

18-740: Computer Architecture  
Recitation 4:  
Rethinking Memory System Design

Prof. Onur Mutlu

Carnegie Mellon University

Fall 2015

September 22, 2015

# Agenda

---

- Review Assignments for Next Week
- Rethinking Memory System Design (Continued)
  - With a lot of discussion, hopefully

# Review Assignments for Next Week

# Required Reviews

---

- Due Tuesday Sep 29 @ 3pm
- Enter your reviews on the review website
- Please discuss ideas and thoughts on Piazza

# Review Paper 1 (Required)

---

- Eiman Ebrahimi, Chang Joo Lee, Onur Mutlu, and Yale N. Patt, **"Fairness via Source Throttling: A Configurable and High-Performance Fairness Substrate for Multi-Core Memory Systems"**  
*Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 335-346, Pittsburgh, PA, March 2010. [Slides \(pdf\)](#)
- Related paper:
  - Kevin Chang, Rachata Ausavarungnirun, Chris Fallin, and Onur Mutlu, **"HAT: Heterogeneous Adaptive Throttling for On-Chip Networks"**  
*Proceedings of the 24th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, New York, NY, October 2012.

# Review Paper 2 (Required)

---

- Rachata Ausavarungnirun, Saugata Ghose, Onur Kayiran, Gabriel L. Loh, Chita R. Das, Mahmut T. Kandemir, and Onur Mutlu, **"Exploiting Inter-Warp Heterogeneity to Improve GPGPU Performance"**  
*Proceedings of the 24th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, San Francisco, CA, USA, October 2015.
- Related paper:
  - Wilson W. L. Fung, Ivan Sham, George Yuan, and Tor M. Aamodt, **Dynamic Warp Formation and Scheduling for Efficient GPU Control Flow**, In proceedings of the 40th IEEE/ACM International Symposium on Microarchitecture (**MICRO-40**), pp. 407-418, Chicago, IL, December 1-5, 2007. [slides](#). [pre-print](#)

# Review Paper 3 (Required)

---

- Donghyuk Lee, Yoongu Kim, Vivek Seshadri, Jamie Liu, Lavanya Subramanian, and Onur Mutlu,  
**"Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture"**  
*Proceedings of the 19th International Symposium on High-Performance Computer Architecture (HPCA)*, Shenzhen, China, February 2013.  
Slides (pptx)
- Related paper
  - Donghyuk Lee, Yoongu Kim, Gennady Pekhimenko, Samira Khan, Vivek Seshadri, Kevin Chang, and Onur Mutlu,  
**"Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case"**  
*Proceedings of the 21st International Symposium on High-Performance Computer Architecture (HPCA)*, Bay Area, CA, February 2015.  
[Slides (pptx) (pdf)] [Full data sets]

# Review Paper 4 (Optional)

---

- Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu,  
**"A Large-Scale Study of Flash Memory Errors in the Field"**  
*Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**), Portland, OR, June 2015.*
- Related paper
  - Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu,  
**"Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field"**  
*Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**), Rio de Janeiro, Brazil, June 2015.*  
[\[Slides \(pptx\)\]](#) [\(pdf\)](#) [\[DRAM Error Model\]](#)



# Project Proposal

---

- Due Friday
  - September 25, 2015
- Make sure your project is vetted by me before you write your proposal

# Still Consider: Another Possible Project

---

- GPU Warp Scheduling Championship
- [http://adwaitjog.github.io/gpu\\_scheduling.html](http://adwaitjog.github.io/gpu_scheduling.html)

# Rethinking Memory System Design

# Some Promising Directions

---

- **New memory architectures**

- **Rethinking DRAM** and flash memory
- **A lot of hope in fixing DRAM**

- **Enabling emerging NVM technologies**

- **Hybrid memory systems**
- **Single-level memory and storage**
- **A lot of hope in hybrid memory systems and single-level stores**

- **System-level memory/storage QoS**

- **A lot of hope in designing a predictable system**

# Rethinking DRAM

---

- In-Memory Computation
- Refresh
- Reliability
- Latency
- Bandwidth
- Energy
- Memory Compression

# Two Key Questions in 3D Stacked PIM

---

- What is the **minimal processing-in-memory support** we can provide ?
  - without changing the system significantly
  - while achieving significant benefits of processing in 3D-stacked memory
- How can we accelerate important applications if we **use 3D-stacked memory as a coarse-grained accelerator?**
  - what is the architecture and programming model?
  - what are the mechanisms for acceleration?

# A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

[A Scalable Processing-in-Memory Accelerator  
for Parallel Graph Processing \(Ahn et al., ISCA 2015\)](#)

# Large-Scale Graph Processing

- Large graphs are everywhere



36 Million  
Wikipedia Pages



1.4 Billion  
Facebook Users

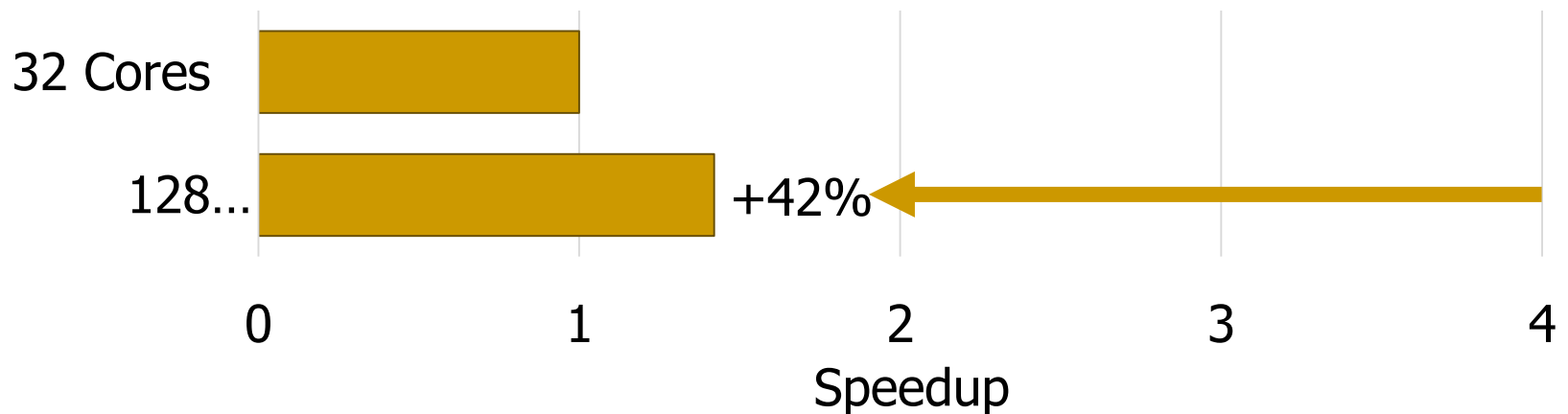


300 Million  
Twitter Users



30 Billion  
Instagram Photos

- Scalable large-scale graph processing is challenging

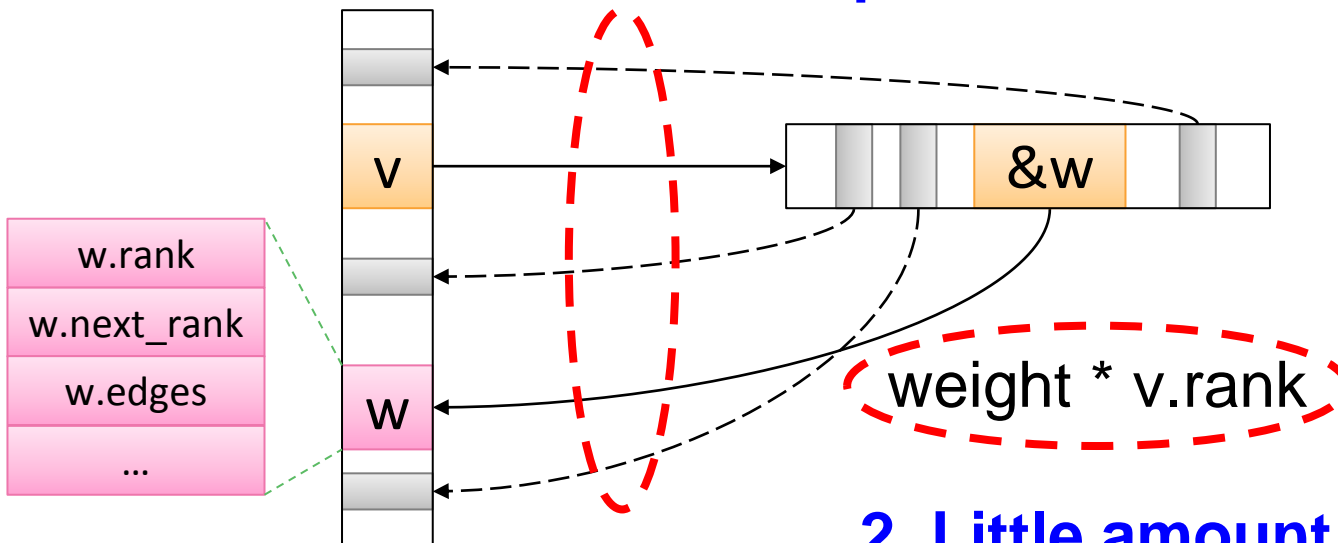




# Key Bottlenecks in Graph Processing

```
for (v: graph.vertices) {  
  for (w: v.successors) {  
    w.next_rank += weight * v.rank;  
  }  
}
```

1. Frequent random memory accesses



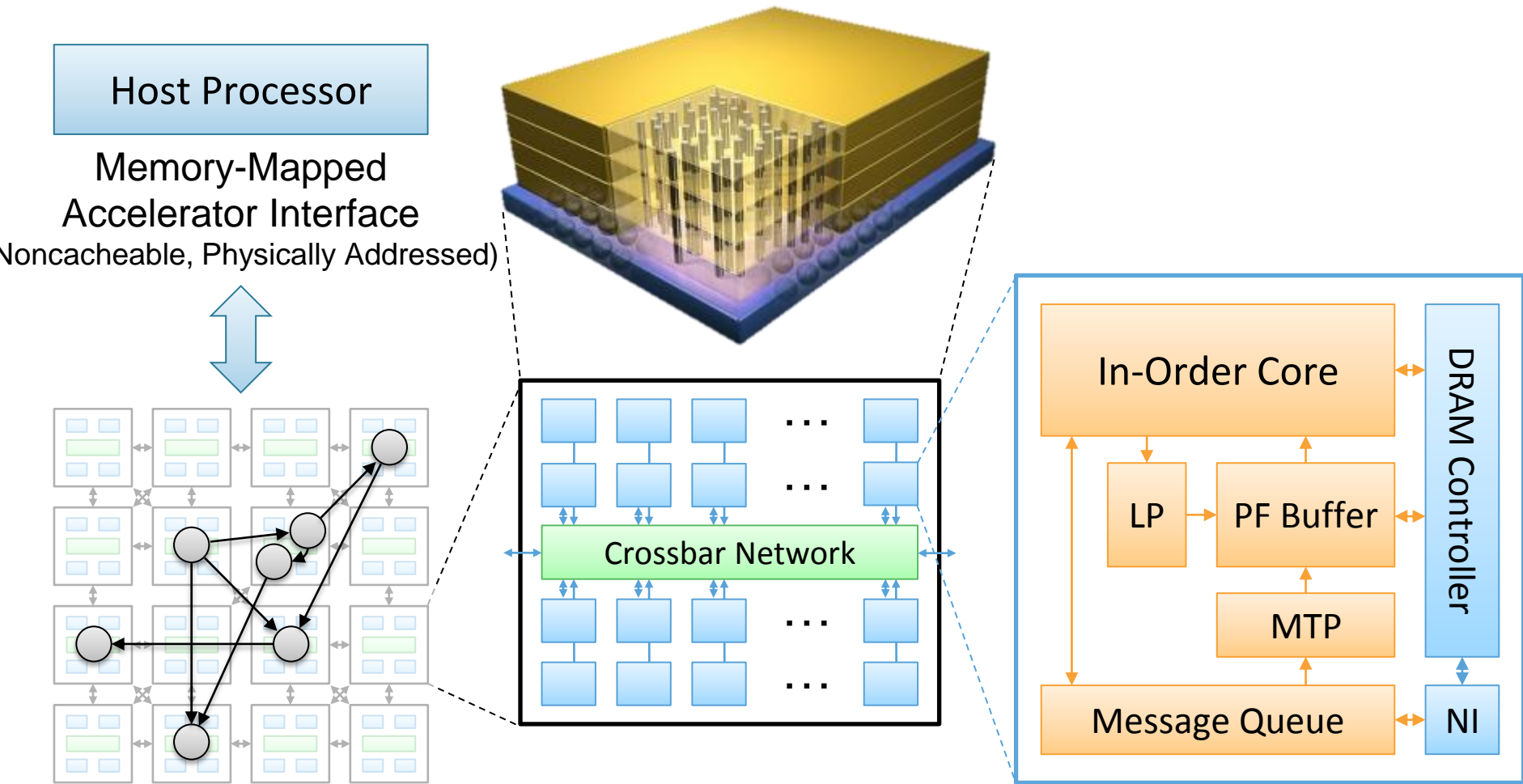
2. Little amount of computation

# Challenges in Scalable Graph Processing

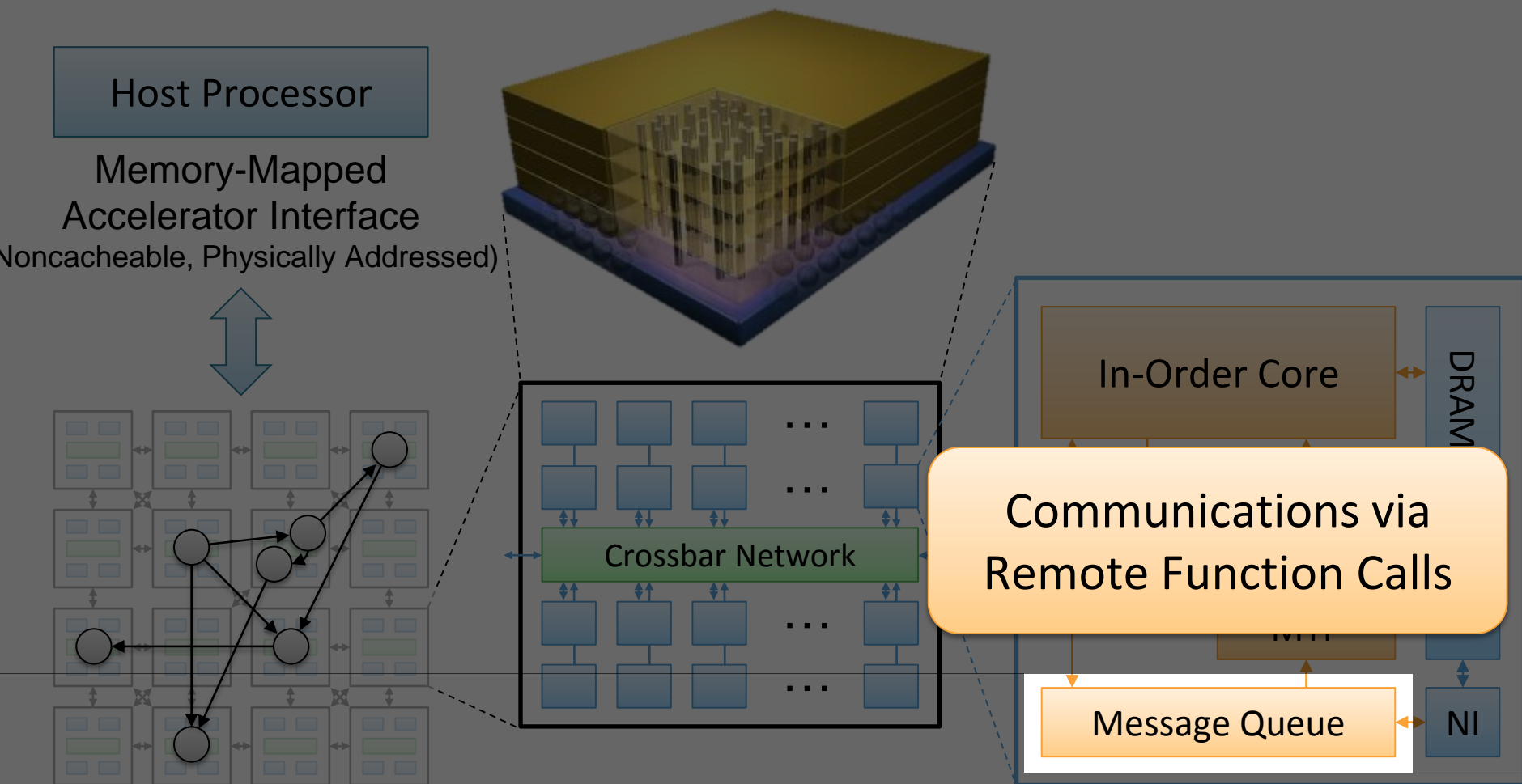
---

- **Challenge 1:** How to provide *high memory bandwidth* to computation units in a practical way?
  - Processing-in-memory based on 3D-stacked DRAM
  
- **Challenge 2:** How to design computation units that *efficiently exploit large memory bandwidth*?
  - Specialized in-order cores called *Tesseract* cores
    - Latency-tolerant programming model
    - Graph-processing-specific prefetching schemes

