Bulk Disambiguation of Speculative Threads in Multiprocessors

Luis Ceze, James Tuck, Călin Cașcaval† and Josep Torrellas

University of Illinois at Urbana-Champaign
†IBM Research

http://iacoma.cs.uiuc.edu

ISCA, June 2006
Speculative Execution of Threads
Speculative Execution of Threads

• Uses: Thread-Level Speculation, Transactional Memory
Speculative Execution of Threads

• Uses: Thread-Level Speculation, Transactional Memory
• Goal: Improve performance & programmability
Speculative Execution of Threads

- Uses: Thread-Level Speculation, Transactional Memory
- Goal: Improve performance & programmability
- Key requirements:
  - Monitor and enforce data dependences across threads
  - Manage buffering of speculative state
Speculative Execution of Threads

• Uses: Thread-Level Speculation, Transactional Memory

• Goal: Improve performance & programmability

• Key requirements:
  ✓ Monitor and enforce data dependences across threads
  ✓ Manage buffering of speculative state

• How typically done?
  • piggyback on coherence protocol, modify L1 cache
Speculative Execution of Threads

• Uses: Thread-Level Speculation, Transactional Memory
• Goal: Improve performance & programmability
• Key requirements:
  ✓ Monitor and enforce data dependences across threads
  ✓ Manage buffering of speculative state
• How typically done?
  • piggyback on coherence protocol, modify L1 cache

➡ Our goal: Simplify concepts and implementation
Current Mechanisms
Current Mechanisms

Address Disambiguation
• enforce data dependences

wr X → rd X
Current Mechanisms

Address Disambiguation

• enforce data dependences

\[ 	ext{wr } X \rightarrow \text{rd } X \]
Current Mechanisms

Address Disambiguation
• enforce data dependences

![Diagram of Address Disambiguation]

wr X ➔ rd X ➔ eager
Current Mechanisms

Address Disambiguation

• enforce data dependences

\[ \text{wr } X \rightarrow \text{commit} \rightarrow \text{rd } X \]

› lazy
Current Mechanisms

Address Disambiguation
• enforce data dependences

\[ \text{wr } X \rightarrow \text{rd } X \]
Current Mechanisms

Address Disambiguation
• enforce data dependences

Discard Incorrect State
Current Mechanisms

Address Disambiguation
• enforce data dependences

Discard Incorrect State
• squash

Thread ID
Current Mechanisms

Address Disambiguation
- enforce data dependences

Discard Incorrect State
- squash

![Diagram of Address Disambiguation and Discard Incorrect State](image)

---

Luis Ceze  Bulk Disambiguation  June 2006
Current Mechanisms

Address Disambiguation
• enforce data dependences

Discard Incorrect State
• squash  • invalidate

Thread ID
Current Mechanisms

Address Disambiguation
• enforce data dependences

Discard Incorrect State
• squash
• invalidate

wr X → rd X

Thread ID

wr Y
Current Mechanisms

Address Disambiguation
- enforce data dependences

Discard Incorrect State
- squash
- invalidate
Current Mechanisms

Address Disambiguation
• enforce data dependences

Discard Incorrect State
• squash
• invalidate

Multiple Speculative Threads per Cache
Current Mechanisms

Address Disambiguation
• enforce data dependences

Discard Incorrect State
• squash
• invalidate

Multiple Speculative Threads per Cache
Current Mechanisms

Address Disambiguation
• enforce data dependences

Discard Incorrect State
• squash
• invalidate

Multiple Speculative Threads per Cache

• Leverage coherence protocol
• Extend primary caches
• Shortcoming:
  • Tight coupling with critical components
Proposal: Bulk Operations
Proposal: Bulk Operations

• Encode addresses accessed by thread in signatures

<table>
<thead>
<tr>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>W</td>
</tr>
</tbody>
</table>
Proposal: Bulk Operations

• Encode addresses accessed by thread in signatures

- Read R
- Write W

• Support signature operations in hardware
  • Process sets of addresses at once — in bulk
Proposal: Bulk Operations

• Encode addresses accessed by thread in signatures

<table>
<thead>
<tr>
<th>Read</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Support signature operations in hardware
  • Process sets of addresses at once — in bulk

• Use signature operations as building blocks to:
  • Monitor and enforce data dependences across threads
  • Manage buffering of speculative state
Example - Bulk Address Disambiguation
Example - Bulk Address Disambiguation

