**Outline**
- Review
- Coherence
- Write Consistency
- Administrivia
- Snooping
- Building Blocks
- Snooping protocols and examples
- Coherence traffic and Performance on MP
- Directory-based protocols and examples
- Conclusion

---

**Example Write Back Snoopy Protocol**

- Invalidation protocol, write-back cache
  - Snoops every address on bus
  - If it has a dirty copy of requested block, provides that block in response to the read request and aborts the memory access

- Each memory block is in one state:
  - Shared: Clean in all caches & up-to-date in memory
  - Exclusive: Dirty in exactly one cache
  - OR Not in any caches

---

**Write-Back State Machine – CPU Events**

- State machine for CPU requests for each CPU
- Non-resident blocks invalid

---

**Write-Back State Machine- Bus events**

- State machine for bus requests for each cache block

---

**Example Write Back Snoopy Protocol**

- Each cache block is in one state (track these):
  - Shared: block can be read
  - Exclusive: cache has only copy, it is writeable & dirty
  - Invalid: block contains no data
    - Also applies to uniprocessor cache

- Read misses: cause all caches to snoop bus
- Writes to clean blocks are treated as misses
  - write-allocate
Example

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Bus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Memory</td>
</tr>
<tr>
<td>P1: Write 10 to A1</td>
<td>Excl.</td>
<td>A1 10</td>
</tr>
<tr>
<td>P2: Read A1</td>
<td>Shar.</td>
<td>A1 10</td>
</tr>
<tr>
<td>P2: Write 20 to A1</td>
<td>Excl.</td>
<td>A1 20</td>
</tr>
<tr>
<td>P2: Write 40 to A2</td>
<td>Excl.</td>
<td>A2 40</td>
</tr>
</tbody>
</table>

Assumes A1 and A2 map to same cache block

Example

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>Bus</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1: Write 10 to A1</td>
<td>Excl.</td>
<td>A1 10</td>
<td>RdMs</td>
<td>P1 A1</td>
</tr>
<tr>
<td>P2: Read A1</td>
<td>Shar.</td>
<td>A1 10</td>
<td>RdMs</td>
<td>P2 A1</td>
</tr>
<tr>
<td>P2: Write 20 to A1</td>
<td>Excl.</td>
<td>A1 20</td>
<td>RdMs</td>
<td>P2 A1</td>
</tr>
<tr>
<td>P2: Write 40 to A2</td>
<td>Excl.</td>
<td>A2 40</td>
<td>RdMs</td>
<td>P2 A2</td>
</tr>
</tbody>
</table>

Assumes A1 and A2 map to same cache block, but A1 != A2

SNOOPING WRITE BACK PROTOCOL

Complications: Write Races

- Cannot update cache until bus is obtained
  - Otherwise, another processor may get bus first, and then write the same cache block!
- Two step process:
  - Arbitrate for bus
  - Place miss on bus and complete operation

Complications: Write Races (cont.)

- If miss occurs to block while waiting for bus
  - Handle miss (invalidate may be needed) and then restart
- Split transaction bus:
  - Bus transaction is not atomic: can have multiple outstanding transactions for a block
  - Multiple misses can interleave, allowing two caches to grab block in the Exclusive state
  - Must track and prevent multiple misses for one block

Limitations in Symmetric Shared-Memory Multiprocessors and Snooping Protocols

- Single memory to accommodate all CPUs
  - Multiple memory banks
- Bus must support both coherence traffic & normal memory traffic
  - Multiple buses or interconnection networks (cross bar or small point-to-point)
- Example - AMD Opteron:
  - Memory connected directly to each dual-core chip
  - Point-to-point connections for up to 4 chips
  - Remote memory and local memory latency are similar, allowing OS to view Opteron as uniform memory access (UMA) computer
Performance of Symmetric Shared-Memory MPs

- Cache performance is combination of
  - Uniprocessor cache miss traffic
  - Traffic caused by communication
    - Results in invalidations and subsequent cache misses

- 4th C: coherency miss
  - Joins Compulsory, Capacity, Conflict

Example: True v. False Sharing v. Hit?

- Assume x1 and x2 in same cache block.
  P1 and P2 both read x1 and x2 before.

