Runtime Solutions to Apply Non-volatile Memories in Future Computer Systems

Hoda Aghaei Khouzani

Fall 2016

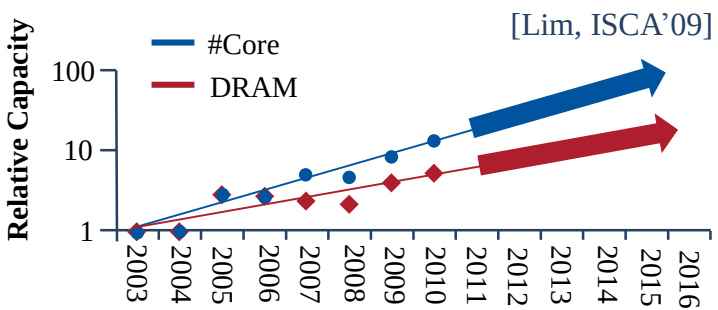
Outline

- Introduction
- Prolonging PCM Limited Lifetime
- Addressing PCM Write Latency and Energy
- Reducing DWM Access Latency
- Summary and Future Works

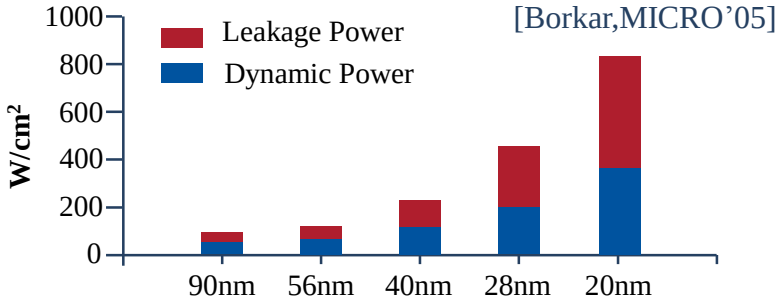
Trends Affecting Main Memory

DRAM has been used as main memory from 1970

However, DRAM technology scaling is ending



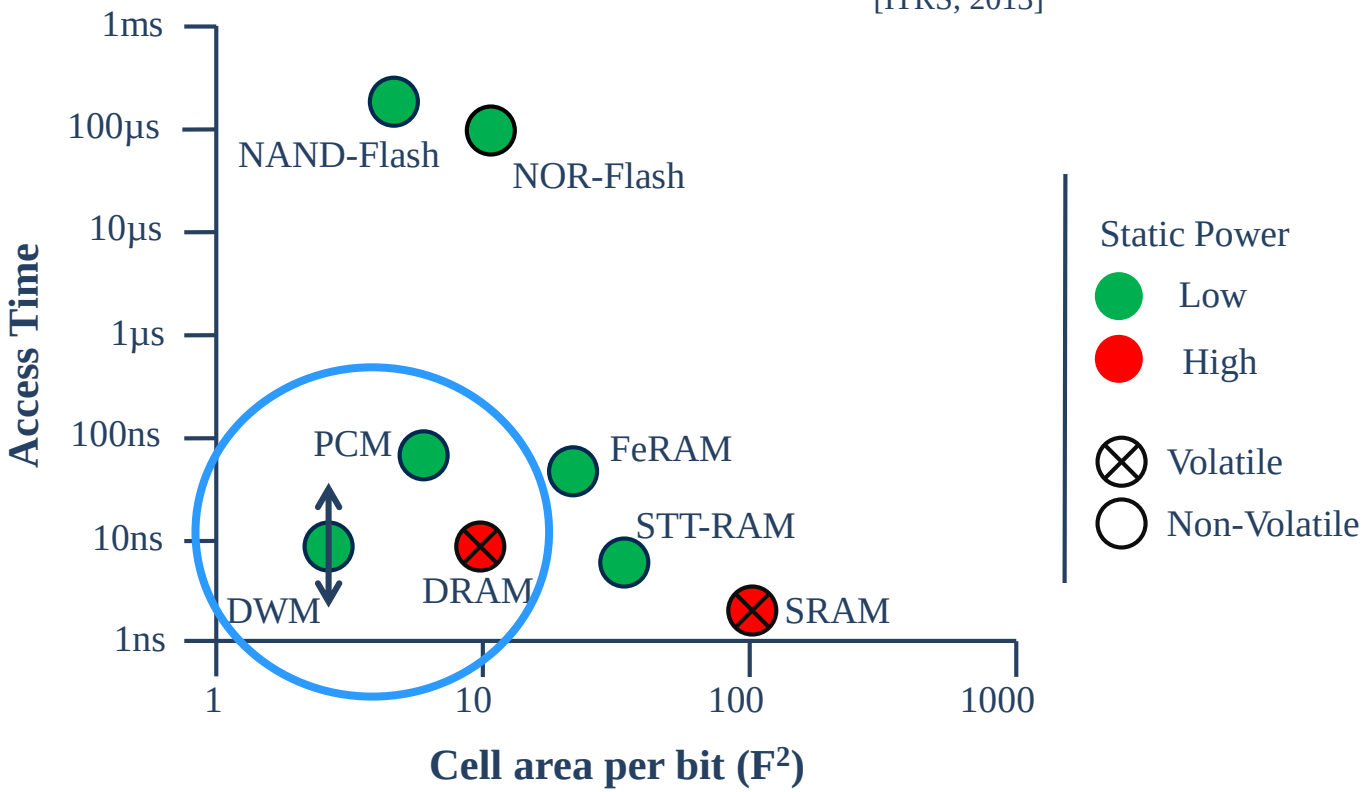
Memory capacity per core expected to drop by **30**% every two years [Mutlu, IMW'13]



DRAM consumes up to **40**% of system **energy** due to the need of consistent refresh cycles [Udipi, ISCA'10]

Emerging Memory Technologies

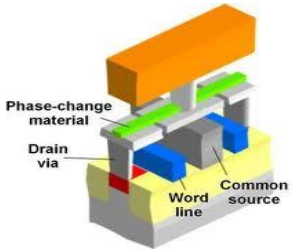
[ITRS, 2013]



Potential Candidates for Future Main Memory

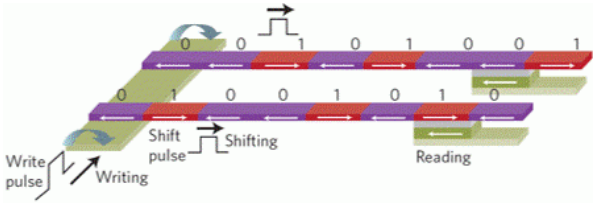
(1) Phase Change Memory (PCM)

- Advantages:
  - Higher scalability than DRAM
  - Near zero static power
  - Similar read performance and energy to DRAM
- Challenges:
  - Limited write endurance (10^6~10^9)
  - Long write latency (10x DRAM)
  - High write energy (4x DRAM)



(2) Domain Wall Memory (DWM)

- Advantages:
  - Ultra dense
  - Near zero static power
  - Similar read/write performance and energy to DRAM
- Challenge:
  - Sequential access structure



Outline

- Introduction
- Prolonging PCM Limited Lifetime
- Addressing PCM Write Latency and Energy
- Reducing DWM Access Latency
- Summary and Future Works

Related Works on PCM Lifetime

- Applies when cells die
- Increases access latency

Defect Tolerance

Write Reduction

Write Balancing

- At bit level
- Reduces average writes
- Lifetime limits by the maximum

- At different levels of granularity
- Balances writes
- Extra writes due to remap

[1] S. Cho, et al, "Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance," in MICRO, 2009.

[2] P. Zhou, et al, "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," in ISCA, 2009.

[3] A. Ferreira, et al, "Increasing PCM main memory lifetime," in DATE, 2010.

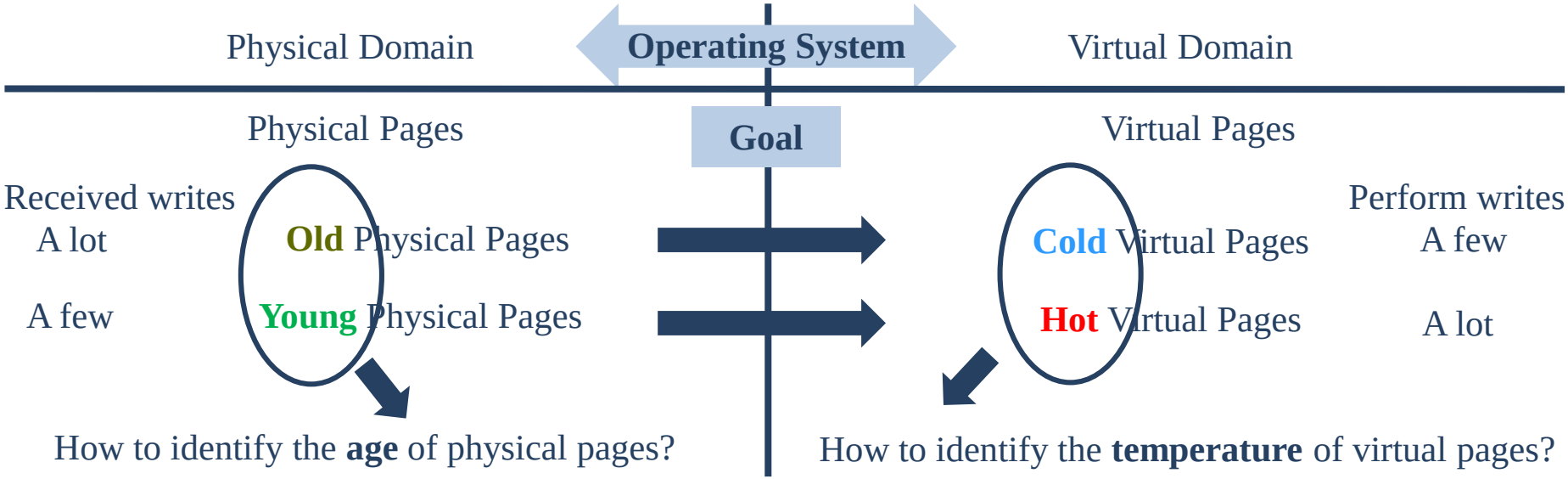
[4] M. Qureshi, et al, "Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling," in MICRO, 2009.

[5] P. Zhou, et al, "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," in ISCA, 2009.

[6] N. H. Seong, et al, "SAFER: Stuck-at-fault Error Recovery for Memories," in MICRO, 2010.

[7] J. Fan, et al, "Aegis: partitioning data block for efficient recovery of stuck-at-faults in phase change memory," in MICRO, 2013.

Goal and Challenges



How frequent should remap be?

Identification Solution

Physical Domain

How to identify the **age** of physical pages?

