Architectural Solutions for Storage-Class Memory in Main Memory

Michele Franceschini
Joint work with: Moinuddin Qureshi, John Karidis, Luis Lastras, Alan Bivens, Parijat Dube, Bulent Abali
IBM T.J. Watson Research Center
Motivations

- The latency and bandwidth gap between last level of memory and first storage level is more than 3 orders of magnitude.
- There is space in the memory hierarchy behind DRAM.
- Every single level of the memory hierarchy nowadays is volatile (except for paging into disk).

- Storage class memories promise:
  - Low cost
  - Ultra-low standby power
  - Bring non-volatility in the memory hierarchy
Why worry about memory?

- DRAM density is evolving more slowly than CPU performance.
Storage Class Memory

- Storage class memory [1,2] (SCM) fills in the gap between memory-type characteristics and storage-type
  - It is persistent
  - It is faster than hard drives
  - It is denser/cheaper than DRAM

- Possible examples could be NAND Flash, Phase-Change Memory, Resistance RAM.

Phase-Change Memory as a SCM

- We will mostly focus on PCM, however
  - Most of the work can be applied to other memory technologies
- Assumptions on PCM
  - Denser than DRAM
  - Slower than DRAM:
    - 2x to 4x read access time
    - 10x to 100x lower write bandwidth
  - Retention > 1 week
    - Refresh of the data has minor impact on system performance
  - Supports both SLC and MLC (e.g. MLC2, MLC4)
  - Iterative (write and verify) programming is required
  - Limited endurance (e.g. $10^8$ rewrite cycles)
  - Large write power (10x greater than DRAM)
Hybrid Memory

- The direct use of SCM in main memory would expose the system to the drawbacks of SCM leading to an unacceptable performance hit.
- An effective solution employs both standard memory technology such as DRAM, and SCM.
- Ideally with:
  - Response time of a DRAM-based memory subsystem
  - Added benefits from SCM (lower cost, larger memory, persistency)
  - No changes in user software
Structure of a Hybrid Memory System

- HMC controls two pools of different memories
- DRAM serves as a large cache for the SCM
The Hybrid Memory Controller

- Controls the memory pools
- Manages DRAM as a cache
- Provides an ECC layer
- Performs wear leveling
- Controls low level access to the SCM
Some architectural solutions to enable Hybrid Memory

We will focus on:

- Efficient wear leveling
- Asymmetry of write and read latency
- Exploiting tradeoffs obtained by operating at different densities
Heavy non-uniformity in writes: <10% lines incur 90%+ of write traffic

Anything below $10^{13}$ endurance is likely to require some form of wear leveling
Possible approaches

- **Table based wear leveling**
  - 😊 Very generic, excellent wear leveling is easy to achieve
  - 😞 Big indirection table required
  - 😞 Additional latency incurred

- **Algorithmic remapping based wear leveling**
  - 😊 Avoids the use of a big table
  - 😊 Low latency
  - 😞 Limitations on line/page/block relocation

- Latency is a key factor: focus on second category
Start-Gap Wear Leveling

Two registers (Start & Gap) + 1 line (GapLine) to support movement. Move GapLine every 100 writes to memory.

PCMAAddr = (Start+Addr); (PCMAAddr >= Gap) PCMAAddr++

Storage overhead: less than 8 bytes (GapLine taken from spares)
Write overhead: One extra write every 100 writes 1%
Latency: Two additions (no table lookup)

Results for Start-Gap

On average, Start-Gap gets 53% normalized endurance. 10X better than baseline, but still 2x lower than Ideal. Why?
Need for Address Space Randomization

Start-Gap moves a line to its neighbor ➔ If heavily written regions are spatially close, Start-Gap may move hot lines to other hot lines.

If address space is randomized, hot regions will be spread uniformly.
Randomized Start Gap

Randomizer must provide one-to-one mapping ➞ Invertible function
Randomized mapping must remains constant (configure at design/boot)
Results for Randomized Start-Gap

Randomized Start-Gap achieves 97% of ideal lifetime
While incurring a total storage overhead of 13 bytes!
Read/Write Asymmetry: Problem of Contention from Slow Writes

PCM writes 4x-8x slower than reads. Writes not latency critical. Typical response: Use large buffers and intelligent scheduling.

