Directory-Based Cache Coherence

Joel Emer
Computer Science and Artificial Intelligence Lab
M.I.T.

http://www.csg.csail.mit.edu/6.823
Maintaining Cache Coherence

It is sufficient to have hardware such that
- only one processor at a time has write permission for a location
- no processor can load a stale copy of the location after a write

A correct approach could be:

write request:
The address is *invalidated* in all other caches *before* the write is performed

read request:
If a dirty copy is found in some cache, a write-back is performed before the memory is read
Directory-Based Coherence (Censier and Feautrier, 1978)

- Snoopy schemes broadcast requests over memory bus
- Difficult to scale to large numbers of processors
- Requires additional bandwidth to cache tags for snoop requests

- Directory schemes send messages to only those caches that might have the line
- Can scale to large numbers of processors
- Requires extra directory storage to track possible sharers
An MSI Directory Protocol

- Cache states: Modified (M) / Shared (S) / Invalid (I)
- Directory states:
  - Uncached (Un): No sharers
  - Shared (Sh): One or more sharers with read permission (S)
  - Exclusive (Ex): A single sharer with read & write permissions (M)
- Transient states not drawn for clarity; for now, assume no racing requests
Transitions initiated by directory requests:

- **M**
  - DownReq / DownResp (with data)
  - InvReq / InvResp (with data)
  - InvReq / InvResp (without data)

- **S**
  - DownReq / DownResp (with data)

- **I**

<table>
<thead>
<tr>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalidation Request (InvReq)</td>
</tr>
<tr>
<td>Downgrade Request (DownReq)</td>
</tr>
<tr>
<td>Invalidation Response (InvResp)</td>
</tr>
<tr>
<td>Downgrade Response (DownResp)</td>
</tr>
</tbody>
</table>
Transitions initiated by evictions:

- Eviction / WbReq (with data)
- Eviction / WbReq (without data)

Actions:

- Writeback Request (WbReq)
MSI Protocol: Caches

- Transitions initiated by processor accesses
- Transitions initiated by directory requests
- Transitions initiated by evictions
Transitions initiated by data requests:

- \( \text{ExReq} / \text{Sharers} = \{P\} \); \( \text{ExResp} \)
- \( \text{ShReq} / \text{Down(Sharer)} \); \( \text{Sharers} = \text{Sharer} + \{P\} \); \( \text{ShResp} \)
- \( \text{ExReq} / \text{Inv(Sharers)} \); \( \text{Sharers} = \{P\} \); \( \text{ExResp} \)
- \( \text{ShReq} / \text{Sharers} = \text{Sharers} + \{P\} \); \( \text{ShResp} \)
- \( \text{ShReq} / \text{Sharers} = \{P\} \); \( \text{ShResp} \)
Transitions initiated by writeback requests:

- **Ex**
  - \( WbReq \land \text{Sharers} = \{\} \); \( WbResp \)

- **Sh**
  - \( WbReq \land |\text{Sharers}| > 1 \)
  - \( \text{Sharers} = \text{Sharers} - \{P\} \); \( WbResp \)

- **Un**
  - \( WbReq \land |\text{Sharers}| = 1 \)
  - \( \text{Sharers} = \{\} \); \( WbResp \)
MSI Directory Protocol Example

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Core 0

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Core 1

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Core 2
MSI Directory Protocol Example

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Core 0

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Core 1

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Core 2

LD 0xA
MSI Directory Protocol Example

1. LD 0xA
MSI Directory Protocol Example

1. LD 0xA
2. ShReq 0xA
MSI Directory Protocol Example

1. LD 0xA

2. ShReq 0xA
MSI Directory Protocol Example

1. **LD 0xA**

2. **ShReq 0xA**

3. **Mem[0xA] = 3**

- **Cache 0**
  - Tag: 0xA
  - State: I->S

- **Cache 1**
  - Tag: 0xA
  - State: 
  - Data: 

- **Cache 2**
  - Tag: 
  - State: 
  - Data: 

- **Directory**
  - Tag: 0xA
  - State: Sh
  - Sharers: \{0\}

- **Main Memory**
MSI Directory Protocol Example

1. LD 0xA
2. ShReq 0xA
3. Mem[0xA] = 3
4. ShResp 0xA, data=3
MSI Directory Protocol Example

1. LD 0xA
2. ShReq 0xA
3. Mem[0xA] = 3
4. ShResp 0xA, data=3
MSI Directory Protocol Example

### Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0}</td>
</tr>
</tbody>
</table>