N-bit counter per physical page

$$N = \log_2(Wear\ out\ Limit)$$

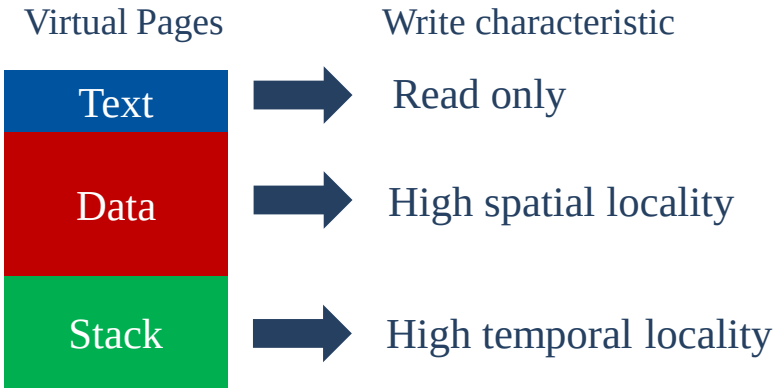
For wear out limit 10^9, almost 30 bits needed per 4KB pages. So the overhead is <0.001.

Virtual Domain

How to identify the **temperature** of virtual pages?

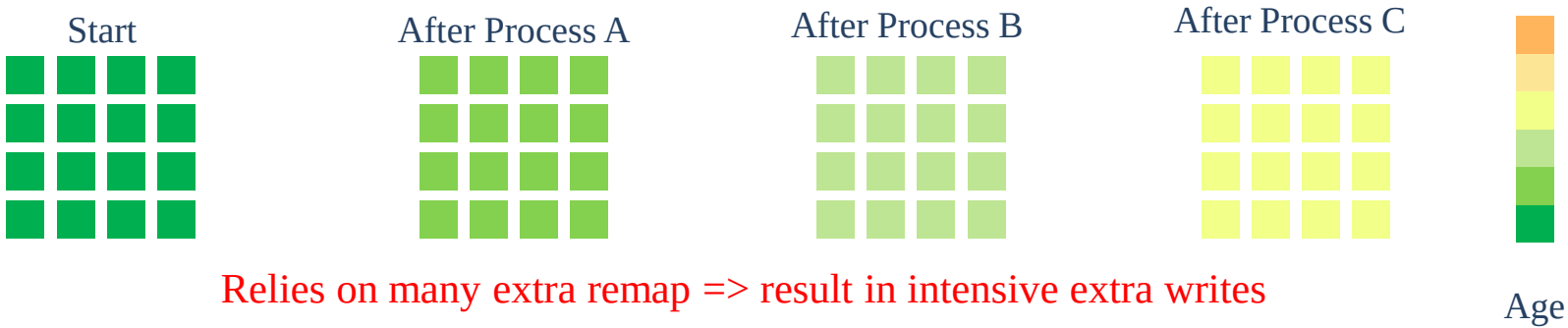
Counters again? No!!

(1) Virtual domain is larger than physical domain.
(2) More importantly, no need.



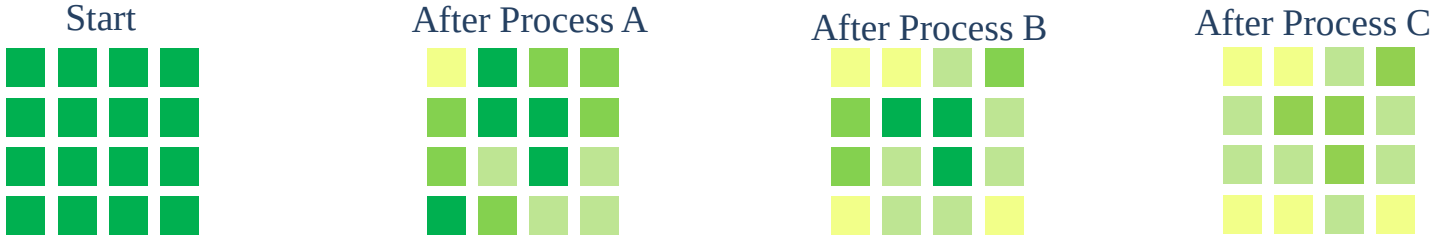
How frequent should remap be?

In existing works



Relies on many extra remap => result in intensive extra writes

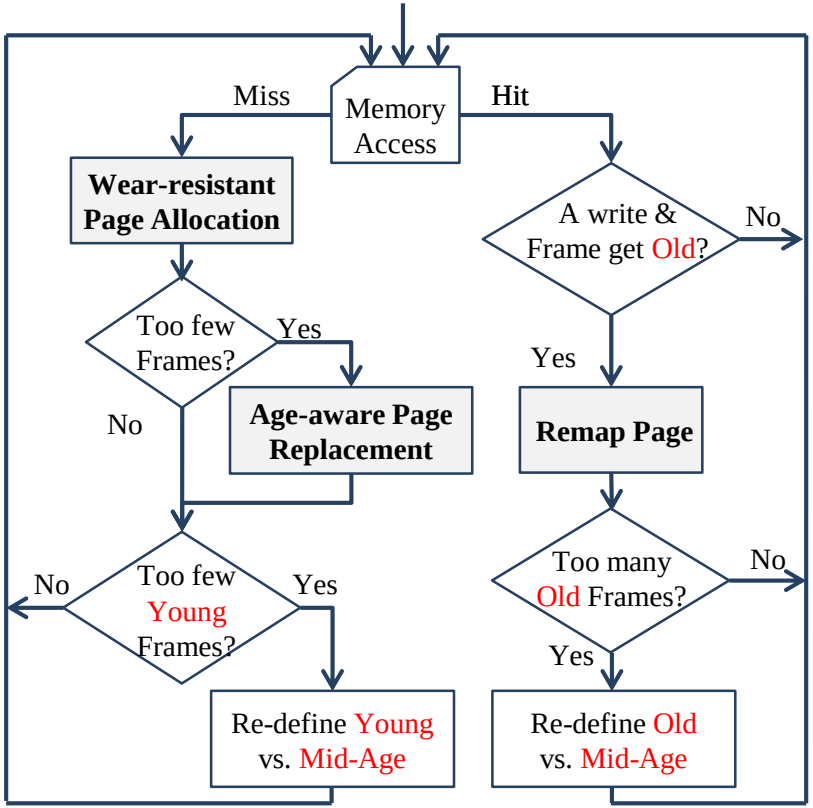
My Idea: Wear leveling can be done across different processes



Almost no extra remap => Very limited extra writes

Implementation: Mapping upon page allocation in OS

System Overview



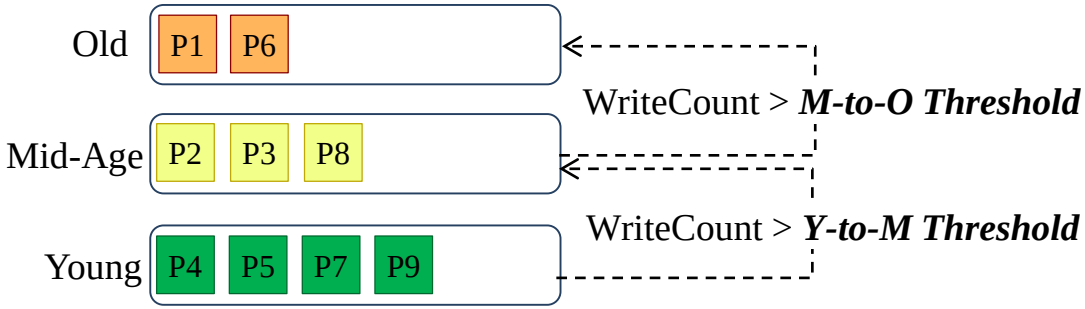
Following are added to the system:

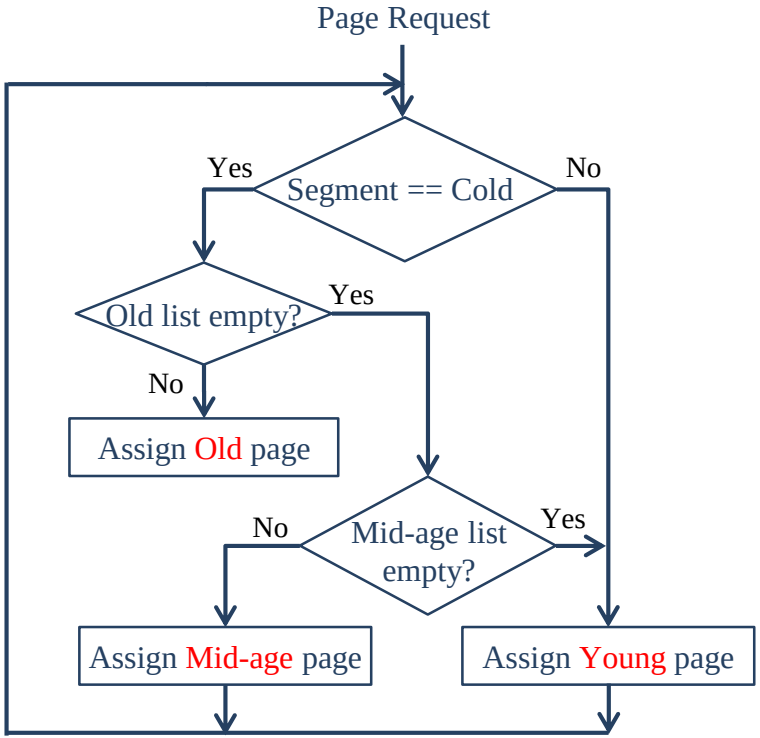
- Three procedures
  - Wear-resistant page allocation
  - Age-aware page replacement
  - Remap hot page
- Two thresholds
  - Young-to-Midage threshold
  - Midage-to-Old threshold

Wear-resistant Page Allocation

How to quickly find the appropriate physical page?

