Memory Coherence in Shared Virtual Memory Systems

K. Li & P. Hudak

Alan Ghobadi 2014-10-07
Distributed Information Processing
Introduction

- Loosely coupled multiprocessors
- Memory Coherence Problem
- Shared Virtual Memory

(K. Li and P. Hudak)
Concepts

- Memory Coherence
- Virtual Memory
- Page Faults
“A memory is coherent if the value returned by a read operation is always the same as the value written by the most recent write operation to the same address.”

– K. Li & P. Hudak
Virtual Memory

http://en.wikipedia.org/wiki/Virtual_memory
Virtual Memory

• Page Read Faults
• Page Write Faults
• Page Fault Handler in the OS
Methods

• Central algorithm

• Distributed algorithm
Central algorithm

P2 Read Fault

P1, Central Manager

Ptable

<table>
<thead>
<tr>
<th>Access</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td>0</td>
</tr>
</tbody>
</table>

Ptable

<table>
<thead>
<tr>
<th>Access</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td>0</td>
</tr>
</tbody>
</table>

Info

<table>
<thead>
<tr>
<th>Owner</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy set</td>
<td>{P1}</td>
</tr>
<tr>
<td>Lock</td>
<td>0</td>
</tr>
</tbody>
</table>
Central algorithm

P2 Read Fault

P1, Central Manager

P2

1. Request

P3

Info
Owner | P3
Copy set | {P1}
Lock | 0

Ptable
Access | 1
Lock | 0

Ptable
Access | Read
Lock | 0
Central algorithm

P2 Read Fault

P1, Central Manager

Ptable
Access
Lock 1

Ptable
Access
Read
Lock 0

Info
Owner P3
Copy set {P1, P2}
Lock 1

P2
1. Request

P3
Central algorithm

P2 Read Fault

P1, Central Manager

Ptable
Access  |  Lock
-------|-------
1      | 1

Ptable
Access  |  Read
-------|------
0      | 0

Info
Owner  |  P3
Copy set  |  {P1,P2}
Lock    |  1

1. Request

2. Request: Send copy to P2
Central algorithm

P2 Read Fault

1. Request

2. Request: Send copy to P2
Central algorithm

P2 Read Fault

P1, Central Manager

1. Request

Info
Owner: P3
Copy set: {P1, P2}
Lock: 1

P2

2. Request: Send copy to P2

P3

3. Send Copy

Ptable
Access: Read
Lock: 1
Central algorithm

P2 Read Fault

P1, Central Manager

Ptable

Access | Lock
--- | ---
1

Ptable

Access | Read
--- | ---
0

Info

<table>
<thead>
<tr>
<th>Owner</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy set</td>
<td>{P1,P2}</td>
</tr>
<tr>
<td>Lock</td>
<td>1</td>
</tr>
</tbody>
</table>

1. Request
2. Request: Send copy to P2
3. Send Copy
4. Confirmation
Central algorithm

P2 Read Fault

P1, Central Manager

P2

1. Request

2. Request: Send copy to P2

3. Send Copy

4. Confirmation

P3

Info

Owner: P3
Copy set: {P1,P2}
Lock: 0

Ptable

Access: Read
Lock: 0
Central algorithm

P2 Write Fault

P1, Central Manager

P2

1. Write Request

P3

Ptable

<table>
<thead>
<tr>
<th>Access</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td>1</td>
</tr>
</tbody>
</table>

Ptable

<table>
<thead>
<tr>
<th>Access</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td>0</td>
</tr>
</tbody>
</table>

Info

<table>
<thead>
<tr>
<th>Owner</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy set</td>
<td>{P1,P2}</td>
</tr>
<tr>
<td>Lock</td>
<td>0</td>
</tr>
</tbody>
</table>
Central algorithm

P2 Write Fault

P1, Central Manager

Ptable

Access
Lock 1

Ptable
Access Read
Lock 0

Info
Owner P2
Copy set {P1,P2}
Lock 1

P2 Write Fault

1. Write Request
2. Invalidation Request
Central algorithm

P2 Write Fault

P1, Central Manager

P2

1. Write Request

2. Invalidation Request

3. Request: Send page to P3

P3

Info

<table>
<thead>
<tr>
<th>Owner</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy set</td>
<td>{}</td>
</tr>
<tr>
<td>Lock</td>
<td>1</td>
</tr>
</tbody>
</table>

Ptable

<table>
<thead>
<tr>
<th>Access</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td>0</td>
</tr>
</tbody>
</table>

Ptable

<table>
<thead>
<tr>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
</tr>
</tbody>
</table>
Central algorithm