### Main Memory

**Core 0**

**Cache 0**

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

**Core 1**

**Cache 1**

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Core 2**

**Cache 2**

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**MSI Directory Protocol Example**

### Main Memory

### Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0}</td>
</tr>
</tbody>
</table>

### Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

### Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. LD 0xA

---

April 4, 2016

http://www.csg.csail.mit.edu/6.823

Sanchez & Emer
MSI Directory Protocol Example

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0}</td>
</tr>
</tbody>
</table>

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;S</td>
<td></td>
</tr>
</tbody>
</table>

Core 0

Core 1

Core 2

1 LD 0xA
MSI Directory Protocol Example

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0}</td>
</tr>
</tbody>
</table>

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
</table>

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;S</td>
<td></td>
</tr>
</tbody>
</table>

Core 0

Core 1

Core 2

LD 0xA

ShReq 0xA

April 4, 2016

http://www.csg.csail.mit.edu/6.823

Sanchez & Emer
MSI Directory Protocol Example

Core 0
Cache 0
- Tag: 0xA
- State: S
- Data: 3

Core 1
Cache 1
- Tag: 0xA
- State: I->S

Core 2
Cache 2
- Tag: 0xA
- State: I->S

Main Memory

Directory
- Tag: 0xA
- State: Sh
- Sharers: {0,2}

ShReq 0xA

LD 0xA
MSI Directory Protocol Example

1. LD 0xA
2. ShReq 0xA
3. Mem[0xA] = 3
MSI Directory Protocol Example

1. LD 0xA
2. ShReq 0xA
3. Mem[0xA] = 3
4. ShResp 0xA, data=3
MSI Directory Protocol Example

1. LD 0xA
2. ShReq 0xA
3. Mem[0xA] = 3
4. ShResp 0xA, data=3
# MSI Directory Protocol Example

## Main Memory

## Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0,2}</td>
</tr>
</tbody>
</table>

## Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

## Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;M</td>
<td></td>
</tr>
</tbody>
</table>

## Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

1. ST 0xA

---

April 4, 2016

http://www.csg.csail.mit.edu/6.823

Sanchez & Emer
MSI Directory Protocol Example

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0,2}</td>
</tr>
</tbody>
</table>

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;M</td>
<td></td>
</tr>
</tbody>
</table>

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

Core 0

Core 1

Core 2

ExReq 0xA

ST 0xA
MSI Directory Protocol Example

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0,2}</td>
</tr>
</tbody>
</table>

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;M</td>
<td></td>
</tr>
</tbody>
</table>

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

Core 0

Core 1

Core 2

1 ST 0xA

2 ExReq 0xA

3 InvReq 0xA

3 InvReq 0xA

April 4, 2016

http://www.csg.csail.mit.edu/6.823
MSI Directory Protocol Example

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0,2}</td>
</tr>
</tbody>
</table>

1. ST 0xA
2. ExReq 0xA
3. InvReq 0xA

Core 0
- Cache 0:
  - Tag: 0xA
  - State: I
  - Data: 3

Core 1
- Cache 1:
  - Tag: 0xA
  - State: I->M
  - Data

Core 2
- Cache 2:
  - Tag: 0xA
  - State: I
  - Data: 3
MSI Directory Protocol Example

Core 0

Cache 0
- Tag: 0xA
- State: I
- Data: 3

Main Memory

Directory
- Tag: 0xA
- State: Sh
- Sharers: \{0, 2\}

Cache 1
- Tag: 0xA
- State: I-\to M

Core 1

Cache 2
- Tag: 0xA
- State: I
- Data: 3

Core 2

1. ST 0xA
2. ExReq 0xA
3. InvReq 0xA
4. InvResp 0xA
3. InvReq 0xA
4. InvResp 0xA
3. InvReq 0xA
4. InvResp 0xA

April 4, 2016

http://www.csg.csail.mit.edu/6.823

Sanchez & Emer
MSI Directory Protocol Example

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;M</td>
<td></td>
</tr>
</tbody>
</table>

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

Core 0

Core 1

Core 2

ST 0xA

InvReq 0xA

ExReq 0xA

InvReq 0xA

InvResp 0xA

InvResp 0xA

April 4, 2016

http://www.csg.csail.mit.edu/6.823

Sanchez & Emer
MSI Directory Protocol Example

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>

Main Memory

Directory

Mem[0xA] = 3

Core 0

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

Core 1

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;M</td>
<td></td>
</tr>
</tbody>
</table>

