Rethinking Memory System Design (for Data-Intensive Computing)

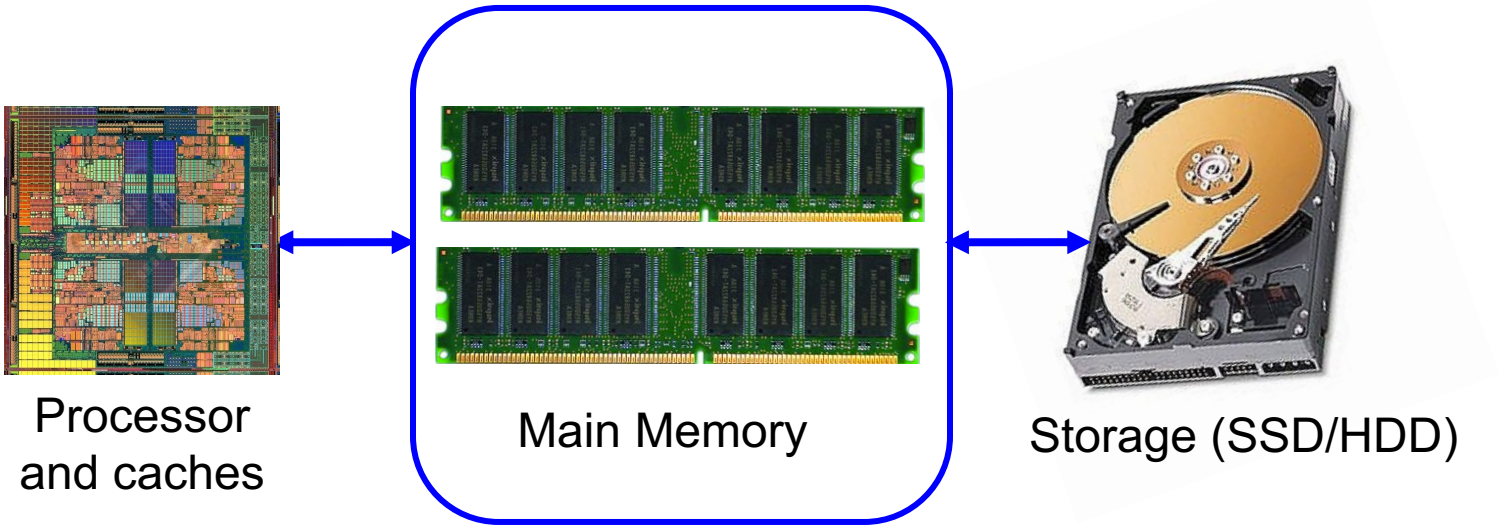
Onur Mutlu
onur@cmu.edu
http://users.ece.cmu.edu/~omutlu/

July 20, 2015

SAMOS Keynote

Carnegie Mellon

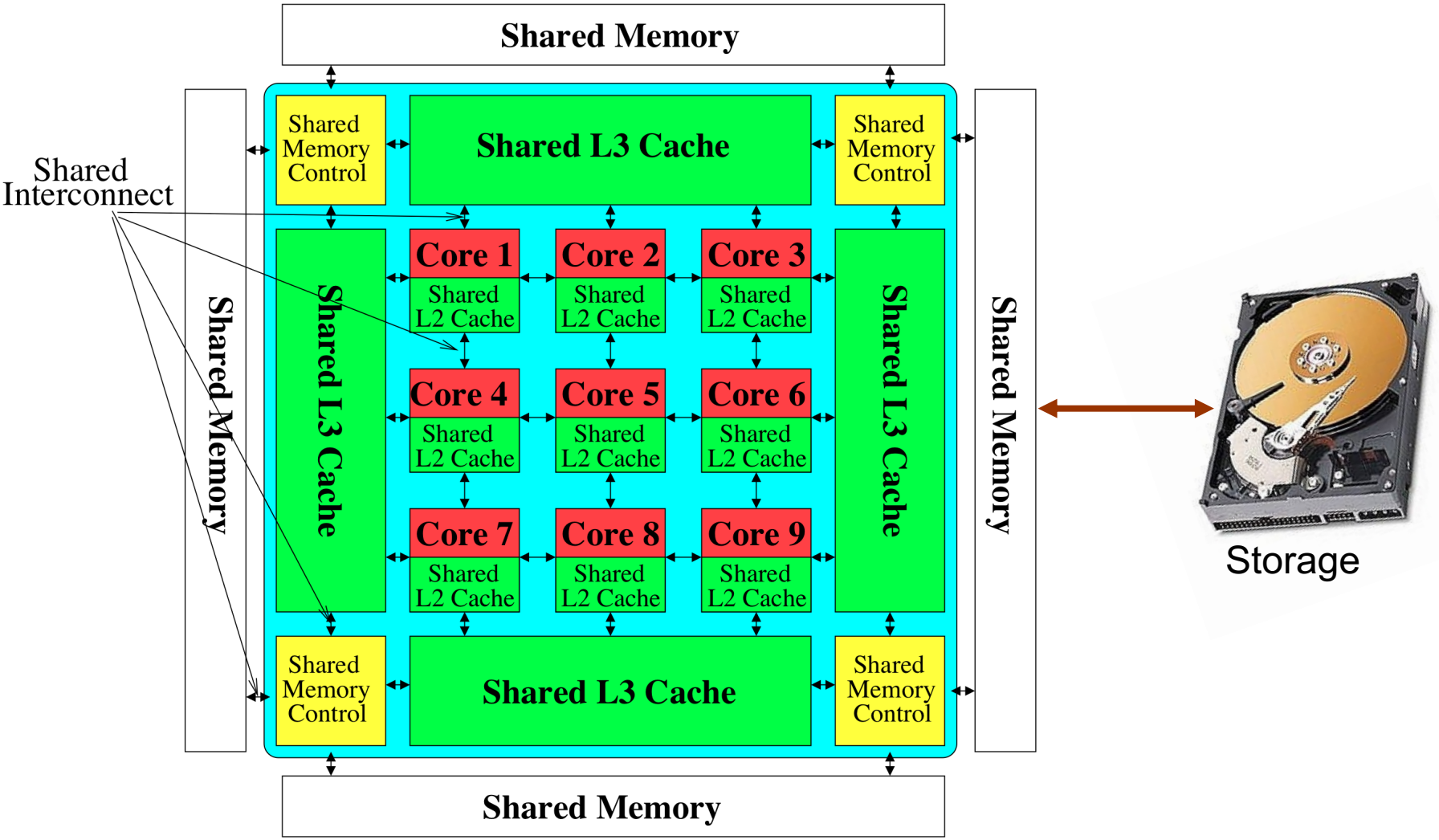
The Main Memory System



- Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor

- Main memory system must scale (in *size*, *technology*, *efficiency*, *cost*, and *management algorithms*) to maintain performance growth and technology scaling benefits

Memory System: A *Shared Resource* View



State of the Main Memory System

- Recent technology, architecture, and application trends
  - lead to new requirements
  - exacerbate old requirements

- DRAM and memory controllers, as we know them today, are (will be) unlikely to satisfy all requirements

- Some emerging non-volatile memory technologies (e.g., PCM) enable new opportunities: memory+storage merging

- We need to rethink the main memory system
  - to fix DRAM issues and enable emerging technologies
  - to satisfy all requirements

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

Major Trends Affecting Main Memory (I)

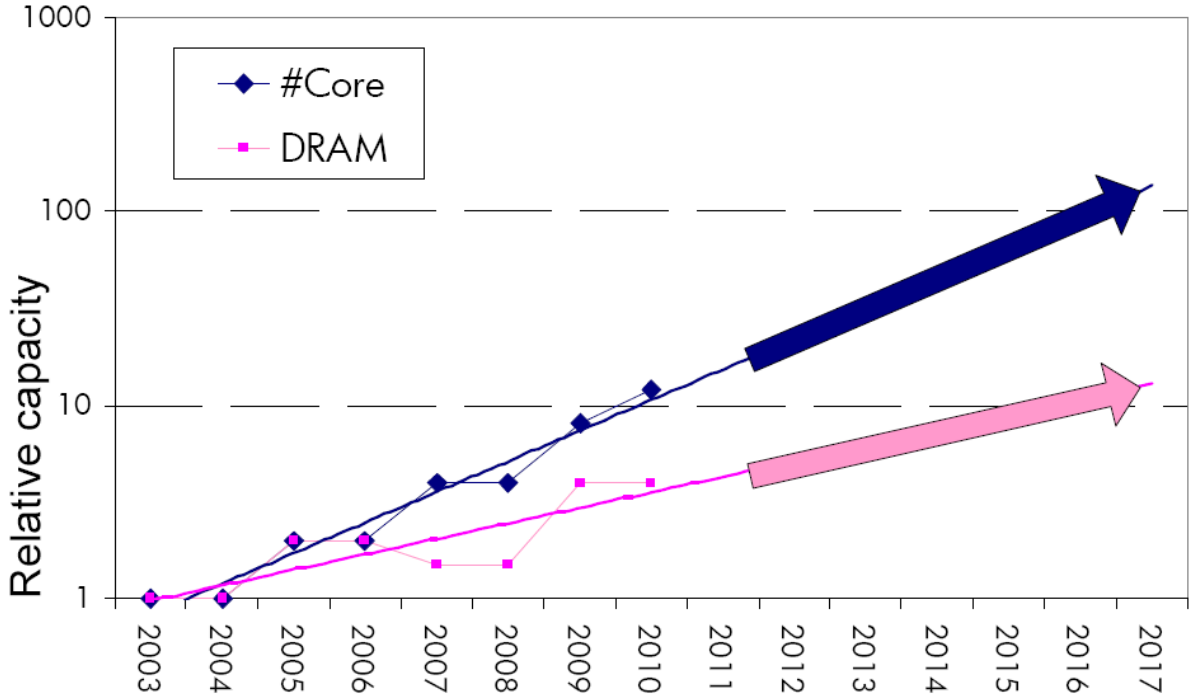
- Need for main memory capacity, bandwidth, QoS increasing
- Main memory energy/power is a key system design concern
- DRAM technology scaling is ending

Major Trends Affecting Main Memory (II)

- Need for main memory capacity, bandwidth, QoS increasing
  - Multi-core: increasing number of cores/agents
  - Data-intensive applications: increasing demand/hunger for data
  - Consolidation: cloud computing, GPUs, mobile, heterogeneity
- Main memory energy/power is a key system design concern
- DRAM technology scaling is ending

Example: The Memory Capacity Gap

Core count doubling ~ every 2 years
DRAM DIMM capacity doubling ~ every 3 years



The picture can't be displayed.

- *Memory capacity per core* expected to drop by 30% every two years
- Trends worse for *memory bandwidth per core*!

Major Trends Affecting Main Memory (III)

- Need for main memory capacity, bandwidth, QoS increasing

- Main memory energy/power is a key system design concern
  - ~40-50% energy spent in off-chip memory hierarchy [Lefurgy, IEEE Computer 2003]
  - DRAM consumes power even when not used (periodic refresh)

- DRAM technology scaling is ending

Major Trends Affecting Main Memory (IV)

- Need for main memory capacity, bandwidth, QoS increasing

- Main memory energy/power is a key system design concern

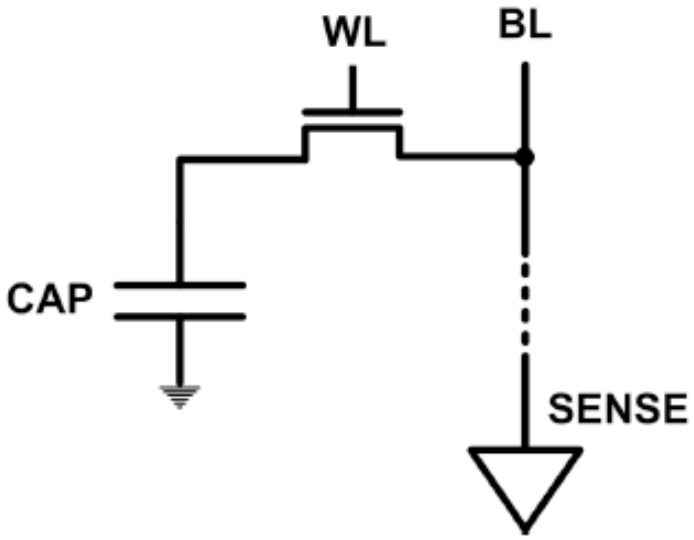
- DRAM technology scaling is ending
  - ITRS projects DRAM will not scale easily below X nm
  - Scaling has provided many benefits:
    - higher capacity (density), lower cost, lower energy

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

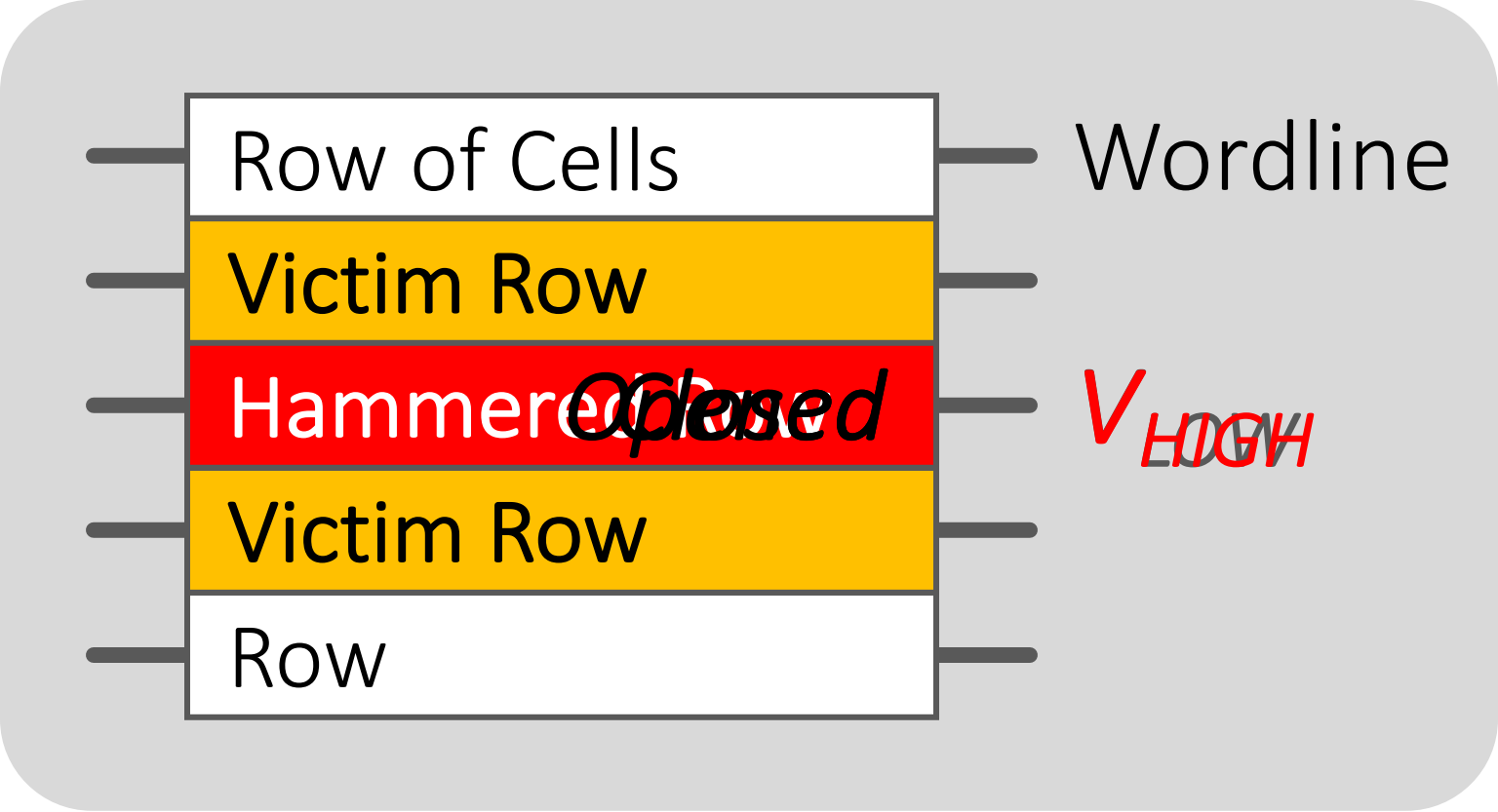
The DRAM Scaling Problem

- DRAM stores charge in a capacitor (charge-based memory)
  - Capacitor must be large enough for reliable sensing
  - Access transistor should be large enough for low leakage and high retention time
  - Scaling beyond 40-35nm (2013) is challenging [ITRS, 2009]



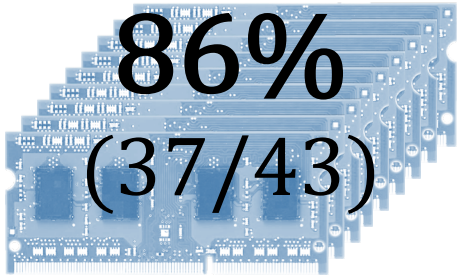
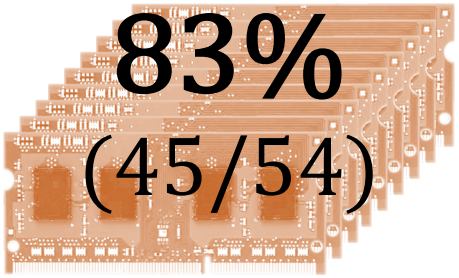
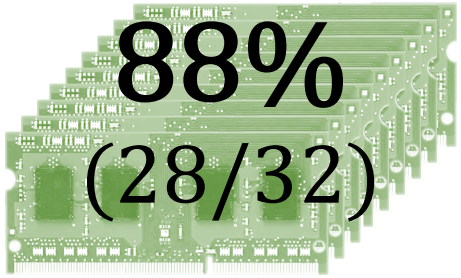
- DRAM capacity, cost, and energy/power hard to scale

An Example of The Scaling Problem



Repeatedly opening and closing a row induces ***disturbance errors*** *in adjacent rows in* ***most real DRAM chips*** [Kim+ ISCA 2014]

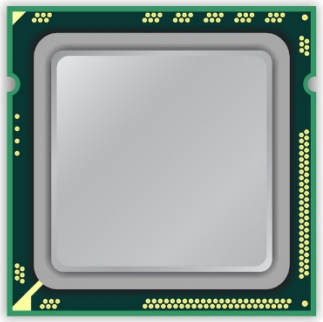
Most DRAM Modules Are At Risk

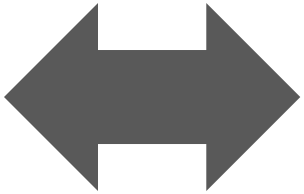
A company	B company	C company
 86% (37/43)	 83% (45/54)	 88% (28/32)
Up to 1.0×10^{7} errors	Up to 2.7×10^{6} errors	Up to 3.3×10^{5} errors

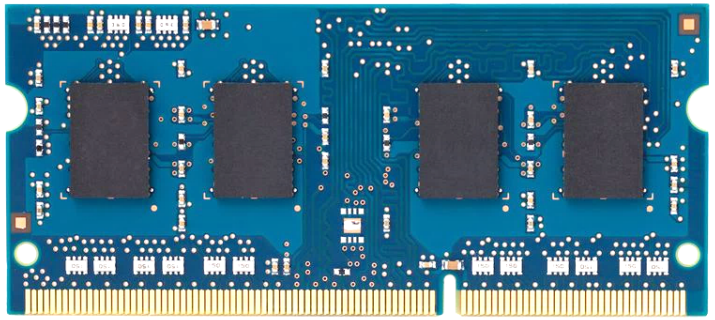
Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

x86 CPU

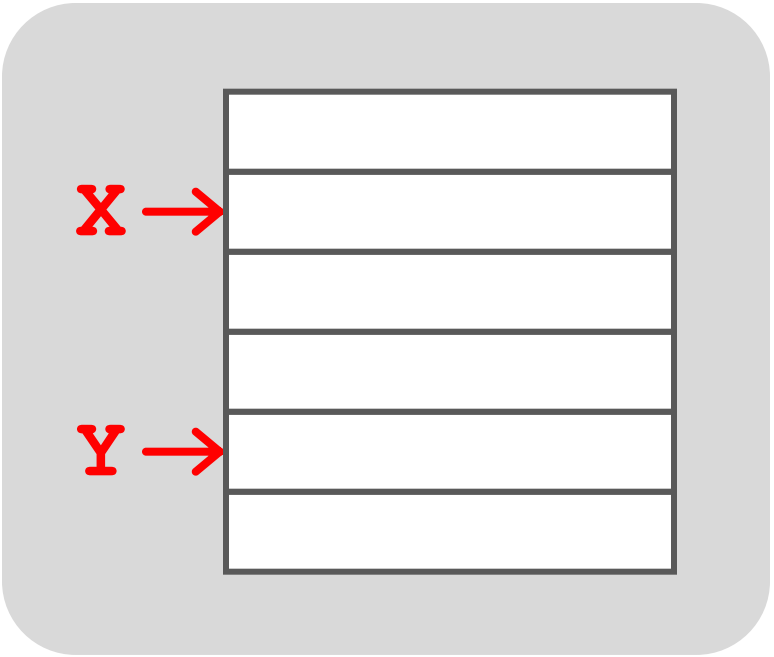
DRAM Module







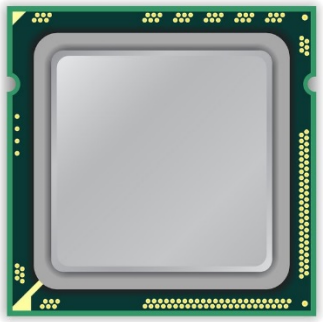
```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```

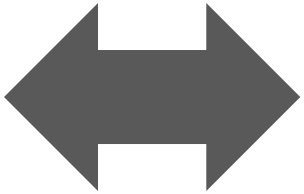


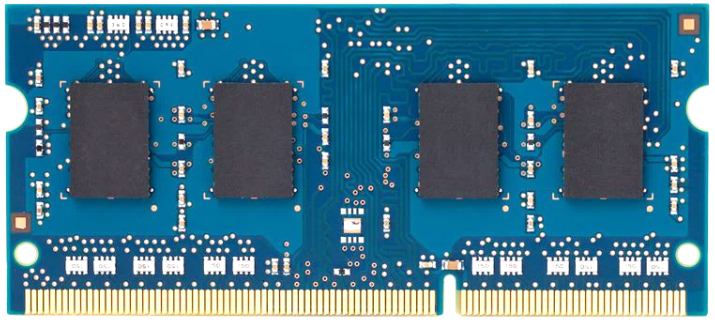
https://github.com/CMU-SAFARI/rowhammer

x86 CPU

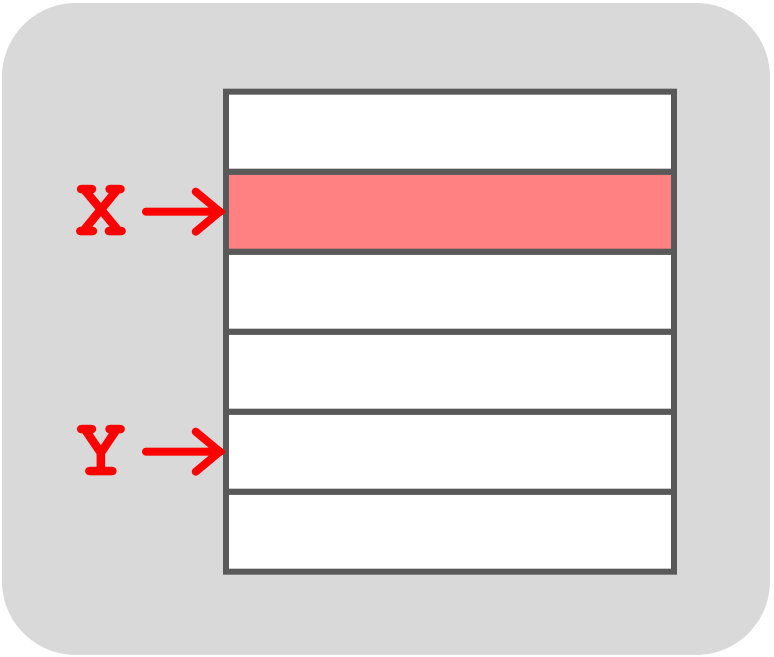
DRAM Module







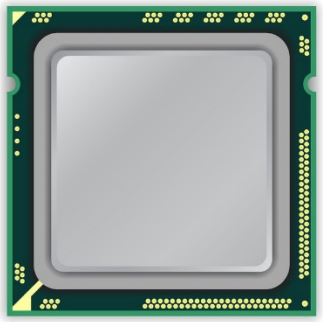
```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```

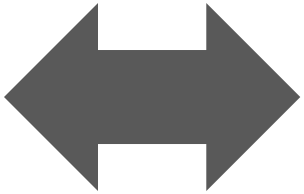


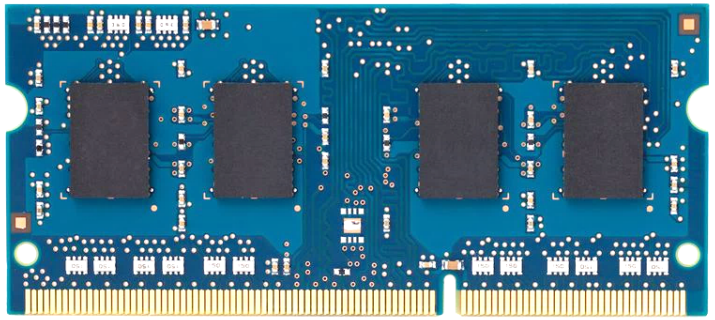
https://github.com/CMU-SAFARI/rowhammer

x86 CPU

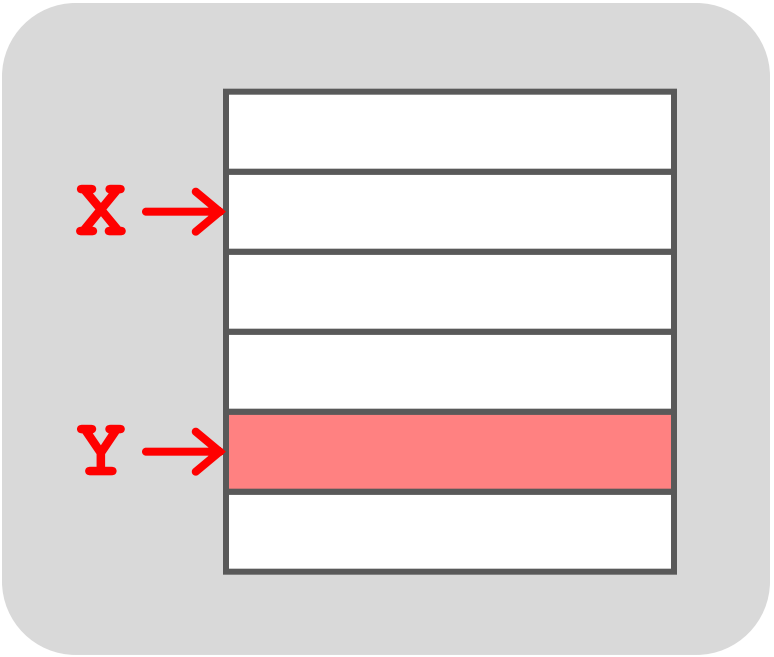
DRAM Module





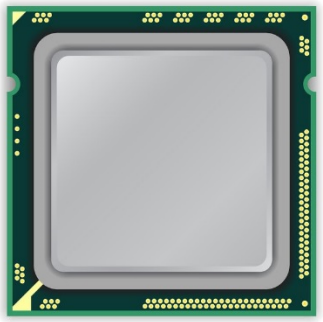


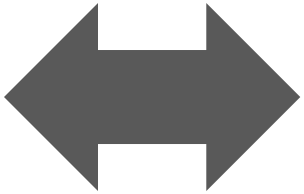
```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```

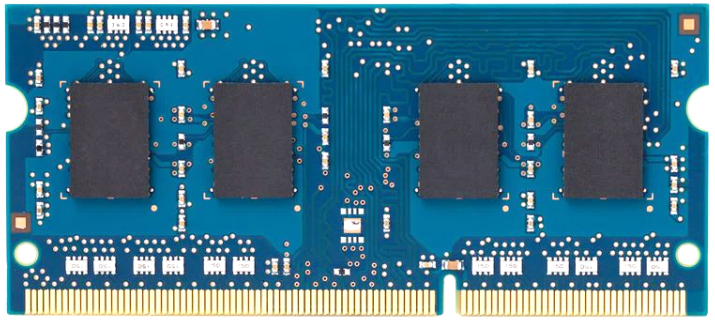


https://github.com/CMU-SAFARI/rowhammer

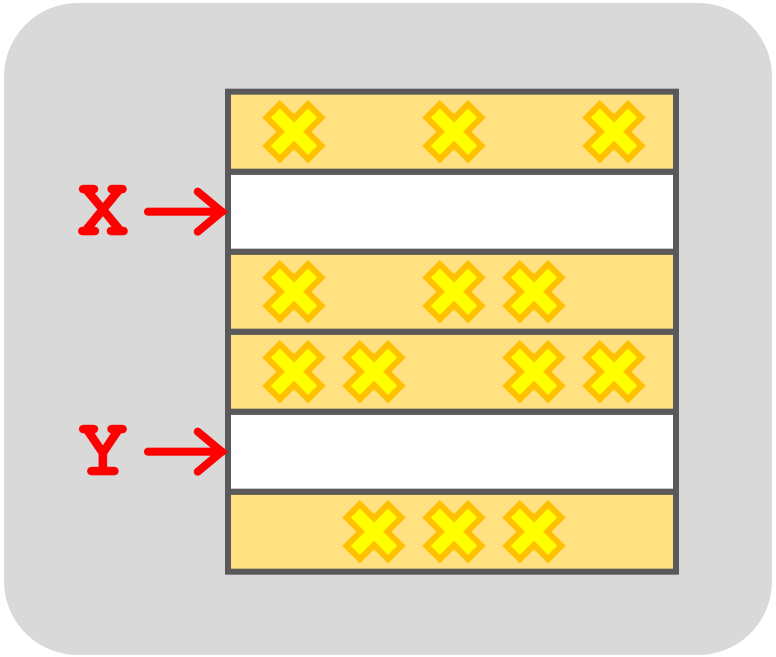
x86 CPU DRAM Module







```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```



https://github.com/CMU-SAFARI/rowhammer

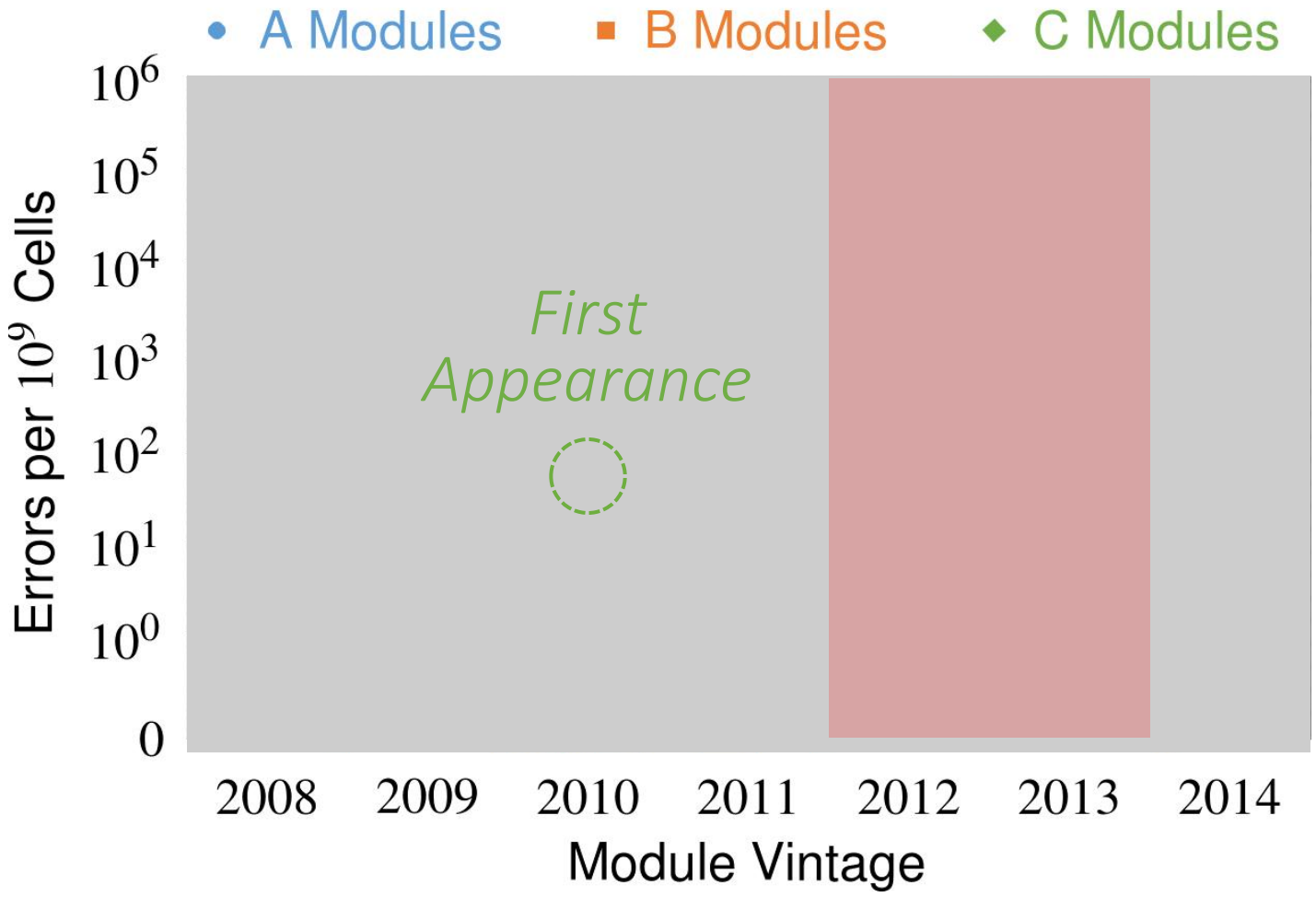
Observed Errors in Real Systems

CPU Architecture	Errors	Access-Rate
Intel Haswell (2013)	22.9K	12.3M/sec
Intel Ivy Bridge (2012)	20.7K	11.7M/sec
Intel Sandy Bridge (2011)	16.1K	11.6M/sec
AMD Piledriver (2012)	59	6.1M/sec

- *A real reliability & security issue*
- *In a more controlled environment, we can induce as many as ten million disturbance errors*

Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

Errors *vs.* Vintage



All modules from 2012–2013 are vulnerable

Experimental Infrastructure (DRAM)

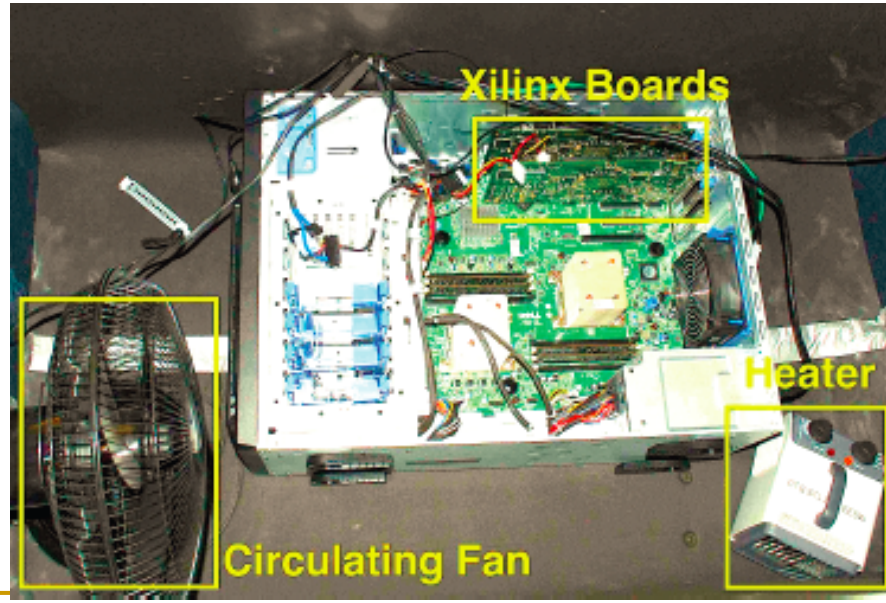
Liu+, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms", ISCA 2013.

