Adaptive Mechanisms and Policies for Managing Cache Hierarchies in Chip Multiprocessors

Evan Speight
Hazim Shafi
Lixin Zhang
Ram Rajamony
Adaptive Write Back Management

- Mechanisms to improve performance through intelligent write back handling
  - Write Back History Table
    - Filter clean write backs based on access history and overall system performance
  - L2-to-L2 Write Backs
    - Allow write backs to be kept on-chip when possible
    - Discussed in paper
- When resource contention is high, these mechanisms can improve performance
Outline

- Motivation
- Write Back History Table
- Simulation Environment
- Results
- Conclusions
Target Chip Architecture

Chip Boundary

- Core/L2 Cluster
- Core/L2 Cluster
- Core/L2 Cluster
- Core/L2 Cluster

Intra-chip network

- CTL
- CTL
- CTL

Victim Cache L3

NCU

L2 Slice

L2 Slice

L2 Slice

L2 Slice

Memory

Core

Core

CIU

NCU

L2 Slice

L2 Slice

L2 Slice

L2 Slice

Motivation (1 of 3)
Motivation

- **Separate L3/Memory Pathways**
  - Increased bandwidth availability to off-chip resources
  - No inclusion of L2 caches for L3

- **Victim L3 Cache**
  - Low access latency for L3 cache relative to memory
    - Even lower if brought on-chip
  - Clean and dirty lines written back from L2 caches to L3
    - Better performance than only writing back dirty lines to L3
  - Clean lines written back to L3 are often already in the L3 cache

- **Increasing Number of Cores/Threads Increases Pressure on L3**
  - L2 cache size per core not dramatically increasing
  - Limited queue sizes to handle incoming L3 requests
Percentage of Clean Write Backs Already Present in L3

CPW2 | Notesbench | TP | Trade2
---|---|---|---
40% | 50% | 30% | 60%

Motivation (3 of 3)
Basic Idea

- **Limit number of unnecessary clean write backs to L3**
  - Write backs of dirty lines are always “necessary”
  - Conserve on-chip network bandwidth, L3 tag and queue contention

- **First cut**
  - L3 cache can squash L2 write back request
    - Line already valid in L3
    - Helps some, but high contention can still lead to poor performance

- **Write back history table (WBHT)**
  - Small table added to each L2 cache
  - Tracks lines that each L2 “believes” are already in the L3
  - Organized as a simple tag cache
  - Request never sent to L3 cache
Write Back History Table

- **Cache for tags of lines thought to be in the L3**
  - Entry allocated when L3 squashes write back request

- **Small relative to L2 size**
  - About 9% for 2 MB L2 and 32K entry WBHT

- **Dynamically turn WBHT on/off depending on contention**
  - WBHT correctly predicts L3 contents between 56% and 75%
  - When contention is high, reducing L3 pressure by not writing back lines helps more than increasing L3 hit rates
  - Contention measured by retry count per interval
Choose clean line as victim \rightarrow Place in WBQ \rightarrow In WBHT? \rightarrow Write back clean line \rightarrow Line valid in L3? \rightarrow L3 takes line

Y \rightarrow Squash write back \arrow{Y} \rightarrow Allocate entry in WBHT \arrow{Y} \rightarrow L3 sets WBI bit

N \rightarrow End of critical miss path

L2 actions \rightarrow L3 actions
Methodology

- **Mambo full system simulator**
  - Cycle-accurate memory subsystem

- **Traces of four commercial applications**
  - Details of applications in paper

- **Vary maximum number of outstanding loads allowed per thread to increase contention**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Processors</strong></td>
<td><strong>16</strong></td>
</tr>
<tr>
<td><strong>L2 Cache</strong></td>
<td><strong>2 MB per 4 cores 20 cycle latency</strong></td>
</tr>
<tr>
<td><strong>L3 Cache</strong></td>
<td><strong>16 MB shared 130 cycle latency</strong></td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td><strong>430 cycle latency</strong></td>
</tr>
<tr>
<td><strong>WBHT</strong></td>
<td><strong>32K entries 16-way</strong></td>
</tr>
</tbody>
</table>
Runtime Improvement Using WBHT

% Reduction in Runtime

Outstanding Loads per Thread

CPW2  Notesbench  TP  Trade2
Runtime Improvement Using WBHT

Entries in WBHT

% Reduction in Runtime

- CPW2
- Notesbench
- TP
- Trade2
Reduction in Load Miss Arbitration Delay

Outstanding Loads per Thread

- CPW2
- Notesbench
- TP
- Trade2

% Reduction Over Baseline

0% 10% 20% 30% 40% 50% 60% 70%
Impact on Write Backs, L3 Hit Rate, and Retries

- L2 Write Backs
- L3 Hit Rate
- L3 Retries

% Reduction Over Baseline

- CPW2
- Notesbench
- TP
- Trade2
Conclusions

- In situations of high system contention, intelligently managing write backs can help performance.
- Need to maintain performance at low contention.
- Mileage may vary.
- Can be combined with allowing L2 → L2 write backs:
  - L2 → L2 write backs not as tied to contention.
  - Performance improvement ranges from 1 to 13% over baseline.
  - Details in paper.
Future Work

- Investigate performance on full system execution-based simulation
  - Limitations of trace-based simulation
- Compare results with adding a similar-sized victim cache to each L2
This work is part of the PERCS project and was supported under DARPA HPCS Phase II

For more information: speight@us.ibm.com