Architectural Support for Scalable Speculative Parallelization in Shared-Memory Multiprocessors

Marcelo Cintra, José F. Martínez, Josep Torrellas

Department of Computer Science
University of Illinois at Urbana-Champaign
Speculative Parallelization

- Codes with access patterns not analyzable by compiler
- Speculative parallelization can extract some parallelism
- Several designs of speculative CMPs
- **Goal & Contribution:** Scalable speculative architecture using speculative CMPs as building blocks
  - trivially defaults for single-processor nodes

Avg. speedup of 5.2 for 16 processors
(for dominant, non-analyzable code sections)
Outline

- Motivation
- Background
- Speculative CMP
- Scalable Speculation
- Evaluation
- Related Work
- Conclusions
Speculative Parallelization

- Assume no dependences and execute threads in parallel
- Track data accesses
- Detect violations
- Squash offending threads and restart them

```
Do I = 1 to N
  ... = A(L(I)) + ...
  A(K(I)) = ...
EndDo
```

Iteration $J$
- $... = A(4) + ...
- $A(5) = ...$

Iteration $J+1$
- $... = A(2) + ...
- $A(2) = ...$

Iteration $J+2$
- $... = A(5) + ...
- $A(6) = ...$

Intl. Symp. on Computer Architecture - June 2000
Outline

- Motivation
- Background
- Speculative CMP
- Scalable Speculation
- Evaluation
- Related Work
- Conclusions
Speculative CMP

MDT Memory Disambiguation Table

[Krishnan and Torrellas, ICS 98]
Memory Disambiguation Table (MDT)

- **Purpose:** detect dependences + locate versions
- **1 entry per memory line touched**
- **Load and Store bits per word per processor**

<table>
<thead>
<tr>
<th></th>
<th>Load</th>
<th>Store</th>
<th>Load</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>P0</td>
<td>P1</td>
<td>P0</td>
<td>P1</td>
</tr>
<tr>
<td>Tag</td>
<td>P2</td>
<td>P3</td>
<td>P2</td>
<td>P3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0x1234</td>
<td>0 0 1 0</td>
<td>0 1 0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

- Processor 2 has loaded word 0
- Processor 1 has modified word 0
Handling Data Dependences: Loads

- On a load use S bits to find most up-to-date version
Handling Data Dependences: Stores (I)

- Use L and S bits to detect RAW violations and squash
Handling Data Dependences: Stores (II)

- Use L and S bits to detect RAW violations and squash

Miss in L1

MDT

No violation: shielding

L1

L2

P0  P1  P2  P3
Summary of Protocol

- Per-word speculative information
- Multiple versions

<table>
<thead>
<tr>
<th>Squash?</th>
<th>RAW</th>
<th>WAR</th>
<th>WAW</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-order</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Out-of-order</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Speculative CMP

- Storing Versions
  - L1 maintains versions of speculative threads
  - L2 maintains only safe versions
  - Commit: write back dirty versions from L1 to L2

- Static mapping and in-order scheduling of tasks

- L1 and MDT overflows cause stalls
Outline

- Motivation
- Background
- Speculative CMP
- Scalable Speculation
- Evaluation
- Related Work
- Conclusions
Scalable Speculative Multiprocessor

- Goals:

  1. Use largely unmodified speculative CMP in a scalable system
     - trivially defaults for single-processor nodes

  2. Provide simple integration into CC-NUMA
Hierarchical Approach: CMP as Nodes

DIR   Directory
NC    Network Controller
LMDT  Local Memory
GMDT  Global Memory
      Disambiguation Table

SCALABLE INTERCONNECT
### Hierarchy

<table>
<thead>
<tr>
<th>Versions</th>
<th>Spec</th>
<th>Safe</th>
<th>Node</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L1</td>
<td>L2</td>
<td>L2</td>
<td>Mem</td>
</tr>
<tr>
<td>Commits</td>
<td>Processor</td>
<td>L1 \rightarrow L2</td>
<td>L2 \rightarrow Mem</td>
<td></td>
</tr>
<tr>
<td>Spec info</td>
<td>LMDT</td>
<td>GMDT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Granularity of info</td>
<td>Processors</td>
<td>Nodes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mapping of tasks</td>
<td>Thread Static</td>
<td>Chunk Dynamic</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Intl. Symp. on Computer Architecture - June 2000
Hierarchical Approach: Loads

