18-740/640
Computer Architecture
Lecture 9: Emerging Memory Technologies

Prof. Onur Mutlu
Carnegie Mellon University
Fall 2015, 9/30/2015

Required Readings

- Required Reading Assignment:
  - **Lee et al., "Phase Change Technology and the Future of Main Memory," IEEE Micro, Jan/Feb 2010.**
- Recommended References:
  - M. Qureshi et al., "Scalable high performance main memory system using phase-change memory technology," ISCA 2009.
  - H. Yoon et al., "Row buffer locality aware caching policies for hybrid memories," ICCD 2012.
  - J. Zhao et al., "FIRM: Fair and High-Performance Memory Control for Persistent Memory Systems," MICRO 2014.

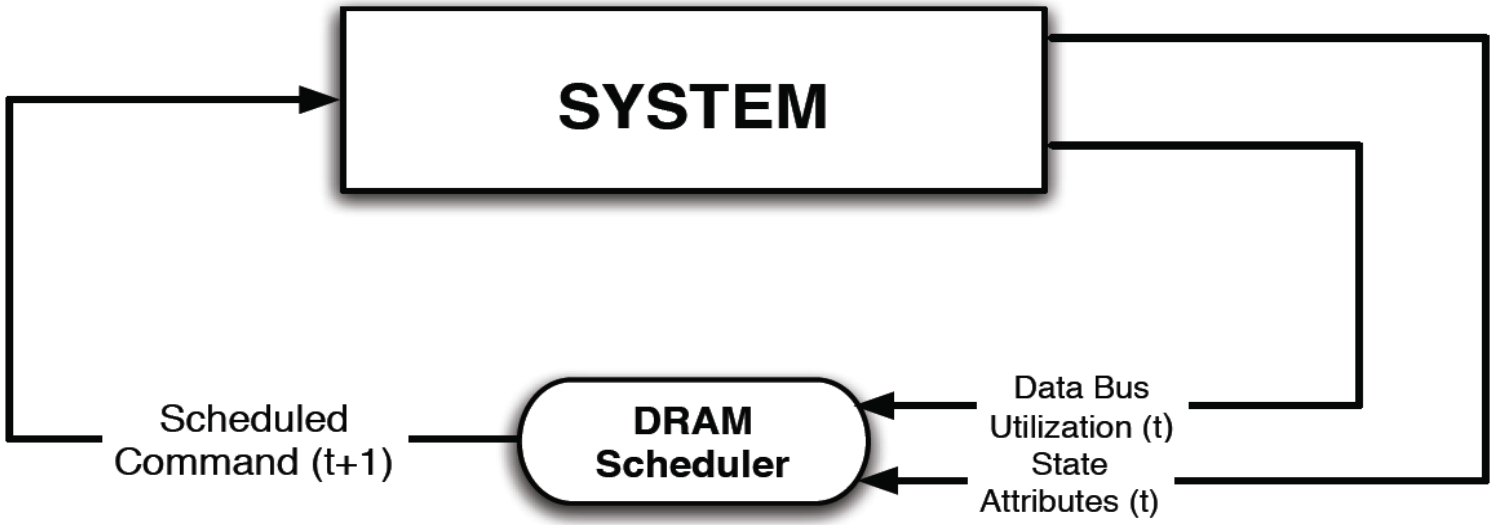
Self-Optimizing DRAM Controllers

- Problem: DRAM controllers difficult to design → It is difficult for human designers to design a policy that can adapt itself very well to different workloads and different system conditions
- Idea: Design a memory controller that adapts its scheduling policy decisions to workload behavior and system conditions using machine learning.
- Observation: Reinforcement learning maps nicely to memory control.
- Design: Memory controller is a reinforcement learning agent that dynamically and continuously learns and employs the best scheduling policy.

Ipek+, "Self Optimizing Memory Controllers: A Reinforcement Learning Approach," ISCA 2008.

Self-Optimizing DRAM Controllers

- Dynamically adapt the memory scheduling policy via interaction with the system at runtime
  - Associate system states and actions (commands) with long term reward values: each action at a given state leads to a learned reward
  - Schedule command with highest estimated long-term reward value in each state
  - Continuously update reward values for <state, action> pairs based on feedback from system



Self-Optimizing DRAM Controllers

- Engin Ipek, Onur Mutlu, José F. Martínez, and Rich Caruana, **"Self Optimizing Memory Controllers: A Reinforcement Learning Approach"**
  *Proceedings of the 35th International Symposium on Computer Architecture (**ISCA**)*, pages 39-50, Beijing, China, June 2008.

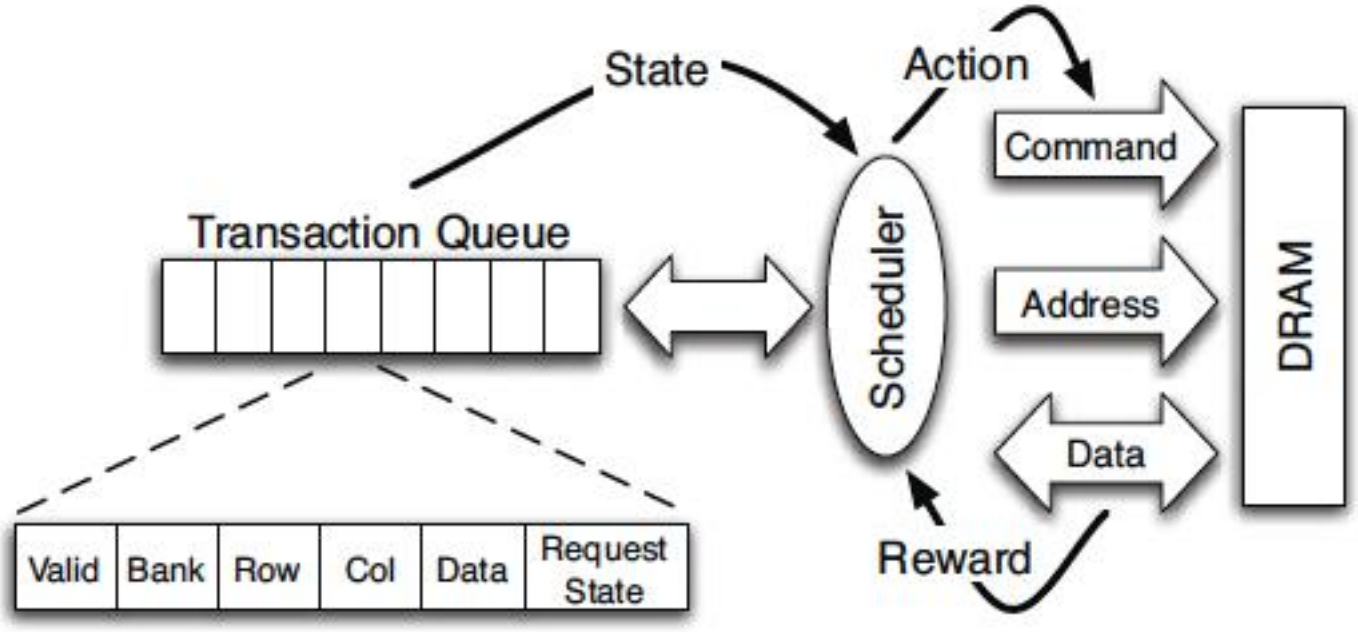


Figure 4: High-level overview of an RL-based scheduler.

States, Actions, Rewards

- Reward function
  - +1 for scheduling Read and Write commands
  - 0 at all other times

  Goal is to maximize data bus utilization
- State attributes
  - Number of reads, writes, and load misses in transaction queue
  - Number of pending writes and ROB heads waiting for referenced row
  - Request's relative ROB order
- Actions
  - Activate
  - Write
  - Read - load miss
  - Read - store miss
  - Precharge - pending
  - Precharge - preemptive
  - NOP

Performance Results

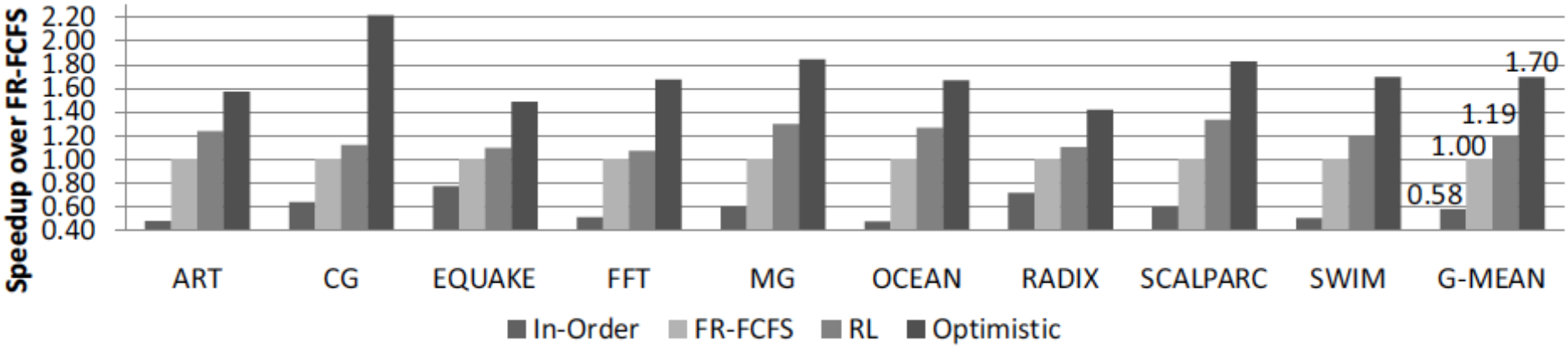


Figure 7: Performance comparison of in-order, FR-FCFS, RL-based, and optimistic memory controllers

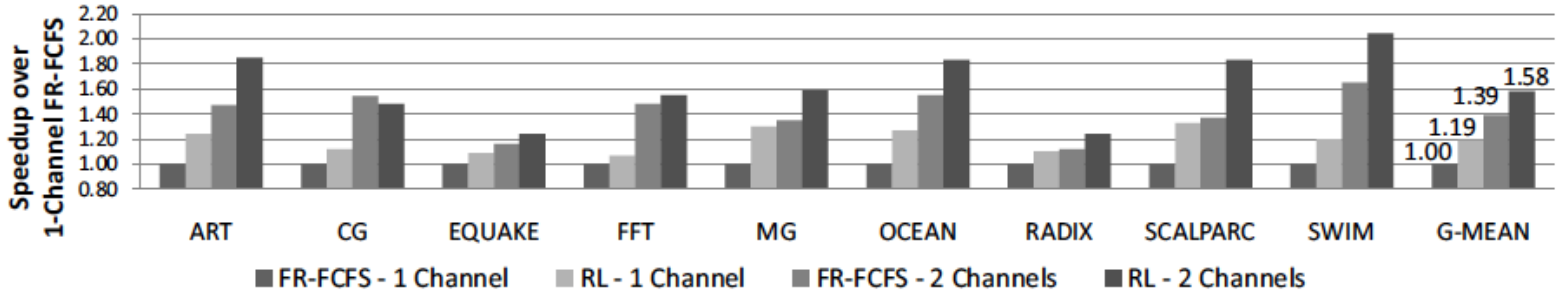


Figure 15: Performance comparison of FR-FCFS and RL-based memory controllers on systems with 6.4GB/s and 12.8GB/s peak DRAM bandwidth

Self Optimizing DRAM Controllers

- Advantages

  + Adapts the scheduling policy dynamically to changing workload behavior and to maximize a long-term target

  + Reduces the designer's burden in finding a good scheduling policy. Designer specifies:

    1) What system variables might be useful

    2) What target to optimize, but not how to optimize it

- Disadvantages and Limitations

  -- Black box: designer much less likely to implement what she cannot easily reason about

  -- How to specify different reward functions that can achieve different objectives? (e.g., fairness, QoS)

  -- Hardware complexity?

More on Self-Optimizing DRAM Controllers

- Engin Ipek, Onur Mutlu, José F. Martínez, and Rich Caruana, **"Self Optimizing Memory Controllers: A Reinforcement Learning Approach"**
  *Proceedings of the 35th International Symposium on Computer Architecture (**ISCA**)*, pages 39-50, Beijing, China, June 2008.

Self-Optimizing Memory Controllers: A Reinforcement Learning Approach

Engin İpek[1,2] **Onur Mutlu**[2] **José F. Martínez**[1] **Rich Caruana**[1]

[1]Cornell University, Ithaca, NY 14850 USA
[2] Microsoft Research, Redmond, WA 98052 USA

Evaluating New Ideas for Future (Memory) Architectures

Simulation: The Field of Dreams

Dreaming and Reality

- An architect is in part a dreamer, a creator
- Simulation is a key tool of the architect
- Simulation enables
  - The exploration of many dreams
  - A reality check of the dreams
  - Deciding which dream is better
- Simulation also enables
  - The ability to fool yourself with false dreams

Why High-Level Simulation?

- Problem: RTL simulation is intractable for design space exploration → too time consuming to design and evaluate
  - Especially over a large number of workloads
  - Especially if you want to predict the performance of a good chunk of a workload on a particular design
  - Especially if you want to consider many design choices
    - Cache size, associativity, block size, algorithms
    - Memory control and scheduling algorithms
    - In-order vs. out-of-order execution
    - Reservation station sizes, ld/st queue size, register file size, …
    - …
- Goal: Explore design choices quickly to see their impact on the workloads we are designing the platform for

Different Goals in Simulation

- Explore the design space quickly and see what you want to
  - potentially implement in a next-generation platform
  - propose as the next big idea to advance the state of the art
  - the goal is mainly to see relative effects of design decisions
- Match the behavior of an existing system so that you can
  - debug and verify it at cycle-level accuracy
  - propose small tweaks to the design that can make a difference in performance or energy
  - the goal is very high accuracy
- Other goals in-between:
  - Refine the explored design space without going into a full detailed, cycle-accurate design
  - Gain confidence in your design decisions made by higher-level design space exploration

Tradeoffs in Simulation

- Three metrics to evaluate a simulator
  - Speed
  - Flexibility
  - Accuracy

- Speed: How fast the simulator runs (xIPS, xCPS)
- Flexibility: How quickly one can modify the simulator to evaluate different algorithms and design choices?
- Accuracy: How accurate the performance (energy) numbers the simulator generates are vs. a real design (Simulation error)

- The relative importance of these metrics varies depending on where you are in the design process

Trading Off Speed, Flexibility, Accuracy

- Speed & flexibility affect:
  - How quickly you can make design tradeoffs
- Accuracy affects:
  - How good your design tradeoffs may end up being
  - How fast you can build your simulator (simulator design time)
- Flexibility also affects:
  - How much human effort you need to spend modifying the simulator
- You can trade off between the three to achieve design exploration and decision goals

High-Level Simulation

- Key Idea: Raise the abstraction level of modeling to give up some accuracy to enable speed & flexibility (and quick simulator design)

- Advantage

  + Can still make the right tradeoffs, and can do it quickly

    + All you need is modeling the key high-level factors, you can omit corner case conditions

    + All you need is to get the "relative trends" accurately, not exact performance numbers

- Disadvantage

  -- Opens up the possibility of potentially wrong decisions

    -- How do you ensure you get the "relative trends" accurately?

Simulation as Progressive Refinement

- High-level models (Abstract, C)
- …
- Medium-level models (Less abstract)
- …
- Low-level models (RTL with eveything modeled)
- …
- Real design

- As you refine (go down the above list)
  - Abstraction level reduces
  - Accuracy (hopefully) increases (not necessarily, if not careful)
  - Speed and flexibility reduce
  - You can loop back and fix higher-level models

Making The Best of Architecture

- A good architect is comfortable at all levels of refinement
  - Including the extremes

- A good architect knows when to use what type of simulation

Ramulator: A Fast and Extensible DRAM Simulator

[IEEE Comp Arch Letters'15]

Ramulator Motivation

- DRAM and Memory Controller landscape is changing
- Many new and upcoming standards
- Many new controller designs
- A fast and easy-to-extend simulator is very much needed

Segment	*DRAM Standards & Architectures*
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLDRAM3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

Table 1. Landscape of DRAM-based memory

Ramulator

- Provides out-of-the box support for many DRAM standards:
  - DDR3/4, LPDDR3/4, GDDR5, WIO1/2, HBM, plus new proposals (SALP, AL-DRAM, TLDRAM, RowClone, and SARP)
- ~2.5X faster than fastest open-source simulator
- Modular and extensible to different standards

Simulator (clang -O3)	*Cycles (10^6)*		*Runtime (sec.)*		*Req/sec (10^3)*		*Memory (MB)*
	Random	*Stream*	*Random*	*Stream*	*Random*	*Stream*	
Ramulator	652	411	752	249	133	402	2.1
DRAMSim2	645	413	2,030	876	49	114	1.2
USIMM	661	409	1,880	750	53	133	4.5
DrSim	647	406	18,109	12,984	6	8	1.6
NVMain	666	413	6,881	5,023	15	20	4,230.0

Table 3. Comparison of five simulators using two traces

Case Study: Comparison of DRAM Standards

Standard	*Rate (MT/s)*	*Timing (CL-RCD-RP)*	*Data-Bus (Width×Chan.)*	*Rank-per-Chan*	*BW (GB/s)*
DDR3	1,600	11-11-11	64-bit × 1	1	11.9
DDR4	2,400	16-16-16	64-bit × 1	1	17.9
SALP†	1,600	11-11-11	64-bit × 1	1	11.9
LPDDR3	1,600	12-15-15	64-bit × 1	1	11.9
LPDDR4	2,400	22-22-22	32-bit × 2*	1	17.9
GDDR5 [12]	6,000	18-18-18	64-bit × 1	1	44.7
HBM	1,000	7-7-7	128-bit × 8*	1	119.2
WIO	266	7-7-7	128-bit × 4*	1	15.9
WIO2	1,066	9-10-10	128-bit × 8*	1	127.2

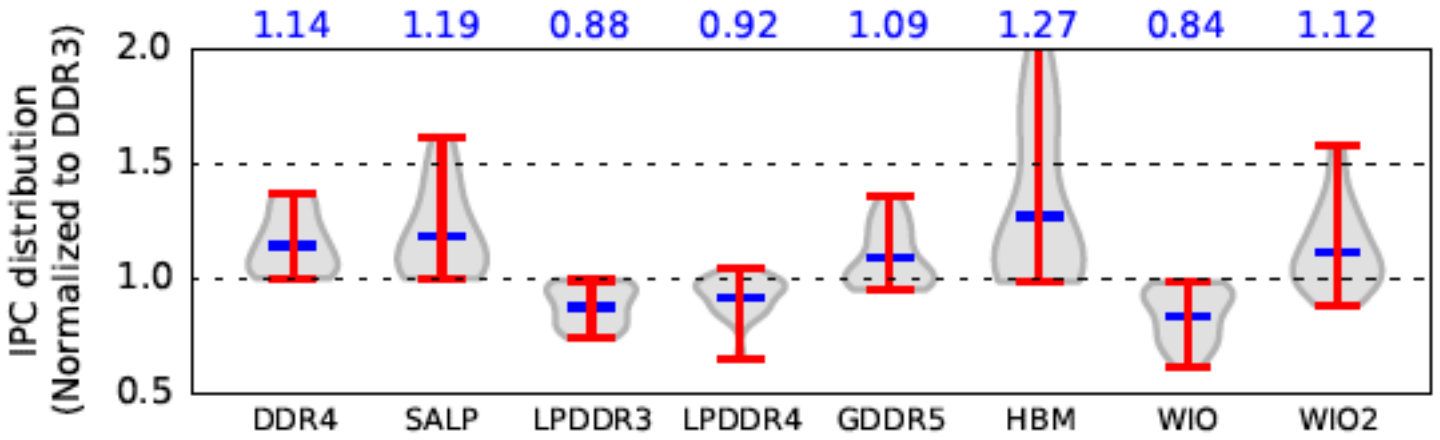


Figure 2. Performance comparison of DRAM standards

Across 22 workloads, simple CPU model

Ramulator Paper and Source Code

- Yoongu Kim, Weikun Yang, and Onur Mutlu,
  "Ramulator: A Fast and Extensible DRAM Simulator"
  IEEE Computer Architecture Letters (***CAL***), March 2015.
  [Source Code]

- Source code is released under the liberal MIT License
  - https://github.com/CMU-SAFARI/ramulator

Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim[1] Weikun Yang[1,2] Onur Mutlu[1]
[1]Carnegie Mellon University [2]Peking University

Extra Credit Assignment

- Review the Ramulator paper
  - Send your reviews to me (omutlu@gmail.com)

- Download and run Ramulator
  - Compare DDR3, DDR4, SALP, HBM for the libquantum benchmark (provided in Ramulator repository)
  - Send your brief report to me

Emerging Memory Technologies

Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
- Conclusions
- Discussion

Major Trends Affecting Main Memory (I)

- Need for main memory capacity and bandwidth increasing
- Main memory energy/power is a key system design concern
- DRAM technology scaling is ending

Trends: Problems with DRAM as Main Memory

- Need for main memory capacity and bandwidth increasing
  - DRAM capacity hard to scale
- Main memory energy/power is a key system design concern
  - DRAM consumes high power due to leakage and refresh
- DRAM technology scaling is ending
  - DRAM capacity, cost, and energy/power hard to scale

Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
- Conclusions
- Discussion

Requirements from an Ideal Memory System

- Traditional
  - Enough capacity
  - Low cost
  - High system performance (high bandwidth, low latency)

- New
  - Technology scalability: lower cost, higher capacity, lower energy
  - Energy (and power) efficiency
  - QoS support and configurability (for consolidation)

Requirements from an Ideal Memory System

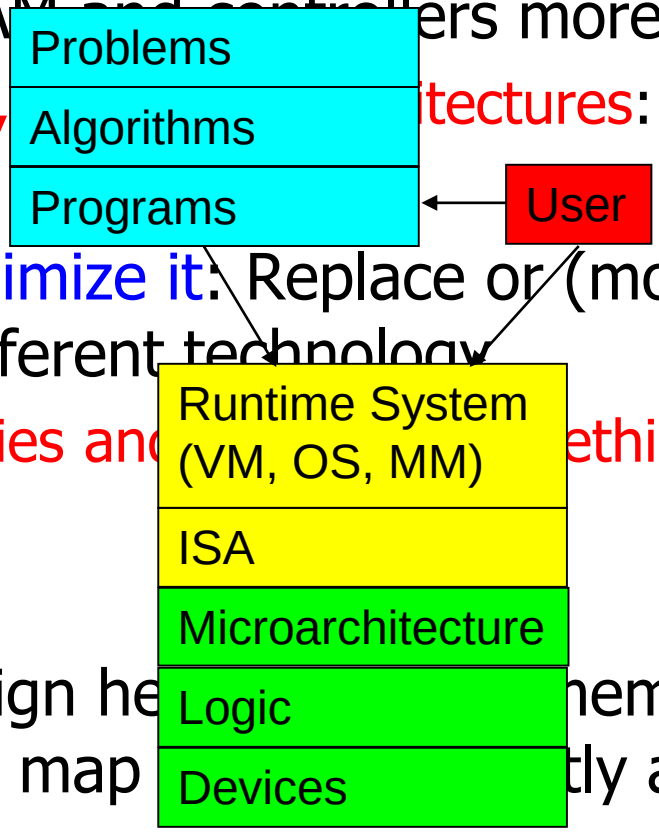
- Traditional
  - Higher capacity
  - Continuous low cost
  - High system performance (higher bandwidth, low latency)

- New
  - Technology scalability: lower cost, higher capacity, lower energy
  - Energy (and power) efficiency
  - QoS support and configurability (for consolidation)

Emerging, resistive memory technologies (NVM) can help

How Do We Solve The Memory Problem?