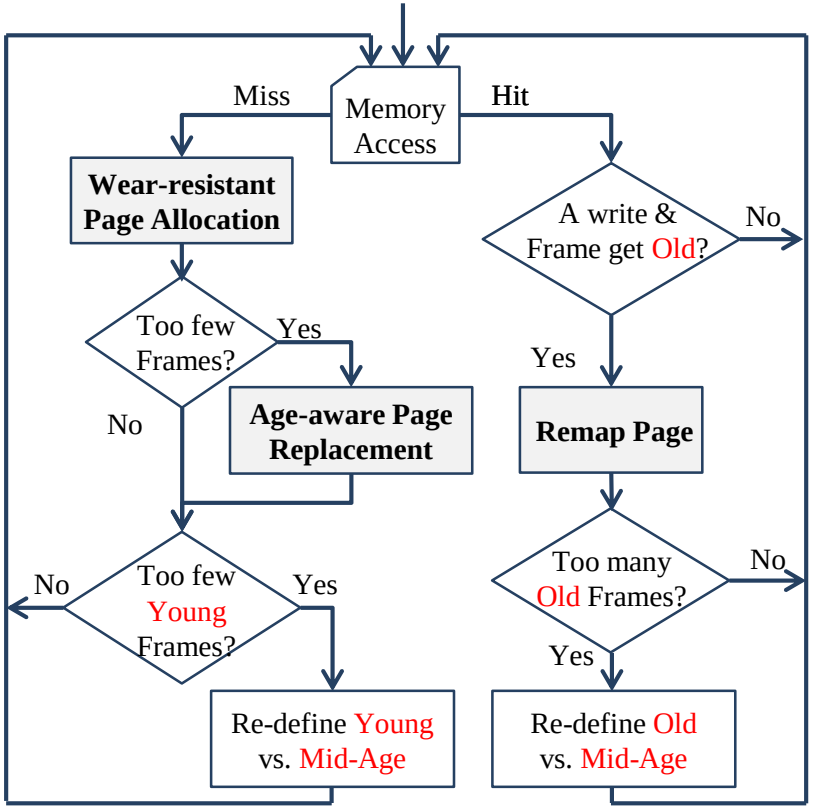
Age-aware free list





Hot is always served with **young**

System Overview



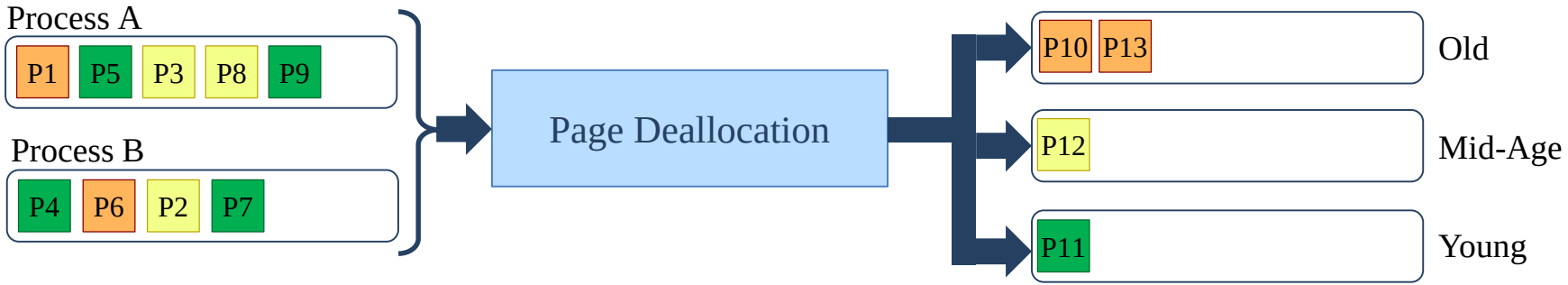
Following are added to the system:

- Three procedures
  - Wear-resistant page allocation
  - Age-aware page replacement
  - Remap hot page
- Two thresholds
  - Young-to-Midage threshold
  - Midage-to-Old threshold

Age-aware Page Deallocation

In OS there is an in-use list per process

Age-aware free list



How to select a page for deallocation?

- Constrained Clock Algorithm
- Constrain is the upper bound to the write count of page

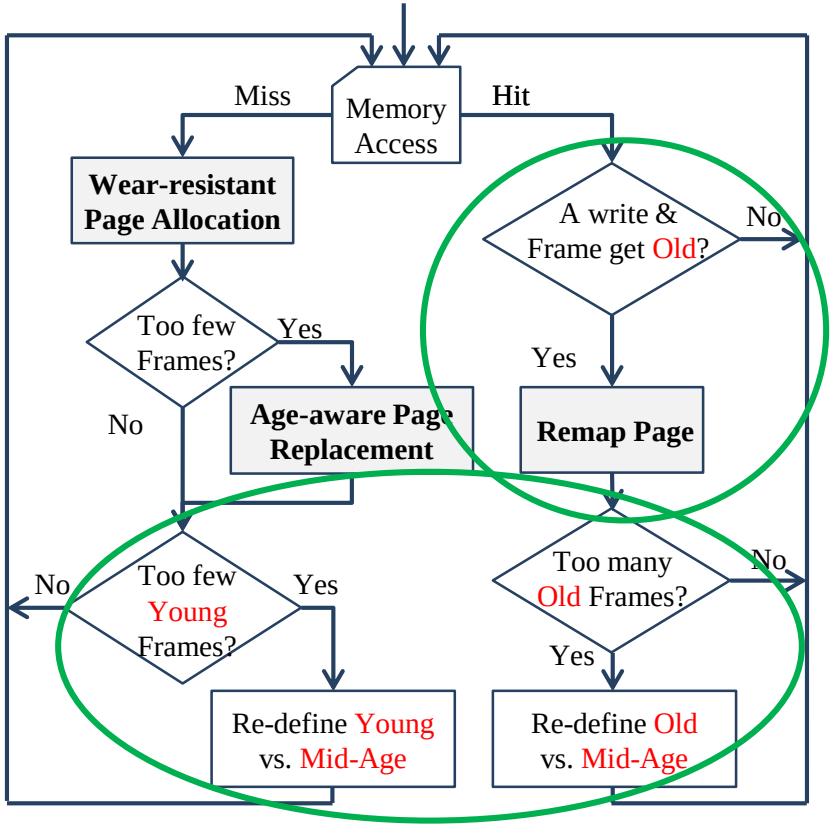
If the young list empty

upper bound = Y-to-M threshold

Else

upper bound = wear out limit

System Overview



- ✓ Effectively balance page write counts across processes
- ✓ Remap is only needed to handle an extremely hot virtual page
- ✓ Redefining the two thresholds across PCM lifetime

Evaluation: Methodology

- **Benchmarks:** SPEC 2000/2006. Memory traces collected with ***Pin***.
- **PCM main memory:** 128MB with 10^6 wear out limit per cell.
- **Y-to-M threshold:** 10^4 initially.
- **M-to-O threshold:** 2×10^5 initially.

Benchmarks	
Group1	leslie3d, omnetpp, vpr
Group2	gzip, gammes, hmmer
Group3	calculix, facerec, fma3d
Group4	gcc, gromacs, milc

Compare with

no-WL	Start Gap [1]	Random Swap [2]	Segment Swap [3]
No wear leveling	• Maintain an extra empty line • Every 50 writes, swap the empty line with the next line	• Every 512 writes, swap the last written page with a randomly selected page	• Group pages into 1MB segment. • Swap the hottest and coldest segments every 2×10^5 writes

[1] M. Qureshi, et al, "Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling," in MICRO, 2009.
[2] A. Ferreira, et al, "Increasing PCM main memory lifetime," in DATE, 2010.
[3] P. Zhou, et al, "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," in ISCA, 2009.

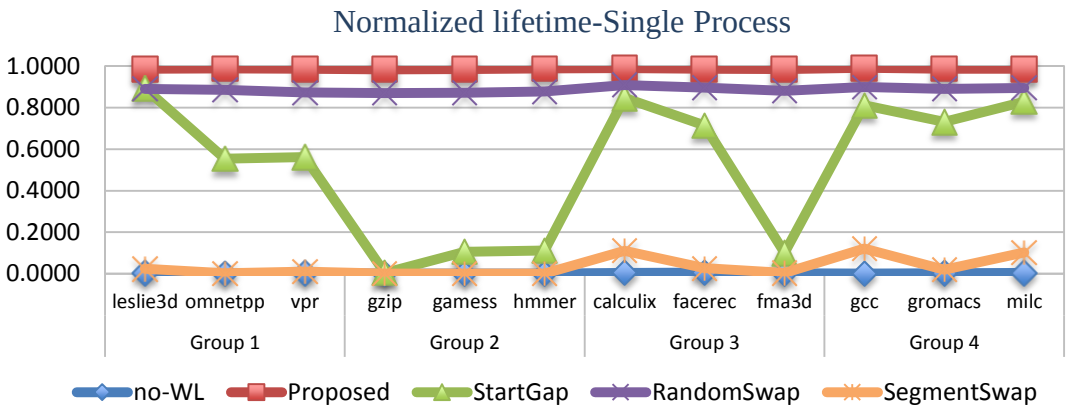
Lifetime Improvement

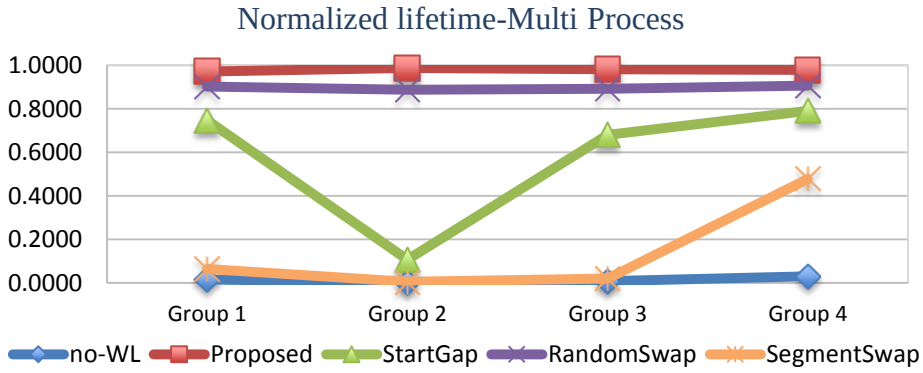
Normalized Lifetime:

$$\frac{Total\ LLC\ Writes}{Wear\ out\ limit \times Total\ pages}$$

- No-WL only explores 1% of potential writes.
- Proposed scheme explores more than 98% of PCM writes.
- Other schemes can only explore up to 89% of writes.

It is mostly due to extra remaps.