1. Miss in L1
2. Identify on-chip version
3. Miss in L2
4. Identify off-chip version
5. Identify on-chip version
6. Provide version

1. Miss in L1
2. No on-chip version
3. Miss in L2
Hierarchical Approach: Stores
## Mapping of Tasks

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Node</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threads ➔ Processors</td>
<td>Chunks ➔ Nodes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mapping</th>
<th>Node</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td></td>
<td>Dynamic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reason</th>
<th>Node</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimize spec CMP modifications</td>
<td>Minimize load imbalance</td>
<td></td>
</tr>
</tbody>
</table>

- Chunk = consecutive set of threads
Mapping of Tasks

Node 1 finishes iterations

Task: 0 1 2 3 4 5 6 7 8 9 10 11

Update queue
Least-Spec

Writeback L1
D data to L2

GMDT

Intl. Symp. on Computer Architecture - June 2000
Mapping of Tasks

Task: 0 1 2 3 4 5 6 7 8 9 10 11

Least-Spec

Most-Spec

Intl. Symp. on Computer Architecture - June 2000
Node Commits

Node 0 finishes iterations

Task: 0 1 2 3 4 5 6 7 8 9 10 11
12 13 14 15

Update queue
Least-Spec
Writeback L1
D data to L2
Writeback L2
D data to memory

Intl. Symp. on Computer Architecture - June 2000
Node Commits

Node 1 has already finished chunk 1

Task:
16 17 18 19

4 5 6 7

8 9 10 11

Update queue
Least-Spec

Writeback L2
D data to memory

Intl. Symp. on Computer Architecture - June 2000
GMDT Features

+ Allows displacement of clean speculative data from caches
  - 30% faster because of no stall
+ Identify versions to read with a single lookup
+ Selective invalidations with shielding
  - 50% fewer network messages
+ Squash all faulting threads in parallel & restart in parallel
  - 5% faster in a 4 node system

- Commit and squash require sync of GMDT (not processors)
Outline

- Motivation
- Background
- Speculative CMP
- Scalable Speculation
- Evaluation
- Related Work
- Conclusions
Simulation Environment

- Execution-driven simulation
- Detailed superscalar processor model
- Coherent+speculative memory back-end
- Scalable multiprocessor: 4 nodes

Node: spec CMP + 1M L2 + 2K-entry GMDT
CMP: 4 x (processor + 32K L1) + 512-entry LMDT
Processor: 4-issue, dynamic
Applications

- Applications dominated by non-analyzable loops (subscripted subscripts)
- Track, BDNA (PERFECT)
  - APSI (SPECfp95)
  - DSMC3D, Euler (HPF2)
  - Tree (Univ. of Hawaii)
- Non-analyzable loops and accesses identified by the Polaris parallelizing compiler
- Results shown for the non-analyzable loops only
Overall Performance \textbf{Avg. speedup} = 4.4
Overall Performance

Avg. speedup = 4.4

Memory time is most significant performance bottleneck
Overall Performance

Avg. speedup = 4.4

L1 overflow is only noticeable for BDNA and APSI

Intl. Symp. on Computer Architecture - June 2000
Overall Performance

Avg. speedup=4.4

Squash time is only noticeable for DSMC3D
Loop Unrolling

Base: 1 iteration per processor
Unrolling: 2 and 4 iterations per processor

- Increased performance: avg. speedup of 5.2
- Reduction in memory time
- Increase in L1 overflows
Granularity of Speculative State

Increased number of squashes

Line = 16 words (64 Bytes)
Related Work (I)