<table>
<thead>
<tr>
<th>Time</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Write x1</td>
<td>True miss; invalidate x1 in P2</td>
</tr>
<tr>
<td>2</td>
<td>Read x2</td>
<td>False miss; x1 irrelevant to P2</td>
</tr>
<tr>
<td>3</td>
<td>Write x1</td>
<td>False miss; x1 irrelevant to P2</td>
</tr>
<tr>
<td>4</td>
<td>Write x2</td>
<td>False miss; x1 irrelevant to P2</td>
</tr>
<tr>
<td>5</td>
<td>Read x2</td>
<td>True miss; invalidate x2 in P1</td>
</tr>
</tbody>
</table>

Coherency Misses

- True sharing misses arise from the communication of data through the cache coherence mechanism
  - Invalidates due to 1st write to shared block
  - Reads by another CPU of modified block in different cache
    - Miss would still occur if block size were 1 word

- False sharing misses when a block is invalidated because some word in the block, other than the one being read, is written into
  - Invalidation does not cause a new value to be communicated, but only causes an extra cache miss
    - Block is shared, but no word in block is actually shared
      ⇒ miss would not occur if block size were 1 word

Outline

- Review
- Coherence
- Write Consistency
- Administrivia
- Snooping
- Building Blocks
- Snooping protocols and examples
- Coherence traffic and Performance on MP
- Directory-based protocols and examples
- Conclusion
A Cache Coherent System Must:

- Provide set of states, state transition diagram, and actions
- Manage coherence protocol
  - (0) Determine when to invoke coherence protocol
  - (a) Find info about other copies to determine action
    - whether need to communicate with other cached copies
  - (b) Locate the other copies
  - (c) Communicate with those copies (invalidate/update)
- (0) is done the same way on all systems
  - state of the line is maintained in the cache
  - protocol is invoked if an "access fault" occurs on the line
- Different approaches distinguished by (a) to (c)

Bus-based Coherence

- All of (a), (b), (c) done through broadcast on bus
  - Faulting processor sends out a "search"
  - Others respond to the search probe and take necessary action
- Could do it in scalable network, too
  - Broadcast to all processors, and let them respond
  - Conceptually simple, but broadcast doesn’t scale with p (# of processors)
  - On bus, bus bandwidth doesn’t scale
  - On scalable network, every fault leads to at least p network transactions

Scalable Approach: Directories

- Scalable coherence uses
  - Same cache states and state transition diagram…
  - But different mechanisms to manage protocol
- Every memory block has associated directory information
  - Keeps track of copies of cached blocks and their states
  - On a miss, find directory entry, look it up, and communicate only with the nodes that have copies if necessary
  - In scalable networks, communication with directory and copies is through network transactions
- Many alternatives for organizing directory information

Basic Operation of Directory

- k processors.
- With each cache-block in memory:
  - k presence-bits, 1 dirty-bit
- With each cache-block in cache:
  - 1 valid bit, and 1 dirty (owner) bit

Picture To Keep In Mind

- Cost structure:
  - cache: ~2 cycles
  - local memory: ~80 cycles
  - remote memory: ~200 cycles

Directory Protocol

- Similar to Snoopy Protocol: Three states
  - Shared: 1 processor has 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 2**

- 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

---

**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

---

**Basic Directory Transactions**

1. Read request to directory

2. Reply with owner identity

3. Read req. to owner

4a. Data

4b. Reply

5. Revision message to directory

---

**A Popular Middle Ground**

- Two-level “hierarchy”
- Individual nodes are multiprocessors, connected non-hierarchically
  - e.g. mesh of SMPs
- Coherence across nodes is directory-based
  - directory keeps track of nodes, not individual processors
- Coherence within nodes is snooping or directory
  - orthogonal, but needs a good interface of functionality
- SMP on a chip directory + snoop?

---

**And in Conclusion ...**

- Caches contain all information on state of cached memory blocks
- Snooping cache over shared medium for smaller MP by invalidating other cached copies on write
- Sharing cached data → Coherence (values returned by a read), Consistency (when a written value will be returned by a read)
- Snooping and Directory Protocols similar; bus makes snooping easier because of broadcast (snooping => uniform memory access)
- Directory has extra data structure to keep track of state of all cache blocks
- Distributing directory => scalable shared address space multiprocessor
  - => Cache coherent, Non uniform memory access