Overhead

Overhead:

$$\frac{(Total\ Page\ Faults + Total\ Remaps) \times 128}{Total\ LLC\ Writes}$$

$$\frac{Page\ Size}{LLC\ Block\ Size} = \frac{2^{12}}{2^{5}}$$

Benchmark	no-WL	Proposed	StartGap	Random Swap	Segment Swap
Group1	2%	7%	145%	37%	34%
Group2	21%	10%	179%	58%	45%
Group3	2%	3 %	119%	123%	33%
Group4	2%	9%	141%	51%	34%

- No-WL overhead is due to page faults.
- Proposed scheme imposes at most 10% extra writes.
- StartGap impose the most extra writes
  - High remap frequency

Outline

- Introduction
- Prolonging PCM Limited Lifetime
- Addressing PCM Write Latency and Energy
- Reducing DWM Access Latency
- Summary and Future Works

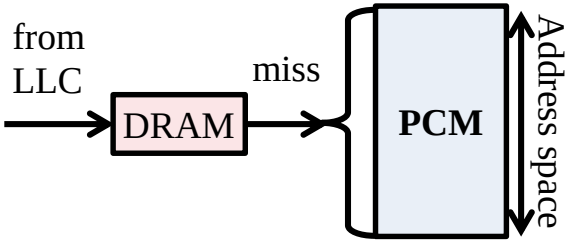
DRAM vs PCM

	DRAM cell	PCM cell	What if
Read Lat	~10	~10	
Read Ene	4.4	2.5	
Write La	~10	~100	
Write En	5.5	14(set)~20(reset)	
Write En	10^{16}	10^{6}~10^{9}	
Leakage Power	High	Low	
Scalability	Not belov	Predicted 9nm	9nm

One attractive solution is a ***DRAM-PCM hybrid architecture***

DRAM-PCM Hybrid Structures

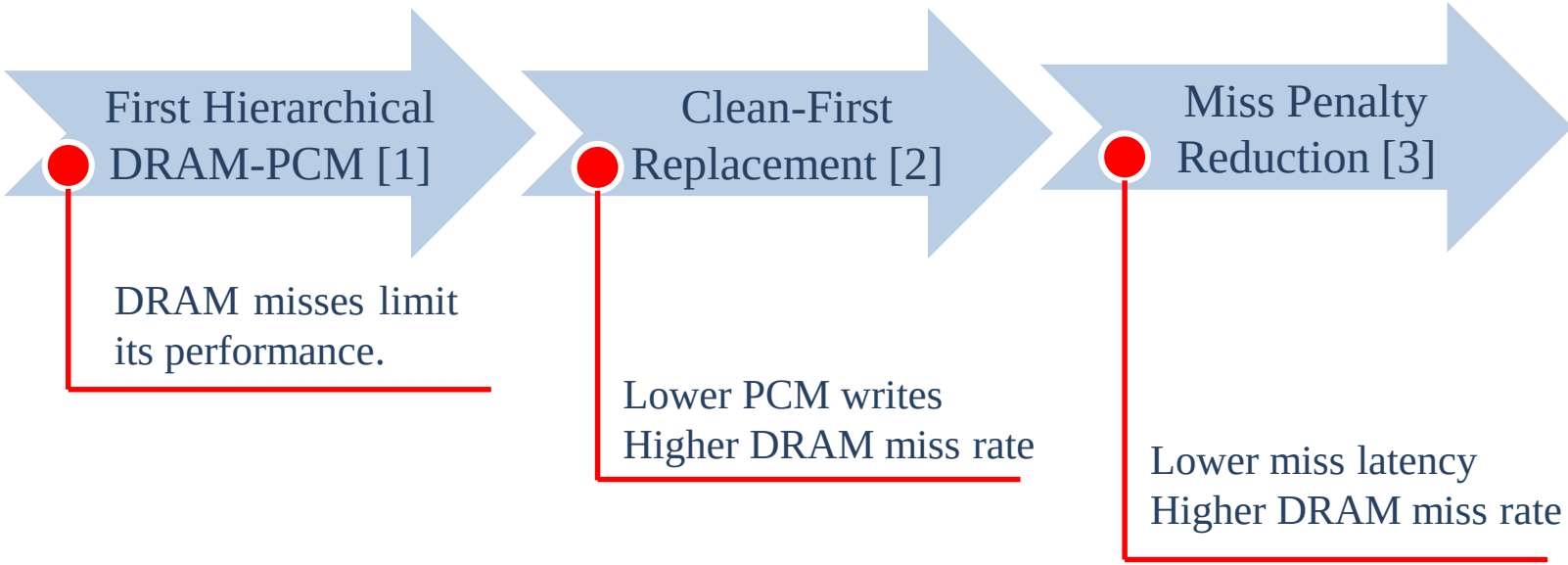
- Use a hierarchical organization (DRAM is a cache for PCM)
- DRAM is size is 3% of PCM size
- Access policy is two steps, first access DRAM, if miss, access PCM



Hierarchical organization

Problem!	Two-step access strategy degrades memory performance

Related work on Hierarchical Architecture



[1] M. Qureshi, et al, "Scalable High Performance Main Memory System Using Phase-Change Memory Technology," in ISCA, 2009.

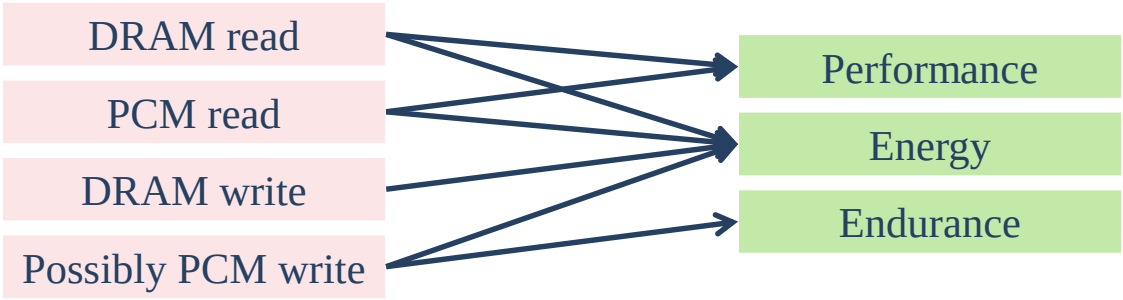
[2] A. Ferreira, et al, "Increasing PCM main memory lifetime," in DATE, 2010.

[3] H. G. Lee, et al, "An Energy- and Performance-Aware DRAM Cache Architecture for Hybrid DRAM/PCM Main Memory Systems," in ICCD, 2011.

Criticality of DRAM Misses

Operations upon DRAM miss:

System metrics:

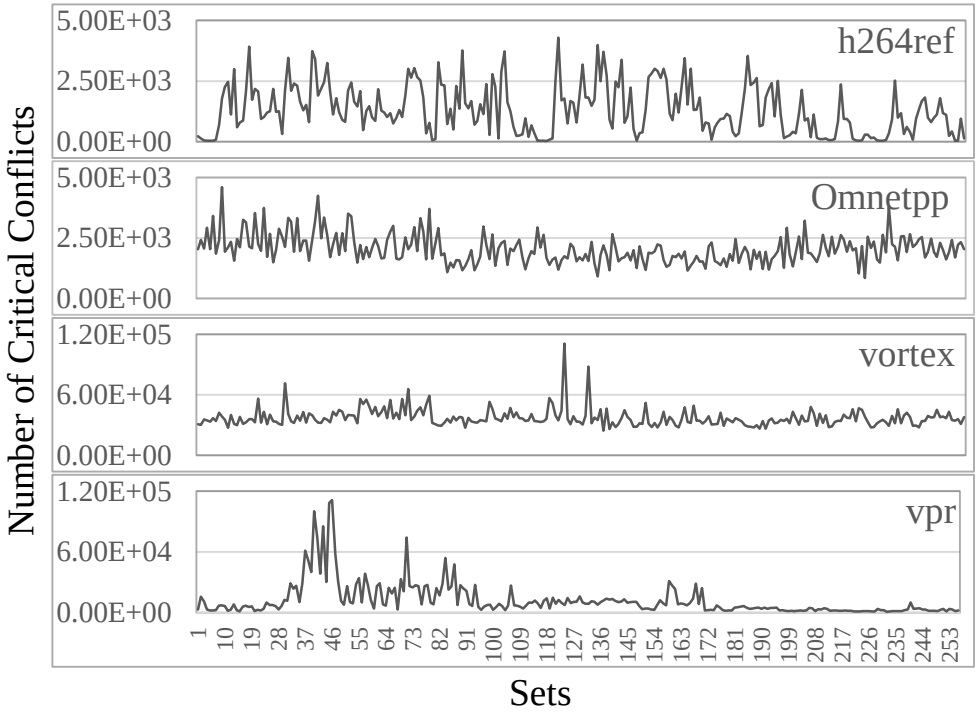


Goal: Reduce DRAM misses, more specifically those misses that generate PCM writes

Distribution of Conflict Misses

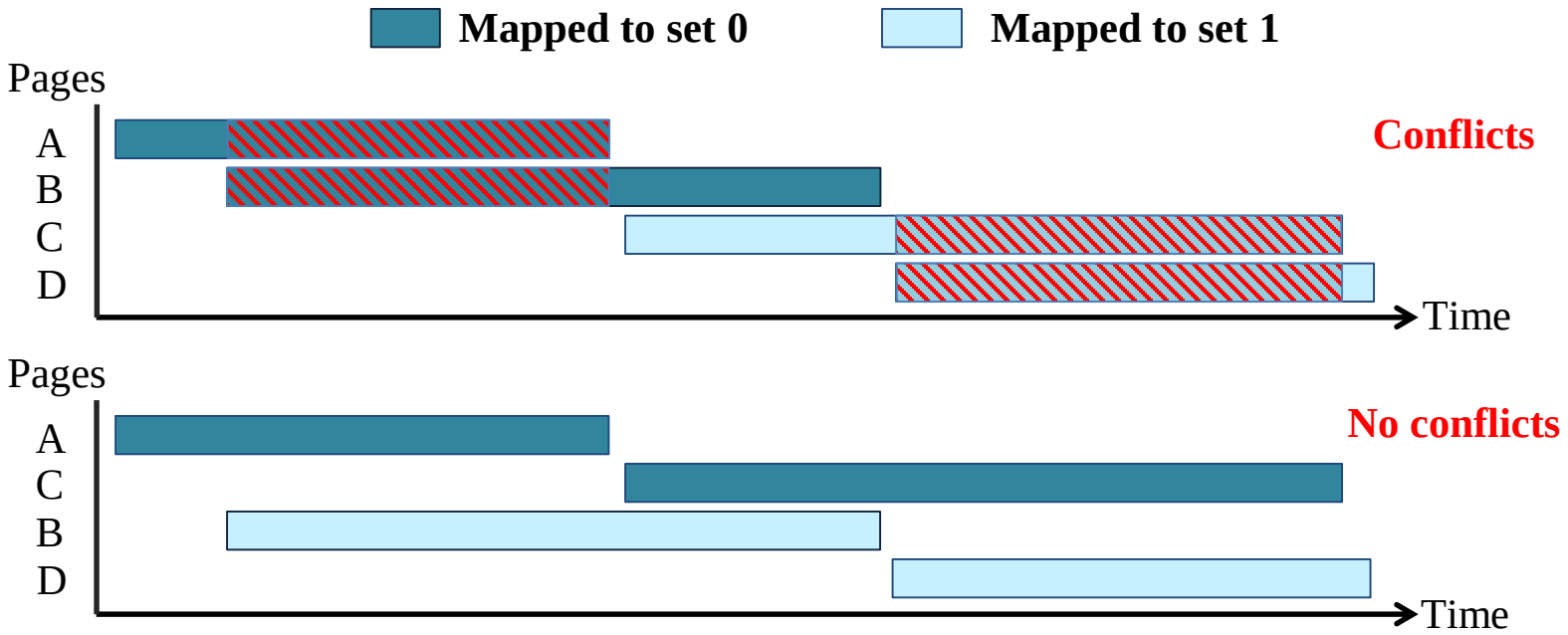
- DRAM Conflict Misses:
  - Hit in PCM
- Critical Conflicts:
  - Generates writeback to PCM

High variation across DRAM sets!!