Core 2

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

1. ST 0xA
2. ExReq 0xA
3. InvReq 0xA
4. InvResp 0xA
5. Mem[0xA] = 3

April 4, 2016

http://www.csg.csail.mit.edu/6.823

Sanchez & Emer
MSI Directory Protocol Example

<table>
<thead>
<tr>
<th>Core 0</th>
<th>Cache 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>State</td>
</tr>
<tr>
<td>0xA</td>
<td>I</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Core 1</th>
<th>Cache 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>State</td>
</tr>
<tr>
<td>0xA</td>
<td>I-&gt;M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Core 2</th>
<th>Cache 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>State</td>
</tr>
<tr>
<td>0xA</td>
<td>I</td>
</tr>
</tbody>
</table>

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>

Main Memory

Mem[0xA] = 3

1. ST 0xA
2. ExReq 0xA
3. InvReq 0xA
4. InvResp 0xA
5. InvResp 0xA
6. ExResp 0xA, data = 3
### MSI Directory Protocol Example

#### Directory Table

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>

#### Memory Access

1. **ST 0xA**

2. **ExReq 0xA**
   - **Cache 1**: Tag 0xA, State = M, Data = 5
   - **Cache 2**: Tag 0xA, State = I, Data = 3

3. **InvReq 0xA**
   - **Cache 0**: Tag 0xA, State = I, Data = 3
   - **Cache 1**: Tag 0xA, State = M, Data = 5

4. **InvResp 0xA**
   - **Cache 0**: Tag 0xA, State = I, Data = 3
   - **Cache 1**: Tag 0xA, State = M, Data = 5

5. **Mem[0xA] = 3**

6. **ExResp 0xA**
   - **Data = 3**
MSI Directory Protocol Example

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>M-&gt;I</td>
<td>5</td>
</tr>
</tbody>
</table>

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

Core 0

1 ST 0xB

Core 1

Core 2
MSI Directory Protocol Example

1. ST 0xB

2. WbReq 0xA, data=5
### MSI Directory Protocol Example

#### Core 0
**Cache 0**
- **Tag**: 0xA
- **State**: I
- **Data**: 3

#### Core 1
**Cache 1**
- **Tag**: 0xA
- **State**: M->I
- **Data**: 5

- **WbReq 0xA, data=5**

#### Core 2
**Cache 2**
- **Tag**: 0xA
- **State**: I
- **Data**: 3

- **ST 0xB**

- **Mem[0xA] = 5**

---

April 4, 2016

http://www.csg.csail.mit.edu/6.823

Sanchez & Emer
 MSI Directory Protocol Example

Core 0

Cache 0

Tag  |  State |  Data |
-----|--------|-------|
0xA  | I      | 3     |

Core 1

Cache 1

Tag  |  State |  Data |
-----|--------|-------|
0xA  | M->I   | 5     |

Core 2

Cache 2

Tag  |  State |  Data |
-----|--------|-------|
0xA  | I      | 3     |

Main Memory

Directory

Tag  |  State |  Sharers |
-----|--------|----------|
0xA  | Un     | {}       |

Mem[0xA] = 5

WbReq 0xA, data=5

ST 0xB
MSI Directory Protocol Example

1. ST 0xB
2. WbReq 0xA, data=5
3. Mem[0xA] = 5
4. WbResp 0xA

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Un</td>
<td>{}</td>
</tr>
</tbody>
</table>

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>M→I</td>
<td>5</td>
</tr>
</tbody>
</table>

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

Core 0

Core 1

Core 2

April 4, 2016

http://www.csg.csail.mit.edu/6.823

Sanchez & Emer
MSI Directory Protocol Example

1. ST 0xB

2. WbReq 0xA, data=5

3. Mem[0xA] = 5

4. WbResp 0xA
MSI Directory Protocol Example

1. ST 0xB
2. WbReq 0xA, data=5
3. Mem[0xA] = 5
4. WbResp 0xA
5. ExReq 0xB
MSI Directory Protocol Example

1. ST 0xB
2. WbReq 0xA, data=5
3. Mem[0xA] = 5
4. WbResp 0xA
5. ExReq 0xB
MSI Directory Protocol Example

1. ST 0xB
2. WbReq 0xA, data=5
3. Mem[0xA] = 5
4. WbResp 0xA
5. ExReq 0xB
6. Mem[0xB] = 10

Cache 0
- Tag: 0xA
- State: I
- Data: 3

Cache 1
- Tag: 0xB
- State: I->M

Cache 2
- Tag: 0xA
- State: I
- Data: 3

Directory
- Tag: 0xB
- State: Ex
- Sharers: {1}

Main Memory
- Mem[0xA] = 5
- Mem[0xB] = 10

Core 0
- Tag: 0xA
- State: I
- Data: 3

Core 1
- Tag: 0xB
- State: I->M

Core 2
- Tag: 0xA
- State: I
- Data: 3
MSI Directory Protocol Example

1. ST 0xB
2. WbReq 0xA, data=5
3. Mem[0xA] = 5
4. WbResp 0xA
5. ExReq 0xB
6. Mem[0xB] = 10
7. ExResp 0xB, data=10

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>

Main Memory

Core 0

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

Core 1

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB</td>
<td>I-&gt;M</td>
<td></td>
</tr>
</tbody>
</table>

