

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



PCM@PITT

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



Santiago Bock

PCM@PITT

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?

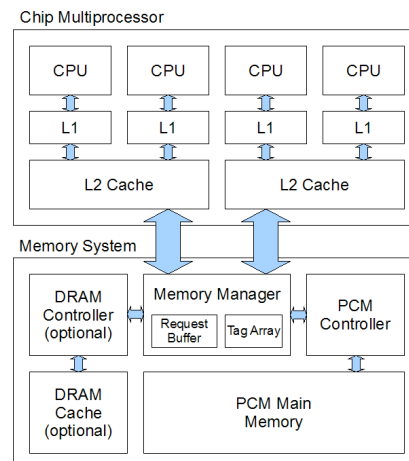
Santiago Bock



Background on PCM Main Memory

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




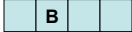

- Main memory architecture
 - L2 cache
 - Small DRAM cache (optional)
 - Large PCM main memory

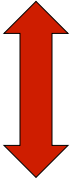


Santiago Bock



Useless Write-Backs

Action	Cache Status	Comment
Write A		A becomes dirty
Read A		A is used
Read A		A is used again
Read B		A is evicted and written back
Write A		Original value of A is overwritten


A is dead

The write-back of A is useless because A is dead

Santiago Bock



Useless Write-Backs

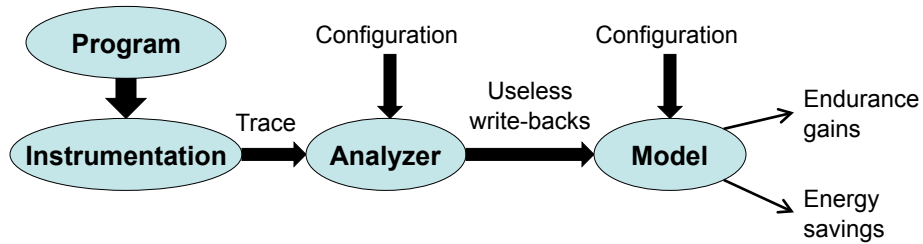
- Detecting useless write-backs
 - Difficult to identify last read before a write
 - Use program information to detect dead memory locations

- 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

Santiago Bock



Analysis Framework

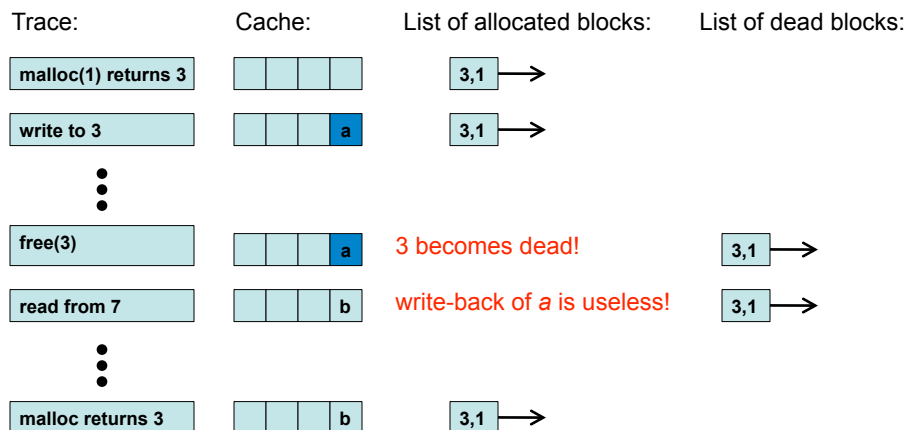


- **Trace**: address and type of each memory reference
- **Analyzer**: cache simulator and list of dead memory locations

