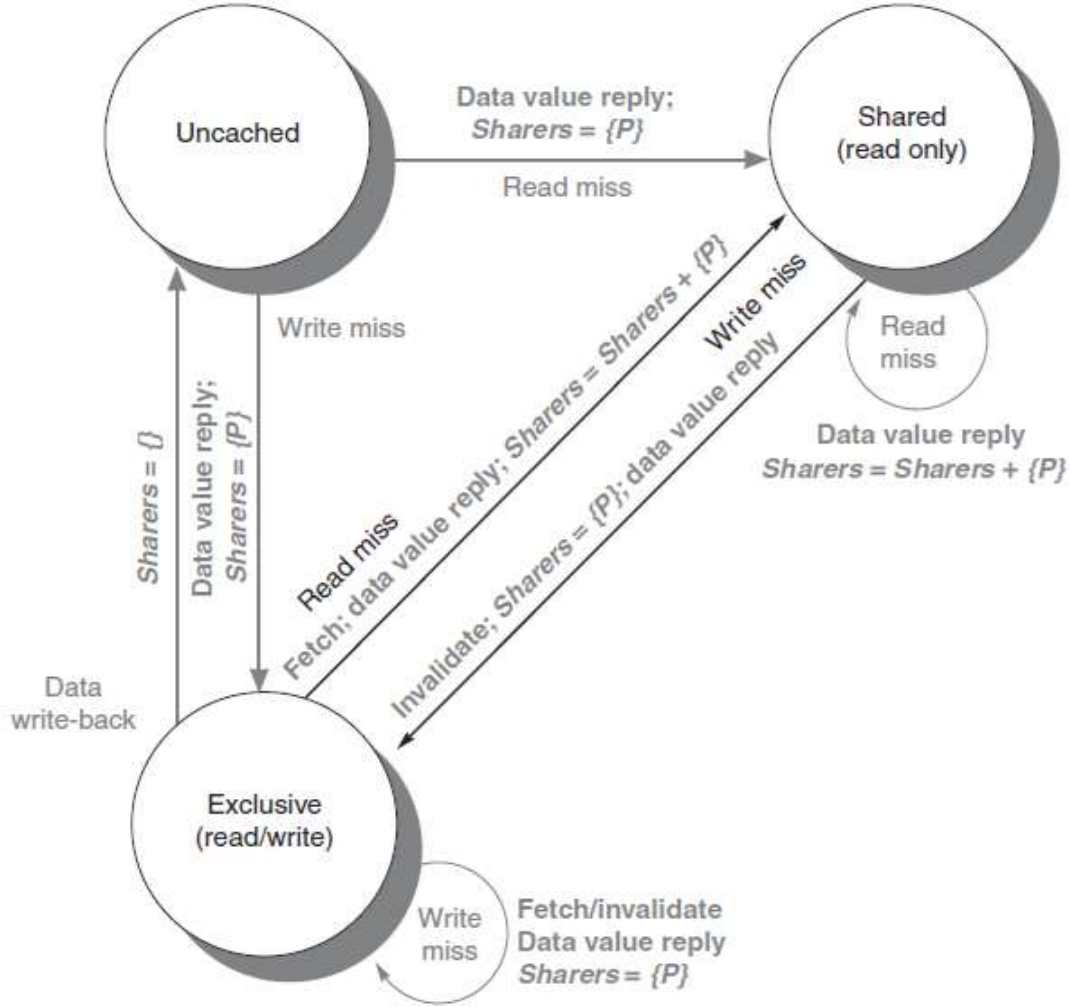


Uncached State

Read miss: Send data to requestor, add to sharers, change to Shared

Write miss: Send data to requestor, make requestor the owner, change to Exclusive