Core 2

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>
MSI Directory Protocol Example

1. ST 0xB
2. WbReq 0xA, data=5
3. Mem[0xA] = 5
4. WbResp 0xA
5. ExReq 0xB
6. Mem[0xB] = 10
7. ExResp 0xB, data=10
MSI Directory Protocol Example

Why are 0xA’s wb and 0xB’s req serialized?
MSI Directory Protocol Example

Why are 0xA’s wb and 0xB’s req serialized?

Structural dependence
MSI Directory Protocol Example

Why are 0xA’s wb and 0xB’s req serialized? Structural dependence
Possible solutions?
MSI Directory Protocol Example

Why are 0xA’s wb and 0xB’s req serialized? **Structural dependence**

Possible solutions? **Buffer outside of cache to hold write data**

---

**Figure Details:**

1. **ST 0xB**
2. **WbReq 0xA, data=5**
3. **Mem[0xA] = 5**
4. **WbResp 0xA**
5. **ExReq 0xB**
6. **Mem[0xB] = 10**
7. **ExResp 0xB, data=10**
Miss Status Handling Register

MSHR – Holds load misses and writes outside of cache

MSHR entry

V | X | Addr | Data

- On eviction/writeback
  - No free MSHR entry: stall
  - Allocate new MSHR entry
  - When channel available send WBReq and data
  - Deallocate entry on WBResp
Miss Status Handling Register

MSHR – Holds load misses and writes outside of cache

<table>
<thead>
<tr>
<th>V</th>
<th>X</th>
<th>Addr</th>
<th>Data</th>
</tr>
</thead>
</table>

| L/S | Inum | Block Offset |

- On cache load miss
  - No free MSHR entry: stall
  - Allocate new MSHR entry
  - Send ShReq (or ExReq)
  - On *Resp forward data to CPU and cache
  - Deallocate MSHR
Miss Status Handling Register

MSHR – Holds load misses and writes outside of cache

- On cache load miss
  - Look for matching address is MSHR
    - If not found
      - If no free MSHR entry: stall
      - Allocate new MSHR entry and fill in
    - If found, just fill in per ld/st slot
  - Send ShReq (or ExReq)
  - On *Resp forward data to CPU and cache
  - Deallocate MSHR

Per ld/st slots allow servicing multiple requests with one entry
Directory Organization

• Requirement: Directory needs to keep track of all the cores that are sharing a cache block.

• Challenge: For each block the space needed to hold the list of sharers grows with number of possible sharers...
Flat, Memory-based Directories

- Dedicate a few bits of main memory to store the state and sharers of every line
- Encode sharers using a bit-vector

![Diagram showing main memory, state, and sharer set]

- Simple
- Slow
- Very inefficient with many processors (~P bits / cache line)
Sparse Full-Map Directories

- Not every line in the system needs to be tracked – only those in private caches!
- Idea: Organize directory as a cache

<table>
<thead>
<tr>
<th>Way 1</th>
<th>Way 2</th>
<th>Way 3</th>
<th>Way 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Directory Entry Format

- **Line Address**: 0xF00
- **State**: Sh
- **Sharer Set**: 01001100

- ✓ Low latency, energy-efficient
- ✗ Bit-vectors grow with # cores → Area scales poorly
- ✗ Limited associativity → Directory-induced invalidations
Directory-Induced Invalidations

- To retain inclusion, must invalidate all sharers of an entry before reusing it for another address
- Example: 2-way set-associative sparse directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0}</td>
<td>0xF</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>

Main Memory

Directory

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF</td>
<td>M</td>
<td>1</td>
</tr>
</tbody>
</table>

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Directory-Induced Invalidations

- To retain inclusion, must invalidate all sharers of an entry before reusing it for another address
- Example: 2-way set-associative sparse directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0}</td>
<td>0xF</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>

Core 0
- Tag 0xA, State S, Data 3

Core 1
- Tag 0xF, State M, Data 1

Core 2
- LD 0xB
Directory-Induced Invalidations

- To retain inclusion, must invalidate all sharers of an entry before reusing it for another address
- Example: 2-way set-associative sparse directory

