Directory Coherence Protocols

Review: Small-Scale MIMD Designs
- Memory: centralized with uniform access time and bus interconnect

Review: Large-Scale MIMD Designs
- Memory: distributed with nonuniform access time and scalable interconnect (distributed memory)

Larger MPs
- Separate Memory per Processor
- Lousy 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 $f$ (memory size) vs. $f$ (cache size)
- Prevent directory as bottleneck: distribute directory entries with memory, each keeping track of which Procs have copies of their blocks
**Directory Protocol**

- Similar to Snoopy Protocol: Three states
  - Shared: $\geq 1$ processors have data, memory up-to-date
  - Uncached
  - Exclusive: 1 processor (owner) has data; memory out-of-date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
- Terms:
  - **Local node** is the node where a request originates
  - **Home node** is the node where the memory location of an address resides
  - **Remote node** is the node that has a copy of a cache block, whether exclusive or shared.

**Directory Protocol Messages**

<table>
<thead>
<tr>
<th>Message type</th>
<th>Source</th>
<th>Destination</th>
<th>Msg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read miss</td>
<td>Local processor</td>
<td>Home directory</td>
<td>P, A</td>
</tr>
<tr>
<td>Write miss</td>
<td>Local processor</td>
<td>Home directory</td>
<td>P, A</td>
</tr>
<tr>
<td>Invalidate</td>
<td>Home directory</td>
<td>Remote caches</td>
<td>A</td>
</tr>
<tr>
<td>Fetch</td>
<td>Home directory</td>
<td>Remote cache</td>
<td>A</td>
</tr>
<tr>
<td>Fetch/Invalidate</td>
<td>Home directory</td>
<td>Remote cache</td>
<td>A</td>
</tr>
<tr>
<td>Data value reply</td>
<td>Home directory</td>
<td>Local cache</td>
<td>Data</td>
</tr>
<tr>
<td>Data write-back</td>
<td>Remote cache</td>
<td>Home directory</td>
<td>A, Data</td>
</tr>
</tbody>
</table>

**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 is sent back the data from memory and the requestor is the only sharing node. The state of the block is made Shared.
  - Write miss: requesting processor is sent the value and becomes the Sharing node. All processors in the set Sharers are sent invalidate messages, & Sharers is set to identity of requesting processor.
- 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 is sent a data fetch message, which causes 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 and sent back to the 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).
  - **Data write-back**: owner processor is replacing the block and hence must write it back. This makes the 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.

State Transition Diagram for an Individual Cache Block in a Directory Based System

- The states are identical to those in the snoopy case, and the transactions are very similar with explicit invalidate and write-back requests replacing the write misses that were formerly broadcast on the bus.

State Transition Diagram for the Directory

- The same states and structure as the transition diagram for an individual cache
  - All actions are in color since they all are externally caused. Italics indicate the action taken the directory in response to the request. Bold italics indicate an action that updates the sharing set, Sharers, as opposed to sending a message.

Example

<table>
<thead>
<tr>
<th>step</th>
<th>P1</th>
<th>P2</th>
<th>Bus</th>
<th>Directory</th>
<th>Memo</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1:</td>
<td>Write 10 to A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1:</td>
<td>Read A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2:</td>
<td>Read A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2:</td>
<td>Write 20 to A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2:</td>
<td>Write 40 to A2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A1 and A2 map to the same cache block
Performance -- snoopy cache miss rates

snoop cache miss rates

Directory Miss Rates
Directory Miss Rates

- FFT
- LU
- Barnes
- Ocean

- Miss rate vs. Cache size (KB)

Directory Miss Rates

- FFT
- LU
- Barnes
- Ocean

- Miss rate vs. Block size (bytes)

Coherence Summary

- Snoopy vs. Directory
- Directory keeps track of both state of line and all users.
- Directory cost is function of memory size
- Coherence misses can be significant part of miss rate