- Fix it: Make DRA[illegible]ers more intelligent
  - New interfaces, [illegible]itectures: system-DRAM codesign
- Eliminate or minimize it: Replace or (more likely) augment DRAM with a different technology
  - New technologies an[illegible]ethinking of memory & storage
- Embrace it: Design he[illegible]nemories (none of which are perfect) and map [illegible]tly across them
  - New models for data management and maybe usage



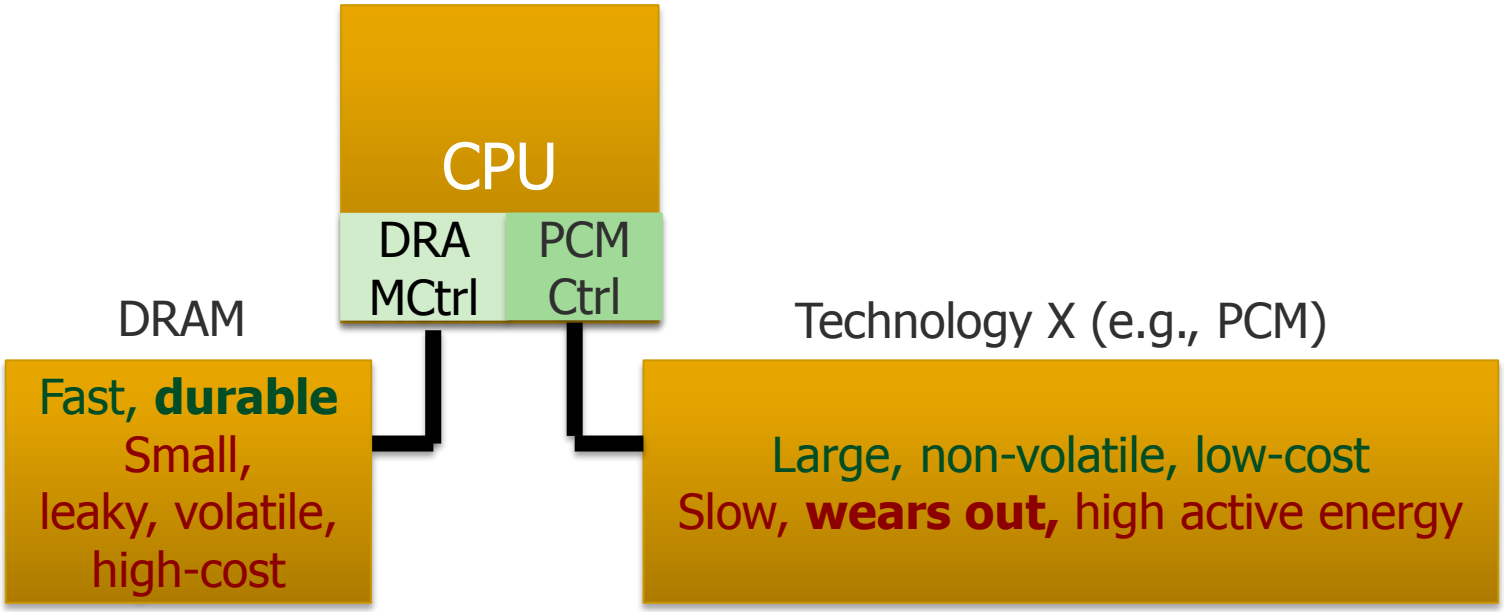
Solutions (to memory scaling) require software/hardware/device cooperation

Solution 2: Emerging Memory Technologies

- Some emerging resistive memory technologies seem more scalable than DRAM (and they are non-volatile)
- Example: Phase Change Memory
  - Expected to scale to 9nm (2022 [ITRS])
  - Expected to be denser than DRAM: can store multiple bits/cell
- But, emerging technologies have shortcomings as well
  - Can they be enabled to replace/augment/surpass DRAM?

- Lee+, "Architecting Phase Change Memory as a Scalable DRAM Alternative," ISCA'09, CACM'10, Micro'10.
- Meza+, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters 2012.
- Yoon, Meza+, "Row Buffer Locality Aware Caching Policies for Hybrid Memories," ICCD 2012.
- Kultursay+, "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," ISPASS 2013.
- Meza+, "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory," WEED 2013.
- Lu+, "Loose Ordering Consistency for Persistent Memory," ICCD 2014.
- Zhao+, "FIRM: Fair and High-Performance Memory Control for Persistent Memory Systems," MICRO 2014.
- Yoon, Meza+, "Efficient Data Mapping and Buffering Techniques for Multi-Level Cell Phase-Change Memories," ACM TACO 2014.
- Ren+, "Dual-Scheme Checkpointing: "A Software-Transparent Mechanism for Supporting Crash Consistency in Persistent Memory Systems," MICRO 2015.

Solution 3: Hybrid Memory Systems



Hardware/software manage data allocation and movement
to achieve the best of multiple technologies

Meza+, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters, 2012.

Yoon, Meza et al., "Row Buffer Locality Aware Caching Policies for Hybrid Memories," ICCD 2012 Best Paper Award.

Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
- Conclusions
- Discussion

The Promise of Emerging Technologies

- Likely need to replace/augment DRAM with a technology that is
  - Technology scalable
  - And at least similarly efficient, high performance, and fault-tolerant
    - or can be architected to be so

- Some emerging resistive memory technologies appear promising
  - Phase Change Memory (PCM)?
  - Spin Torque Transfer Magnetic Memory (STT-MRAM)?
  - Memristors? RRAM? ReRAM?
  - And, maybe there are other ones
  - Can they be enabled to replace/augment/surpass DRAM?

Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
  - Background
  - PCM (or Technology X) as DRAM Replacement
  - Hybrid Memory Systems
- Conclusions
- Discussion

Charge vs. Resistive Memories

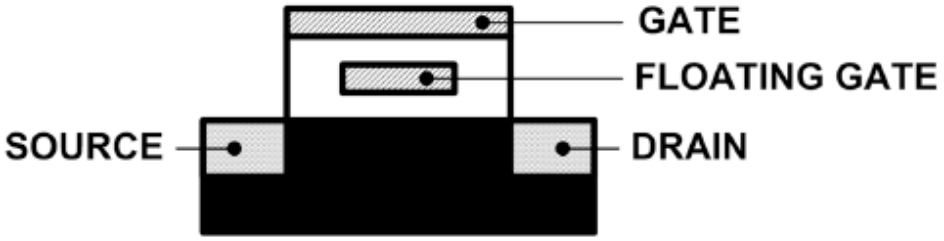
- Charge Memory (e.g., DRAM, Flash)
  - Write data by capturing charge Q
  - Read data by detecting voltage V

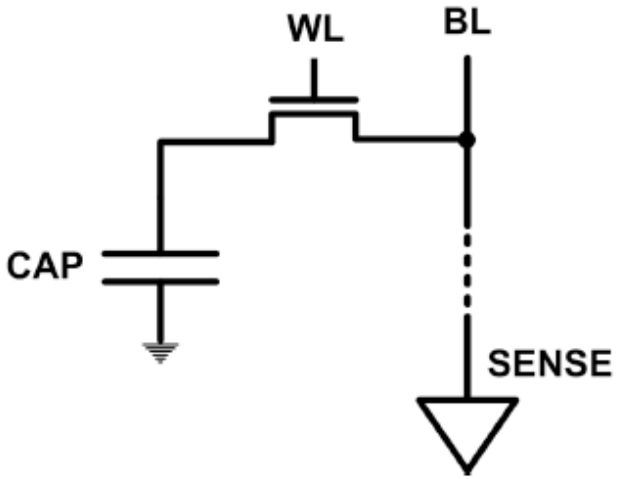
- Resistive Memory (e.g., PCM, STT-MRAM, memristors)
  - Write data by pulsing current dQ/dt
  - Read data by detecting resistance R

Limits of Charge Memory

- Difficult charge placement and control
  - Flash: floating gate charge
  - DRAM: capacitor charge, transistor leakage

- Reliable sensing becomes difficult as charge storage unit size reduces



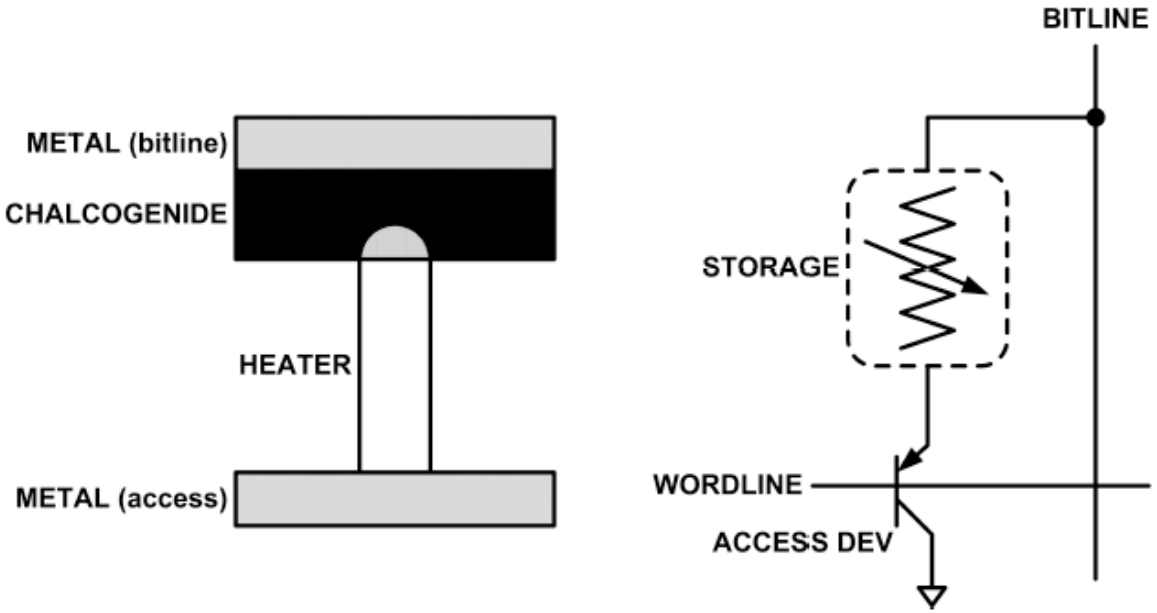


Promising Resistive Memory Technologies

- PCM
  - Inject current to change material phase
  - Resistance determined by phase
- STT-MRAM
  - Inject current to change magnet polarity
  - Resistance determined by polarity
- Memristors/RRAM/ReRAM
  - Inject current to change atomic structure
  - Resistance determined by atom distance

What is Phase Change Memory?

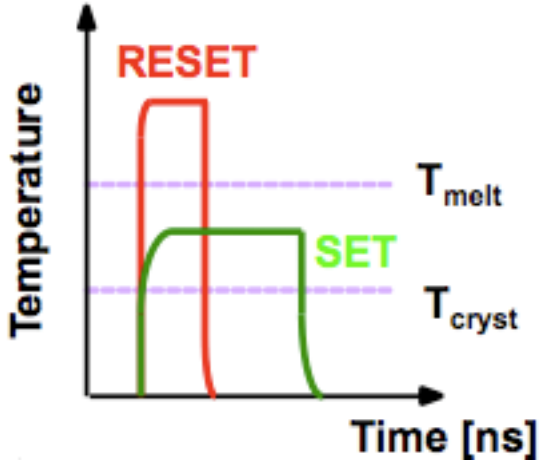
- Phase change material (chalcogenide glass) exists in two states:
  - Amorphous: Low optical reflexivity and high electrical resistivity
  - Crystalline: High optical reflexivity and low electrical resistivity

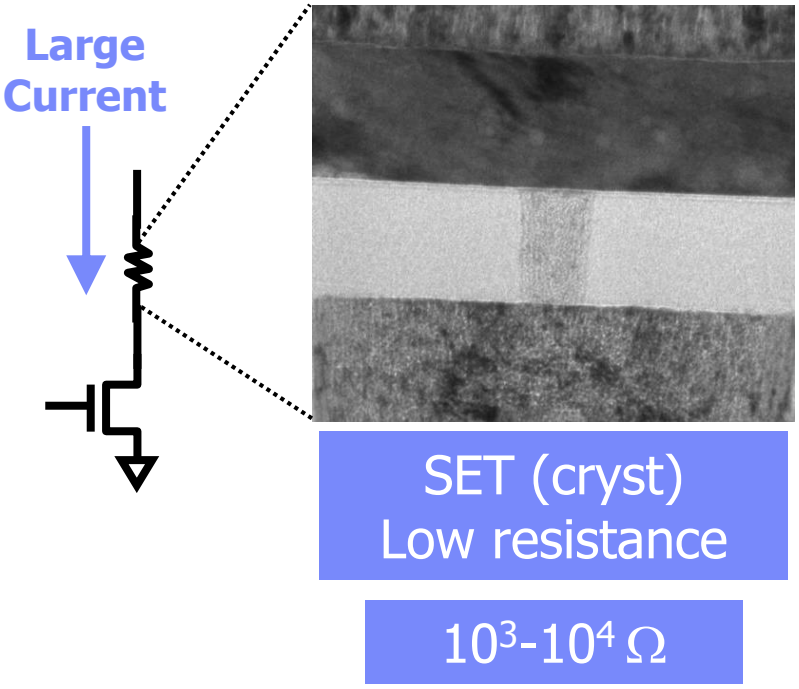


PCM is resistive memory: High resistance (0), Low resistance (1)
PCM cell can be switched between states reliably and quickly

How Does PCM Work?

- Write: change phase via current injection
  - SET: sustained current to heat cell above T*cryst*
  - RESET: cell heated above T*melt* and quenched
- Read: detect phase via material resistance
  - amorphous/crystalline





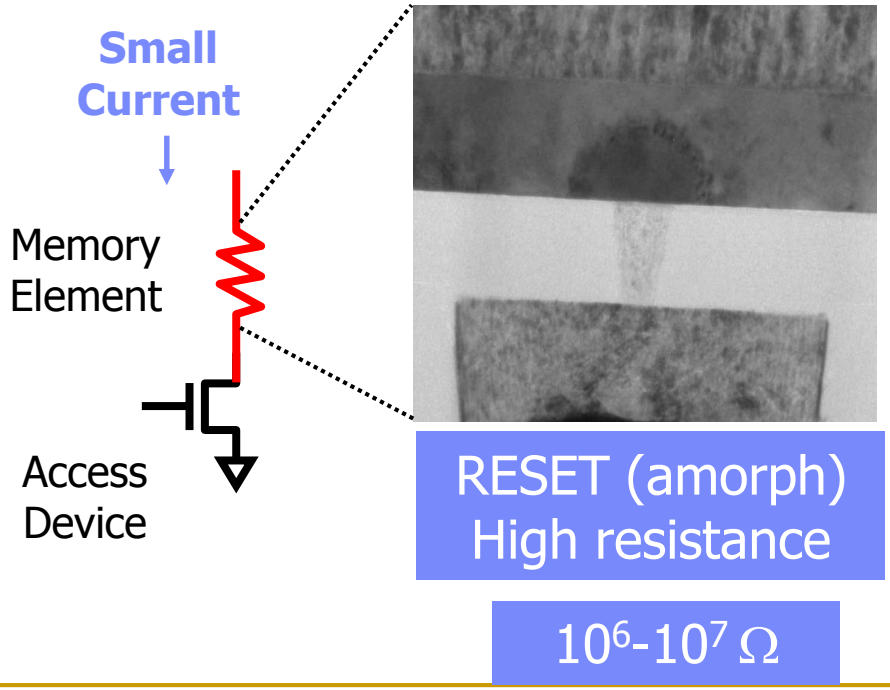
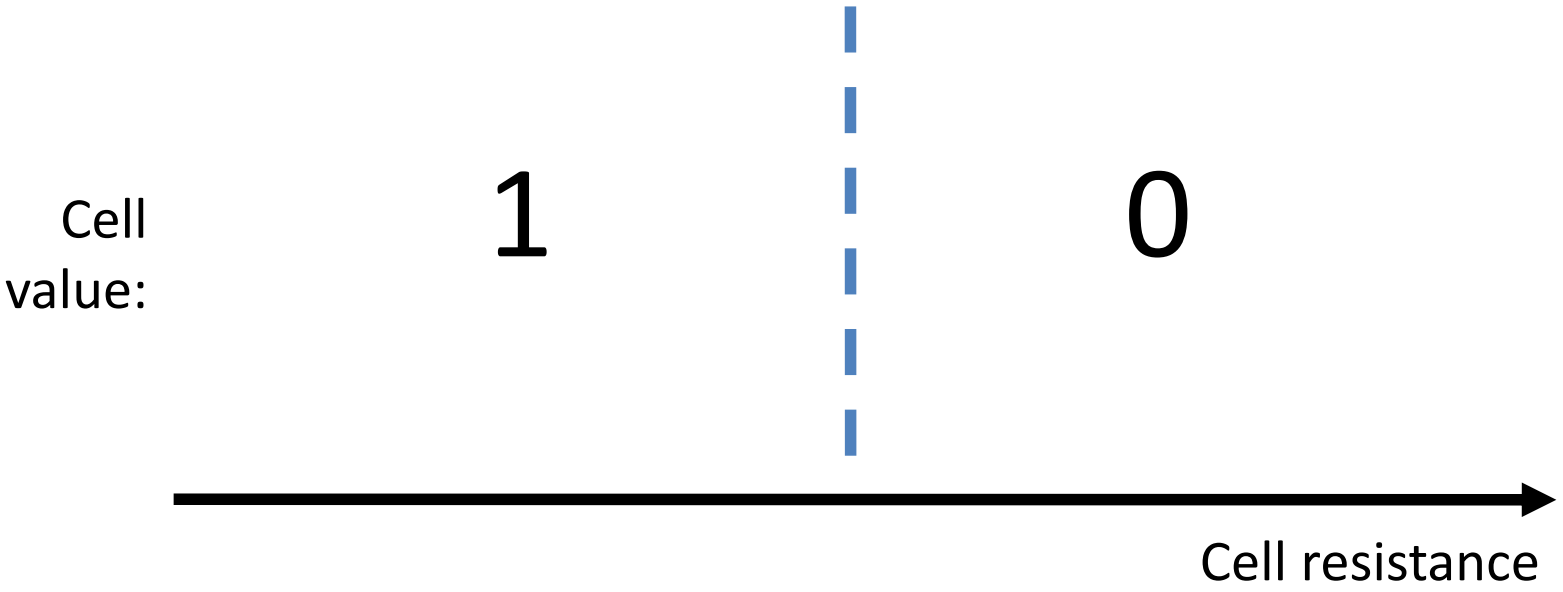


Photo Courtesy: Bipin Rajendran, IBM Slide Courtesy: Moinuddin Qureshi, IBM

Opportunity: PCM Advantages

- Scales better than DRAM, Flash
  - Requires current pulses, which scale linearly with feature size
  - Expected to scale to 9nm (2022 [ITRS])
  - Prototyped at 20nm (Raoux+, IBM JRD 2008)
- Can be denser than DRAM
  - Can store multiple bits per cell due to large resistance range
  - Prototypes with 2 bits/cell in ISSCC' 08, 4 bits/cell by 2012
- Non-volatile
  - Can retain data for >10 years at 85C
- No refresh needed, low idle power

PCM Resistance → Value

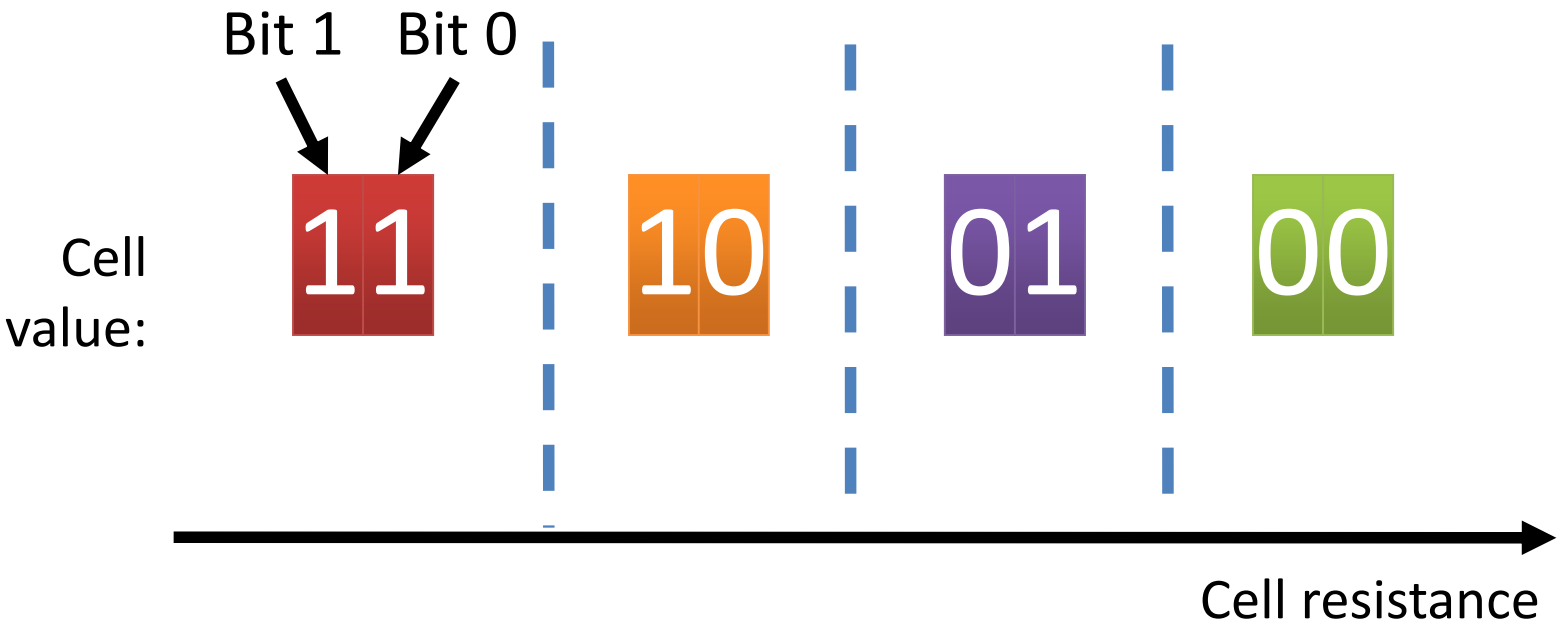


Multi-Level Cell PCM

- Multi-level cell: more than 1 bit per cell
  - Further increases density by 2 to 4x [Lee+,ISCA'09]

- But MLC-PCM also has drawbacks
  - Higher latency and energy than single-level cell PCM

MLC-PCM Resistance → Value

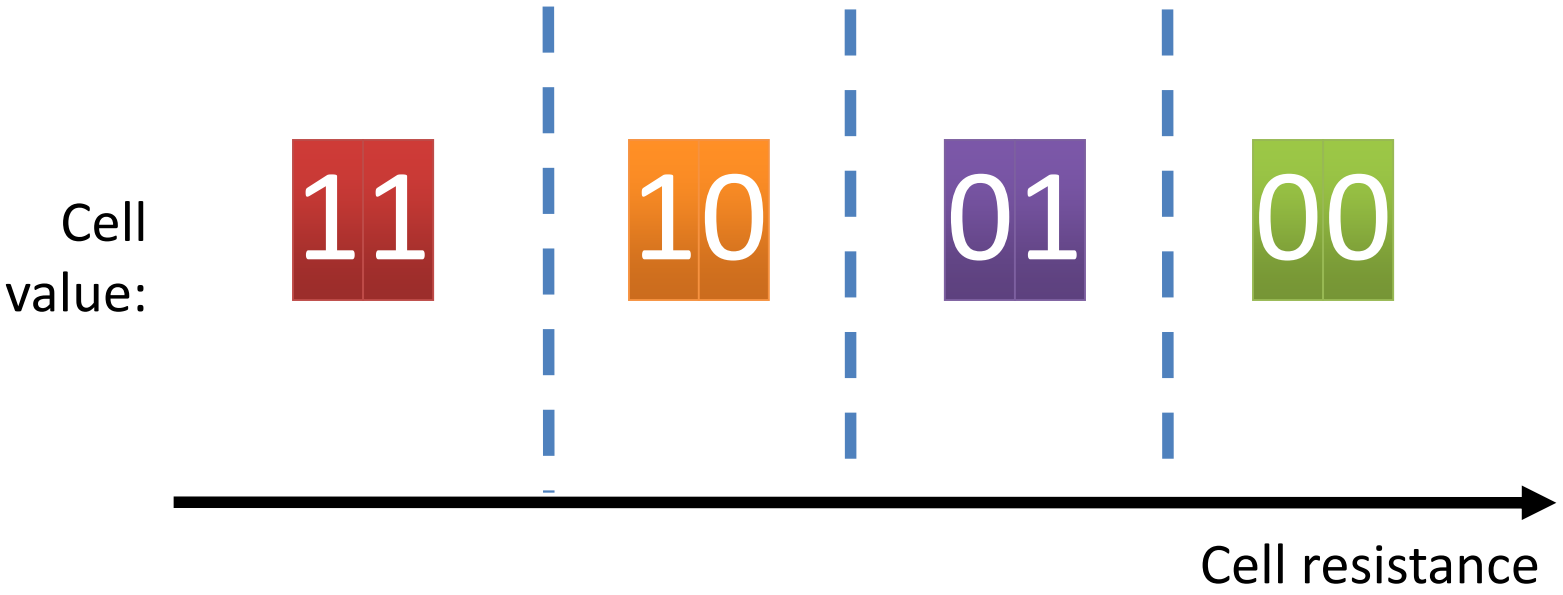


MLC-PCM Resistance → Value

Less margin between values

→ need more precise sensing/modification of cell contents

→ higher latency/energy (~2x for reads and 4x for writes)



Phase Change Memory Properties

- Surveyed prototypes from 2003-2008 (ITRS, IEDM, VLSI, ISSCC)
- Derived PCM parameters for F=90nm

- Lee, Ipek, Mutlu, Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," ISCA 2009.