Khan+, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," SIGMETRICS 2014.

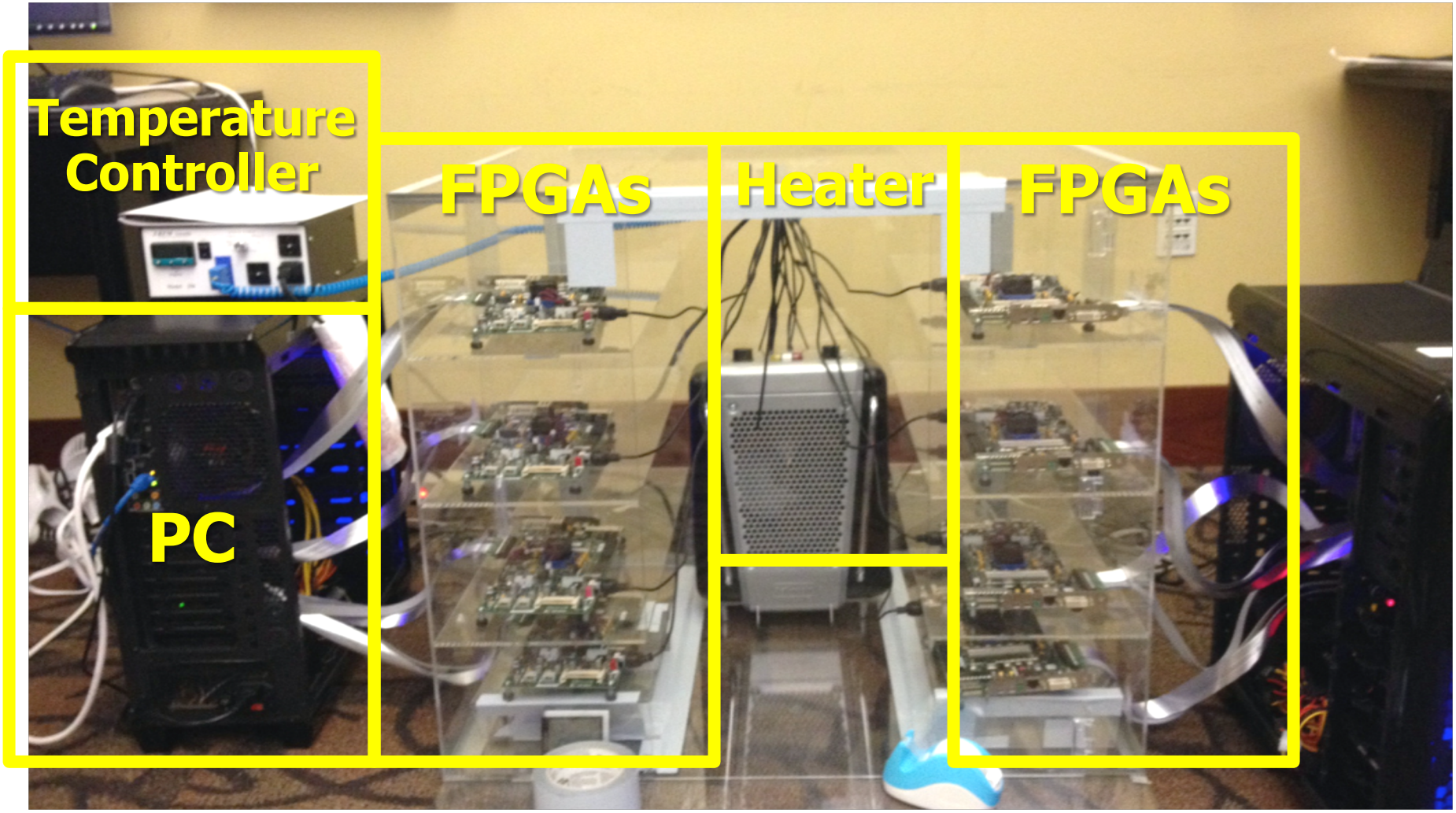
Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors", ISCA 2014.

Lee+, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," HPCA 2015.

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



Experimental Infrastructure (DRAM)



Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

RowHammer Characterization Results

1. Most Modules Are at Risk
2. Errors vs. Vintage
3. Error = Charge Loss
4. Adjacency: Aggressor & Victim
5. Sensitivity Studies
6. Other Results in Paper
7. Solution Space

Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

Security Implications

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Abstract. Memory isolation is a key property of a reliable and secure computing system — an access to one memory address should not have unintended side effects on data stored in other addresses. However, as DRAM process technology

http://users.ece.cmu.edu/~omutlu/pub/dram-row-hammer_isca14.pdf

Project Zero

News and updates from the Project Zero team at Google

http://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

Security Implications

Recap: The DRAM Scaling Problem

DRAM Process Scaling Challenges

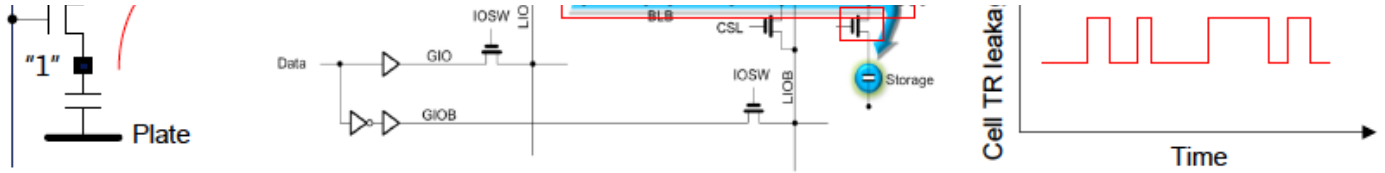
- **Refresh**
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

THE MEMORY FORUM 2014

Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling

Uksong Kang, Hak-soo Yu, Churoo Park, *Hongzhong Zheng,
**John Halbert, **Kuljit Bains, SeongJin Jang, and Joo Sun Choi

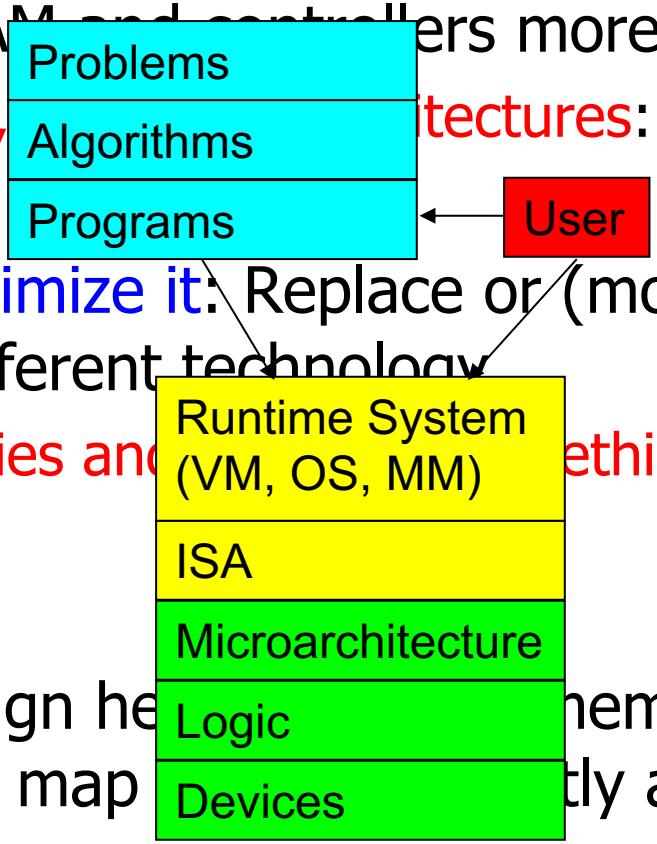
*Samsung Electronics, Hwasung, Korea / *Samsung Electronics, San Jose / **Intel*



Refresh **tWR** **VRT**

How Do We Solve The Problem?

- Fix it: Make DRA[illegible]ers more intelligent
  - New interfaces, [illegible]tectures: 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 1: Fix DRAM

- Overcome DRAM shortcomings with
  - System-DRAM co-design
  - Novel DRAM architectures, interface, functions
  - Better waste management (efficient utilization)

- Key issues to tackle
  - Enable reliability at low cost
  - Reduce energy
  - Improve latency and bandwidth
  - Reduce waste (capacity, bandwidth, latency)
  - Enable computation close to data

Solution 1: Fix DRAM

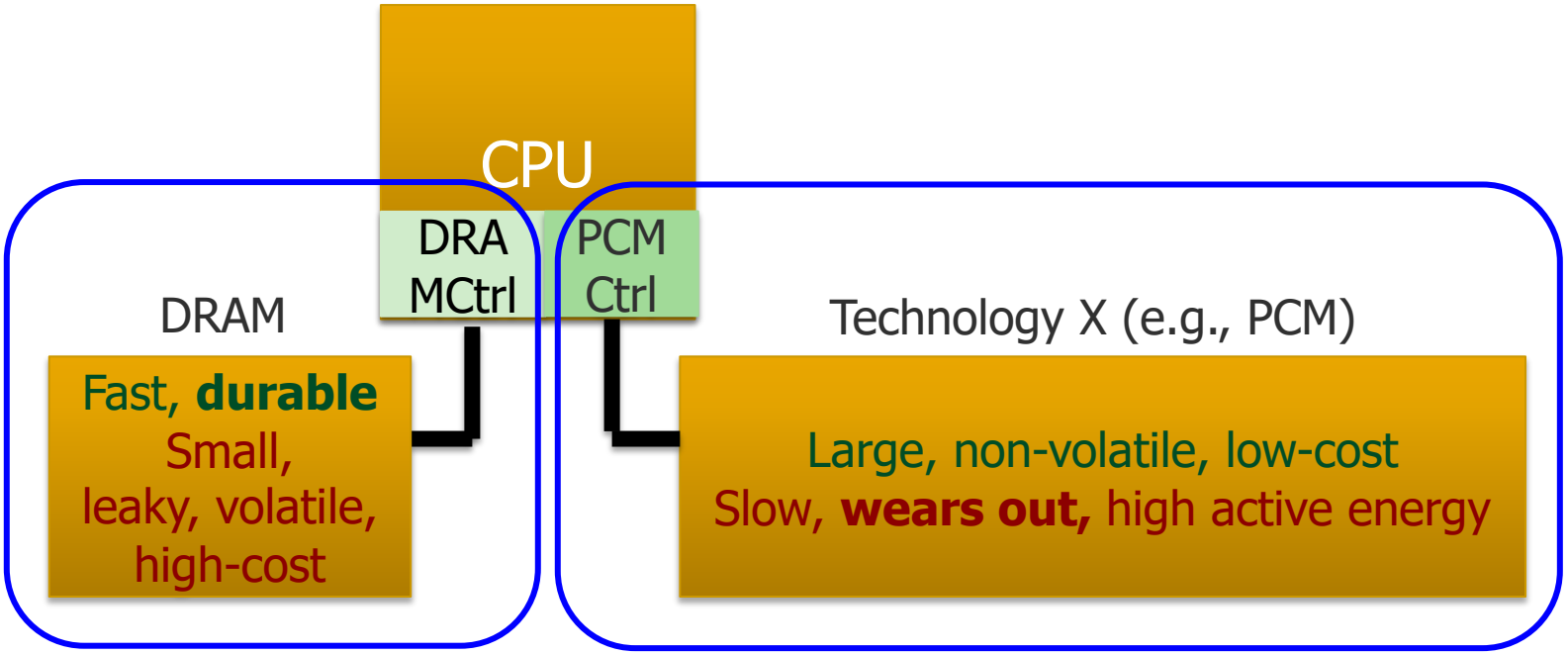
- Liu+, "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.
- Kim+, "A Case for Exploiting Subarray-Level Parallelism in DRAM," ISCA 2012.
- Lee+, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013.
- Liu+, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices," ISCA 2013.
- Seshadri+, "RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data," MICRO 2013.
- Pekhimenko+, "Linearly Compressed Pages: A Main Memory Compression Framework," MICRO 2013.
- Chang+, "Improving DRAM Performance by Parallelizing Refreshes with Accesses," HPCA 2014.
- Khan+, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," SIGMETRICS 2014.
- Luo+, "Characterizing Application Memory Error Vulnerability to Optimize Data Center Cost," DSN 2014.
- Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.
- Lee+, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," HPCA 2015.
- Qureshi+, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems," DSN 2015.
- Meza+, "Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field," DSN 2015.
- Kim+, "Ramulator: A Fast and Extensible DRAM Simulator," IEEE CAL 2015.
- Seshadri+, "Fast Bulk Bitwise AND and OR in DRAM," IEEE CAL 2015.
- Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing," ISCA 2015.
- Ahn+ "PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture," ISCA 2015.
- Avoid DRAM:
  - Seshadri+, "The Evicted-Address Filter: A Unified Mechanism to Address Both Cache Pollution and Thrashing," PACT 2012.
  - Pekhimenko+, "Base-Delta-Immediate Compression: Practical Data Compression for On-Chip Caches," PACT 2012.
  - Seshadri+, "The Dirty-Block Index," ISCA 2014.
  - Pekhimenko+, "Exploiting Compressed Block Size as an Indicator of Future Reuse," HPCA 2015.
  - Vijaykumar+, "A Case for Core-Assisted Bottleneck Acceleration in GPUs: Enabling Flexible Data Compression with Assist Warps," ISCA 2015.

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.

Solution 3: Hybrid Memory Systems

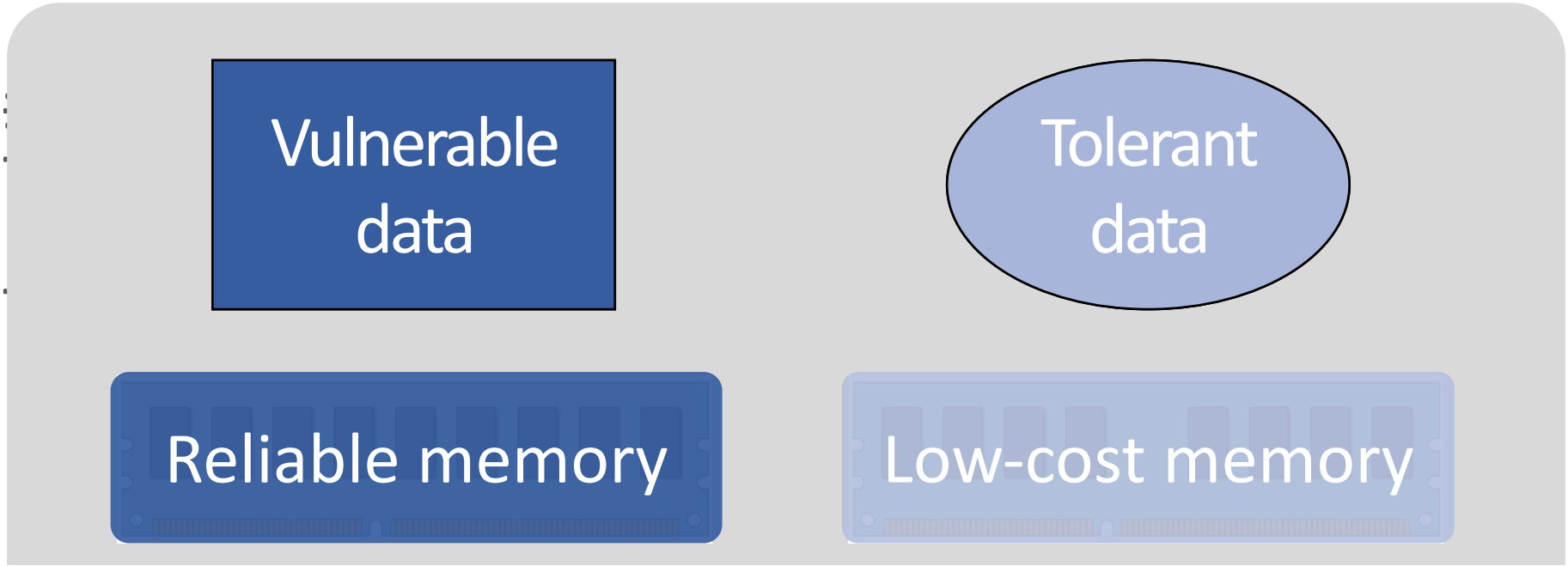


Hardware/software manage data allocation and movement
to achieve the best of multiple technologies

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.

Exploiting Memory Error Tolerance with Hybrid Memory Systems



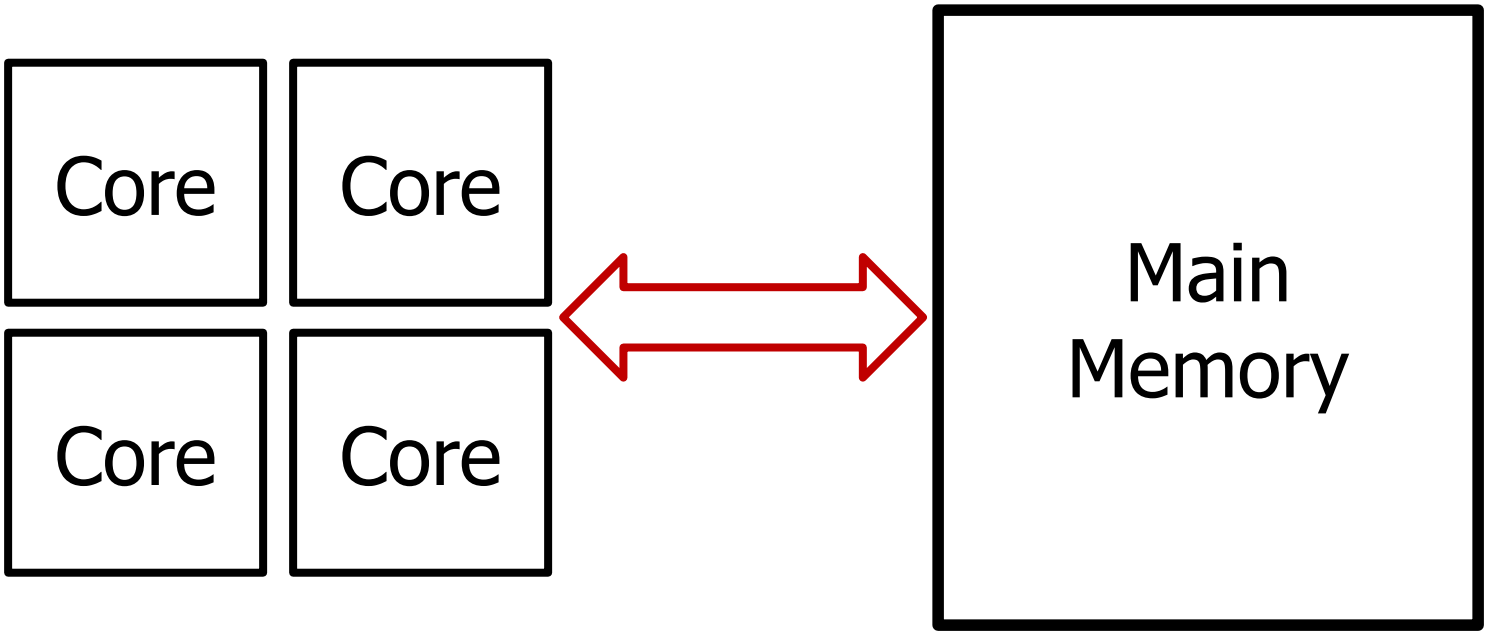
On Microsoft's Web Search workload

Reduces server hardware cost by 4.7 %

Achieves single server availability target of 99.90 %

Heterogeneous-**R**eliability **M**emory [DSN 2014]

An Orthogonal Issue: Memory Interference



Cores' interfere with each other when accessing shared main memory

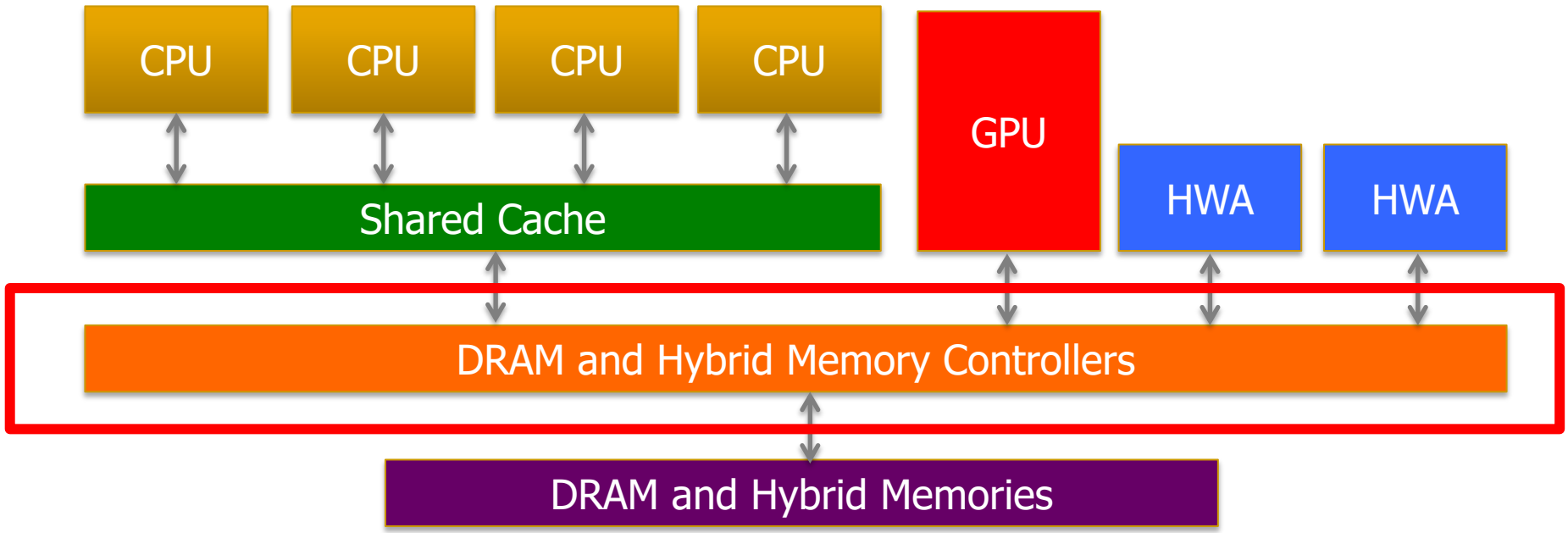
An Orthogonal Issue: Memory Interference

- Problem: Memory interference between cores is uncontrolled
  → unfairness, starvation, low performance
  → uncontrollable, unpredictable, vulnerable system

- Solution: QoS-Aware Memory Systems
  - Hardware designed to provide a configurable fairness substrate
    - Application-aware memory scheduling, partitioning, throttling
  - Software designed to configure the resources to satisfy different QoS goals

- QoS-aware memory systems can provide predictable performance and higher efficiency

Goal: Predictable Performance in Complex Systems



- Heterogeneous agents: CPUs, GPUs, and HWAs
- Main memory interference between CPUs, GPUs, HWAs

How to allocate resources to heterogeneous agents to mitigate interference and provide predictable performance?

Strong Memory Service Guarantees

- Goal: Satisfy performance/SLA requirements in the presence of shared main memory, heterogeneous agents, and hybrid memory/storage

- Approach:
  - Develop techniques/models to accurately estimate the performance loss of an application/agent in the presence of resource sharing
  - Develop mechanisms (hardware and software) to enable the resource partitioning/prioritization needed to achieve the required performance levels for all applications
  - All the while providing high system performance

- Example work: Subramanian et al., "MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems," HPCA 2013.

