Directory-based Cache Coherence Protocols

- Material in this lecture in Henessey and Patterson, Chapter 8
  - pgs. 677-685
- Some material from David Patterson’s slides for CS 252 at Berkeley

Interconnection Networks for Multiprocessors

- Buses have limitations for scalability:
  - Physical (number of devices that can be attached)
  - Performance (contention on a shared resource: the bus)
- Instead, provide each processor (or cluster of processors) with its own memory
- A general interconnection network allows processors to communicate
- Processors have fast access to local memory, slower access to "remote" memory located at another processor
  - NUMA machines (non-uniform memory access)
- What problems does this introduce?
Multiprocessor with Distributed Memory (NUMA)

Cache Coherence in NUMA Machines

- Snooping is not possible on media other than bus/ring
- Broadcast / multicast is not that easy
  - In Multistage Interconnection Networks (MINs), potential for blocking is very large
  - In mesh-like networks, broadcast to every node is very inefficient
- How to enforce cache coherence
  - Having no caches (Tera MTA)
  - By software: disallow caching of shared variables (Cray 3TD)
  - By hardware: having a data structure (a directory) that records the state of each block
Directory Protocol

- Similar to Snoopy Protocol: Three states
  - **Shared**: At least 1 processor has data cached, memory up-to-date
  - **Uncached/Invalid**: No processor has data cached, memory up-to-date
  - **Exclusive**: 1 processor (owner) has data cached; memory out-of-date
- In addition to cache state, Directory 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

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 (interact with CPU cache)
- Home node where the memory location of an address resides (interact with directory in memory)
- Remote node has a copy of a cache block, whether exclusive or shared (interact with CPU cache)

Example messages on following slide:
P = processor number, A = address
Example: Read Miss to an Uncached Block

Example: Read Miss to an Exclusive Block
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>– Processor P reads data at address A; make P a read sharer and arrange to send data back</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>– Processor P writes data at address A; make P the exclusive owner and arrange to send data back</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>– Invalidate a shared copy at address A.</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>– Fetch the block at address A and send it to its home directory</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>– Fetch the block at address A and send it to its home directory; invalidate the block in the cache</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>– Return a data value from the home memory (read miss response)</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>– Write-back a data value for address A (invalidate response)</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 actions for each cache block
- Invalid state if only in memory

Invalid

Shared (read/only)

Exclusive (read/write)

CPU Read hit

CPU Read Miss

Send Read Miss message

CPU Write:
Send Write Miss message to home directory

CPU Write miss:
Data write-back
Send Write miss

Fetch/Invalidate
send Data Write
Back message to home directory

CPU read hit
CPU write hit

Fetch: send
Data Write Back message
to home directory
(also do same for Read Miss)

Invalidate
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 actions for each memory block
- Uncached state if only in memory

(Uncached)

(Exclusive (read/write))

(Shared (read only))
Example Directory Protocol

- Message sent to directory causes two actions:
  - Update the directory
  - More messages to satisfy request

- Block is in Uncached state: the copy in memory is the current value; only possible requests for that block are:
  - Read miss: requesting processor sent data from memory & requestor made only sharing node; state of block made Shared.
  - Write miss: requesting processor is sent the value & becomes the Sharing node. The block is made Exclusive to indicate that the only valid copy is cached. Sharers indicates the identity of the owner.

- Block is Shared => the memory value is up-to-date:
  - Read miss: requesting processor is sent back the data from memory & requesting processor is added to the sharing set.
  - Write miss: requesting processor is sent the value. All processors in the set Sharers are sent invalidate messages, & Sharers is set to identity of requesting processor. The state of the block is made Exclusive.
Example Directory Protocol

- Block is Exclusive: current value of the block is held in the cache of the processor identified by the set Sharers (the owner) => three possible directory requests:
  - Read miss: owner processor sent data fetch message, causing state of block in owner’s cache to transition to Shared and causes owner to send data to directory, where it is written to memory & sent back to requesting processor. Identity of requesting processor is added to set Sharers, which still contains the identity of the processor that was the owner (since it still has a readable copy). State is shared.
  - Data write-back: owner processor is replacing the block and hence must write it back, making memory copy up-to-date (the home directory essentially becomes the owner), the block is now Uncached, and the Sharer set is empty.
  - Write miss: block has a new owner. A message is sent to old owner causing the cache to send the value of the block to the directory from which it is sent to the requesting processor, which becomes the new owner. Sharers is set to identity of new owner, and state of block is made Exclusive.

Implementing a Directory

- We assume operations atomic, but they are not; reality is much harder; must avoid deadlock when run out of buffers in network (see Appendix E)
- Optimizations:
  - read miss or write miss in Exclusive: send data directly to requestor from owner vs. 1st to memory and then from memory to requestor