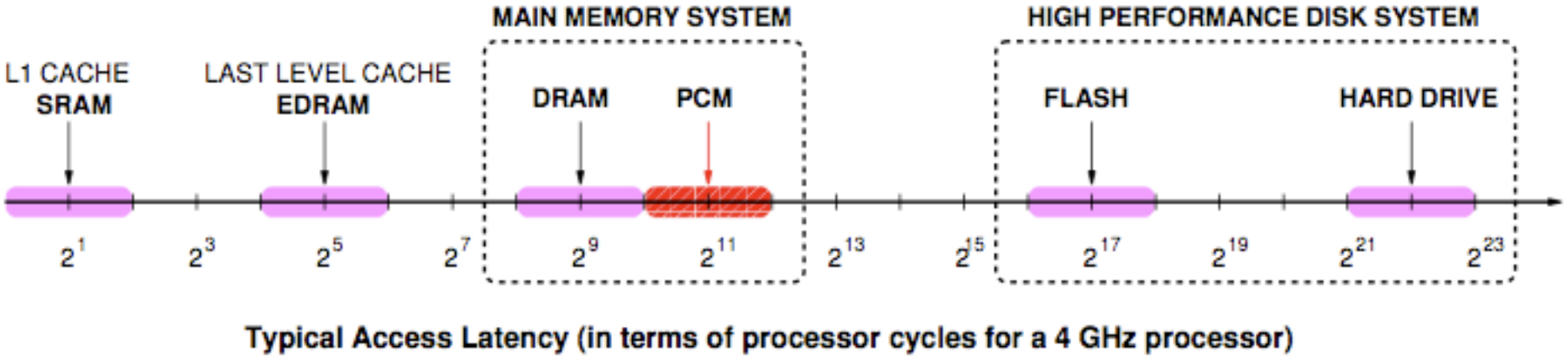
Table 1. Technology survey.

Parameter*	Published prototype									
	Horri[6]	Ahn[12]	Bedeschi[13]	Oh[14]	Pellizer[15]	Chen[5]	Kang[16]	Bedeschi[9]	Lee[10]	Lee[2]
Year	2003	2004	2004	2005	2006	2006	2006	2008	2008	**
Process, F (nm)	**	120	180	120	90	**	100	90	90	90
Array size (Mbytes)	**	64	8	64	**	**	256	256	512	**
Material	GST, N-d	GST, N-d	GST	GST	GST	GS, N-d	GST	GST	GST	GST, N-d
Cell size (μm^2)	**	0.290	0.290	**	0.097	60 nm^2	0.166	0.097	0.047	0.065 to 0.097
Cell size, F^2	**	20.1	9.0	**	12.0	**	16.6	12.0	5.8	9.0 to 12.0
Access device	**	**	BJT	FET	BJT	**	FET	BJT	Diode	BJT
Read time (ns)	**	70	48	68	**	**	62	**	55	48
Read current (μA)	**	**	40	**	**	**	**	**	**	40
Read voltage (V)	**	3.0	1.0	1.8	1.6	**	1.8	**	1.8	1.0
Read power (μW)	**	**	40	**	**	**	**	**	**	40
Read energy (pJ)	**	**	2.0	**	**	**	**	**	**	2.0
Set time (ns)	100	150	150	180	**	80	300	**	400	150
Set current (μA)	200	**	300	200	**	55	**	**	**	150
Set voltage (V)	**	**	2.0	**	**	1.25	**	**	**	1.2
Set power (μW)	**	**	300	**	**	34.4	**	**	**	90
Set energy (pJ)	**	**	45	**	**	2.8	**	**	**	13.5
Reset time (ns)	50	10	40	10	**	60	50	**	50	40
Reset current (μA)	600	600	600	600	400	90	600	300	600	300
Reset voltage (V)	**	**	2.7	**	1.8	1.6	**	1.6	**	1.6
Reset power (μW)	**	**	1620	**	**	80.4	**	**	**	480
Reset energy (pJ)	**	**	64.8	**	**	4.8	**	**	**	19.2
Write endurance (MLC)	10^7	10^9	10^6	**	10^8	10^4	**	10^5	10^5	10^8

* BJT: bipolar junction transistor; FET: field-effect transistor; GST: $Ge_2Sb_2Te_5$; MLC: multilevel cells; N-d: nitrogen doped.
** This information is not available in the publication cited.

Phase Change Memory Properties: Latency

- Latency comparable to, but slower than DRAM



- Read Latency
  - 50ns: 4x DRAM, 10^{-3}x NAND Flash
- Write Latency
  - 150ns: 12x DRAM
- Write Bandwidth
  - 5-10 MB/s: 0.1x DRAM, 1x NAND Flash

Phase Change Memory Properties

- Dynamic Energy
  - 40 uA Rd, 150 uA Wr
  - 2-43x DRAM, 1x NAND Flash
- Endurance
  - Writes induce phase change at 650C
  - Contacts degrade from thermal expansion/contraction
  - 10^8 writes per cell
  - 10^{-8}x DRAM, 10^3x NAND Flash
- Cell Size
  - 9-12F^2 using BJT, single-level cells
  - 1.5x DRAM, 2-3x NAND (will scale with feature size, MLC)

Phase Change Memory: Pros and Cons

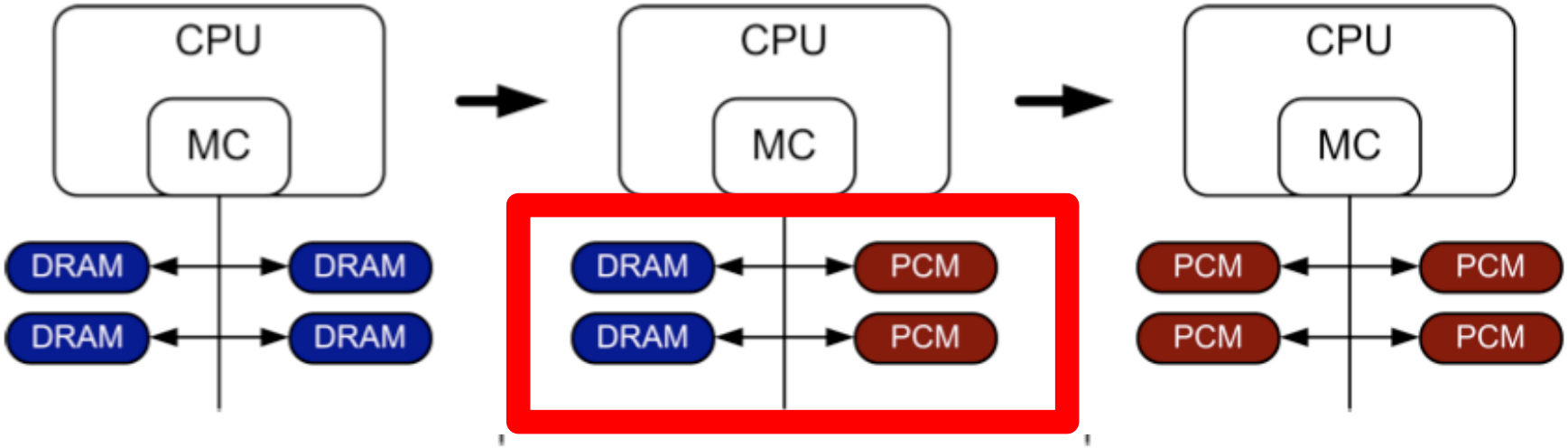
- Pros over DRAM
  - Better technology scaling
  - Non volatility
  - Low idle power (no refresh)
- Cons
  - Higher latencies: ~4-15x DRAM (especially write)
  - Higher active energy: ~2-50x DRAM (especially write)
  - Lower endurance (a cell dies after ~10^8 writes)
- Challenges in enabling PCM as DRAM replacement/helper:
  - Mitigate PCM shortcomings
  - Find the right way to place PCM in the system
  - Ensure secure and fault-tolerant PCM operation

PCM-based Main Memory: Challenges

- Where to place PCM in the memory hierarchy?
  - Hybrid OS controlled PCM-DRAM
  - Hybrid OS controlled PCM and hardware-controlled DRAM
  - Pure PCM main memory
- How to mitigate shortcomings of PCM?
- How to minimize amount of DRAM in the system?
- How to take advantage of (byte-addressable and fast) non-volatile main memory?
- Can we design specific-NVM-technology-agnostic techniques?

PCM-based Main Memory (I)

- How should PCM-based (main) memory be organized?



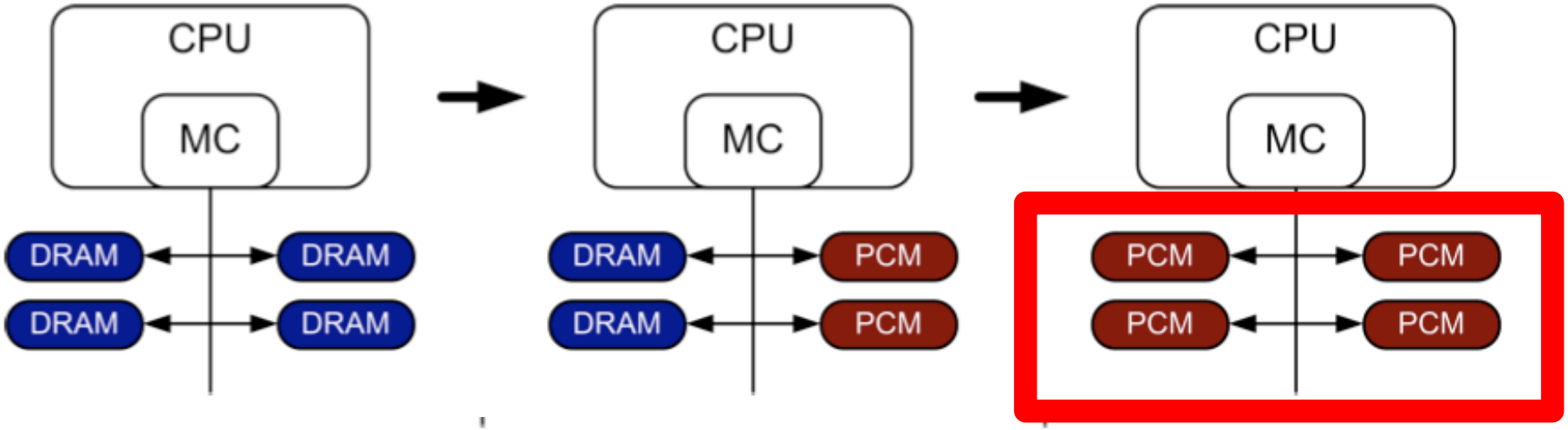
- Hybrid PCM+DRAM [Qureshi+ ISCA'09, Dhiman+ DAC'09, Meza+ IEEE CAL'12]:
  - How to partition/migrate data between PCM and DRAM

Hybrid Memory Systems: Challenges

- Partitioning
  - Should DRAM be a cache or main memory, or configurable?
  - What fraction? How many controllers?
- Data allocation/movement (energy, performance, lifetime)
  - Who manages allocation/movement?
  - What are good control algorithms?
  - How do we prevent degradation of service due to wearout?
- Design of cache hierarchy, memory controllers, OS
  - Mitigate PCM shortcomings, exploit PCM advantages
- Design of PCM/DRAM chips and modules
  - Rethink the design of PCM/DRAM with new requirements

PCM-based Main Memory (II)

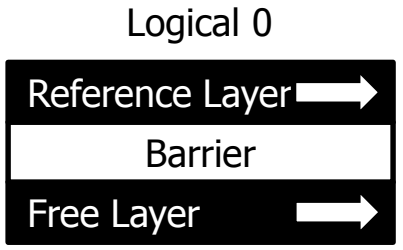
- How should PCM-based (main) memory be organized?



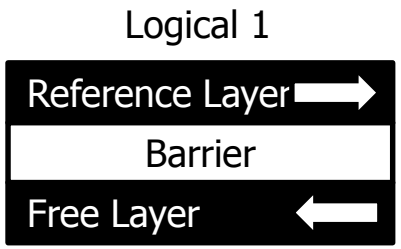
- Pure PCM main memory [Lee et al., ISCA'09, Top Picks'10]:
  - How to redesign entire hierarchy (and cores) to overcome PCM shortcomings

STT-MRAM as Main Memory

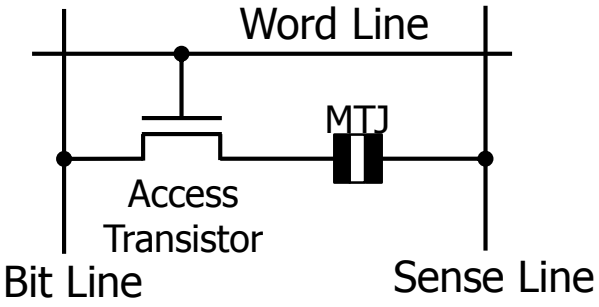
- Magnetic Tunnel Junction (MTJ) device
  - Reference layer: Fixed magnetic orientation
  - Free layer: Parallel or anti-parallel



- Magnetic orientation of the free layer determines logical state of device
  - High vs. low resistance



- Write: Push large current through MTJ to change orientation of free layer
- Read: Sense current flow



- Kultursay et al., "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," ISPASS 2013.

Aside: STT MRAM: Pros and Cons

- Pros over DRAM
  - Better technology scaling
  - Non volatility
  - Low idle power (no refresh)
- Cons
  - Higher write latency
  - Higher write energy
  - Reliability?
- Another level of freedom
  - Can trade off non-volatility for lower write latency/energy (by reducing the size of the MTJ)

Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
  - Background
  - PCM (or Technology X) as DRAM Replacement
  - Hybrid Memory Systems
- Conclusions
- Discussion

An Initial Study: Replace DRAM with PCM

- Lee, Ipek, Mutlu, Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," ISCA 2009.
  - Surveyed prototypes from 2003-2008 (e.g. IEDM, VLSI, ISSCC)
  - Derived "average" PCM parameters for F=90nm

Density

- 9 - 12F^2 using BJT
- 1.5× DRAM

Latency

- 50ns Rd, 150ns Wr
- 4×, 12× DRAM

Endurance

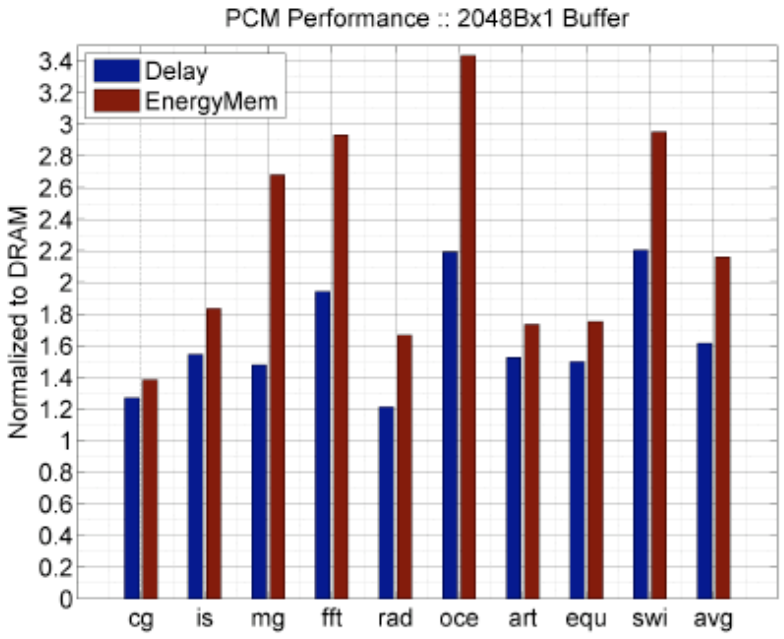
- 1E+08 writes
- 1E-08× DRAM

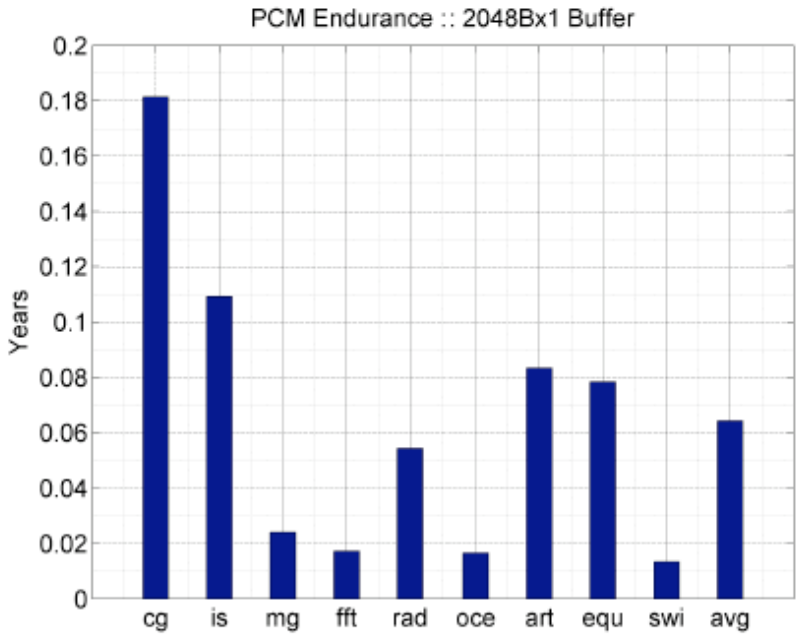
Energy

- 40μA Rd, 150μA Wr
- 2×, 43× DRAM

Results: Naïve Replacement of DRAM with PCM

- Replace DRAM with PCM in a 4-core, 4MB L2 system
- PCM organized the same as DRAM: row buffers, banks, peripherals
- 1.6x delay, 2.2x energy, 500-hour average lifetime



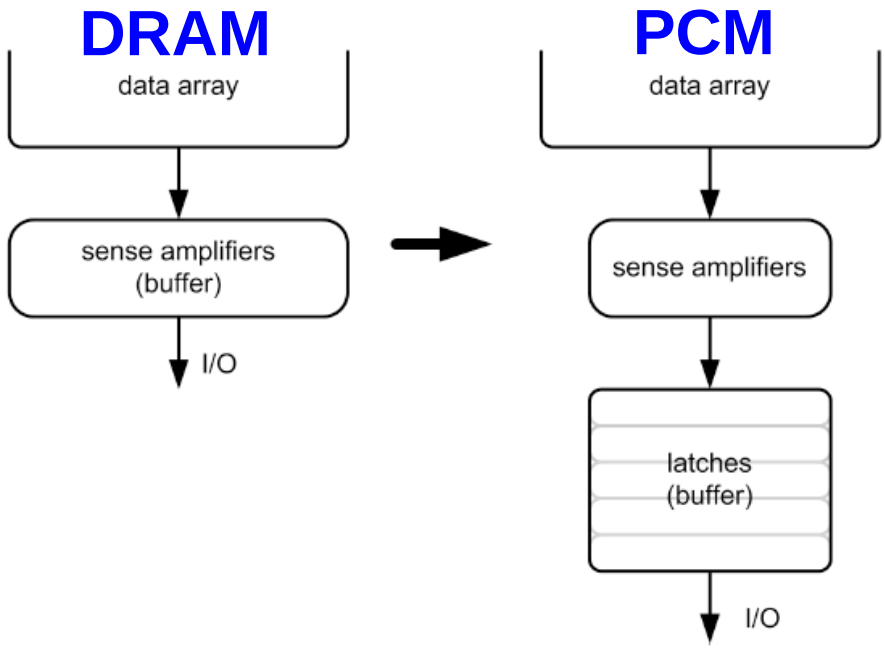


- Lee, Ipek, Mutlu, Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," ISCA 2009.

Architecting PCM to Mitigate Shortcomings

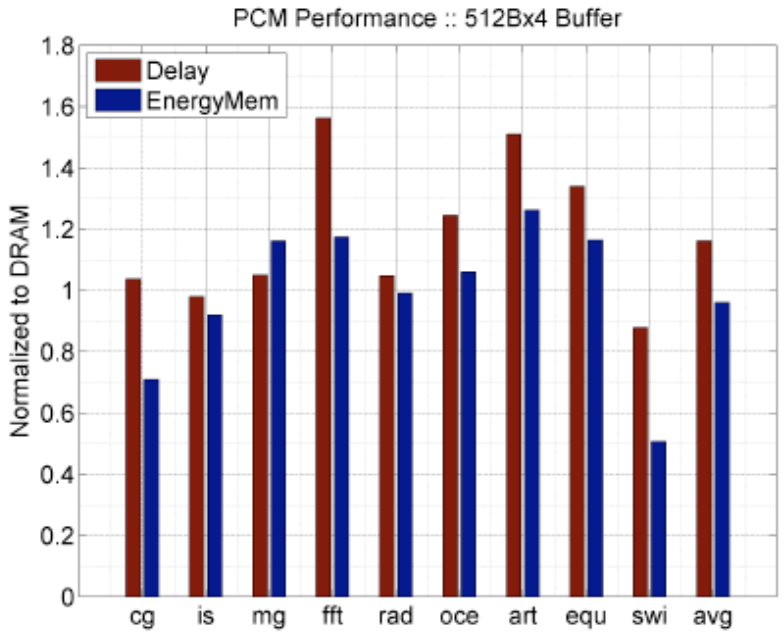
- Idea 1: Use multiple narrow row buffers in each PCM chip
  → Reduces array reads/writes → better endurance, latency, energy

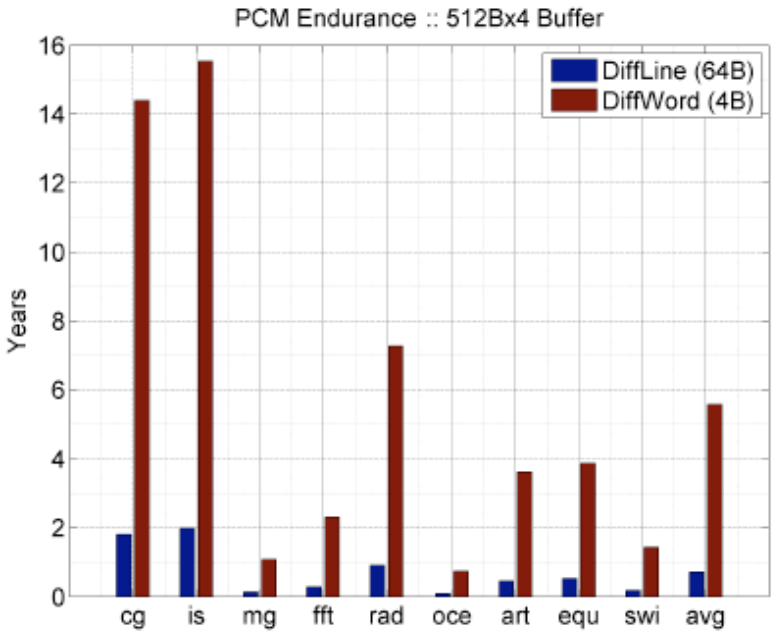
- Idea 2: Write into array at cache block or word granularity
  → Reduces unnecessary wear



Results: Architected PCM as Main Memory

- 1.2x delay, 1.0x energy, 5.6-year average lifetime
- Scaling improves energy, endurance, density





- Caveat 1: Worst-case lifetime is much shorter (no guarantees)
- Caveat 2: Intensive applications see large performance and energy hits
- Caveat 3: Optimistic PCM parameters?