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**

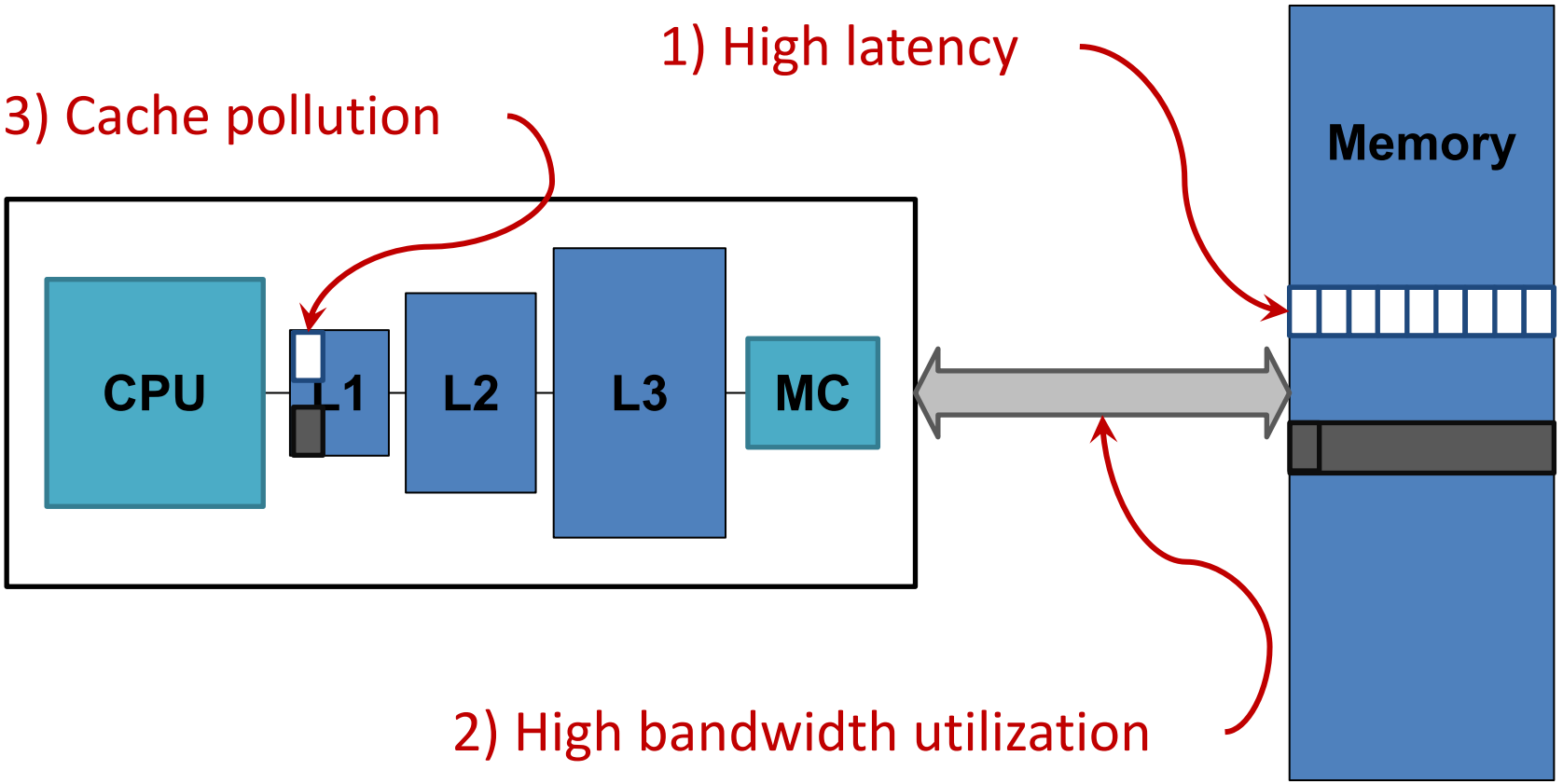
Agenda

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

Rethinking DRAM

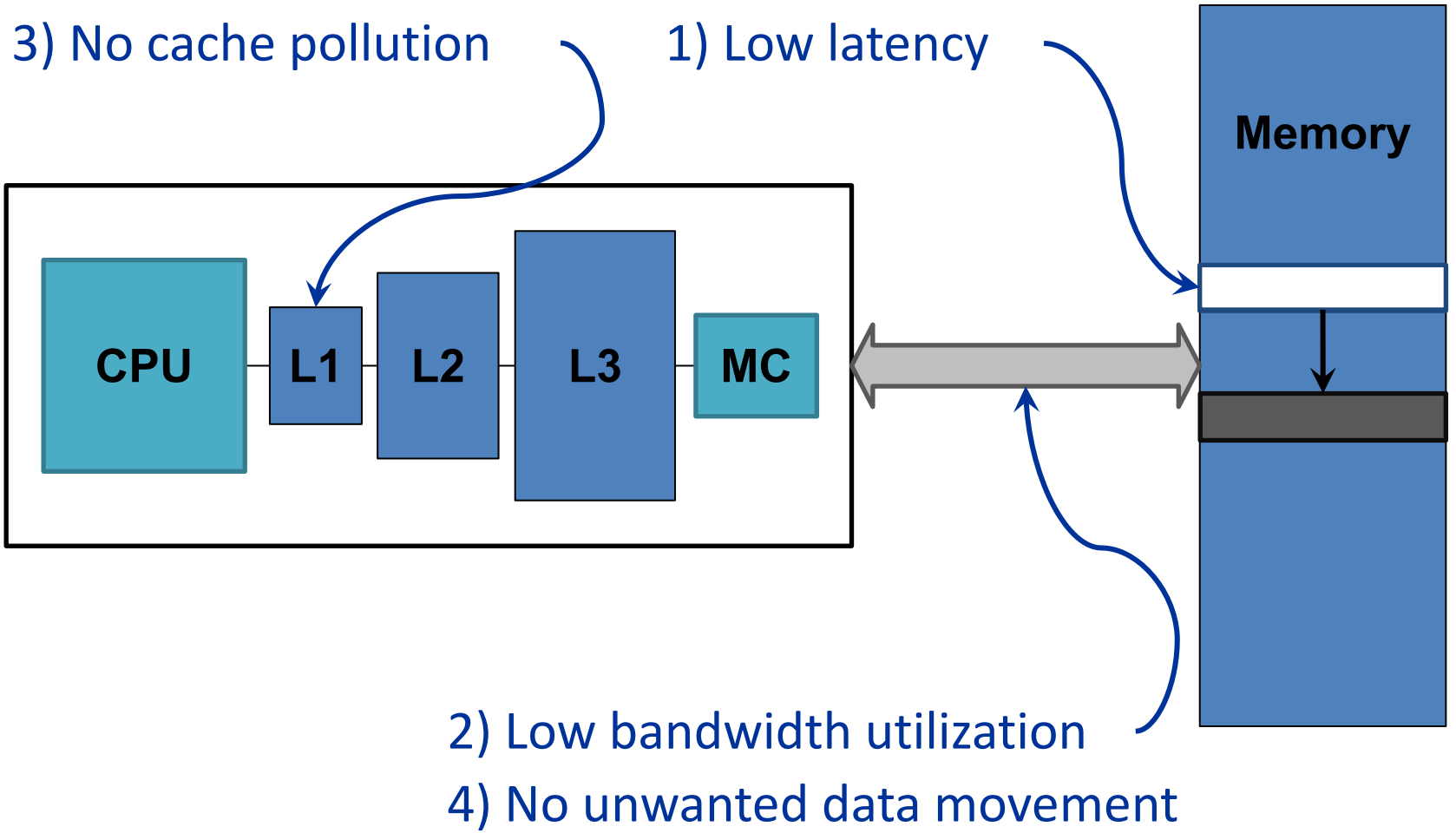
- In-Memory Computation
- Refresh
- Reliability
- Latency
- Bandwidth
- Energy
- Memory Compression

Today's Memory: Bulk Data Copy

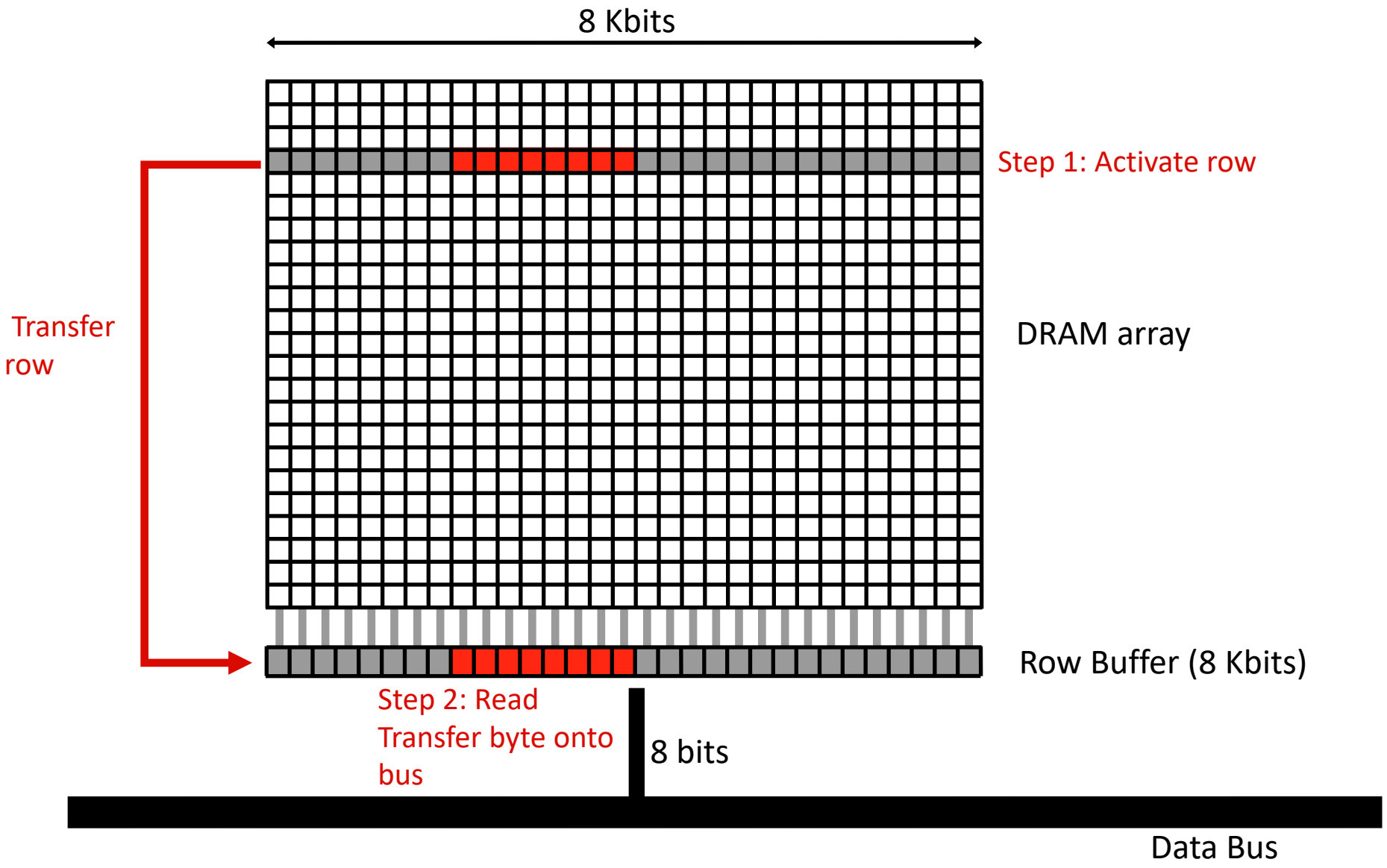


1046ns, 3.6uJ (for 4KB page copy via DMA)

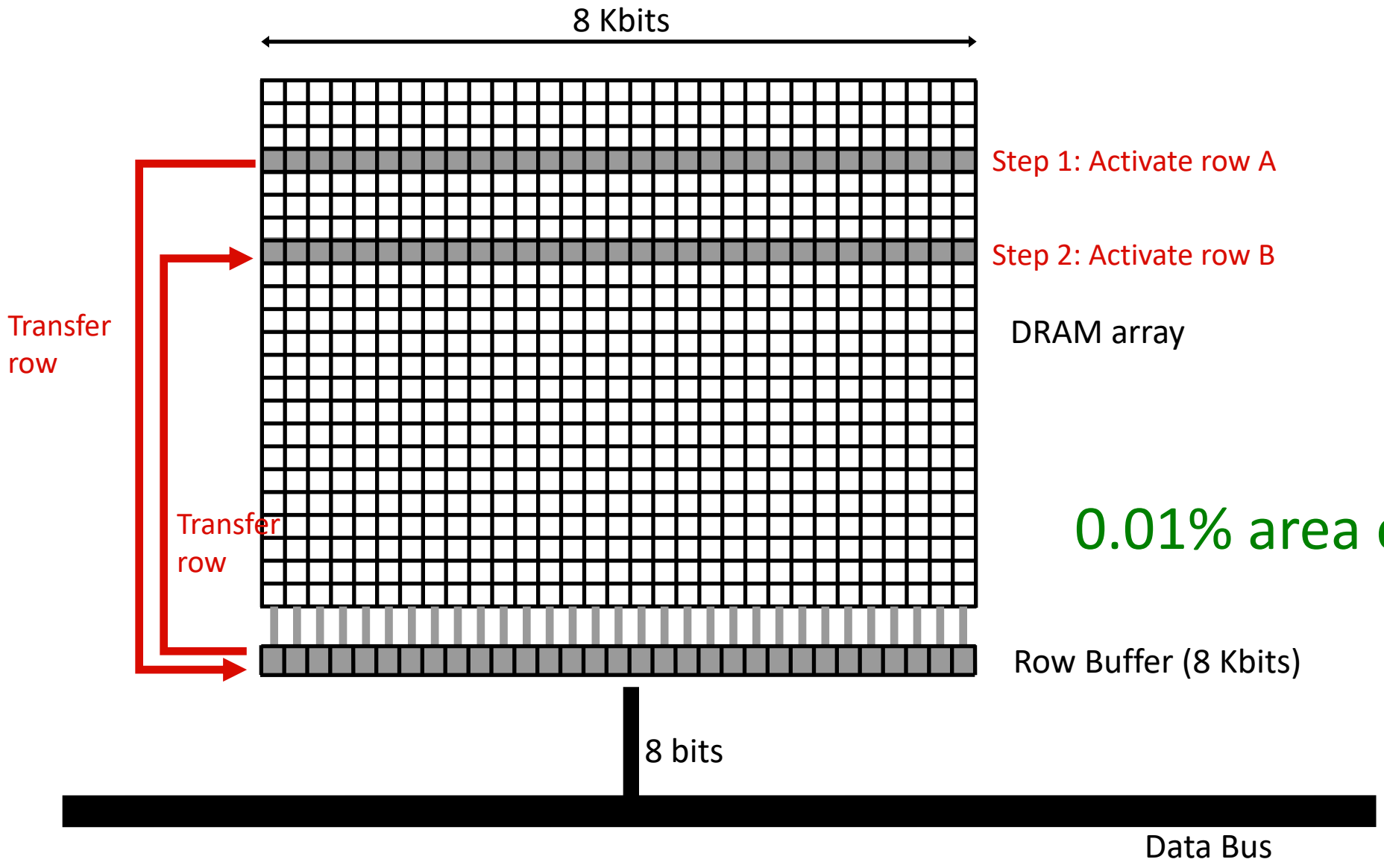
Future: RowClone (In-Memory Copy)



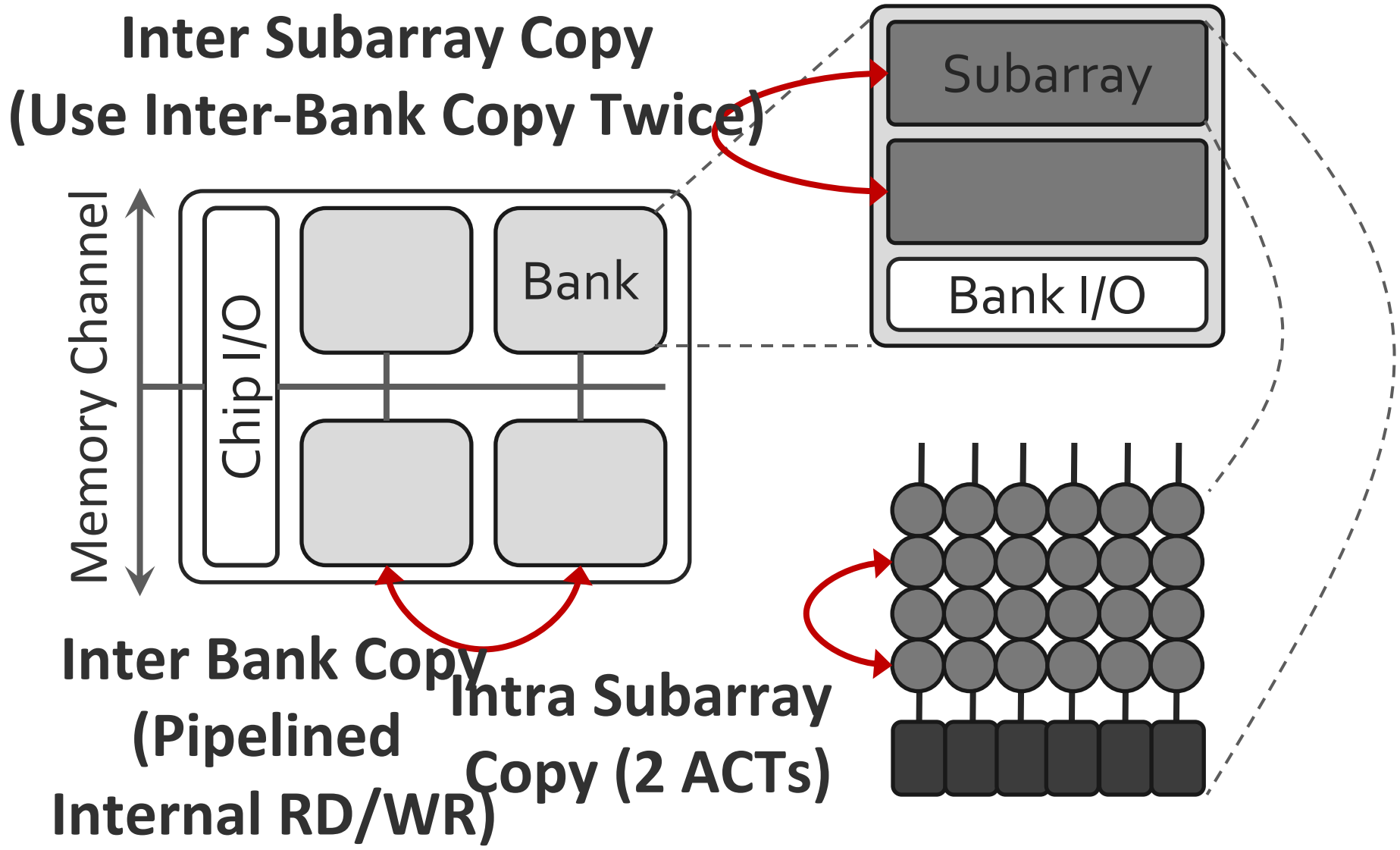
DRAM Subarray Operation (load one byte)



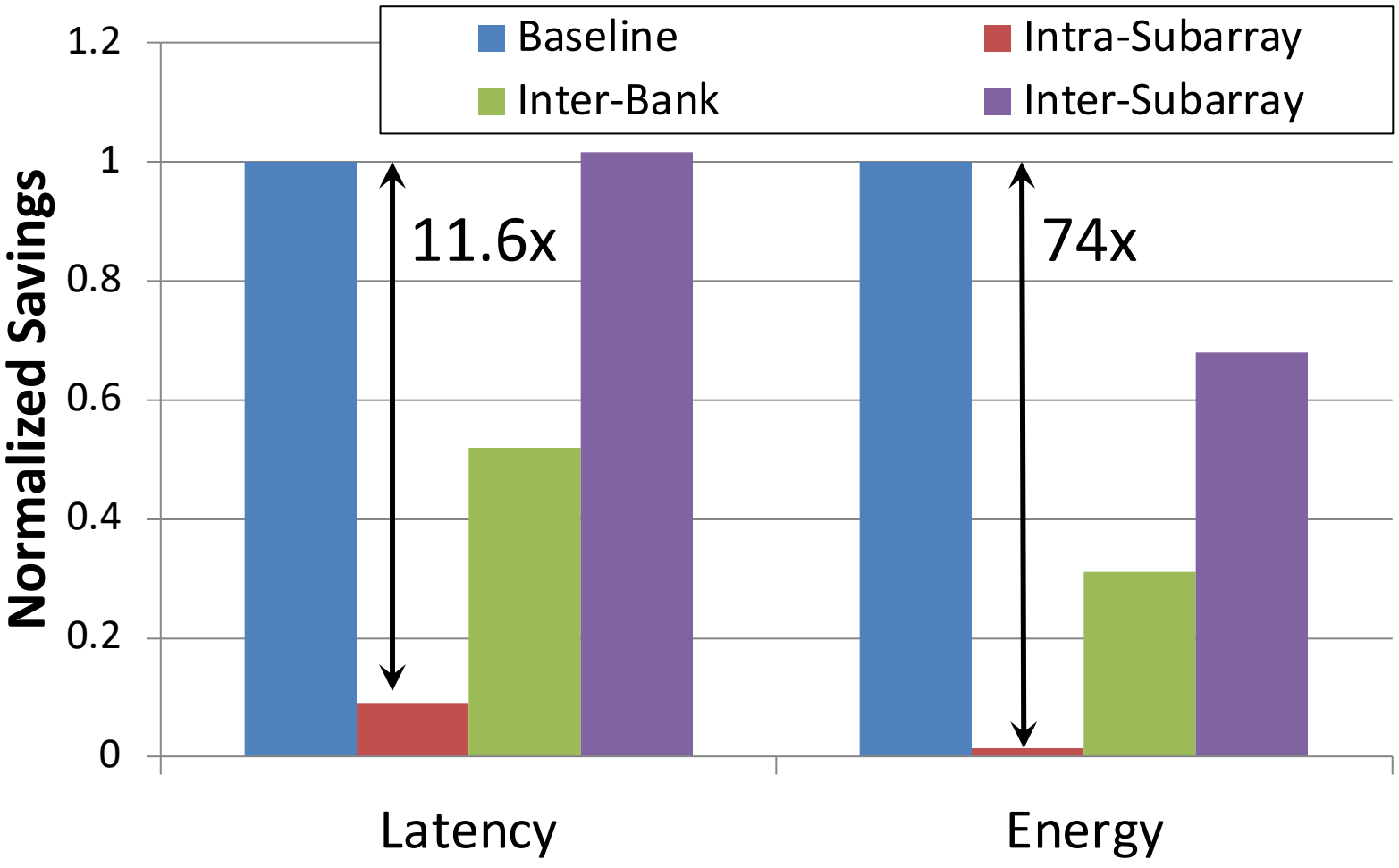
RowClone: In-DRAM Row Copy



Generalized RowClone

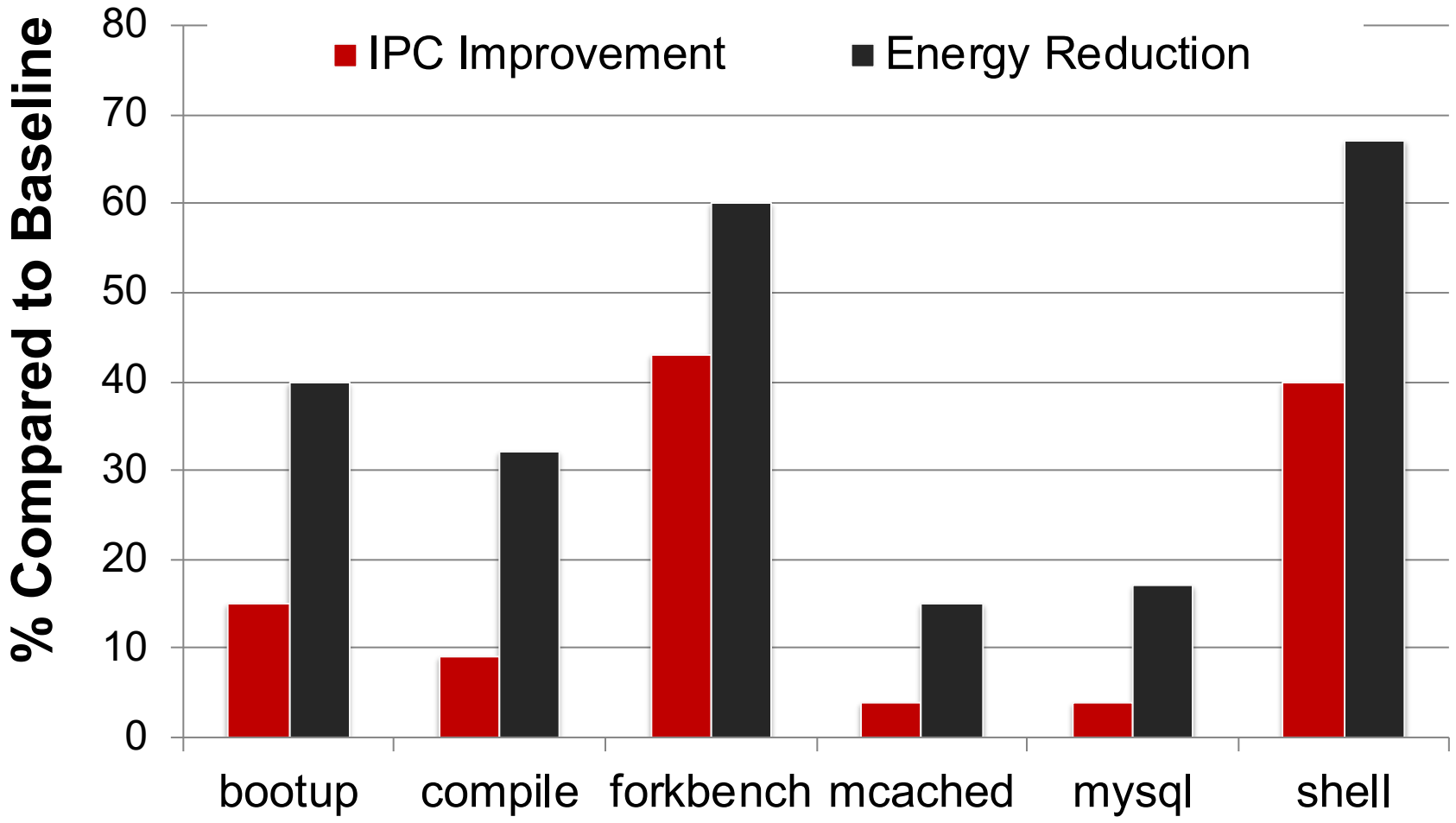


RowClone: Latency and Energy Savings

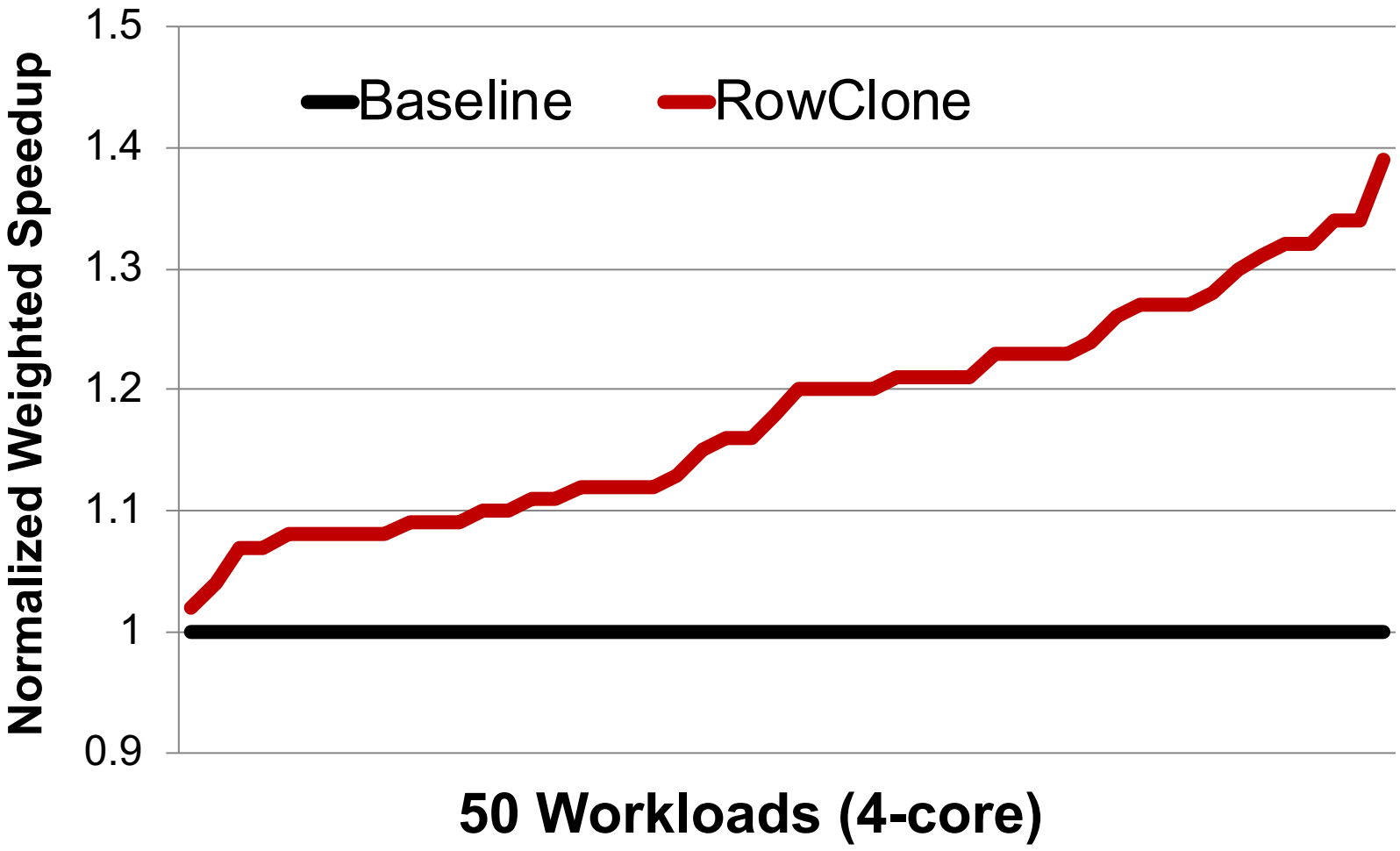


Seshadri et al., "RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data," MICRO 2013.

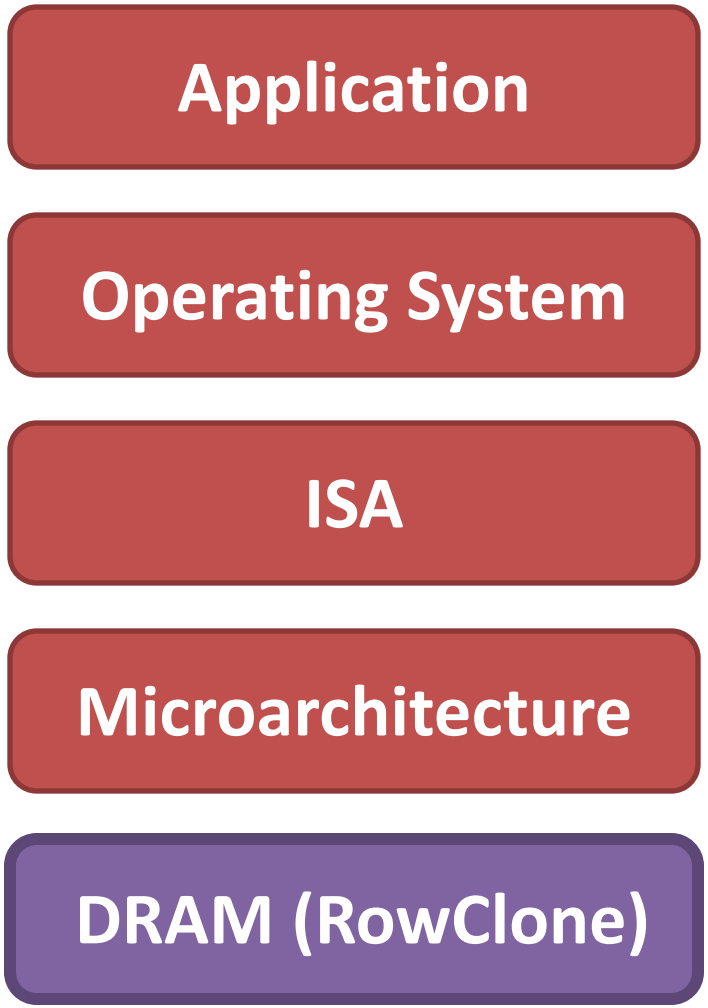
RowClone: Application Performance



RowClone: Multi-Core Performance



End-to-End System Design



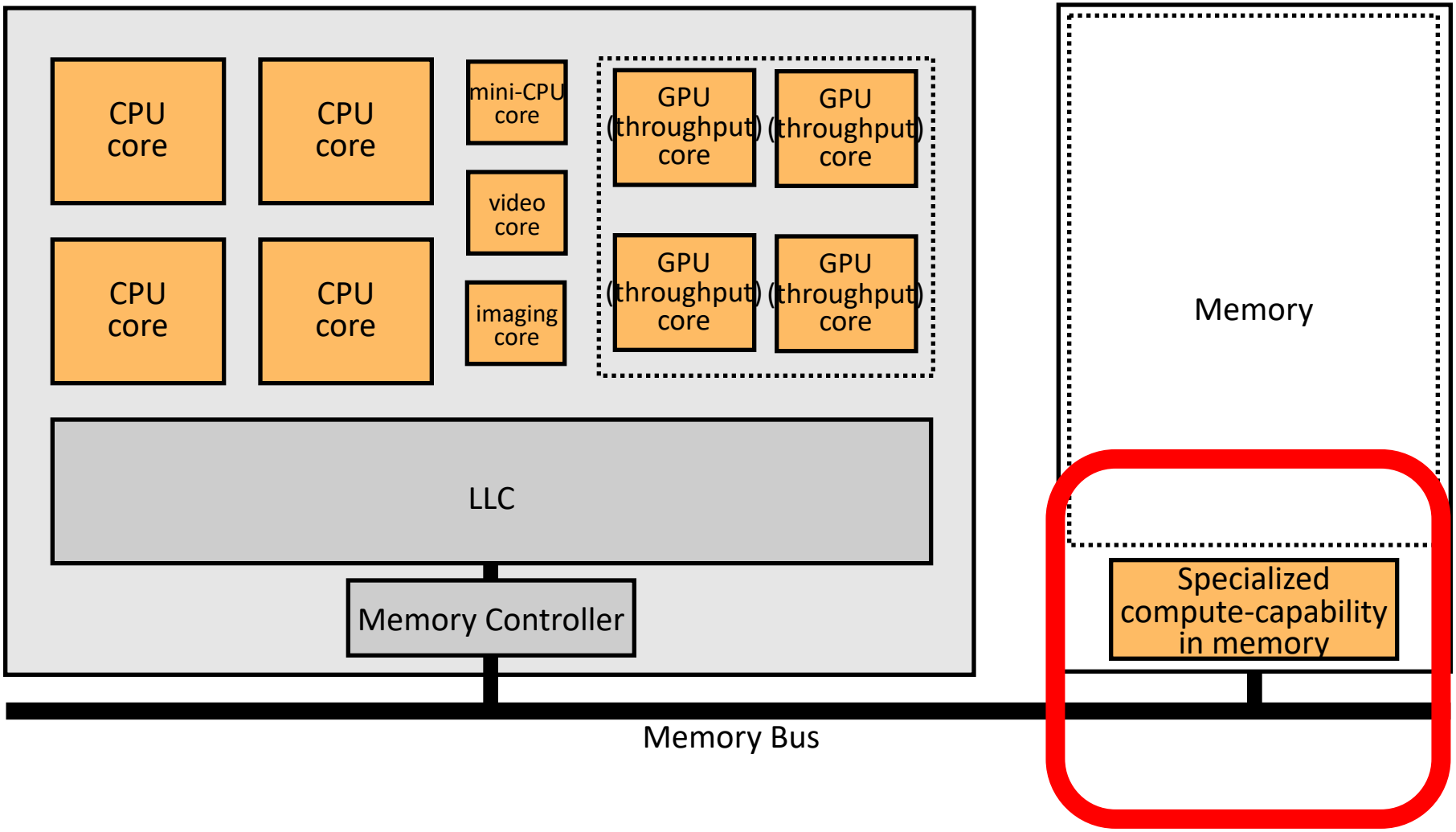
How to communicate occurrences of bulk copy/initialization across layers?

