Thread-Level Parallelism

ECE 154B
Dmitri Strukov
Introduction

• Thread-Level parallelism
  – Have multiple program counters and resources
  – Uses MIMD model
  – Targeted for tightly-coupled shared-memory multiprocessors

• Threads can be used for data-level parallelism, but the overheads may outweigh the benefit
Natural Extension of Memory System

- Shared Cache
- Centralized Memory Dance Hall, UMA
- Distributed Memory (NUMA)
SMP vs DSM

- **Symmetric multiprocessors (SMP)**
  - Small number of cores
  - Share single memory with uniform memory latency
- **Distributed shared memory (DSM)**
  - Memory distributed among processors
  - Non-uniform memory access/latency (NUMA)
  - Processors connected via direct (switched) and non-direct (multi-hop) interconnection networks
Example Cache Coherence Problem

Things to note:

- Processors see different values for u after event 3
- With write back caches, value written back to memory depends on happenstance of which cache flushes or writes back value when
  - Processes accessing main memory may see very stale value
- Unacceptable to programs, and frequent!
Cache Coherence

• Snoopy schemes vs. Directory schemes
  – Snooping Solution:
    • Send all requests for data to all processors
    • Processors snoop to see if they have a copy and respond accordingly
    • Requires broadcast, since caching information is at processors
    • Works well with bus (natural broadcast medium)
    • Dominates for small scale machines (most of the market)
  – Directory-Based Schemes
    • Keep track of what is being shared in a centralized place (logically)
    • Distributed memory ⇒ distributed directory for scalability (avoids bottlenecks)
    • Send point-to-point requests to processors via network
    • Scales better than Snooping

• Write through vs. write-back protocols
• Update vs. invalidation protocols
Invalidation vs. Update (Broadcast)

• Write Invalidate Protocol:
  – Write to shared data: an invalidate is sent to all caches which snoop and invalidate any copies
  – Read Miss:
    » Write-through: memory is always up-to-date
    » Write-back: snoop in caches to find most recent copy

• Write Broadcast Protocol:
  – Write to shared data: broadcast on bus, processors snoop, and update copies
  – Read miss: memory is always up-to-date

• Bandwidth vs. latency tradeoff
An Example Snoopy Protocol

• Invalidation protocol, write-back cache
• Each block of memory is in one state:
  – Clean in all caches and up-to-date in memory (Shared)
  – OR Dirty in exactly one cache (Exclusive)
  – OR Not in any caches
• Each cache block is in one state (track these):
  – Shared: block can be read
  – OR Exclusive: cache has only copy, its writeable, and dirty
  – OR Invalid: block contains no data
• Read misses: cause all caches to snoop bus
• Writes to clean line are treated as misses
# Snoopy Coherence Protocols

