Parallel Processing: Multiprocessor Systems (MIMD)

• MP - A computer system with at least 2 processors:
  - Processor
  - Cache
  - Interconnection Network
  - Memory
  - I/O

• Q1 – How do they share data?
• Q2 – How do they coordinate?
• Q3 – How many processors can be supported?
Shared Memory Multiprocessor (SMP)

• Q1 – Single address space shared by all processors/cores

• Q2 – Processors coordinate/communicate through shared variables in memory (via loads and stores)
  – Use of shared data must be coordinated via synchronization primitives (locks) that allow access to data to only one processor at a time

• All multicore computers today are SMP
Three Key Questions about Multiprocessors

• Q3 – How many processors can be supported?
• Key bottleneck in an SMP is the memory system
• Caches can effectively increase memory bandwidth/open the bottleneck
• But what happens to the memory being actively shared among the processors through the caches?
Shared Memory and Caches

• What if?
  – Processors 1 and 2 read Memory[1000] (value 20)
Shared Memory and Caches

• What if?
  – Processors 1 and 2 read Memory[1000]
  – Processor 0 writes Memory[1000] with 40

Processor 0
Write
Invalidates
Other Copies
(Easy! Turn
Valid bit off)

Interconnection Network
Keeping Multiple Caches Coherent

- Architect’s job: shared memory ➔ keep cache values coherent
- Idea: When any processor has cache miss or writes, notify other processors via interconnection network
  - If only reading, many processors can have copies
  - If a processor writes, invalidate all other copies
- Shared written result can “ping-pong” between caches
How Does HW Keep $ Coherent?

Each cache tracks state of each block in cache:

*Shared*: up-to-date data, not allowed to write
other caches may have a copy

*Modified*: up-to-date, changed (dirty), OK to write
no other cache has a copy,
copy in memory is out-of-date
- must respond to read request

*Invalid*: Not really in the cache
2 Optional Performance Optimizations of Cache Coherency via new States

*Exclusive*: up-to-date data, OK to write (change to modified)
   - no other cache has a copy,
   - copy in memory up-to-date
   - Avoids writing to memory if block replaced
   - Supplies data on read instead of going to memory

*Owner*: up-to-date data, OK to write (if invalidate shared copies first then change to modified)
   - other caches may have a copy (they must be in Shared state)
   - copy in memory not up-to-date
   - So, owner must supply data on read instead of going to memory
Common Cache Coherency Protocol: MOESI (snoopy protocol)

- Each block in each cache is in one of the following states:
  - **Modified** (in cache)
  - **Owned** (in cache)
  - **Exclusive** (in cache)
  - **Shared** (in cache)
  - **Invalid** (not in cache)

**Compatibility Matrix:** Allowed states for a given cache block in any pair of caches
Common Cache Coherency Protocol: MOESI (snoopy protocol)

- Each block in each cache is in one of the following states:
  - **M**odified (in cache)
  - **O**wned (in cache)
  - **E**xclusive (in cache)
  - **S**hared (in cache)
  - **I**nvalid (not in cache)

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>O</th>
<th>E</th>
<th>S</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M</strong></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>O</strong></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✔</td>
</tr>
<tr>
<td><strong>E</strong></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✔</td>
</tr>
<tr>
<td><strong>S</strong></td>
<td>X</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>I</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

**Compatibility Matrix**: Allowed states for a given cache block in any pair of caches
Cache Coherency (MOESI protocol)

- **Invalid**: Read Miss Exclusive → Exclusive, Read Miss Shared → Probe Read Hit
- **Exclusive**: Write Hit → Modified, Write Miss (WB memory) → Shared
- **Shared**: Read Hit → Owner, Probe Read Hit → Read Hit, Read Miss Shared → Read Miss Exclusive
- **Modified**: Write Hit → Owner, Read Hit → Read Hit
- **Owner**: Read Hit → Read Hit, Read Hit → Read Hit

(May consider Owned as special case of Shared)

• "Read" and "Write" are by this core.
• "Probe Read" and "Probe Write" are reads and writes by others, that must probe this core's caches.
Cache Coherency and Block Size

• Suppose block size is 32 bytes
• Suppose Processor 0 reading and writing variable X, Processor 1 reading and writing variable Y
• Suppose in X location 4000, Y in 4012
• What will happen?
• Effect called *false sharing*
• How can you prevent it?
And In Conclusion, ...

• Sequential software is slow software
  – SIMD and MIMD only path to higher performance
• Multiprocessor (Multicore) uses Shared Memory (single address space)
• Cache coherency implements shared memory even with multiple copies in multiple caches
  – False sharing a concern
• Next Time: OpenMP as simple parallel extension to C