Analyzing the Impact of Useless Write-Backs on the Endurance and Energy Consumption of PCM Main Memory

Santiago Bock, Bruce Childers, Rami Melhem, Daniel Mossé and Youtao Zhang

University of Pittsburgh
Introduction

- Datacenters are growing in size and number
  - Energy consumption will cost $7.4 billion in 2011

- Memory consumes 20% to 40% of energy in a typical server
  - Larger memories due to multi-core
  - Smaller transistor sizes leak more current

- PCM for main memory
  - Low static power due to non-volatility
  - Read performance comparable to DRAM
  - Better scalability than DRAM
  - High energy cost of writes
  - Limited write endurance
Motivation

- A write-back is *useless* when its data is not used again
  - Avoiding useless write-backs requires future knowledge

- Idea: use application information
  - Memory allocator
  - Control flow analysis
  - Stack pointer

- Focus of this work
  - How many useless write-backs can be avoided?
  - What’s the impact on endurance and energy consumption?
Outline

- Introduction
- Motivation
- What is Phase Change Memory?
- What are useless write-backs?
- How do we count useless write-backs?
- How much can we gain?
- Conclusions
Background on PCM Main Memory

- **PCM writes**
  - Modify physical state
  - Slow
  - High energy cost
  - Limited to $10^6$ to $10^8$

- **Main memory architecture**
  - L2 cache
  - Small DRAM cache (optional)
  - Large PCM main memory
# Useless Write-Backs

<table>
<thead>
<tr>
<th>Action</th>
<th>Cache Status</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write A</td>
<td></td>
<td>A becomes dirty</td>
</tr>
<tr>
<td>Read A</td>
<td></td>
<td>A is used</td>
</tr>
<tr>
<td>Read A</td>
<td></td>
<td>A is used again</td>
</tr>
<tr>
<td>Read B</td>
<td></td>
<td>A is evicted and written back</td>
</tr>
<tr>
<td>Write A</td>
<td></td>
<td>Original value of A is overwritten</td>
</tr>
</tbody>
</table>

*The write-back of A is useless because A is dead*
Useless Write-Backs

- Detecting useless write-backs
  - Difficult to identify last read before a write
  - Use program information to detect dead memory locations

- Detecting dead memory locations depends on the type of memory region
  - **Heap**: use calls to `malloc()` and `free()`
  - **Global**: use control flow analysis
  - **Stack**: use the stack pointer
Analysis Framework

- **Trace**: address and type of each memory reference
- **Analyzer**: cache simulator and list of dead memory locations

Endurance gains
Energy savings
Analysis for Heap Data

Trace:
- malloc(1) returns 3
- write to 3
- free(3)
- read from 7
- malloc returns 3

Cache:
- malloc(1) returns 3
- write to 3
- free(3)
- read from 7
- malloc returns 3

List of allocated blocks:
- 3,1

List of dead blocks:
- 3,1
- write-back of a is useless!
- 3 becomes dead!
Analysis for Global Data

Trace:

1. write 5
2. 
3. read 5
4. 
5. 
6. 
7. read 9
8. 
9. write 5

Cache:

Objects (id, last access, last write-back):

1. → 5,1,0
2. 
3. → 5,3,0
4. 
5. 
6. 
7. → 5,3,7
8. 
9. → 5,9,7

3 < 7: useless write-back!
Analysis for Stack Data

Trace:
- read 3, stack 100
- write 90, stack 80
- read 5, stack 100
- read 2, stack 100

Cache:

Min Stack Pointer:
- 100
- 80

Stack:
- 100:
- 96:
- 92:
- 88:
- 84:
- 80:

Stack frame becomes dead
write-back of a is useless
Methodology

- SPEC CPU2006 benchmark suite
  - 26 benchmarks
  - 52 combinations of benchmark/input
- Pin collects traces
  - 100 billion instructions
- L2 Cache
  - 1MB
  - 8-way, LRU
- DRAM Cache
  - No cache, 8MB, 16MB, 32MB and 64MB
  - 16-way, LRU
- Cache line size
  - 8B (limit study), 32B, 64B and 128B
Experimental Results