<table>
<thead>
<tr>
<th>Request</th>
<th>Source</th>
<th>State of addressed cache block</th>
<th>Type of cache action</th>
<th>Function and explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read hit</td>
<td>Processor</td>
<td>Shared or modified</td>
<td>Normal hit</td>
<td>Read data in local cache.</td>
</tr>
<tr>
<td>Read miss</td>
<td>Processor</td>
<td>Invalid</td>
<td>Normal miss</td>
<td>Place read miss on bus.</td>
</tr>
<tr>
<td>Read miss</td>
<td>Processor</td>
<td>Shared</td>
<td>Replacement</td>
<td>Address conflict miss: place read miss on bus.</td>
</tr>
<tr>
<td>Read miss</td>
<td>Processor</td>
<td>Modified</td>
<td>Replacement</td>
<td>Address conflict miss: write-back block, then place read miss on bus.</td>
</tr>
<tr>
<td>Write hit</td>
<td>Processor</td>
<td>Modified</td>
<td>Normal hit</td>
<td>Write data in local cache.</td>
</tr>
<tr>
<td>Write hit</td>
<td>Processor</td>
<td>Shared</td>
<td>Coherence</td>
<td>Place invalidate on bus. These operations are often called upgrade or ownership misses, since they do not fetch the data but only change the state.</td>
</tr>
<tr>
<td>Write miss</td>
<td>Processor</td>
<td>Invalid</td>
<td>Normal miss</td>
<td>Place write miss on bus.</td>
</tr>
<tr>
<td>Write miss</td>
<td>Processor</td>
<td>Shared</td>
<td>Replacement</td>
<td>Address conflict miss: place write miss on bus.</td>
</tr>
<tr>
<td>Write miss</td>
<td>Processor</td>
<td>Modified</td>
<td>Replacement</td>
<td>Address conflict miss: write-back block, then place write miss on bus.</td>
</tr>
<tr>
<td>Read miss</td>
<td>Bus</td>
<td>Shared</td>
<td>No action</td>
<td>Allow shared cache or memory to service read miss.</td>
</tr>
<tr>
<td>Read miss</td>
<td>Bus</td>
<td>Modified</td>
<td>Coherence</td>
<td>Attempt to share data: place cache block on bus and change state to shared.</td>
</tr>
<tr>
<td>Invalidate</td>
<td>Bus</td>
<td>Shared</td>
<td>Coherence</td>
<td>Attempt to write shared block; invalidate the block.</td>
</tr>
<tr>
<td>Write miss</td>
<td>Bus</td>
<td>Shared</td>
<td>Coherence</td>
<td>Attempt to write shared block; invalidate the cache block.</td>
</tr>
<tr>
<td>Write miss</td>
<td>Bus</td>
<td>Modified</td>
<td>Coherence</td>
<td>Attempt to write block that is exclusive elsewhere; write-back the cache block and make its state invalid in the local cache.</td>
</tr>
</tbody>
</table>
Snoopy-Cache State Machine-I

- State machine for CPU requests for each cache block

**Invalid**
- CPU Read
- Place read miss on bus

**Shared** (read/only)
- CPU Read hit
- CPU Read miss
- Place read miss on bus

**Exclusive** (read/write)
- CPU Read hit
- CPU Write
- Place Write Miss on bus
- CPU Write Miss
- Write back block, Place read miss on bus
- CPU Write Miss
- Place write miss on bus
Snoopy-Cache State Machine-II

- State machine for *bus* requests for each *cache block*
- Appendix I gives details of bus requests
Snoopy-Cache State Machine-III

- State machine for **CPU** requests for each cache block **and** for **bus** requests for each cache block.

  **Cache Block State**
  - Invalid
  - Shared (read/only)
  - Exclusive (read/write)

  **CPU Read**
  - Place read miss on bus
  - Write miss for this block

  **CPU Write**
  - Place Write Miss on bus
  - Write back block, Place read miss on bus
  - Write miss for this block

  **Write Back Block; (abort memory access)**

  **Write Back Block; (abort memory access)**

  **CPU Read hit**
  - Read miss for this block

  **CPU Write Miss**
  - Write back cache block
  - Place write miss on bus
Example

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

What happen if P1 reads A1 at this time?
Coherence Protocols: Extensions

- Shared memory bus and snooping bandwidth is bottleneck for scaling symmetric multiprocessors
  - Duplicating tags
  - Place directory in outermost cache
  - Use crossbars or point-to-point networks with banked memory
True vs. False Sharing Misses

• Coherence influences cache miss rate
  – Coherence misses
    • True sharing misses
      – Write to shared block (transmission of invalidation)
      – Read an invalidated block
    • False sharing misses
      – Read an unmodified word in an invalidated block
Performance Study: Commercial Workload

- Performed on AlphaServer (21164) with three level memory hierarchy (1998)
- OLTP, DSS and AltaVista benchmarks

True sharing misses is constant with $L3$ size... grows with processor count ... and drops with $L3$ block size
Another Example for Snooping