More on PCM As Main Memory

- Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger, **"Architecting Phase Change Memory as a Scalable DRAM Alternative"**
  *Proceedings of the 36th International Symposium on Computer Architecture (**ISCA**)*, pages 2-13, Austin, TX, June 2009. Slides (pdf)

Architecting Phase Change Memory as a Scalable DRAM Alternative

Benjamin C. Lee† Engin Ipek† Onur Mutlu‡ Doug Burger†

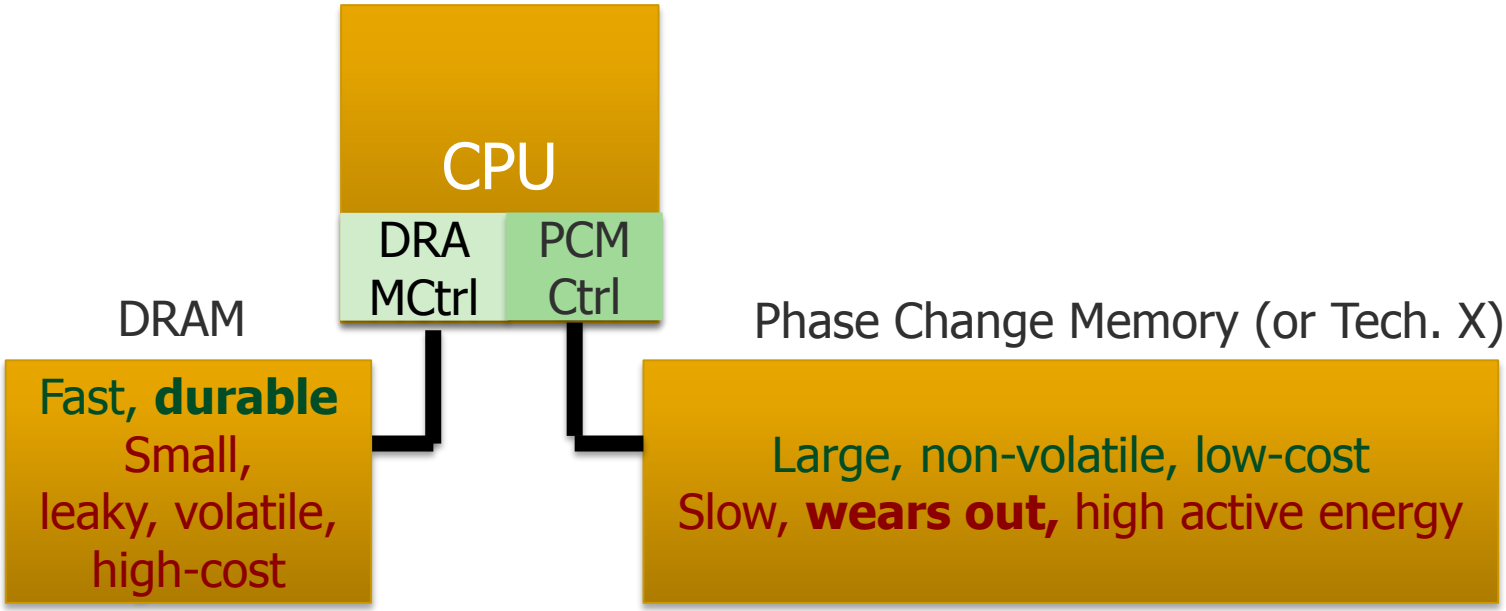
†Computer Architecture Group
Microsoft Research
Redmond, WA
{blee, ipek, dburger}@microsoft.com

‡Computer Architecture Laboratory
Carnegie Mellon University
Pittsburgh, PA
onur@cmu.edu

Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
  - Background
  - PCM (or Technology X) as DRAM Replacement
  - Hybrid Memory Systems
- Conclusions
- Discussion

Hybrid Memory Systems



Hardware/software manage data allocation and movement
to achieve the best of multiple technologies

Meza, Chang, Yoon, Mutlu, Ranganathan, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters, 2012.

One Option: DRAM as a Cache for PCM

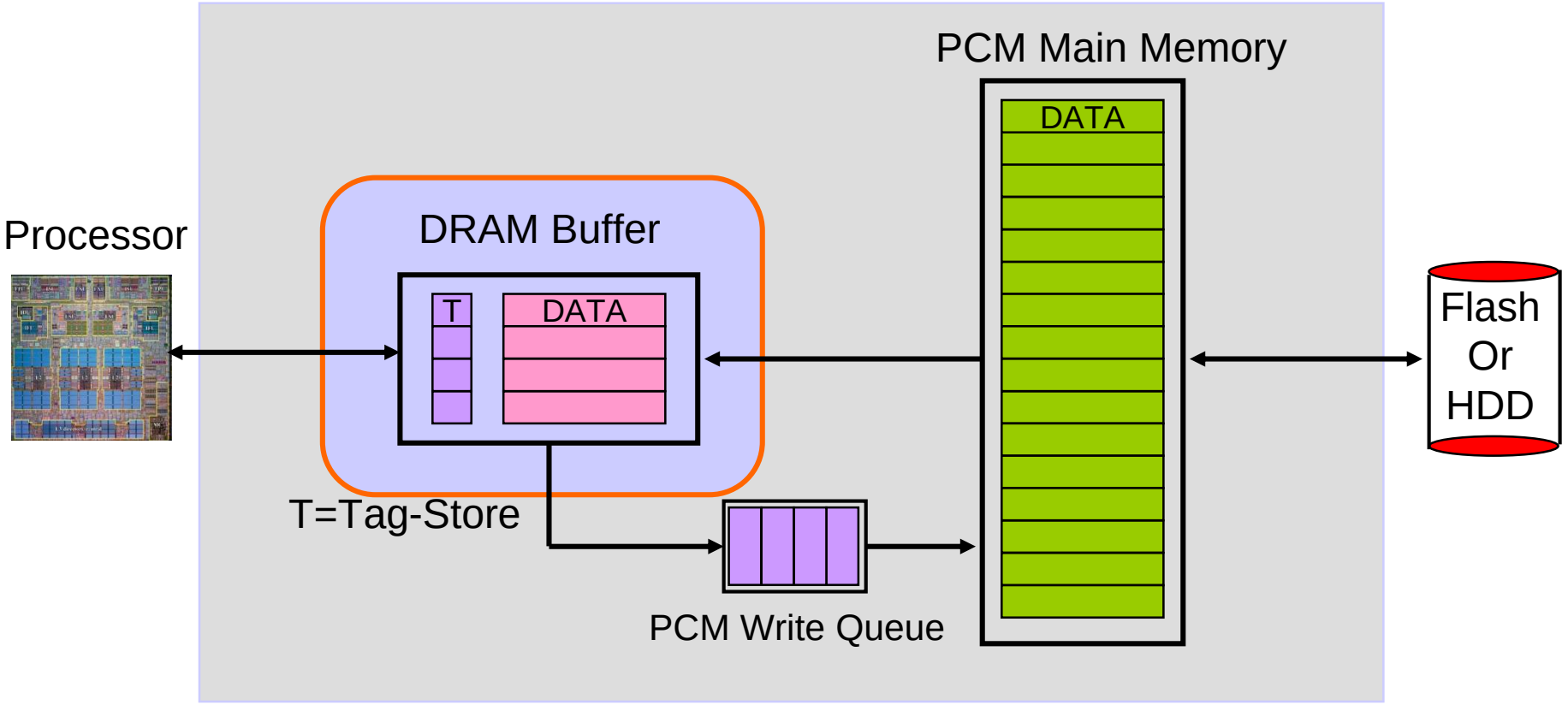
- PCM is main memory; DRAM caches memory rows/blocks
  - Benefits: Reduced latency on DRAM cache hit; write filtering
- Memory controller hardware manages the DRAM cache
  - Benefit: Eliminates system software overhead

- Three issues:
  - What data should be placed in DRAM versus kept in PCM?
  - What is the granularity of data movement?
  - How to design a low-cost hardware-managed DRAM cache?

- Two idea directions:
  - Locality-aware data placement **[Yoon+ , ICCD 2012]**
  - Cheap tag stores and dynamic granularity **[Meza+, IEEE CAL 2012]**

DRAM as a Cache for PCM

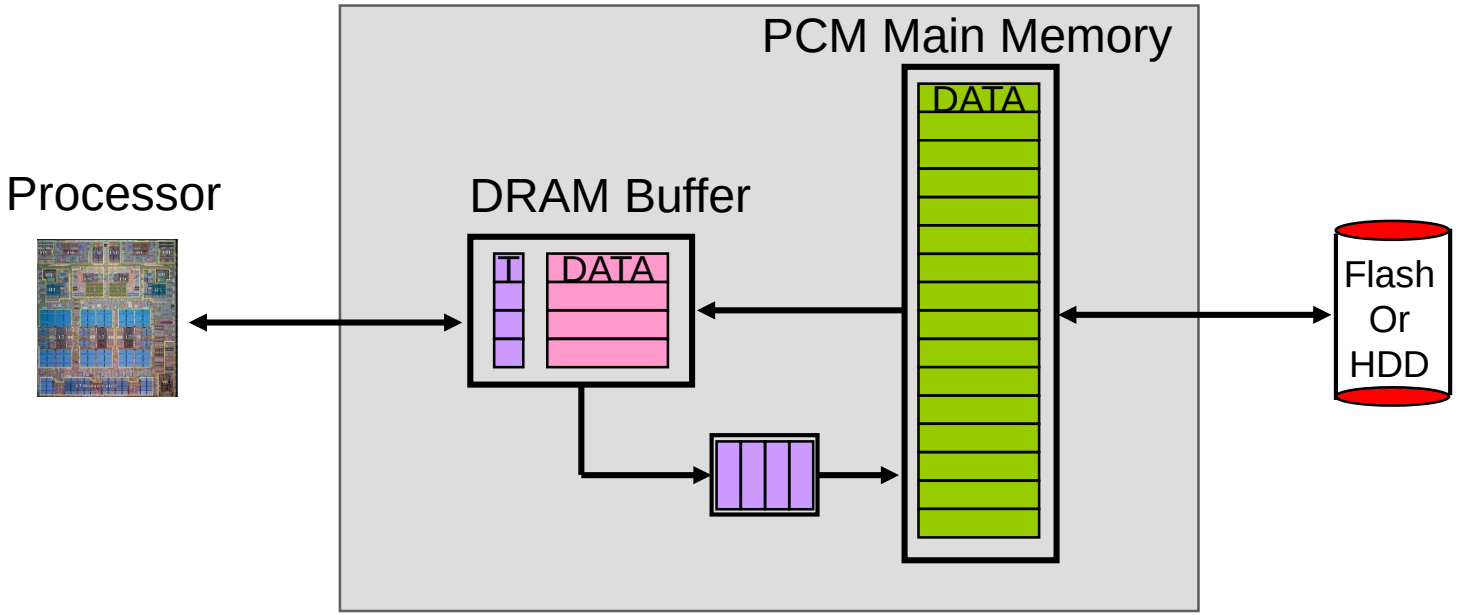
- Goal: Achieve the best of both DRAM and PCM/NVM
  - Minimize amount of DRAM w/o sacrificing performance, endurance
  - DRAM as cache to tolerate PCM latency and write bandwidth
  - PCM as main memory to provide large capacity at good cost and power



Qureshi+, "Scalable high performance main memory system using phase-change memory technology," ISCA 2009.

Write Filtering Techniques

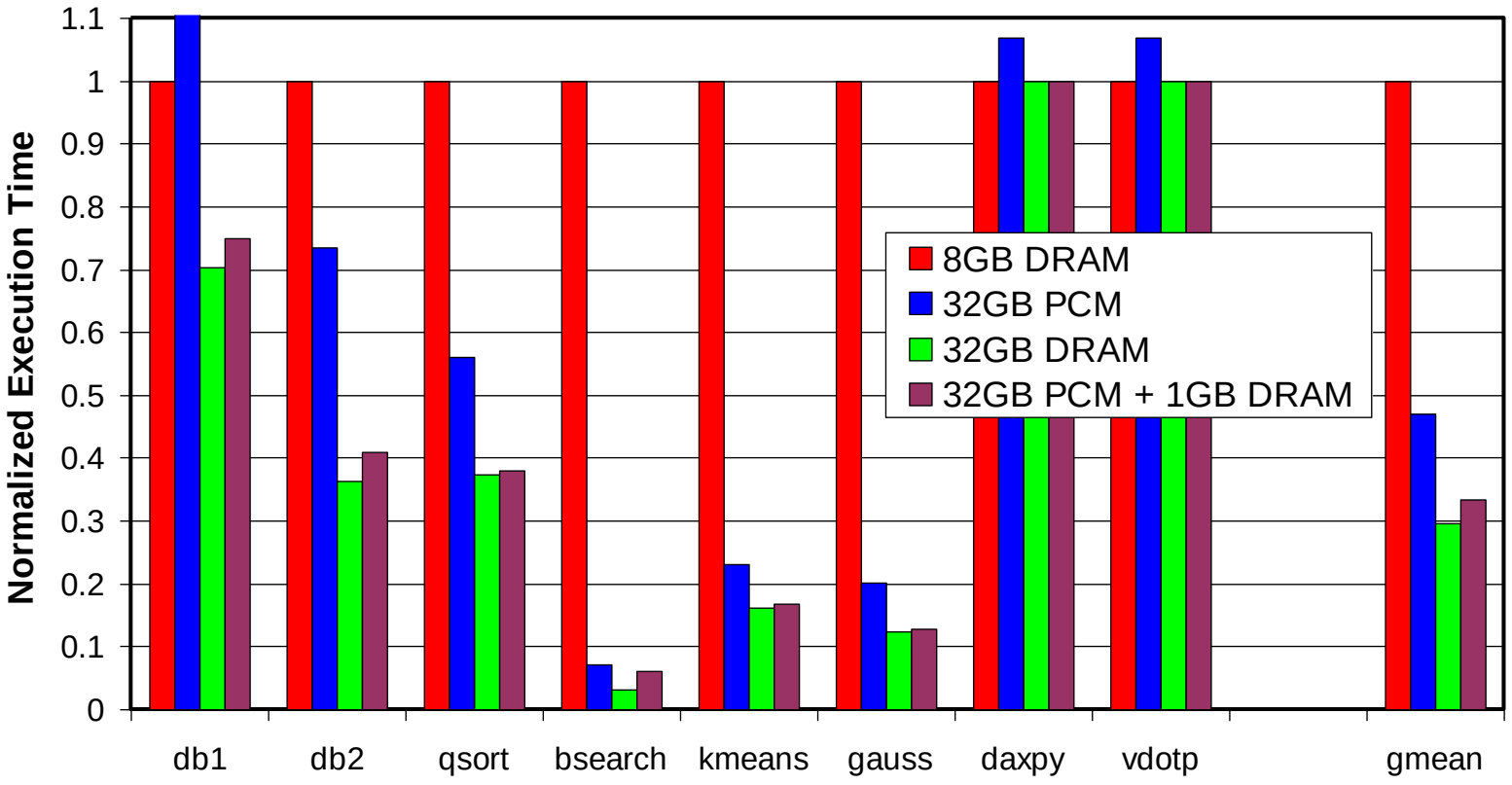
- Lazy Write: Pages from disk installed only in DRAM, not PCM
- Partial Writes: Only dirty lines from DRAM page written back
- Page Bypass: Discard pages with poor reuse on DRAM eviction



- Qureshi et al., "Scalable high performance main memory system using phase-change memory technology," ISCA 2009.

Results: DRAM as PCM Cache (I)

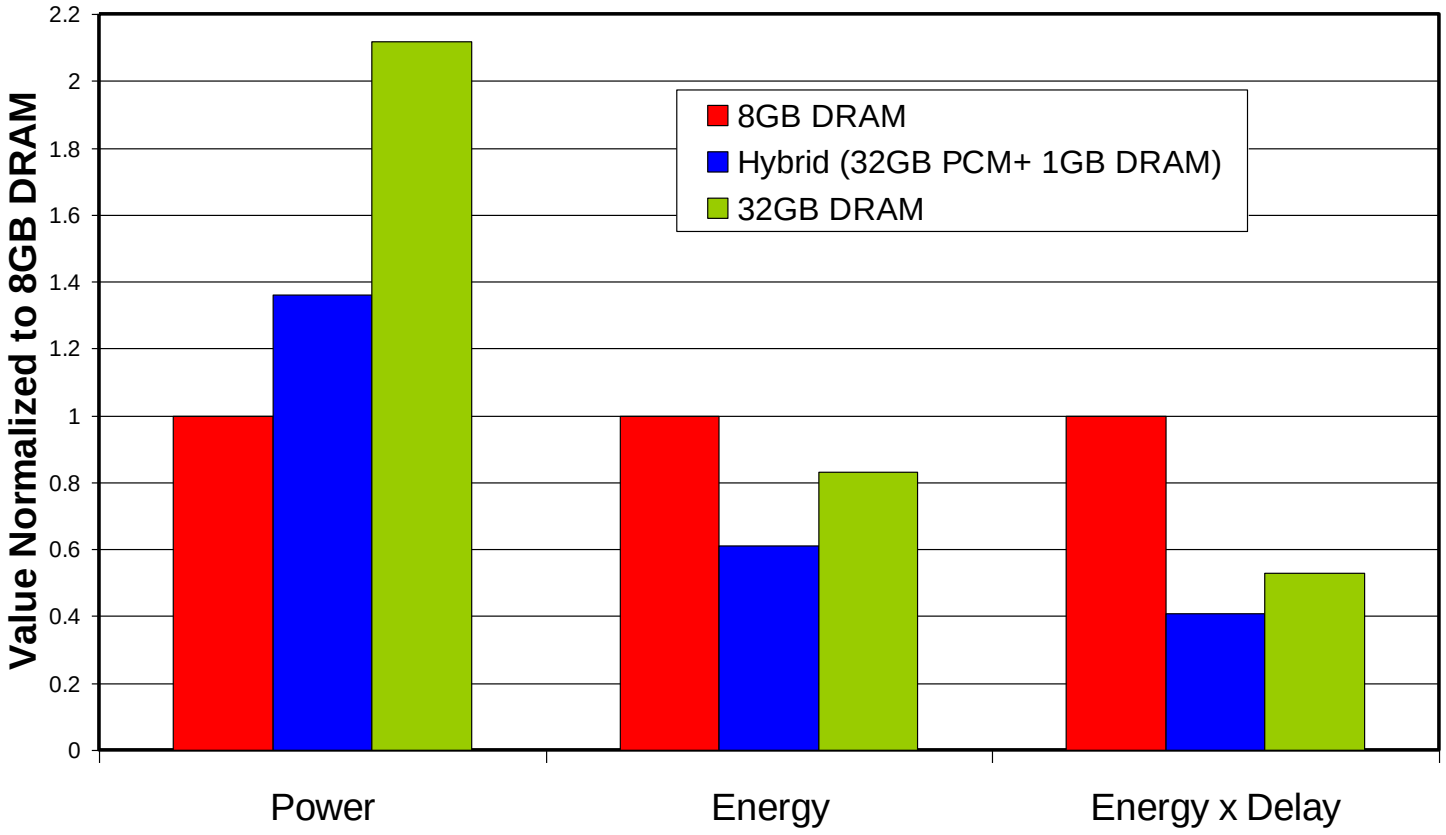
- Simulation of 16-core system, 8GB DRAM main-memory at 320 cycles, HDD (2 ms) with Flash (32 us) with Flash hit-rate of 99%
- Assumption: PCM 4x denser, 4x slower than DRAM
- DRAM block size = PCM page size (4kB)



Qureshi+, "Scalable high performance main memory system using phase-change memory technology," ISCA 2009.

Results: DRAM as PCM Cache (II)

- PCM-DRAM Hybrid performs similarly to similar-size DRAM
- Significant energy savings with PCM-DRAM Hybrid
- Average lifetime: 9.7 years (no guarantees)



Qureshi+, "Scalable high performance main memory system using phase-change memory technology," ISCA 2009.

More on DRAM-PCM Hybrid Memory

- **Scalable High-Performance Main Memory System Using Phase-Change Memory Technology.**
  Moinuddin K. Qureshi, Viji Srinivasan, and Jude A. Rivers
  Appears in the International Symposium on Computer Architecture (ISCA) 2009.

Scalable High Performance Main Memory System Using Phase-Change Memory Technology

Moinuddin K. Qureshi Vijayalakshmi Srinivasan Jude A. Rivers

IBM Research
T. J. Watson Research Center, Yorktown Heights NY 10598

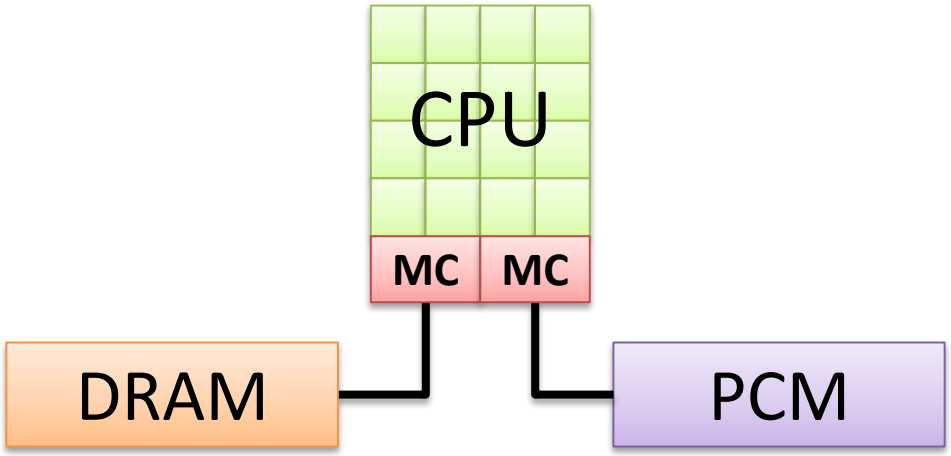
{mkquresh, viji, jarivers}@us.ibm.com

Agenda

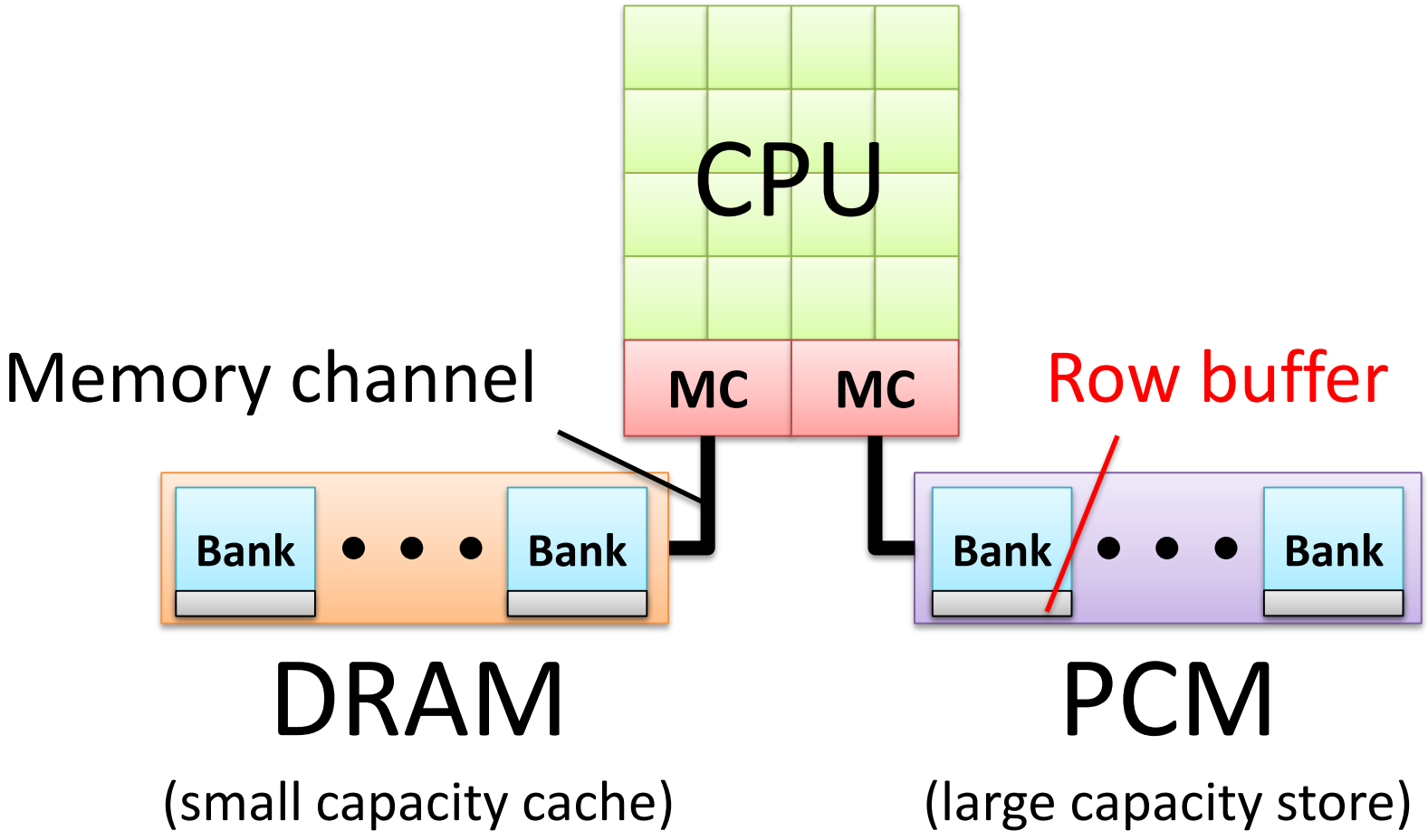
- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
  - Background
  - PCM (or Technology X) as DRAM Replacement
  - Hybrid Memory Systems
    - Row-Locality Aware Data Placement
    - Efficient DRAM (or Technology X) Caches
- Conclusions
- Discussion