How to ensure cache coherence?

How to maximize latency and energy savings?

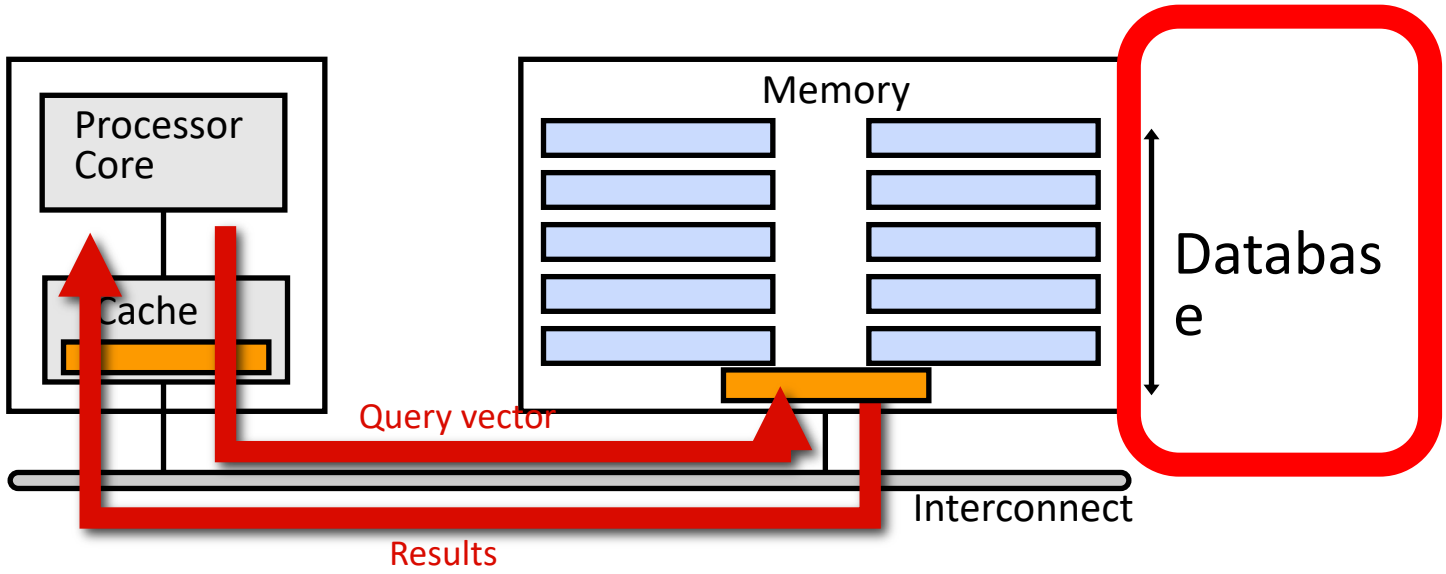
How to handle data reuse?

Goal: Ultra-Efficient Processing Near Data



Memory similar to a "conventional" accelerator

Enabling In-Memory Search

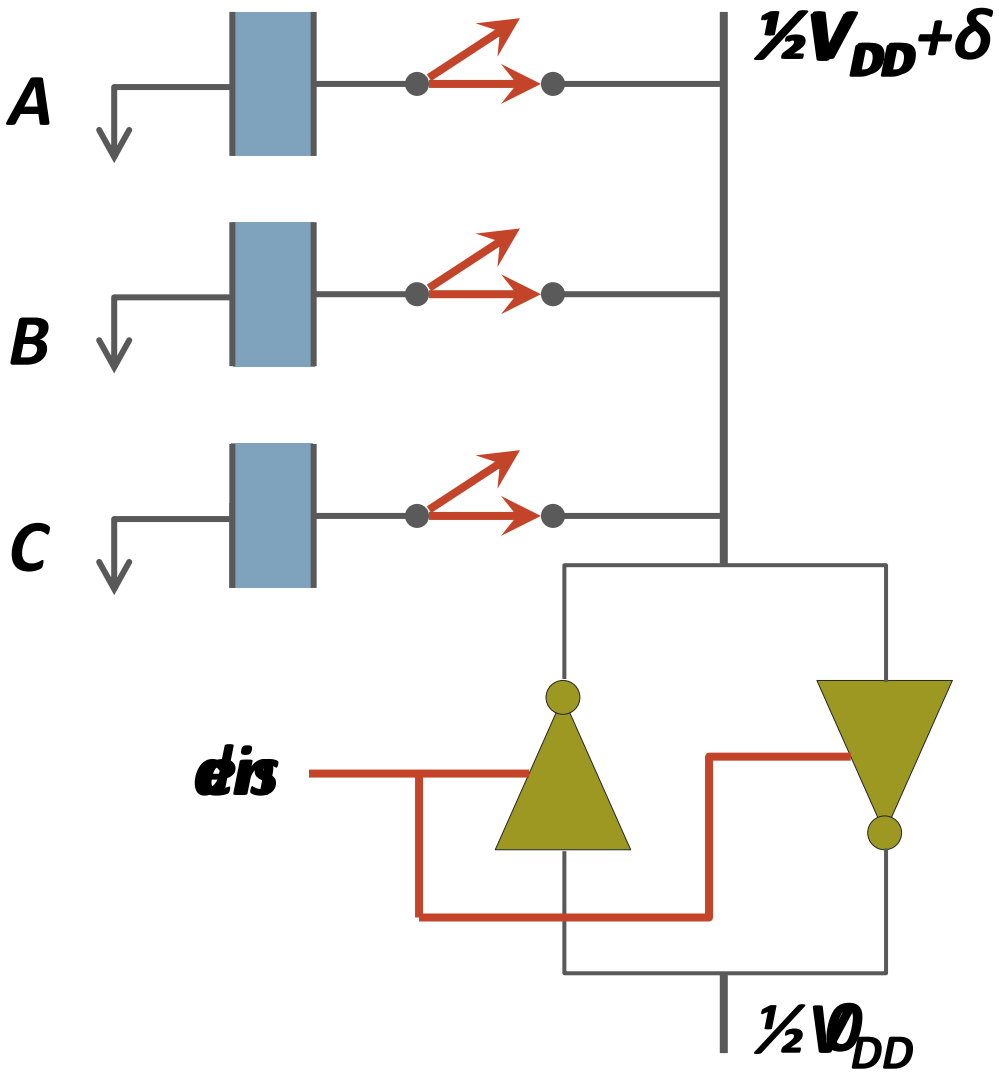


- What is a flexible and scalable memory interface?
- What is the right partitioning of computation capability?
- What is the right low-cost memory substrate?
- What memory technologies are the best enablers?
- How do we rethink/ease search

Enabling In-Memory Computation

DRAM Support	Cache Coherence	Virtual Memory Support
RowClone (MICRO 2013)	Dirty-Block Index (ISCA 2014)	Page Overlays (ISCA 2015)
In-DRAM Gather Scatter	Non-contiguous Cache lines	Gathered Pages
In-DRAM Bitwise Operations (IEEE CAL 2015)	?	?

In-DRAM AND/OR: Triple Row Activation



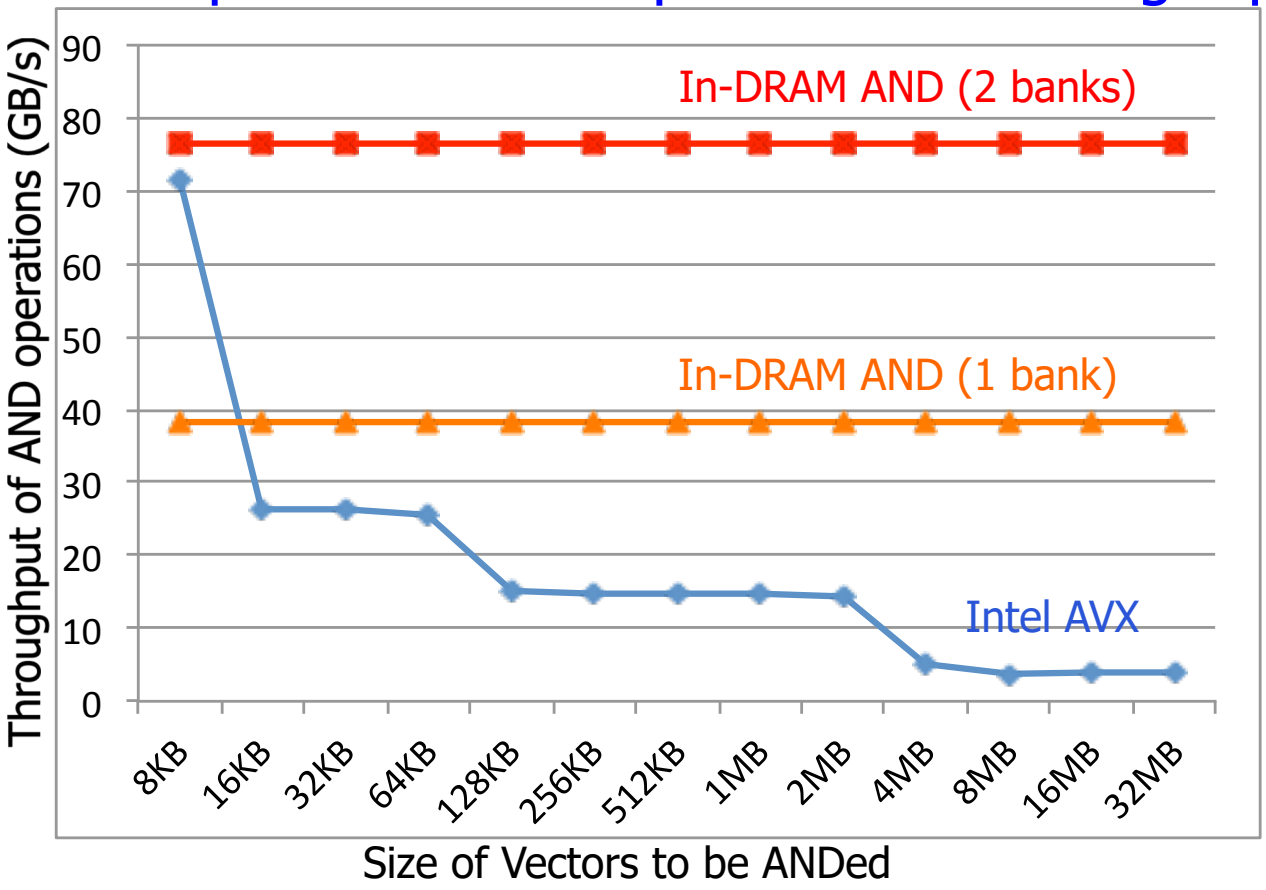
Final State
AB + BC + AC

C(A + B) + ~C(AB)

Seshadri+, "Fast Bulk Bitwise AND and OR in DRAM", IEEE CAL 2015.

In-DRAM AND/OR Results

- 20X improvement in AND/OR throughput vs. Intel AVX
- 50.5X reduction in memory energy consumption
- At least 30% performance improvement in range queries

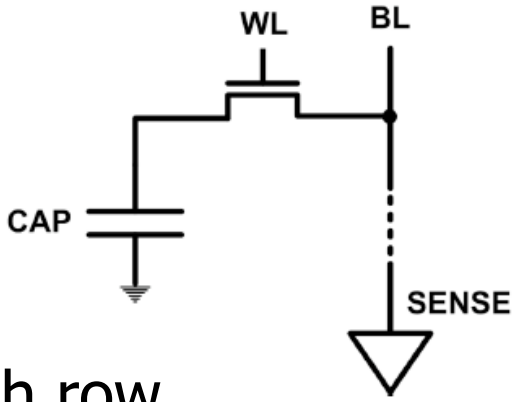


Seshadri+, "Fast Bulk Bitwise AND and OR in DRAM", IEEE CAL 2015.

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

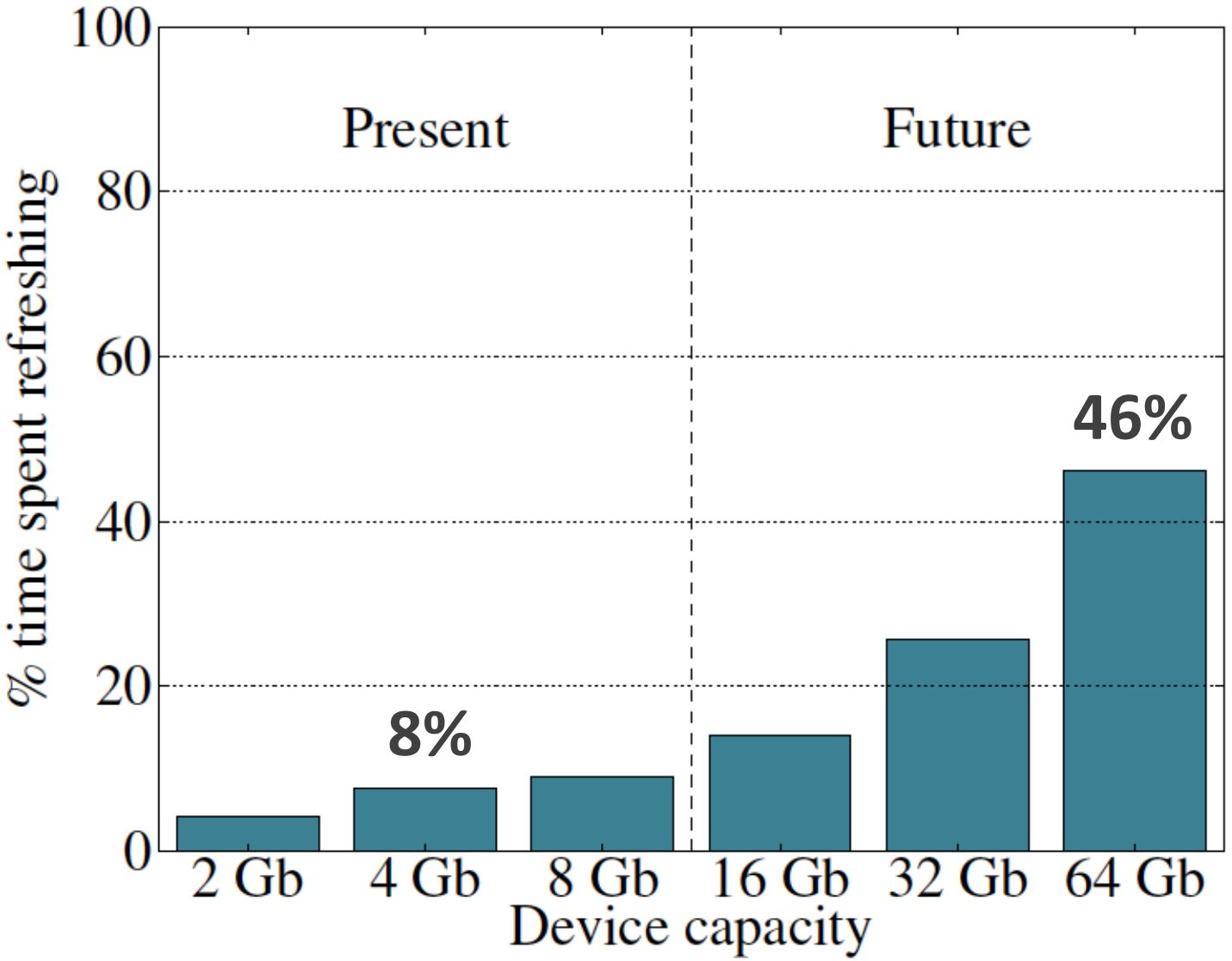
  -- Energy consumption: Each refresh consumes energy

  -- Performance degradation: DRAM rank/bank unavailable while refreshed

  -- QoS/predictability impact: (Long) pause times during refresh

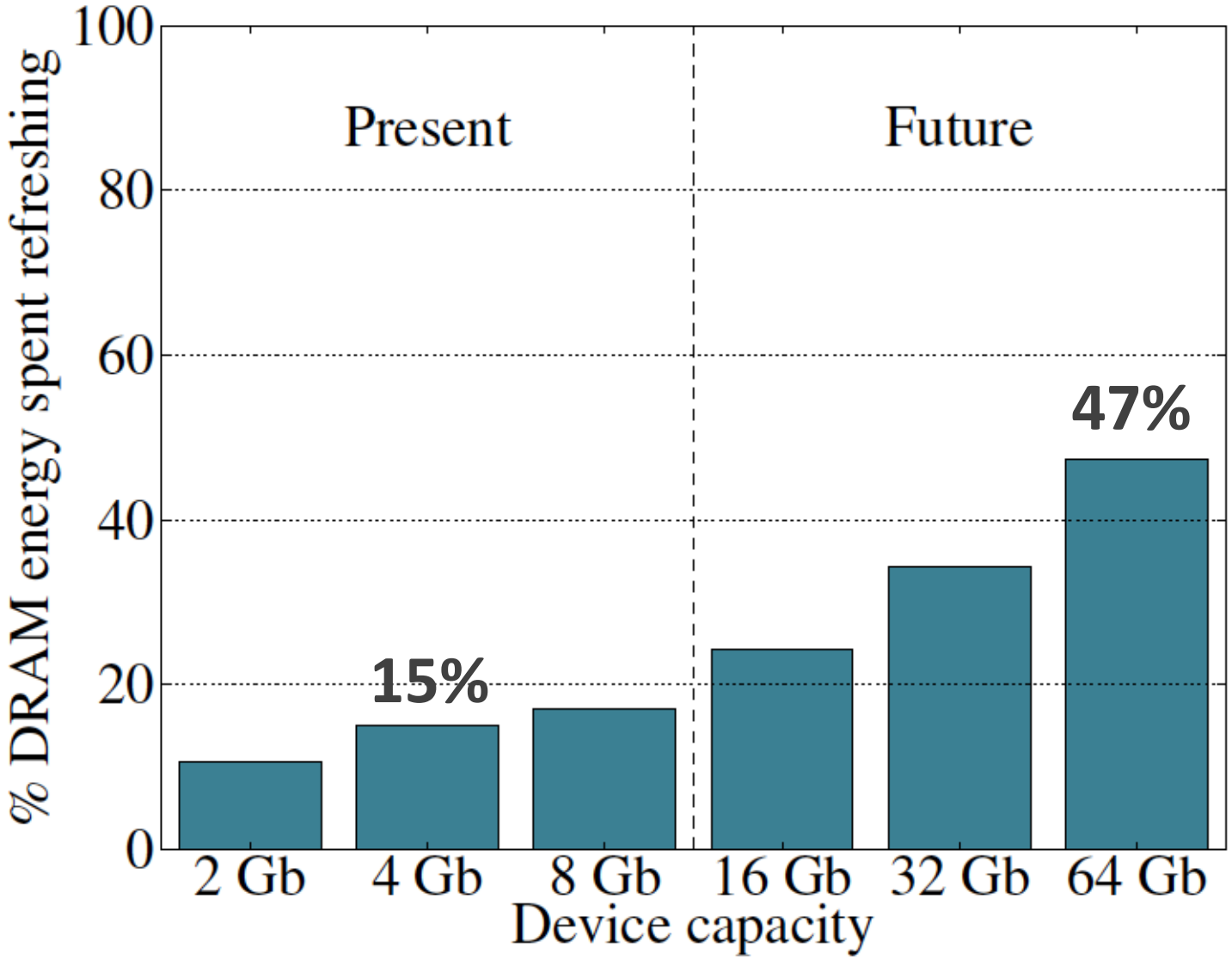
  -- Refresh rate limits DRAM capacity scaling

Refresh Overhead: Performance



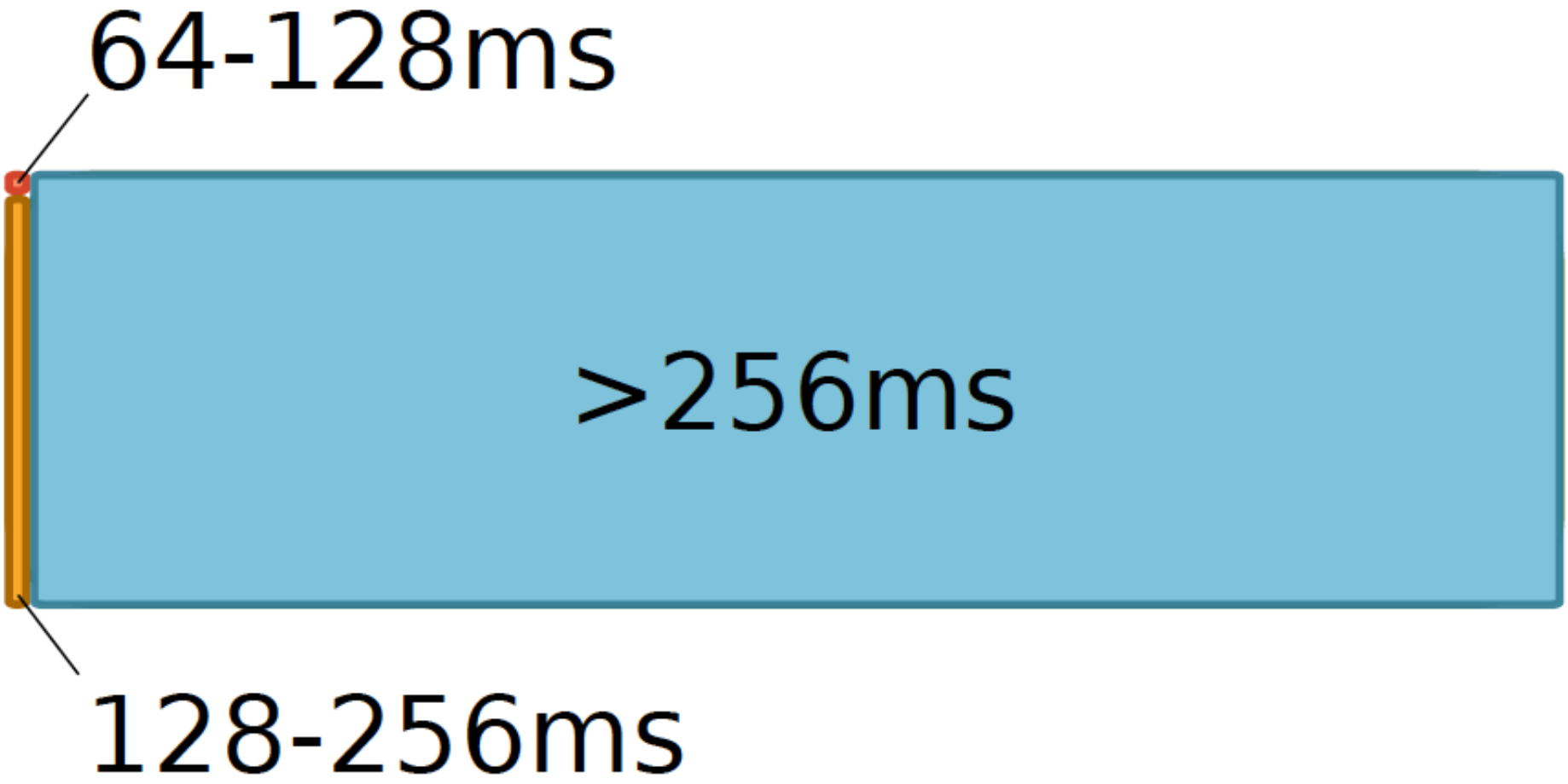
Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

Refresh Overhead: Energy



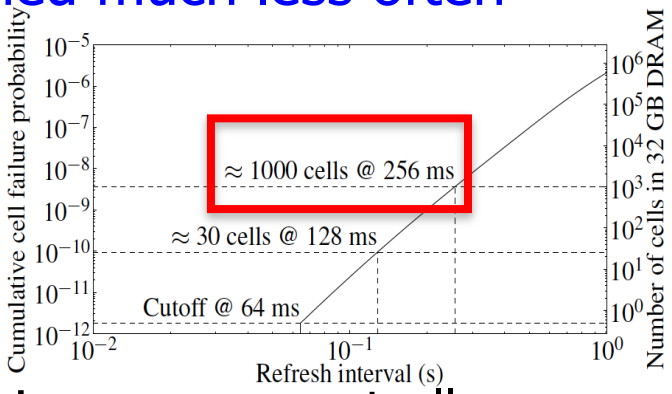
Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

Retention Time Profile of DRAM



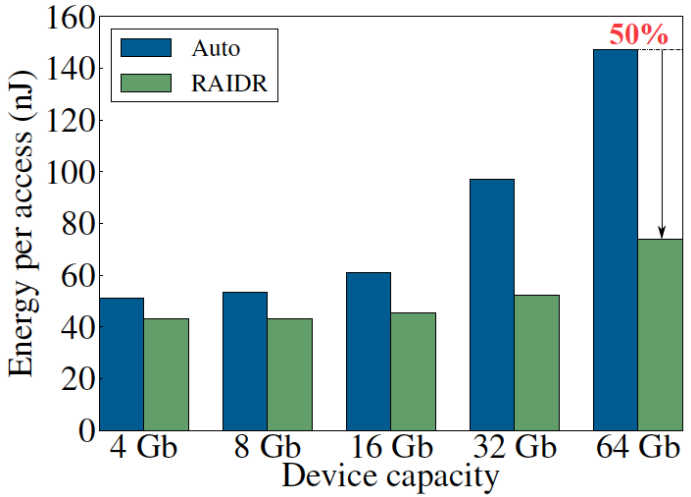
RAIDR: Eliminating Unnecessary Refreshes

- 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



Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

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 Infrastructure (DRAM)

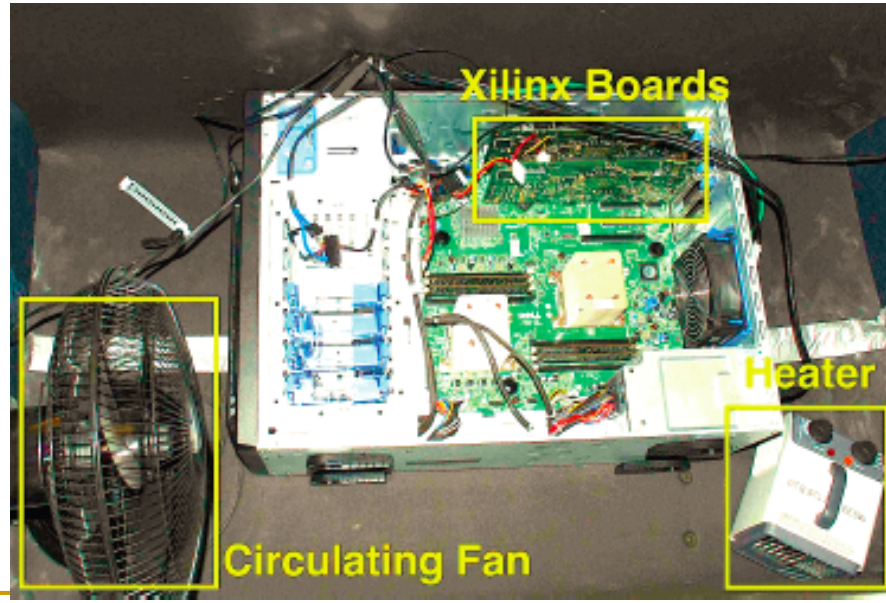
Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors", ISCA 2014.

Lee+, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," HPCA 2015.

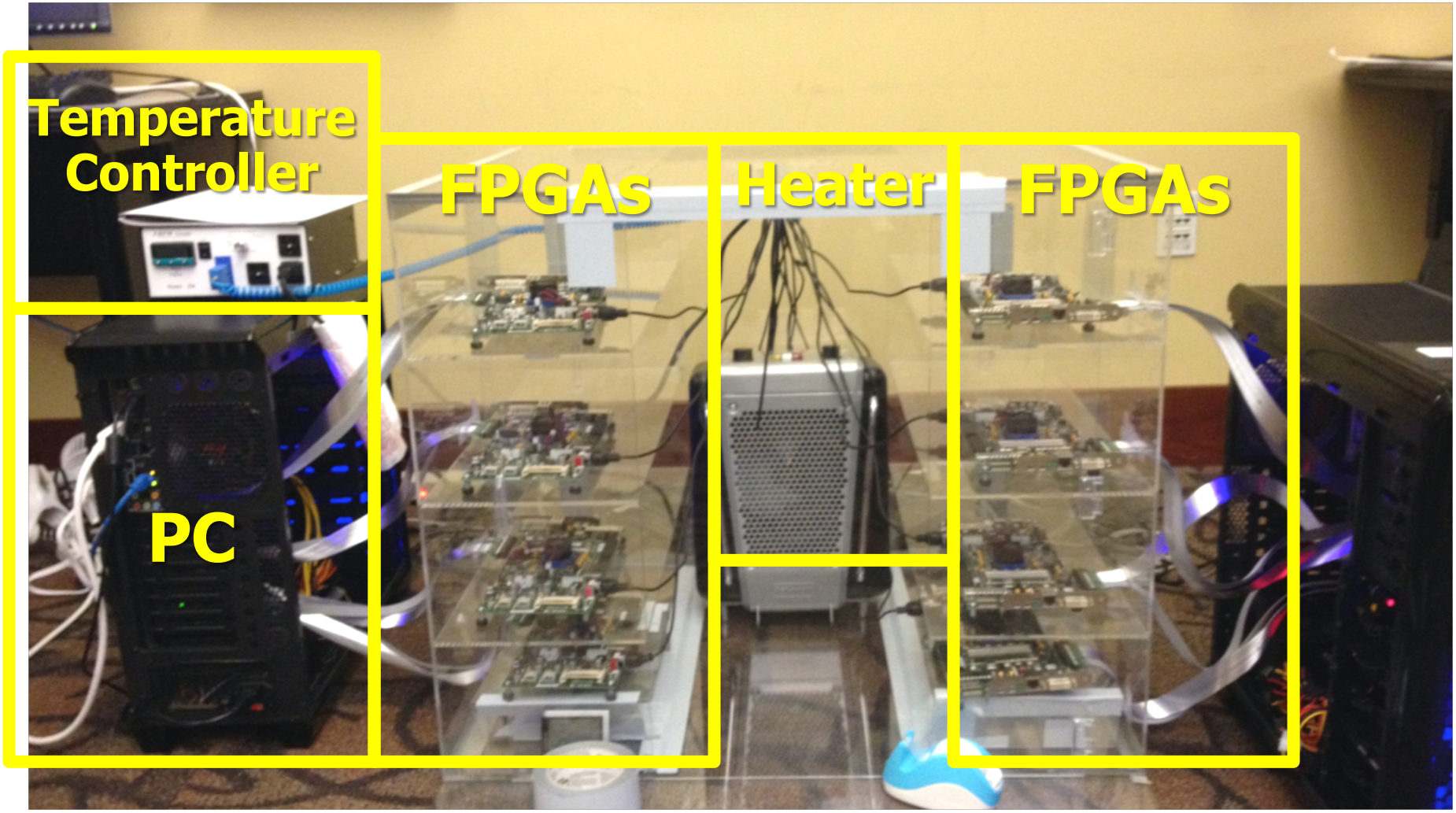
Qureshi+, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems," DSN 2015.

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.