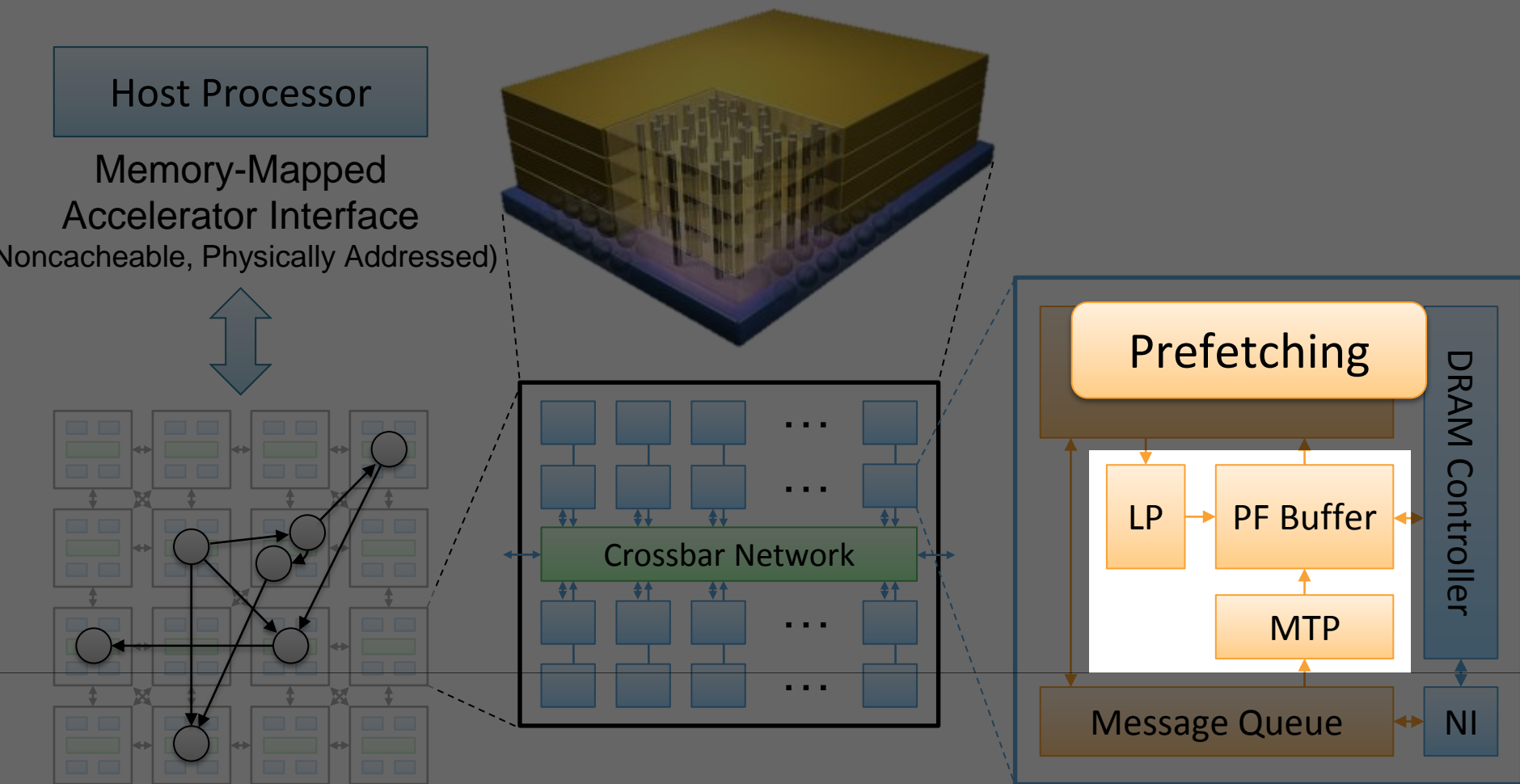
# Tesseract System for Graph Processing



# Tesseract System for Graph Processing

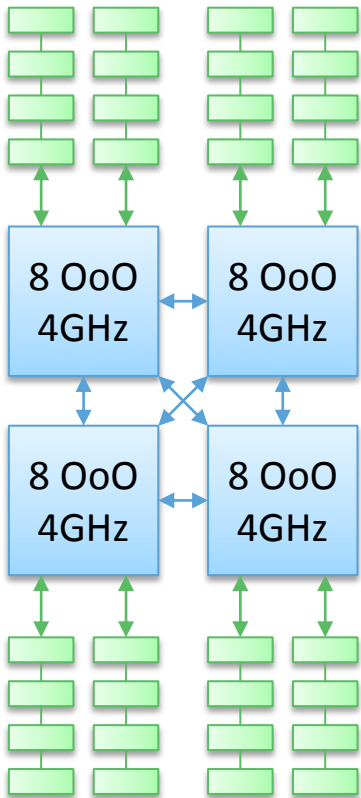


# Tesseract System for Graph Processing



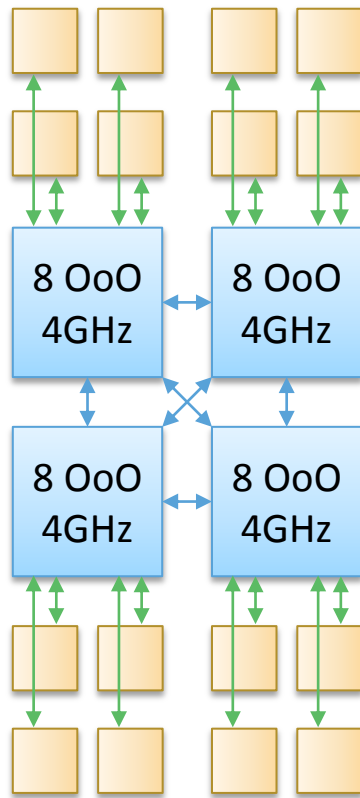
# Evaluated Systems

**DDR3-OoO**  
(with FDP)



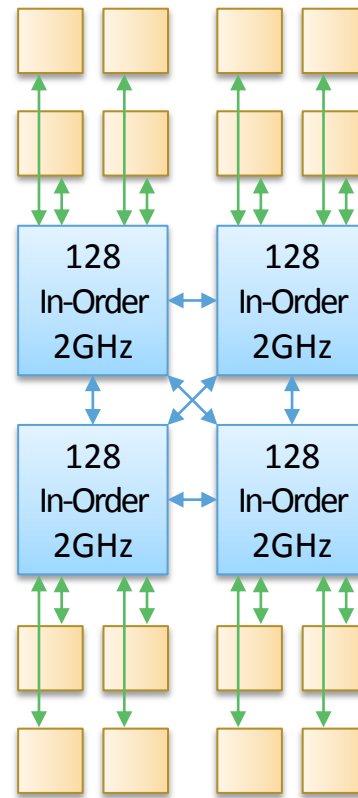
102.4GB/s

**HMC-OoO**  
(with FDP)



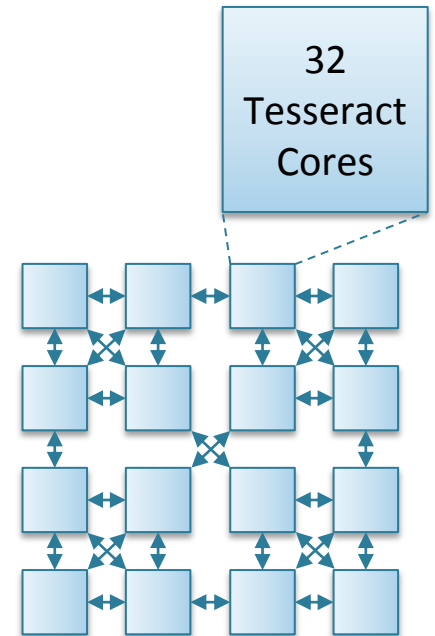
640GB/s

**HMC-MC**



640GB/s

**Tesseract**  
(32-entry MQ, 4KB PF Buffer)



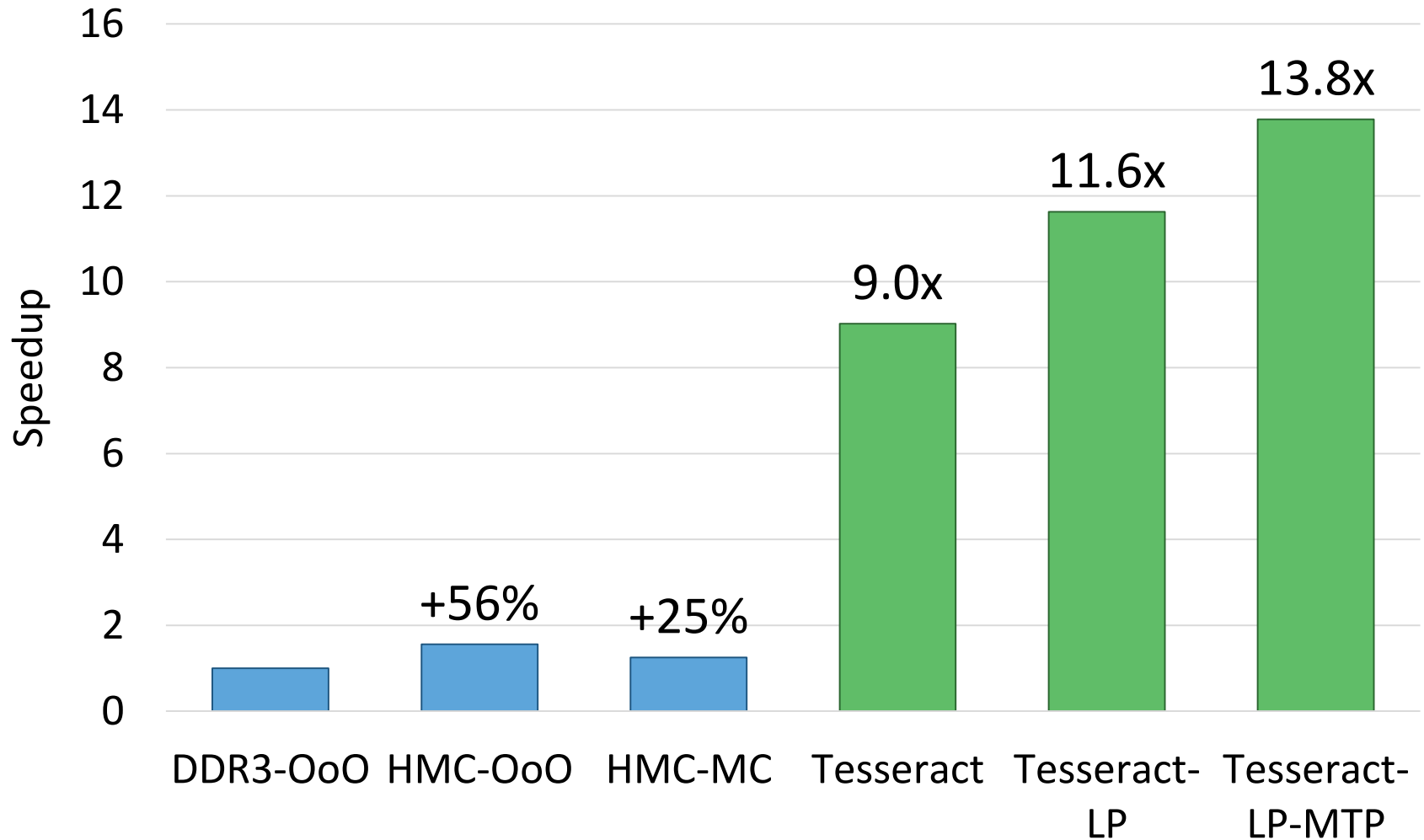
8TB/s

# Workloads

---

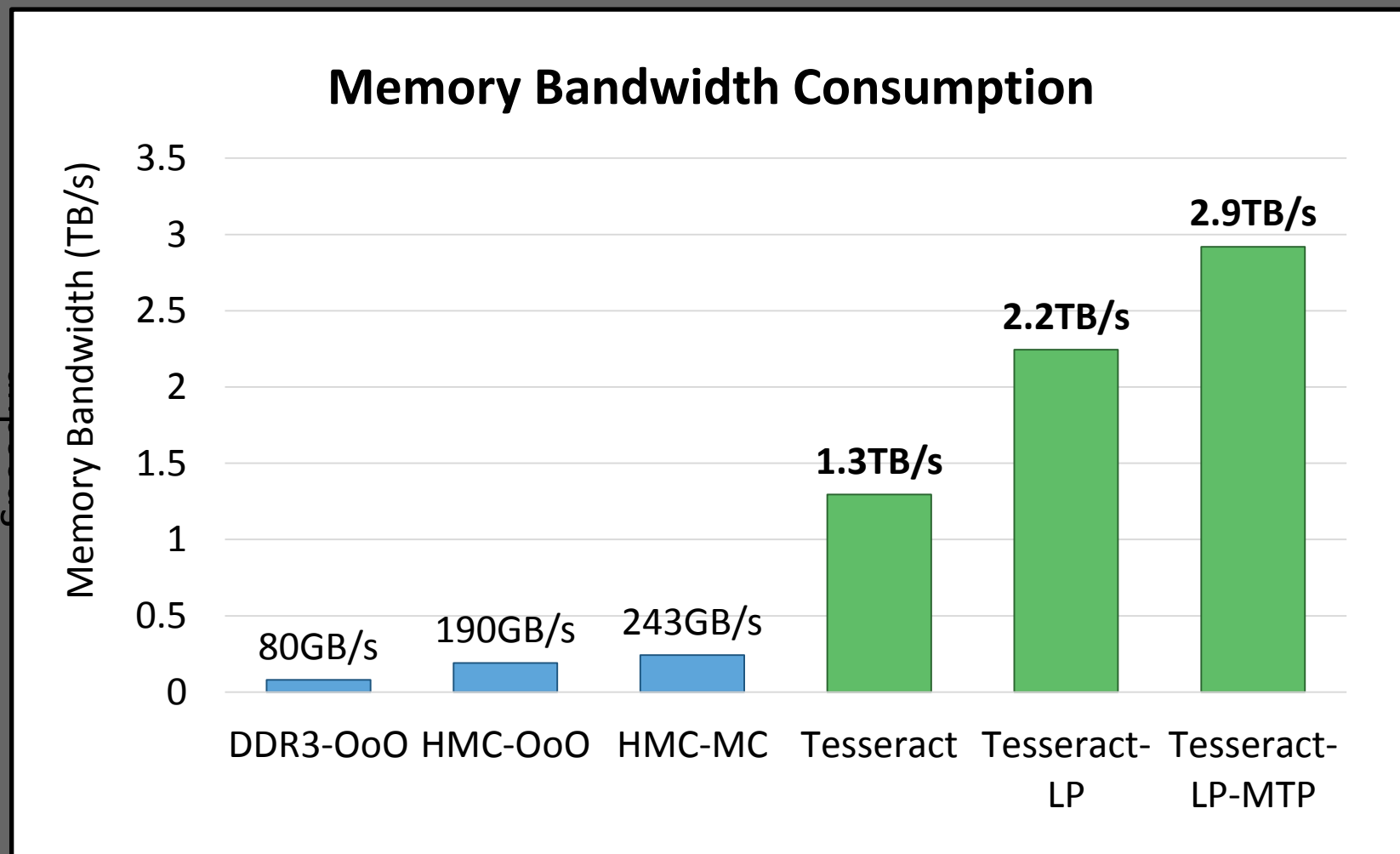
- Five graph processing algorithms
  - Average teenage follower
  - Conductance
  - PageRank
  - Single-source shortest path
  - Vertex cover
  
- Three real-world large graphs
  - ljournal-2008 (social network)
  - enwiki-2003 (Wikipedia)
  - indochina-0024 (web graph)
  - 4~7M vertices, 79~194M edges