Directory Protocol Operations

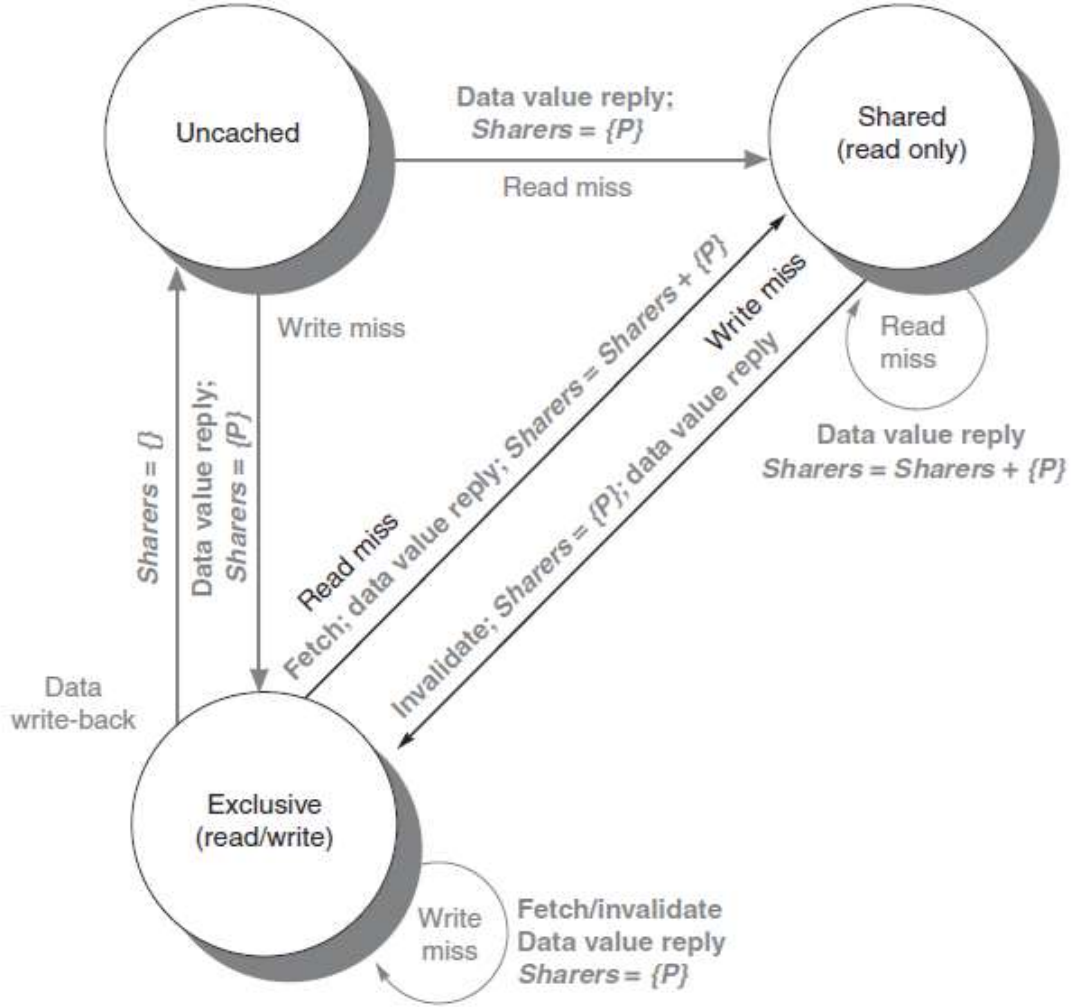


Shared State

Read miss: Send data to requestor, add to sharers

Write miss: Send data to requestor, invalidate all sharers, make requestor the owner, change to Exclusive

Directory Protocol Operations



Exclusive State

Read miss: Fetch data from owner, send to requestor, add both to sharers, change to Shared

Write miss: Fetch/invalidate from owner, send to requestor, make requestor the owner

Data write-back: Update memory, change to Uncached, clear sharers

Protocol Optimizations and Challenges

- Common optimizations
  - Direct forwarding from owner to requestor (bypassing home directory)
  - Reducing latency by avoiding memory updates on clean replacements
  - Specialized handling of upgrade requests (shared to exclusive)
- Implementation challenges
  - Handling non-atomic memory transactions
  - Avoiding deadlock conditions
  - Managing multiple message types
  - Ensuring forward progress