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0}</td>
<td>0xF</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF</td>
<td>M</td>
<td>1</td>
</tr>
</tbody>
</table>

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB</td>
<td>I-&gt;S</td>
<td></td>
</tr>
</tbody>
</table>

Core 2

1 LD 0xB

Core 1

Core 0
Directory-Induced Invalidations

- To retain inclusion, must invalidate all sharers of an entry before reusing it for another address
- Example: 2-way set-associative sparse directory

![Diagram of directory and caches](http://www.csg.csail.mit.edu/6.823)
Directory-Induced Invalidations

- To retain inclusion, must invalidate all sharers of an entry before reusing it for another address
- Example: 2-way set-associative sparse directory
Directory-Induced Invalidations

- To retain inclusion, must invalidate all sharers of an entry before reusing it for another address
- Example: 2-way set-associative sparse directory

```
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0}</td>
</tr>
<tr>
<td>0xF</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>
```

Core 0
- InvReq 0xA

Cache 0
- Tag: 0xA
- State: I
- Data: 3

Core 1
- ShReq 0xB

Cache 1
- Tag: 0xF
- State: M
- Data: 1

Core 2
- LD 0xB

Cache 2
- Tag: 0xB
- State: I->S
- Data: 3

Main Memory

April 4, 2016

http://www.csg.csail.mit.edu/6.823
Directory-Induced Invalidations

- To retain inclusion, must invalidate all sharers of an entry before reusing it for another address
- Example: 2-way set-associative sparse directory

```
Core 0
Cache 0
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

Core 1
Cache 1
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF</td>
<td>M</td>
<td>1</td>
</tr>
</tbody>
</table>

Core 2
Cache 2
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB</td>
<td>I-&gt;S</td>
<td></td>
</tr>
</tbody>
</table>
```

```
Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0}</td>
<td>0xF</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>

Core 0
Cache 0
LD 0xB

Core 1
Cache 1
InvReq 0xA
InvResp 0xA

Core 2
Cache 2
ShReq 0xB

April 4, 2016
http://www.csg.csail.mit.edu/6.823
Sanchez & Emer
### Directory-Induced Invalidations

- To retain inclusion, must invalidate all sharers of an entry before reusing it for another address
- Example: 2-way set-associative sparse directory

#### Main Memory

#### Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB</td>
<td>Sh</td>
<td>{2}</td>
</tr>
<tr>
<td>0xF</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>

#### Core 0

- Cache 0
  - Tag: 0xA
  - State: I
  - Data: 3

#### Core 1

- Cache 1
  - Tag: 0xF
  - State: M
  - Data: 1

#### Core 2

- Cache 2
  - Tag: 0xB
  - State: I→S

---

**Events:**

1. **LD 0xB**
2. **ShReq 0xB**
3. **InvReq 0xA**
4. **InvResp 0xA**
Directory-Induced Invalidations

- To retain inclusion, must invalidate all sharers of an entry before reusing it for another address
- Example: 2-way set-associative sparse directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB</td>
<td>Sh</td>
<td>{2}</td>
</tr>
<tr>
<td>0xF</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>

1. LD 0xB
2. ShReq 0xB
3. InvReq 0xA
4. InvResp 0xA
5. Mem[0xB] = 5
Directory-Induced Invalidations

- To retain inclusion, must invalidate all sharers of an entry before reusing it for another address
- Example: 2-way set-associative sparse directory

<table>
<thead>
<tr>
<th>Core 0</th>
<th>Core 1</th>
<th>Core 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache 0</td>
<td>Cache 1</td>
<td>Cache 2</td>
</tr>
<tr>
<td>SD 0xB</td>
<td>SD 0xF</td>
<td>SD 0xB</td>
</tr>
<tr>
<td>State</td>
<td>State</td>
<td>State</td>
</tr>
<tr>
<td>I</td>
<td>M</td>
<td>I-&gt;S</td>
</tr>
<tr>
<td>Data</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Mem[0xB]</td>
</tr>
</tbody>
</table>

Mem[0xB] = 5

1. LD 0xB
2. ShReq 0xB
3. InvReq 0xA
4. InvResp 0xA
5. ShResp 0xB, data=5
Directory-Induced Invalidations

- To retain inclusion, must invalidate all sharers of an entry before reusing it for another address
- Example: 2-way set-associative sparse directory

Main Memory

```
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB</td>
<td>Sh</td>
<td>{2}</td>
</tr>
</tbody>
</table>
```

Directory