<table>
<thead>
<tr>
<th>Request</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0: read 120</td>
<td>P0.B0: (S, 120, 0020) returns 0020</td>
</tr>
<tr>
<td>P0: write 120</td>
<td>P0.B0: (S, 120, 0020) returns 0020</td>
</tr>
<tr>
<td>P1: read 110</td>
<td>P0.B0: (S, 120, 0020) returns 0020</td>
</tr>
<tr>
<td>P0: write 108</td>
<td>P0.B0: (S, 120, 0020) returns 0020</td>
</tr>
<tr>
<td>P0: write 130</td>
<td>P0.B0: (S, 120, 0020) returns 0020</td>
</tr>
<tr>
<td>P3: write 130</td>
<td>P0.B0: (S, 120, 0020) returns 0020</td>
</tr>
</tbody>
</table>
Another Example for Snooping

<table>
<thead>
<tr>
<th>Request</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P0: read 120</strong></td>
<td><strong>P0.B0: (S, 120, 0020) returns 0020</strong></td>
</tr>
<tr>
<td>P0: write 120 ← 80</td>
<td><strong>P0.B0: (M, 120, 0080)</strong></td>
</tr>
<tr>
<td></td>
<td><strong>P3.B0: (I, 120, 0020)</strong></td>
</tr>
<tr>
<td>P1: read 110</td>
<td></td>
</tr>
<tr>
<td>P0: write 108 ← 48</td>
<td></td>
</tr>
<tr>
<td>P0: write 130 ← 78</td>
<td></td>
</tr>
<tr>
<td>P3: write 130 ← 78</td>
<td></td>
</tr>
</tbody>
</table>
Another Example for Snooping

<table>
<thead>
<tr>
<th>Request</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0: read 120</td>
<td>P0:B0: (S, 120, 0020) returns 0020</td>
</tr>
<tr>
<td>P0: write 120</td>
<td>P0:B0: (M, 120, 0080)</td>
</tr>
<tr>
<td></td>
<td>P3:B0: (I, 120, 0020)</td>
</tr>
<tr>
<td>P1: read 110</td>
<td>P0:B2: (S, 110, 0030)</td>
</tr>
<tr>
<td></td>
<td>P1:B2: (S, 110, 0030) returns 0030</td>
</tr>
<tr>
<td></td>
<td>M: 110 ← 0030 (writeback)</td>
</tr>
<tr>
<td>P0: write 108</td>
<td>48</td>
</tr>
<tr>
<td>P0: write 130</td>
<td>78</td>
</tr>
<tr>
<td>P3: write 130</td>
<td>78</td>
</tr>
</tbody>
</table>
Another Example for Snooping

<table>
<thead>
<tr>
<th>Request</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P0: read 120</strong></td>
<td>P0.B0: (S, 120, 0020) returns 0020</td>
</tr>
<tr>
<td>P0: write 120 $\leftarrow$ 80</td>
<td>P0.B0: (M, 120, 0080)</td>
</tr>
<tr>
<td></td>
<td>P3.B0: (I, 120, 0020)</td>
</tr>
<tr>
<td>P1: read 110</td>
<td>P0.B2: (S, 110, 0030)</td>
</tr>
<tr>
<td></td>
<td>P1.B2: (S, 110, 0030) returns 0030</td>
</tr>
<tr>
<td></td>
<td>M: 110 $\leftarrow$ 0030 (writeback)</td>
</tr>
<tr>
<td>P0: write 108 $\leftarrow$ 48</td>
<td>P0.B1: (M, 108, 0048)</td>
</tr>
<tr>
<td>P0: write 130 $\leftarrow$ 78</td>
<td></td>
</tr>
<tr>
<td>P3: write 130 $\leftarrow$ 78</td>
<td></td>
</tr>
</tbody>
</table>
Another Example for Snooping

