More on Coherence

- A memory system is coherent if:
  - A read by a processor P to a location X that follows a write by P to X, with no writes of X by another processor occurring between the write and the read by P, always returns the value written by P.
  - A read by a processor to location X that follows a write by another processor to X returns the written value if the read and write are sufficiently separated in time and no other writes to X occur between the two accesses.
  - Writes to the same location are serialized; that is, two writes to the same location by any two processors are seen in the same order by all processors.

Memory Coherence in SMPs

Suppose CPU-1 updates A to 200.
- write-back: memory and cache-2 have stale values
- write-through: cache-2 has a stale value

Do these stale values matter?
What is the view of shared memory for programming?

Problems with Parallel I/O

- Memory → Disk: Physical memory may be stale if cache copy is dirty
- Disk → Memory: Cache may hold stale data and not see memory writes

Snoopy Cache Goodman 1983

- Idea: Have cache watch (or snoop upon) DMA transfers, and then “do the right thing”
- Snoopy cache tags are dual-ported
### Snoopy Cache Actions for DMA

<table>
<thead>
<tr>
<th>Observed Bus Cycle</th>
<th>Cache State</th>
<th>Cache Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA Read Memory -&gt; Disk</td>
<td>Address not cached</td>
<td>No action</td>
</tr>
<tr>
<td></td>
<td>Cached, unmodified</td>
<td>No action</td>
</tr>
<tr>
<td></td>
<td>Cached, modified</td>
<td>Cache intervenes</td>
</tr>
<tr>
<td>DMA Write Disk -&gt; Memory</td>
<td>Address not cached</td>
<td>No action</td>
</tr>
<tr>
<td></td>
<td>Cached, unmodified</td>
<td>Cache purges its copy</td>
</tr>
<tr>
<td></td>
<td>Cached, modified</td>
<td>??</td>
</tr>
</tbody>
</table>

### Shared Memory Multiprocessor

Use snoopy mechanism to keep all processors’ view of memory coherent.

### Snoopy Cache Coherence Protocols

**write miss:**
- The address is invalidated in all other caches before the write is performed.

**read miss:**
- If a dirty copy is found in some cache, a write-back is performed before the memory is read.

### Cache State Transition Diagram

The MSI protocol:

- Each cache line has state bits:
  - **M**: Modified
  - **S**: Shared
  - **I**: Invalid

- Address tag state bits:
  - Write miss (P1 gets line from memory)
  - Other processor reads (P1 writes back)
  - Other processor intent to write (P1 writes back)
  - Other processor intent to write

- Other processor reads (P1 gets line from memory)

- Other processor intent to write

- Other processor writes

- Other processor writes

- Other processor writes

- Other processor writes

- Other processor writes

- Other processor writes

### Observation

- If a line is in the M state then no other cache can have a copy of the line!
  - Memory stays coherent, multiple differing copies cannot exist.
MESI: An Enhanced MSI protocol
increased performance for private data

Each cache line has a tag

M: Modified Exclusive
E: Exclusive but unmodified
S: Shared
I: Invalid

Cache state in processor $P_1$

Acknowledgements

- These slides heavily contain material developed and copyright by:
  - Krste Asanovic (MIT/UCB)
  - David Patterson (UCB)
- And also by:
  - Arvind (MIT)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
- MIT material derived from course 6.823
- UCB material derived from course CS252

CSE 490/590 Administrivia

- Keyboards available for pickup at my office
- Project 2: 2 weeks left (Deadline 5/2)
  - Will have demo sessions
  - Keyboard helper code will be available
- Final exam: Thursday 5/5, 11:45pm – 2:45pm