Experimental Infrastructure (DRAM)



Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

More Information [ISCA'13, SIGMETRICS'14]

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

Samira Khan[†*]
samirakhan@cmu.edu

Donghyuk Lee[†]
donghyuk1@cmu.edu

Yoongu Kim[†]
yoongukim@cmu.edu

Alaa R. Alameldeen[*]
alaa.r.alameldeen@intel.com

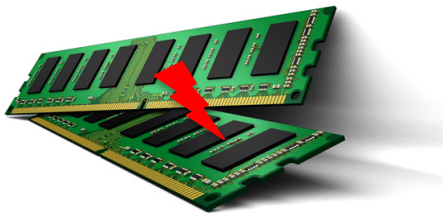
Chris Wilkerson[*]
chris.wilkerson@intel.com

Onur Mutlu[†]
onur@cmu.edu

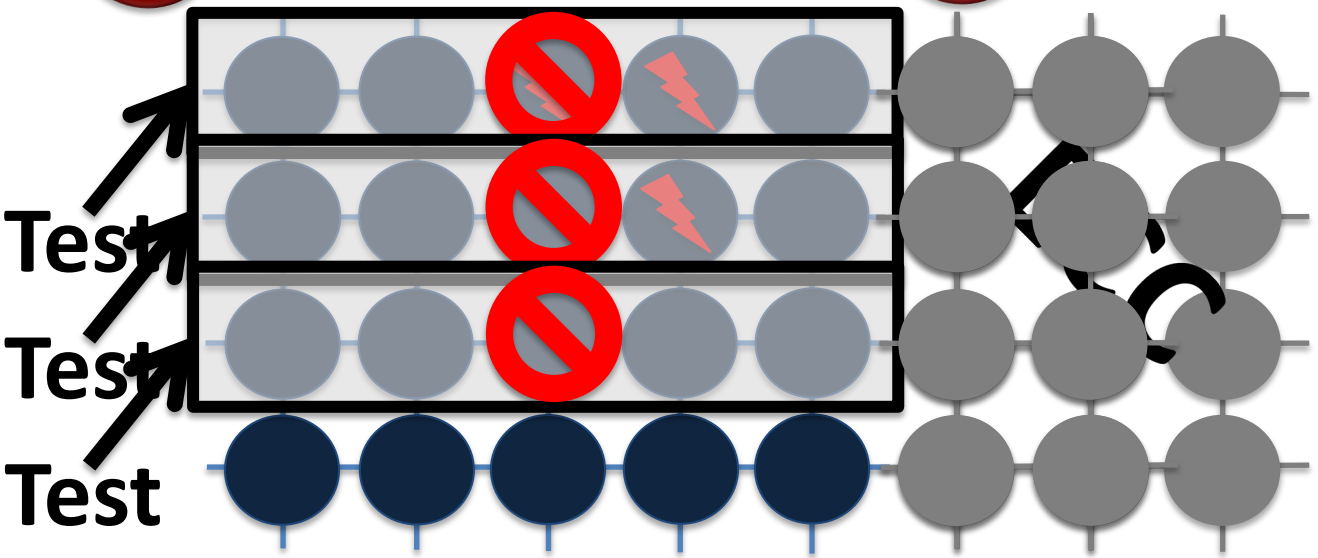
[†]Carnegie Mellon University [*]Intel Labs

Online Profiling of DRAM In the Field

Initially protect DRAM with ECC 1



Periodically test parts of DRAM 2



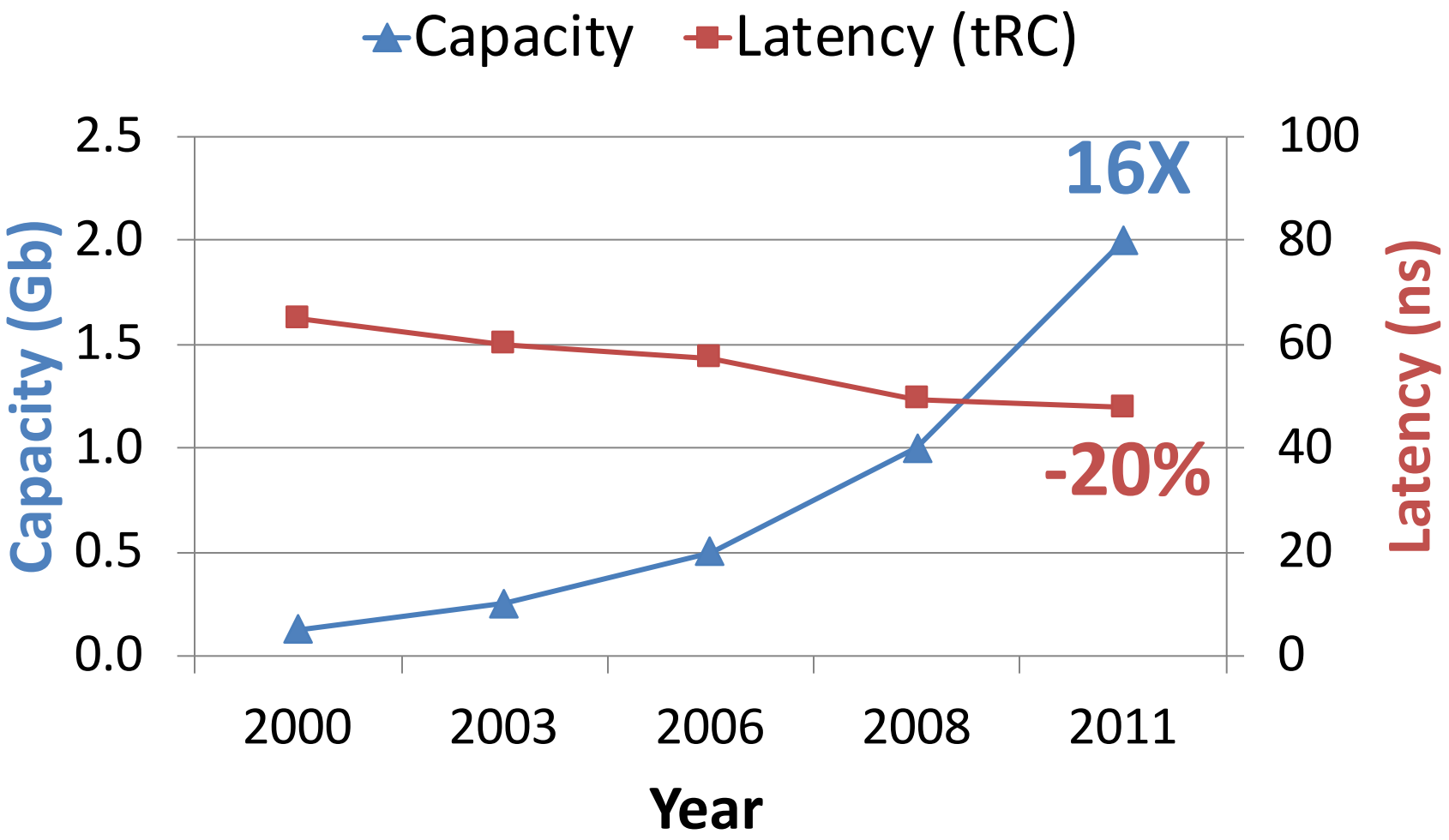
Adjust refresh rate and reduce ECC 3

Optimize DRAM and mitigate errors online without disturbing the system and applications

Rethinking DRAM

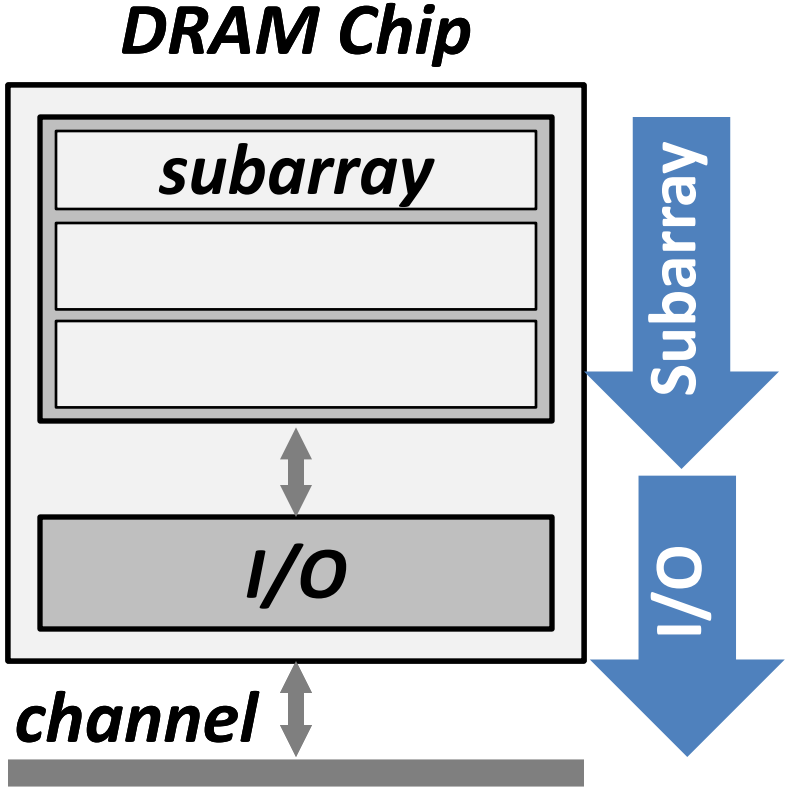
- In-Memory Computation
- Refresh
- Reliability
- Latency
- Bandwidth
- Energy
- Memory Compression

DRAM Latency-Capacity Trend



DRAM latency continues to be a critical bottleneck, especially for response time-sensitive

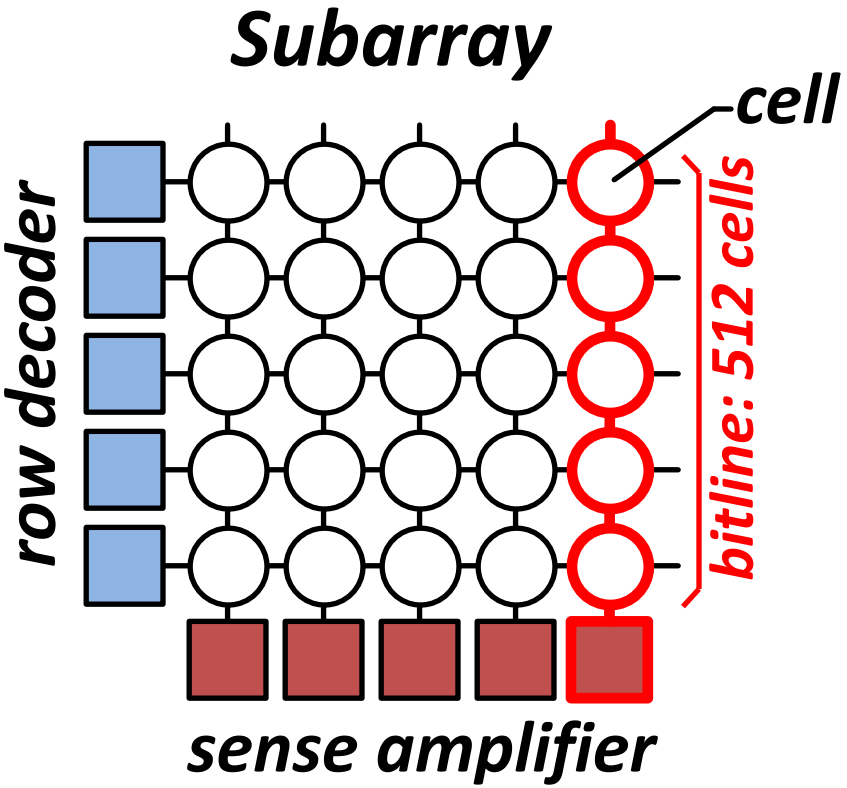
What Causes the Long Latency?

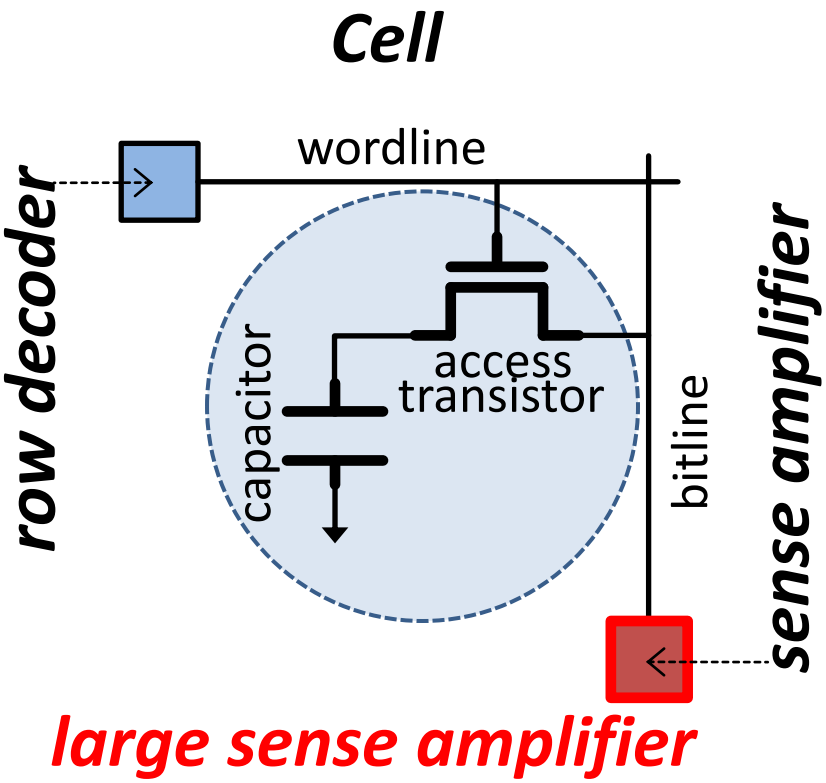


DRAM Latency = Subarray Latency + I/O Latency

Dominant

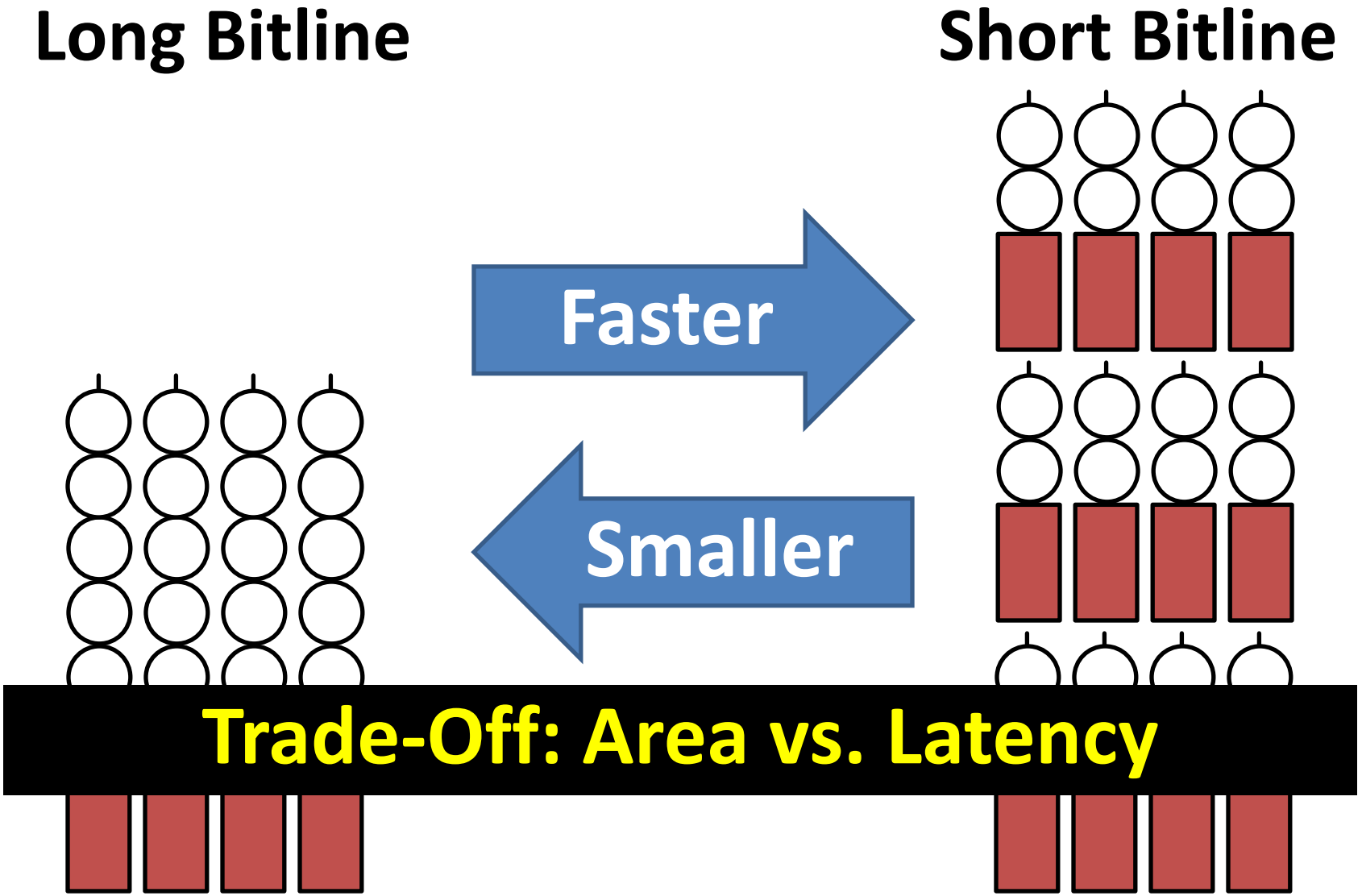
Why is the Subarray So Slow?



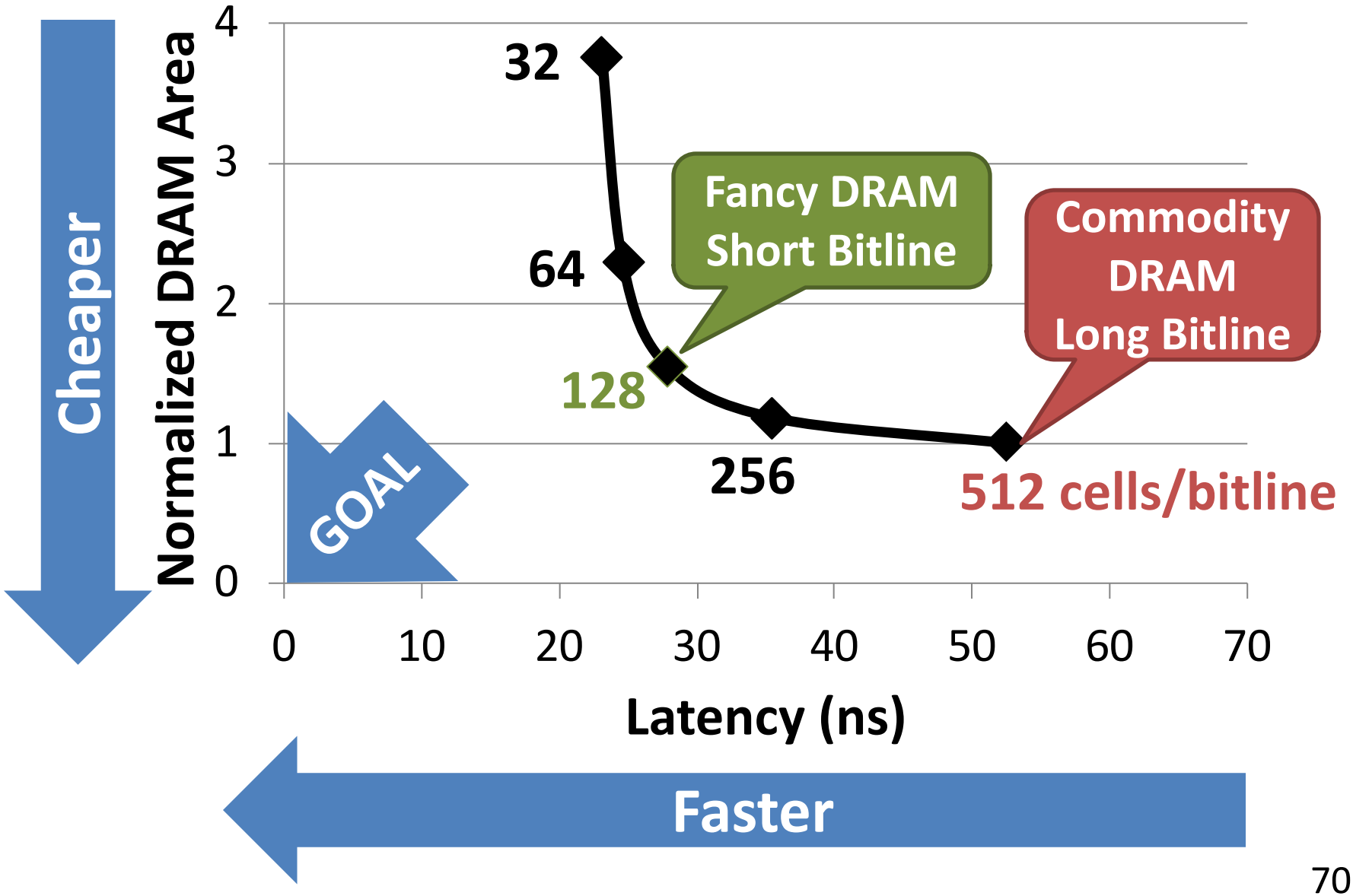


- **Long bitline**
  - **Amortizes sense amplifier cost → Small area**
  - **Large bitline capacitance → High latency & power**

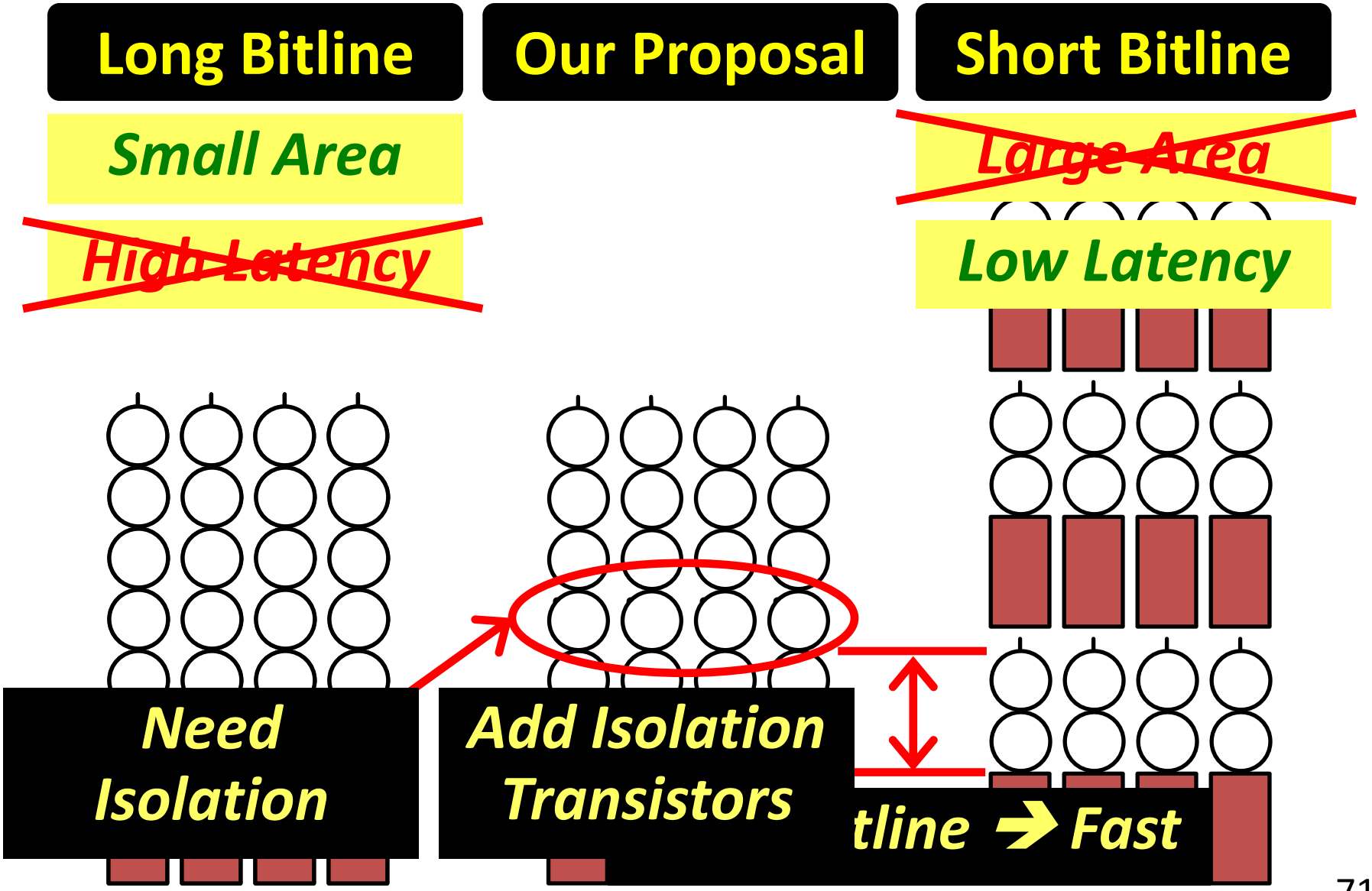
Trade-Off: Area (Die Size) vs. Latency



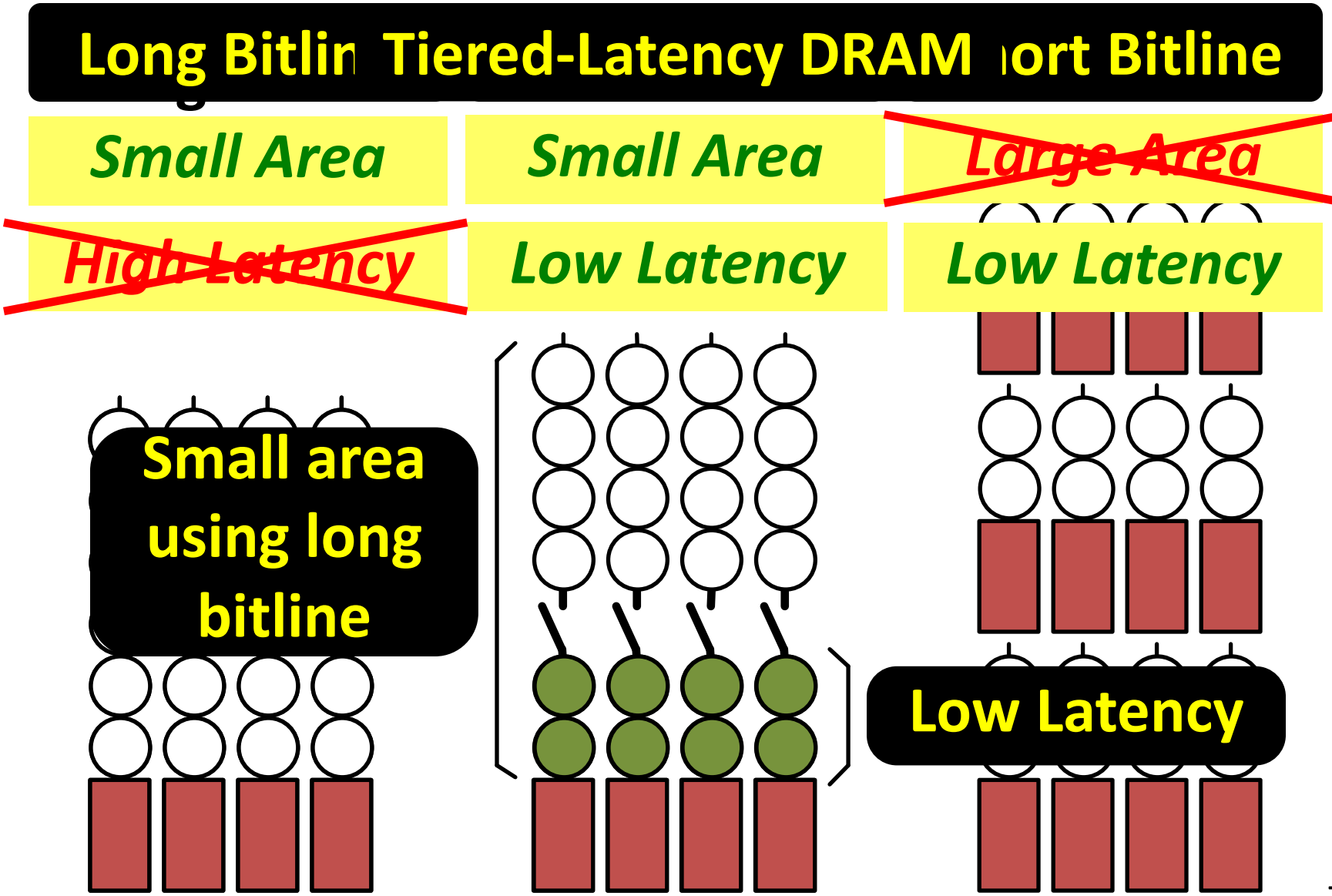
Trade-Off: Area (Die Size) vs. Latency



Approximating the Best of Both Worlds

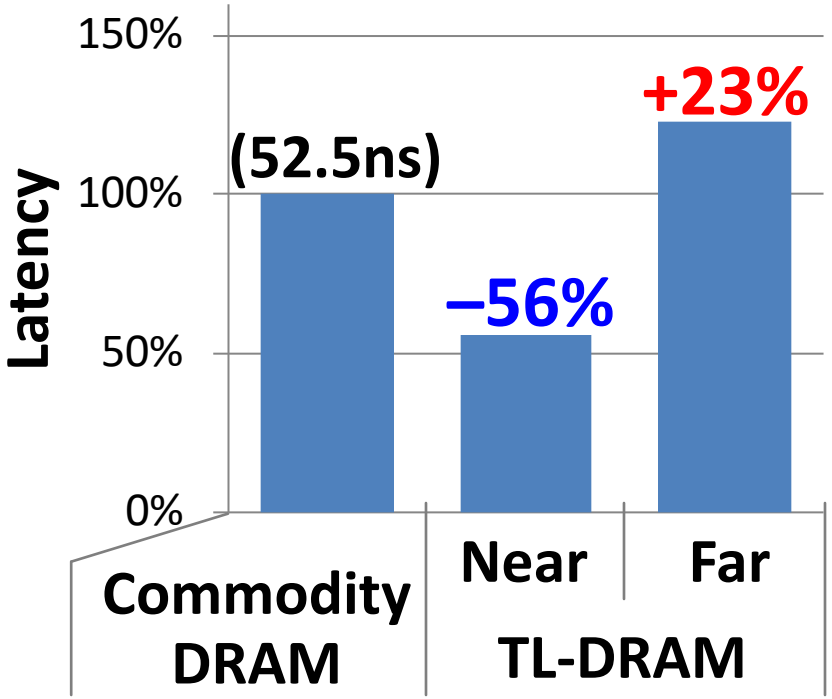


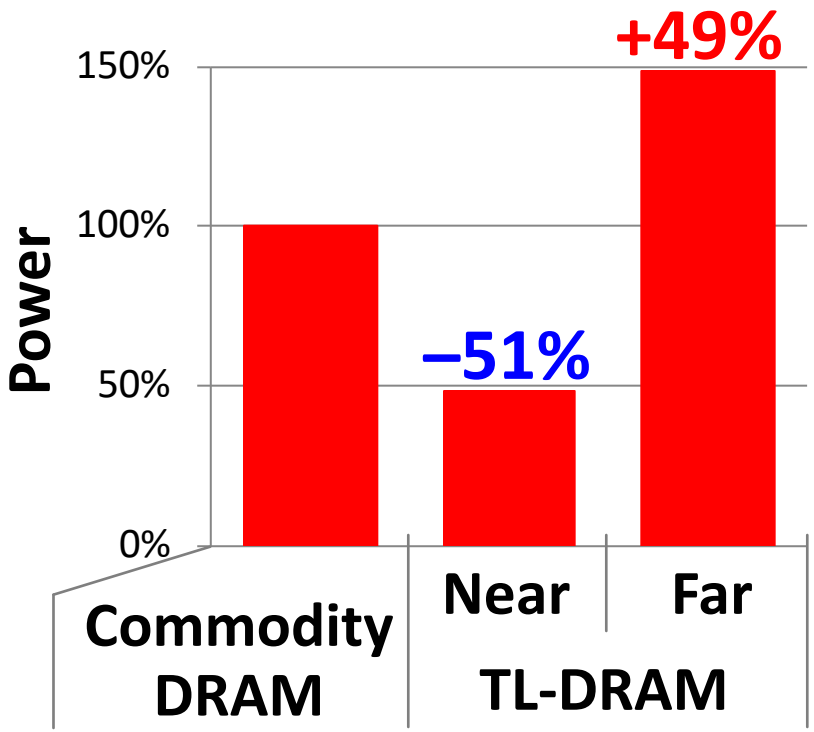
Approximating the Best of Both Worlds



Commodity DRAM vs. TL-DRAM [HPCA 2013]