<table>
<thead>
<tr>
<th>Request</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P0: read 120</strong></td>
<td><strong>P0.B0: (S, 120, 0020) returns 0020</strong></td>
</tr>
<tr>
<td><strong>P0: write 120 ← 80</strong></td>
<td><strong>P0.B0: (M, 120, 0080)</strong></td>
</tr>
<tr>
<td></td>
<td><strong>P3.B0: (I, 120, 0020)</strong></td>
</tr>
<tr>
<td><strong>P1: read 110</strong></td>
<td><strong>P0.B2: (S, 110, 0030)</strong></td>
</tr>
<tr>
<td></td>
<td><strong>P1.B2: (S, 110, 0030) returns 0030</strong></td>
</tr>
<tr>
<td></td>
<td><strong>M: 110 ← 0030 (writeback)</strong></td>
</tr>
<tr>
<td><strong>P0: write 108 ← 48</strong></td>
<td><strong>P0.B1: (M, 108, 0048)</strong></td>
</tr>
<tr>
<td><strong>P0: write 130 ← 78</strong></td>
<td><strong>P0.B2: (M, 130, 0078)</strong></td>
</tr>
<tr>
<td><strong>P3: write 130 ← 78</strong></td>
<td></td>
</tr>
</tbody>
</table>
Another Example for Snooping

<table>
<thead>
<tr>
<th>Request</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0: read 120</td>
<td>P0.B0: (S, 120, 0020) returns 0020</td>
</tr>
</tbody>
</table>
| P0: write 120 ← 80  | P0.B0: (M, 120, 0080)  
                    | P3.B0: (I, 120, 0020)                                                 |
| P1: read 110   | P0.B2: (S, 110, 0030)  
                    | P1.B2: (S, 110, 0030) returns 0030  
                    | M: 110 ← 0030 (writeback)                                              |
| P0: write 108 ← 48  | P0.B1: (M, 108, 0048)  
| P0: write 130 ← 78  | P0.B2: (M, 130, 0078)                                                 |
| P3: write 130 ← 78  | P0.B2: (I, 130, 0078)  
                       | P3.B2: (M, 130, 0078) returns 0078  
                       | M: 110 ← 0078 (writeback)                                              |
Directory Protocols

• Directory keeps track of every block
  – Which caches have each block
  – Dirty status of each block

• Implement in shared L3 cache
  – Keep bit vector of size = # cores for each block in L3
  – Not scalable beyond shared L3

• Implement in a distributed fashion:
Directory Protocols

• For each block, maintain state:
  – Shared
    • One or more nodes have the block cached, value in memory is up-to-date
    • Set of node IDs
  – Uncached
  – Modified
    • Exactly one node has a copy of the cache block, value in memory is out-of-date
    • Owner node ID

• Directory maintains block states and sends invalidation messages
Directory Protocols

FSM for individual cache block

FSM for memory block in directory
## Messages

<table>
<thead>
<tr>
<th>Message type</th>
<th>Source</th>
<th>Destination</th>
<th>Message contents</th>
<th>Function of this message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read miss</td>
<td>Local cache</td>
<td>Home directory</td>
<td>P, A</td>
<td>Node P has a read miss at address A; request data and make P a read sharer.</td>
</tr>
<tr>
<td>Write miss</td>
<td>Local cache</td>
<td>Home directory</td>
<td>P, A</td>
<td>Node P has a write miss at address A; request data and make P the exclusive owner.</td>
</tr>
<tr>
<td>Invalidate</td>
<td>Local cache</td>
<td>Home directory</td>
<td>A</td>
<td>Request to send invalidates to all remote caches that are caching the block at address A.</td>
</tr>
<tr>
<td>Invalidate</td>
<td>Home directory</td>
<td>Remote cache</td>
<td>A</td>
<td>Invalidate a shared copy of data at address A.</td>
</tr>
<tr>
<td>Fetch</td>
<td>Home directory</td>
<td>Remote cache</td>
<td>A</td>
<td>Fetch the block at address A and send it to its home directory; change the state of A in the remote cache to shared.</td>
</tr>
<tr>
<td>Fetch/invalidate</td>
<td>Home directory</td>
<td>Remote cache</td>
<td>A</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>D</td>
<td>Return a data value from the home memory.</td>
</tr>
<tr>
<td>Data write-back</td>
<td>Remote cache</td>
<td>Home directory</td>
<td>A, D</td>
<td>Write-back a data value for address A.</td>
</tr>
</tbody>
</table>
Directory Protocols