Hybrid Memory

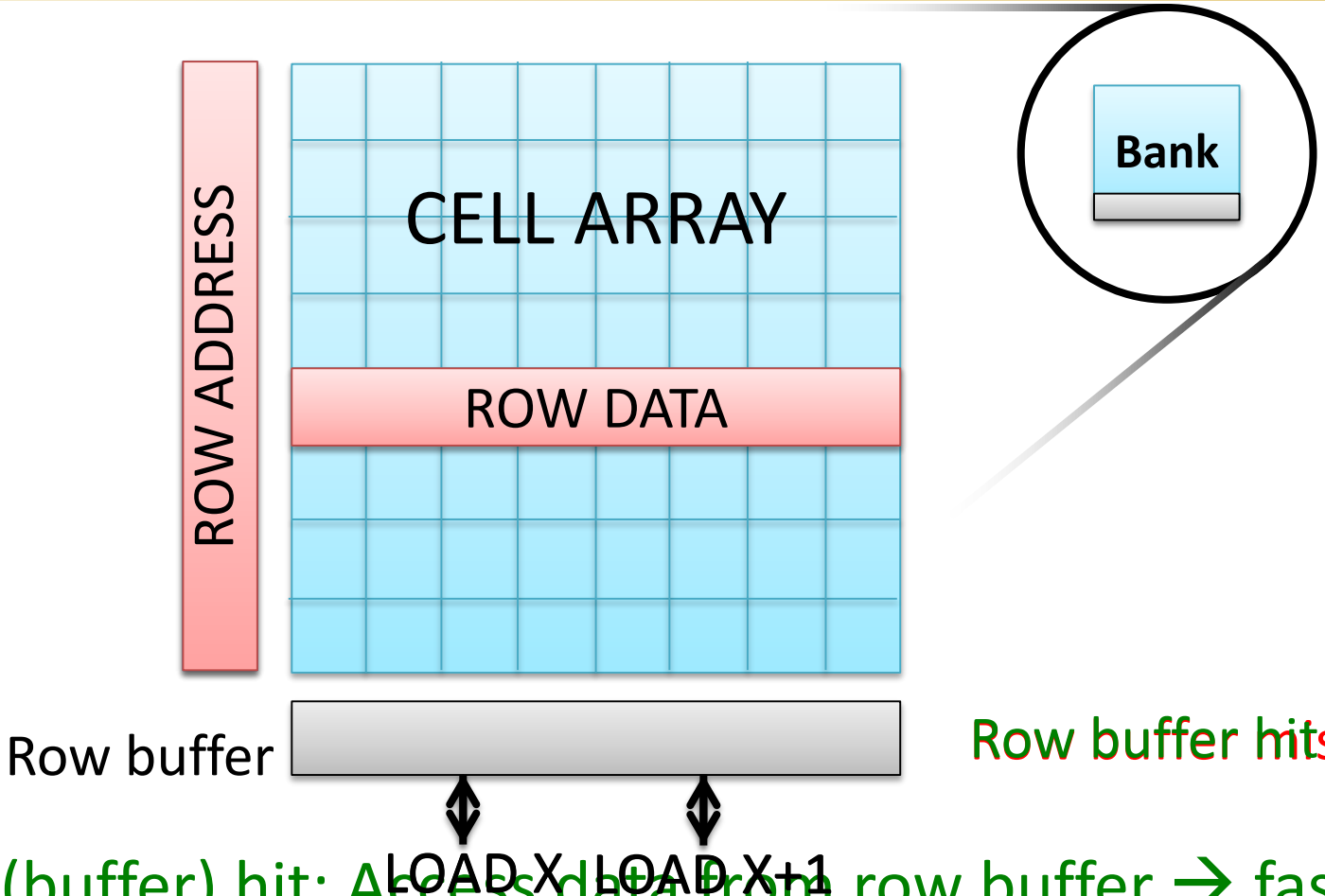
- Key question: How to place data between the heterogeneous memory devices?



Hybrid Memory: A Closer Look



Row Buffers and Latency



Row (buffer) hit: Access data from row buffer → fast

Row (buffer) miss: Access data from cell array → slow

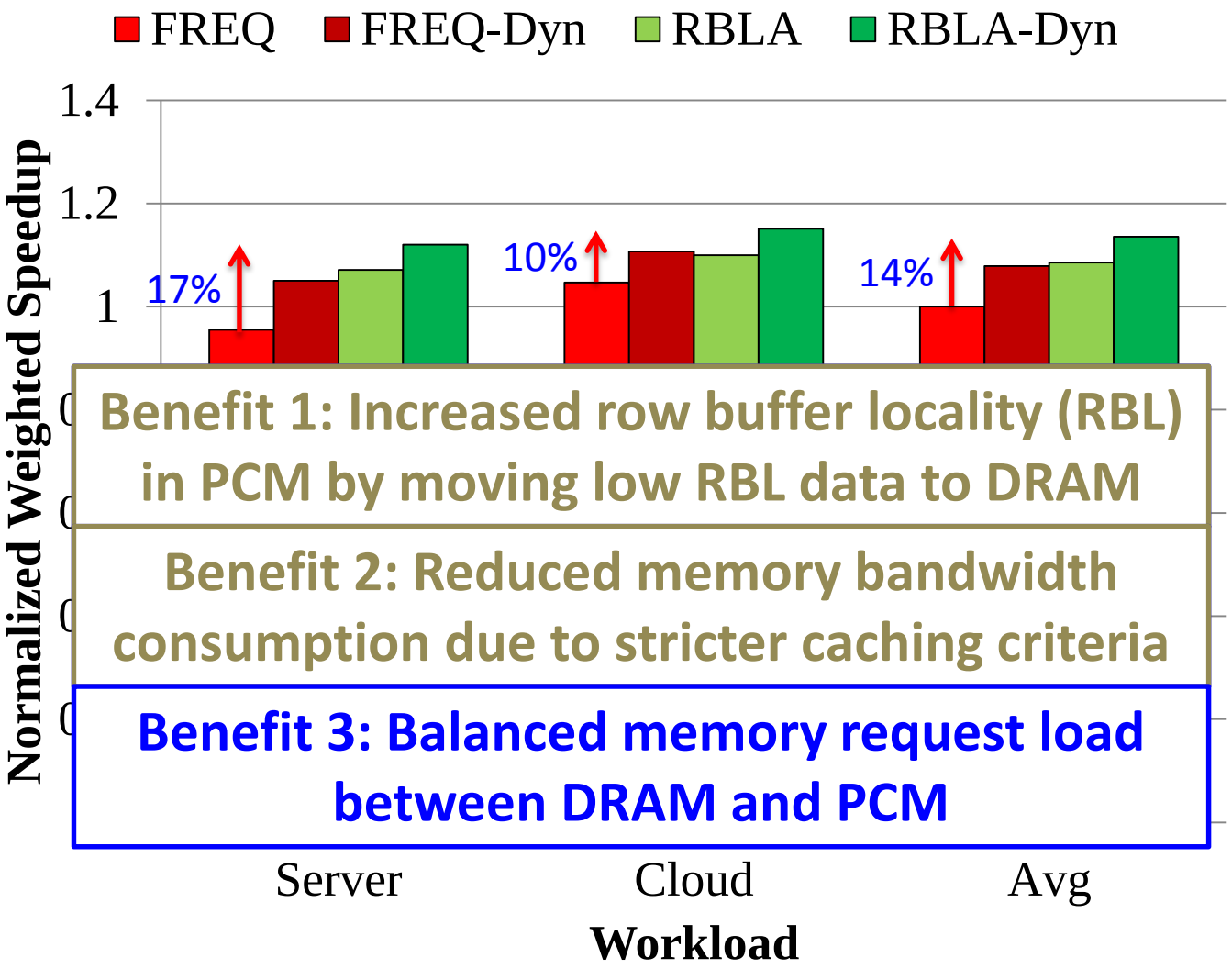
Key Observation

- Row buffers exist in both DRAM and PCM
  - Row hit latency **similar** in DRAM & PCM [Lee+ ISCA'09]
  - Row miss latency **small** in DRAM, **large** in PCM

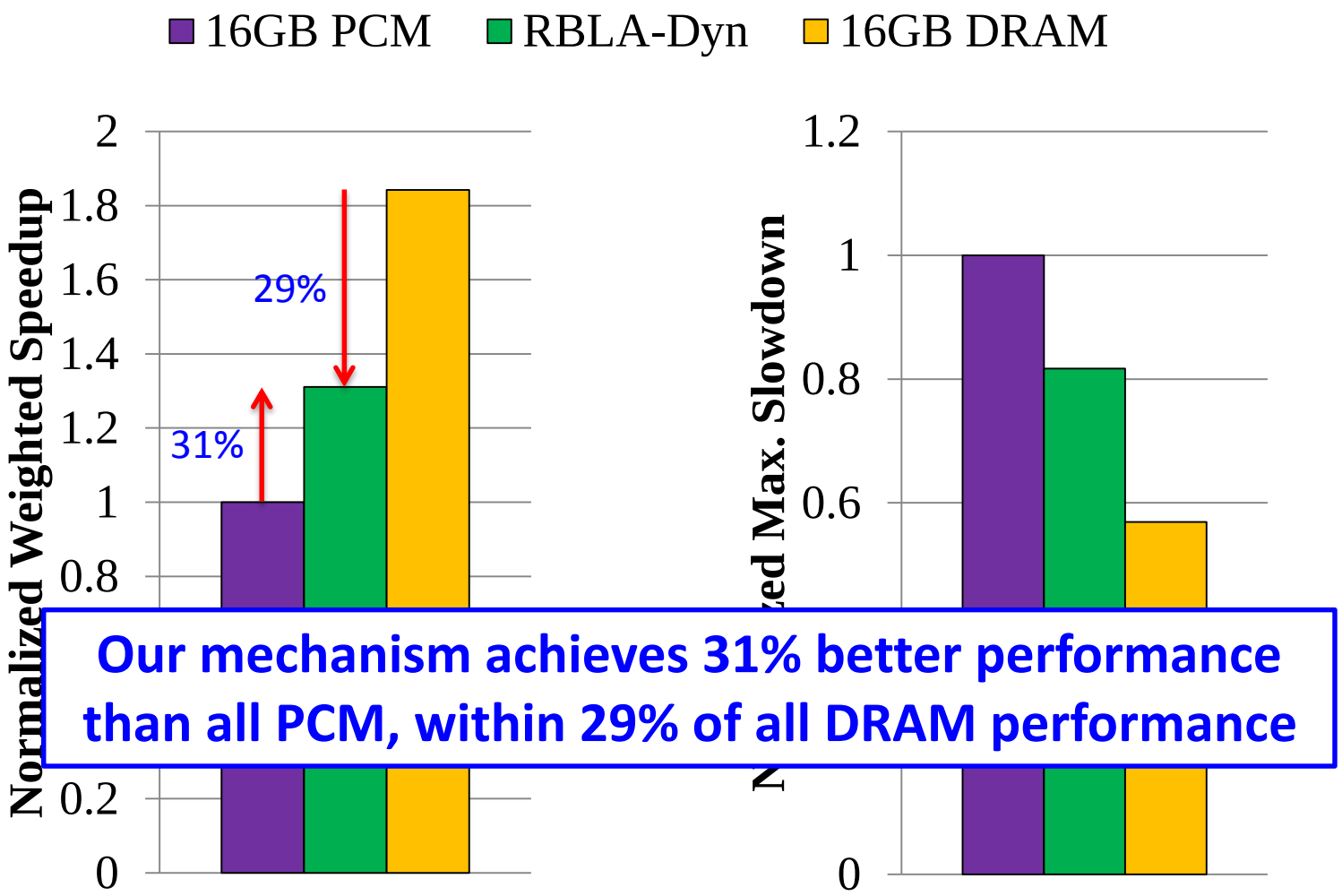
- Place data in DRAM which
  - is likely to miss in the row buffer (low row buffer locality)→ miss penalty is smaller in DRAM

    AND
  - is reused many times → cache only the data worth the movement cost and DRAM space

System Performance



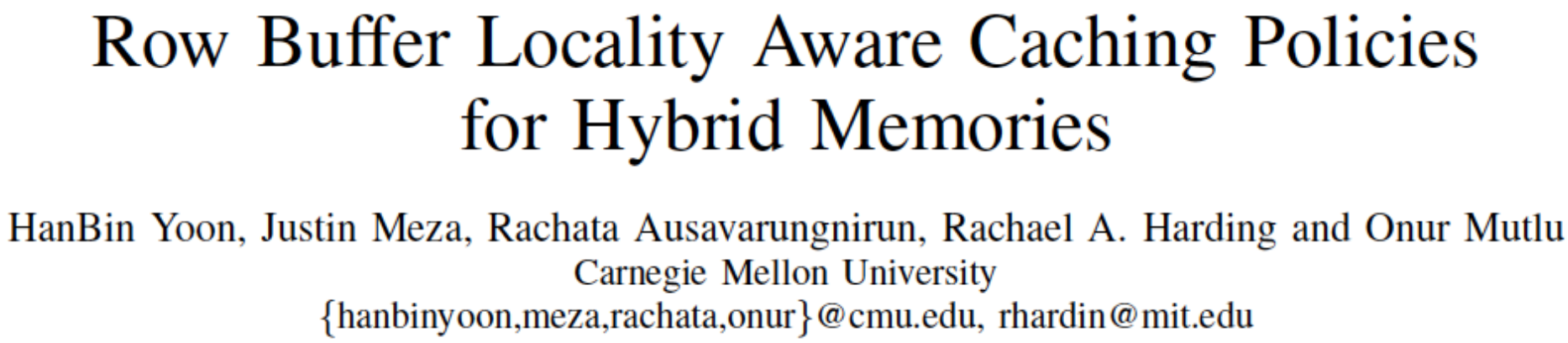
Compared to All-PCM/DRAM



Our mechanism achieves 31% better performance than all PCM, within 29% of all DRAM performance

For More on Hybrid Memory Data Placement

- HanBin Yoon, Justin Meza, Rachata Ausavarungnirun, Rachael Harding, and Onur Mutlu,
  "Row Buffer Locality Aware Caching Policies for Hybrid Memories"
  *Proceedings of the 30th IEEE International Conference on Computer Design (**ICCD**)*, Montreal, Quebec, Canada, September 2012. Slides (pptx) (pdf)



Row Buffer Locality Aware Caching Policies for Hybrid Memories

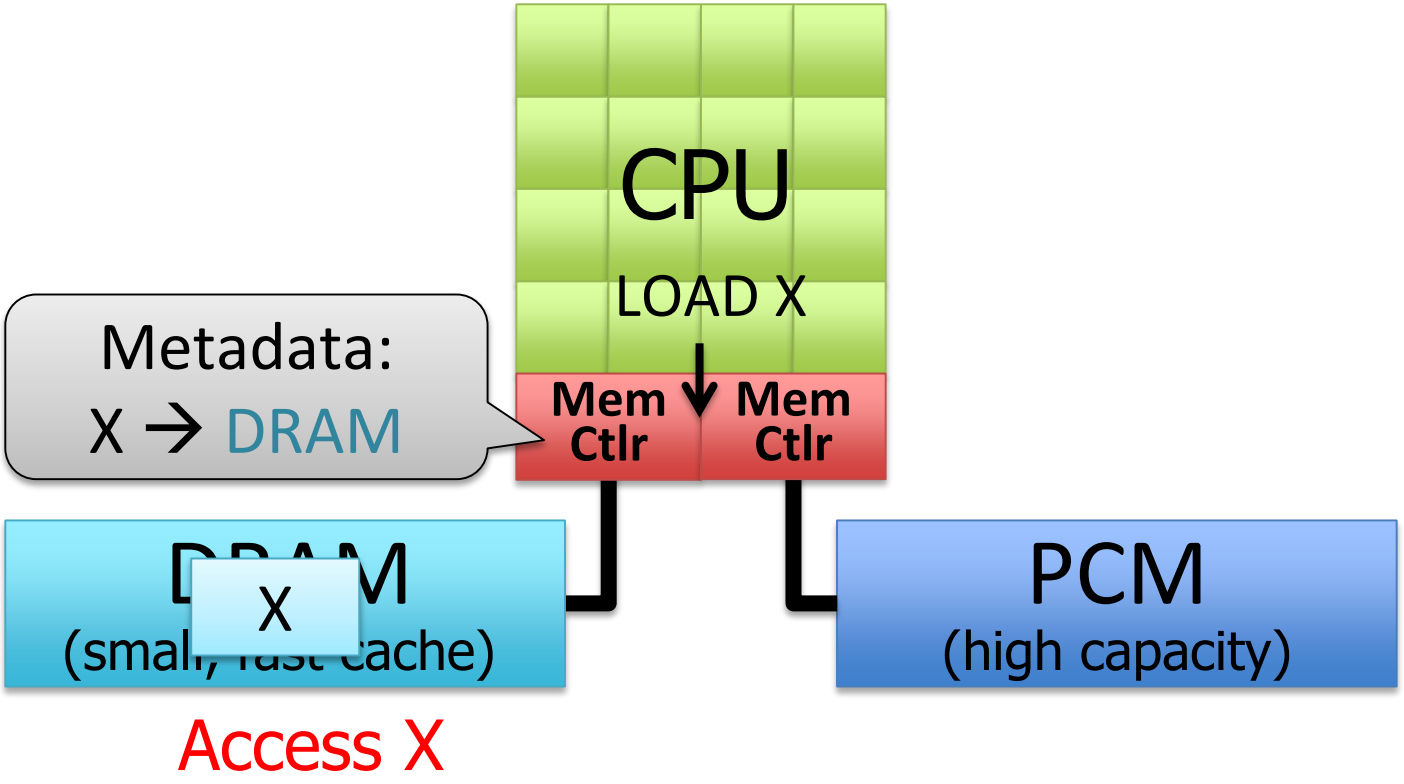
HanBin Yoon, Justin Meza, Rachata Ausavarungnirun, Rachael A. Harding and Onur Mutlu
Carnegie Mellon University
{hanbinyoon,meza,rachata,onur}@cmu.edu, rhardin@mit.edu

Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
  - Background
  - PCM (or Technology X) as DRAM Replacement
  - Hybrid Memory Systems
    - Row-Locality Aware Data Placement
    - Efficient DRAM (or Technology X) Caches
- Conclusions
- Discussion

The Problem with Large DRAM Caches

- A large DRAM cache requires a large metadata (tag + block-based information) store
- How do we design an efficient DRAM cache?



Idea 1: Tags in Memory

- Store tags in the same row as data in DRAM
  - Store metadata in same row as their data
  - Data and metadata can be accessed together

- Benefit: No on-chip tag storage overhead
- Downsides:
  - Cache hit determined only after a DRAM access
  - Cache hit requires two DRAM accesses

Idea 2: Cache Tags in SRAM

- Recall Idea 1: Store all metadata in DRAM
  - To reduce metadata storage overhead

- Idea 2: Cache in on-chip SRAM frequently-accessed metadata
  - Cache only a small amount to keep SRAM size small

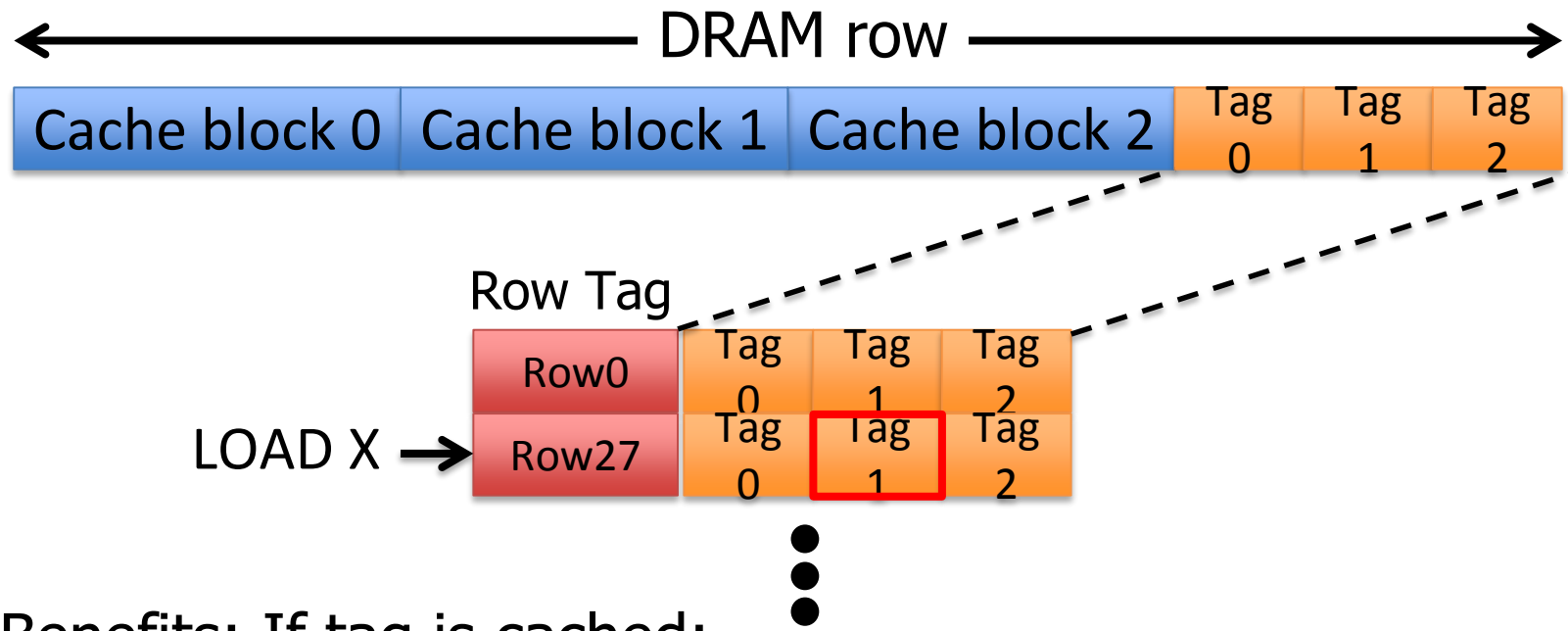
Idea 3: Dynamic Data Transfer Granularity

- Some applications benefit from caching more data
  - They have good spatial locality
- Others do not
  - Large granularity wastes bandwidth and reduces cache utilization

