Proximity-Aware Directory-based Coherence for Multi-core Processor Architectures

Jeff Brown
Rakesh Kumar
Dean Tullsen

UC San Diego • University of Illinois at Urbana-Champaign
SPAA 19 • June 9, 2007
Introduction

- The chip multiprocessor (CMP) era is upon us!
- Caching complicate writes
- **Cache Coherence** ensures caching is done safely
- Multi-core designs offer new tradeoffs
Introduction

- The chip multiprocessor (CMP) era is upon us!
- Caching complicate writes
- **Cache Coherence** ensures caching is done safely
- Multi-core designs offer new tradeoffs
Introduction

- The chip multiprocessor (CMP) era is upon us!
- Caching complicate writes
- **Cache Coherence** ensures caching is done safely
- Multi-core designs offer new tradeoffs
Background: Directory-based Cache Coherence

• **Directory-based**; explicit per-block accounting
  – Doesn't rely on broadcasts
• Directory operation: client/server
Background: Directory-based Cache Coherence

- **Directory-based**: explicit per-block accounting
  - Doesn't rely on broadcasts
- Directory operation: client/server
  - Processors request data, permissions
Background: Directory-based Cache Coherence

- **Directory-based**: explicit per-block accounting
  - Doesn't rely on broadcasts
- Directory operation: client/server
  - Processors request data, permissions
  - Directory controllers manage memory access
Background: Directory-based Cache Coherence

- **Directory-based**: explicit per-block accounting
  - Doesn't rely on broadcasts
- Directory operation: client/server
  - Processors request data, permissions
  - Directory controllers manage memory access
Background: Directory-based Cache Coherence

- **Directory-based**: explicit per-block accounting
  - Doesn't rely on broadcasts
- Directory operation: client/server
  - Processors request data, permissions
  - Directory controllers manage memory access
Background: Directory-based Cache Coherence

- **Directory-based**: explicit per-block accounting
  - Doesn't rely on broadcasts
- Directory operation: client/server
  - Processors request data, permissions
  - Directory controllers manage memory access
    - Updates, conflicts
Background: Historical MP Cache Coherence

- Distributed directory, memory
Background: Historical MP Cache Coherence

- Distributed directory, memory

```
Cache Miss
```

```

- P  M
- P  M
- P  M
- P  M
```
Background: Historical MP Cache Coherence

- Distributed directory, memory
Background: Historical MP Cache Coherence

- Distributed directory, memory
Background: Historical MP Cache Coherence

- Distributed directory, memory
Background: Historical MP Cache Coherence

- Distributed directory, memory
Motivation: *Multi-core Cache Coherence*
Motivation: *Multi-core Cache Coherence*
Motivation: *Multi-core Cache Coherence*
Motivation: *Multi-core Cache Coherence*
Motivation: *Multi-core Cache Coherence*

Data Request → "Home Node"

Cache Miss

P P P

M M
Motivation: Multi-core Cache Coherence

Data Request -> "Home Node" -> Cache Miss

[P P P] -> [M M M]
Motivation: *Multi-core Cache Coherence*
Motivation: *Multi-core Cache Coherence*
Motivation: *Multi-core* Cache Coherence

- Multi-core designs present radically different relative latency & bandwidth
Outline

- Introduction & Background
- **System Architecture**
- Proximity-Aware Coherence
- Results
- Conclusion
Directory-based Cache Coherence

- Directory structures
Directory-based Cache Coherence

- Directory structures
Directory-based Cache Coherence

- Directory structures
Directory-based Cache Coherence

- Directory structures
  - Directory Memory
Directory-based Cache Coherence

• Directory structures
  – Directory Memory
  – Directory Entries
Directory-based Cache Coherence

- Directory structures
  - Directory Memory
  - Directory Entries
  - Directory Controller
A Traditional Multiprocessor
A Traditional Multiprocessor

(Chassis, board, etc.)

[Diagram showing a multiprocessor system with cores, L2 caches, directories, and memory]

Interconnect
A Traditional Multiprocessor

(Chassis, board, etc.)

Interconnect
Our 16-Core Chip Multiprocessor

Diagram showing a 16-core chip multiprocessor with core, L2 cache, bus, directory control, directory, memory channel, network switch, and tiles 0 to 15.
Our 16-Core Chip Multiprocessor
Our 16-Core Chip Multiprocessor

Diagram showing the architecture of a 16-core chip multiprocessor with tiles connected by a network switch and a bus.
Our 16-Core Chip Multiprocessor
Our 16-Core Chip Multiprocessor
Outline

- Introduction & Background
- System Architecture
- **Proximity-Aware Coherence**
- Results
- Conclusion
Proximity-Aware Coherence

- Idea: home node asks sharer nearest request to forward its cached copy
Proximity-Aware Coherence

- Idea: home node asks sharer nearest requester to forward its cached copy
  - Stay on-chip when possible
Proximity-Aware Coherence

- Idea: home node asks sharer nearest requester to forward its cached copy
  - Stay on-chip when possible
  - Minimize transit of large data-carrying replies
Proximity-Aware Coherence

- Idea: home node asks sharer nearest requester to forward its cached copy
  - Stay on-chip when possible
  - Minimize transit of large data-carrying replies
Proximity-Aware Coherence