• For uncached block:
  – Read miss
    • Requesting node is sent the requested data and is made the only sharing node, block is now shared
  – Write miss
    • The requesting node is sent the requested data and becomes the sharing node, block is now exclusive

• For shared block:
  – Read miss
    • The requesting node is sent the requested data from memory, node is added to sharing set
  – Write miss
    • The requesting node is sent the value, all nodes in the sharing set are sent invalidate messages, sharing set only contains requesting node, block is now exclusive
Directory Protocols

• For exclusive block:
  – Read miss
    • The owner is sent a data fetch message, block becomes shared, owner sends data to the directory, data written back to memory, sharers set contains old owner and requestor
  – Data write back
    • Block becomes uncached, sharer set is empty
  – Write miss
    • Message is sent to old owner to invalidate and send the value to the directory, requestor becomes new owner, block remains exclusive
Directory-based Protocol

Interconnection Network

Directories

Memories

Caches

CPU 0

CPU 1

CPU 2

Bit Vector

X

U 0 0 0

X

7
CPU 0 Reads X

Interconnection Network

Directories

Memories

Caches

CPU 0

CPU 1

CPU 2

Read Miss

X U 0 0 0

X 7
CPU 0 Reads X

Interconnection Network

Directories

- CPU 0
- CPU 1
- CPU 2

Memories

- CPU 0
- CPU 1
- CPU 2

Caches

- CPU 0
- CPU 1
- CPU 2

S 1 0 0

7
CPU 0 Reads X

Interconnection Network

Directories

Memories

Caches

CPU 0

CPU 1

CPU 2

X 7

X S 1 0 0

X 7
CPU 2 Reads X
CPU 2 Reads X

Interconnection Network

Directories

Memories

Caches

CPU 0

CPU 1

CPU 2

S 1 0 1

7

X

X

X
CPU 2 Reads X

Interconnection Network

Directories

Memories

Caches

CPU 0

CPU 1

CPU 2

S 1 0 1

7

7

7
CPU 0 Writes 6 to X

Directories

Memories

Caches

Write Miss

CPU 0

CPU 1

CPU 2

Interconnection Network

S 1 0 1

7

7

7
CPU 0 Writes 6 to X
CPU 0 Writes 6 to X
CPU 1 Reads X

Interconnection Network

Directories

Memories

Caches

CPU 0

CPU 1

CPU 2
CPU 1 Reads X

Interconnection Network

Switch to Shared

Directories

Memories

Caches

X 6

CPU 0

X

CPU 1

X E 1 0 0

CPU 2

X 7
CPU 1 Reads X

Interconnection Network

CPU 0

CPU 1

CPU 2

Directories

Memories

Caches

X 6

E 1 0 0

X 6
CPU 1 Reads X
CPU 2 Writes 5 to X

Interconnection Network

Directories

Memories

Caches

CPU 0

CPU 1

CPU 2

Write Miss

S 1 1 0

X

X

X

6

6

6
CPU 2 Writes 5 to X

Interconnection Network

Directories

Memories

Caches

CPU 0

CPU 1

CPU 2

Invalidation
CPU 2 Writes 5 to X (Write back)
CPU 0 Writes 4 to X
CPU 0 Writes 4 to X
CPU 0 Writes 4 to X
CPU 0 Writes 4 to X
CPU 0 Writes 4 to X
CPU 0 Writes 4 to X
Acknowledgements

Some of the slides contain material developed and copyrighted by D. Patterson (UCB), B. Chen (U Arkansas) and instructor material for the textbook