- Idea 3: Simple dynamic caching granularity policy
  - Cost-benefit analysis to determine best DRAM cache block size
  - Group main memory into sets of rows
  - Some row sets follow a fixed caching granularity
  - The rest of main memory follows the best granularity
    - Cost–benefit analysis: access latency versus number of cachings
    - Performed every quantum

TIMBER Tag Management

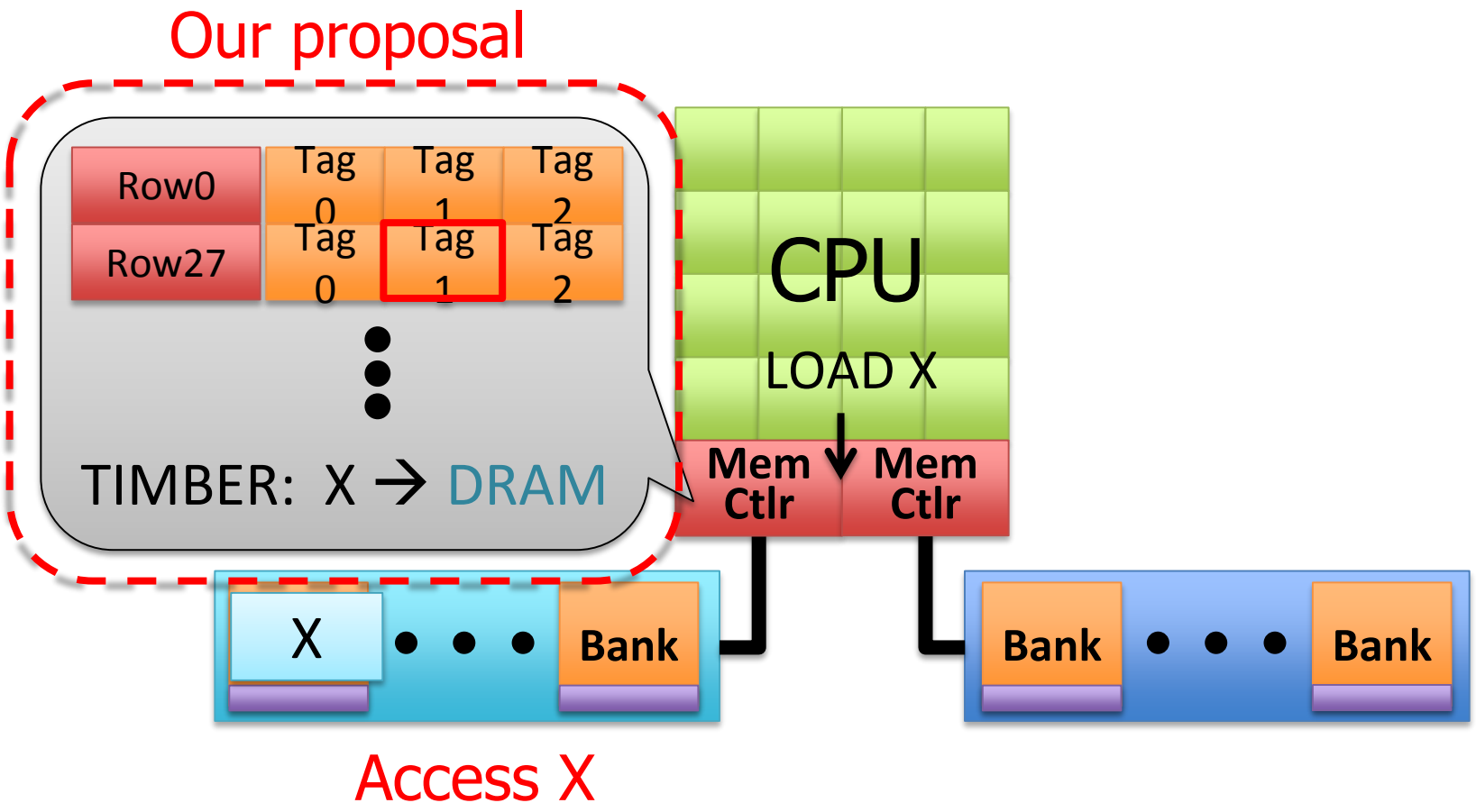
- A Tag-In-Memory BuffER (TIMBER)
  - Stores recently-used tags in a small amount of SRAM



- Benefits: If tag is cached:
  - no need to access DRAM twice
  - cache hit determined quickly

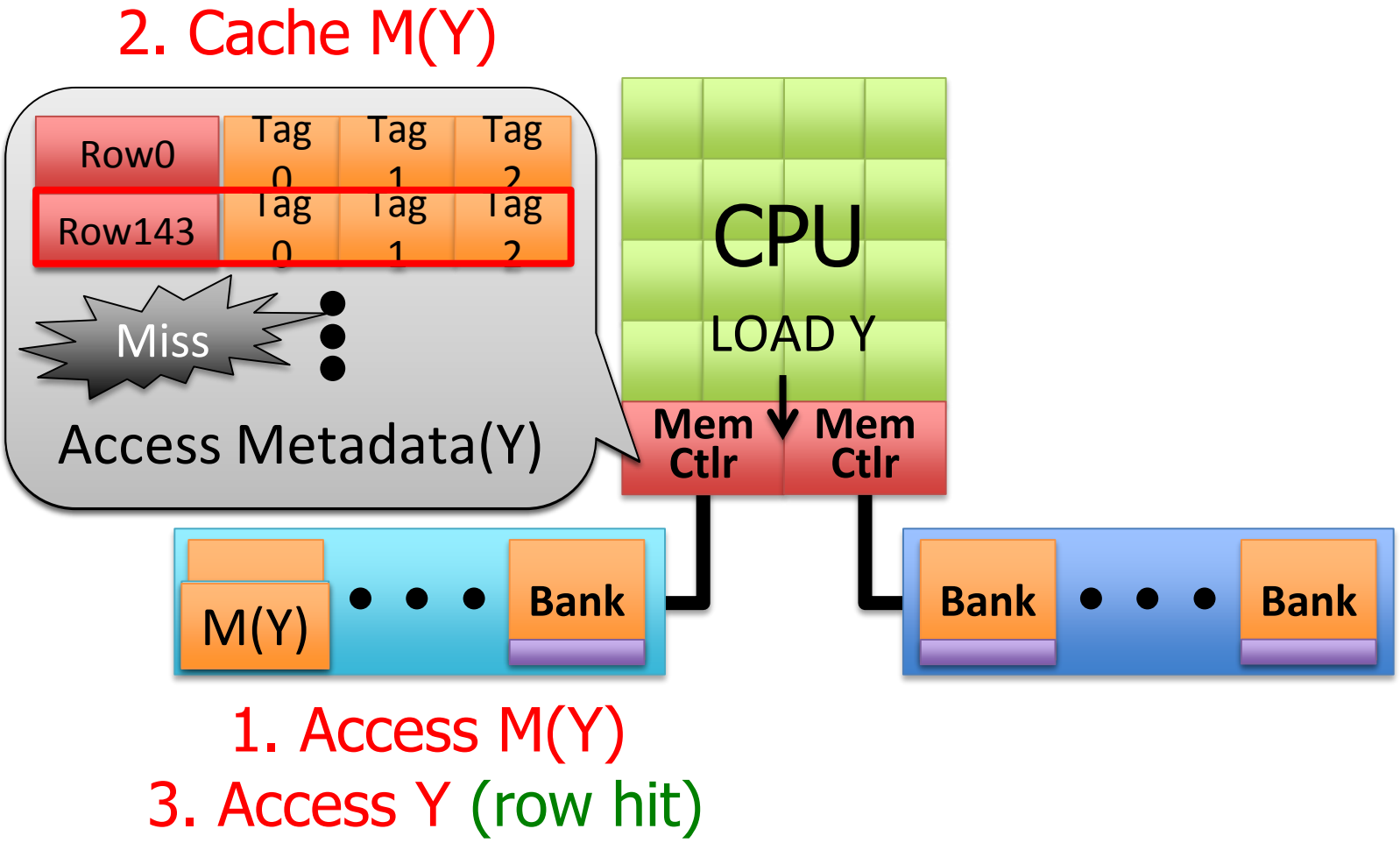
TIMBER Tag Management Example (I)

- Case 1: TIMBER hit

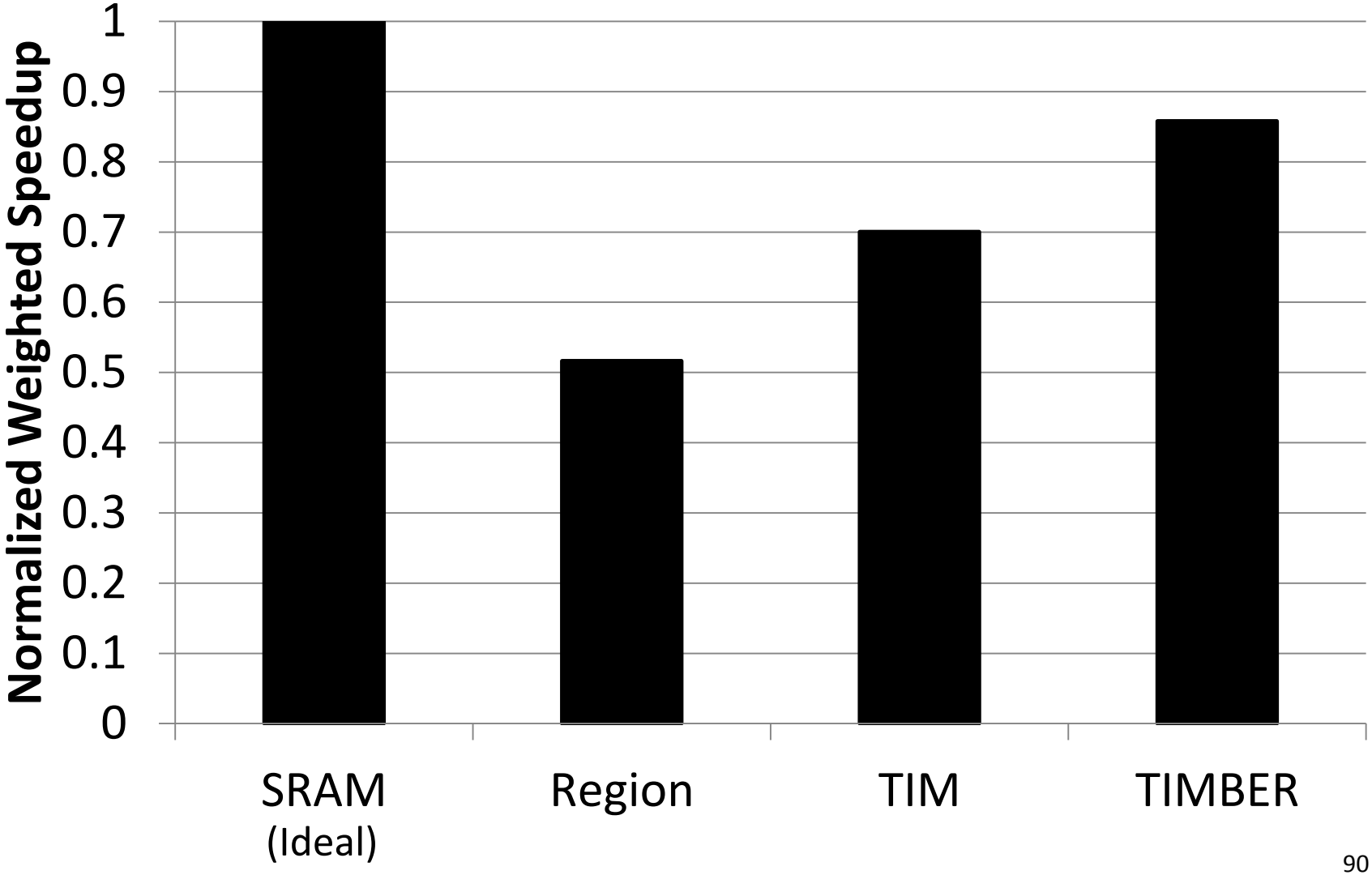


TIMBER Tag Management Example (II)