- CMP schemes:
  Multiscalar (Wisconsin), TLDS (CMU), Hydra (Stanford), MDT (Illinois), Superthreaded (Minnesota), Speculative Multithreading (UPC)

  – designed for tightly-coupled systems: not scalable
Related Work (II)

- Scalable schemes:
  - TLDS (CMU), Zhang et al. (Illinois):
    - Speculative state dispersed along with data
    - Flat view of processors
    - Zhang et al: More sophisticated (handles reduction, load imbalance, large working sets) but complex
Conclusions

- Extended speculative parallelization to scalable system
- Integrated largely unmodified speculative CMP
  - trivially defaults for single-processor nodes
- Promising results: speedup of 5.2 for 16 processors
- Need to support per-word speculative state to avoid excessive squashes
Architectural Support for Scalable Speculative Parallelization in Shared-Memory Multiprocessors

Marcelo Cintra, José F. Martínez, Josep Torrellas

Department of Computer Science
University of Illinois at Urbana-Champaign
Cross Iteration Dependences

<table>
<thead>
<tr>
<th>Application</th>
<th>Parameter (Average)</th>
<th>RAW</th>
<th>WAR</th>
<th>WAW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Same Word</td>
<td>False</td>
<td>Same Word</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>4,869</td>
<td>0.1</td>
</tr>
<tr>
<td>Track</td>
<td>Number</td>
<td>1.0</td>
<td>1.6</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Distance</td>
<td>0</td>
<td>0</td>
<td>95,232</td>
</tr>
<tr>
<td>APSI</td>
<td>Number</td>
<td>0</td>
<td>0</td>
<td>102,912</td>
</tr>
<tr>
<td></td>
<td>Distance</td>
<td>0</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>DSMC3D</td>
<td>Number</td>
<td>147,390</td>
<td>9,350,766</td>
<td>102,912</td>
</tr>
<tr>
<td></td>
<td>Distance</td>
<td>2,640</td>
<td>225</td>
<td>260,051</td>
</tr>
<tr>
<td>Euler</td>
<td>Number</td>
<td>0</td>
<td>104,066</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Distance</td>
<td>0</td>
<td>415</td>
<td>0</td>
</tr>
<tr>
<td>BDNA</td>
<td>Number</td>
<td>0</td>
<td>0</td>
<td>32,422</td>
</tr>
<tr>
<td></td>
<td>Distance</td>
<td>0</td>
<td>0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

*False*: dependence between different words of the same cache line
GMDT Features

+ Allows displacement of clean speculative data from caches
  - 30% faster because of no stall

+ Identify versions to read with a single lookup

+ Selective invalidations with shielding
  - 50% fewer network messages

+ Squash all faulting threads in parallel & restart in parallel
  - 5% faster in a 4 node system

- Commit and squash require sync of GMDT (not processors)
## Application Behavior

<table>
<thead>
<tr>
<th>Access Pattern</th>
<th>Multiple Versions</th>
<th>Per-Word State</th>
<th>Appl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>i0, i1, i2</td>
<td>1. <em>Random, clustered</em></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Track, DSMC3D</td>
</tr>
<tr>
<td>i0, i1, i2</td>
<td>2. <em>Random, sparse</em></td>
<td>No, but may suffer more squashes</td>
<td>DSMC3D, Euler</td>
</tr>
<tr>
<td>i0, i1, i2</td>
<td>3. <em>Often write followed by read</em></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>APSI, BDNA, Tree</td>
</tr>
</tbody>
</table>

*DSMC3D* and *Euler* are specific applications or parts of applications that are mentioned in the table.

*Track* is another application that is mentioned in the table.