P2 Write Fault

P1, Central Manager

P2 Write Fault

1. Write Request

2. Invalidation Request

3. Request: Send page to P3

4. Send Page

Info

Owner: P2
Copy set: {}
Lock: 1

Ptable

Access

Lock: 1

Ptable

Access

Lock: 1
Central algorithm

P2 Write Fault

P1, Central Manager

1. Write Request
2. Invalidation Request
5. Confirmation
3. Request: Send page to P3
4. Send Page

Ptable
Access | Write
Lock   | 0

Info
Owner  | P2
Copy set | {}
Lock   | 0

P2

P3
Central algorithm

- Easy to implement
- Lots of message traffic
- Synchronisation Lock
- Improvement: Move synchronization to processors
Broadcast Distributed Manager Algorithm
Broadcast Distributed Manager Algorithm

P2 Read Fault

1. Broadcast Read Request

P1

Ptable
Access: Read
Lock: Read
Copy set: {}
Owner: P3

P2

P3

Ptable
Access: Read
Lock: Read
Copy set: {P1}
Owner: P3
Broadcast Distributed Manager Algorithm

**P2 Read Fault**

1. Broadcast Read Request

   - **Ptable**
     - Access: Read
     - Lock: Read
     - Copy set: {}
     - Owner: P3

2. Send copy

   - **Ptable**
     - Access: Read
     - Lock: Read
     - Copy set: {P1, P2}
     - Owner: P3
Broadcast Distributed Manager Algorithm

- Broadcast based
- Fixed number of pages spread out across processors
- Atomic operations
- Message received by all processors
- No Confirmations
Dynamic Distributed Algorithm
Dynamic Distributed Algorithm

P2 Read Fault

1. Read Request

Ptable:

<table>
<thead>
<tr>
<th>Access</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td></td>
</tr>
<tr>
<td>Copy set</td>
<td>{P1}</td>
</tr>
<tr>
<td>ProbOwner</td>
<td>P3</td>
</tr>
</tbody>
</table>
Dynamic Distributed Algorithm

P2 Read Fault

1. Read Request

P1

Access
Lock
Copy set {P1}
ProbOwner P3

P2

2. Forward Request

P3

Ptable
Access Read
Lock
Copy set {P1}
ProbOwner P3
Dynamic Distributed Algorithm

P2 Read Fault

1. Read Request

P1

Copy set: \{P1, P3\}
ProbOwner: P2

P2

2. Forward Request

P3

Copy set: \{P1, P3\}
ProbOwner: P2

3. Send Copy

Ptable

<table>
<thead>
<tr>
<th>Access</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td></td>
</tr>
<tr>
<td>Copy set</td>
<td>{P1}</td>
</tr>
<tr>
<td>ProbOwner</td>
<td>P2</td>
</tr>
</tbody>
</table>

Ptable

<table>
<thead>
<tr>
<th>Access</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td></td>
</tr>
<tr>
<td>Copy set</td>
<td>{P1, P3}</td>
</tr>
<tr>
<td>ProbOwner</td>
<td>P2</td>
</tr>
</tbody>
</table>
Dynamic Distributed Algorithm

P1 Write Fault

1. Write Request
Dynamic Distributed Algorithm

P1 Write Fault

1. Write Request

2. Send Copy set
Dynamic Distributed Algorithm

P1 Write Fault

1. Write Request
2. Send Copy set
3. Invalidate Copy Set

Ptable

<table>
<thead>
<tr>
<th>Access</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td></td>
</tr>
<tr>
<td>Copy set</td>
<td>{}</td>
</tr>
<tr>
<td>ProbOwner</td>
<td>P2</td>
</tr>
</tbody>
</table>

P2

<table>
<thead>
<tr>
<th>Access</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td></td>
</tr>
<tr>
<td>Copy set</td>
<td>{}</td>
</tr>
<tr>
<td>ProbOwner</td>
<td>P1</td>
</tr>
</tbody>
</table>

P3

<table>
<thead>
<tr>
<th>Access</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td></td>
</tr>
<tr>
<td>Copy set</td>
<td>{}</td>
</tr>
<tr>
<td>ProbOwner</td>
<td>P1</td>
</tr>
</tbody>
</table>
Dynamic Distributed Algorithm

P1 Write Fault

1. Write Request
2. Send Copy set
3. Invalidate Copy Set

Ptable

<table>
<thead>
<tr>
<th>Access</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td></td>
</tr>
<tr>
<td>Copy set</td>
<td>{}</td>
</tr>
<tr>
<td>ProbOwner</td>
<td>P1</td>
</tr>
</tbody>
</table>

Ptable

<table>
<thead>
<tr>
<th>Access</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td></td>
</tr>
<tr>
<td>Copy set</td>
<td>{}</td>
</tr>
<tr>
<td>ProbOwner</td>
<td>P1</td>
</tr>
</tbody>
</table>
Dynamic Distributed Algorithm

- No fixed owner
- Follows the ProbOwner field (forwarding)
- Falls back on broadcast if path to owner is too long
Conclusion

With experiments indicated that "many parallel programs exhibit good speedups on loosely coupled multiprocessors using a shared virtual memory."

- Central Manager - traffic bottleneck
- Fixed Distributed Manager - Less traffic bottleneck but still many messages
- Dynamic Distributed Manager - Best overall performance