Santiago Bock



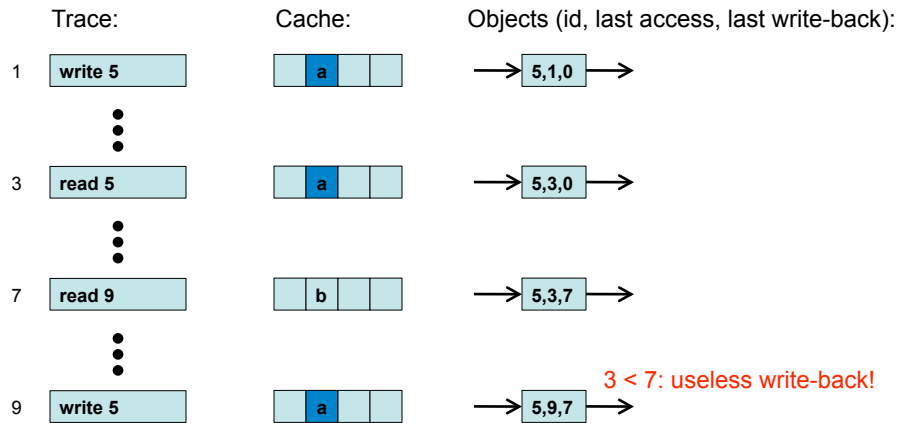
Analysis for Heap Data



Santiago Bock



Analysis for Global Data



Santiago Bock



Methodology

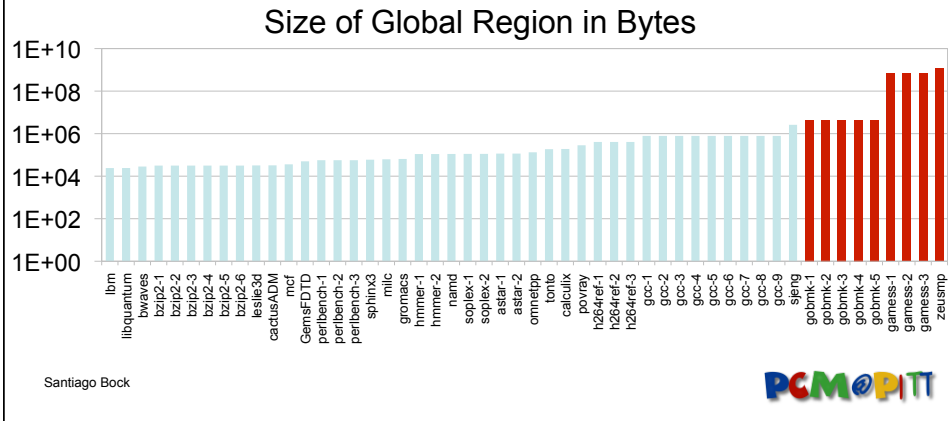
- SPEC CPU2006 benchmark suite
 - 26 benchmarks
 - 52 combinations of benchmark/input
- Pin collects traces
 - 100 billion instructions
- L2 Cache
 - 1MB
 - 8-way, LRU
- DRAM Cache
 - No cache, 8MB, 16MB, 32MB and 64MB
 - 16-way, LRU
- Cache line size
 - 8B (limit study), 32B, 64B and 128B

