**Directory Implementation**

Distributed memory
- each processor (or cluster of processors) has its own memory
- processor-memory pairs are connected via a multi-path interconnection network
  - snooping with broadcasting is wasteful
  - **point-to-point communication** instead
- a processor has fast access to its local memory & slower access to “remote” memory located at other processors
  - **NUMA** (non-uniform memory access) machines

How cache coherency is handled
- no caches (Tera (Cray) MTA)
- disallow caching of shared data (Cray 3TD)
- hardware directories that record cache block state

**A High-end MP**
**Directory Implementation**

Coherency state is associated with memory blocks that are the size of cache blocks

- cache state
  - **shared:**
    - at least 1 processor has the data cached & memory is up-to-date
    - block can be read by any processor
  - **exclusive:**
    - 1 processor (the owner) has the data cached & memory is stale
    - only that processor can write to it
  - **invalid:**
    - no processor has the data cached & memory is up-to-date
  - How tell which processors have read/write access
    - bit vector in which 1 means the processor has the data
    - optimization: space for 4 processors & trap for more
    - write bit

---

**Directory Implementation**

Directories have different meanings (& therefore uses) to different processors

- **home** node: where the memory location of an address resides (and cached data may be there too)
- **local** node: where the request initiated
- **remote** node: alternate location for the data if this processor has requested it

In satisfying a memory request:

- messages sent between the different nodes in point-to-point communication
- messages get explicit replies

Some simplifying assumptions for using the protocol

- processor blocks until the access is complete
- messages processed in the order received
Read Miss for an Uncached Block

1: read miss
2: data value reply

Read Miss for an Exclusive, Remote Block

1: read miss
2: fetch
4: data value reply
3: data write-back
**Write Miss for an Exclusive, Remote Block**

1: write miss

2: fetch & invalidate

4: data value reply

3: data write-back

**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 or write 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>
CPU FSM for a Cache Block

States identical to the snooping protocol
Transactions very similar
• read & write misses sent to home directory
• invalidate & data fetch requests to the node with the data
• replace broadcasted read/write misses
Directory FSM for a Memory Block

Same states and structure as for the cache block FSM
Tracks all copies of a memory block
Two state changes:
  • update coherency state
  • alter the number of sharers in the sharing set

Directory FSM for a Memory Block (directory state)

States:
- Uncached
- Shared (read only)
- Exclusive (read/write)

Transitions:
- Read miss:
  - Send data value reply
  - Sharers = {P}, W = 0
- Write miss:
  - Send data value reply
  - Sharers = {P}, W = P
- Write miss:
  - Send invalidate to all sharers
  - Send data value reply
  - Sharers = {P}, W = P
- Read miss:
  - Send data fetch to current owner
    (Data write-back)
  - Send data value reply
    Sharers += {P}, W = P
False Sharing

Processes share cache blocks, not data

Impact aggravated by:
  • block size: why?
  • cache size: why?
  • large miss penalties: why?

Reduced by:
  • coherency protocols (state per subblock)
    • let cache blocks become incoherent as long as there is only false sharing
    • make them coherent if any processor true shares
  • compiler optimizations (group & transpose, cache block padding)
  • cache-conscious programming wrt initial data structure layout