# Tesseract Graph Processing Performance

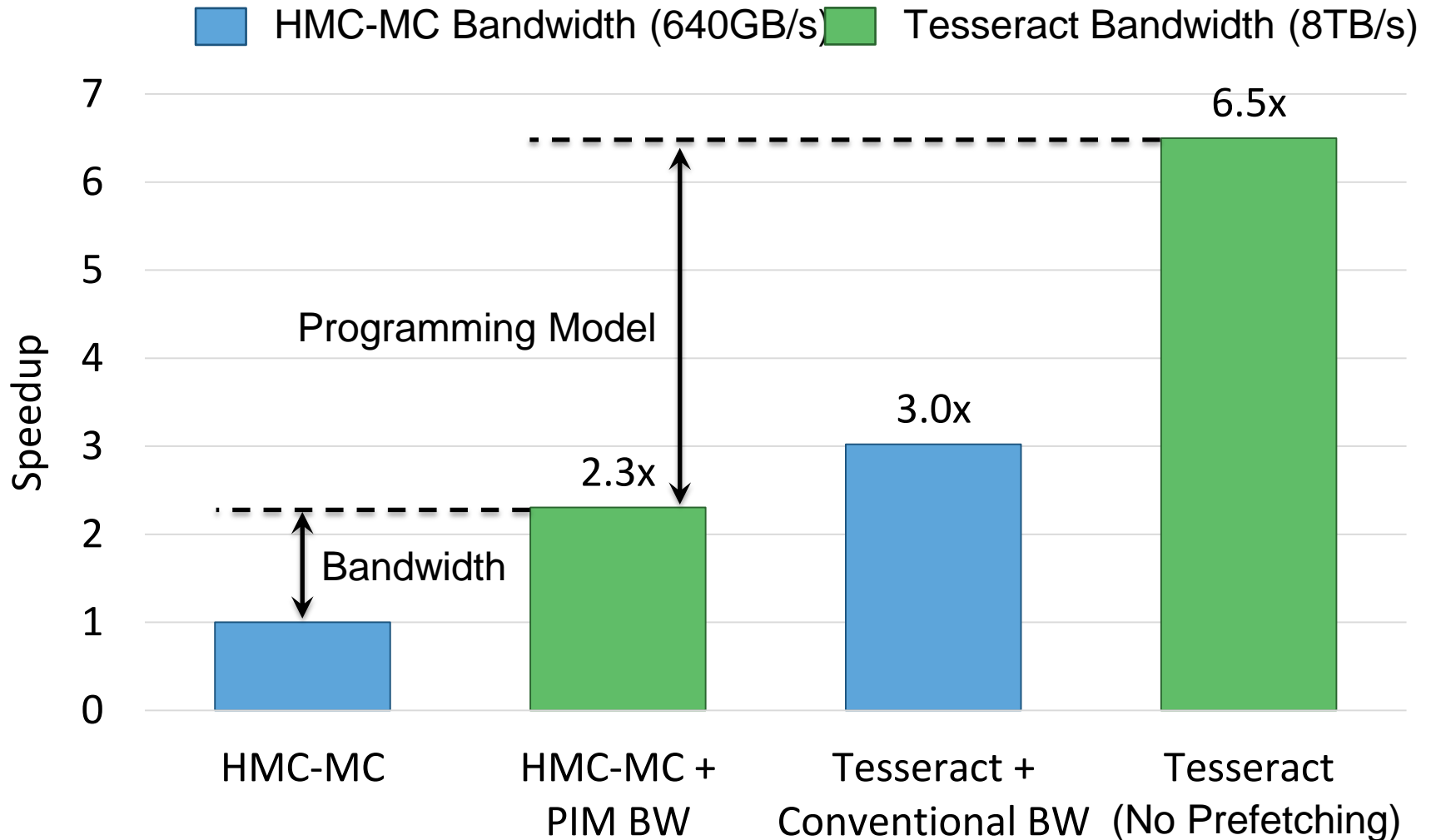




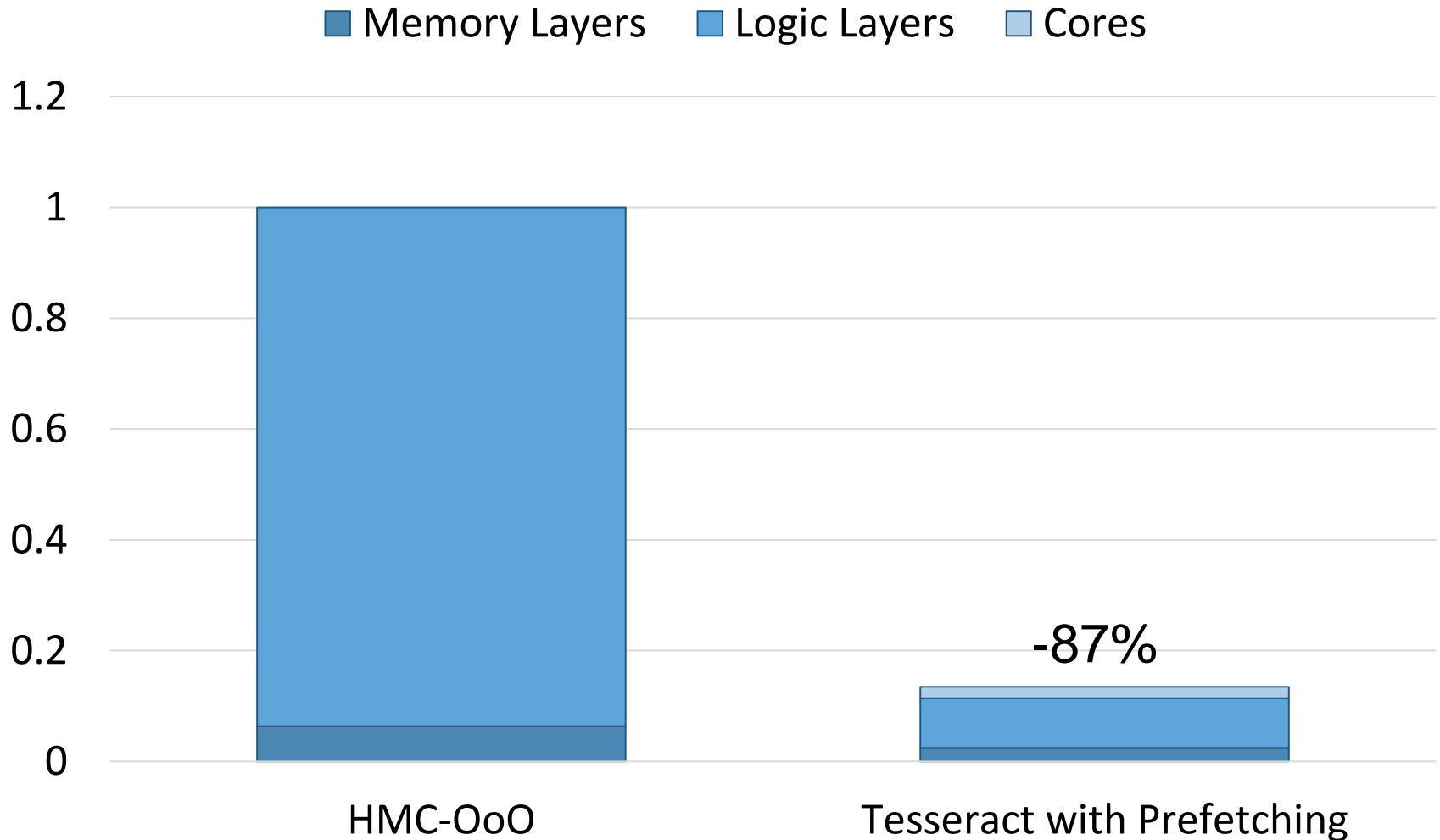
# Tesseract Graph Processing Performance



# Effect of Bandwidth & Programming Model



# Memory Energy Consumption (Normalized)



# Tesseract Summary

---

- How can we accelerate large-scale graph processing using 3D-stacked memory as a coarse-grained accelerator?
- Tesseract: 3D-stacked PIM accelerator for graph processing
  - Many in-order cores in a memory chip
  - New message passing mechanism for latency hiding
  - New hardware prefetchers for graph processing
  - Programming interface that exploits our hardware design
- Promising results on five graph processing workloads
  - ~14x performance improvement & 87% energy reduction
  - Scalable: *memory-capacity-proportional* performance

# Two Approaches to In-Memory Processing

---

- 1. **Minimally change DRAM** to enable simple yet powerful computation primitives
  - RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data (Seshadri et al., MICRO 2013)
  - Fast Bulk Bitwise AND and OR in DRAM (Seshadri et al., IEEE CAL 2015)
  
- 2. **Exploit the control logic in 3D-stacked memory** to enable more comprehensive computation near memory
  - PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture (Ahn et al., ISCA 2015)
  - A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing (Ahn et al., ISCA 2015)

# In-Memory Computation: Summary

---

- It is time to enable mechanisms for performing computation where it makes sense
  - Push from memory technology
  - Pull from systems and applications
- Multiple approaches for in-memory computation can be successful
  - Minimally changing DRAM to enable a bulk computation model
  - Exploiting the control logic in 3D-stacked memory
- Approaches require cross-layer cooperation and research
  - Architecture, systems, compilers, programming models, algorithms, ...

# Rethinking DRAM

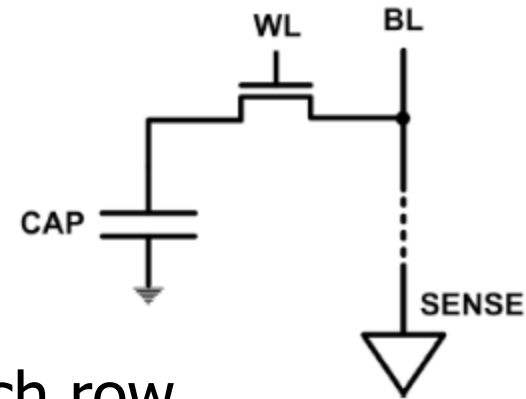
---

- In-Memory Computation
  - Refresh
  - Reliability
- Latency
- Bandwidth
- Energy
- Memory Compression

# DRAM Refresh

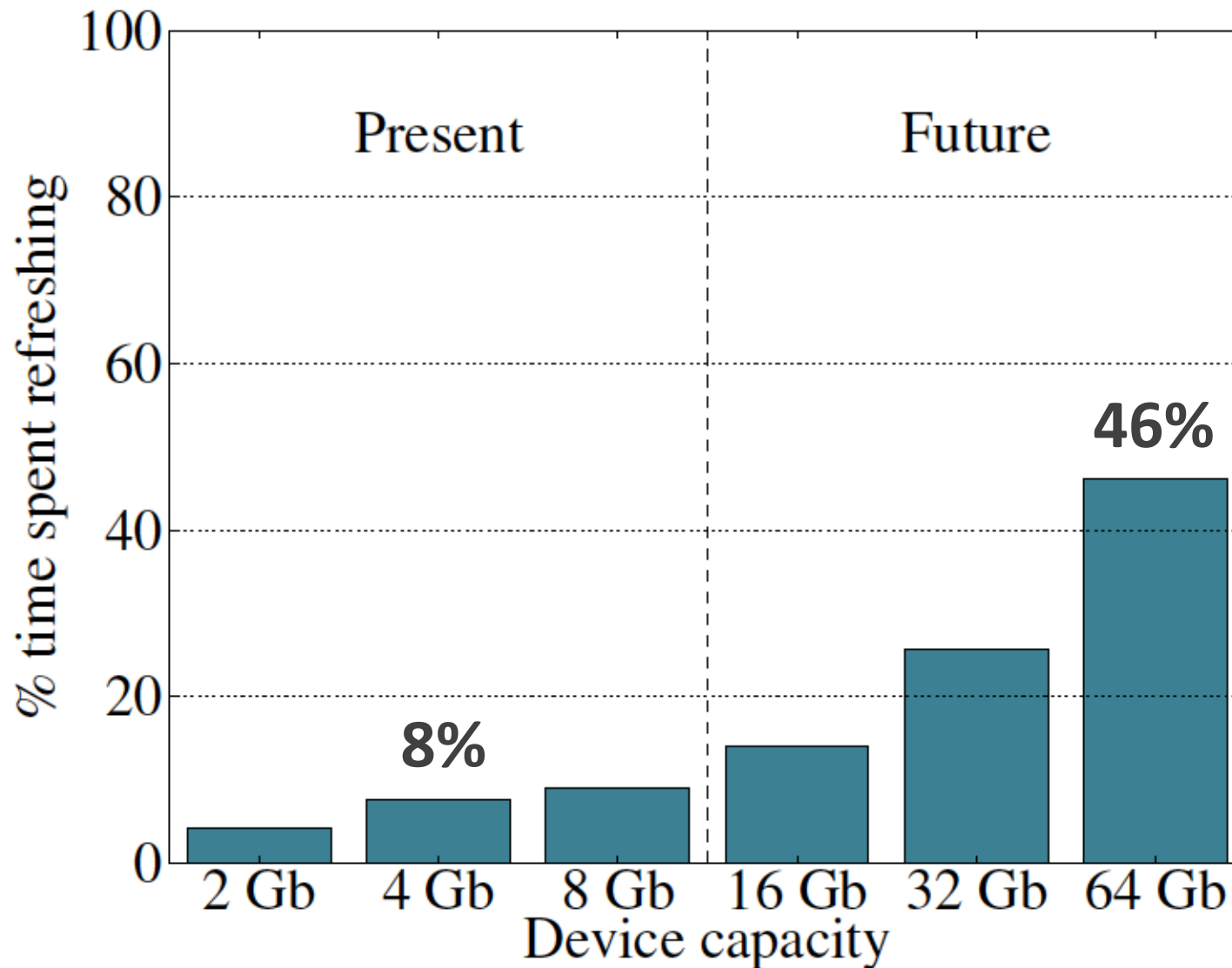
---

- DRAM capacitor charge leaks over time
- The memory controller needs to refresh each row periodically to restore charge
  - Activate each row every N ms
  - Typical N = 64 ms
- Downsides of refresh
  - **Energy consumption**: Each refresh consumes energy
  - **Performance degradation**: DRAM rank/bank unavailable while refreshed
  - **QoS/predictability impact**: (Long) pause times during refresh
  - **Refresh rate limits DRAM capacity scaling**

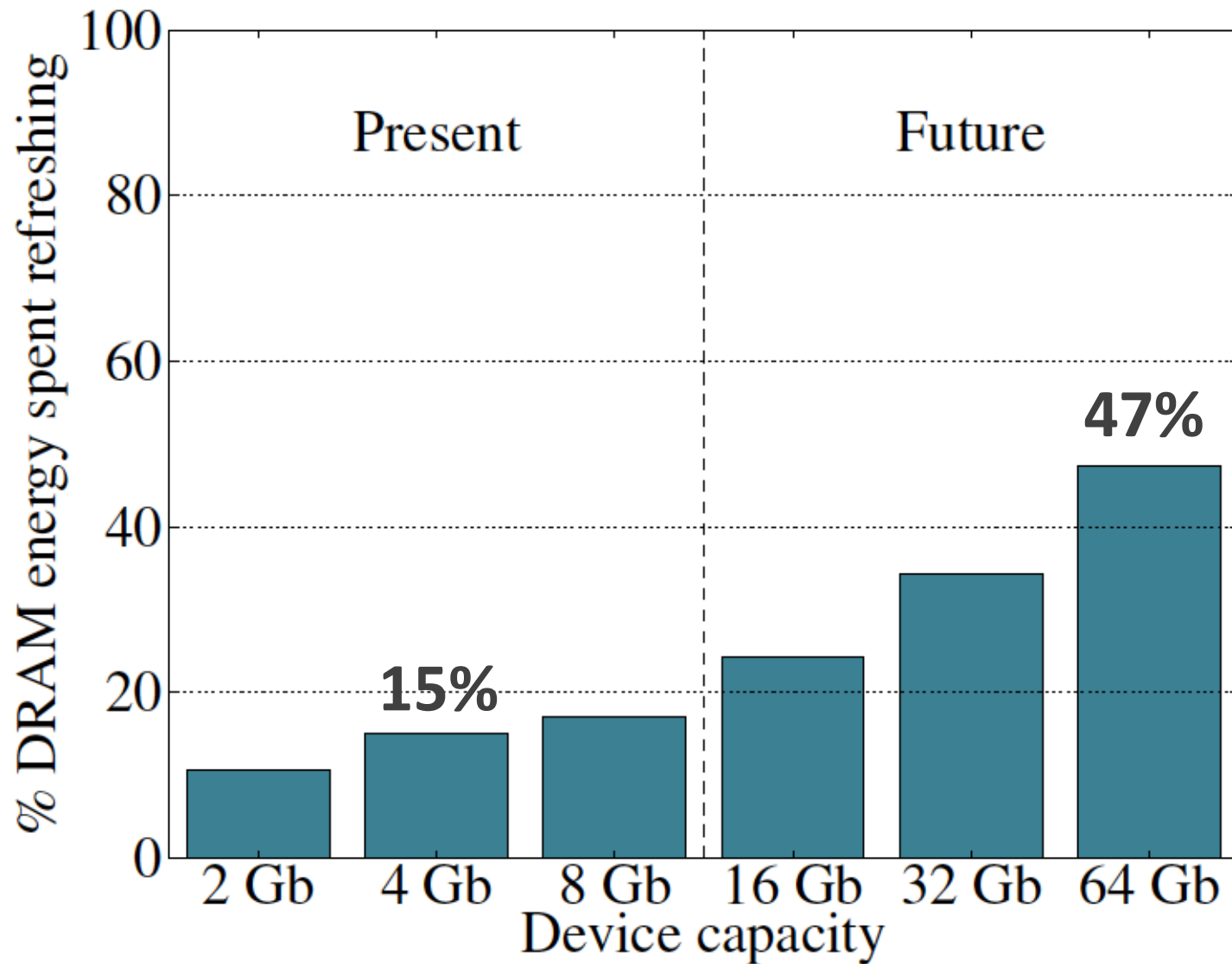




# Refresh Overhead: Performance

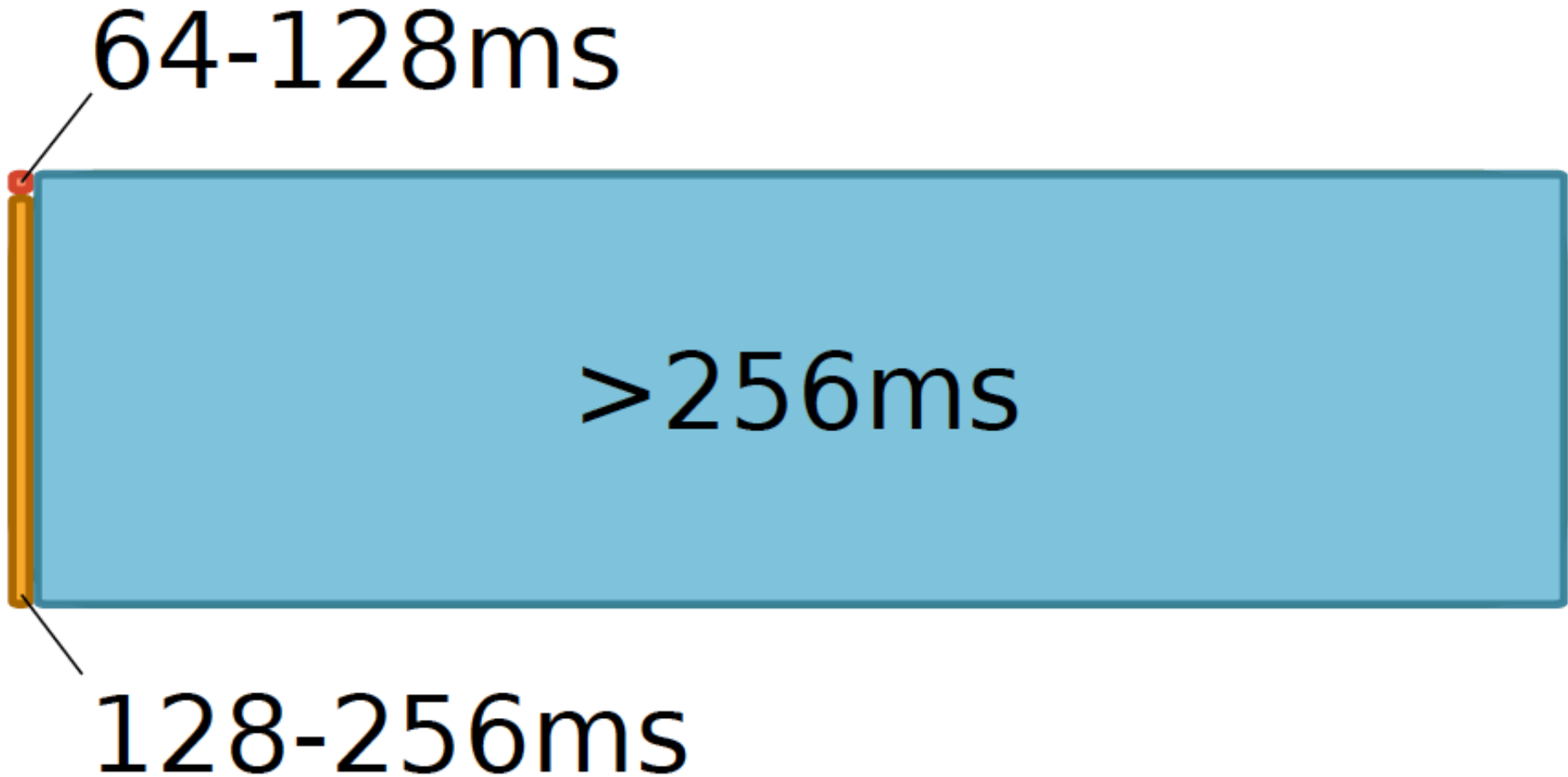


# Refresh Overhead: Energy



# Retention Time Profile of DRAM

---



# RAIDR: Eliminating Unnecessary Refreshes

■ Observation: Most DRAM rows can be refreshed much less often without losing data [Kim+, EDL'09][Liu+ ISCA'13]