- **DRAM Latency** (tRC)
- **DRAM Power**

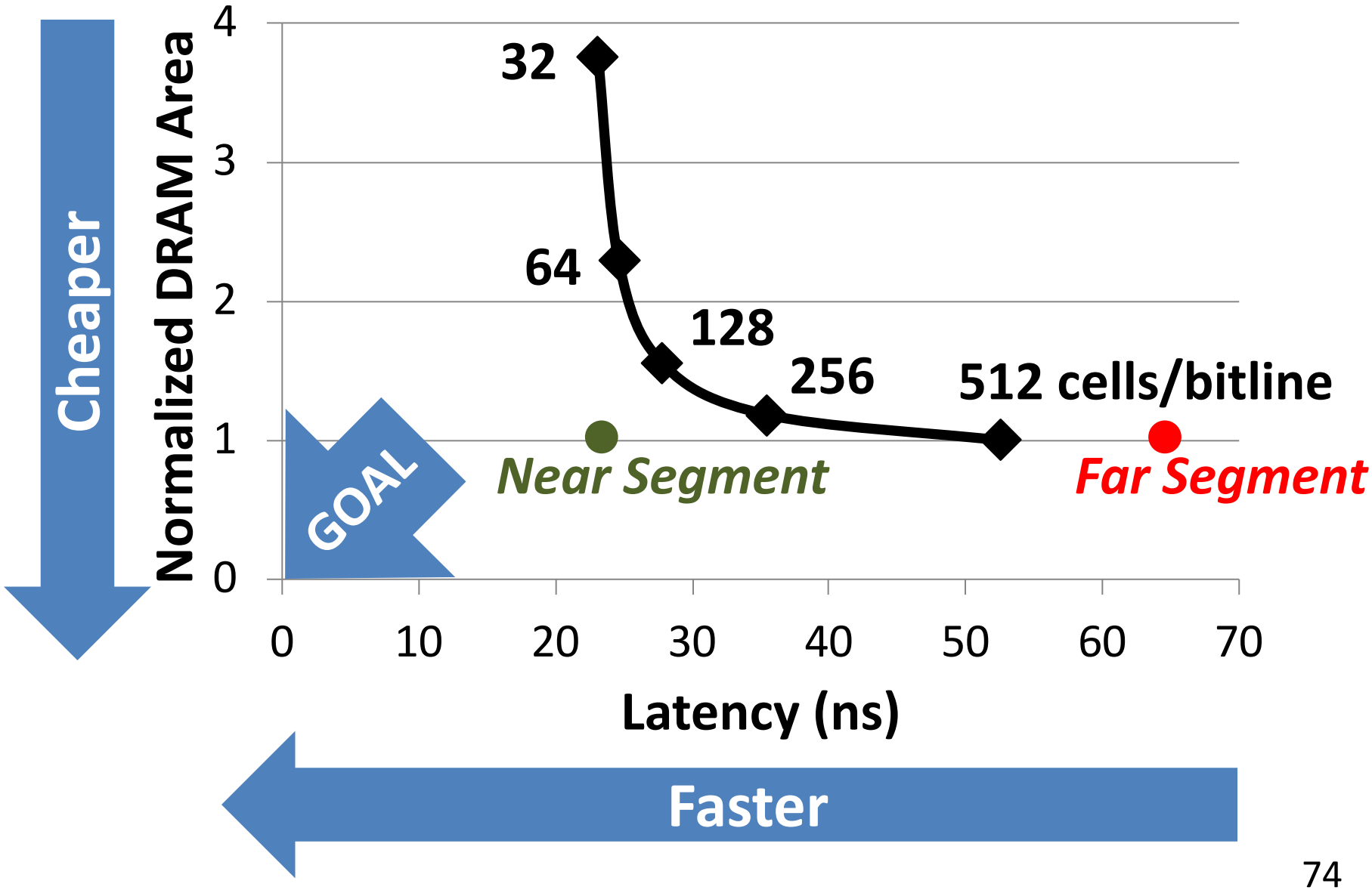




- **DRAM Area Overhead**

  ~3%: mainly due to the isolation transistors

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



Leveraging Tiered-Latency DRAM

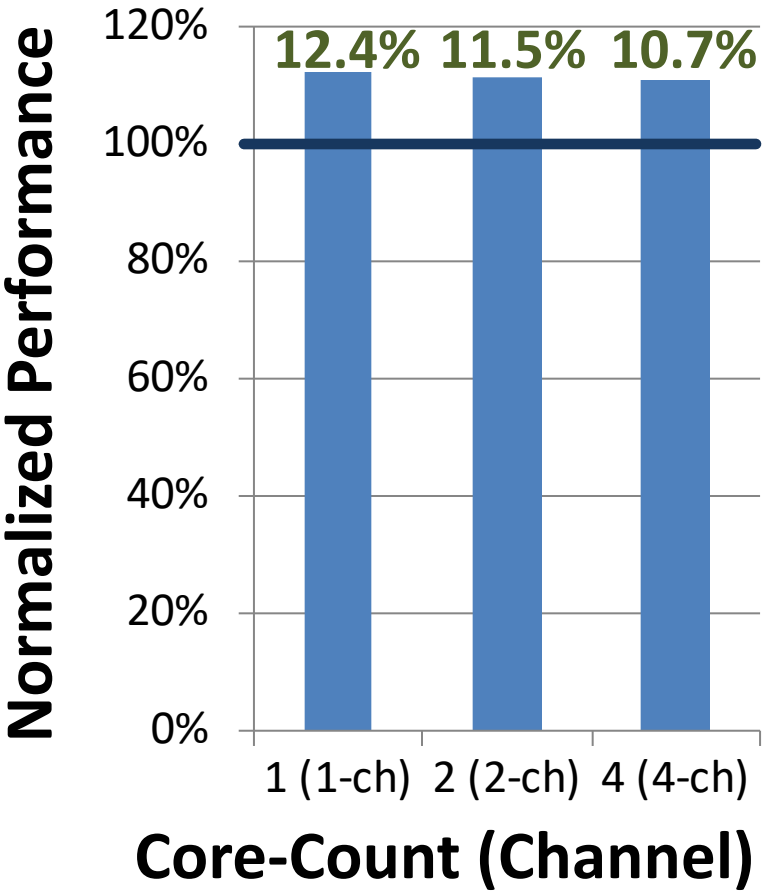
- TL-DRAM is a ***substrate*** that can be leveraged by the hardware and/or software

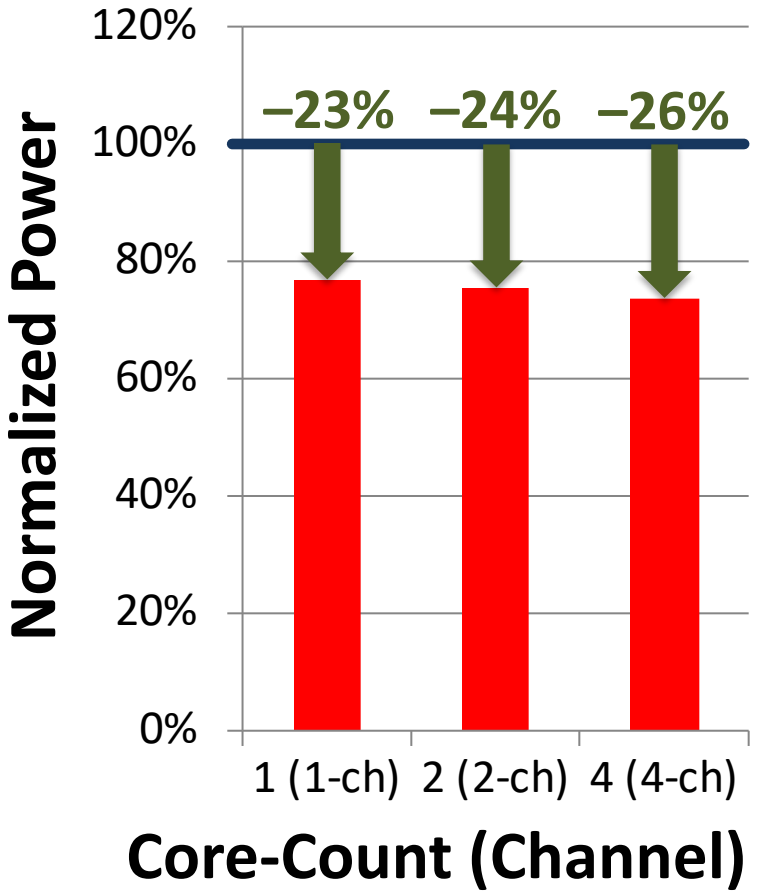
- Many potential uses

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

Lee+, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013.

Performance & Power Consumption





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

Lee+, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013.

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

Lee+, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," HPCA 2015.

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%***

Lee+, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," HPCA 2015.

AL-DRAM: Real System Evaluation

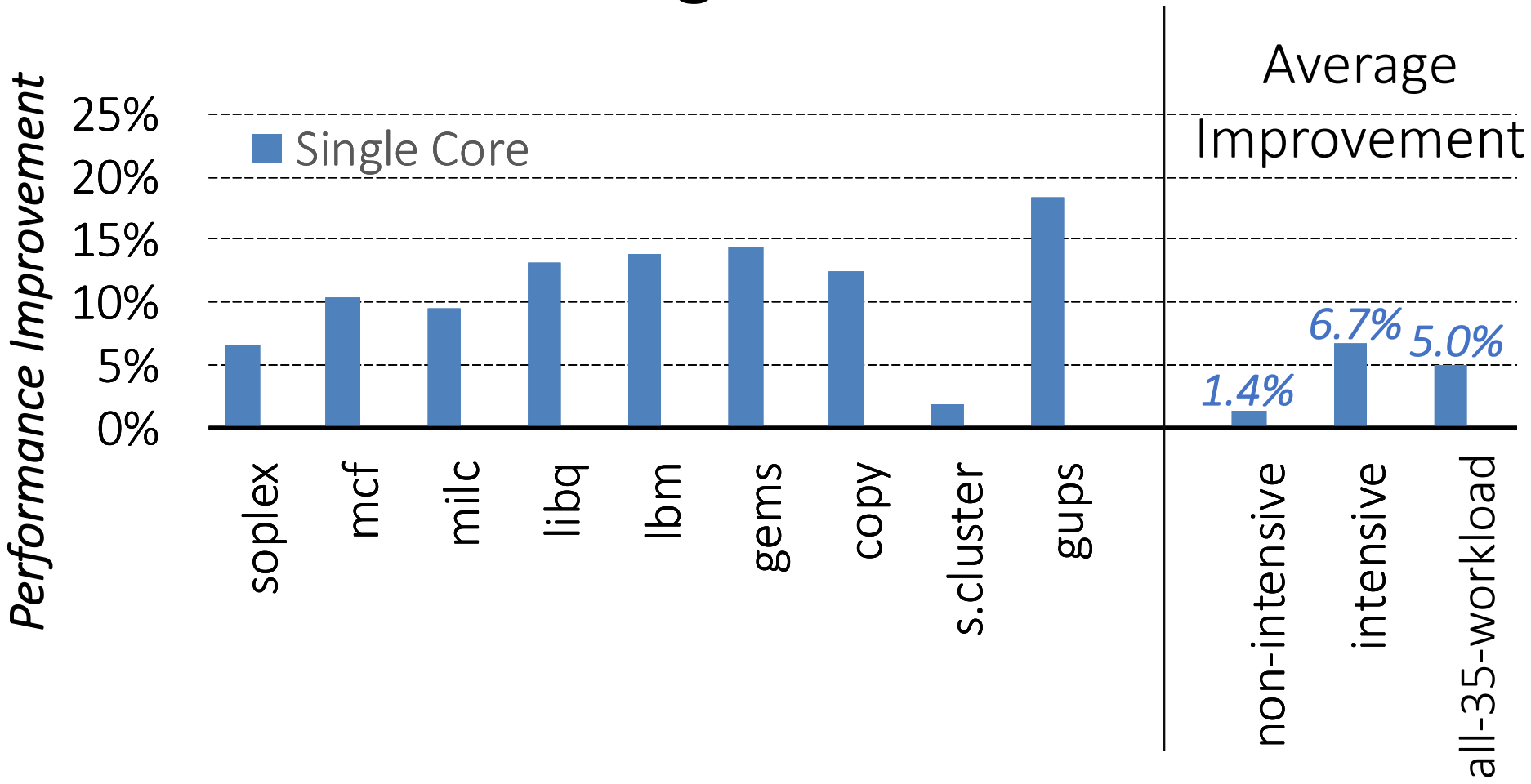
- *System*
  - *CPU: AMD 4386 (8 Cores, 3.1GHz, 8MB LLC)*

D18F2x200_dct[0]_mp[1:0] DDR3 DRAM Timing 0

Reset: 0F05_0505h. See 2.9.3 [DCT Configuration Registers].

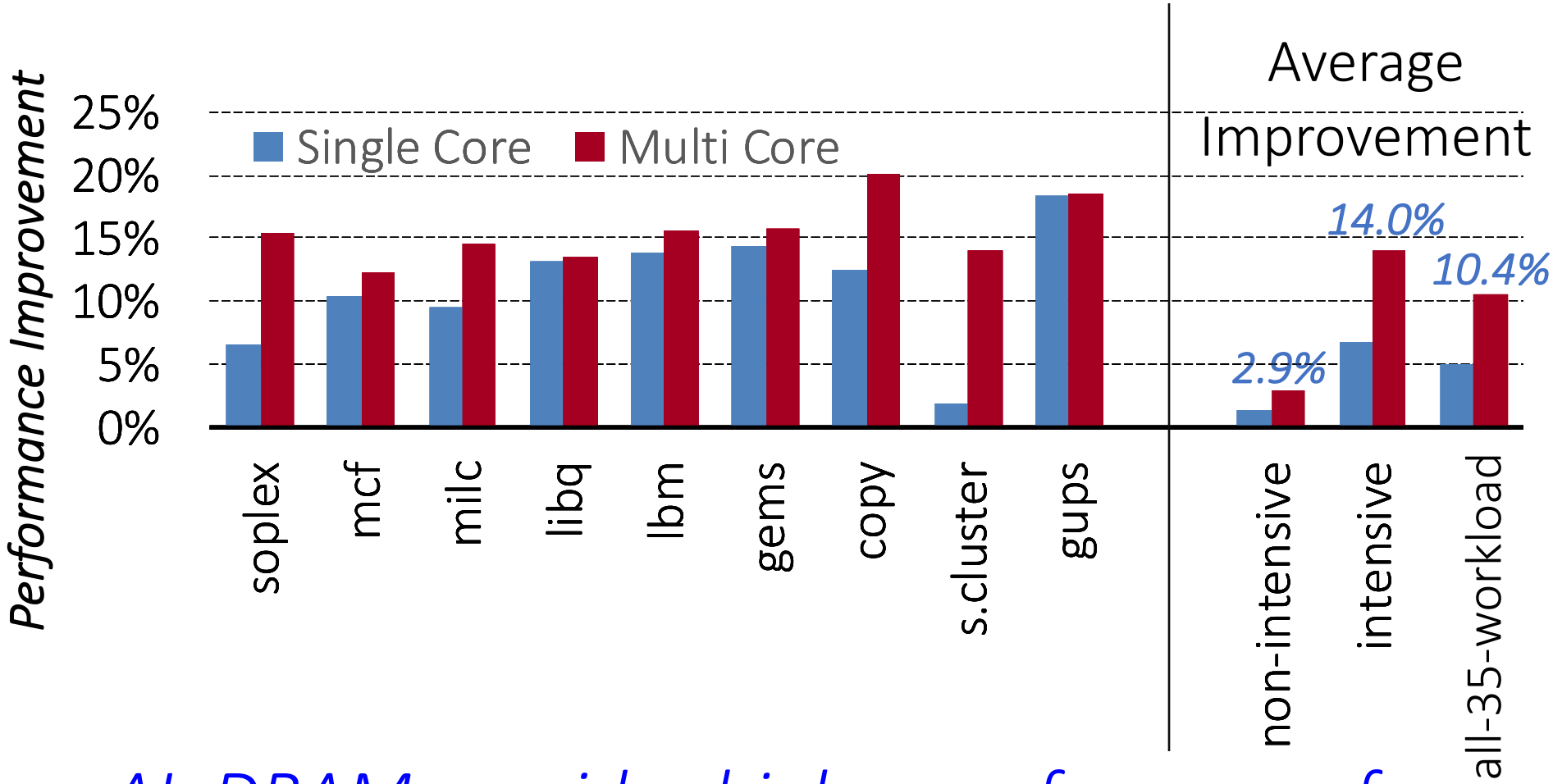
Bits	Description
31:30	Reserved.
29:24	**Tras: row active strobe**. Read-write. BIOS: See 2.9.7.5 [SPD ROM-Based Configuration]. Specifies the minimum time in memory clock cycles from an activate command to a precharge command, both to the same chip select bank. Bits — Description 07h-00h — Reserved 2Ah-08h — <Tras> clocks 3Fh-2Bh — Reserved
23:21	Reserved.
20:16	**Trp: row precharge time**. 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

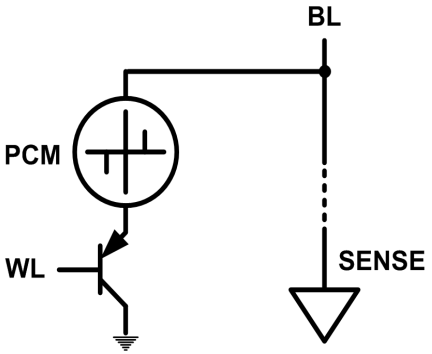
Agenda

- 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

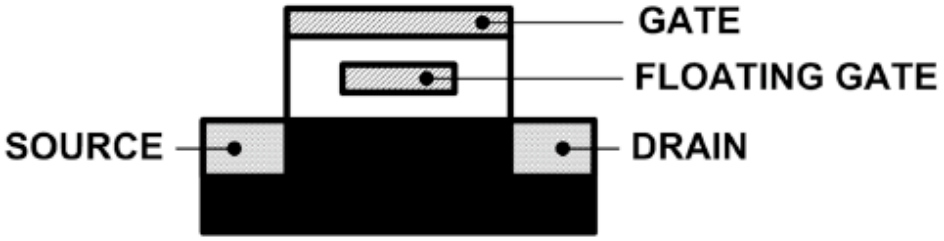


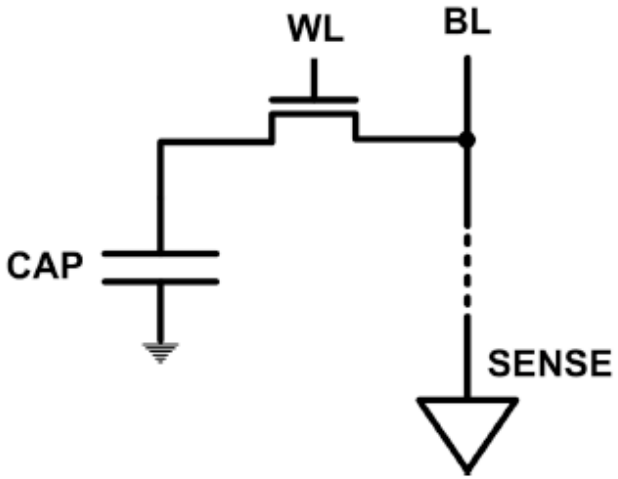
- But, emerging technologies have (many) shortcomings
  - Can they be enabled to replace/augment/surpass DRAM?

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

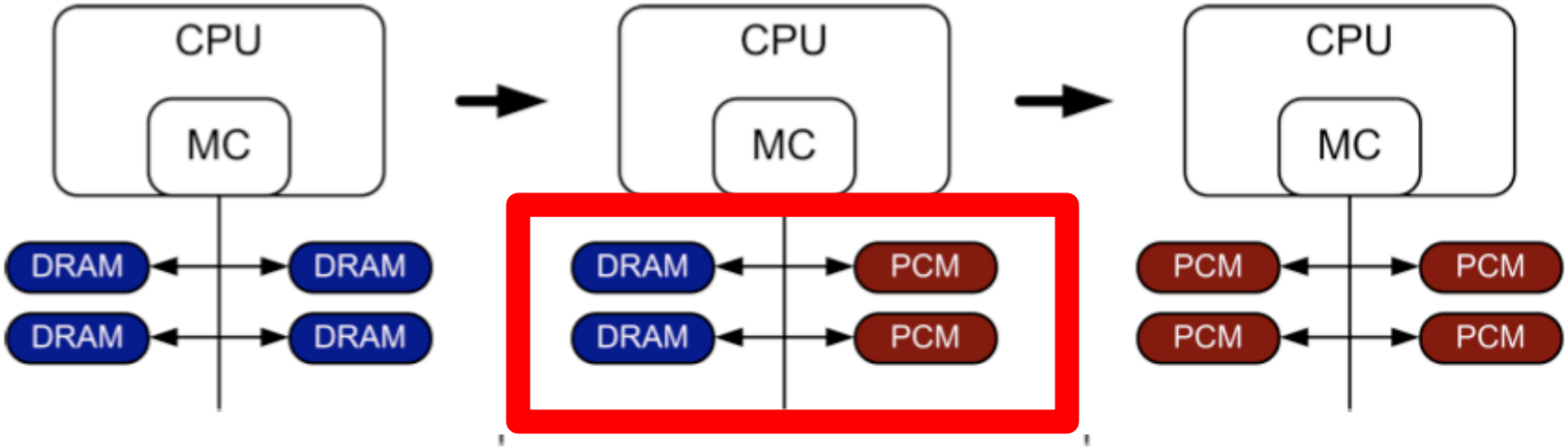
- PCM
  - Inject current to change material phase
  - Resistance determined by phase
- STT-MRAM
  - Inject current to change magnet polarity
  - Resistance determined by polarity
- Memristors/RRAM/ReRAM
  - Inject current to change atomic structure
  - Resistance determined by atom distance

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: ~4-15x DRAM (especially write)
  - Higher active energy: ~2-50x DRAM (especially write)
  - Lower endurance (a cell dies after ~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 (I)

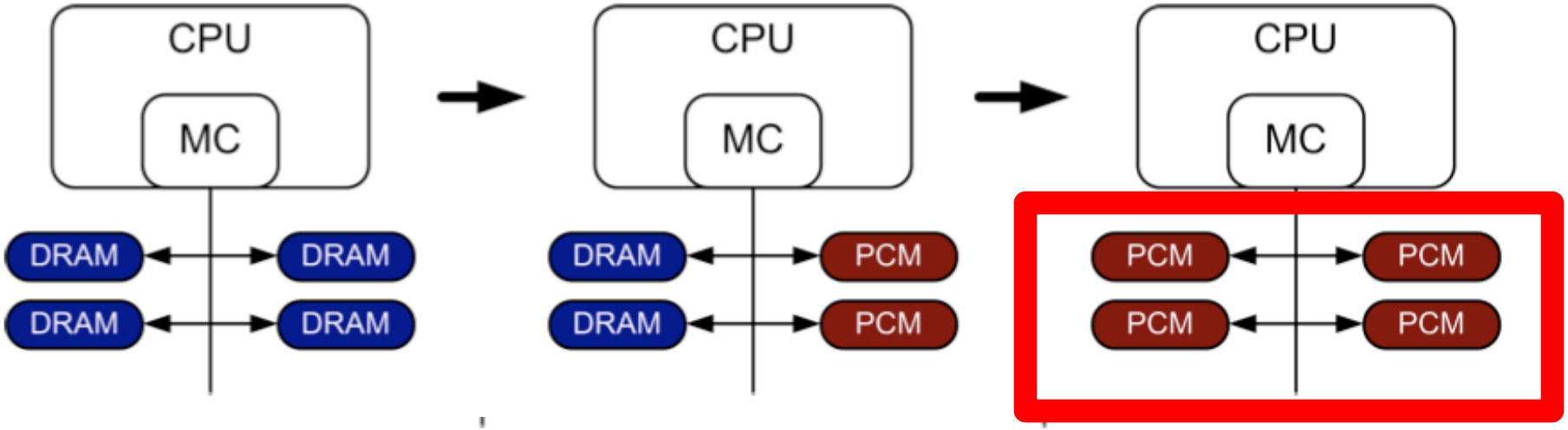
- How should PCM-based (main) memory be organized?



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

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^2 using BJT
- 1.5× DRAM

Latency

- 50ns Rd, 150ns Wr
- 4×, 12× DRAM

Endurance

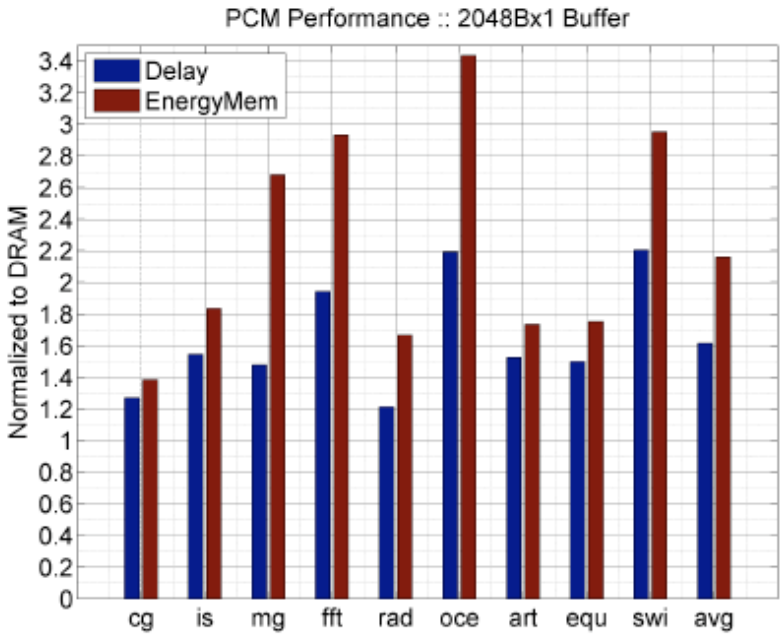
- 1E+08 writes
- 1E-08× DRAM

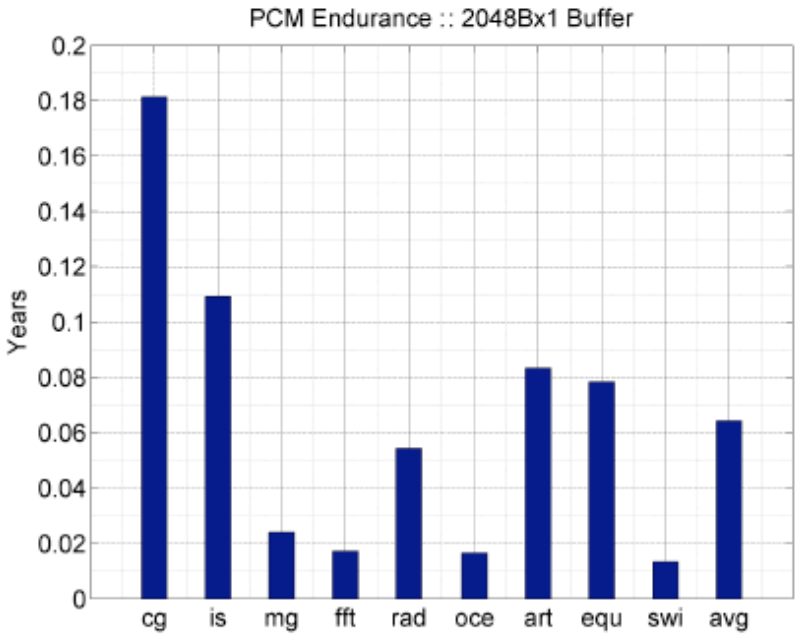
Energy

- 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

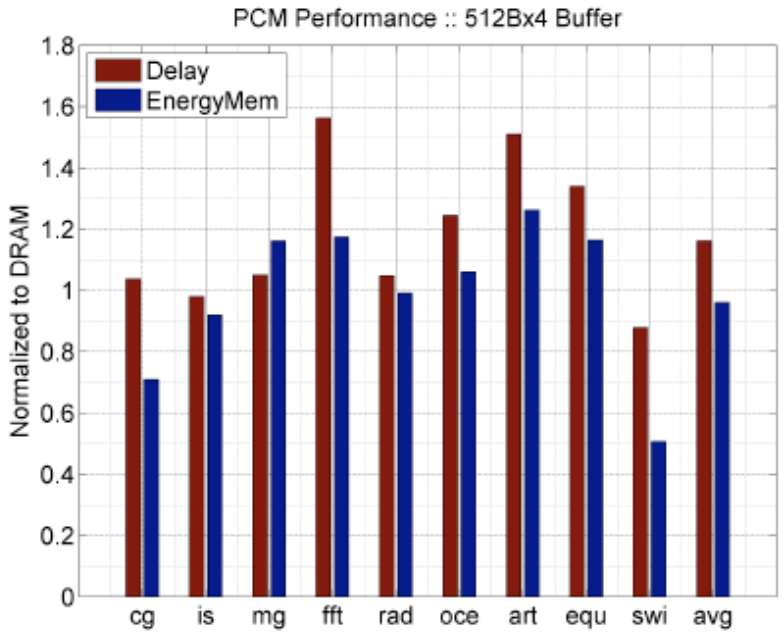


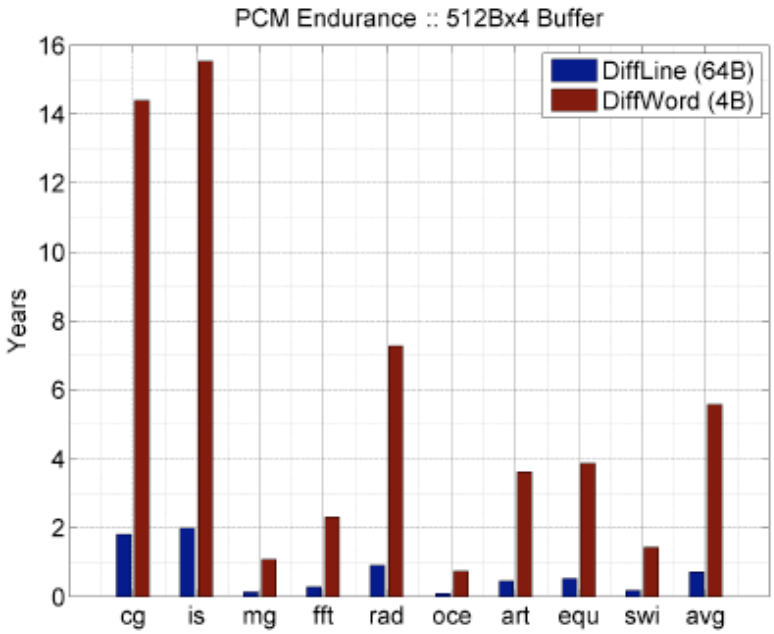


- Lee, Ipek, Mutlu, Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," ISCA 2009.