Thread 0  Thread 1

Luis Ceze  Bulk Disambiguation  June 2006
Example - Bulk Address Disambiguation

Thread 0

 ld X
 st B
 st C
 ld Y

Thread 1

Luis Ceze
Bulk Disambiguation
June 2006
Example - Bulk Address Disambiguation

\[
W_0 = \text{sig}(B, C) \\
R_0 = \text{sig}(X, Y)
\]

Thread 0

Thread 1

\[
\begin{align*}
&\text{ld } X \\
&\text{st } B \\
&\text{st } C \\
&\text{ld } Y
\end{align*}
\]
Example - Bulk Address Disambiguation

\[ W_0 = \text{sig}(B, C) \]
\[ R_0 = \text{sig}(X, Y) \]

Thread 0:
- ld X
- st B
- st C
- ld Y

Thread 1:
- ld B
- st T
- ld C
Example - Bulk Address Disambiguation

\[ W_0 = \text{sig}(B, C) \]
\[ R_0 = \text{sig}(X, Y) \]

\[ W_1 = \text{sig}(T) \]
\[ R_1 = \text{sig}(B, C) \]

\[ \text{Thread 0} \]
- \text{ld} X
- \text{st} B
- \text{st} C
- \text{ld} Y

\[ \text{Thread 1} \]
- \text{ld} B
- \text{st} T
- \text{ld} C

Luis Ceze
Bulk Disambiguation
June 2006
Example - Bulk Address Disambiguation

Thread 0

$W_0 = \text{sig}(B, C)$
$R_0 = \text{sig}(X, Y)$

✓ commit

Thread 1

$W_1 = \text{sig}(T)$
$R_1 = \text{sig}(B, C)$

$W_1 = \text{sig}(T)$
$R_1 = \text{sig}(B, C)$
Example - Bulk Address Disambiguation

\[ W_0 = \text{sig}(B, C) \]
\[ R_0 = \text{sig}(X, Y) \]

\[ W_1 = \text{sig}(T) \]
\[ R_1 = \text{sig}(B, C) \]
Example - Bulk Address Disambiguation

\[ W_0 = \text{sig}(B, C) \]
\[ R_0 = \text{sig}(X, Y) \]

\[ (W_0 \cap R_1) \lor (W_0 \cap W_1) \]
Bulk Operations Pros & Cons
Bulk Operations Pros & Cons

✓ Conceptual and implementation simplicity
Bulk Operations Pros & Cons

✔ Conceptual and implementation simplicity

❌ Inexact operations (superset)
Bulk Operations Pros & Cons

✓ Conceptual and implementation simplicity

✗ Inexact operations (superset)

✓ Correctness is guaranteed
Bulk Operations Pros & Cons

✓ Conceptual and implementation simplicity

✗ Inexact operations (superset)

✓ Correctness is guaranteed

✓ Competitive performance compared to current schemes

⇒ Evaluated in the context of TLS & TM
Outline

- Introduction
- Signatures and Signature Operations
- Commit Process using Signature Operations
- Evaluation
- Conclusion
Accumulating Addresses into Signatures
Signature Operations
Signature Operations

\[ S_1 \cap S_2 \]

intersection
Signature Operations
Signature Operations

\[ S_1 \cap S_2 \]

\[ S_1 \cup S_2 \]
Signature Operations
Signature Operations

\[ S_1 \cap S_2 \]

intersection

\[ S_1 \cup S_2 \]

union

\[ S = \emptyset \, ? \]

is empty?
Signature Operations

Intersection: $S_1 \cap S_2$

Union: $S_1 \cup S_2$

Is empty?: $S = \emptyset$?
Signature Operations

Intersection

Union

Membership

S = Ø?

is empty?

Address

Signature

Encode

= Ø?

T/F

a ∈ S

Luis Ceze

Bulk Disambiguation

June 2006
Signature Operations

\[ S_1 \cap S_2 \]
intersection

\[ S_1 \cup S_2 \]
union

\[ S = \emptyset? \]
is empty?

\[ a \in S \]
membership

---

Luis Ceze  Bulk Disambiguation  June 2006
Signature Operations

\[ \delta(S) \]

decode

Intersection: \( S_1 \cap S_2 \)

Union: \( S_1 \cup S_2 \)

Set membership: \( a \in S \)

Set empty: \( S = \emptyset ? \)

Logic circuit for set operations

Cache set bitmask

Luis Ceze  Bulk Disambiguation  June 2006
Bulk Disambiguation

\((W_C \cap R_R \neq \emptyset) \lor (W_C \cap W_R \neq \emptyset)\)
Bulk Disambiguation

\[(W_C \cap R_R \neq \emptyset) \lor (W_C \cap W_R \neq \emptyset)\]

