Shared Memory SMP and Cache Coherence (cont)
Review: Snoopy Cache Protocol

- **Write Invalidate Protocol:**
  - Multiple readers, single writer
  - Write to shared data: an invalidate is sent to all caches which snoop and invalidate any copies
  - Read Miss:
    - Write-through: memory is always up-to-date
    - Write-back: snoop in caches to find most recent copy

- **Write Broadcast Protocol (typically write through):**
  - **Write serialization:** bus serializes requests!
    - Bus is single point of arbitration

- **Good for a small number of processors; how about 16 or more?**
Larger MPs

- Separate Memory per Processor
- Local or Remote access via memory controller
- 1 Cache Coherency solution: non-cached pages
- Alternative: directory per cache that tracks state of every block in every cache
  - Which caches have a copies of block, dirty vs. clean, ...
- Info per memory block vs. per cache block?
  - PLUS: In memory => simpler protocol (centralized/one location)
  - MINUS: In memory => directory is $\mathcal{O}(\text{memory size})$ vs. $\mathcal{O}(\text{cache size})$
- Prevent directory as bottleneck?
  - distribute directory entries with memory, each keeping track of which Procs have copies of their blocks
Distributed Directory MPs
Directory Protocol

Similar to Snoopy Protocol: Three states
- **Shared**: ≥ 1 processors have data, memory up-to-date
- **Uncached**: (no processor has it; not valid in any cache)
- **Exclusive**: 1 processor (owner) has data; memory out-of-date

In addition to cache state, must track **which processors** have data when in the shared state (usually bit vector, 1 if processor has copy)

Keep it simple(r):
- Writes to non-exclusive data
  => write miss
- Processor blocks until access completes
- Assume messages received and acted upon in order sent
Directory Protocol

No bus and don’t want to broadcast:
- interconnect no longer single arbitration point
- all messages have explicit responses

Terms: typically 3 processors involved
- Local node where a request originates
- Home node where the memory location of an address resides
- Remote node has a copy of a cache block, whether exclusive or shared

Example messages on next slide:
P = processor number, A = address
# Directory Protocol Messages

<table>
<thead>
<tr>
<th>Message type</th>
<th>Source</th>
<th>Destination</th>
<th>Msg Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read miss</td>
<td>Local cache</td>
<td>Home directory</td>
<td>P, A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Processor P reads data at address A;</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>make P a read sharer and arrange to send data back</strong></td>
</tr>
<tr>
<td>Write miss</td>
<td>Local cache</td>
<td>Home directory</td>
<td>P, A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Processor P writes data at address A;</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>make P the exclusive owner and arrange to send data back</strong></td>
</tr>
<tr>
<td>Invalidate</td>
<td>Home directory</td>
<td>Remote caches</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Invalidate a shared copy at address A.</strong></td>
</tr>
<tr>
<td>Fetch</td>
<td>Home directory</td>
<td>Remote cache</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Fetch the block at address A and send it to its home directory</strong></td>
</tr>
<tr>
<td>Fetch/Invalidate</td>
<td>Home directory</td>
<td>Remote cache</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Fetch the block at address A and send it to its home directory; invalidate</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>the block in the cache</strong></td>
</tr>
<tr>
<td>Data value reply</td>
<td>Home directory</td>
<td>Local cache</td>
<td>Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Return a data value from the home memory (read miss response)</strong></td>
</tr>
<tr>
<td>Data write-back</td>
<td>Remote cache</td>
<td>Home directory</td>
<td>A, Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Write-back a data value for address A (invalidate response)</strong></td>
</tr>
</tbody>
</table>
State Transition Diagram for an Individual Cache Block in a Directory Based System

- States identical to snoopy case; transactions very similar.
- Transitions caused by read misses, write misses, invalidates, data fetch requests
- Generates read miss & write miss msg to home directory.
- Write misses that were broadcast on the bus for snooping => explicit invalidate & data fetch requests.
- Note: on a write, a cache block is bigger, so need to read the full cache block
CPU-Cache State Machine

- State machine for CPU requests for each memory block.
- Invalid state if in memory.

**Invalid**
- Send Data Write Back message to home directory.

**Shared** (read/only)
- Send Read Miss message.
- Send Write Miss message to home directory.
- Send Data Write Back message.

**Exclusive** (read/write)
- CPU Read hit.
- CPU Write hit.

- Fetch/Invalidate: Send Write Miss message.
- Fetch: Send Data Write Back message to home directory.
- CPU read miss: Send Data Write Back message and read miss to home directory.
- CPU write miss: Send Data Write Back message and Write Miss to home directory.
State Transition Diagram for the Directory

- Same states & structure as the transition diagram for an individual cache
- 2 actions: update of directory state & send msgs to satisfy requests
- Tracks all copies of memory block.
- Also indicates an action that updates the sharing set, Sharers, as well as sending a message.
Directory State Machine

- State machine for **Directory** requests for each memory block
- Uncached state if in memory

**Uncached**
- Data Write Back: Sharers = {} (Write back block)
- Write Miss: Sharers = {P}; send Data Value Reply msg

**Exclusive** (read/writ)
- Write Miss: Sharers = {P}; send Fetch/Invalidate; send Data Value Reply msg to remote cache (Write back block)

**Shared** (read only)
- Write Miss: send Invalidate to Sharers; then Sharers = {P}; send Data Value Reply msg
- Read miss: Sharers += {P}; send Fetch; send Data Value Reply msg to remote cache

11