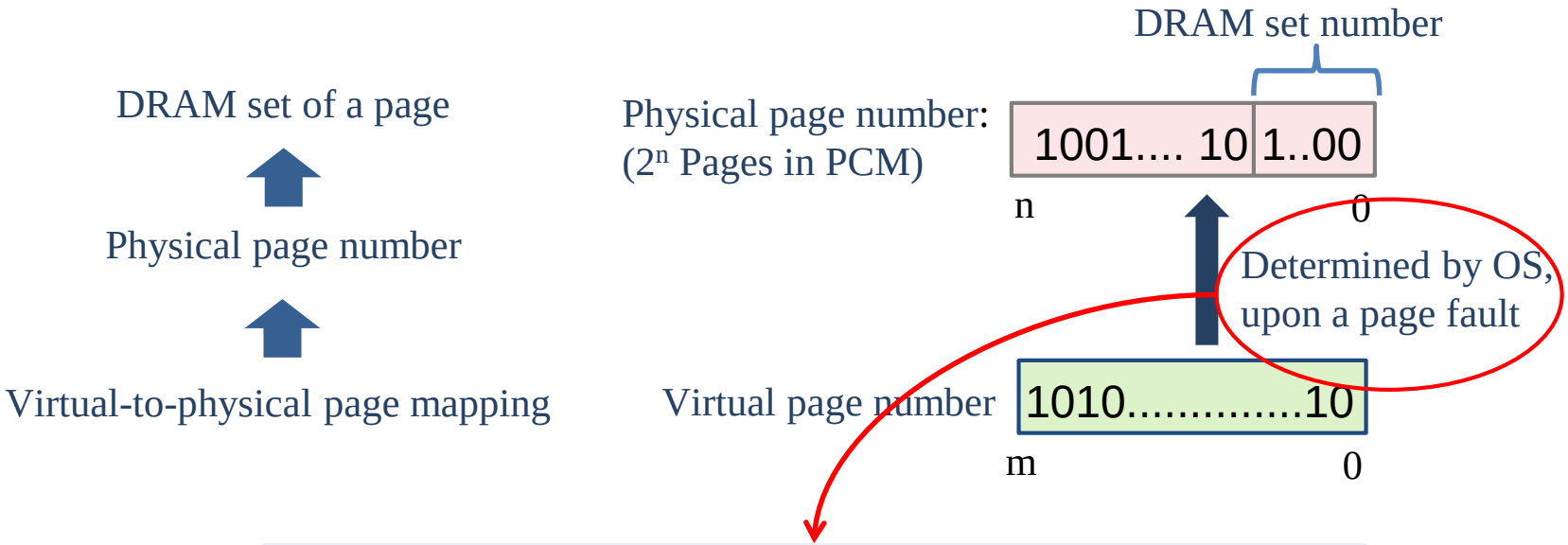
A Motivating Example

Consider a directly mapped DRAM with 2 sets



Idea: when bring a page to DRAM, map it to a less conflicting set

How the DRAM Set is Determined?



Exploiting the flexibility of virtual-to-physical page mapping

How to identify a less conflicting set?

A counter per set is used.

Size of Counter

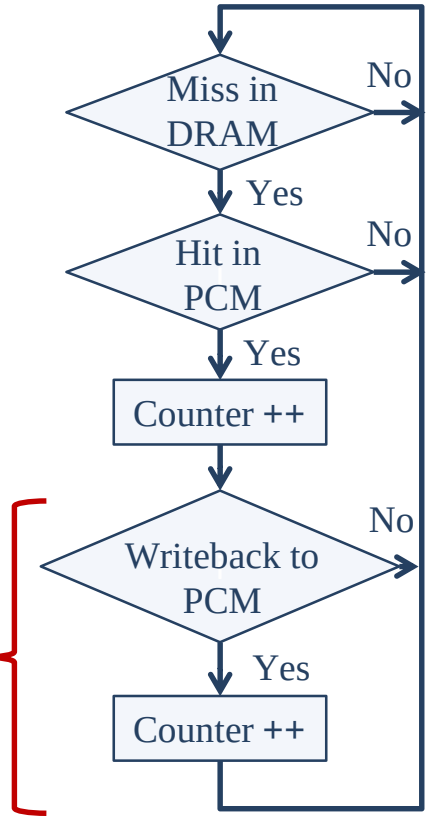
Higher Overhead
(Power and Storage)

Lower Accuracy

2-bit saturating counters per DRAM set.

For a 2MB, 4-way associative DRAM, storage overhead is 256 bits out of 2MB **(< 0.01%)**

In parallel with data transfer, no performance overhead



A higher priority is given to *critical conflicts*

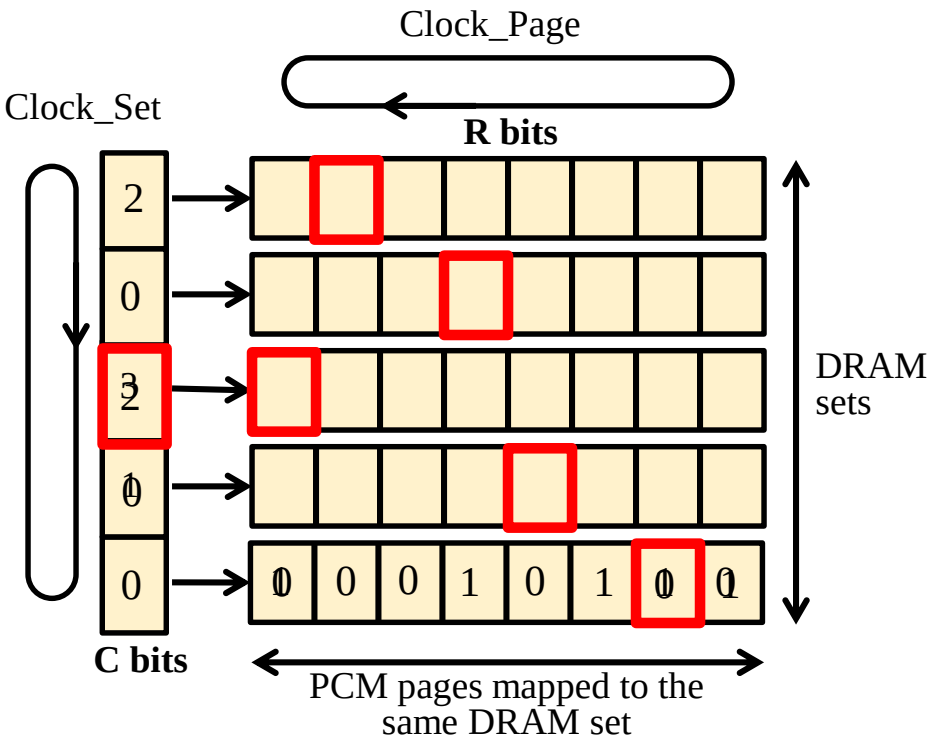
Look for a Page More Efficiently

Previously: Reference bits of **all** in-use pages are searched.

Proposed Algorithm:

- Group pages of the same DRAM set into a list.
- Search only the list associated with the selected DRAM set.

n = total number of in-use pages,
m = total number of DRAM sets
Complexity of conventional clock = O(n)
Complexity of our algorithm = O(m) + O(n/m)



Evaluation: Methodology

- **Benchmarks:** SPEC 2000/2006. Memory traces collected with ***Pin***.
- **Last Level Cache (LLC):** 1MB, 128B Block, and 4-way associative
- **PCM main memory:** 1 GB, page size of 4KB
- **DRAM Cache:** 32MB, 4-way associative

Compare with

Baseline [1]	N-Chance [2]	Reduced Miss Penalty [3]
Basic hierarchical DRAM-PCM	Clean first replacement in DRAM	Maintain an empty block per set in DRAM

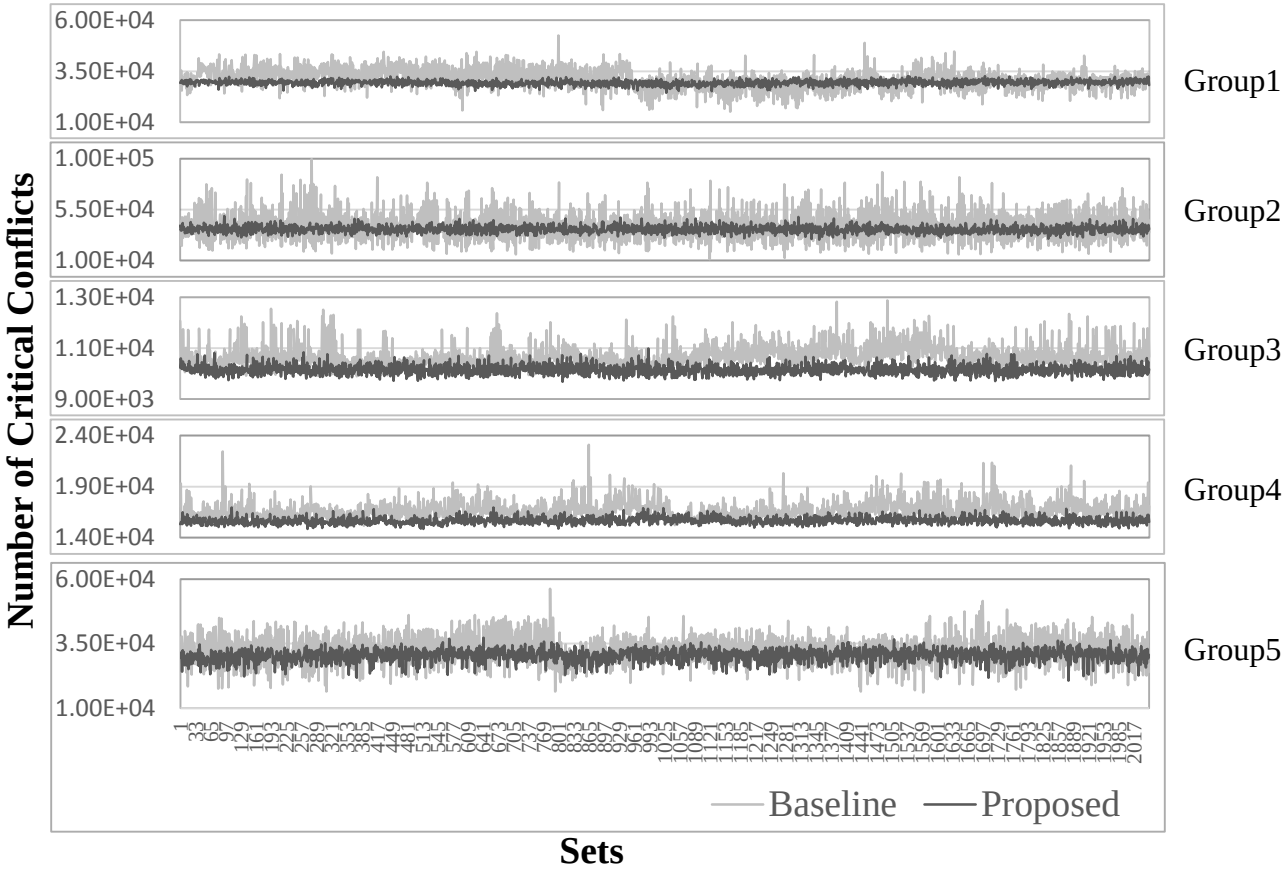
[1] M. Qureshi, et al, "Scalable High Performance Main Memory System Using Phase-Change Memory Technology," in ISCA, 2009.
[2] A. Ferreira, et al, "Increasing PCM main memory lifetime," in DATE, 2010.
[3] H. G. Lee, et al, "An Energy- and Performance-Aware DRAM Cache Architecture for Hybrid DRAM/PCM Main Memory Systems," in ICCD, 2011.