- Set operations map directly to signature operations
**Bulk Disambiguation**

- Set operations map directly to signature operations
- Can choose disambiguation granularity
  - depends on the address encoded (line, word or byte)
  - reduce squashes due to false sharing

\[(W_C \cap R_R \neq \emptyset) \lor (W_C \cap W_R \neq \emptyset)\]
Bulk Disambiguation

- Set operations map directly to signature operations
- Can choose disambiguation granularity
  - depends on the address encoded (line, word or byte)
  - reduce squashes due to false sharing
- Encoding may cause unnecessary squashes

\[(W_C \cap R_R \neq \emptyset) \lor (W_C \cap W_R \neq \emptyset)\]
**Composed Operation: Signature Expansion**

- Select lines in the cache that belong to the signature
  - used in bulk invalidations
Composed Operation: Signature Expansion

- Select lines in the cache that belong to the signature
  - used in bulk invalidations
Composed Operation: Signature Expansion

- Select lines in the cache that belong to the signature
  - used in bulk invalidations

![Diagram showing the relationship between FSM, Signature, and Cache]

Luis Ceze
Bulk Disambiguation
June 2006
Composed Operation: Signature Expansion

• Select lines in the cache that belong to the signature
  • used in bulk invalidations
Composed Operation: Signature Expansion

- Select lines in the cache that belong to the signature
  - used in bulk invalidations
Composed Operation: Signature Expansion

- Select lines in the cache that belong to the signature
  - used in bulk invalidations

![Diagram showing the process of selecting lines in the cache based on signatures and their use in bulk invalidations.](image-url)
Composed Operation: Signature Expansion

- Select lines in the cache that belong to the signature
  - used in bulk invalidations
Composed Operation: Signature Expansion

- Select lines in the cache that belong to the signature
  - used in bulk invalidations

![Diagram of cache and FSM]

- FSM
- Signature
- δ
- ε
- Selected line addresses
- Cache
  - Tags
  - Data
  - All valid line addresses from selected cache set
Bulk Invalidation
Bulk Invalidation
Bulk Invalidation
Bulk Invalidation
Bulk Invalidation

commit

Comitting (C)  Receiving (R)

Wc

Signature → Signature Expansion → Invalidate Lines

Luis Ceze  Bulk Disambiguation  June 2006
Bulk Invalidation

• Used in the receiver cache to:
Bulk Invalidation

- Used in the receiver cache to:
  - invalidate lines written by the committing thread (using committing thread’s signature $W_C$)
Bulk Invalidation

• Used in the receiver cache to:
  • invalidate lines written by the committing thread (using committing thread’s signature WC)
  • if thread squash, discard speculative state (using local write signature WR)
Commit Process

Committing (C)  

Receiving (R)  

commit

$W_C$
Commit Process

Committing Thread (C)

Thread C commits
Commit Process

Committing Thread (C)

- Thread C commits
- Send out $W_C$
- Set $W_C = R_C = \emptyset$
Commit Process

Comitting Thread (C)

- Thread C commits
- Send out $W_C$
- Set $W_C = R_C = \emptyset$

Receiving Thread (R)

- Thread R receives $W_C$
Commit Process

Committing Thread (C)

Thread C commits

Send out $W_C$
Set $W_C = R_C = \emptyset$

Receiving Thread (R)

Thread R receives $W_C$

Bulk disambiguation $W_C \cap R_R \neq \emptyset \lor W_C \cap W_R \neq \emptyset$

Luis Ceze

Bulk Disambiguation

June 2006
Commit Process

Committing Thread (C)

Thread C commits

Send out $W_C$
Set $W_C = R_C = \emptyset$

Receiving Thread (R)

Thread R receives $W_C$

Bulk disambiguation
$W_C \cap R_R \neq \emptyset \lor W_C \cap W_R \neq \emptyset$

Squash ?