But once write is scheduled to a bank, later arriving read waits.

Write request causes contention for reads \(\rightarrow\) increased read latency.

Problem

Read Latency=1k cycles  Write Latency=8k cycles
12 workloads: each with 8 benchmarks from SPEC06

Writes significantly increase read latency
(Problem only for asymmetric memories)
Iterative Write in PCM devices

In Multi-Level Cells (MLC), the programming precision requirement increases linearly with the number of levels.

PCM cells respond differently to same programming pulse.

Acknowledged solution to address uncertainty: Iterative writes.

Each iteration consists of steps of: write-read-verify.
Model for Iterative Writes

We develop an analytical model to capture number of iterations: In terms of bits/cell, num levels written in one shot, and learning Time required to write a line is worst-case of all cells in line

Avg number of iterations: 8.3 (consistent with MLC literature)
Write Cancellation

Write Cancellation: “abort” on-going write to Improve read latency

Line in non-deterministic state: read matching read request from WRQ

Perform write cancellation as soon as a read request arrives at a bank (as long as the write is not done in forced-mode)
Write Pausing

Iterative writes can be paused to service pending read requests

Potential Pause Points
Iter 1 Iter 2 Iter 3 Iter 4

Reads can be performed at the end of each iteration (potential pause point)

Rd X
Iter 1 Iter 2 Rd X Iter 3 Iter 4

Better read latency with negligible write overhead

We extend the iterative write algorithm of Nirschl et al. [IEDM’07] to support Write Pausing
Write Pausing + WCAT

Only one iteration is cancelled ➔ “micro-cancellation” has low overhead
Results for Write Pausing

Write Pausing at end of iteration gets 85% of benefit of “Anytime” Pause
Results

Baseline: 2365 cycles Ideal: 1K cycles

Write Pause + Micro Cancellation very close to Anytime Pause
(re-execution overhead of micro cancellation <4% extra iterations)
Exploiting dynamic cell density

- PCM is a flexible memory technology which allows to trade-off storage density with
  - write bandwidth
  - write power
  - read latency
- For example: a line of PCM cells can be programmed at
  - 2 bits per cell in average in 600ns
  - 4 bits per cell in average in 2400ns
- The same line of PCM cells can be read
  - 2 bits per cell in average in 120ns [1]
  - 4 bits per cell in average in 180ns
- (actual numbers can vary with technology)

Idea: Morphable Memory

- Remember the access pattern:

  - Most of the accesses happen in a fraction of the memory

- Idea:
  - Low density, high speed region to serve this fraction
  - High density, lower speed, to provide a large main memory
Memory structure

Memory

Remap Table

PA

Counters

MRU

LRU

MMU

X=OptimalMem

TQ

PageStatusTable

MA
Preliminary results

- To evaluate the performance impact:
  - 8GB memory with 4 bit/cell memory accessible at 1000 cycle latency.
  - 6GB main memory with 3 bits/cell memory accessible at 750 cycle latency.
  - 4GB at 2 bits/cell with 500 cycle access latency.
  - Two benchmarks (kmeans and binary search) are analyzed with different input sets: 40%, 60% and 90% of 8GB.

- As shown below for small datasets overall execution time is minimum for the 4GB system and for the med input set the execution time is minimized for 6GB memory system.

For more details please refer to our upcoming ISCA 2010 paper:

Moinuddin K. Qureshi Michele M. Franceschini Luis A. Lastras-Montaño John P. Karidis, "Morphable Memory System: A Robust Architecture for Exploiting Multi-Level Phase Change Memories"
Concluding Remarks

- Storage Class Memories can provide interesting benefits to main memory
- Emerging memory technologies don’t fit directly in current memory subsystems
- The use of a Hybrid Memory system provides a viable solution and can alleviate the major drawbacks of SCM
  - limited endurance, write bandwidth and large read latency
- Exposing low level access to the device is important
  - not all tradeoffs should be pin-pointed at design time
  - Several can be used to provide enhanced performance
Thank you!