Groups of Benchmarks

Benchmarks are grouped to simulate multi-process environment

	Benchmarks
Group1	gromacs, h264ref, hmmer, leslie3d, omnetpp, sjeng, tonto, zeusmp
Group2	ammp, gromacs, leslie3d, mgrid, milc, sjeng, tonto, wupwise
Group3	bwaves, gzip, leslie3d, milc, sjeng, vortex, wupwise, zeusmp
Group4	facerec, gzip, h264ref, mgrid, omnetpp, tonto, vpr, wupwise
Group5	facerec, gamess, gcc, gzip, h264ref, vortex, vpr, zeusmp

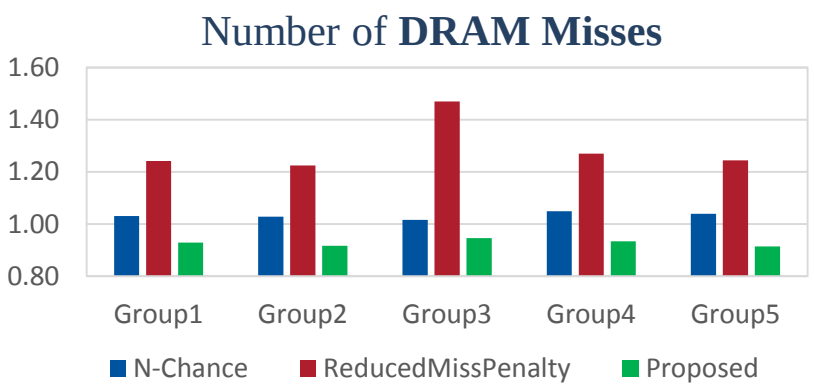
Balanced Critical Conflicts

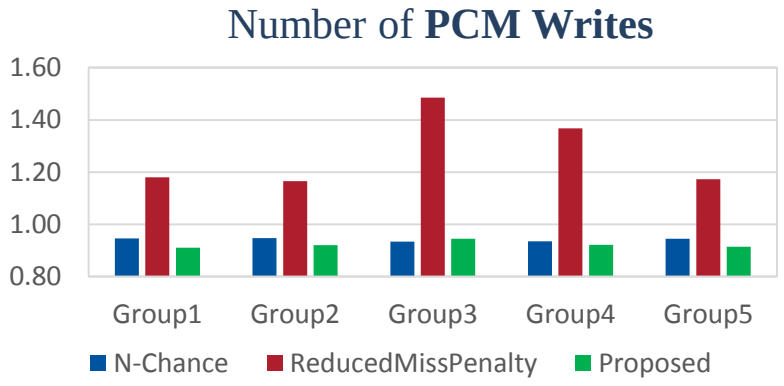


Reduced standard deviation by 64.78%

Results- DRAM misses and PCM writes

All values are normalized to the baseline
Baseline is basic hierarchical DRAM-PCM





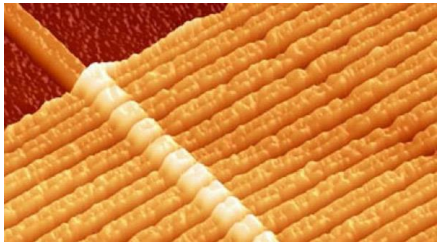
- *Proposed scheme* reduces both DRAM misses and PCM writes
  - 7% reduction in DRAM misses
  - 6% reduction in PCM writes
- *N-Chance* only reduces PCM writes but induces more misses
- *Reduced miss penalty* increases both.

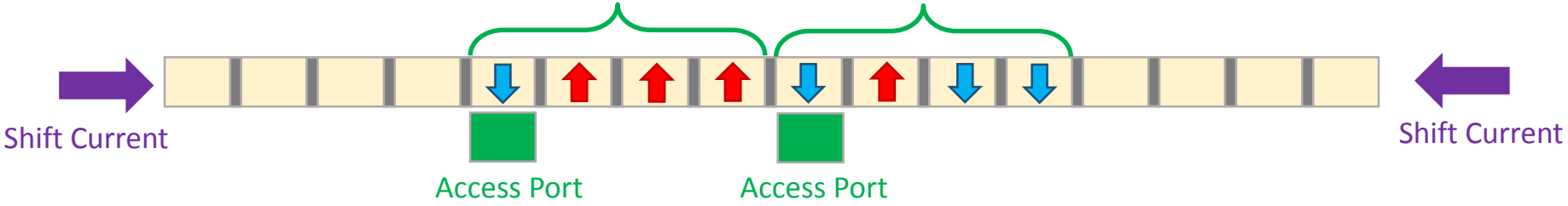
Outline

- Introduction
- Prolonging PCM Limited Lifetime
- Addressing PCM Write Latency and Energy
- Reducing DWM Access Latency
- Summary and Future Works

Domain Wall Memory (DWM)

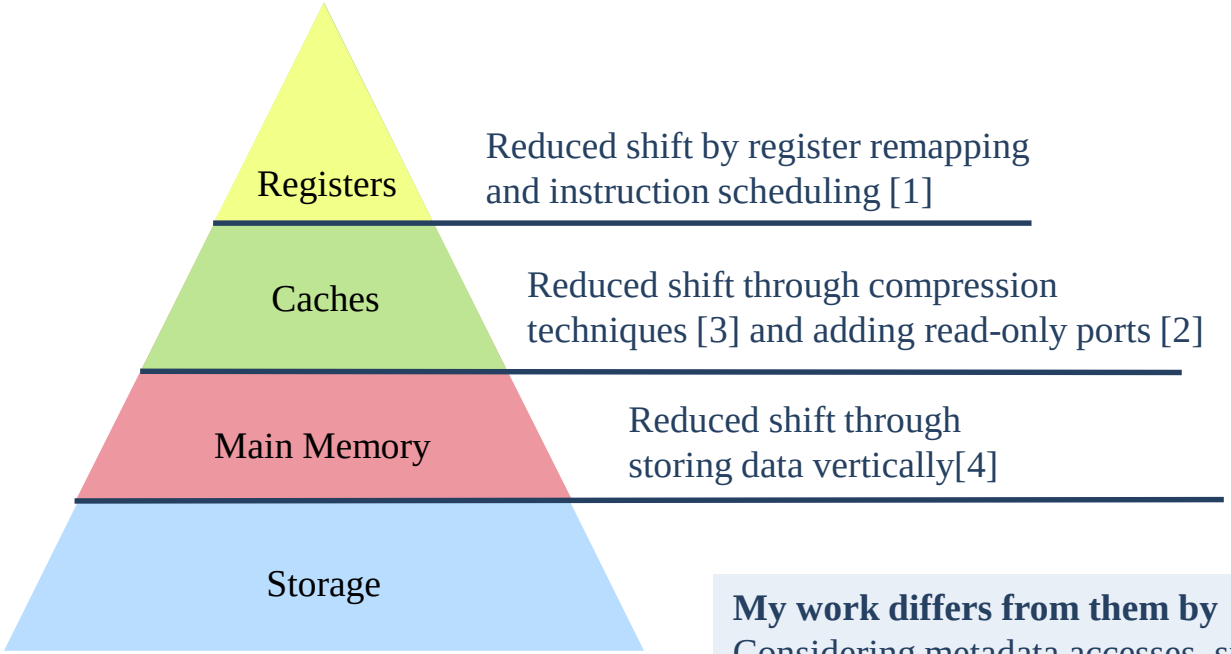
- Made of hundreds of millions of ferromagnetic nanowires.
- Each nanowire stores many bits or domains.
- To read/write data several access ports are placed at fixed position.
- Shift operation required to access each bit.





Access latency and energy is affected by **shift operation**.

Related Work on DWM



Memory Hierarchy

[1] M. Mao, et al, "Exploration of GPGPU Register File Architecture Using Domain-wall-shift-write based Racetrack Memory," in DAC, 2014.

[2] R. Venkatesan, et al, "TapeCache: A High Density, Energy Efficient Cache Based on Domain Wall Memory," in ISLPED, 2012.

[3] H. Xu, et al, "Multilane Racetrack Caches: Improving Efficiency through Compression and Independent Shifting," in ASP-DAC, 2015.

[4] Q. Hu, et al, "Exploring Main Memory Design Based on Racetrack Memory Technology," in GLVLSI, 2016.

My work differs from them by

Considering metadata accesses, specifically page table.

