CSE548

Cache Coherence Protocols

Rahul Banerjee
13th Feb 2013

What is Cache Coherence?

- Multiple cores w. caches, each may buffer same data
- Writes to block in one cache may need to be reflected in other caches, else caches become architecturally visible
- Example: P1 and P2 read and modify the same block, then P3 reads from that block. Which value will it get?

  (In the absence of caches, SC answer is "last writer")
- Solution:
  - Track state of every block residing in every cache
  - Enforce SWMR and Data Value invariants by changing state, and shuffling data around as necessary
- Two main challenges:
  - Provide total ordering of state change requests/responses
  - Maximize bandwidth among cache controllers for messages and data
- Two solutions:
  - Snooping protocols
  - Ditrectory-based protocols

Cache Block States

- For a given block, we care about these attributes: Validity, Dirtiness, Exclusivity, and Ownership
- Commonly-Used states are:
  - **M**odified: Valid, exclusive, owned and potentially dirty. The only copy
  - **S**hared: Valid but not exclusive, not dirty, not owned. Other copies
  - **I**nvalid: Either not present, or present and stale, not to be read/written
- Other states are:
  - **O**wned: Valid, owned and potentially dirty, but not exclusive. RO copy
  - **E**xclusive: Valid, exclusive, and clean. Only copy. Usually owned.
- Coherence via messages/data:
  - GetM: Precedes a write to a block, to transition from MR to SW
  - GetS: Precedes a read from a block that was previously Invalid.

Cache Coherence via Snooping

- Totally-ordered broadcast (all contr recv all messages)
- Only the owner of a block responds to the message
- Two messages complete a transaction

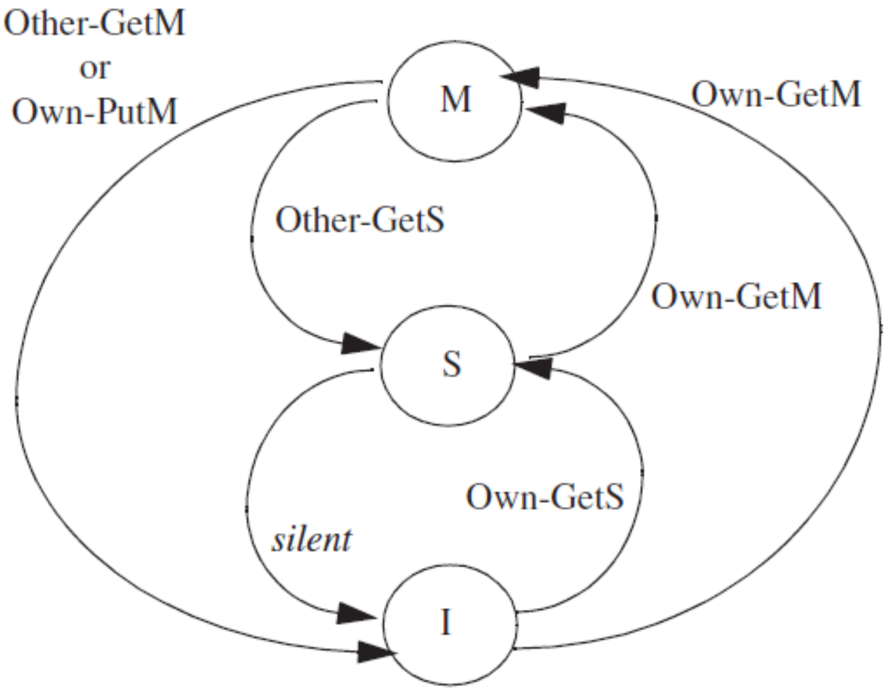

FIGURE 7.1: MSI: Transitions between stable states at cache controller.

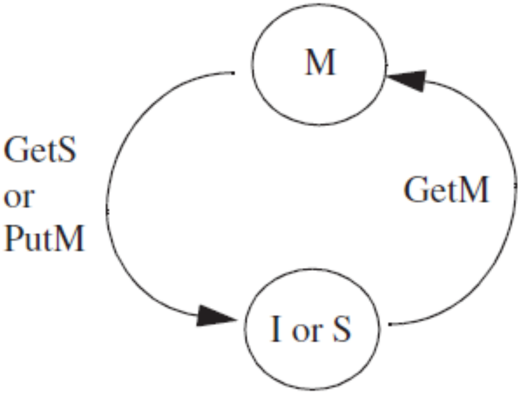

FIGURE 7.2: MSI: Transitions between stable states at memory controller.

Cache Coherence via Directory Protocols

- Point-to-point ordered network
- Directory owns block (unless it is in **M** state)
- Messages of three kinds:
  - Request: GetS, GetM and PutM
  - Forwarded: Fwd-GetS, Fwd-GetM, Inv(alidate), and Put-Ack
  - Response: Data and Inv-Ack
- Three messages to complete a transaction

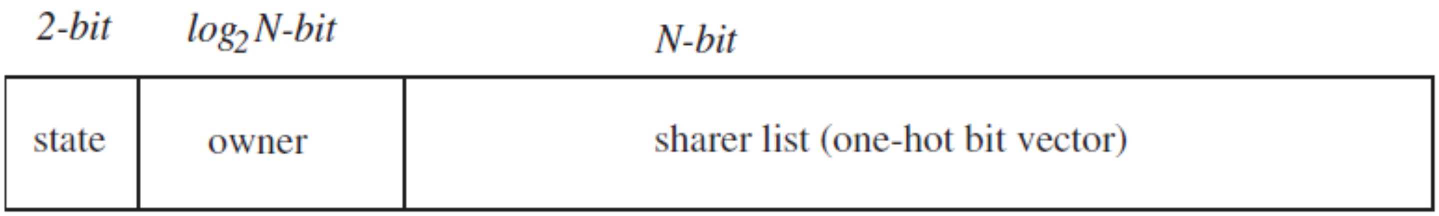

FIGURE 8.2: Directory entry for a block in a system with N nodes.

Cache Coherence via Token Coherence

- Have a "Correctness Substrate" that enforces coherence by ensuring safety (all reads/writes coherent) and starvation avoidance (via persistent requests)
- Build a "Performance protocol" on top, that sends "transient requests" as hints to speed up the common case

Questions

- How are these protocols verified?
- Does this practically scale to 10s of 1000s of procs?