Santiago Bock

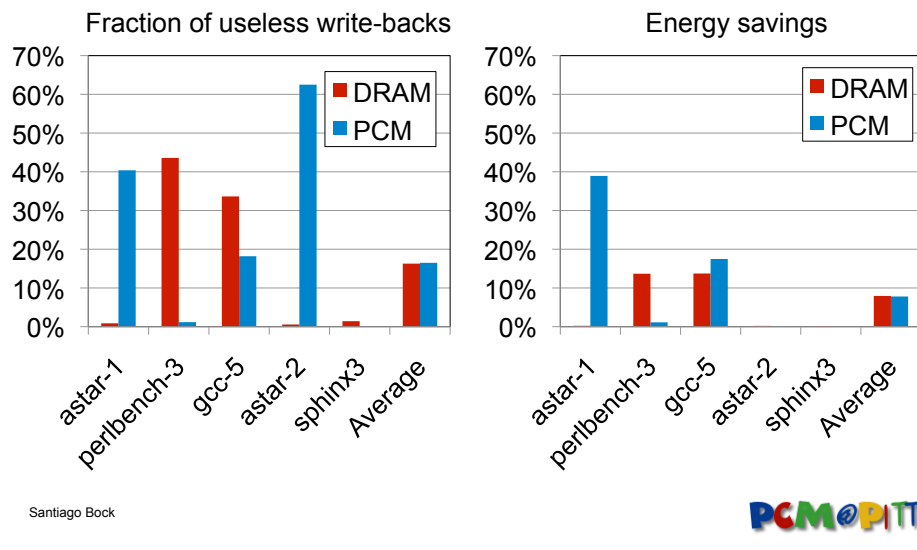


Experimental Results

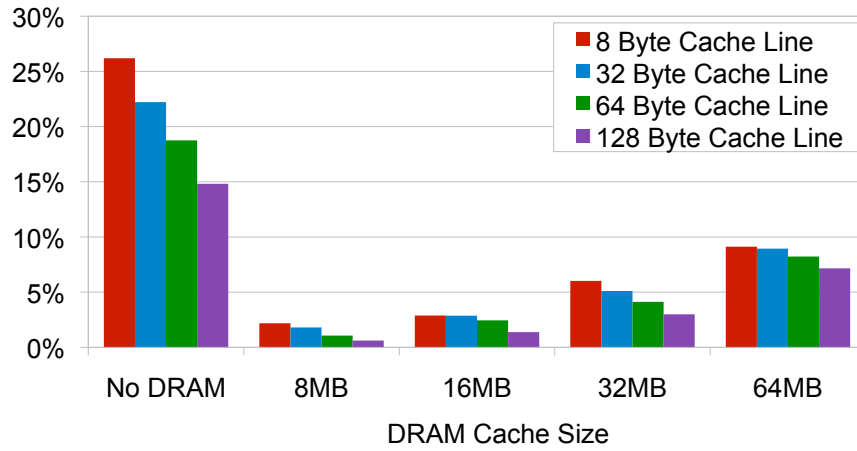
- Categorization of benchmarks based on memory region
 - Heap intensive: more than 1 million object allocations
 - Global intensive: more than 4MB global size



Heap (8-byte cache line)



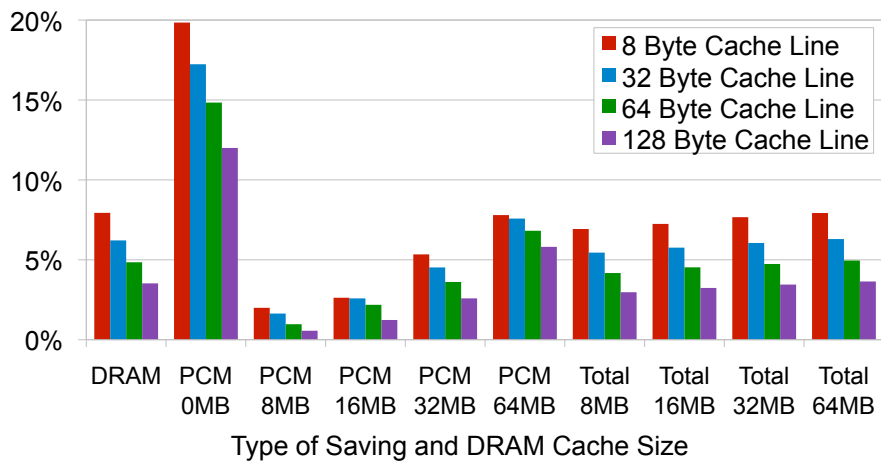
Heap (Average Endurance Gains)



Santiago Bock



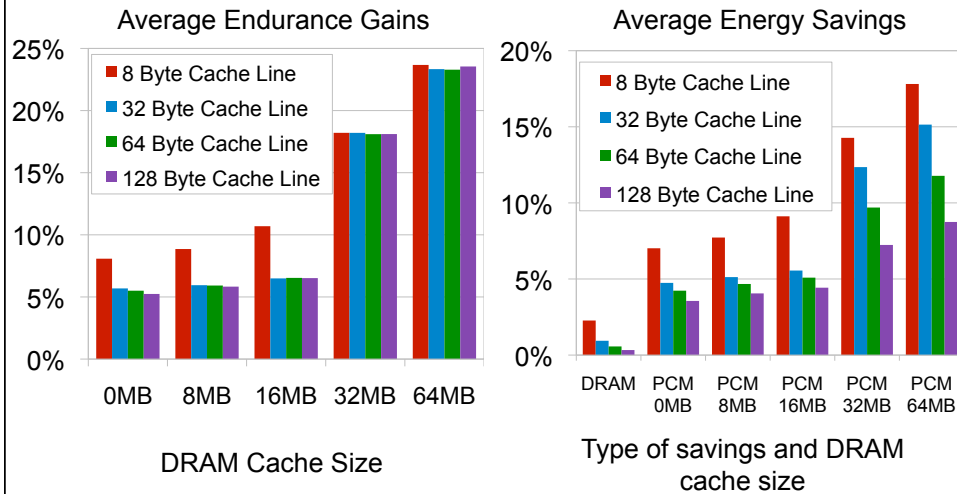
Heap (Average Energy Savings)



Santiago Bock



Global



Santiago Bock

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

Santiago Bock

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

Santiago Bock



Thank you!

Questions?

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

Santiago Bock