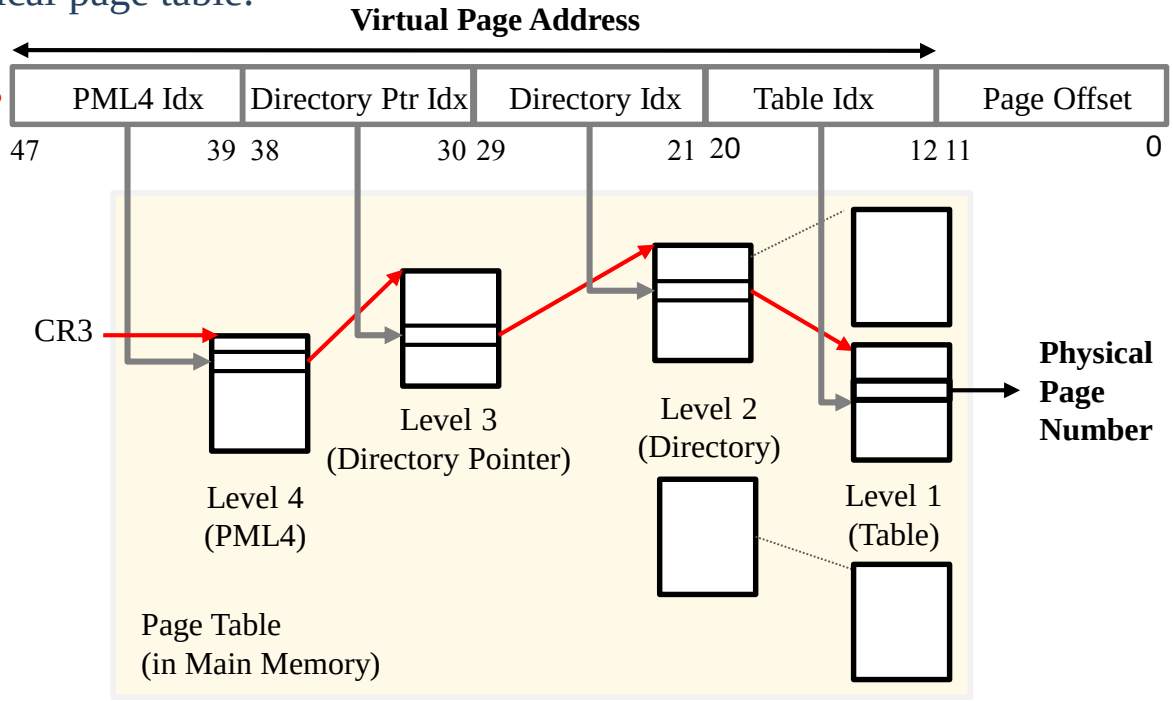
(1) Reducing shifts

(2) Leveraging access port position for metadata interpretation

Page Table

- Contains the virtual-to-physical page address mapping.
- Modern systems adopt a hierarchical page table.

The latency of this 4-step process is 50% of total execution time!!! [1]

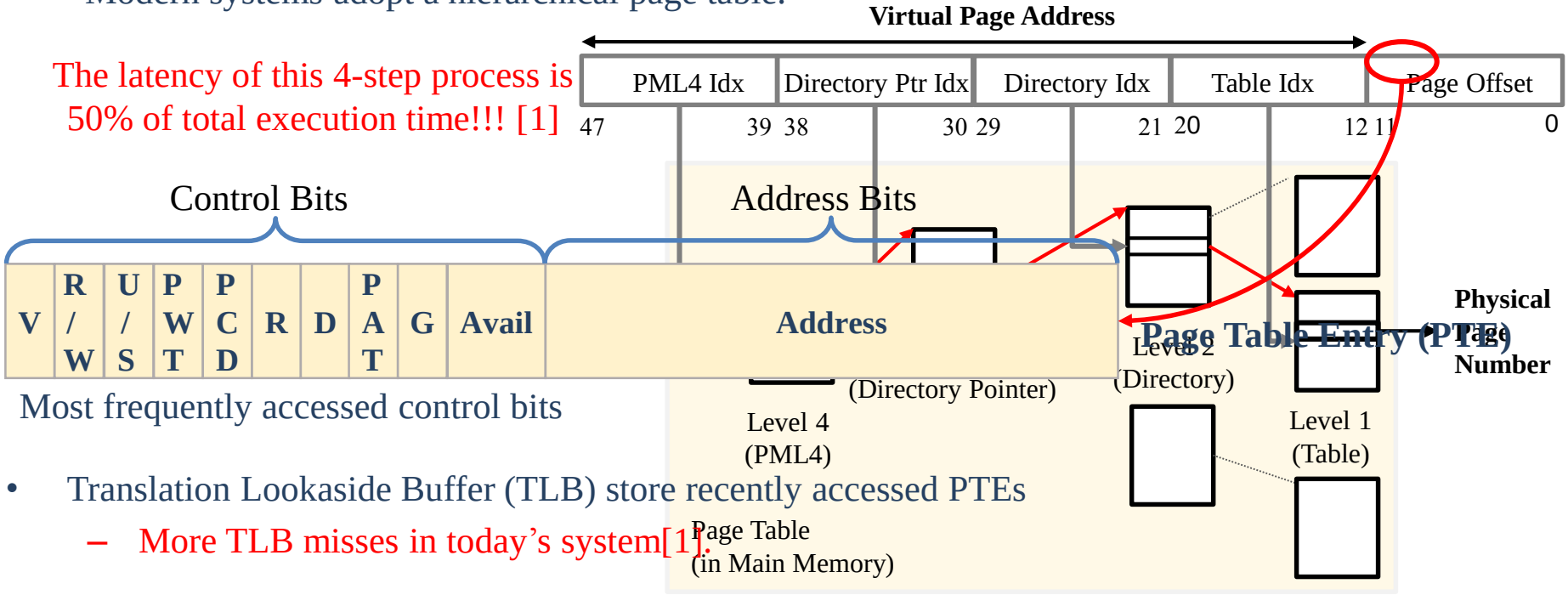


[1] V. Karakostas, et al, "Redundant Memory Mappings for Fast Access to Large Memories," in ISCA, 2015.

Page Table

- Contains the virtual-to-physical page address mapping.
- Modern systems adopt a hierarchical page table.

The latency of this 4-step process is 50% of total execution time!!! [1]



- Translation Lookaside Buffer (TLB) store recently accessed PTEs
  - More TLB misses in today's system[1].

[1] V. Karakostas, et al, "Redundant Memory Mappings for Fast Access to Large Memories," in ISCA, 2015.

Intuitive Read/Write a PTE

Currently upon access to page table

Aligning the first bit to the access port.

Read/write the whole entry.

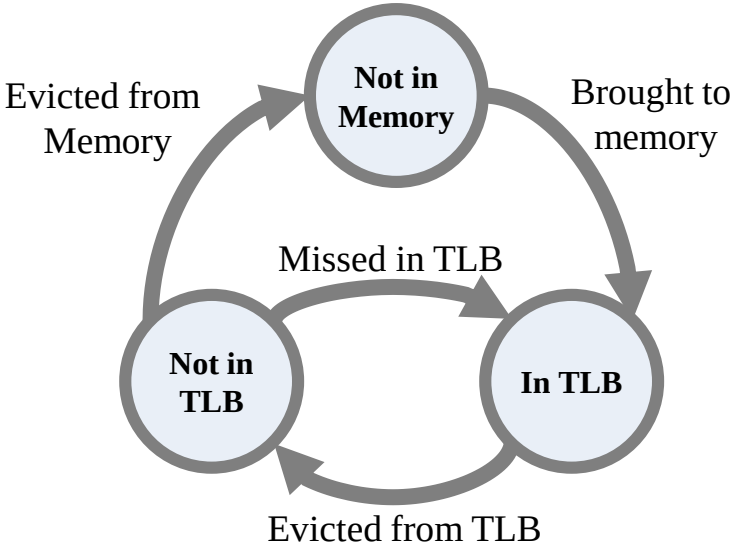
$$\#of\ Shifts\ = 2\ \times MaxShift$$

$MaxShift =$ distance between two neighbouring port

My observation: There is no need to read the entire PTE

Observation on PTE States

In system with TLB, PTE have three states



	TLB	Main memory
Not in Memory	✗	✗
In TLB	✓	✓
Not in TLB	✗	✓

Transition	Action	Accessed Field in PTE
Not in Memory → In TLB	Write	The entire entry
In TLB → Not in TLB	Update	Referenced (R) bit and/or Dirty (D) bit
Not in TLB → In TLB	Read	The entire entry
Not in TLB → Not in Memory	Update	Validity (V) bit

An Example

Necessary Shift: Shifting through the bits that are listed in table.

Extra Shift: Shifting through the bits that are **not** listed in table

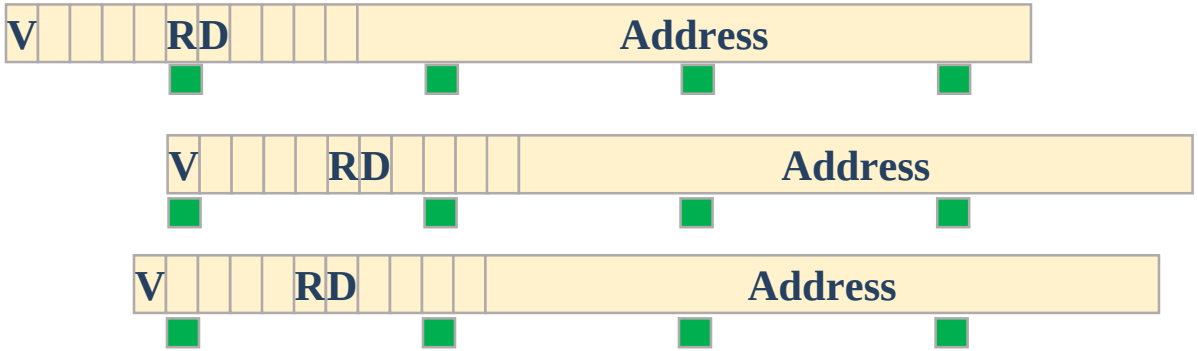
Alignment Shift: Align first bit with access port.

Transition	Action	Accessed Field in PTE
In TLB → Not in TLB	Update	Referenced (R) bit and/or Dirty (D) bit

Pre-Alignment Technique

Pre-align based on the state:

- **In TLB:** Pre-align to R bit
- **Not in TLB:** Pre-align to V bit
- **Not in Memory:** Pre-align to the bit adjacent to the V bit



Two advantages:

(1) Remove the alignment shifts from the PTE access path

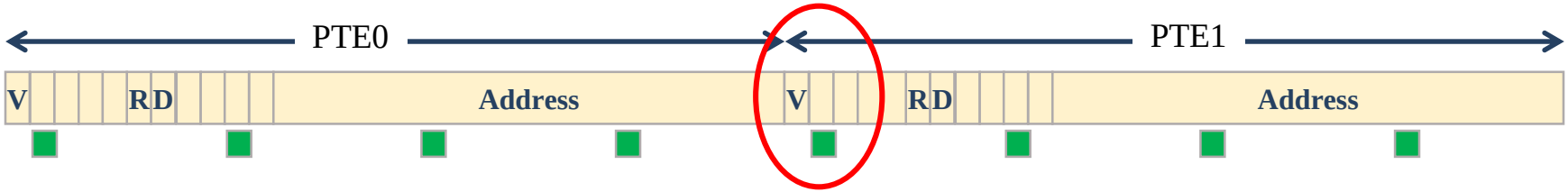
(2) The state of PTE is known prior to reading PTE

Transition	Action	Accessed Field in PTE
Not in Memory → In TLB	Write	The entire entry
In TLB → Not in TLB	Update	Referenced (R) bit and/or Dirty (D) bit
Not in TLB → In TLB	Read	The entire entry
Not in TLB → Not in Memory	Update	Validity (V) bit

Placement – Motivating example

Multiple PTEs can share a track.