Results: Architected PCM as Main Memory

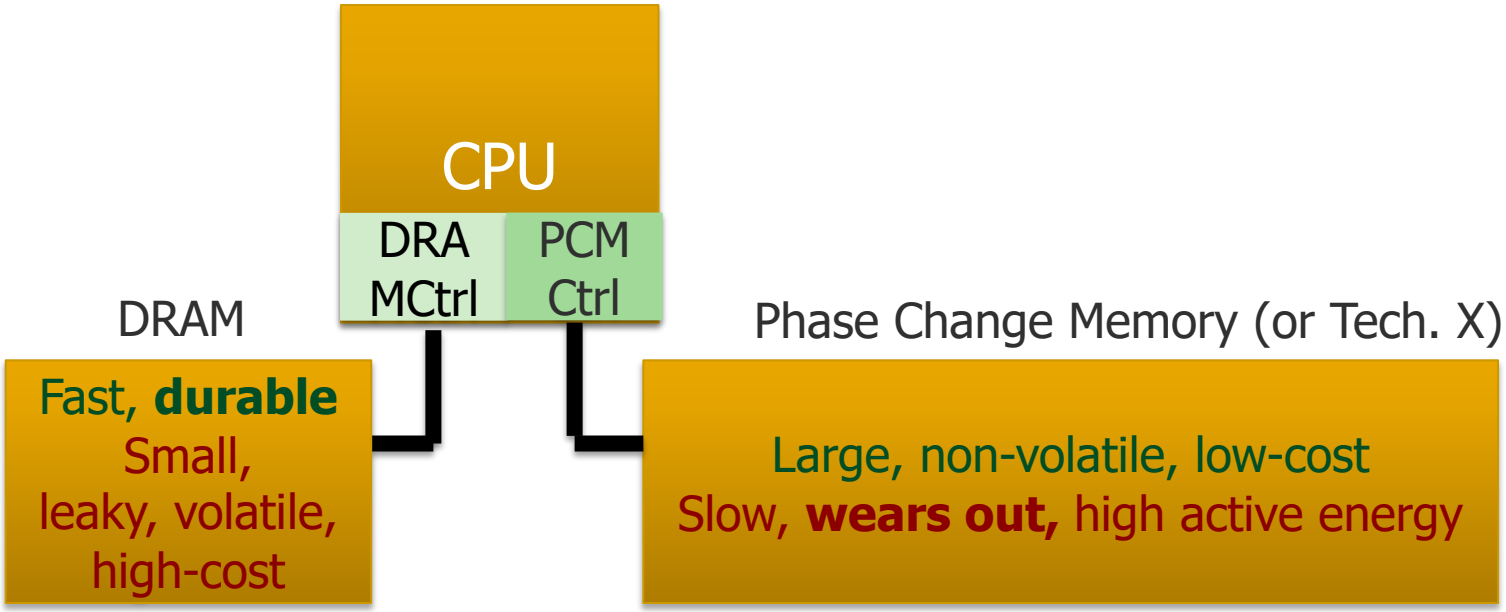
- 1.2x delay, 1.0x energy, 5.6-year average lifetime
- Scaling improves energy, endurance, density





- 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

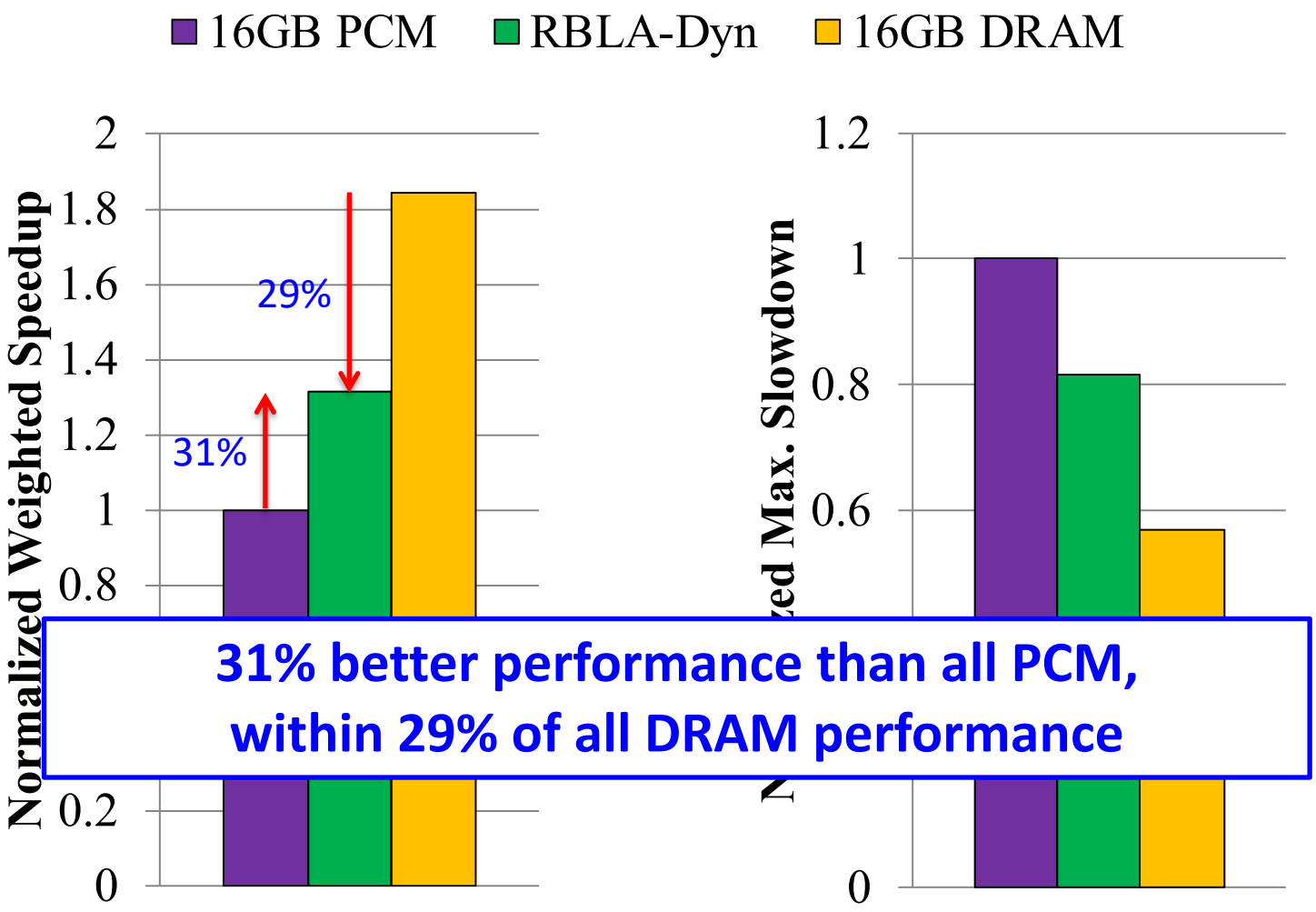


Hardware/software manage data allocation and movement
to achieve the best of multiple technologies

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.

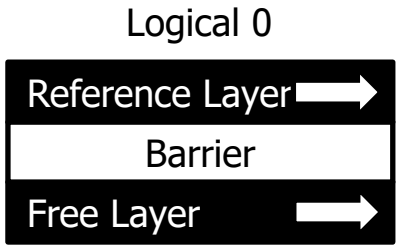
Hybrid vs. All-PCM/DRAM [ICCD'12]



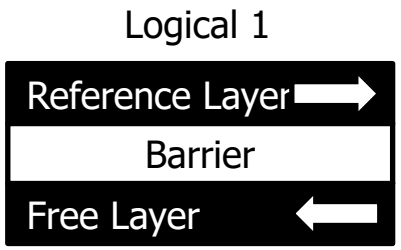
Yoon+, "Row Buffer Locality-Aware Data Placement in Hybrid Memories," ICCD 2012 Best Paper Award.

STT-MRAM as Main Memory

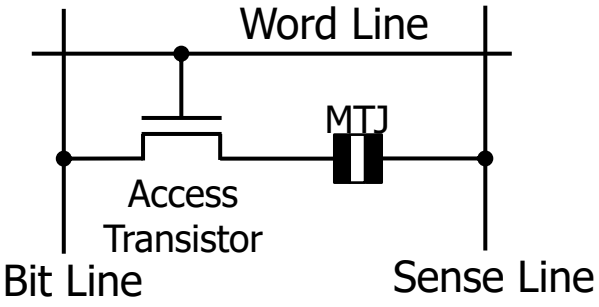
- Magnetic Tunnel Junction (MTJ) device
  - Reference layer: Fixed magnetic orientation
  - Free layer: Parallel or anti-parallel



- 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



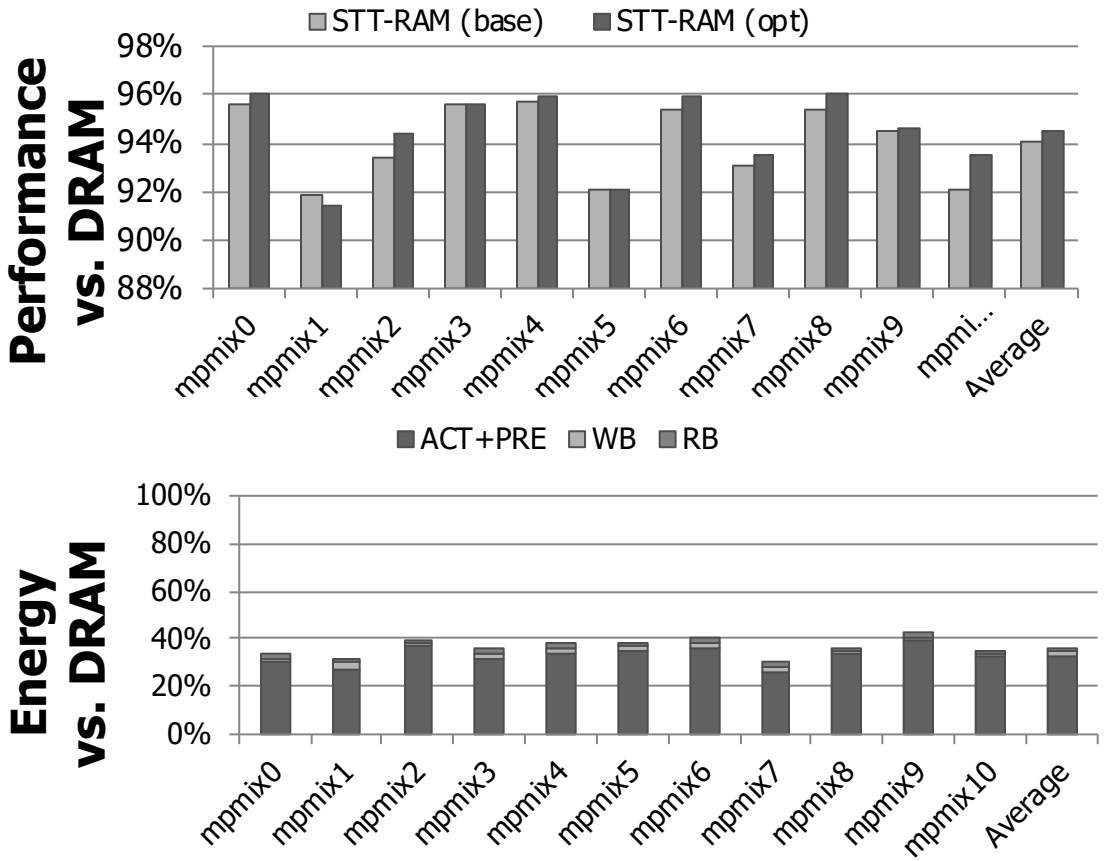
- Kultursay et al., "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," ISPASS 2013.

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



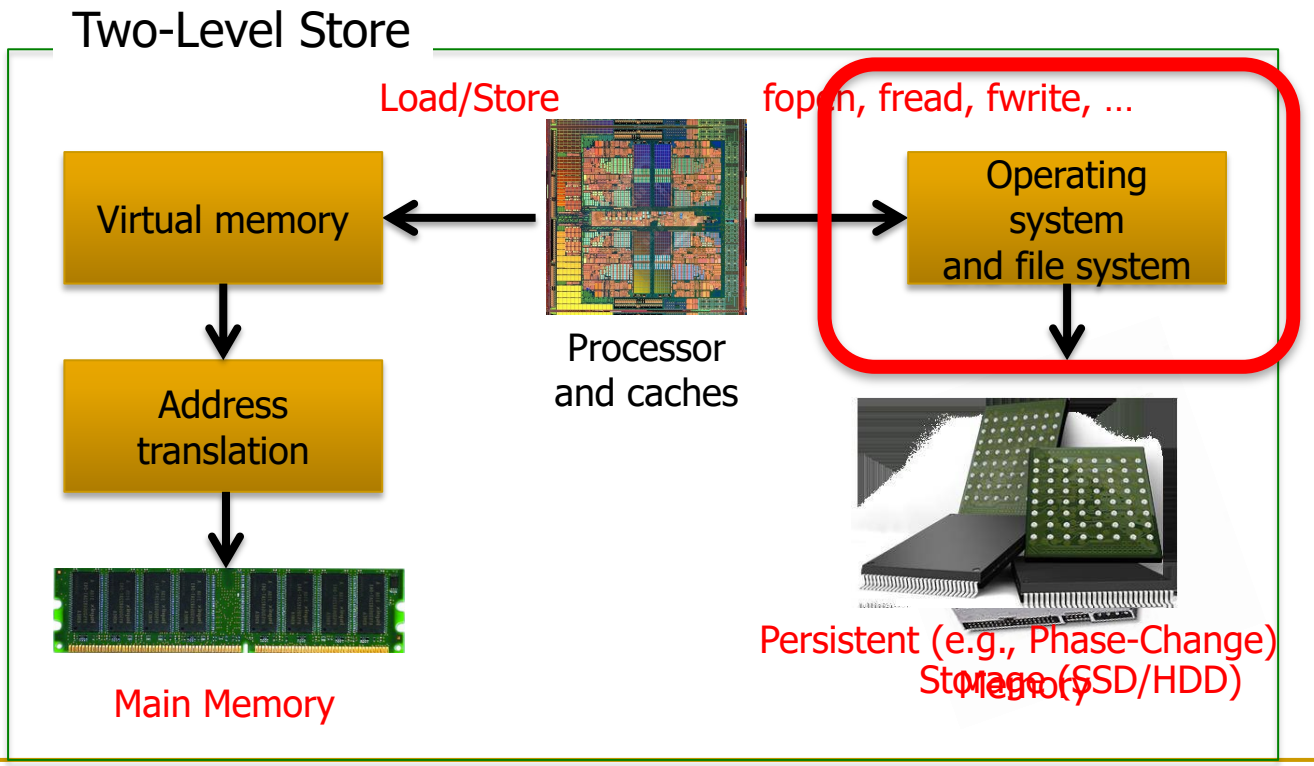
Kultursay+, "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," ISPASS 2013.

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

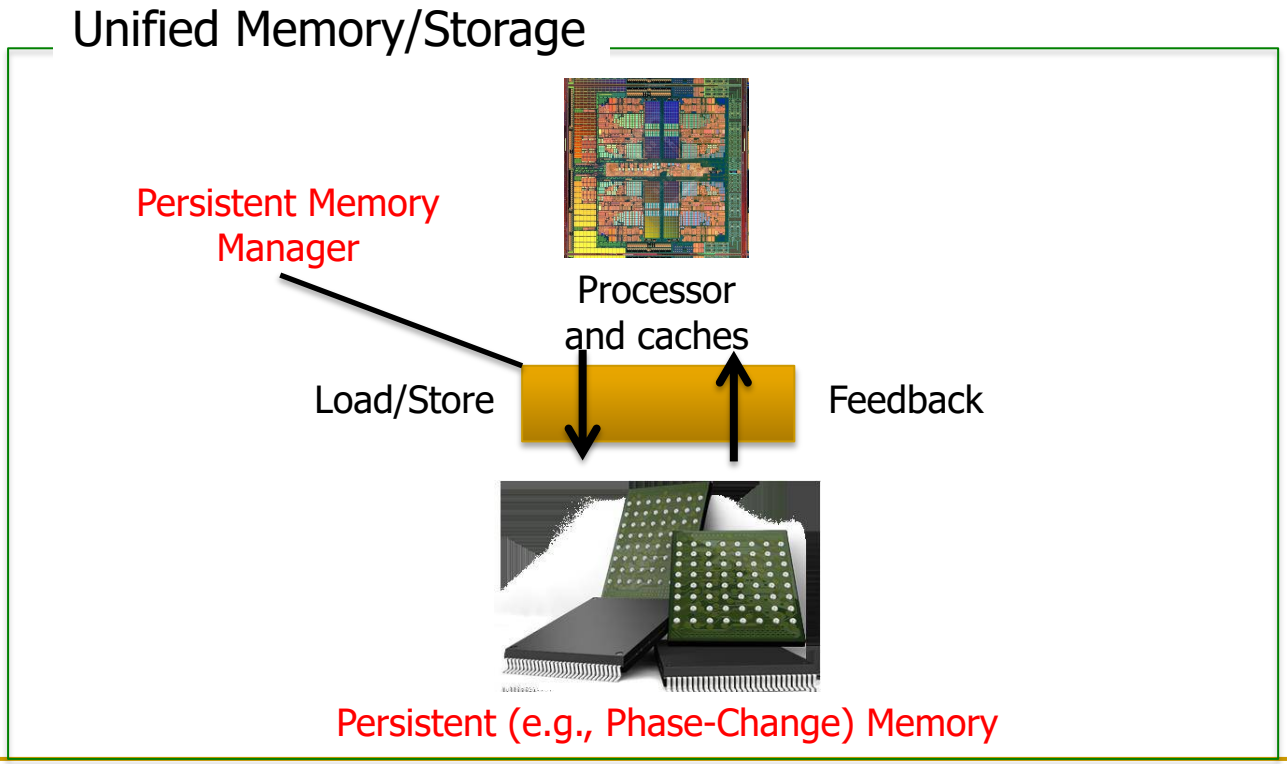
Coordinated Memory and Storage with NVM (I)

- 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.

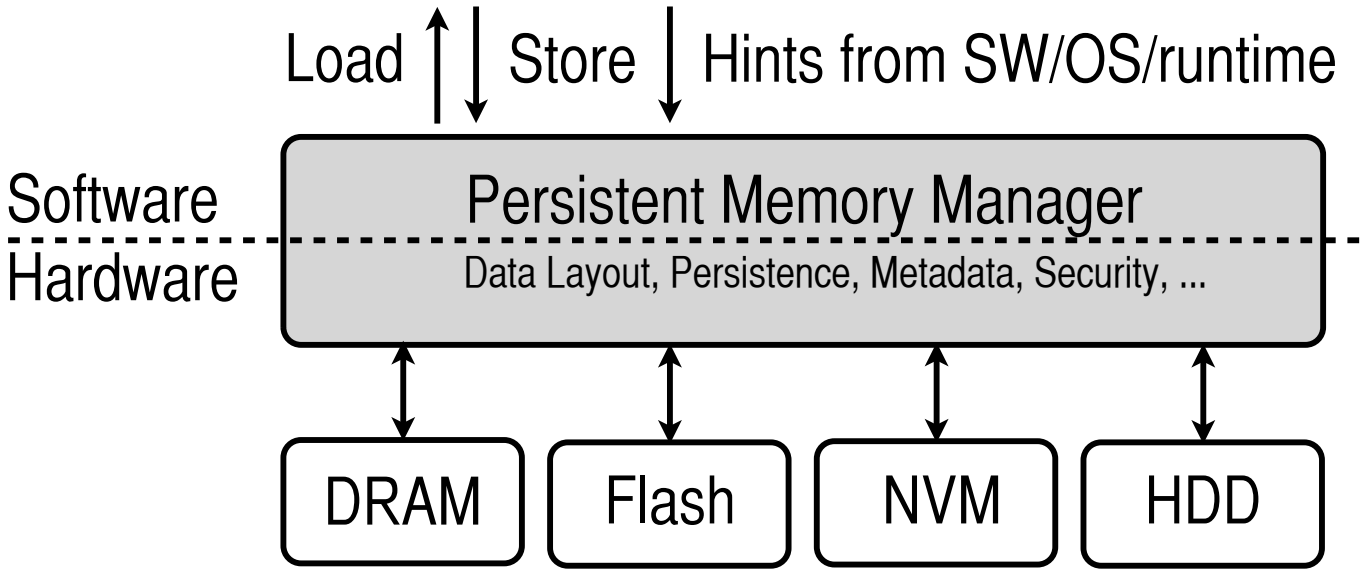
The Persistent Memory Manager (PMM)

- 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)

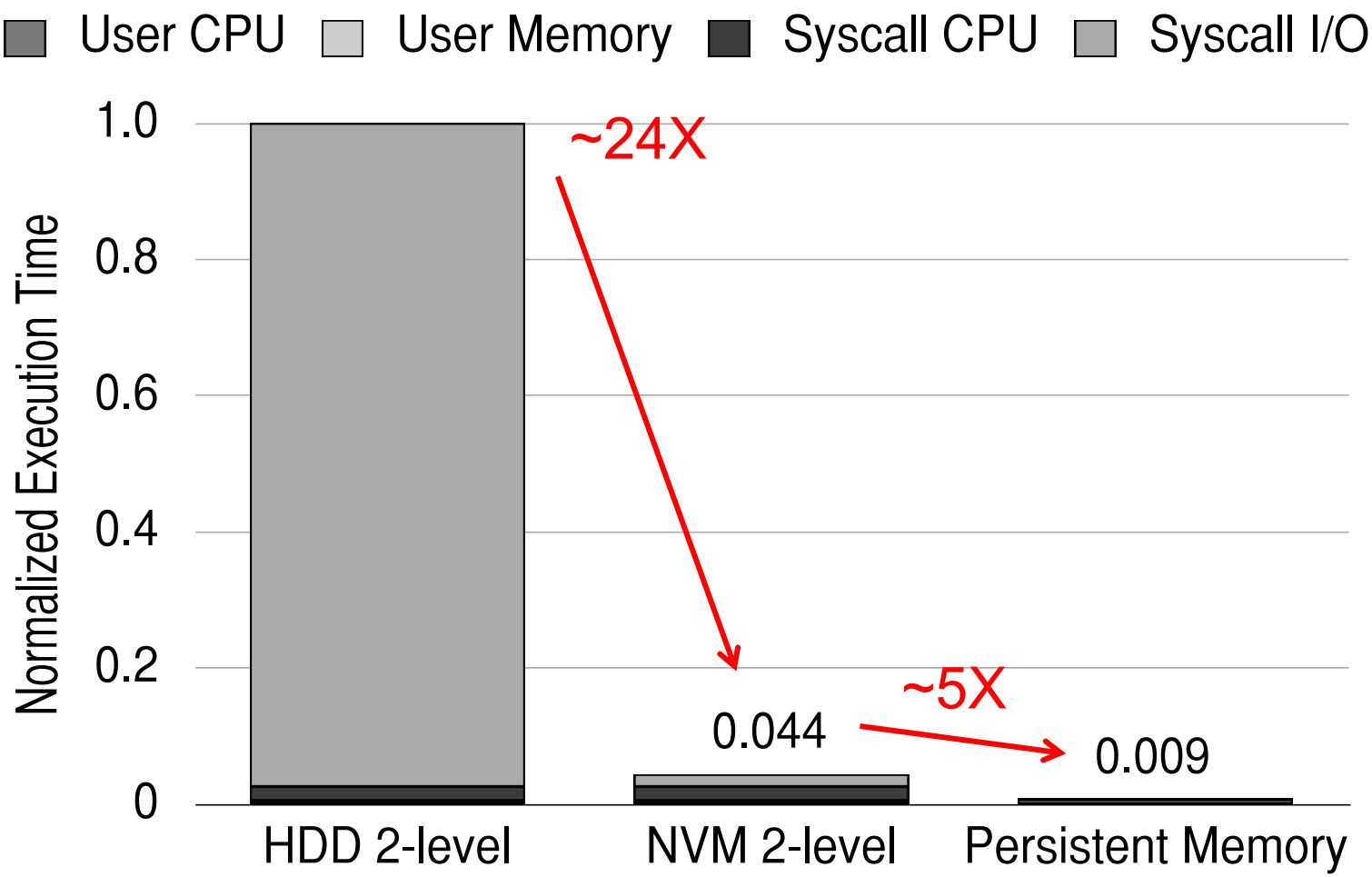
```
1  int main(void) {
2    // data in file.dat is persistent
3    FILE myData = "file.dat";
4    myData = new int[64];
5  }
6  void updateValue(int n, int value) {
7    FILE myData = "file.dat";
8    myData[n] = value; // value is persistent
9  }
```

Persistent objects



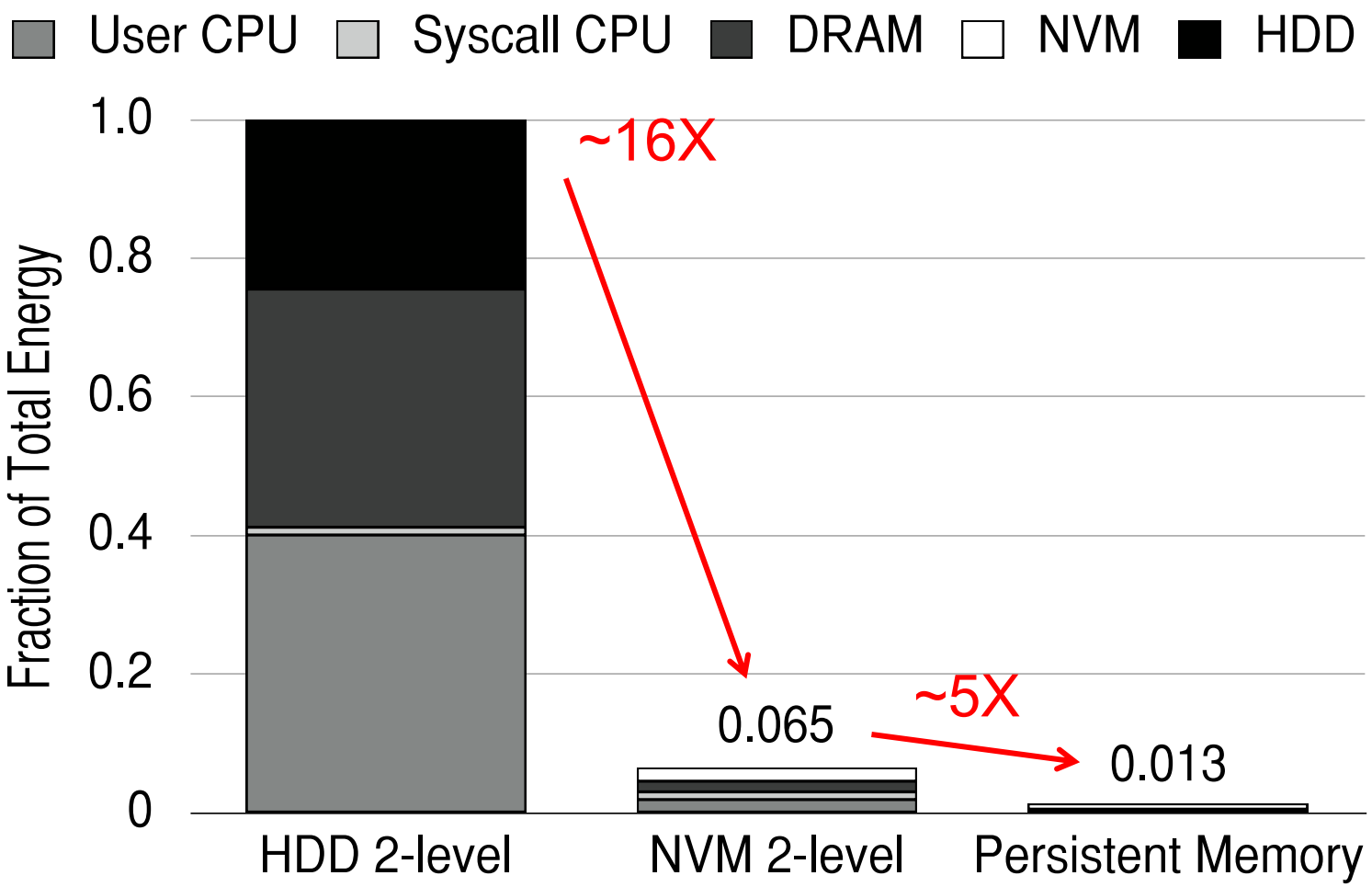
PMM uses access and hint information to allocate, locate, migrate and access data in the heterogeneous array of devices

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.

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

Principles (So Far)

- Better cooperation between devices and the system
  - Expose more information about devices to upper layers
  - More flexible interfaces
- Better-than-worst-case design
  - Do not optimize for the worst case
  - Worst case should not determine the common case
- Heterogeneity in design (specialization, asymmetry)
  - Enables a more efficient design (No one size fits all)
- These principles are coupled

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

Summary: Memory Scaling

- Memory scaling problems are a critical bottleneck for system performance, efficiency, and usability
- New memory architectures
  - **A lot of hope in fixing DRAM**
- Enabling emerging NVM technologies
  - **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**
- Three principles are essential for scaling
  - Software/hardware/device cooperation
  - Better-than-worst-case design
  - Heterogeneity (specialization, asymmetry)

Acknowledgments

- My current and past students and postdocs
  - Rachata Ausavarungnirun, Abhishek Bhowmick, Amirali Boroumand, Rui Cai, Yu Cai, Kevin Chang, Saugata Ghose, Kevin Hsieh, Tyler Huberty, Ben Jaiyen, Samira Khan, Jeremie Kim, Yoongu Kim, Yang Li, Jamie Liu, Lavanya Subramanian, Donghyuk Lee, Yixin Luo, Justin Meza, Gennady Pekhimenko, Vivek Seshadri, Lavanya Subramanian, Nandita Vijaykumar, HanBin Yoon, Jishen Zhao, ...
- My collaborators at CMU
  - Greg Ganger, Phil Gibbons, Mor Harchol-Balter, James Hoe, Mike Kozuch, Ken Mai, Todd Mowry, ...
- My collaborators elsewhere
  - Can Alkan, Chita Das, Sriram Govindan, Norm Jouppi, Mahmut Kandemir, Konrad Lai, Yale Patt, Moinuddin Qureshi, Partha Ranganathan, Bikash Sharma, Kushagra Vaid, Chris Wilkerson, ...

Funding Acknowledgments

- NSF
- GSRC
- SRC
- CyLab
- AMD, Google, Facebook, HP Labs, Huawei, IBM, Intel, Microsoft, Nvidia, Oracle, Qualcomm, Rambus, Samsung, Seagate, VMware

Open Source Tools

- Rowhammer
  - https://github.com/CMU-SAFARI/rowhammer
- Ramulator
  - https://github.com/CMU-SAFARI/ramulator
- MemSim
  - https://github.com/CMU-SAFARI/memsim
- NOCulator
  - https://github.com/CMU-SAFARI/NOCulator
- DRAM Error Model
  - http://www.ece.cmu.edu/~safari/tools/memerr/index.html
- Other open-source software from my group
  - https://github.com/CMU-SAFARI/
  - http://www.ece.cmu.edu/~safari/tools.html

Referenced Papers

- All are available at
  http://users.ece.cmu.edu/~omutlu/projects.htm
  http://scholar.google.com/citations?user=7XyGUGkAAAAJ&hl=en

- A detailed accompanying overview paper

  - Onur Mutlu and Lavanya Subramanian,
    "Research Problems and Opportunities in Memory Systems"
    *Invited Article in Supercomputing Frontiers and Innovations (**SUPERFRI**)*, 2015.

Related Videos and Course Materials

- **Undergraduate Computer Architecture Course Lecture Videos (2013, 2014, 2015)**
- **Undergraduate Computer Architecture Course Materials (2013, 2014, 2015)**
- **Graduate Computer Architecture Course Materials (Lecture Videos)**
- **Parallel Computer Architecture Course Materials (Lecture Videos)**
- **Memory Systems Short Course Materials (Lecture Video on Main Memory and DRAM Basics)**

Thank you.

onur@cmu.edu
http://users.ece.cmu.edu/~omutlu/

Rethinking Memory System Design (for Data-Intensive Computing)

Onur Mutlu
onur@cmu.edu
http://users.ece.cmu.edu/~omutlu/

July 20, 2015

SAMOS Keynote

Carnegie Mellon

Another Talk: NAND Flash Scaling Challenges

- Onur Mutlu,
  "Error Analysis and Management for MLC NAND Flash Memory"
  *Technical talk at Flash Memory Summit 2014 (**FMS**)*, Santa Clara, CA, August 2014. Slides (ppt) (pdf)

Cai+, "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis," DATE 2012.