■ Key idea: Refresh rows containing weak cells more frequently, other rows less frequently

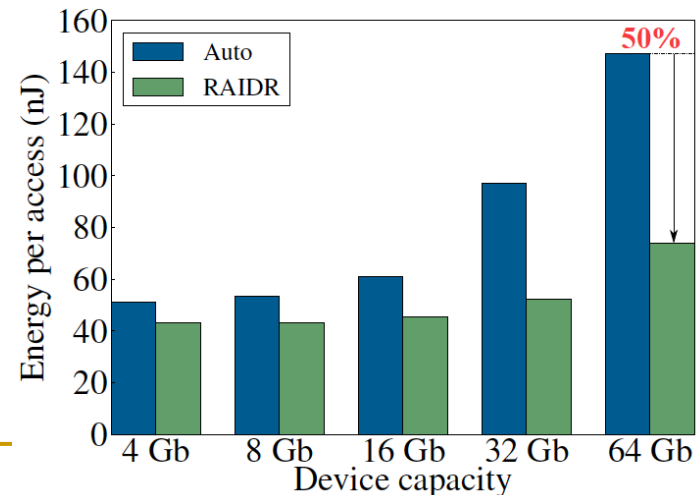
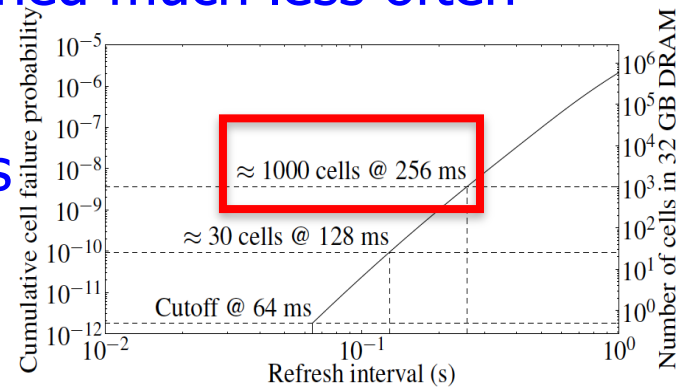
1. Profiling: Profile retention time of all rows

2. Binning: Store rows into bins by retention time in memory controller  
*Efficient storage with Bloom Filters (only 1.25KB for 32GB memory)*

3. Refreshing: Memory controller refreshes rows in different bins at different rates

■ Results: 8-core, 32GB, SPEC, TPC-C, TPC-H

- 74.6% refresh reduction @ 1.25KB storage
- ~16%/20% DRAM dynamic/idle power reduction
- ~9% performance improvement
- Benefits increase with DRAM capacity



# Going Forward (for DRAM and Flash)

---

- **How to find out weak memory cells/rows**
  - Liu+, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms", ISCA 2013.
  - Khan+, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," SIGMETRICS 2014.
- **Low-cost system-level tolerance of memory errors**
  - Luo+, "Characterizing Application Memory Error Vulnerability to Optimize Data Center Cost," DSN 2014.
  - Cai+, "Error Analysis and Retention-Aware Error Management for NAND Flash Memory," Intel Technology Journal 2013.
  - Cai+, "Neighbor-Cell Assisted Error Correction for MLC NAND Flash Memories," SIGMETRICS 2014.
- **Tolerating cell-to-cell interference at the system level**
  - Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.
  - Cai+, "Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation," ICCD 2013.

# Experimental DRAM Testing Infrastructure



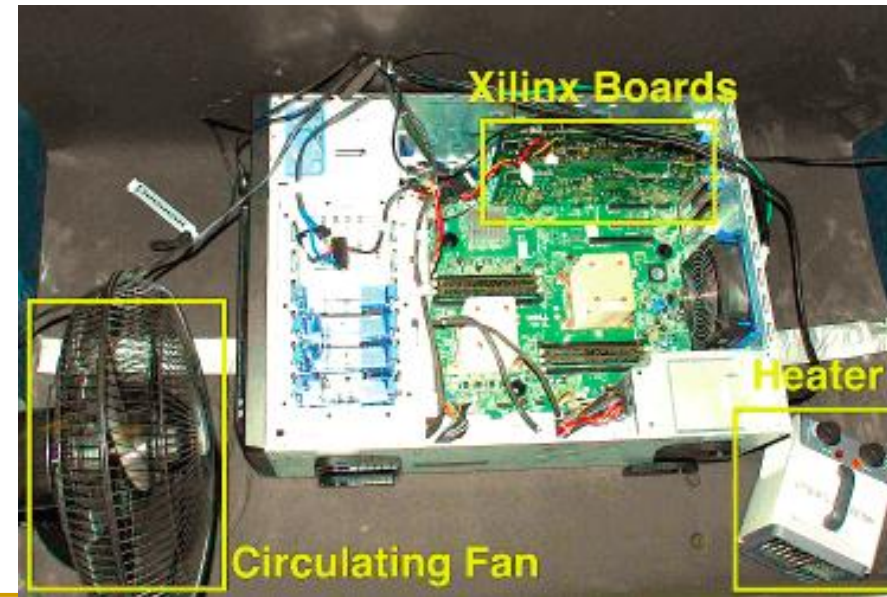
An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study (Khan et al., SIGMETRICS 2014)

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

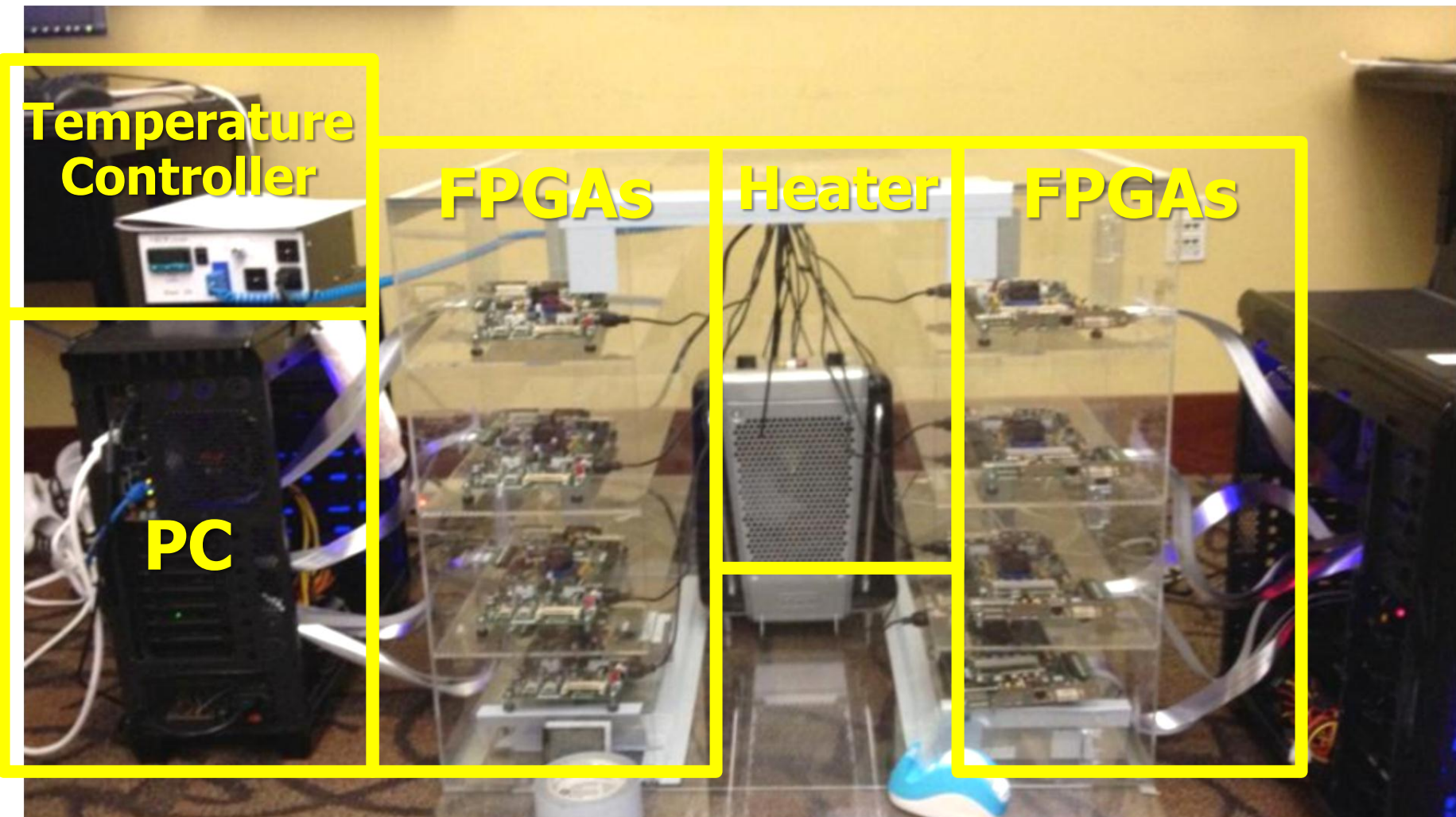
Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case (Lee et al., HPCA 2015)

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems (Qureshi et al., DSN 2015)





# Experimental Infrastructure (DRAM)



# More Information [ISCA'13, SIGMETRICS'14]

---

## The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study

Samira Khan<sup>†\*</sup>  
samirakhan@cmu.edu

Donghyuk Lee<sup>†</sup>  
donghyuk1@cmu.edu

Yoongu Kim<sup>†</sup>  
yoongukim@cmu.edu

Alaa R. Alameldeen<sup>\*</sup>  
alaa.r.alameldeen@intel.com

Chris Wilkerson<sup>\*</sup>  
chris.wilkerson@intel.com

Onur Mutlu<sup>†</sup>  
onur@cmu.edu

<sup>†</sup>Carnegie Mellon University

<sup>\*</sup>Intel Labs



# Online Profiling of DRAM In the Field

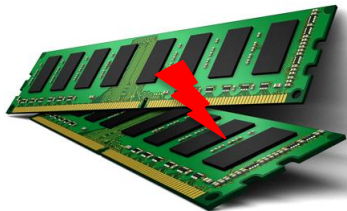
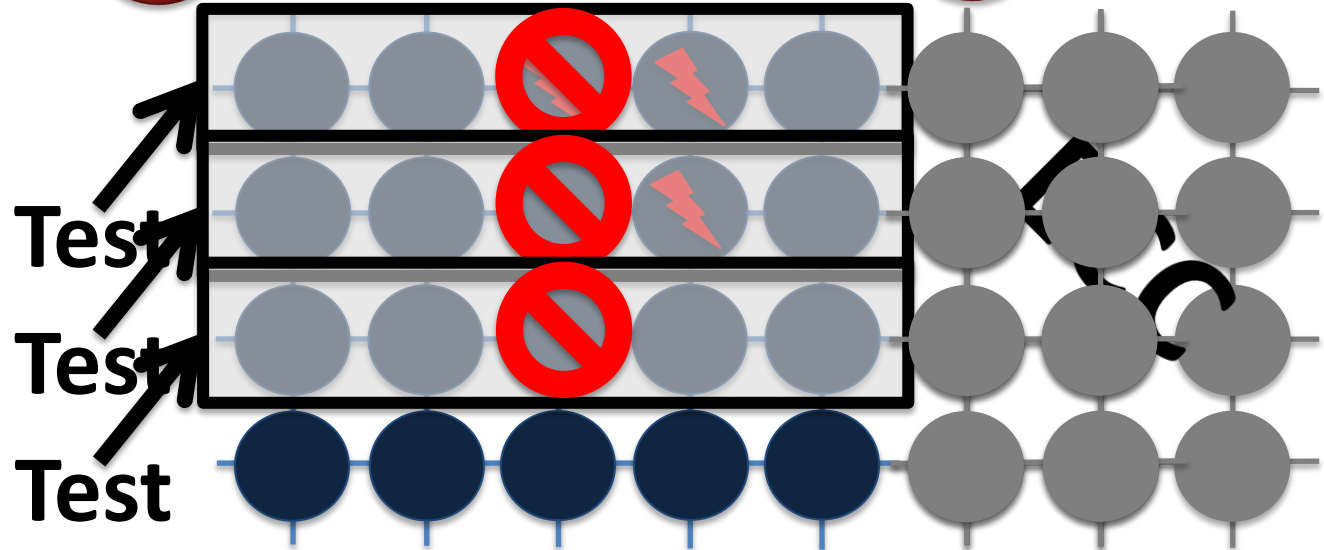
Initially protect DRAM  
with ECC

1



Periodically test  
parts of DRAM

2



Adjust refresh rate and  
reduce ECC

3

**Optimize DRAM and mitigate errors online**

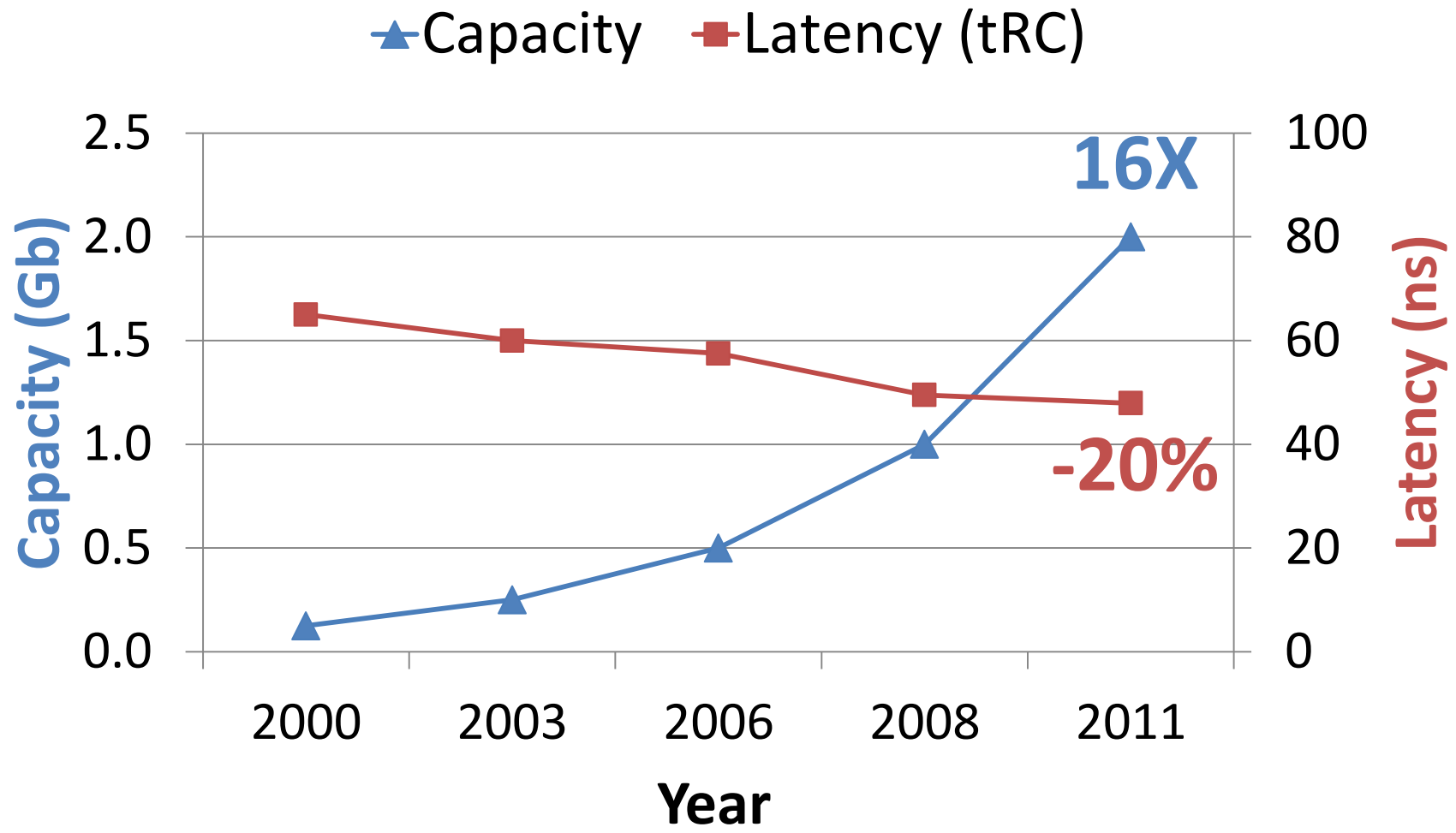
**without disturbing the system and applications**