*APSI* (Application Specific Instruction Set) and *BDNA* (Bayesian Decision Network Architecture) are also mentioned in the table.
Minimizing Squashes

<table>
<thead>
<tr>
<th>Support for</th>
<th>Squash?</th>
</tr>
</thead>
<tbody>
<tr>
<td>multiple versions</td>
<td>per-word state</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

*False:* dependence between different words of the same cache line
**Interaction GMDT-Directory**

- In general: GMDT operates on speculative data
  - Dir operates on coherent data

- However: Dir sharing vector is kept up-to-date for speculative data for two reasons:
  - smoothen transition in and out of speculative sections
  - further filter our invalidation messages on speculative stores
Multiprogramming

- Replicate window for each job
- Add job id to GMDT entries
## Simulation Environment

<table>
<thead>
<tr>
<th>Processor Param.</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue width</td>
<td>4</td>
</tr>
<tr>
<td>Instruction window size</td>
<td>64</td>
</tr>
<tr>
<td>No. functional units (Int, FP, Ld/St)</td>
<td>3,2,2</td>
</tr>
<tr>
<td>No. renaming registers (Int, FP)</td>
<td>32,32</td>
</tr>
<tr>
<td>No. pending memory ops. (Ld, St)</td>
<td>8,16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory Param.</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1, L2, VC size</td>
<td>32KB, 1MB, 64KB</td>
</tr>
<tr>
<td>L1, L2, VC assoc.</td>
<td>2-way, 4-way, 8-way</td>
</tr>
<tr>
<td>L1, L2, VC, line size</td>
<td>64B, 64B, 64B</td>
</tr>
<tr>
<td>L1, L2, VC, latency</td>
<td>1, 12, 12 cycles</td>
</tr>
<tr>
<td>L1, L2, VC, banks</td>
<td>2, 3, 2</td>
</tr>
<tr>
<td>Local memory latency</td>
<td>75 cycles</td>
</tr>
<tr>
<td>2-hop memory latency</td>
<td>290 cycles</td>
</tr>
<tr>
<td>3-hop memory latency</td>
<td>360 cycles</td>
</tr>
<tr>
<td>LMDT, GMDT size</td>
<td>512, 2K entries</td>
</tr>
<tr>
<td>LMDT, GMDT assoc.</td>
<td>8-way, 8-way</td>
</tr>
<tr>
<td>LMDT, GMDT lookup</td>
<td>4, 20 cycles</td>
</tr>
<tr>
<td>L1-to-LMDT latency</td>
<td>3 cycles</td>
</tr>
<tr>
<td>LMDT-to-L2 latency</td>
<td>8 cycles</td>
</tr>
<tr>
<td>Max. active window</td>
<td>8 chunks</td>
</tr>
</tbody>
</table>
## Application Characteristics

<table>
<thead>
<tr>
<th>Application</th>
<th>Loops to Parallelize</th>
<th>% of Sequential Time</th>
<th>Speculative Data (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track</td>
<td>nfilt_300</td>
<td>41</td>
<td>240</td>
</tr>
<tr>
<td>APSI</td>
<td>run_[20,30,40,60,100]</td>
<td>21</td>
<td>40</td>
</tr>
<tr>
<td>DSMC3D</td>
<td>move3_200</td>
<td>33</td>
<td>24767</td>
</tr>
<tr>
<td>Euler</td>
<td>dflux_[100,200]</td>
<td>90</td>
<td>686</td>
</tr>
<tr>
<td></td>
<td>eflux_[100,200,300]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>psmoo_20</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>BDNA</td>
<td>actfor_240</td>
<td>32</td>
<td>7</td>
</tr>
<tr>
<td>Tree</td>
<td>accel_10</td>
<td>90</td>
<td>1</td>
</tr>
</tbody>
</table>

average: 51

- Performance data reported refer to the loops only
Loop Unrolling

Avg. speedup = 4.7 (Blk 2)
4.6 (Blk 4)
Loop Unrolling

Avg. speedup=4.7 (Blk 2)
4.6 (Blk 4)

Memory time is reduced

Intl. Symp. on Computer Architecture - June 2000
Loop Unrolling

Avg. speedup = 4.7 (Blk 2)
4.6 (Blk 4)

Potential increase in L1 overflows

Intl. Symp. on Computer Architecture - June 2000