A Systematic Methodology to Develop Resilient Cache Coherence Protocols

Konstantinos Aispos    Li-Shiuan Peh

Presented by: Helen Arefayne  Milind Furia
What is cache coherence?

- Loading the correct value when the same data is stored in multiple caches
Types of cache coherence

- Directory-based
- Broadcast-based
- Snooping
Problem

Unreliable interconnect  Suspended transaction
Cause

- Transient faults
Solution

- Extend coherence protocols for resilience
  - Detect deadlocks
  - Retransmit lost messages
Related work

- Checkpointing [Prvulovic et al., Sorin, et al.]
  - Pro-active

- FTDirCMP [Pascual et al.]
  - Protocol-specific
Characters of a resilient protocol

- **Property 1**
  - All initiators of transactions stay in transient state until all state go to stable state

- **Property 2**
  - Previously transmitted messages can be retransmitted

- **Property 3**
  - All nodes can tolerate duplicate messages and produce same outcome
Directory based coherence - A

request
DATA
unblock

Memory
mem_req
DATA

requestor
IS
DATA
unblock

S
E

directory
request
NP
memory request
mb
DATA
DATA
b
unblock
S v[1000]

a) data: NP/S, request: S
Enforcing property 1

- Data: NP/S, Request: S

**request**

- **IS**
- **Sd**
- **Ed**

**mem_req**

- **DATA**
- **unblock**
- **done**

**Memory**

- **requestor**

- **directory**

- **request**
- **NP**
- **mb**
- **b**
- **Sv[1000]**
- **unblock**
- **done**

a) data: NP/S, request: S
Directory based coherence - B

b) data: M, request: M/S

requestor

exclusive sharer

request

unblock

DATA
Enforcing property 2

b) data: M, request: M/S
Directory based coherence - C

c) data: S, request: M

- Request
- Invalidate
- Unblock
- Ack count
- Ack
- S, request: M
- IM, a=1
- IM, a=2
- M
- S
- IM, a=1
- IM, a=2
- M

Requestor

Sharer

Invalidation

Directory
Invalidate 1000,0100,0001

request

unblock

Invalidate

expected acks

ack

ack

S

request

expected acks

unblock

done

I

requestor

ack [1000]

ack [1101]

IM, a[1000]

IM, a[1001]

IM

M

Md

done

unblock

S

c) data: S, request: M

directory

S

I

I

S

Invalidate

ack [id]

sharer

Enforcing property 3
Experimental setup

- Wisconsin Multifacet GEMS simulator
- 64-core tiled CMP
- Private split L1 caches
- Physically distributed shared L2 cache
- Fault rates of 1 fault/ms – 1 fault/µs
Performance overhead

Large working sets’ traffic saturate NoC

Higher fault rate → Higher execution overhead

More exclusive requests

Low overhead when no faults

SPLASH Benchmark

Average overhead all benchmarks
Network congestion

- Baseline protocol, no faults
- Resilient protocol, no faults
- Resilient protocol, 1 fault/10µs

8% increase!

Most congested link

Average over all links
Hardware overhead

- **MSHR table**

<table>
<thead>
<tr>
<th>PC</th>
<th>Req.</th>
<th>....</th>
<th>State</th>
<th>Transaction ID</th>
<th>Sender bitvector</th>
<th>Timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \rightarrow 1 \text{ bit} \leftrightarrow 6 \text{ bits} \leftrightarrow 64 \text{ bits} \rightarrow \leftrightarrow 13 \text{ bits} \rightarrow \]

- **Router**
  - Adder per buffered packet

Total = 352 bytes/node, 20 X 16 bit adder/router << core gate count
Conclusion

- Lost messages lead to suspended transactions.
- Three properties were defined that guarantee transactions will eventually complete.
- Experimental results indicated negligible hardware overhead and execution overhead of 0.8% during fault-free operation.
Questions?
Discussion

- Does addressing only transient faults guarantee sufficient resilience?

- The resilient version of the protocol is much more elaborate than the baseline. Is this worth it?
THANK YOU!