Commit Pr
ocess

Committing (C) Receiving (R)

commit

$W_C$

Luis Ceze Bulk Disambiguation June 2006
Commit Process

Committing Thread (C)

Thread C commits

Send out $W_C$
Set $W_C = R_C = \emptyset$

Receiving Thread (R)

Thread R receives $W_C$

Bulk disambiguation
$W_C \cap R_R \neq \emptyset \lor W_C \cap W_R \neq \emptyset$

Y

Squash ?
Commit Process

Committing Thread (C)

- Thread C commits
- Send out $W_C$
- Set $W_C = R_C = \emptyset$

Receiving Thread (R)

- Thread R receives $W_C$
- Bulk disambiguation $W_C \cap R_R \neq \emptyset \lor W_C \cap W_R \neq \emptyset$
- Squash ?
- Bulk invalidation of cache using $W_R$
Commit Process

Committing Thread (C)

Thread C commits

Send out WC
Set WC = RC = Ø

Receiving Thread (R)

Thread R receives WC

Bulk disambiguation
WC ∩ RR ≠ Ø ∨ WC ∩ WR ≠ Ø

Squash?

Y

Bulk invalidation of cache using WR

Set WR = RR = Ø

Luis Ceze  Bulk Disambiguation  June 2006
Commit Process

Committing Thread (C)

Thread C commits

Send out $W_C$
Set $W_C = R_C = \emptyset$

Receiving Thread (R)

Thread R receives $W_C$

Bulk disambiguation $W_C \cap R_R \neq \emptyset \lor W_C \cap W_R \neq \emptyset$

Squash?

Y

Bulk invalidation of cache using $W_R$
Set $W_R = R_R = \emptyset$

N

Bulk invalidation of cache using $W_C$

Luis Ceze

Bulk Disambiguation
June 2006
Commit Process

All Bulk-based operations
Bulk Disambiguation Module

- Signature Functional Units
- Controller
- Network
- Processor
- Cache and Cache/Coherence Controller

# of Versions
W Signature  R Signature
Bulk Disambiguation Module

- Multiple Speculative Threads in Processor
- Signature Functional Units
- Controller
- Processor
- Cache and Cache/Coherence Controller
- Network

# of Versions
W Signature
R Signature
Bulk Disambiguation Module

- # of Versions
- W Signature
- R Signature
- Signature Functional Units
- Controller
- Processor
- Standard Interface
- Cache and Cache/Coherence Controller
- Network
- Multiple Speculative Threads in Processor
Why Simpler Architecture?
Why Simpler Architecture?

• Compact representation of sets of addresses
Why Simpler Architecture?

- Compact representation of sets of addresses
- Well-defined operations that map directly into hardware
Why Simpler Architecture?

- Compact representation of sets of addresses
- Well-defined operations that map directly into hardware
- No tight coupling with coherence protocol or cache implementation
Why Simpler Architecture?

• Compact representation of sets of addresses
• Well-defined operations that map directly into hardware
• No tight coupling with coherence protocol or cache implementation

See paper for more details
Evaluation

• TM
  • Modified Jikes RVM to insert transaction annotations
  • SPECjbb2000, Multithreaded Java Grande applications

• TLS
  • Binaries generated by POSH TLS compiler [PPoPP’06]
  • SPECint 2000

• Used SESC simulator [sesc.sourceforge.net]
Signature Accuracy

False Positives (%)

Compressed Size (bits)

Signature Size (bits)

Luis Ceze  Bulk Disambiguation  June 2006
Signature Accuracy

- 2 Kbit signature:
  - moderate compressed size (~375b)
  - few false positives (~5%)
Performance in TLS

• Bulk: Only 5% performance degradation over Eager

• Most performance loss comes from Eager → Lazy
Performance in TM

- Bulk performance degradation over Lazy negligible
You will also find on the paper...
You will also find on the paper...

- Transaction nesting using multiple pairs of signatures
You will also find on the paper...

• Transaction nesting using multiple pairs of signatures
• Overflow and context switch discussion
You will also find on the paper...

- Transaction nesting using multiple pairs of signatures
- Overflow and context switch discussion
- In-depth characterization (including bandwidth)
You will also find on the paper...

- Transaction nesting using multiple pairs of signatures
- Overflow and context switch discussion
- In-depth characterization (including bandwidth)
- Supporting forwarding of speculative values in TLS
Conclusion
Conclusion

• Bulk-only design of speculative multithreading
  • for TM and TLS
Conclusion

• Bulk-only design of speculative multithreading
  • for TM and TLS

• Major conceptual and implementation simplification
Conclusion

• Bulk-only design of speculative multithreading
  • for TM and TLS

• Major conceptual and implementation simplification

• Competitive performance (~5% degradation)
Bulk Disambiguation of Speculative Threads in Multiprocessors

Luis Ceze, James Tuck, Călin Cașcaval† and Josep Torrellas

University of Illinois at Urbana-Champaign
†IBM Research

http://iacoma.cs.uiuc.edu

ISCA, June 2006