- Idea: home node asks sharer nearest requester to forward its cached copy
  - Stay on-chip when possible
  - Minimize transit of large data-carrying replies
Proximity-Aware Coherence

- Idea: home node asks sharer nearest requester to forward its cached copy
  - Stay on-chip when possible
  - Minimize transit of large data-carrying replies
Proximity-Aware Coherence

- Idea: home node asks sharer nearest requester to forward its cached copy
  - Stay on-chip when possible
  - Minimize transit of large data-carrying replies
Proximity-Aware Coherence

- Idea: home node asks sharer nearest requester to forward its cached copy
  - Stay on-chip when possible
  - Minimize transit of large data-carrying replies
Proximity-Aware Coherence

- To service read misses for shared data, traditional protocols use main memory
- Other nodes may hold copies
- On the CMP landscape, inter-node latency is much less than memory latency
Sharer Selection

- When the home node lacks a cached copy, it selects a sharer to ask
Sharer Selection

- When the home node lacks a cached copy, it selects a sharer to ask
Sharer Selection

- When the home node lacks a cached copy, it selects a sharer to ask
  - \textit{rand}
Sharer Selection

- When the home node lacks a cached copy, it selects a sharer to ask
  - rand
  - near1
Sharer Selection

- When the home node lacks a cached copy, it selects a sharer to ask
  - rand
  - near1
  - via1
Sharer Selection

• When the home node lacks a cached copy, it selects a sharer to ask
  – \textit{rand}
  – \textit{near1}
  – \textit{via1}

• Retries didn't prove beneficial
Outline

- Introduction & Background
- System Architecture
- Proximity-Aware Coherence
- Results
- Conclusion
Methodology

- Detailed, execution-driven processor and network simulation
- "RSIM" simulator, adapted to our CMP model
- Parallel workloads from several suites
- Hardware, benchmark details in paper
Proximity-Aware: Potential Coverage

Fraction of read misses to shared lines

- appbt
- fft
- lu
- mp3d
- ocean
- quicksort
- unstruct

Colors represent different performance metrics.
Proximity-Aware: Potential Coverage

Fraction of read misses to shared lines

- appbt
- fft
- lu
- mp3d
- ocean
- quicksort
- unstruct

Legend:
- 6
- 5
- 4
- 3
- 2
- 1
Proximity-Aware: Potential Coverage

Overall $\bar{x}=43\%$
Proximity-Aware: Potential Coverage

Overall $\bar{x}=43\%$
Proximity-Aware: Potential Coverage

Overall \( \bar{x} = 43\% \)

dist 1 \( \bar{x} = 75\% \)
Proximity-Aware: Latency Benefit

Normalized L2 miss latency

- appbt
- fft
- lu
- mp3d
- ocean
- quick sort
- unstruct
- mean

- rand
- near1
- via1
Proximity-Aware: Latency Benefit

The graph shows the normalized L2 miss latency for different applications. The x-axis represents various applications, and the y-axis represents the latency. The bars are color-coded to indicate three different scenarios: rand, near1, and via1.
Proximity-Aware: Latency Benefit

Normalized L2 miss latency

- appbt
- fft
- lu
- mp3d
- ocean
- quick sort
- un-struct
- mean

Latency
-25%
Proximity-Aware: Latency Benefit

Normalized L2 miss latency

Latency -25%
Proximity-Aware: Latency Benefit

Normalized L2 miss latency

Latency -25%

Reply traffic -6%

Normalized L2 miss latency

appbt, fft, lu, mp3d, ocean, quick sort, un- struct, mean

Latency -25%

Normalized L2 miss latency

Latency -25%

Normalized L2 miss latency

Latency -25%
Proximity-Aware: Latency Benefit

Normalized L2 miss latency

appbt  fft  lu  mp3d  ocean  quicksort  unstructured  mean

Reply traffic
-6%

Latency
-25%

Latency:
-25%

Normalized L2 miss latency:
-0.1
-0.1
-0.1
-0.1
-0.1
-0.1
-0.1
-0.1
Proximity-Aware: Speedup

The diagram above shows the speedup of various applications under different scenarios. The applications include appbt, fft, lu, mp3d, ocean, quick sort, unstruct, and mean. The scenarios are labeled as rand, near1, and via1. The x-axis represents the applications, and the y-axis represents the speedup. The chart highlights the performance differences across these applications and scenarios.
Proximity-Aware: Speedup
Proximity-Aware: Speedup

Speedup

- appbt
- fft
- lu
- mp3d
- ocean
- quick sort
- un-struct
- mean

Speedup 16%
Proximity-Aware: Speedup

- L2 latency sensitivity of workloads

- Speedup 16%
Conclusion

• The latency/bandwidth aspects of CMPs motivates multicore-aware coherence redesign
• *One* such change: Proximity-Aware Coherence Coherence
  – Ideas: stay on-chip, decrease "bulk" transit
  – Mean speedup 16%, mean L2 latency down 25%

• More aggressive techniques are under study
Conclusion

- The latency/bandwidth aspects of CMPs motivates multicore-aware coherence redesign
- One such change: Proximity-Aware Coherence Coherence
  - Ideas: stay on-chip, decrease "bulk" transit
  - Mean speedup 16%, mean L2 latency down 25%

- More aggressive techniques are under study

- Questions?