```
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB</td>
<td>Sh</td>
<td>{2}</td>
</tr>
<tr>
<td>0xF</td>
<td>Ex</td>
<td>{1}</td>
</tr>
</tbody>
</table>
```

Cache 0

```
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>
```

Cache 1

```
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF</td>
<td>M</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Cache 2

```
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB</td>
<td>S</td>
<td>5</td>
</tr>
</tbody>
</table>
```

Core 0

Core 1

Core 2

1. LD 0xB
2. ShReq 0xB
3. InvReq 0xA
4. InvResp 0xA
5. Mem[0xB] = 5
6. ShResp 0xB, data=5

How many entries should the directory have?

http://www.csg.csail.mit.edu/6.823

Sanchez & Emer
Inexact Representations of Sharer Sets

- Coarse-grain bit-vectors (e.g., 1 bit per 4 cores)
  
  Sharer Set
  
  \[
  \begin{array}{ccccccc}
  & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0-3 & 4-7 & 8-11 & 12-15 & 16-19 & 20-23 \\
  \end{array}
  \]

- Limited pointers: Maintain a few sharer pointers, on overflow mark ‘all’ and broadcast or invalidate
  
  Sharer Set
  
  \[
  \begin{array}{cccc}
  & 0 & 8 & 14 & 33 \\
  all & sharer 1 & sharer 2 & sharer 3 \\
  \end{array}
  \]

- Allow false positives (e.g., Bloom filters)
  
  ✓ Reduced area & energy
  × Overheads still not scalable (these techniques simply play with constant factors)
  × Inexact sharers → Broadcasts, invalidations or spurious invalidations and downgrades
Protocol Races

- Directory serializes multiple requests for the same address
  - Same-address requests are queued or NACKed and retried
- But races still exist due to conflicting requests
- Example: Upgrade race

![Diagram of main memory and directories](http://www.csg.csail.mit.edu/6.823)
Protocol Races

- Directory serializes multiple requests for the same address
  - Same-address requests are queued or NACKed and retried
- But races still exist due to conflicting requests
- Example: Upgrade race

```
Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0,2}</td>
</tr>
</tbody>
</table>

ReqQ
```

```
Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

Core 0

1 ST 0xA

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

Core 1

1’ ST 0xA
```
Protocol Races

- Directory serializes multiple requests for the same address
  - Same-address requests are queued or NACKed and retried
- But races still exist due to conflicting requests
- Example: Upgrade race

<table>
<thead>
<tr>
<th>ReqQ</th>
<th>Directory</th>
<th>Core 0</th>
<th>Core 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tag</td>
<td>State</td>
<td>Sharers</td>
</tr>
<tr>
<td></td>
<td>0xA</td>
<td>Sh</td>
<td>{0,2}</td>
</tr>
</tbody>
</table>

Main Memory

Cache 0

- Tag: 0xA
- State: S->M
- Data: 3

Cache 1

- Tag: 0xA
- State: S->M
- Data: 3

Core 0

ST 0xA

Core 1

ST 0xA
Protocol Races

- Directory serializes multiple requests for the same address
  - Same-address requests are queued or NACKed and retried
- But races still exist due to conflicting requests
- Example: Upgrade race

Caches 0 and 1 issue simultaneous ExReqs

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Sh</td>
<td>{0,2}</td>
</tr>
</tbody>
</table>

ExReq 0xA

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S-&gt;M</td>
<td>3</td>
</tr>
</tbody>
</table>

ST 0xA

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>S-&gt;M</td>
<td>3</td>
</tr>
</tbody>
</table>

ST 0xA

Core 0

Cache 0

Cache 1
Protocol Races

- Directory serializes multiple requests for the same address
  - Same-address requests are queued or NACKed and retried
- But races still exist due to conflicting requests
- Example: Upgrade race

Caches 0 and 1 issue simultaneous ExReqs
Directory starts serving cache 0’s ExReq, queues cache 1’s

April 4, 2016

http://www.csg.csail.mit.edu/6.823
Protocol Races

- Directory serializes multiple requests for the same address
  - Same-address requests are queued or NACKed and retried
- But races still exist due to conflicting requests
- Example: Upgrade race

Caches 0 and 1 issue simultaneous ExReqs
Directory starts serving cache 0’s ExReq, queues cache 1’s

Cache 1 expected ExResp, but got InvReq!
Protocol Races

- Directory serializes multiple requests for the same address
  - Same-address requests are queued or NACKed and retried
- But races still exist due to conflicting requests
- Example: Upgrade race

Caches 0 and 1 issue simultaneous ExReqs
Directory starts serving cache 0’s ExReq, queues cache 1’s

Cache 1 expected ExResp, but got InvReq!
Cache 1 should transition from S->M to I->M and send InvResp
## CC and False Sharing

### Performance Issue - 1

<table>
<thead>
<tr>
<th>state</th>
<th>blk addr</th>
<th>data0</th>
<th>data1</th>
<th>...</th>
<th>dataN</th>
</tr>
</thead>
</table>

A cache block contains more than one word and cache-coherence is done at the block-level and not word-level.