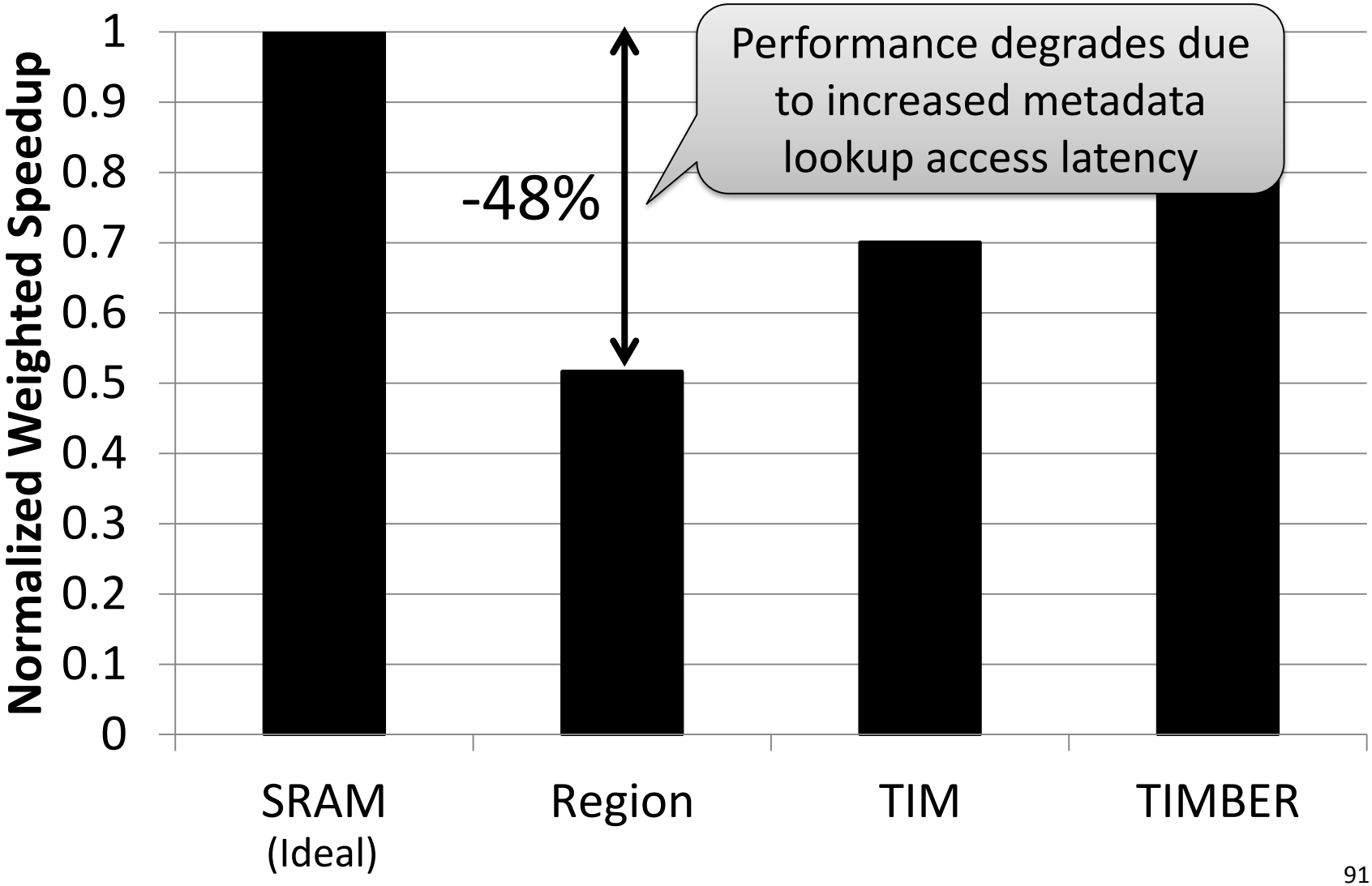
- Case 2: TIMBER miss



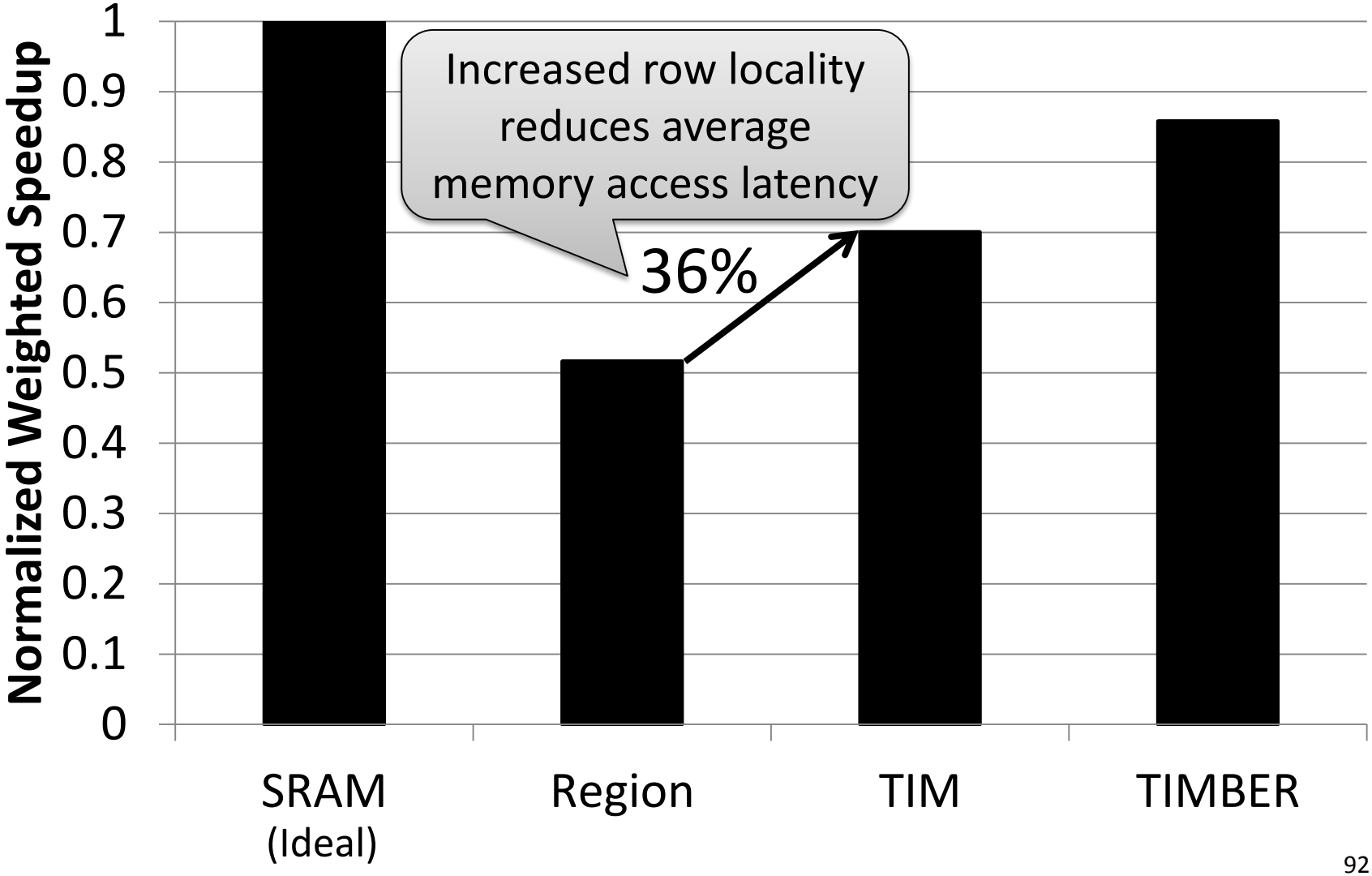
Metadata Storage Performance



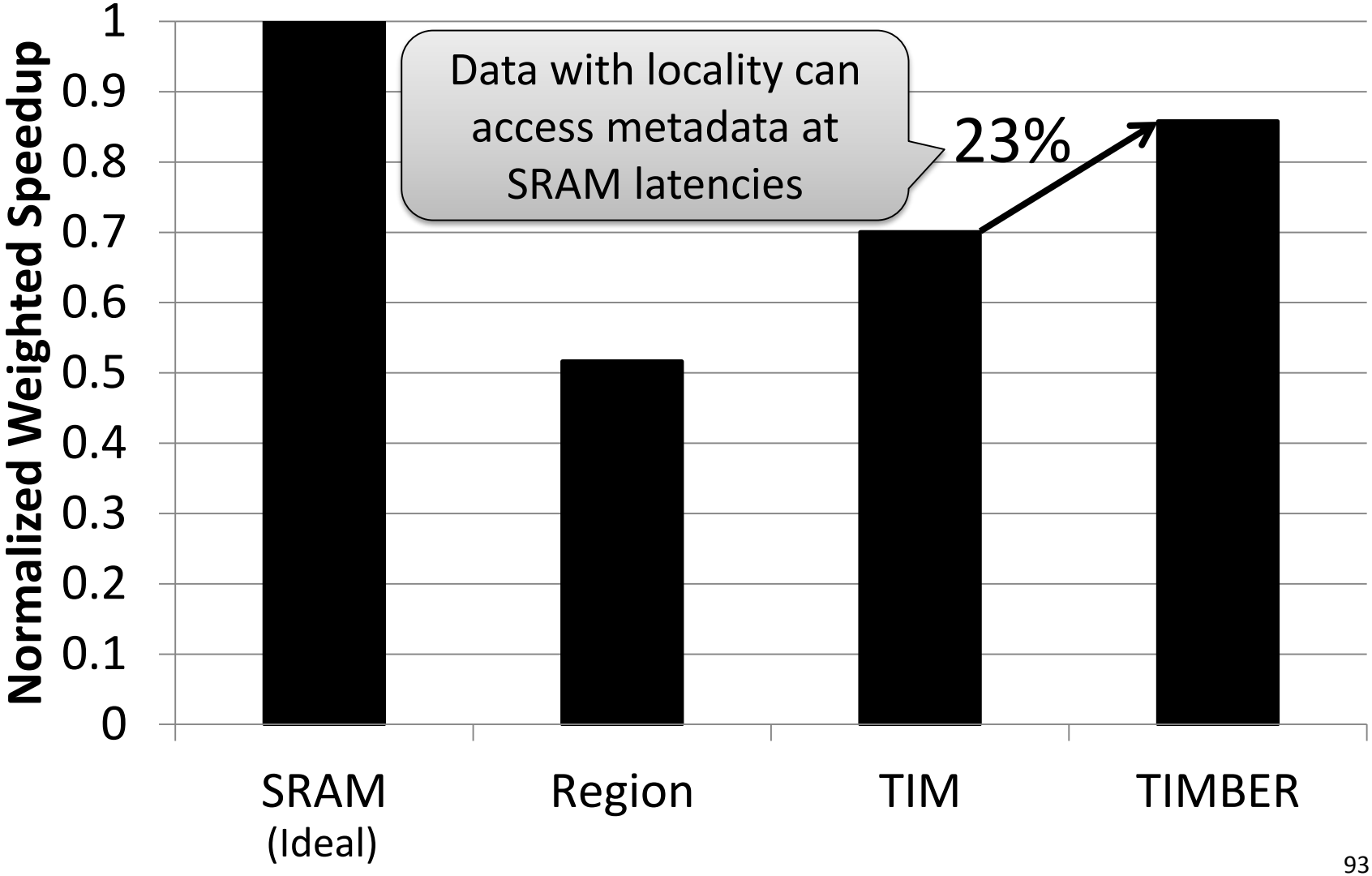
Metadata Storage Performance



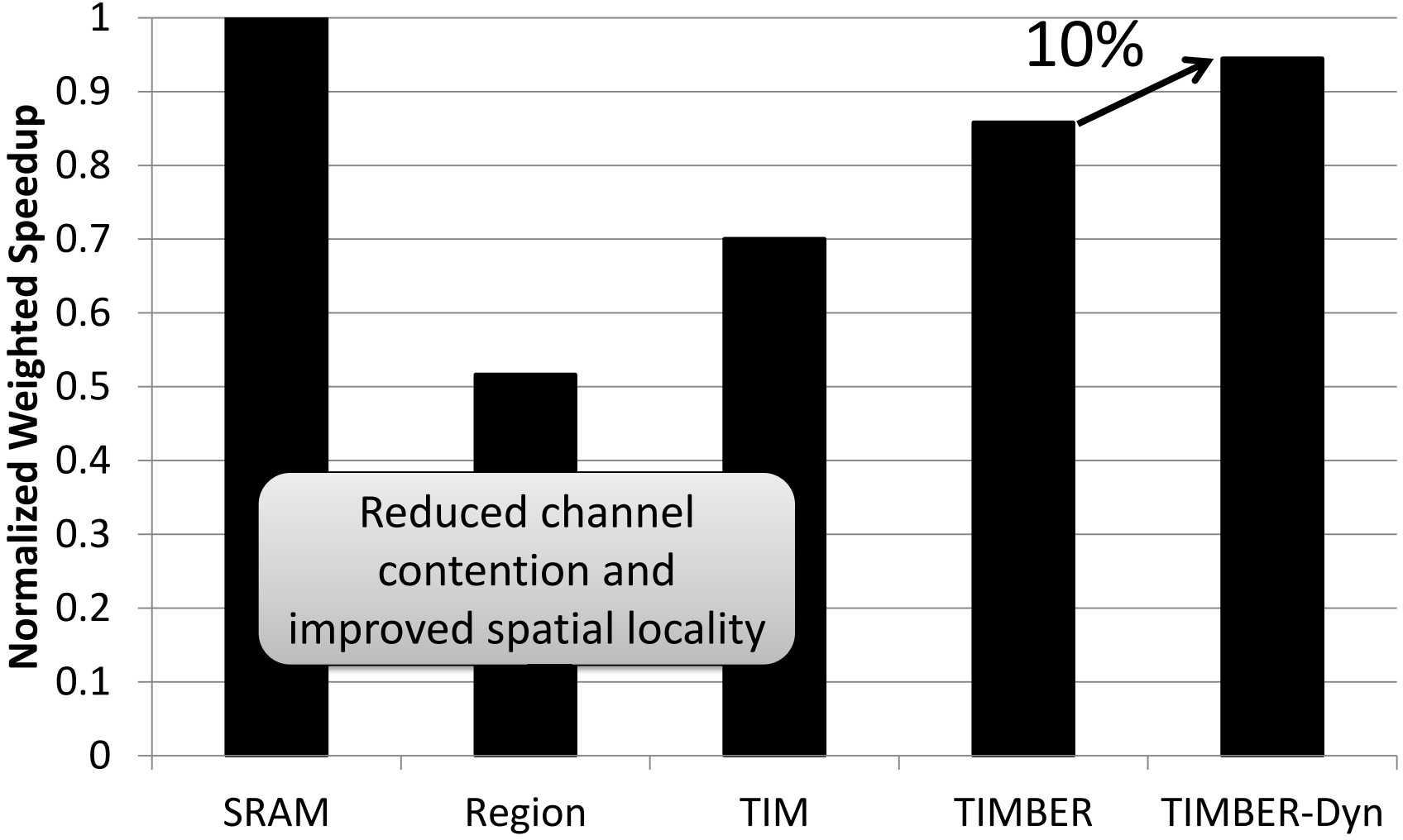
Metadata Storage Performance



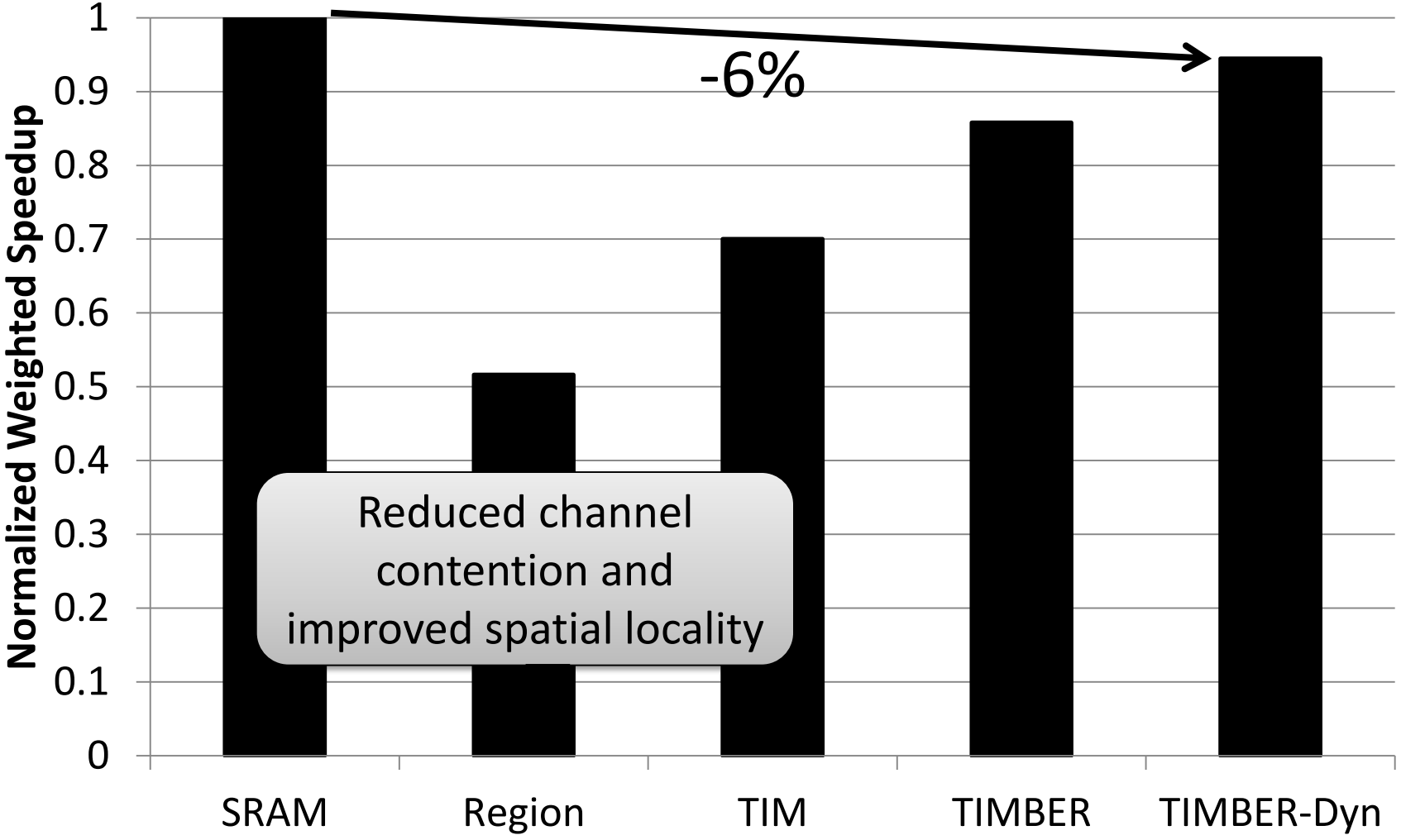
Metadata Storage Performance



Dynamic Granularity Performance

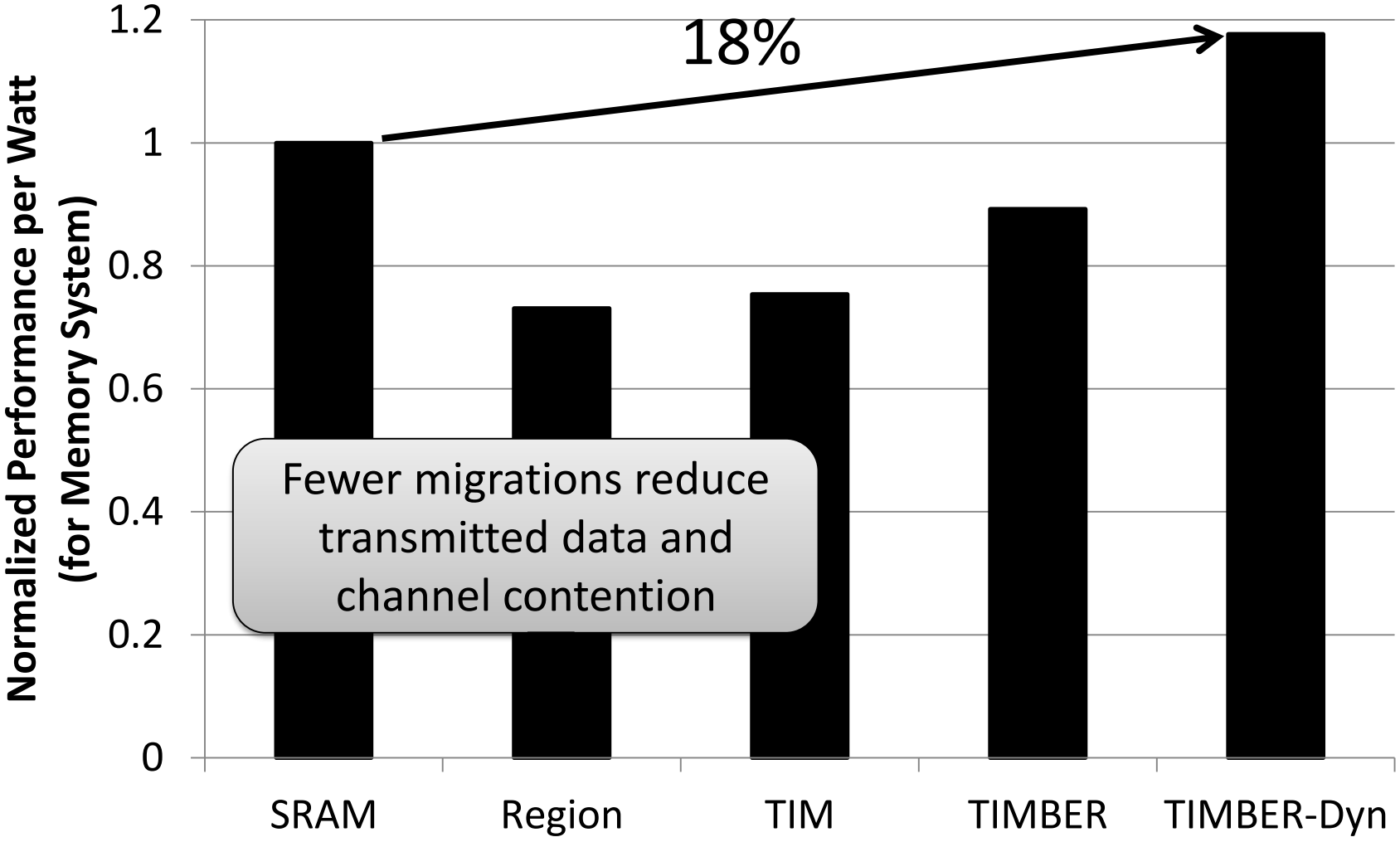


TIMBER Performance



Meza, Chang, Yoon, Mutlu, Ranganathan, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters, 2012.

TIMBER Energy Efficiency



Meza, Chang, Yoon, Mutlu, Ranganathan, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters, 2012.

More on Large DRAM Cache Design

- Justin Meza, Jichuan Chang, HanBin Yoon, Onur Mutlu, and Parthasarathy Ranganathan,
  "Enabling Efficient and Scalable Hybrid Memories Using Fine-Granularity DRAM Cache Management"
  IEEE Computer Architecture Letters (***CAL***), February 2012.

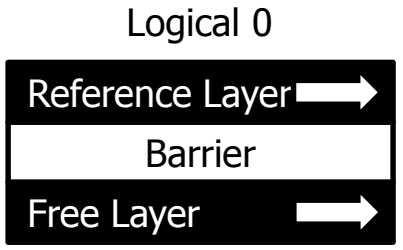
Enabling Efficient and Scalable Hybrid Memories Using Fine-Granularity DRAM Cache Management

Justin Meza* Jichuan Chang† HanBin Yoon* Onur Mutlu* Parthasarathy Ranganathan†
*Carnegie Mellon University †Hewlett-Packard Labs
{meza,hanbinyoon,onur}@cmu.edu {jichuan.chang,partha.ranganathan}@hp.com

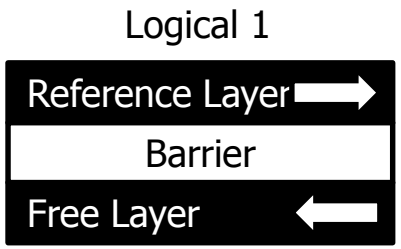
STT-MRAM As Main Memory

STT-MRAM as Main Memory

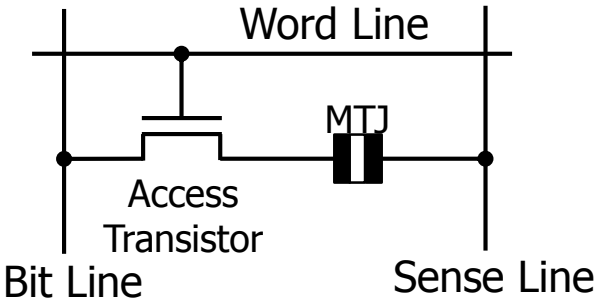
- Magnetic Tunnel Junction (MTJ) device
  - Reference layer: Fixed magnetic orientation
  - Free layer: Parallel or anti-parallel



- Magnetic orientation of the free layer determines logical state of device
  - High vs. low resistance



- Write: Push large current through MTJ to change orientation of free layer
- Read: Sense current flow



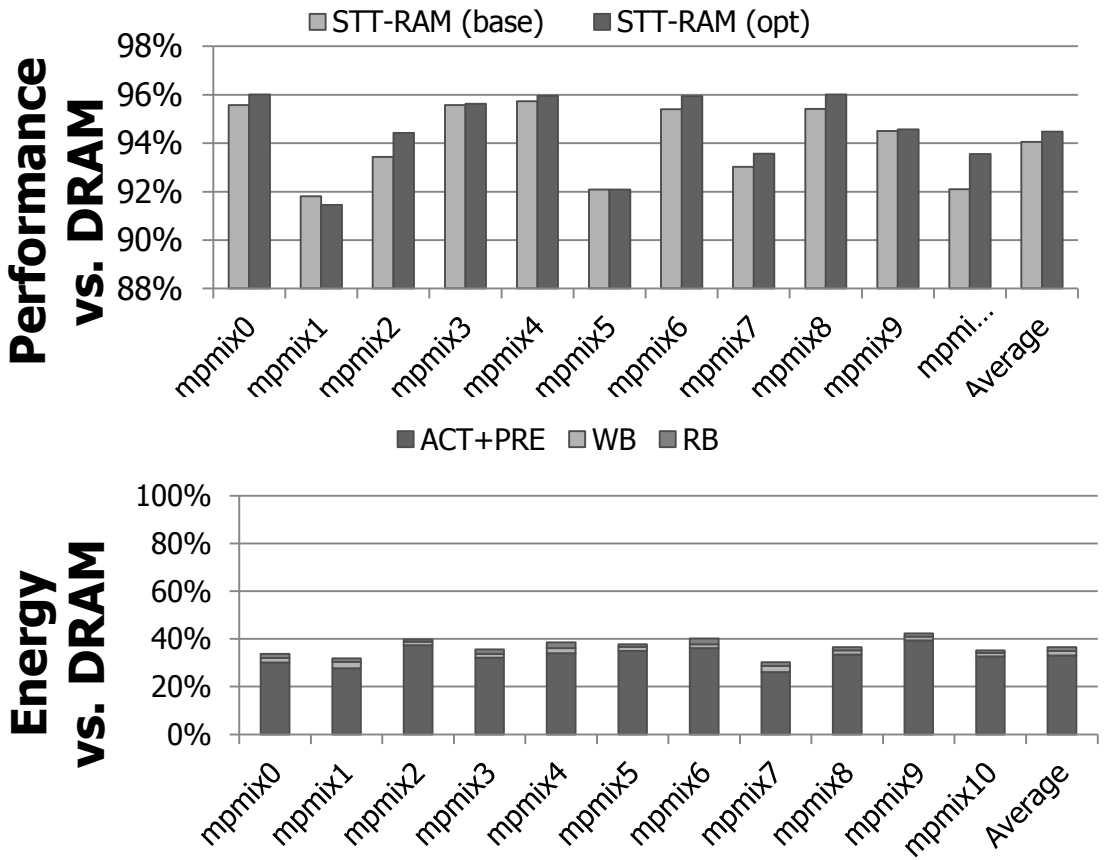
- Kultursay et al., "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," ISPASS 2013.

STT-MRAM: Pros and Cons

- Pros over DRAM
  - Better technology scaling
  - Non volatility
  - Low idle power (no refresh)
- Cons
  - Higher write latency
  - Higher write energy
  - Reliability?
- Another level of freedom
  - Can trade off non-volatility for lower write latency/energy (by reducing the size of the MTJ)

Architected STT-MRAM as Main Memory

- 4-core, 4GB main memory, multiprogrammed workloads
- ~6% performance loss, ~60% energy savings vs. DRAM



Kultursay+, "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," ISPASS 2013.

More on STT-MRAM as Main Memory

- Emre Kultursay, Mahmut Kandemir, Anand Sivasubramaniam, and Onur Mutlu,
  "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative"
  *Proceedings of the 2013 IEEE International Symposium on Performance Analysis of Systems and Software (**ISPASS**)*, Austin, TX, April 2013. Slides (pptx) (pdf)

Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative

Emre Kültürsay*, Mahmut Kandemir*, Anand Sivasubramaniam*, and Onur Mutlu†
*The Pennsylvania State University and †Carnegie Mellon University

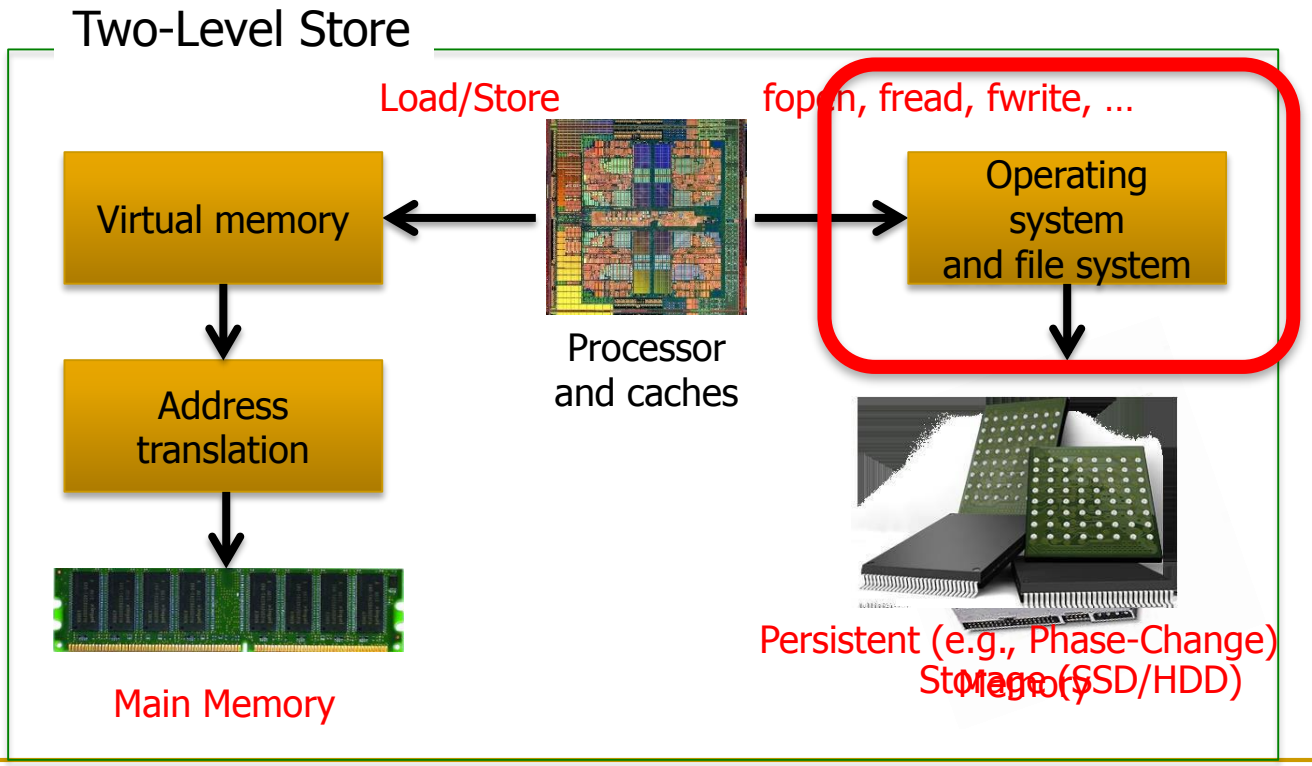
Other Opportunities with Emerging Technologies

- Merging of memory and storage
  - e.g., a single interface to manage all data
- New applications
  - e.g., ultra-fast checkpoint and restore
- More robust system design
  - e.g., reducing data loss
- Processing tightly-coupled with memory
  - e.g., enabling efficient search and filtering

Merging of Memory and Storage: Persistent Memory Managers

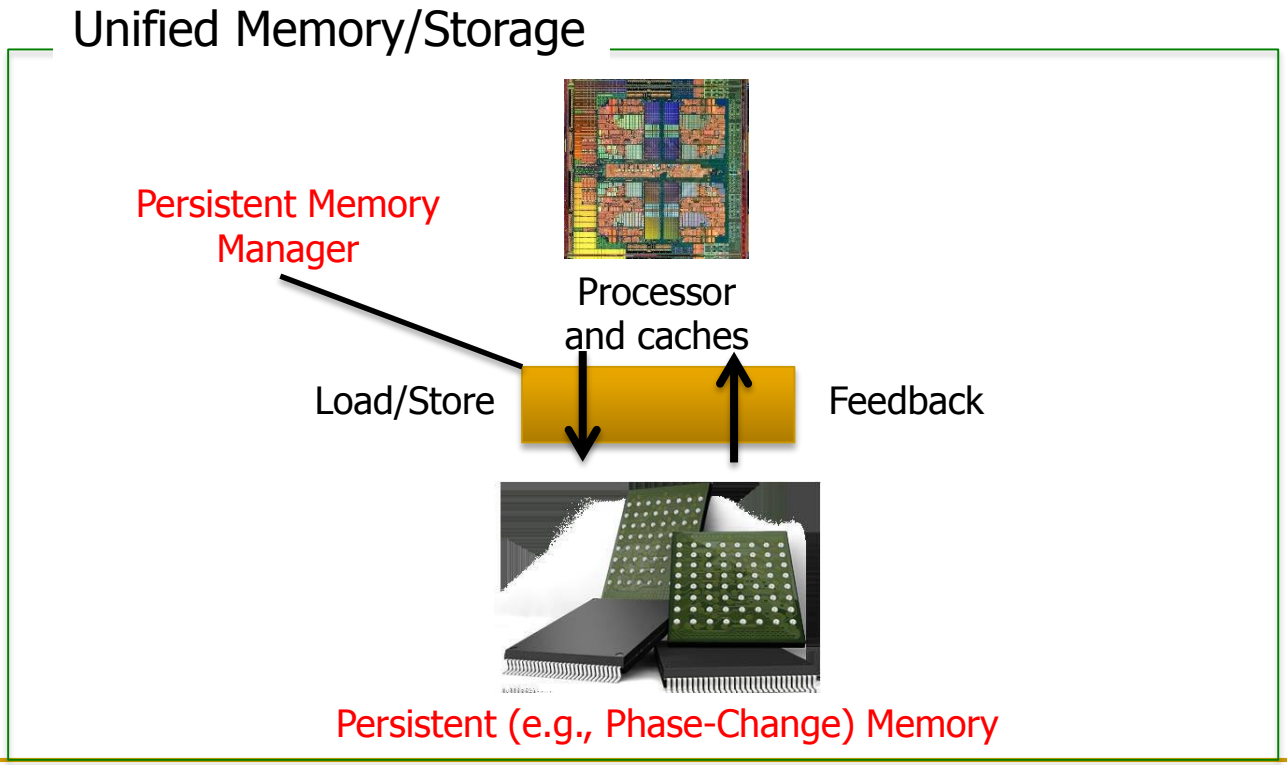
Coordinated Memory and Storage with NVM (I)

- The traditional two-level storage model is a bottleneck with NVM
  - **Volatile** data in memory → a **load/store** interface
  - **Persistent** data in storage → a **file system** interface
  - Problem: Operating system (OS) and file system (FS) code to locate, translate, buffer data become performance and energy bottlenecks with fast NVM stores



Coordinated Memory and Storage with NVM (II)

- Goal: Unify memory and storage management in a single unit to eliminate wasted work to locate, transfer, and translate data
  - Improves both energy and performance
  - Simplifies programming model as well



Meza+, "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory," WEED 2013.