# Rethinking DRAM

---

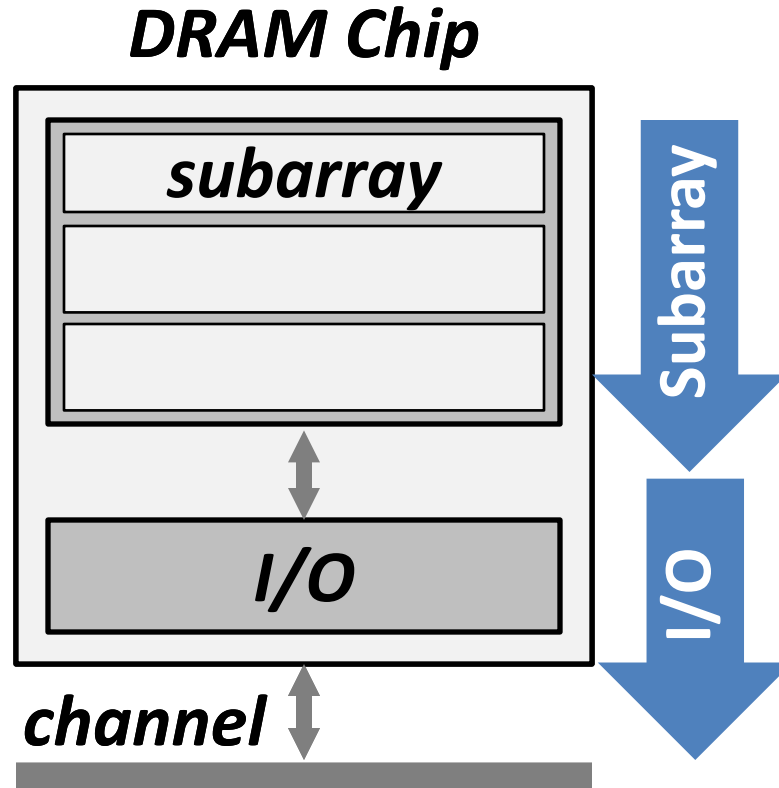
- In-Memory Computation
- Refresh
- Reliability
- Latency
- Bandwidth
- Energy
- Memory Compression

# DRAM Latency-Capacity Trend



*DRAM latency continues to be a critical bottleneck, especially for response time-sensitive* <sup>43</sup>

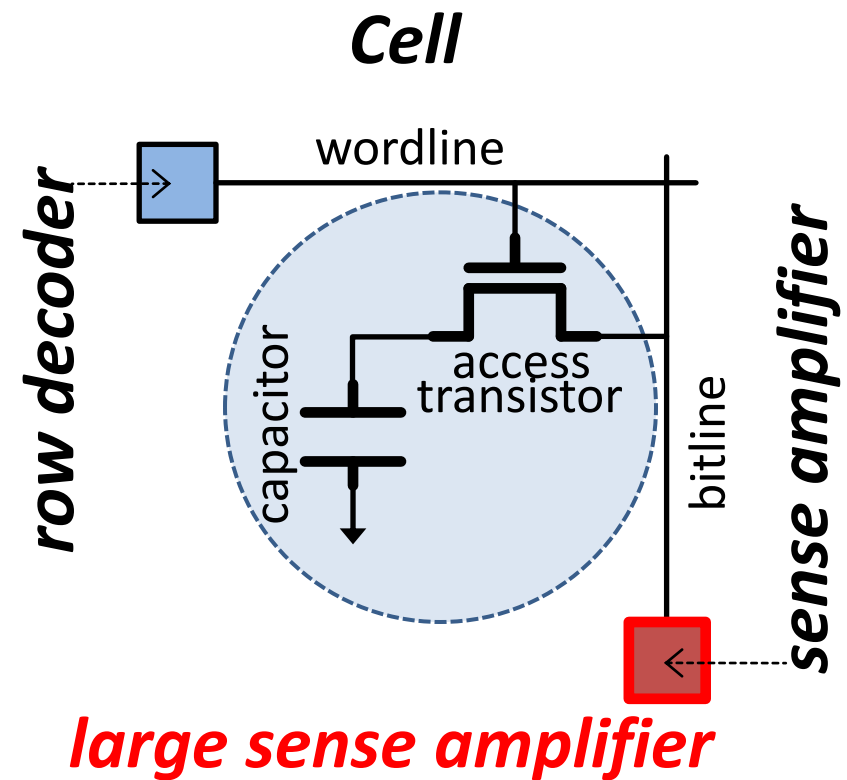
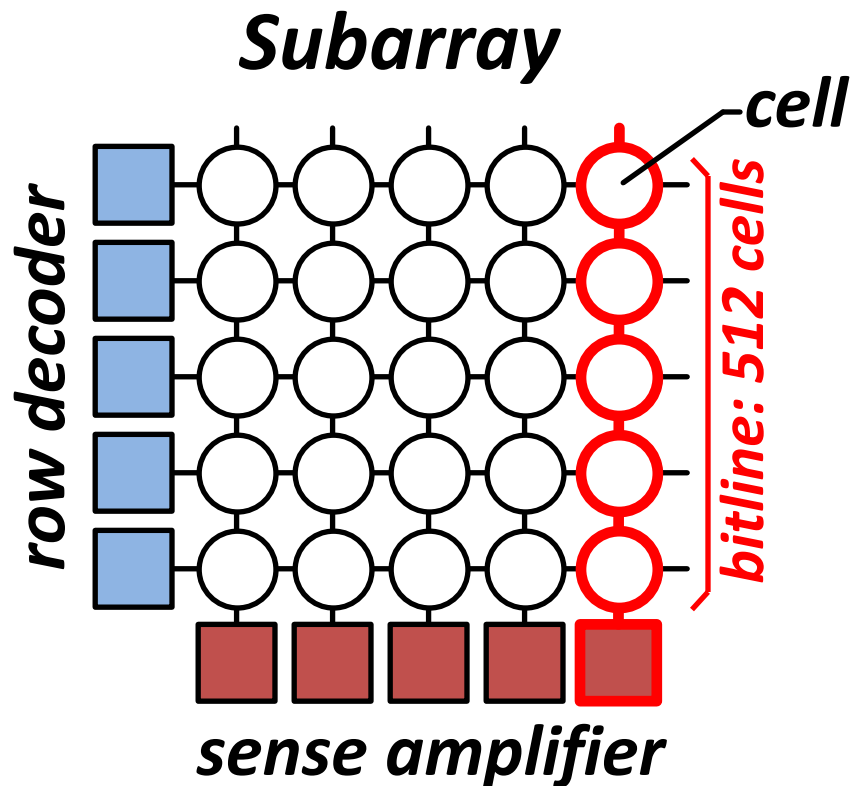
# What Causes the Long Latency?



*DRAM Latency = **Subarray Latency** + I/O Latency*

**Dominant**

# Why is the Subarray So Slow?

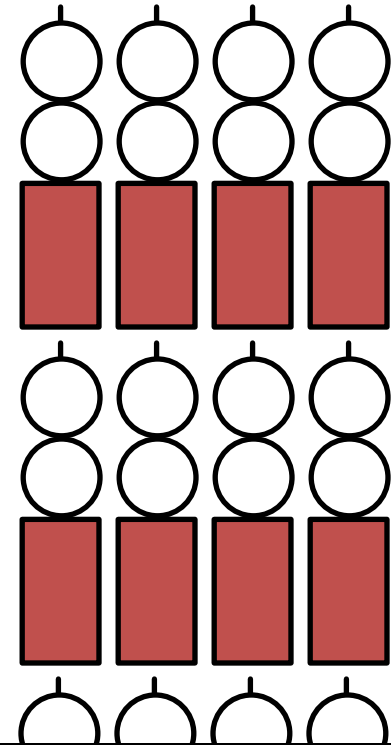
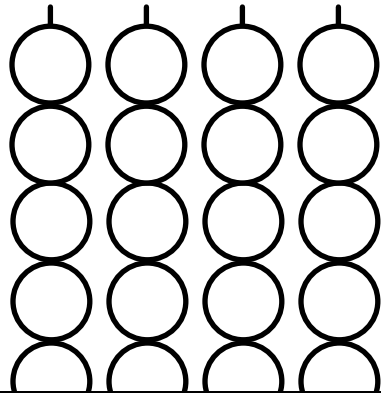


- Long bitline
  - Amortizes sense amplifier cost → Small area
  - Large bitline capacitance → High latency & power

# Trade-Off: Area (Die Size) vs. Latency

Long Bitline

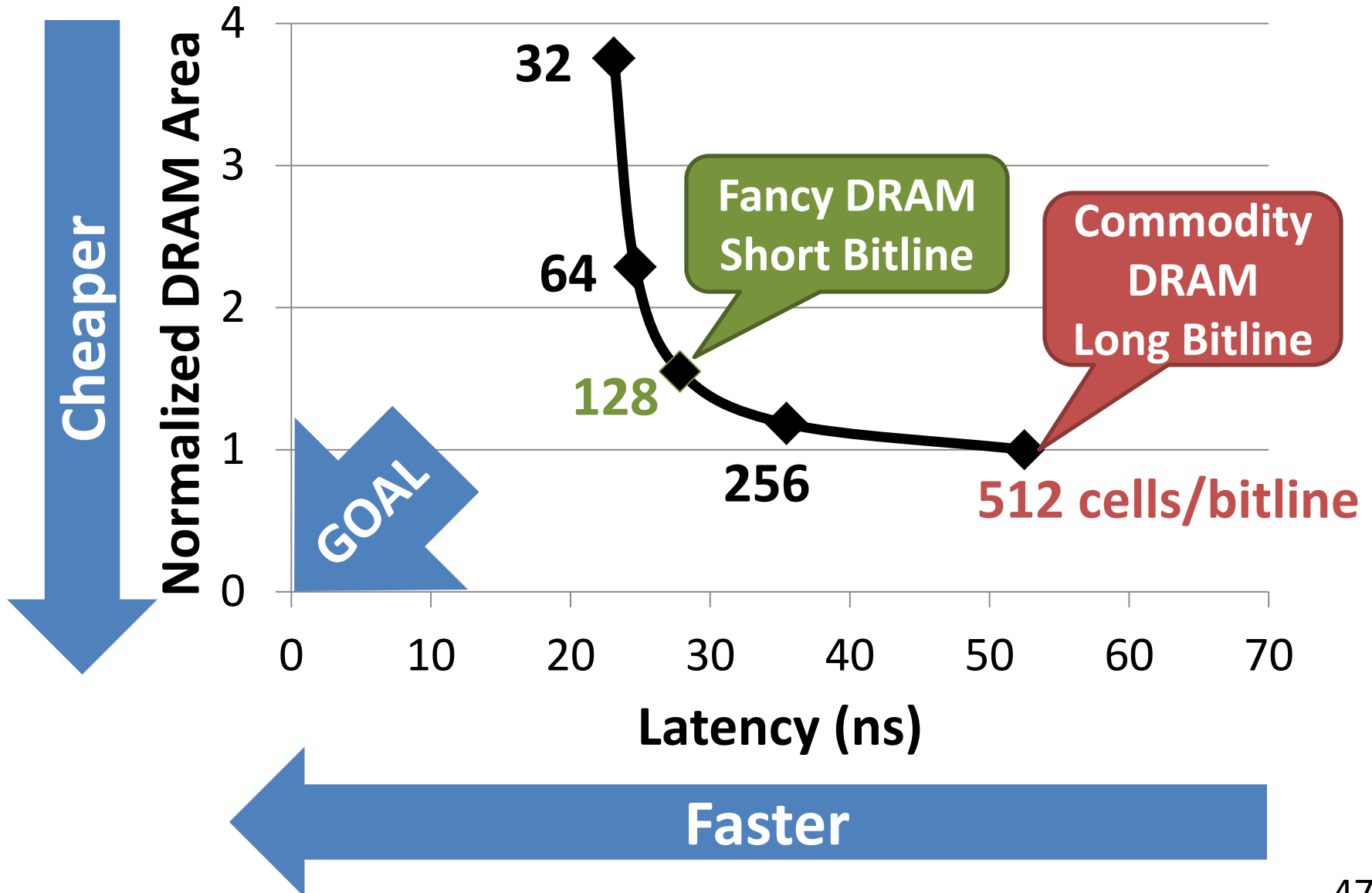
Short Bitline



**Trade-Off: Area vs. Latency**



# Trade-Off: Area (Die Size) vs. Latency

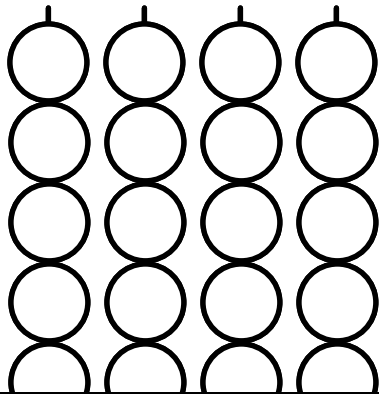


# Approximating the Best of Both Worlds

**Long Bitline**

*Small Area*

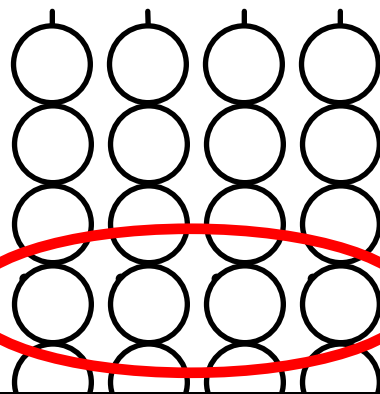
~~High Latency~~



*Need Isolation*

**Our Proposal**

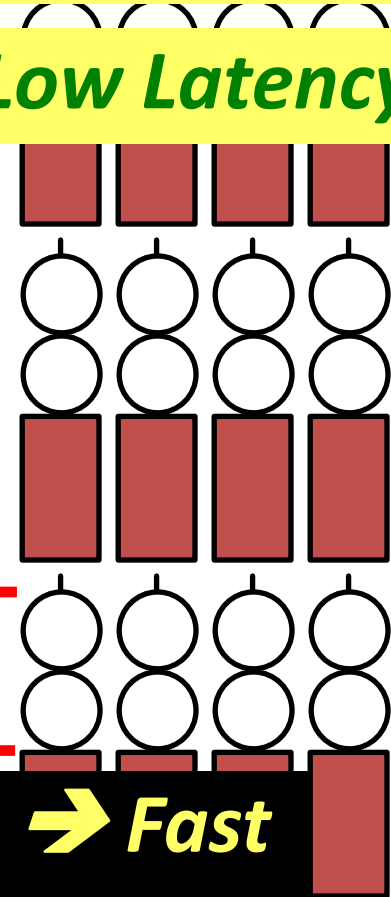
*Add Isolation Transistors*



**Short Bitline**

~~Large Area~~

*Low Latency*



*Fast*



# Approximating the Best of Both Worlds

**Long Bitline Tiered-Latency DRAM** | **Short Bitline**

*Small Area*

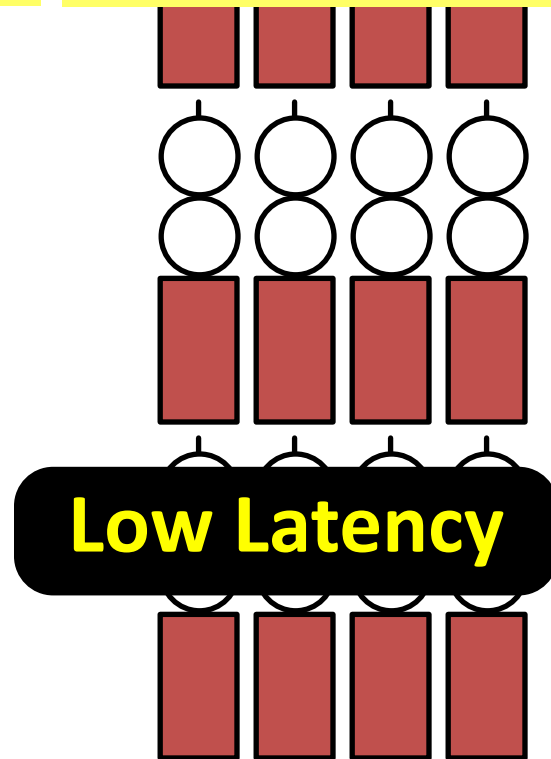
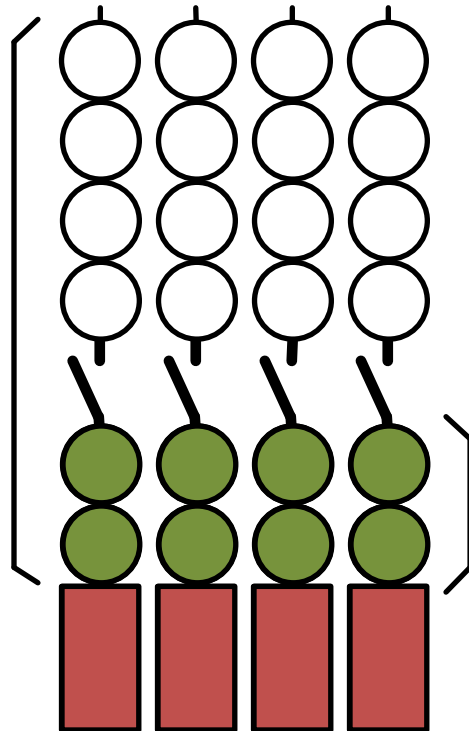
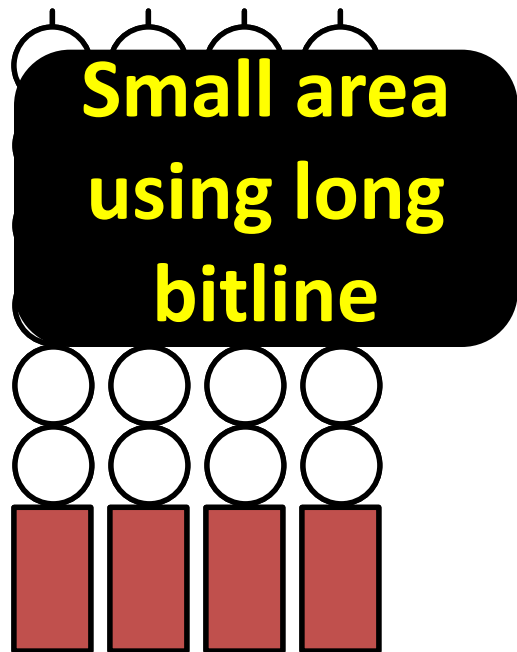
*Small Area*