Suppose $P_1$ writes word $i$ and $P_2$ writes word $k$ and both words have the same block address.

**What can happen?** The block may be invalidated (ping pong) many times unnecessarily because the addresses are in same block.
Cache-coherence protocols will cause mutex to ping-pong between P1’s and P2’s caches.

Ping-ponging can be reduced by first reading the mutex location (*non-atomically*) and executing a swap only if it is found to be zero.
CC and Bus Occupancy
Performance Issue - 3

In general, an atomic read-modify-write instruction requires two memory (bus) operations without intervening memory operations by other processors.

In a multiprocessor setting, bus needs to be locked for the entire duration of the atomic read and write operation:

- expensive for simple buses
- very expensive for split-transaction buses

Modern processors use:

- load-reserve
- store-conditional
Load-reserve & Store-conditional

Special register(s) to hold reservation flag and address, and the outcome of store-conditional

Load-reserve R, (a):
<flag, adr> = <1, a>;
R = M[a];

Store-conditional (a), R:
if <flag, adr> == <1, a> then cancel other procs’ reservation on a;
M[a] = <R>;
status = succeed;
else status = fail;

If the snooper sees a store transaction to the address in the reserve register, the reserve bit is set to 0

• Several processors may reserve ‘a’ simultaneously
• These instructions are like ordinary loads and stores with respect to the bus traffic
Load-Reserve/Store-Conditional

Swap implemented with Ld-Reserve/St-Conditional

# Swap(R1, mutex):

R1 <- 1

L: Ld-Reserve R2, (mutex)
   St-Conditional (mutex), R1
   if (status == fail) goto L
   R1 <- R2
Performance:
Load-reserve & Store-conditional

The total number of memory (bus) transactions is not necessarily reduced, but splitting an atomic instruction into load-reserve & store-conditional:

- *increases bus utilization* (and reduces processor stall time), especially in split-transaction buses

- *reduces cache ping-pong effect* because processors trying to acquire a semaphore do not have to perform stores each time
Extra hops: 2-hop Protocols
Performance Issue - 4

- Data in another cache needs to pass through main memory, instead of being forwarded directly.

### Main Memory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex</td>
<td>{0}</td>
</tr>
</tbody>
</table>

### Directory

### Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>M</td>
<td>3</td>
</tr>
</tbody>
</table>

### Cache 1

### Cache 2

Core 0
Core 1
Core 2
Extra hops: 2-hop Protocols
Performance Issue - 4

- Data in another cache needs to pass through main memory, instead of being forwarded directly.

<table>
<thead>
<tr>
<th></th>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Memory</td>
<td>0xA</td>
<td>Ex</td>
<td>{0}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache 0</td>
<td>0xA</td>
<td>M</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache 2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Core 0

Core 1

Core 2

1 ST 0xA
• Data in another cache needs to pass through main memory, instead of being forwarded directly.

### Main Memory

### Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex</td>
<td>{0}</td>
</tr>
</tbody>
</table>

### Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>M</td>
<td>3</td>
</tr>
</tbody>
</table>

### Core 0

### Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;M</td>
<td></td>
</tr>
</tbody>
</table>

### Core 1

### Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;M</td>
<td></td>
</tr>
</tbody>
</table>

### Core 2

1 ST 0xA
Extra hops: 2-hop Protocols

Performance Issue - 4

- Data in another cache needs to pass through main memory, instead of being forwarded directly.

### Main Memory

### Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex</td>
<td>{0}</td>
</tr>
</tbody>
</table>

### Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>M</td>
<td>3</td>
</tr>
</tbody>
</table>

### Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;M</td>
<td></td>
</tr>
</tbody>
</table>

1. **ST 0xA**
2. **ExReq 0xA**
Extra hops: 2-hop Protocols

Performance Issue - 4

- Data in another cache needs to pass through main memory, instead of being forwarded directly.

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex-&gt;Ex</td>
<td>{2}</td>
</tr>
</tbody>
</table>

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>M</td>
<td>3</td>
</tr>
</tbody>
</table>

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
</table>

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;M</td>
<td></td>
</tr>
</tbody>
</table>

Core 0

Core 1

Core 2

April 4, 2016

http://www.csg.csail.mit.edu/6.823

Sanchez & Emer
Extra hops: 2-hop Protocols
Performance Issue - 4

- Data in another cache needs to pass through main memory, instead of being forwarded directly.

