Cache Coherence

Ryan Lee
(Adapted from prior slides by Suvinay Subramanian)
6.823 Spring 2019
Cache Coherence Problem

» Parallel programs require processes/threads to share data
  - Shared memory
  - Message passing

» Problem: If multiple processors cache the same block, how do they ensure “correct” view of the data?
Without private caches

0: addi r1,r3,1
1: ld r4,0(r3)
2: blt r4,r2,6
3: sub r4,r2,r4
4: st r4,0(r3)
5: call _foo

Memory

500
400
300
With writeback caches

0: addi r1,r3,1
1: ld r4,0(r3)
2: blt r4,r2,6
3: sub r4,r2,r4
4: st r4,0(r3)
5: call _foo

Caches

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>400</td>
<td>500</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>500</td>
</tr>
<tr>
<td>4</td>
<td>400</td>
<td>400</td>
</tr>
</tbody>
</table>

Memory

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
</tr>
</tbody>
</table>

5/3/19
Cache Coherence Strategy

» Two Rules:
  1. Write propagation: Writes eventually become visible to other processors
  2. Write serialization: Writes to same location are serialized

» Invalidation-based: A write to a cache line invalidates all other copies

» Update-based: A write to a cache line updates all other copies
Cache Coherence Strategy

» **Snoopy** coherence protocol
  All caches observe other caches’ actions through a shared bus(-like) interconnect.

» **Directory-based** coherence protocol
  A coherence directory tracks contents of caches and sends (and receives) messages to maintain coherence.

Tradeoffs?
MSI Protocol

Three states per cache-line
- Invalid (I): Cache does not have a copy
- Shared (S): Cache has read-only copy; clean
- Modified (M): Cache has only copy; writable; (potentially) dirty

Processor Actions: Read (PrRd), Write (PrWr)
MSI Protocol

» Directory maintains its own coherence state, and sharer set

» States:
  - Uncached (Un): No cache has a valid copy
  - Shared (S): One or more caches in S state
  - Exclusive (Ex): One of the caches in M state

» Sharer Set:
  - Tracks which caches hold the line
Lab Task: MSI Coherence Protocol

» Implement with Murphi description language
  - Explores all reachable states
  - Uses symmetry to canonicalize redundant states
  - Rules: Transitions between states
  - Invariants and asserts: Capture protocol correctness
Principles

» Think of sending and receiving messages as separate events.

» At each step, think what new requests can occur
  - Messages overtaking other messages

» Two messages in the same direction implies a race
  - Consider both deliver orders
  - Often, only one node knows how to resolve a race
    (might send other nodes msgs suitably)