~~*Large Area*~~

~~*High Latency*~~

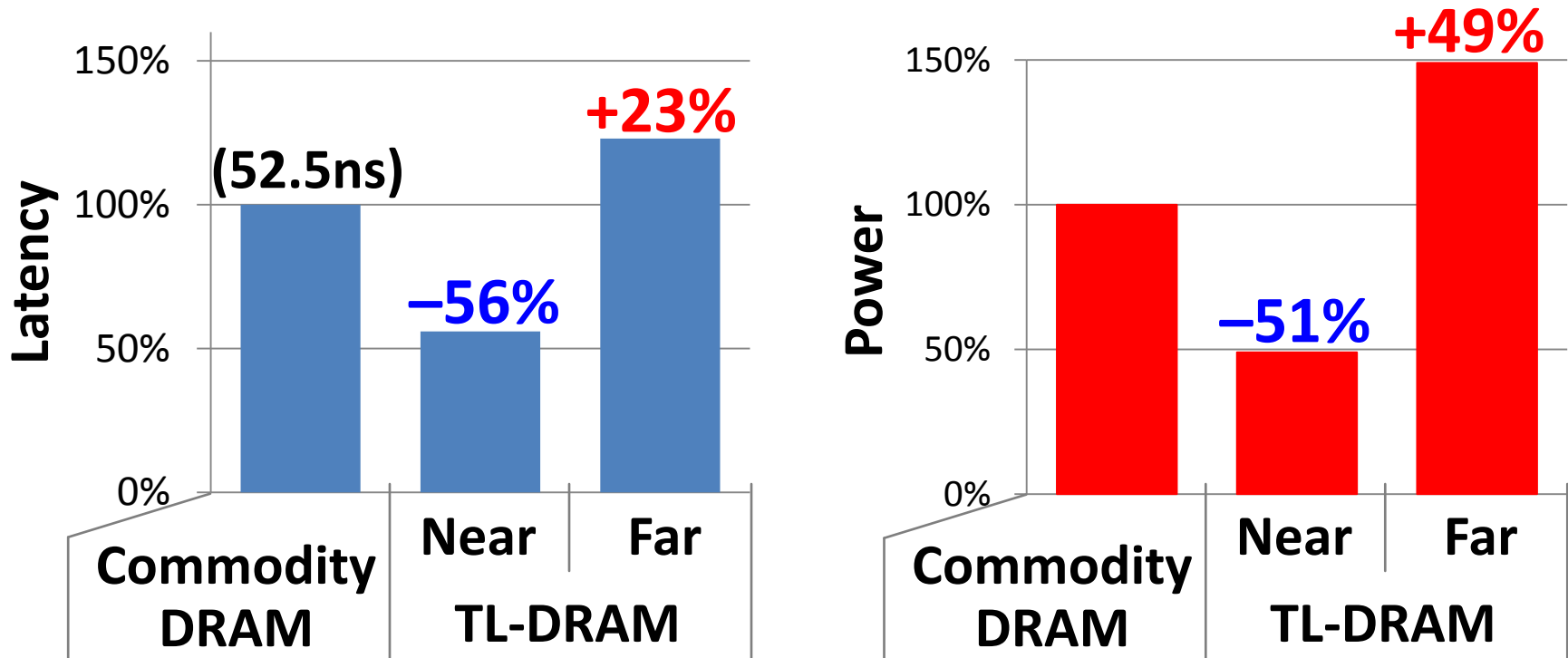
*Low Latency*

*Low Latency*



# Commodity DRAM vs. TL-DRAM [HPCA 2013]

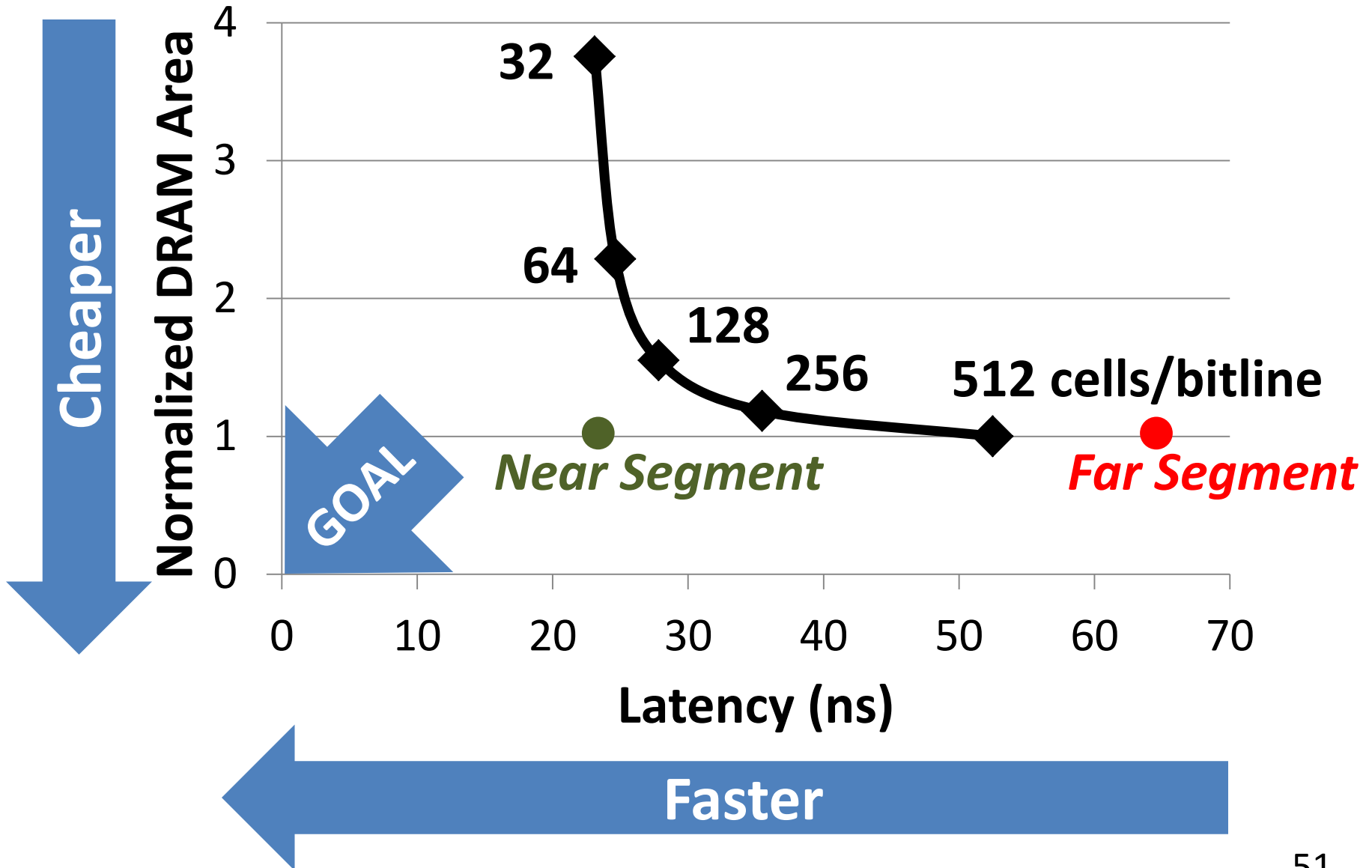
- DRAM Latency (tRC) • DRAM Power



- DRAM Area Overhead

~3%: mainly due to the isolation transistors

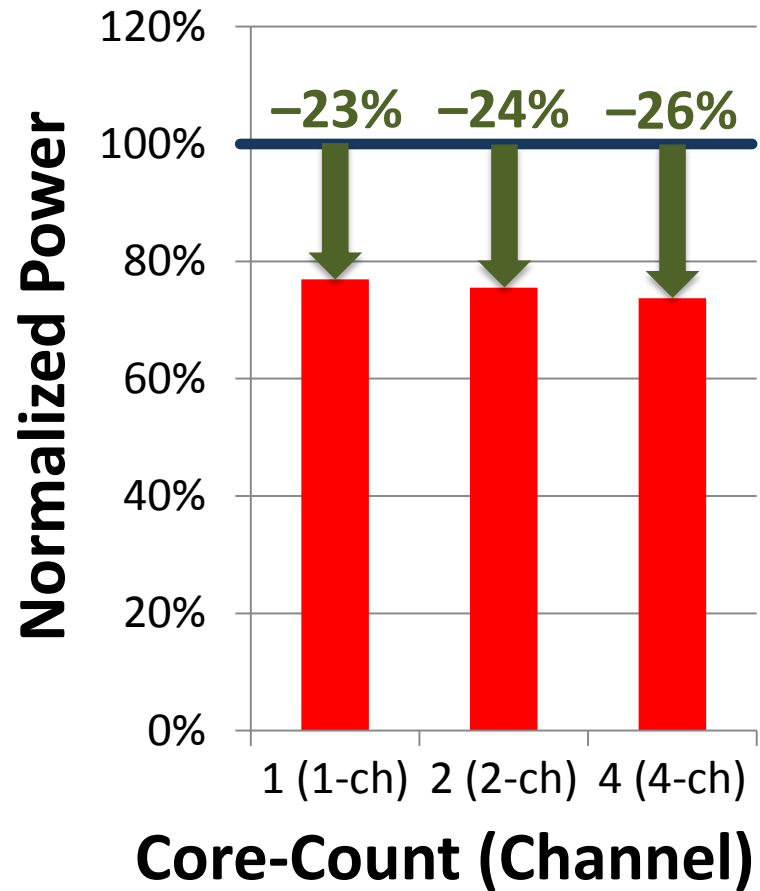
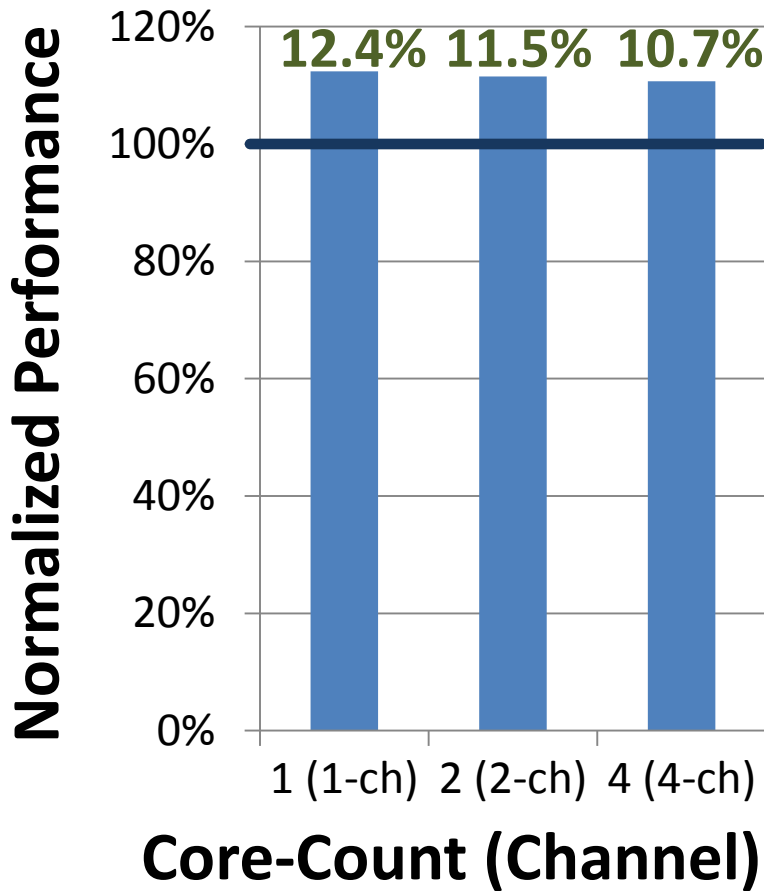
# Trade-Off: Area (Die-Area) vs. Latency



# Leveraging Tiered-Latency DRAM

- TL-DRAM is a *substrate* that can be leveraged by the hardware and/or software
- Many potential uses
  1. Use near segment as hardware-managed *inclusive* cache to far segment
  2. Use near segment as hardware-managed *exclusive* cache to far segment
  3. Profile-based page mapping by operating system
  4. Simply replace DRAM with TL-DRAM

# Performance & Power Consumption



*Using near segment as a cache improves performance and reduces power consumption*

# What Else Causes the Long DRAM Latency?

---

- **Conservative timing margins!**
- DRAM timing parameters are set to cover the worst case
- **Worst-case temperatures**
  - 85 degrees vs. common-case
  - to enable a wide range of operating conditions
- **Worst-case devices**
  - DRAM cell with smallest charge across any acceptable device
  - to tolerate process variation at acceptable yield
- This leads to large timing margins for the common case

# Adaptive-Latency DRAM [HPCA 2015]

---

- Idea: Optimize DRAM timing for the common case
  - Current temperature
  - Current DRAM module
- Why would this reduce latency?
  - A DRAM cell can store much more charge in the common case (low temperature, strong cell) than in the worst case
  - More charge in a DRAM cell
    - Faster sensing, charge restoration, precharging
    - Faster access (read, write, refresh, ...)

# AL-DRAM

- *Key idea*
  - Optimize DRAM timing parameters online
- *Two components*
  - DRAM manufacturer provides multiple sets of **reliable DRAM timing parameters** at different temperatures for each DIMM
  - System monitors **DRAM temperature** & uses appropriate DRAM timing parameters



# Latency Reduction Summary of 115 DIMMs

- *Latency reduction for read & write (55°C)*
  - *Read Latency: 32.7%*
  - *Write Latency: 55.1%*
- *Latency reduction for each timing parameter (55°C)*
  - *Sensing: 17.3%*
  - *Restore: 37.3% (read), 54.8% (write)*
  - *Precharge: 35.2%*

# AL-DRAM: Real System Evaluation

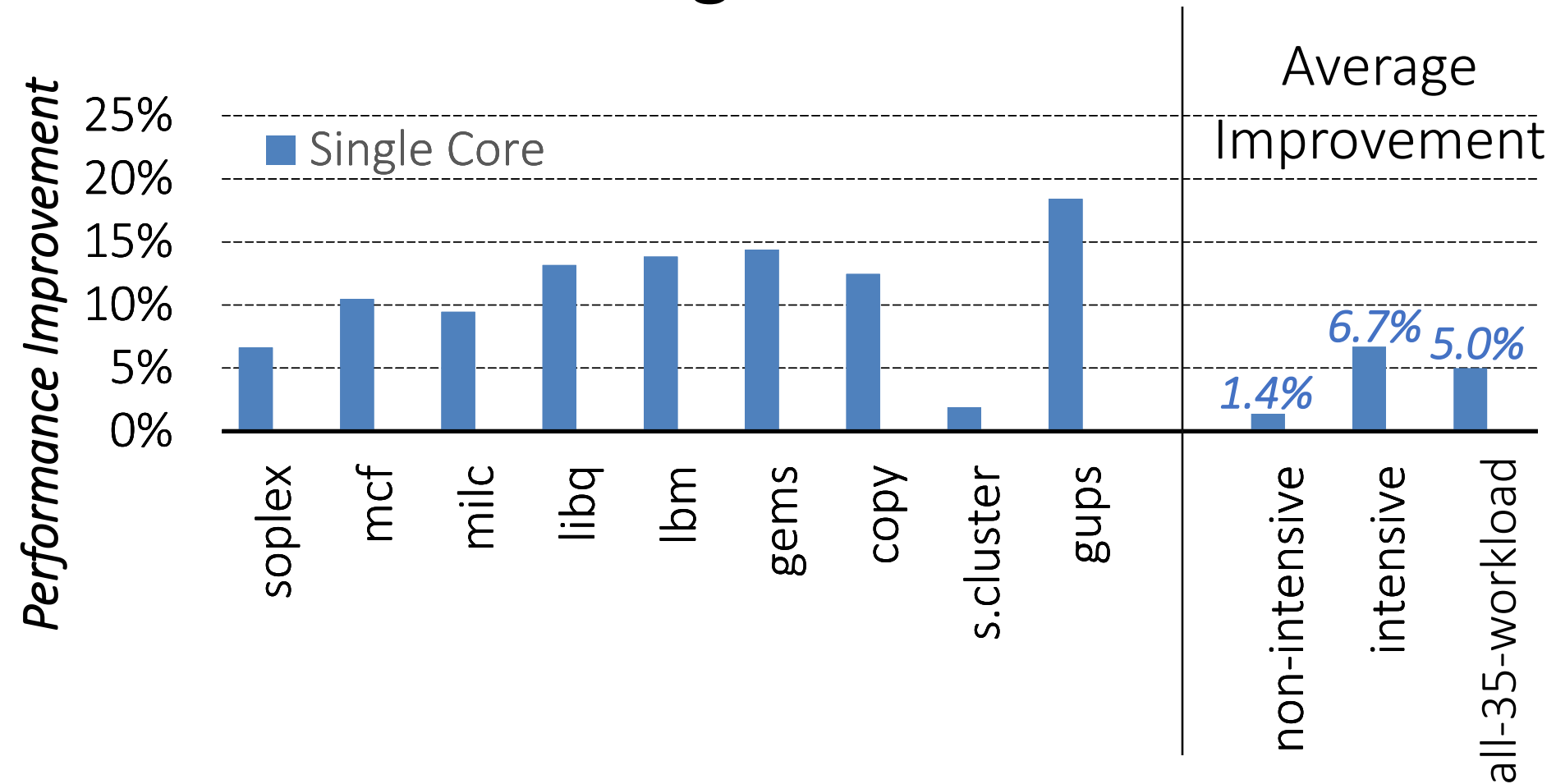
- *System*
  - CPU: AMD 4386 ( 8 Cores, 3.1GHz, 8MB LLC)

## D18F2x200\_dct[0]\_mp[1:0] DDR3 DRAM Timing 0

Reset: 0F05\_0505h. See 2.9.3 [DCT Configuration Registers].