But it can create Conflict of interest!!!



At the beginning

PTE0 → Not In Memory ➡ PTE0 → In TLB

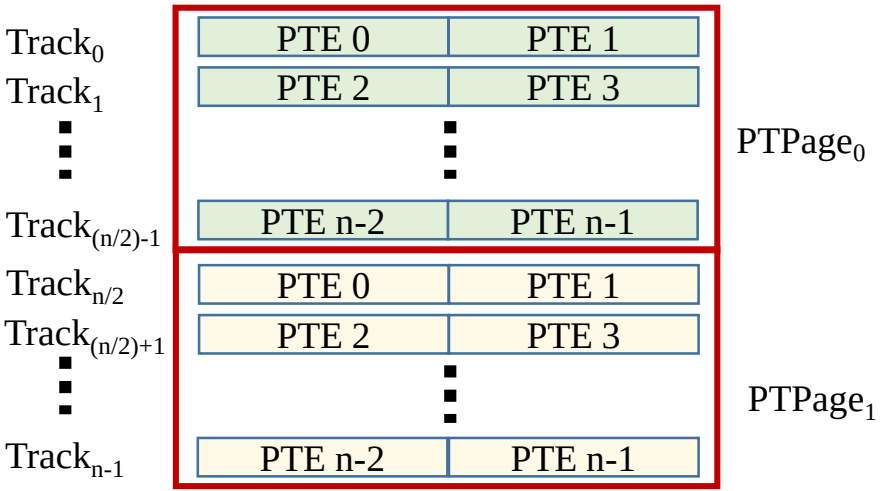
PTE1 → Not In Memory

Idea: Place PTEs with minimum conflict on the same track

Placement

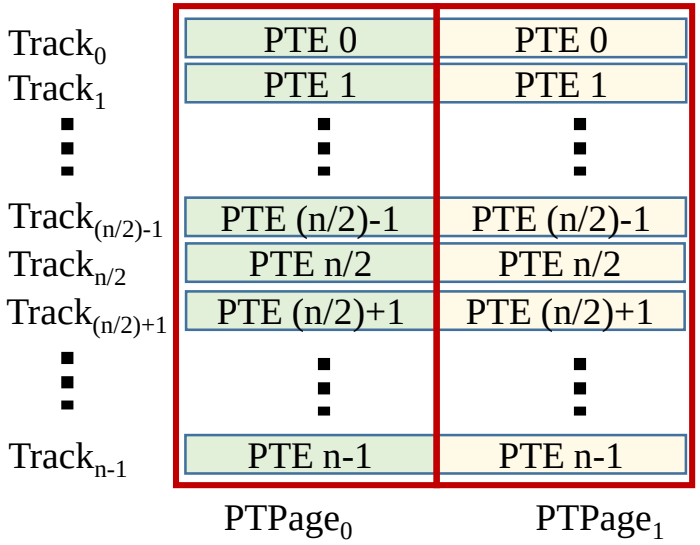
PTEs are stored in granularity of pages. (**PTpage**)

Horizontal Placement



PTEs in the same page have high spatial locality ⇒ highly conflicting

Vertical Placement



Less spatial locality between PTEs of different pages ⇒ less conflicting

Evaluation: Methodology

- **Benchmarks:** SPEC 2000/2006. TLB traces collected with ***Pin***.
- **Instruction and Data TLB:** 4-way associative 128-entry
- **PCM main memory:** 4 GB, page size of 4KB
- **Page Table Reserved Size:** 2MB

Compare with

DWM-based baseline	Traditional DRAM-based baseline
• DWM main memory. • No pre-alignment. • Horizontal placement.	• DRAM main memory.

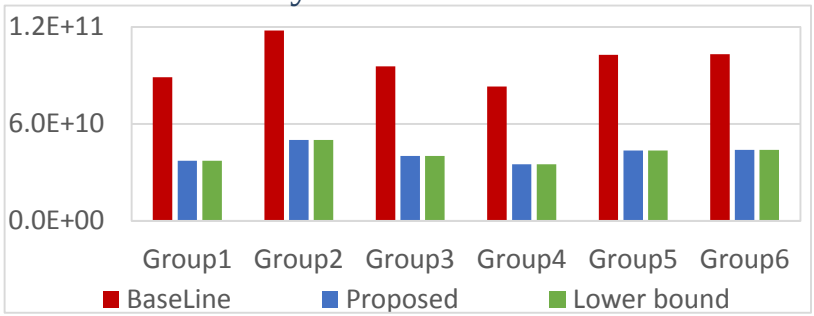
Groups of Benchmarks

Benchmarks are grouped to simulate multi-process environment

	Benchmarks
Group1	gemsFDTD, gobmk, gromacs, h264ref
Group2	astar, bwaves, bzip, cactus
Group3	calculix, deal, gamess, gcc
Group4	hmmer, zeusmp, leslie3d, libquantum
Group5	mcf, milc, namd, omnetpp
Group6	perlbench, sjeng, soplex, tonto

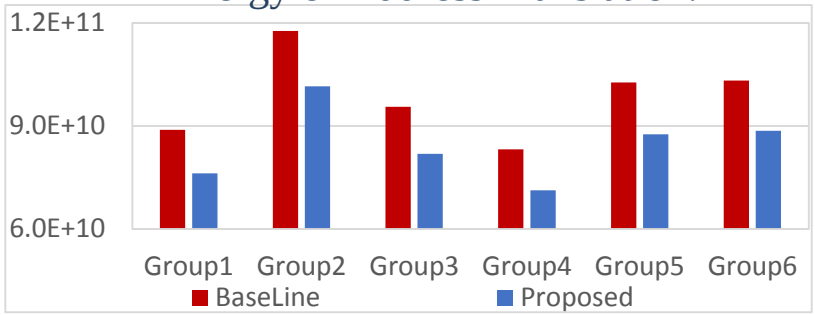
Comparison with DWM-based Baseline

Latency of Address Translation.



69% reduction

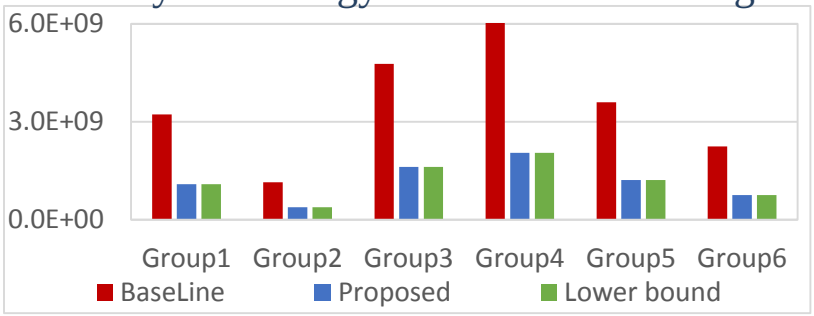
Energy of Address Translation.



15% reduction

Lower bound = necessary shifts only

Latency and Energy of Context Switching.



71% reduction

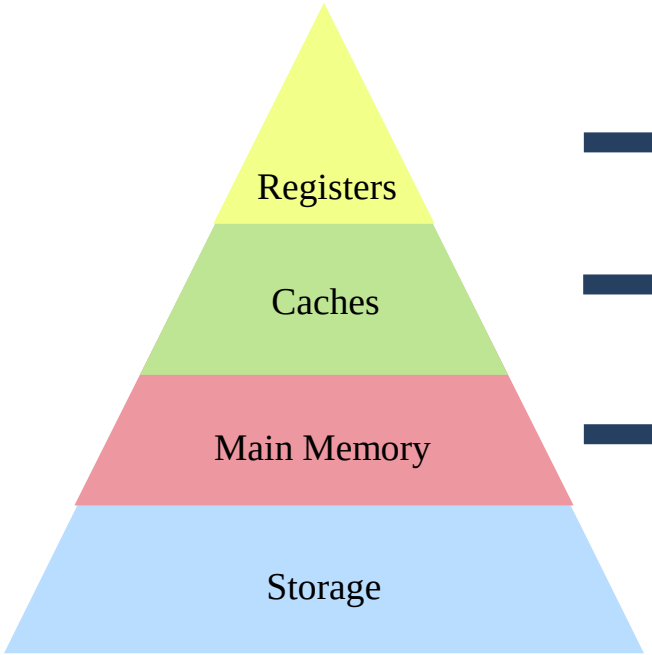
Outline

- Introduction
- Prolonging PCM Limited Lifetime
- Addressing PCM Write Latency and Energy
- Reducing DWM Access Latency
- Summary and Future Works

Summary

- Showed the necessity to change main memory technology.
- Introduced two candidates:
  - Phase Change Memory (PCM)
  - Domain Wall Memory (DWM)
- Prolonged PCM main memory lifetime
  - Proposed a segment-aware page allocation.
- Overcame write limitation of PCM
  - Proposed a conflict-aware page allocation for PCM-DRAM hybrid main memory.
- Improved the performance and energy page table accesses
  - Proposed a pre-alignment technique as well as a new placement in DWM main memory.

Future Work



Memory Hierarchy

DWM-based Stack Architecture

STT-RAM-based Cache

PCM-based Main memory
DWM-based Main memory

Future Work – DWM-based Stack Architecture

Energy harvesting devices become popular

Concerns:	Solutions:
(1) Intermittent power supply	Non-volatile memory register files [1]
(2) Tend to be smaller everyday	Alternative architecture such as Stack architecture

DWM is a good choice for stack.

My proposal

Goal: Improving the reliability, energy, and execution time of energy harvesting devices

[1] K. Ma, et al, "Nonvolatile Processor Architecture Exploration for Energy-Harvesting Applications," in MICRO, 2015.

Future Work – STT-RAM-based Cache

- Over 50% of chip area devoted to cache.
- STT-RAM is more scalable than SRAM.
- STT-RAM has long write latency and high write energy,
  - however they have a trade of with non-volatility property

[Jog, DAC'12]

Retention Time	10 years	1 sec	10 ms
Write Latency @2GHz	22 cycles	12 cycles	6 cycles

Goal: Explore the flexibility of cache replacement algorithm to reduce retention time and improve access latency and energy.

Thank You!
Q&A