Main Memory

```
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex-&gt;Ex</td>
<td>{2}</td>
</tr>
</tbody>
</table>
```

Cache 0
```
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>M</td>
<td>3</td>
</tr>
</tbody>
</table>
```

Cache 1
```
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
</table>
```

Cache 2
```
<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;M</td>
<td></td>
</tr>
</tbody>
</table>
```

Core 0

Core 1

Core 2

4. ExFwd 0xA, req=2
2. ExReq 0xA
1. ST 0xA

April 4, 2016
http://www.csg.csail.mit.edu/6.823
Sanchez & Emer
Extra hops: 2-hop Protocols
Performance Issue - 4

- Data in another cache needs to pass through main memory, instead of being forwarded directly.

### Main Memory

### Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex-&gt;Ex</td>
<td>{2}</td>
</tr>
</tbody>
</table>

1. **ST 0xA**
2. **ExReq 0xA**
3. **ExFwd 0xA, req=2**
Extra hops: 2-hop Protocols
Performance Issue - 4

- Data in another cache needs to pass through main memory, instead of being forwarded directly.

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex-&gt;Ex</td>
<td>{2}</td>
</tr>
</tbody>
</table>

3. ExFwd 0xA, req=2
2. ExReq 0xA
1. ST 0xA

Core 0

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

Core 1

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Core 2

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I-&gt;M</td>
<td></td>
</tr>
</tbody>
</table>

ExResp 0xA, data=3
Extra hops: 2-hop Protocols
Performance Issue - 4

- Data in another cache needs to pass through main memory, instead of being forwarded directly.

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex-&gt;Ex</td>
<td>{2}</td>
</tr>
</tbody>
</table>

ExFwd 0xA, req=2

ExReq 0xA

ExResp 0xA, data=3

ST 0xA
Extra hops: 2-hop Protocols
Performance Issue - 4

- Data in another cache needs to pass through main memory, instead of being forwarded directly.

Main Memory

Directory

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex-&gt;Ex</td>
<td>{2}</td>
</tr>
</tbody>
</table>

3 ExFwd 0xA, req=2
4 ExAck 0xA
2 ExReq 0xA

Cache 0

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

Cache 1

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cache 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>M</td>
<td>3</td>
</tr>
</tbody>
</table>

Core 0

Core 1

Core 2

3 ExResp 0xA, data=3
1 ST 0xA

April 4, 2016

http://www.csg.csail.mit.edu/6.823
Extra hops: 2-hop Protocols
Performance Issue - 4

- Data in another cache needs to pass through main memory, instead of being forwarded directly.

**Main Memory**

**Directory**

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Sharers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>Ex</td>
<td>{2}</td>
</tr>
</tbody>
</table>

**Cache 0**

<table>
<thead>
<tr>
<th>Tag</th>
<th>State</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA</td>
<td>I</td>
<td>3</td>
</tr>
</tbody>
</table>

**Cache 1**

**Cache 2**

**Core 0**

**Core 1**

**Core 2**

**ExFwd 0xA, req=2**

**ExReq 0xA**

**ExAck 0xA**

**ExResp 0xA, data=3**

http://www.csg.csail.mit.edu/6.823
Coherence in Multi-Level Hierarchies

- Can use the same or different protocols to keep coherence across multiple levels
- Key invariant: Ensure sufficient permissions in all intermediate levels
- Example: 8-socket Xeon E7 (8 cores/socket)

![Diagram of memory hierarchy with interconnect and protocols like MESIF and MESI]
In-Cache Directories

- Common multicore memory hierarchy:
  - 1+ levels of private caches
  - A shared last-level cache
  - Need to enforce coherence among private caches

- Idea: Embed the directory information in shared cache tags
  - Shared cache must be inclusive

✓ Avoids tag overheads & separate lookups
× Can be inefficient if shared cache size >> \( \sum(\text{private cache sizes}) \)
Avoiding Protocol Deadlock

- Protocols can cause deadlocks even if network is deadlock-free!

Example: Both nodes saturate all intermediate buffers with requests to each other, blocking responses from entering the network.

- Solution: Separate virtual networks
  - Different sets of virtual channels and endpoint buffers
  - Same physical routers and links

- Most protocols require at least 2 virtual networks (for requests and replies), often >2 needed
Next Lecture:
Consistency and
Relaxed Memory Models