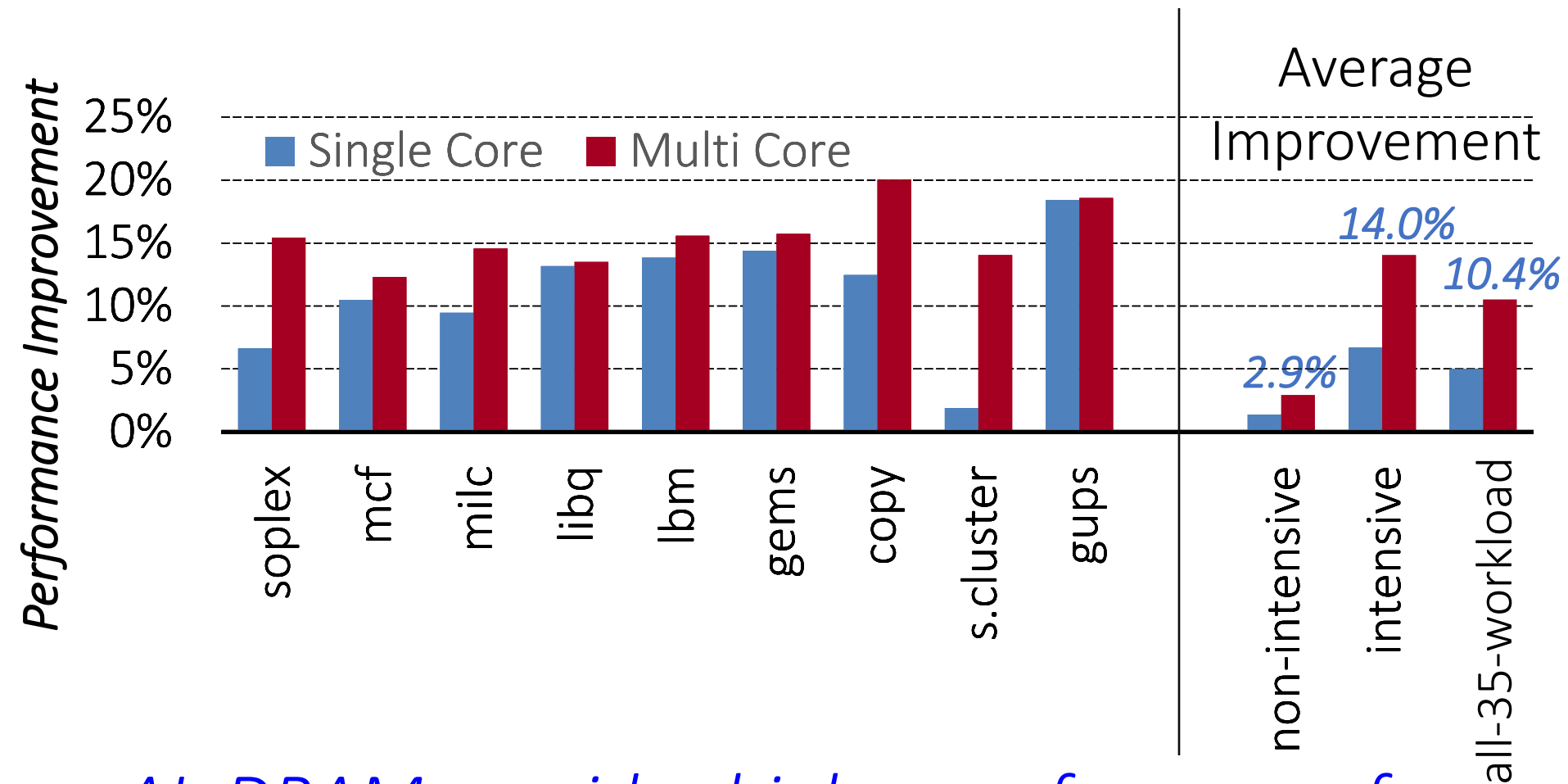
Bits	Description								
31:30	Reserved.								
29:24	<b>Tras: row active strobe.</b> Read-write. BIOS: See 2.9.7.5 [SPD ROM-Based Configuration]. Specifies the minimum time in memory clock cycles from an activate command to a precharge command, both to the same chip select bank. <table border="1"><thead><tr><th>Bits</th><th>Description</th></tr></thead><tbody><tr><td>07h-00h</td><td>Reserved</td></tr><tr><td>2Ah-08h</td><td>&lt;Tras&gt; clocks</td></tr><tr><td>3Fh-2Bh</td><td>Reserved</td></tr></tbody></table>	Bits	Description	07h-00h	Reserved	2Ah-08h	<Tras> clocks	3Fh-2Bh	Reserved
Bits	Description								
07h-00h	Reserved								
2Ah-08h	<Tras> clocks								
3Fh-2Bh	Reserved								
23:21	Reserved.								
20:16	<b>Trp: row precharge time.</b> Read-write. BIOS: See 2.9.7.5 [SPD ROM-Based Configuration]. Specifies the minimum time in memory clock cycles from a precharge command to an activate command or auto refresh command, both to the same bank.								

# AL-DRAM: Single-Core Evaluation



*AL-DRAM improves performance on a real system*

# AL-DRAM: Multi-Core Evaluation



*AL-DRAM provides higher performance for multi-programmed & multi-threaded workloads*

# Rethinking DRAM

---

- In-Memory Computation
- Refresh
- Reliability
- Latency
- Bandwidth
- Energy
- Memory Compression

# Agenda

---

- Major Trends Affecting Main Memory
- The Memory Scaling Problem and Solution Directions
  - New Memory Architectures
  - Enabling Emerging Technologies
- How Can We Do Better?
- Summary

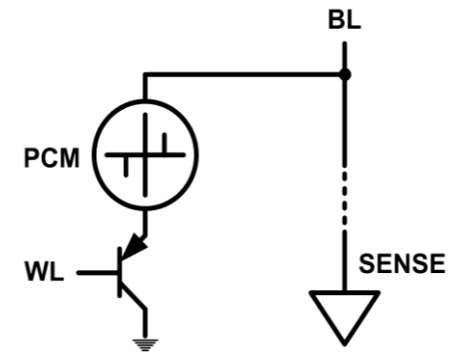
# Solution 2: Emerging Memory Technologies

---

- Some emerging resistive memory technologies seem more scalable than DRAM (and they are non-volatile)

- Example: Phase Change Memory

- Data stored by changing phase of material
- Data read by detecting material's resistance
- Expected to scale to 9nm (2022 [ITRS])
- Prototyped at 20nm (Raoux+, IBM JRD 2008)
- Expected to be denser than DRAM: can store multiple bits/cell



- But, emerging technologies have (many) shortcomings
  - Can they be enabled to replace/augment/surpass DRAM?

# Charge vs. Resistive Memories

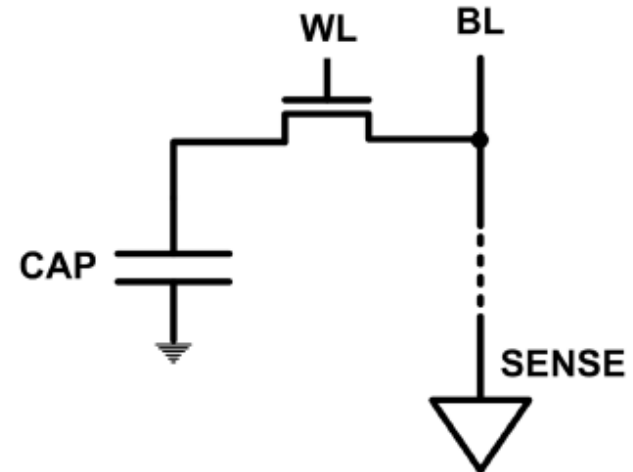
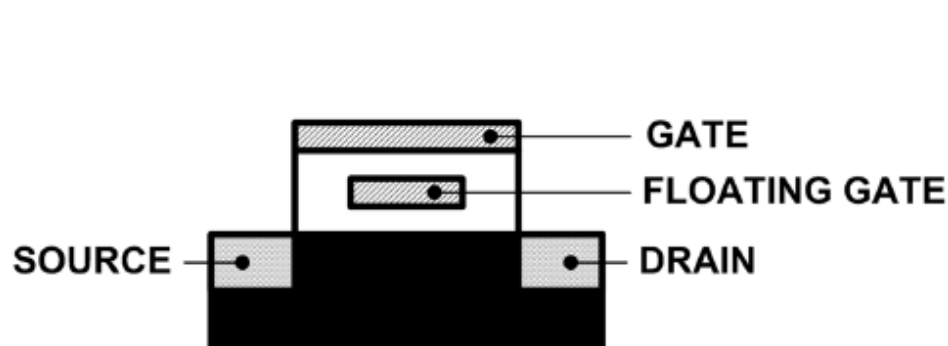
---

- 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



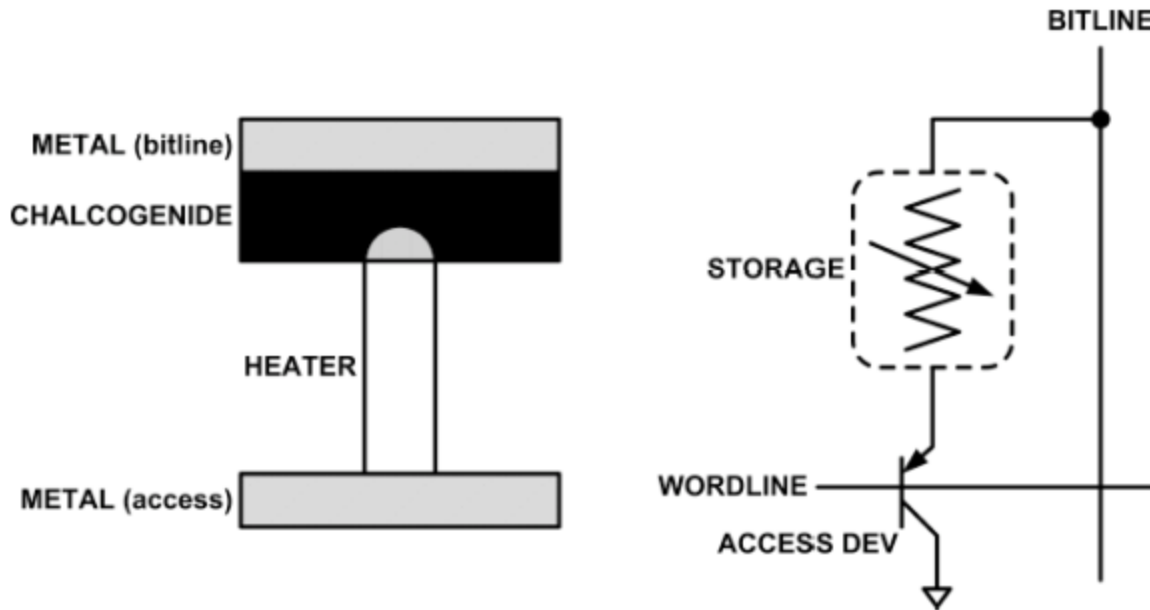
# Promising Resistive Memory Technologies

---

- 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?

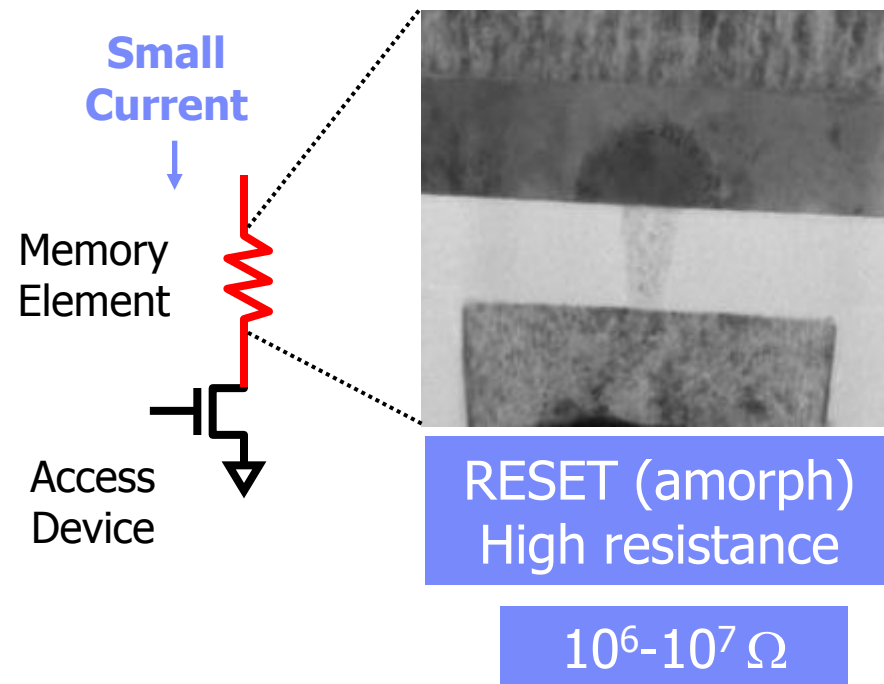
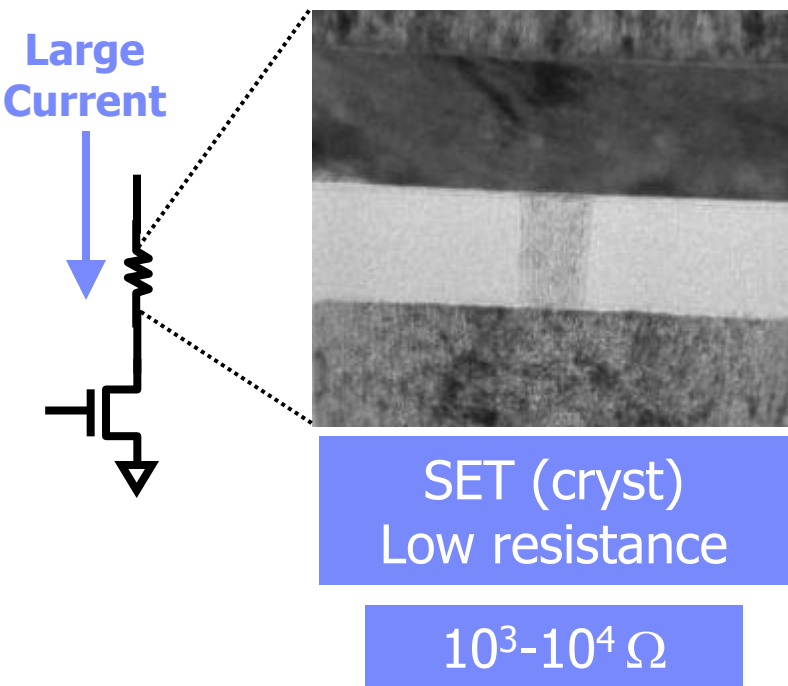
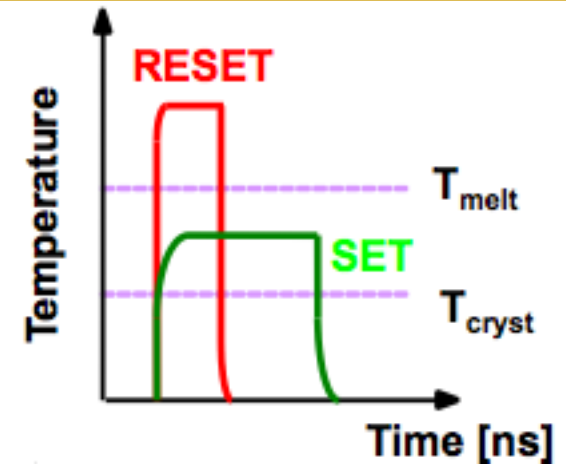
- Phase change material (chalcogenide glass) exists in two states:
  - Amorphous: Low optical reflexivity and high electrical resistivity
  - Crystalline: High optical reflexivity and low electrical resistivity



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



# 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
  - Retain data for >10 years at 85C
- No refresh needed, low idle power

# Phase Change Memory Properties

---

- Surveyed prototypes from 2003-2008 (ITRS, IEDM, VLSI, ISSCC)
- Derived PCM parameters for  $F=90\text{nm}$
  
- Lee, Ipek, Mutlu, Burger, “Architecting Phase Change Memory as a Scalable DRAM Alternative,” ISCA 2009.

Table 1. Technology survey.