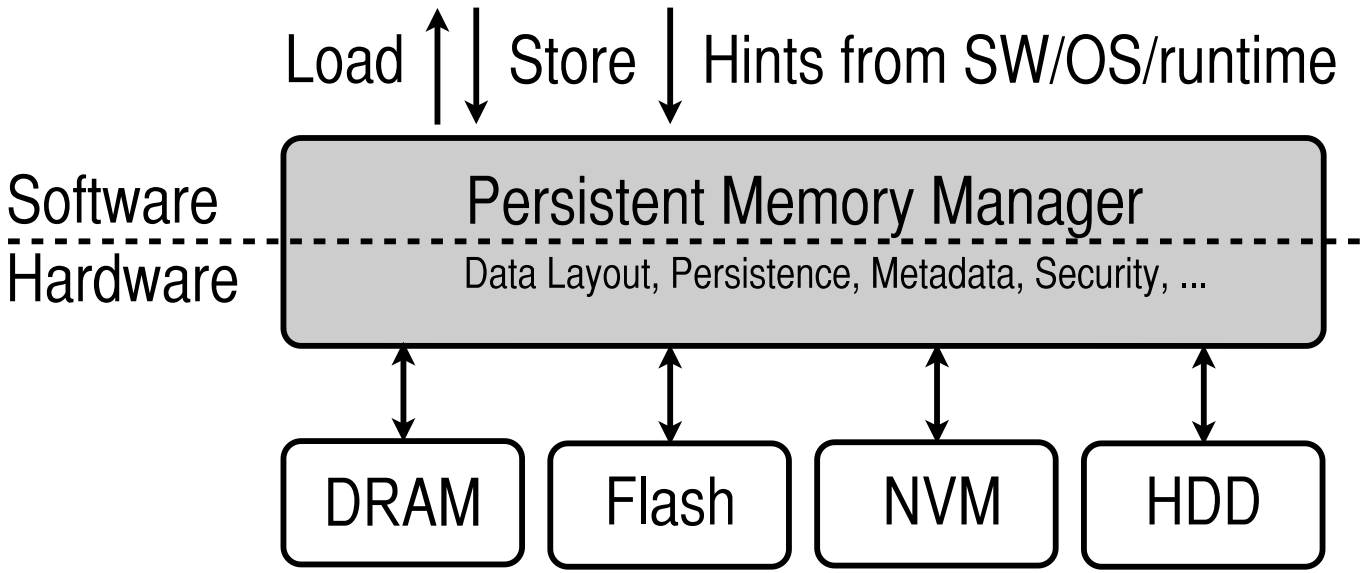
The Persistent Memory Manager (PMM)

- Exposes a load/store interface to access persistent data
  - Applications can directly access persistent memory → no conversion, translation, location overhead for persistent data
- Manages data placement, location, persistence, security
  - To get the best of multiple forms of storage
- Manages metadata storage and retrieval
  - This can lead to overheads that need to be managed
- Exposes hooks and interfaces for system software
  - To enable better data placement and management decisions
- Meza+, "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory," WEED 2013.

The Persistent Memory Manager (PMM)

```
1  int main(void) {
2    // data in file.dat is persistent
3    FILE myData = "file.dat";
4    myData = new int[64];
5  }
6  void updateValue(int n, int value) {
7    FILE myData = "file.dat";
8    myData[n] = value; // value is persistent
9  }
```

Persistent objects



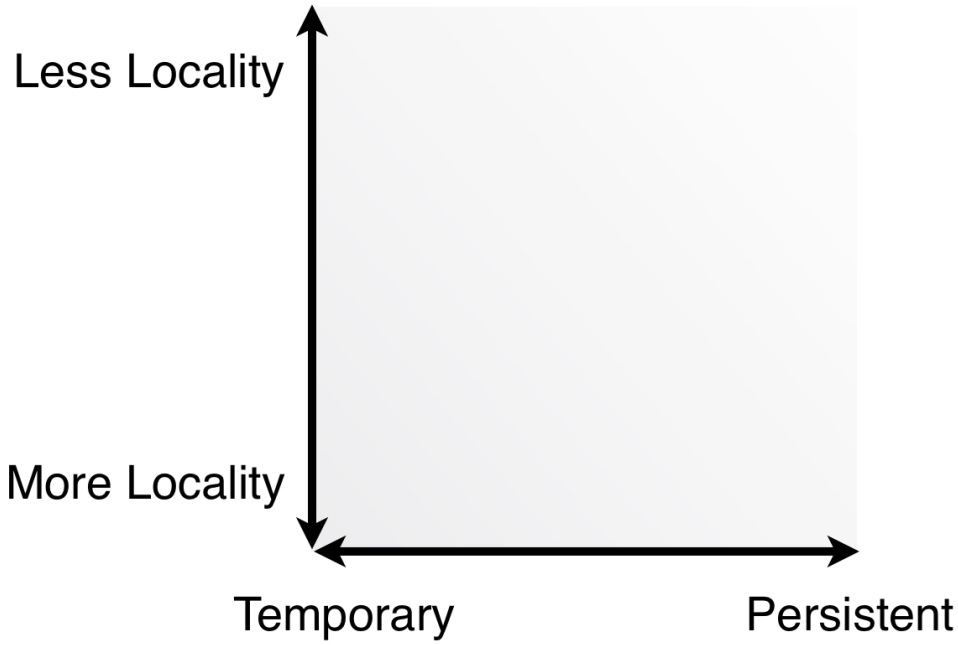
PMM uses access and hint information to allocate, locate, migrate and access data in the heterogeneous array of devices

Efficient Data Mapping among Heterogeneous Devices

- A persistent memory exposes a large, persistent address space
  - But it may use many different devices to satisfy this goal
  - From fast, low-capacity volatile DRAM to slow, high-capacity non-volatile HDD or Flash
  - And other NVM devices in between

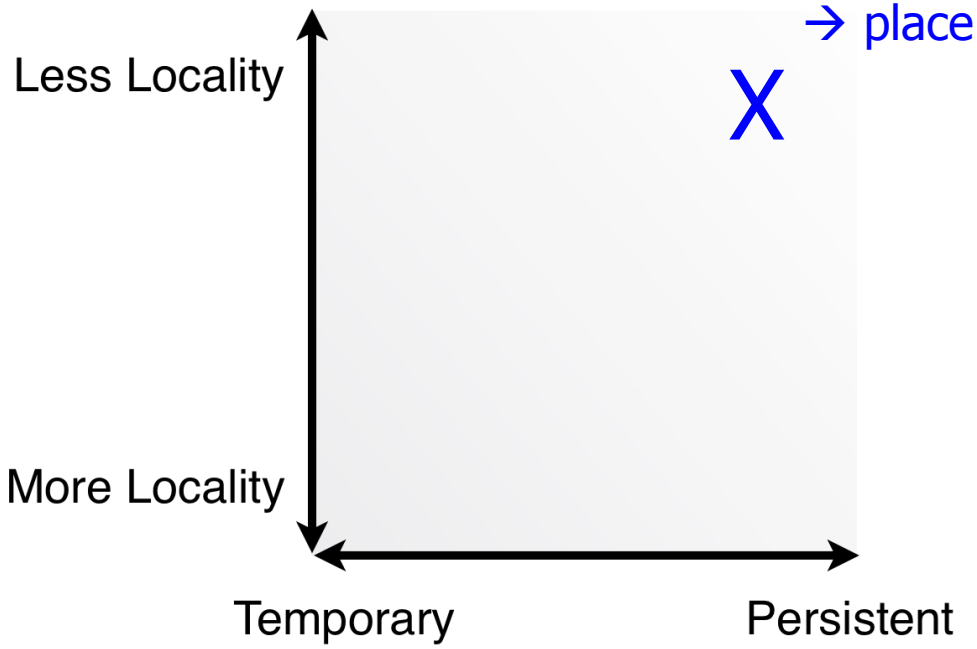
- Performance and energy can benefit from good placement of data among these devices
  - Utilizing the strengths of each device and avoiding their weaknesses, if possible
  - For example, consider two important application characteristics: locality and persistence

Efficient Data Mapping among Heterogeneous Devices

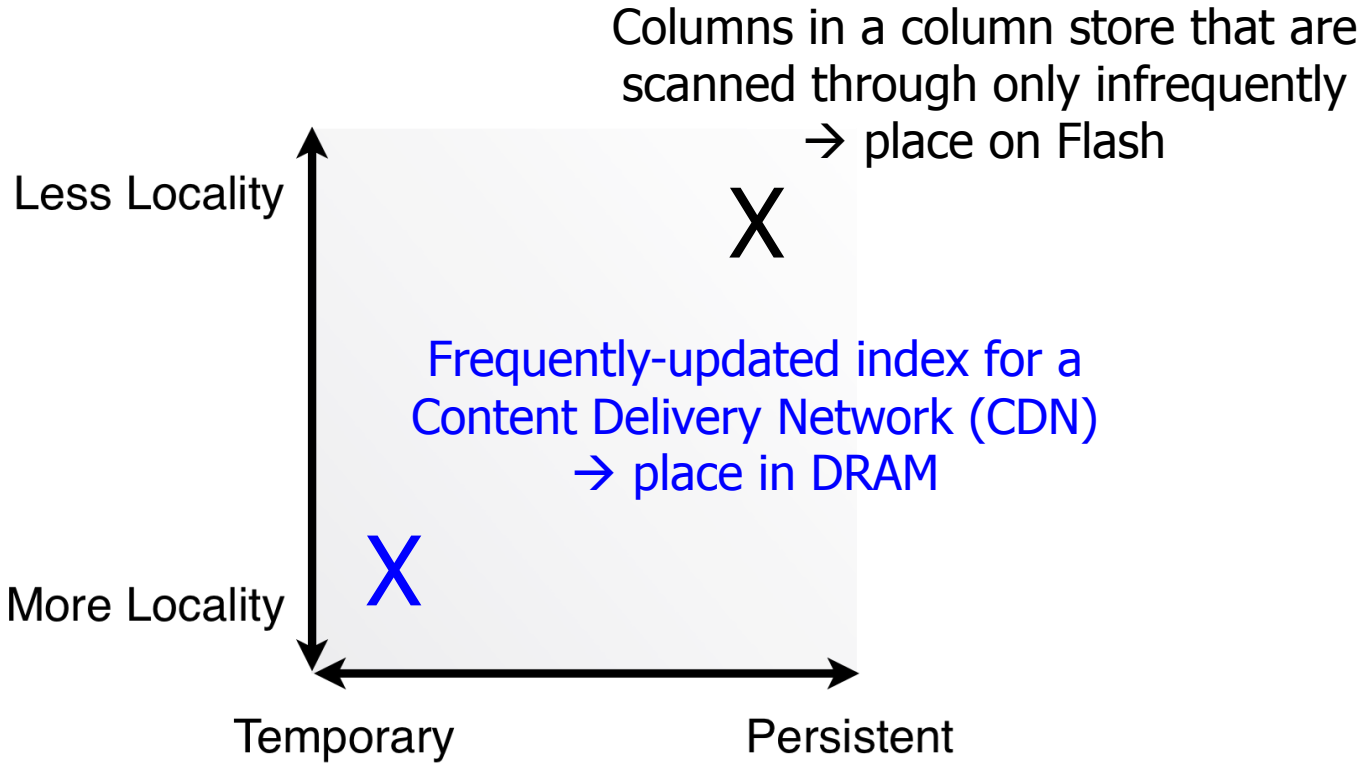


Efficient Data Mapping among Heterogeneous Devices

Columns in a column store that are scanned through only infrequently → place on Flash

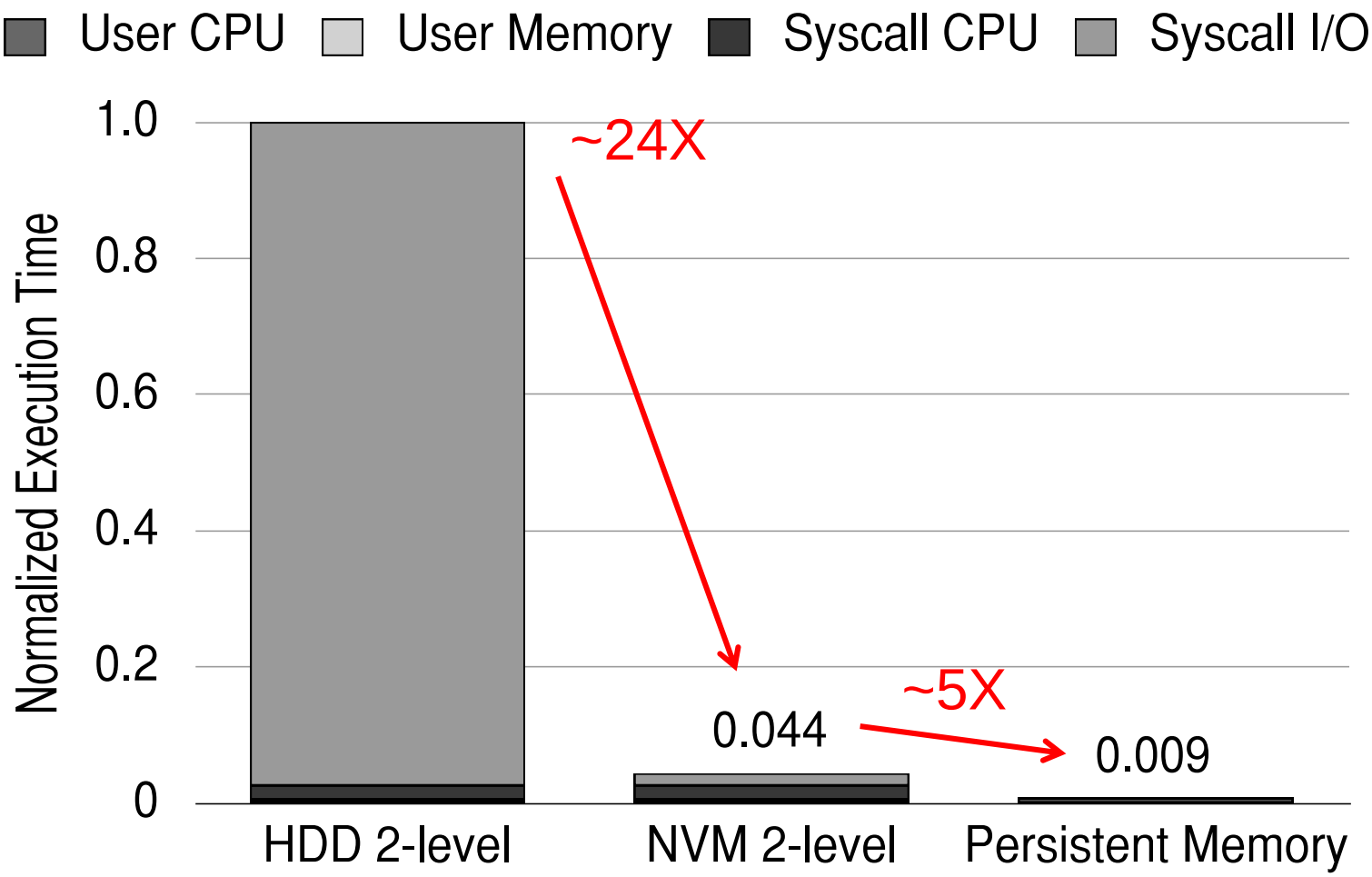


Efficient Data Mapping among Heterogeneous Devices



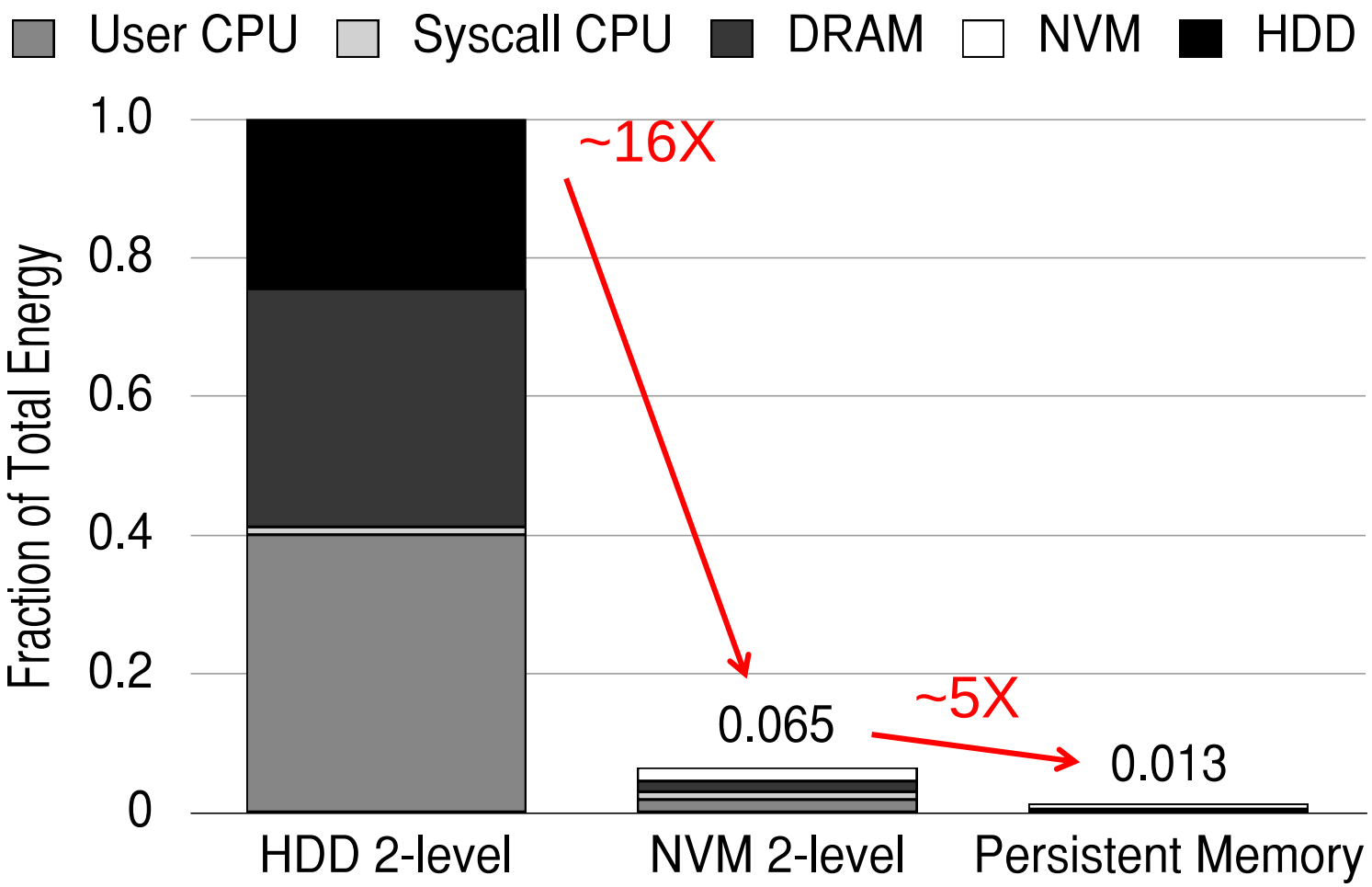
Applications or system software can provide hints for data placement

Performance Benefits of a Single-Level Store



Meza+, "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory," WEED 2013.

Energy Benefits of a Single-Level Store



Meza+, "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory," WEED 2013.

More on Single-Level Stores

- Justin Meza, Yixin Luo, Samira Khan, Jishen Zhao, Yuan Xie, and Onur Mutlu,
  "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory"
  *Proceedings of the 5th Workshop on Energy-Efficient Design (**WEED**)*, Tel-Aviv, Israel, June 2013. Slides (pptx) Slides (pdf)

A Case for Efficient Hardware/Software Cooperative Management of Storage and Memory

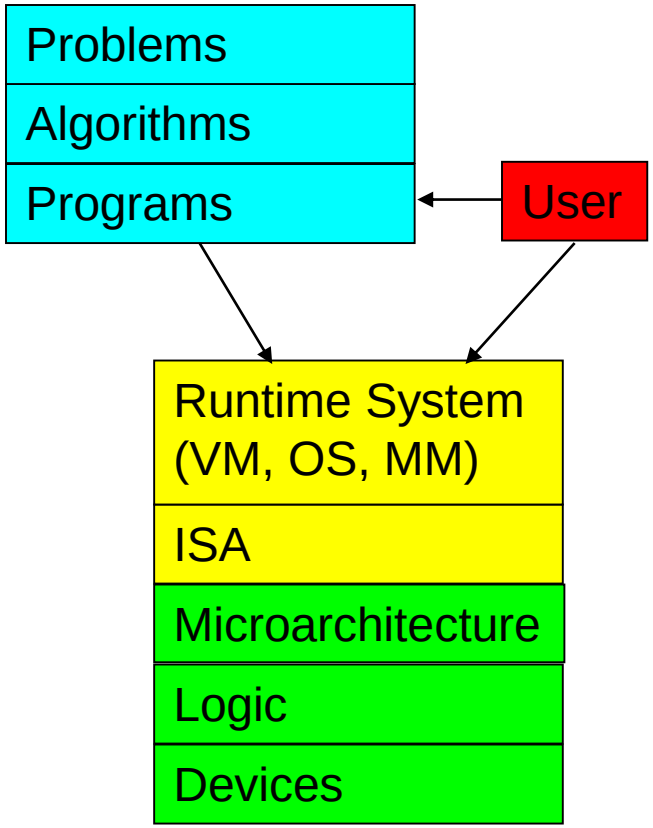
Justin Meza* Yixin Luo* Samira Khan*‡ Jishen Zhao† Yuan Xie†§ Onur Mutlu*
*Carnegie Mellon University †Pennsylvania State University ‡Intel Labs §AMD Research

Enabling and Exploiting NVM: Issues

- Many issues and ideas from technology layer to algorithms layer

- Enabling NVM and hybrid memory
  - How to tolerate errors?
  - How to enable secure operation?
  - How to tolerate performance and power shortcomings?
  - How to minimize cost?

- Exploiting emerging technologies
  - How to exploit non-volatility?
  - How to minimize energy consumption?
  - How to exploit NVM on chip?



Security Challenges of Emerging Technologies

1. Limited endurance → Wearout attacks

2. Non-volatility → Data persists in memory after crash/poweroff
   → Inconsistency and retrieval of privileged information

3. Multiple bits per cell → Information leakage (via side channel)

Securing Emerging Memory Technologies

1. Limited endurance → Wearout attacks
   - Better architecting of memory chips to absorb writes
   - Hybrid memory system management
   - Online wearout attack detection

2. Non-volatility → Data persists in memory after crash/poweroff
   → Inconsistency and retrieval of privileged information
   - Efficient encryption/decryption of whole main memory
   - Hybrid memory system management

3. Multiple bits per cell → Information leakage (via side channel)
   - System design to hide side channel information

Agenda

- Major Trends Affecting Main Memory
- Requirements from an Ideal Main Memory System
- Opportunity: Emerging Memory Technologies
  - Background
  - PCM (or Technology X) as DRAM Replacement
  - Hybrid Memory Systems
- Conclusions
- Discussion

Summary: Memory Scaling (with NVM)

- Main memory scaling problems are a critical bottleneck for system performance, efficiency, and usability

- Solution 1: Tolerate DRAM
- Solution 2: Enable emerging memory technologies
  - Replace DRAM with NVM by architecting NVM chips well
  - Hybrid memory systems with automatic data management

- An exciting topic with many other solution directions & ideas
  - Hardware/software/device cooperation essential
  - Memory, storage, controller, software/app co-design needed
  - Coordinated management of persistent memory and storage
  - Application and hardware cooperative management of NVM

18-740/640
Computer Architecture
Lecture 9: Emerging Memory Technologies

Prof. Onur Mutlu
Carnegie Mellon University
Fall 2015, 9/30/2015