- Categorization of benchmarks based on memory region
  - Heap intensive: more than 1 million object allocations
  - Global intensive: more than 4MB global size

Size of Global Region in Bytes
Heap (8-byte cache line)

Fraction of useless write-backs

- astar-1
- perlbench-3
- gcc-5
- astar-2
- sphinx3
- Average

Energy savings

- astar-1
- perlbench-3
- gcc-5
- astar-2
- sphinx3
- Average

Santiago Bock
Heap (Average Endurance Gains)

<table>
<thead>
<tr>
<th>DRAM Cache Size</th>
<th>8 Byte Cache Line</th>
<th>32 Byte Cache Line</th>
<th>64 Byte Cache Line</th>
<th>128 Byte Cache Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>No DRAM</td>
<td>25%</td>
<td>20%</td>
<td>15%</td>
<td>10%</td>
</tr>
<tr>
<td>8MB</td>
<td>8 MB</td>
<td>16 MB</td>
<td>32 MB</td>
<td>64 MB</td>
</tr>
<tr>
<td>16MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Heap (Average Energy Savings)

<table>
<thead>
<tr>
<th>Type of Saving and DRAM Cache Size</th>
<th>8 Byte Cache Line</th>
<th>32 Byte Cache Line</th>
<th>64 Byte Cache Line</th>
<th>128 Byte Cache Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM 0MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCM 8MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCM 16MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCM 32MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCM 64MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total 8MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total 16MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total 32MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total 64MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Global (8-byte cache line)

Fraction of useless write-backs

<table>
<thead>
<tr>
<th>Game</th>
<th>DRAM</th>
<th>PCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>games-1</td>
<td>30%</td>
<td>5%</td>
</tr>
<tr>
<td>games-2</td>
<td>40%</td>
<td>10%</td>
</tr>
<tr>
<td>gobmk-2</td>
<td>35%</td>
<td>15%</td>
</tr>
<tr>
<td>zeusmp</td>
<td>30%</td>
<td>10%</td>
</tr>
<tr>
<td>Average</td>
<td>30%</td>
<td>10%</td>
</tr>
</tbody>
</table>

Energy savings

<table>
<thead>
<tr>
<th>Game</th>
<th>DRAM</th>
<th>PCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>games-2</td>
<td>25%</td>
<td>5%</td>
</tr>
<tr>
<td>zeusmp</td>
<td>30%</td>
<td>15%</td>
</tr>
<tr>
<td>games-1</td>
<td>20%</td>
<td>10%</td>
</tr>
<tr>
<td>gobmk-2</td>
<td>15%</td>
<td>5%</td>
</tr>
<tr>
<td>Average</td>
<td>20%</td>
<td>10%</td>
</tr>
</tbody>
</table>
Global (Average Energy Savings)

Type of savings and DRAM cache size
Global (Average Energy Savings)

Type of savings and DRAM cache size

- 8 Byte Cache Line
- 32 Byte Cache Line
- 64 Byte Cache Line
- 128 Byte Cache Line
Stack

- Very few useless write-backs
  - Fraction of useless write-backs between 0% and 2.3%
  - Average endurance gains and energy savings between 0% and 0.1%

- Programs use a small part of the stack
  - 10KB to 20KB
  - Kept mostly in the cache
  - Few opportunities to evict dead data from the cache
Conclusions

- We showed that a considerable amount of write-backs are useless.

- We showed there is potential:
  - Up to 20% energy savings
  - Up to 26% endurance gains

- Next step: develop techniques to avoid useless write-backs:
  - Low energy cost
  - Low performance impact
Thank you!

Questions?

sab104@cs.pitt.edu
http://www.cs.pitt.edu/~sab104