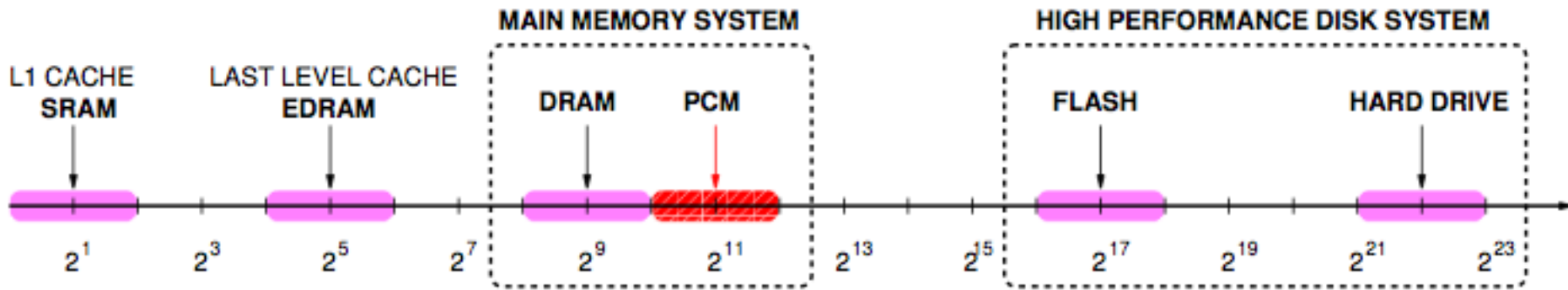
Parameter*	Published prototype									
	Horri <sup>6</sup>	Ahn <sup>12</sup>	Bedeschi <sup>13</sup>	Oh <sup>14</sup>	Pellizer <sup>15</sup>	Chen <sup>5</sup>	Kang <sup>16</sup>	Bedeschi <sup>9</sup>	Lee <sup>10</sup>	Lee <sup>2</sup>
Year	2003	2004	2004	2005	2006	2006	2006	2008	2008	**
Process, $F$ (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 ( $\mu\text{m}^2$ )	**	0.290	0.290	**	0.097	60 nm <sup>2</sup>	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 ( $\mu\text{A}$ )	**	**	40	**	**	**	**	**	**	40
Read voltage (V)	**	3.0	1.0	1.8	1.6	**	1.8	**	1.8	1.0
Read power ( $\mu\text{W}$ )	**	**	40	**	**	**	**	**	**	40
Read energy (pJ)	**	**	2.0	**	**	**	**	**	**	2.0
Set time (ns)	100	150	150	180	**	80	300	**	400	150
Set current ( $\mu\text{A}$ )	200	**	300	200	**	55	**	**	**	150
Set voltage (V)	**	**	2.0	**	**	1.25	**	**	**	1.2
Set power ( $\mu\text{W}$ )	**	**	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 ( $\mu\text{A}$ )	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 ( $\mu\text{W}$ )	**	**	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<sub>2</sub>Sb<sub>2</sub>Te<sub>5</sub>; 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



Typical Access Latency (in terms of processor cycles for a 4 GHz processor)

- 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  $\mu\text{A}$  Rd, 150  $\mu\text{A}$  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
- $10^{-8}\text{x}$  DRAM,  $10^3\text{x}$  NAND Flash

## ■ Cell Size

- 9-12F<sup>2</sup> 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 (capacity and cost)
- Non volatility
- Low idle power (no refresh)

## ■ Cons

- Higher latencies:  $\sim 4\text{-}15\times$  DRAM (especially write)
- Higher active energy:  $\sim 2\text{-}50\times$  DRAM (especially write)
- Lower endurance (a cell dies after  $\sim 10^8$  writes)
- Reliability issues (resistance drift)

## ■ Challenges in enabling PCM as DRAM replacement/helper:

- Mitigate PCM shortcomings
- Find the right way to place PCM in the system

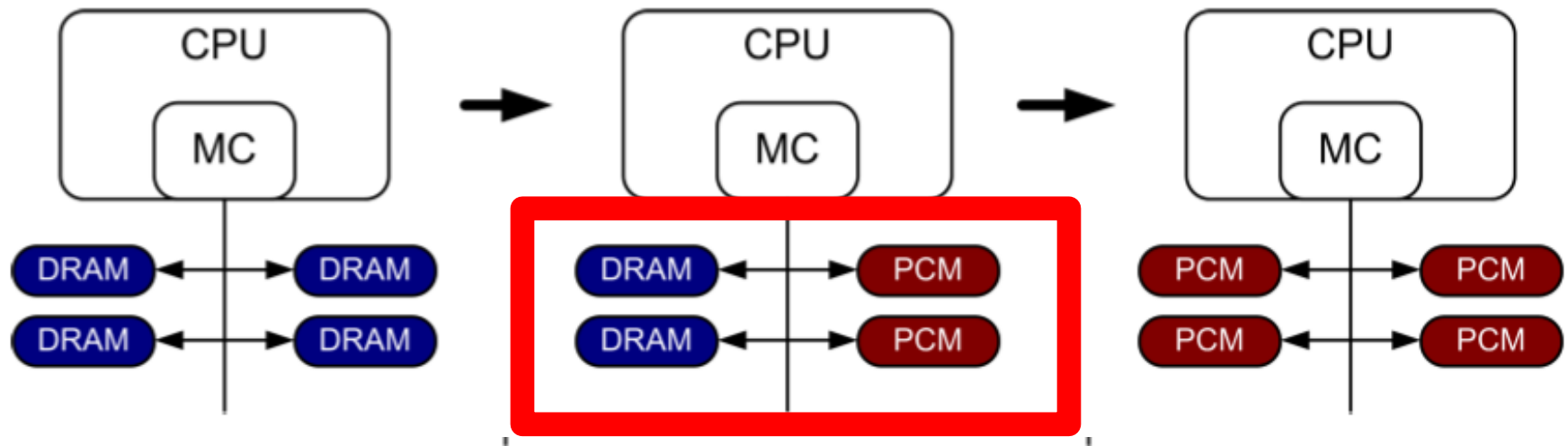
# PCM-based Main Memory: Research Challenges

---

- 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]:
  - How to partition/migrate data between PCM and DRAM

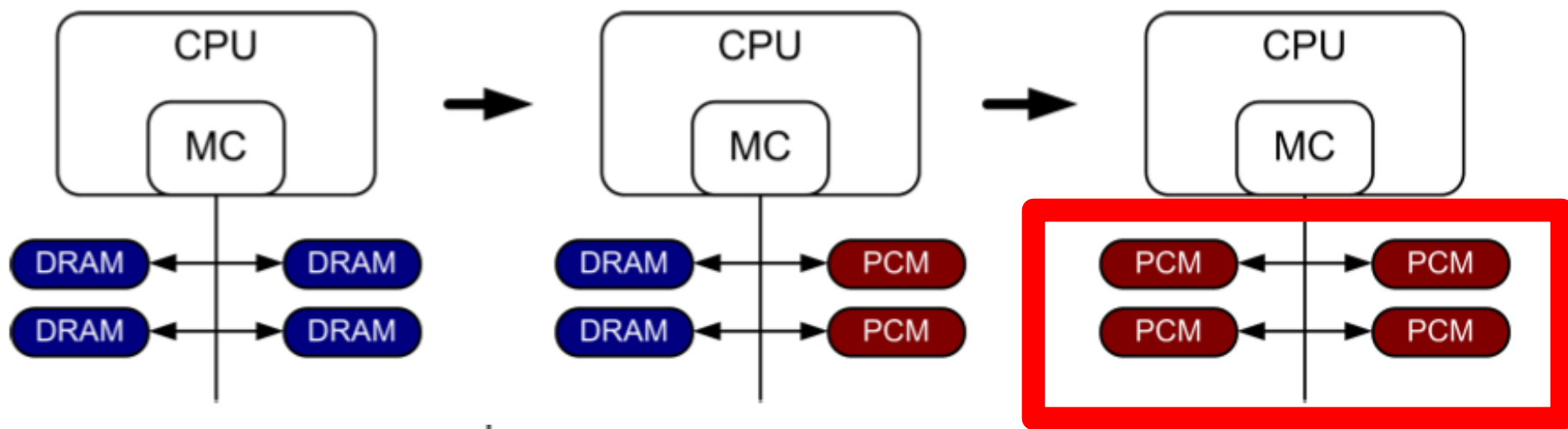
# Hybrid Memory Systems: Challenges

---

- **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

# 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<sup>2</sup> using BJT
- ▷ 1.5× DRAM

## Latency

- ▷ 50ns Rd, 150ns Wr
- ▷ 4×, 12× DRAM

## Endurance

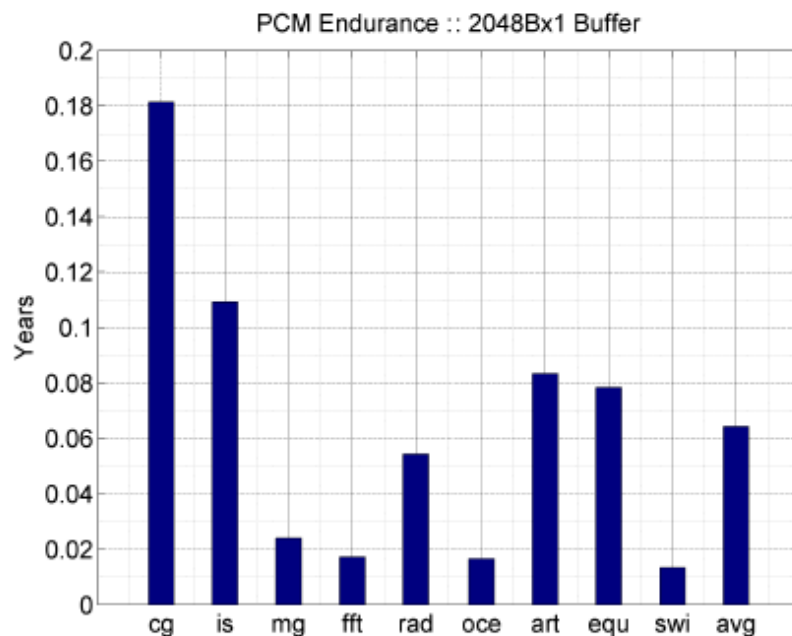
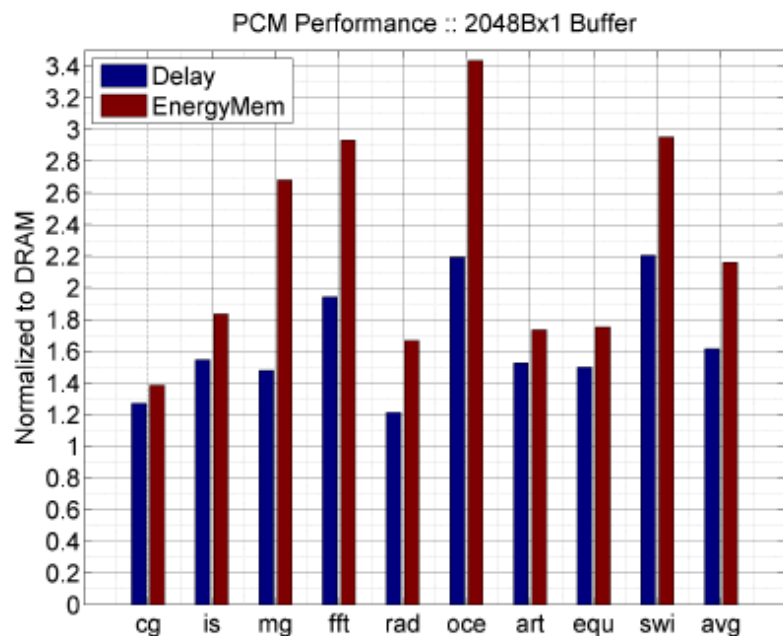
- ▷ 1E+08 writes
- ▷ 1E-08× DRAM

## 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

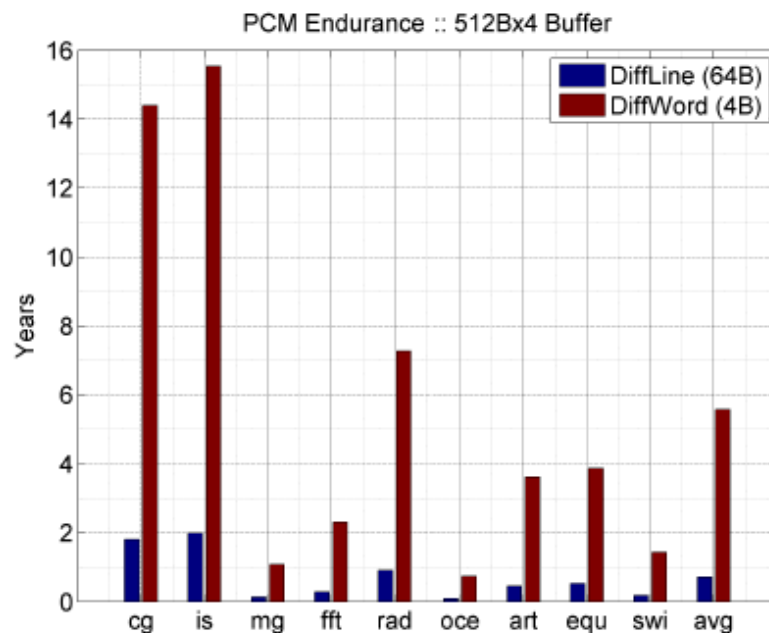
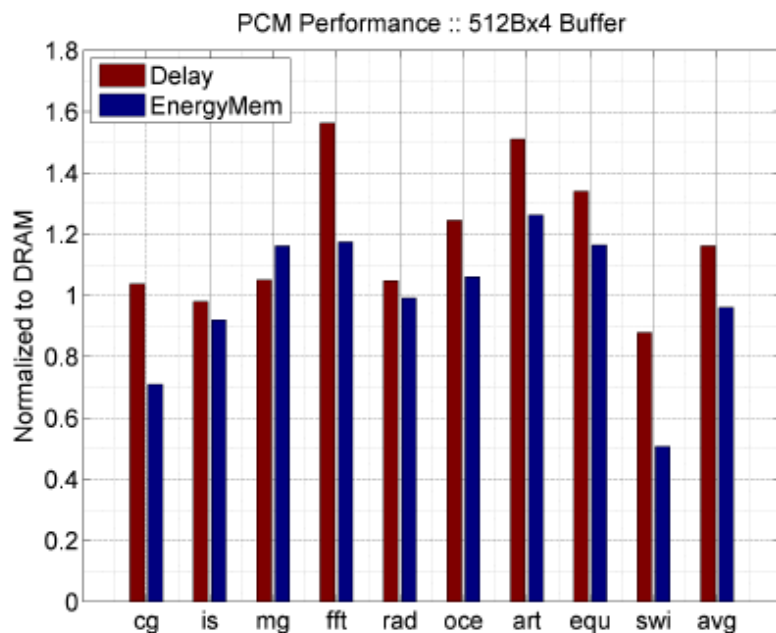


- Lee, Ipek, Mutlu, Burger, “Architecting Phase Change Memory as a Scalable DRAM Alternative,” ISCA 2009.



# Results: Architected PCM as Main Memory

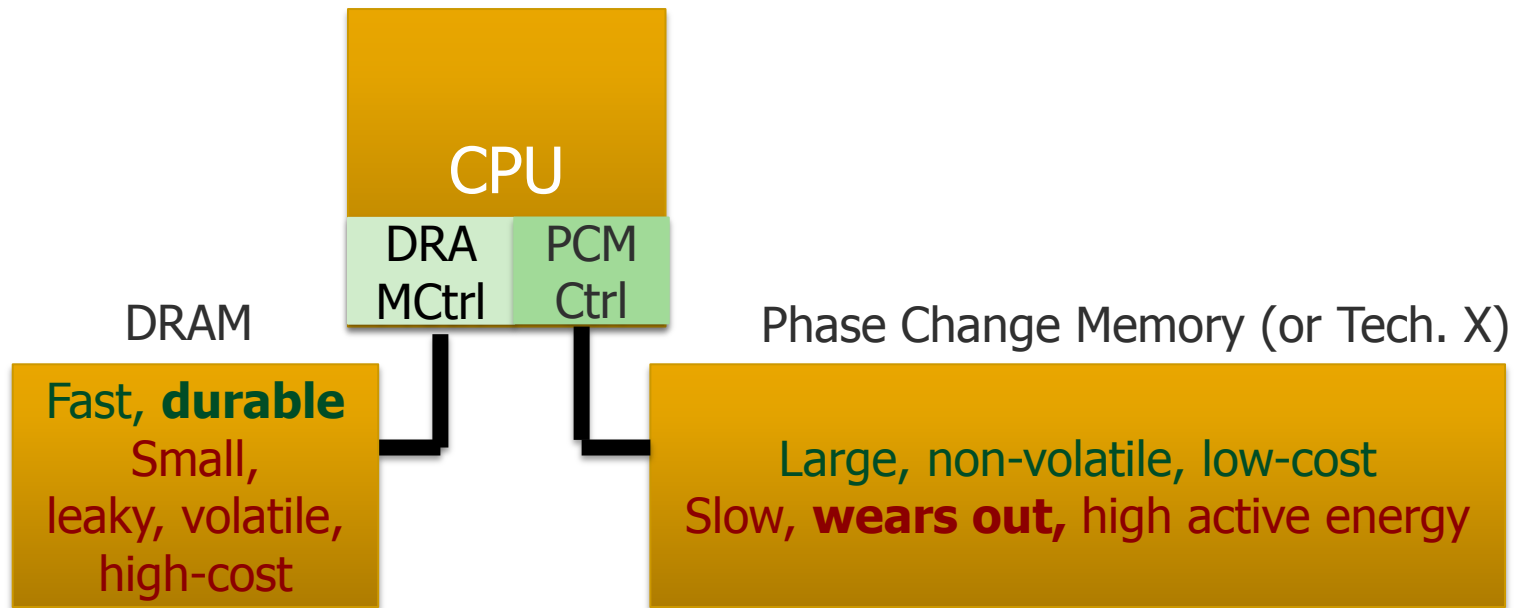
- 1.2x delay, 1.0x energy, 5.6-year average lifetime
- Scaling improves energy, endurance, density



- 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?

# 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+, "Row Buffer Locality Aware Caching Policies for Hybrid Memories," ICCD 2012 Best Paper Award.

18-740: Computer Architecture  
Recitation 4:  
Rethinking Memory System Design

Prof. Onur Mutlu

Carnegie Mellon University

Fall 2015

September 22, 2015