Cai+, "Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime," ICCD 2012.

Cai+, "Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis and Modeling," DATE 2013.

Cai+, "Error Analysis and Retention-Aware Error Management for NAND Flash Memory," Intel Technology Journal 2013.

Cai+, "Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation," ICCD 2013.

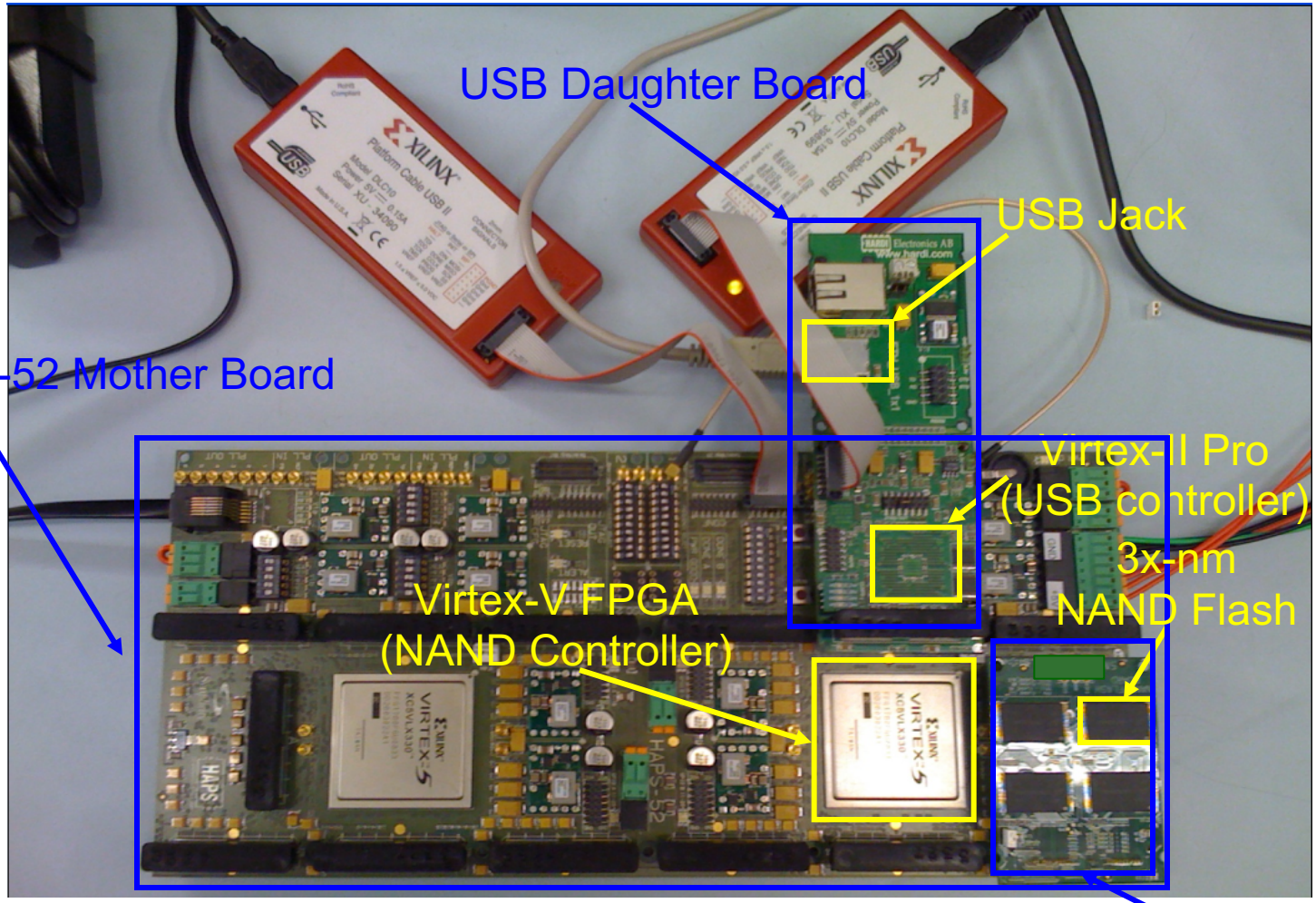
Cai+, "Neighbor-Cell Assisted Error Correction for MLC NAND Flash Memories," SIGMETRICS 2014.

Cai+,"Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery," HPCA 2015.

Cai+, "Read Disturb Errors in MLC NAND Flash Memory: Characterization and Mitigation," DSN 2015.

Luo+, "WARM: Improving NAND Flash Memory Lifetime with Write-hotness Aware Retention Management," MSST 2015.

Experimental Infrastructure (Flash)



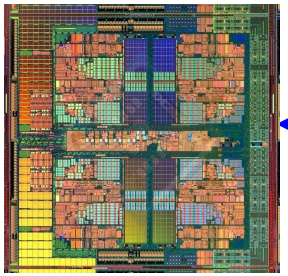
[Cai+, DATE 2012, ICCD 2012, DATE 2013, ITJ 2013, ICCD 2013, SIGMETRICS 2014, HPCA 2015, DSN 2015, MSST 2015]

Backup Slides

Current Research Focus Areas

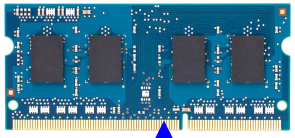
***Research Focus:* Computer architecture, HW/SW, bioinformatics**

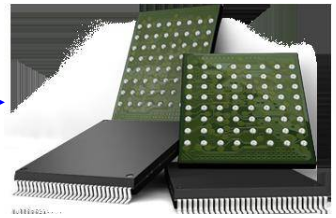
- *Memory, memory, memory, storage, interconnects*
- *Parallel architectures, heterogeneous architectures, GP-GPUs*
- *System/architecture interaction, new execution models*
- *Energy efficiency, fault tolerance, hardware security*
- *Genome sequence analysis & assembly algorithms and architectures*



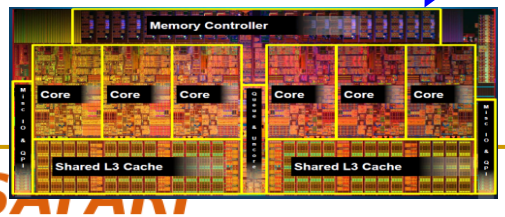
Heterogeneous Processors and Accelerators

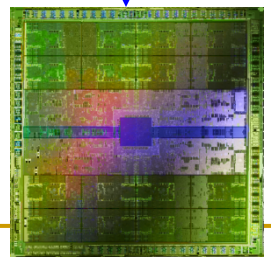
Hybrid Main Memory





Persistent Memory/Storage





General Purpose GPUs

Broad research spanning apps, systems, logic with architecture at the center

Some Current Directions and Projects

- *Rethinking Memory System Design for Data-Intensive Computing*
  - *All aspects of DRAM, Flash Memory, Emerging Technologies*
- *Single-Level Stores: Merging Memory and Storage with Fast NVM*
- *GPUs as First-Class Computing Engines*
- *In-memory Computing: Enabling Near-Data Processing*
- *Predictable Systems: QoS Everywhere in the System*
- *Secure and Easy-to-Program/Manage Memories: DRAM, Flash, NVM*
- *Heterogeneous Systems: Architecting and Exploiting Asymmetry*
- *Efficient and Scalable Interconnects*
- *Genome Sequence Analysis & Assembly: Algorithms and Architectures*

In-Memory Processing

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

Junwhan Ahn, Sungpack Hong*, Sungjoo Yoo,
Onur Mutlu+, Kiyoung Choi

Seoul National University *Oracle Labs +Carnegie Mellon University

Challenges in Scalable Graph Processing

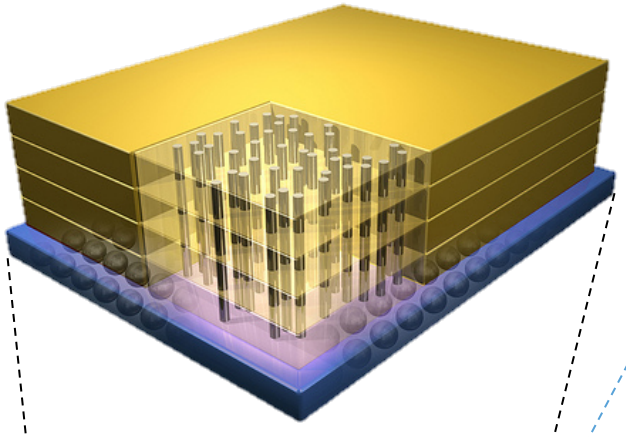
- **Challenge 1**: How to provide *high memory bandwidth* to computation units in a practical way?
  - Processing-in-memory based on 3D-stacked DRAM

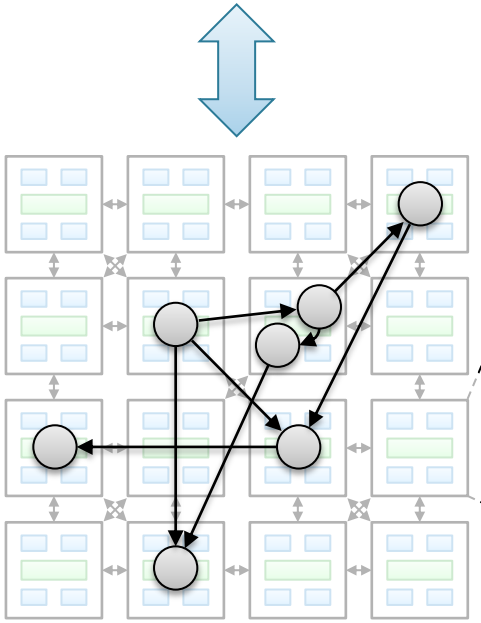
- **Challenge 2**: How to design computation units that *efficiently exploit large memory bandwidth*?
  - Specialized in-order cores called *Tesseract* cores
    - Latency-tolerant programming model
    - Graph-processing-specific prefetching schemes

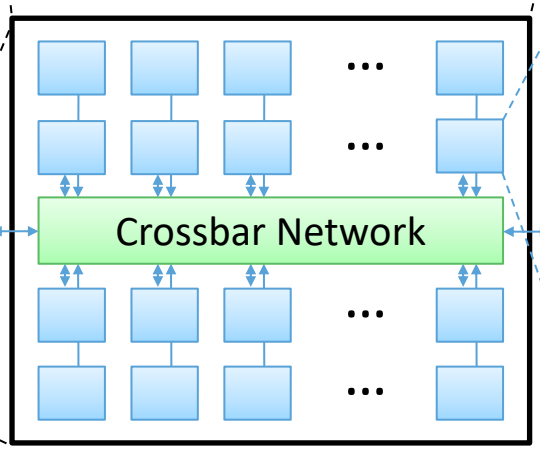
Tesseract System fo Graph Processing

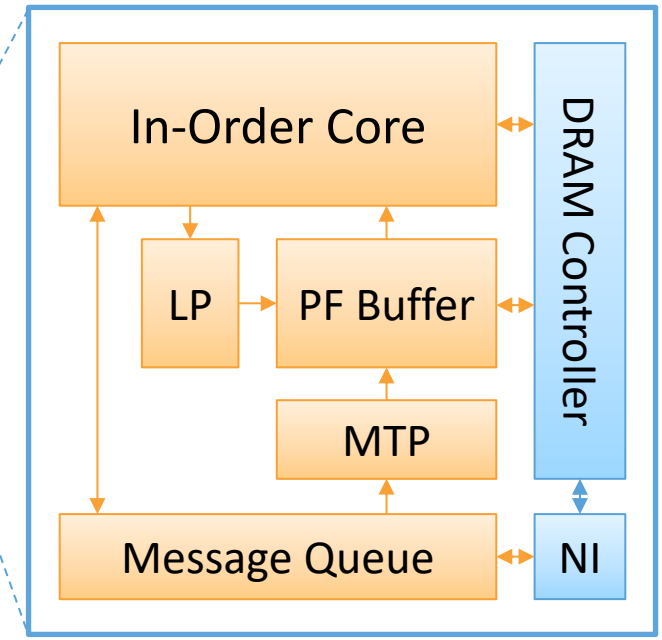
Host Processor

Memory-Mapped
Accelerator Interface
(Noncacheable, Physically Addressed)

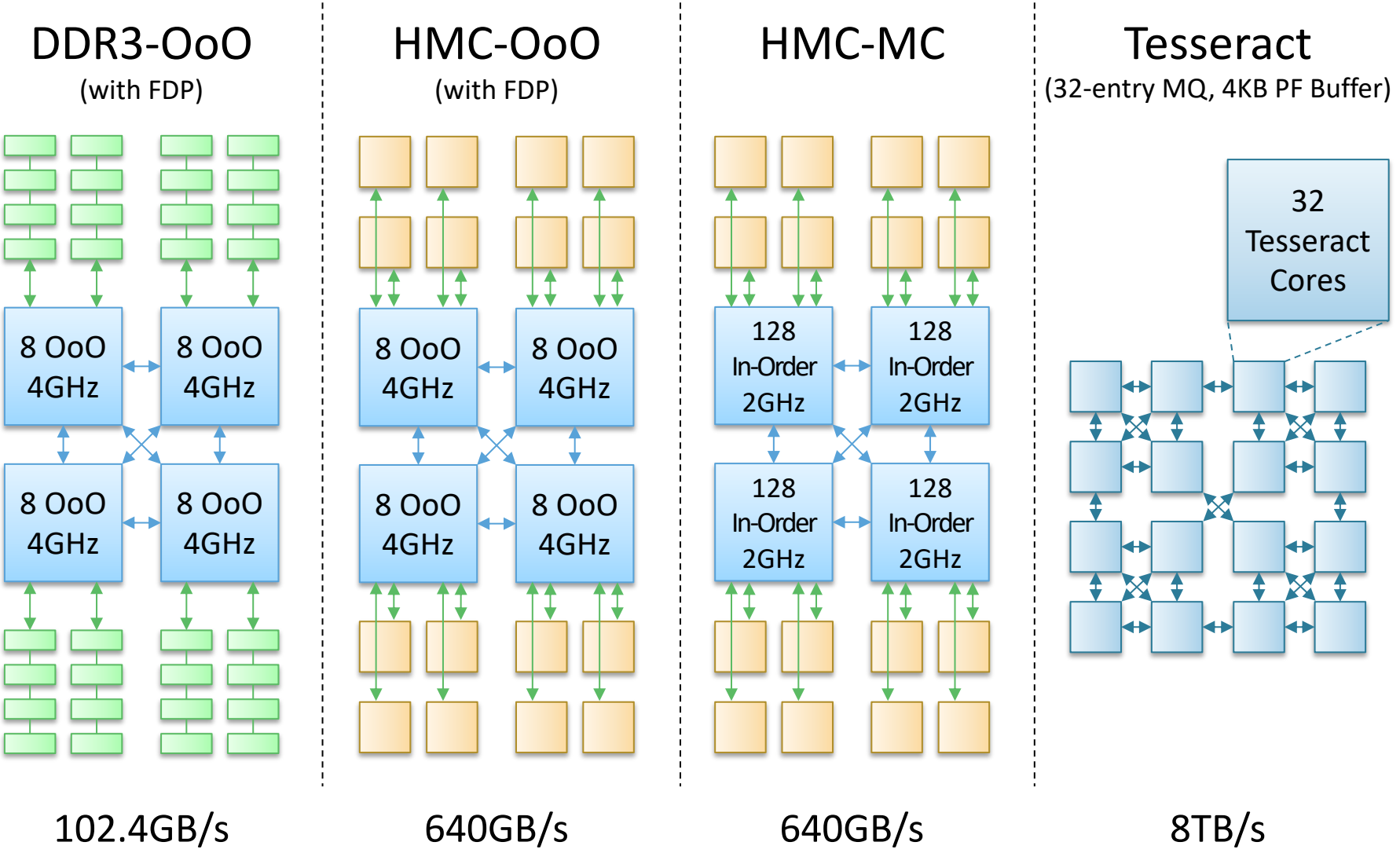








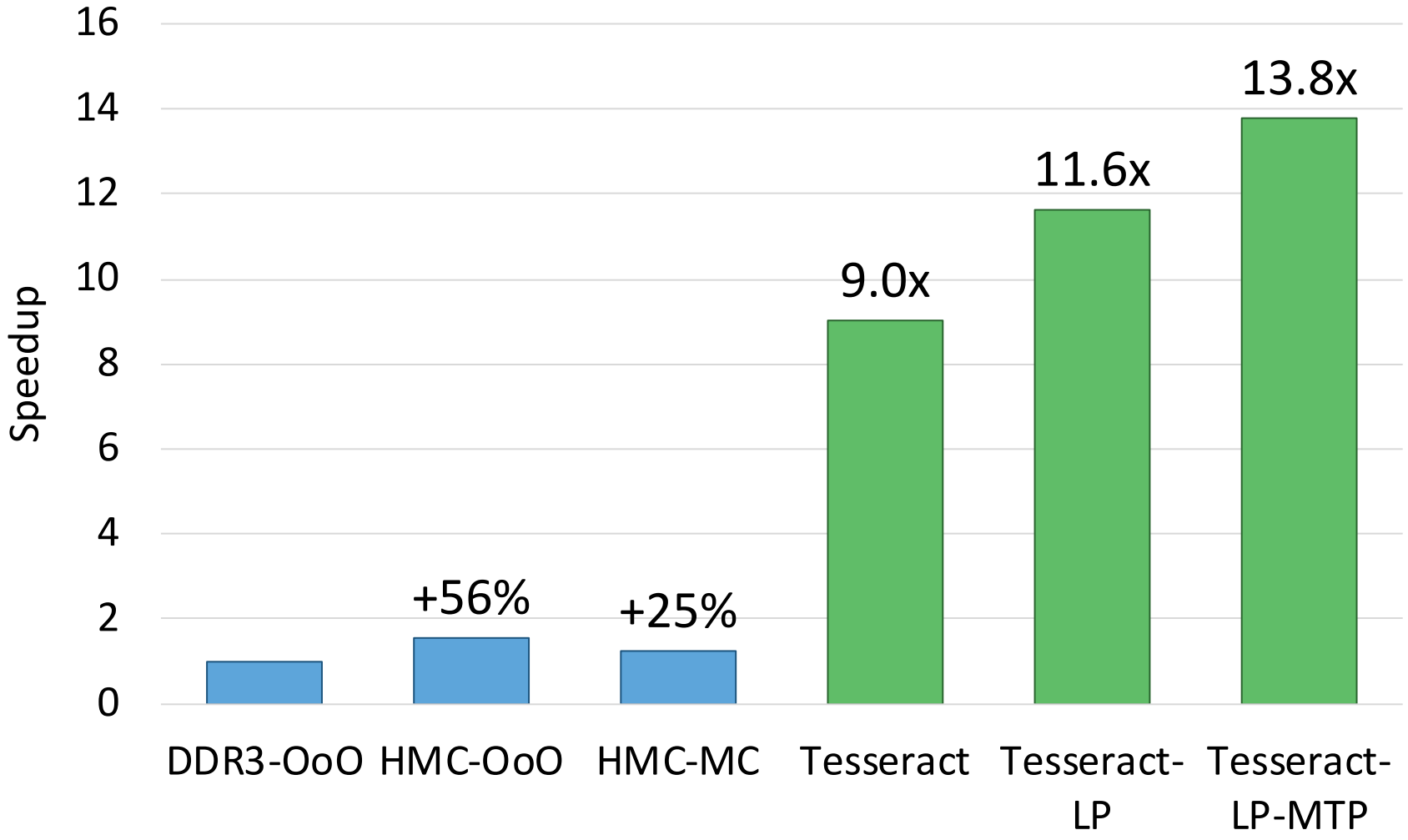
Evaluated Systems



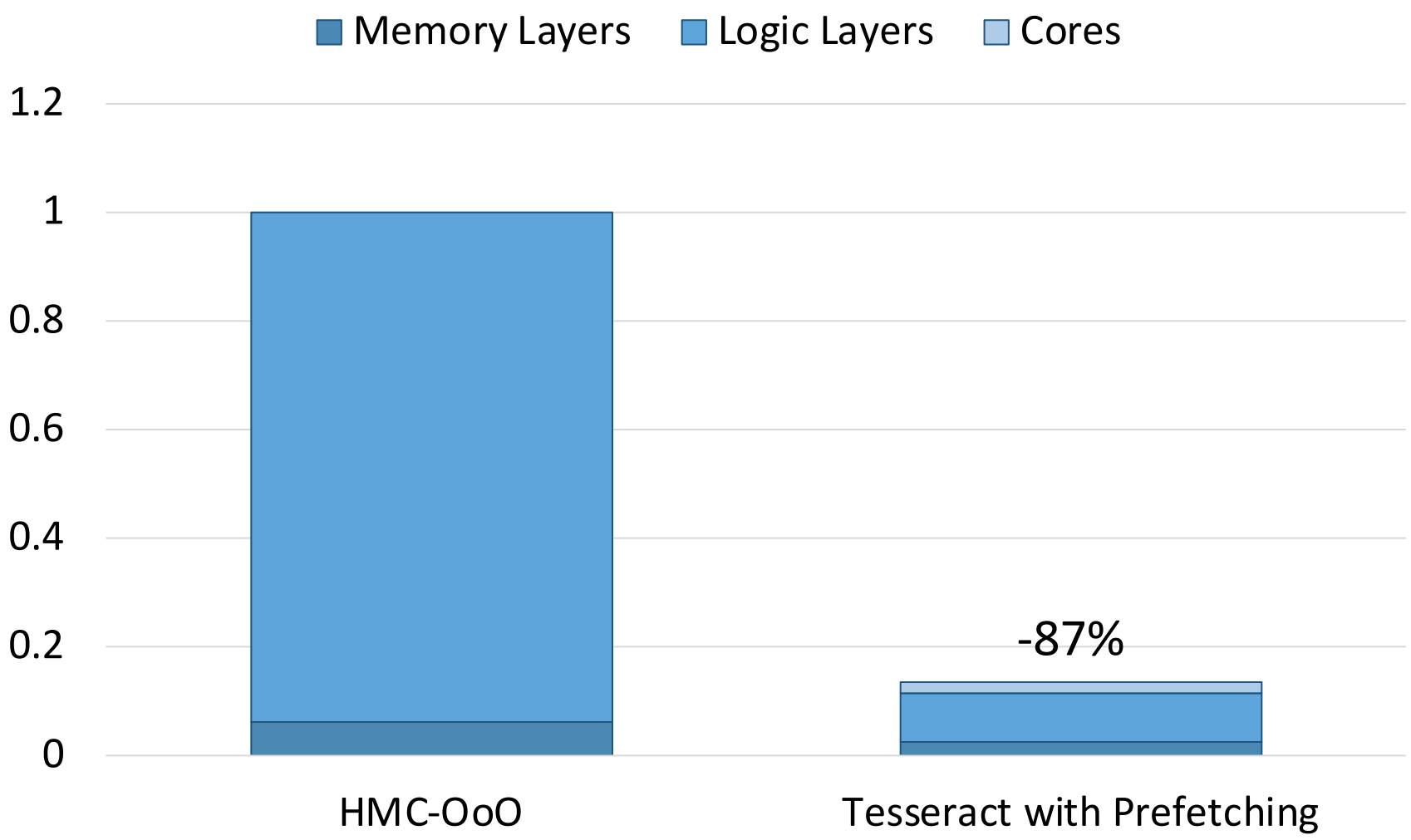
Workloads

- 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

Performance



Memory Energy Consumption



Conclusion

- Revisiting the PIM concept in a new context
  - Cost-effective 3D integration of logic and memory
  - Graph processing workloads demanding high memory bandwidth
- Tesseract: scalable PIM 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
- Evaluations demonstrate the benefits of Tesseract
  - 14x performance improvement & 87% energy reduction
  - Scalable: *memory-capacity-proportional* performance

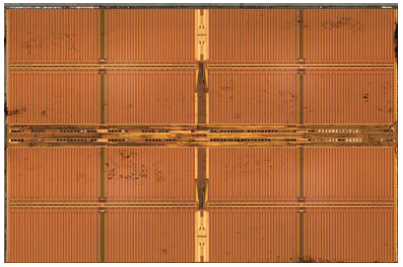
PIM-Enabled Instructions: A Low-Overhead, Locality-Aware PIM Architecture

Junwhan Ahn, Sungjoo Yoo, Onur Mutlu[+], and Kiyoung Choi

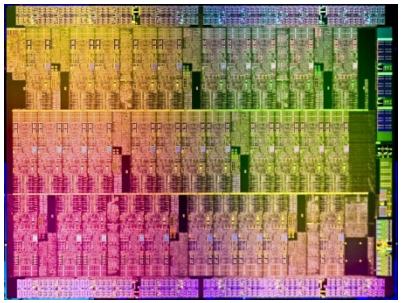
Seoul National University [+]Carnegie Mellon University

Challenges in Processing-in-Memory

Cost-effectiveness



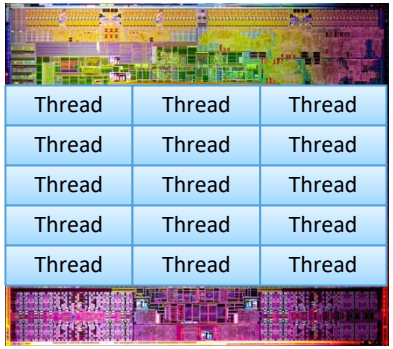
DRAM die

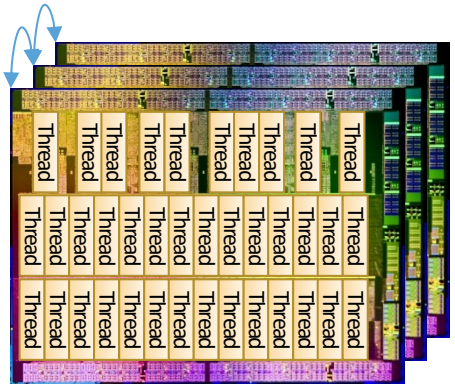


Complex Logic

Programming Model

Host Processor

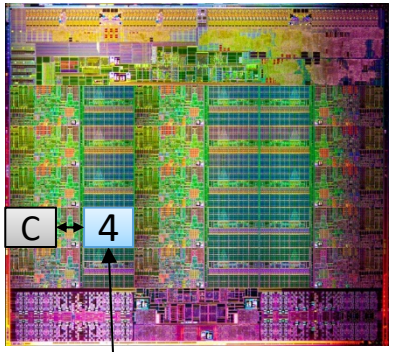


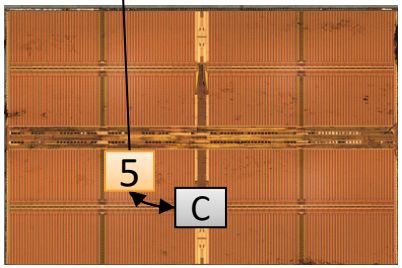


In-Memory Processors

Coherence & VM

Host Processor





DRAM die

PIM-Enabled Instructions

- Our direction: simple PIM operations as ISA extension
  - Simple: low-overhead implementation
  - ISA extension: No changes to existing programming models
  - *One more thing*: locality-aware dynamic PIM execution
    - Adaptation between host-side and memory-side execution

- Evaluation highlight
  - 47% speedup over conventional systems in large inputs
  - 32% speedup over PIM-only systems in small inputs
  - Impact of data locality, energy efficiency, and more…

Session 6A: Memory Systems I (10:20~10:45)

NVM and Emerging Technologies

Error Management in MLC NAND Flash

- Problem: MLC NAND flash memory reliability/endurance is a key challenge for satisfying future storage systems' requirements
- Our Goals: (1) Build reliable error models for NAND flash memory via experimental characterization, (2) Develop efficient techniques to improve reliability and endurance
- This talk provides a "flash" summary of our recent results published in the past 3 years:
  - Experimental error and threshold voltage characterization **[DATE'12&13]**
  - Retention-aware error management **[ICCD'12]**
  - Program interference analysis and read reference V prediction **[ICCD'13]**
  - Neighbor-assisted error correction **[SIGMETRICS'14]**

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

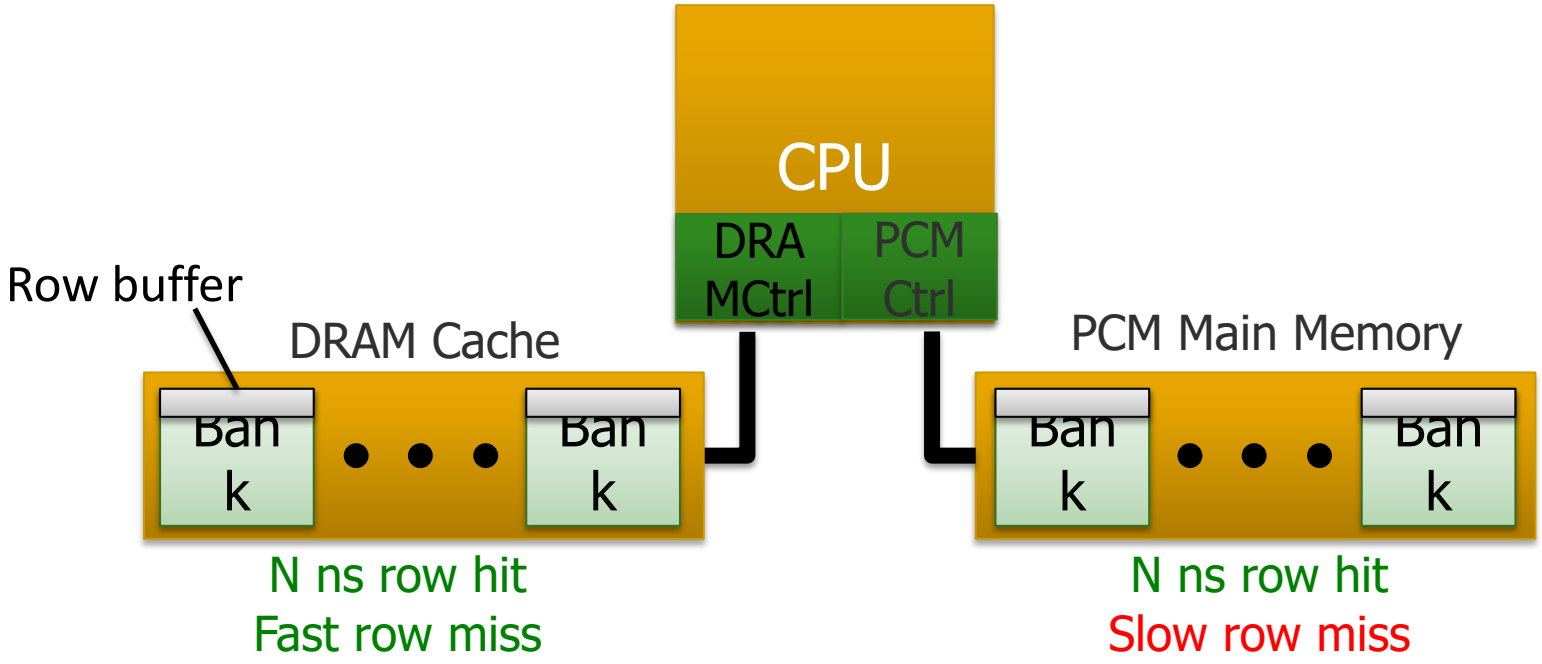
Table 1. Technology survey.

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.

DRAM vs. PCM: An Observation

- Row buffers are the same in DRAM and PCM
- Row buffer hit latency **same** in DRAM and PCM
- Row buffer miss latency **small** in DRAM, **large** in PCM



- Accessing the row buffer in PCM is fast
- What incurs high latency is the PCM array access → avoid this

Row-Locality-Aware Data Placement

- Idea: Cache in DRAM only those rows that
  - Frequently cause row buffer conflicts → because row-conflict latency is smaller in DRAM
  - Are reused many times → to reduce cache pollution and bandwidth waste

- Simplified rule of thumb:
  - Streaming accesses: Better to place in PCM
  - Other accesses (with some reuse): Better to place in DRAM

- Yoon et al., "Row Buffer Locality-Aware Data Placement in Hybrid Memories," ICCD 2012 Best Paper Award.

More Detail on New Memory Architectures

In-DRAM Bitwise AND/OR

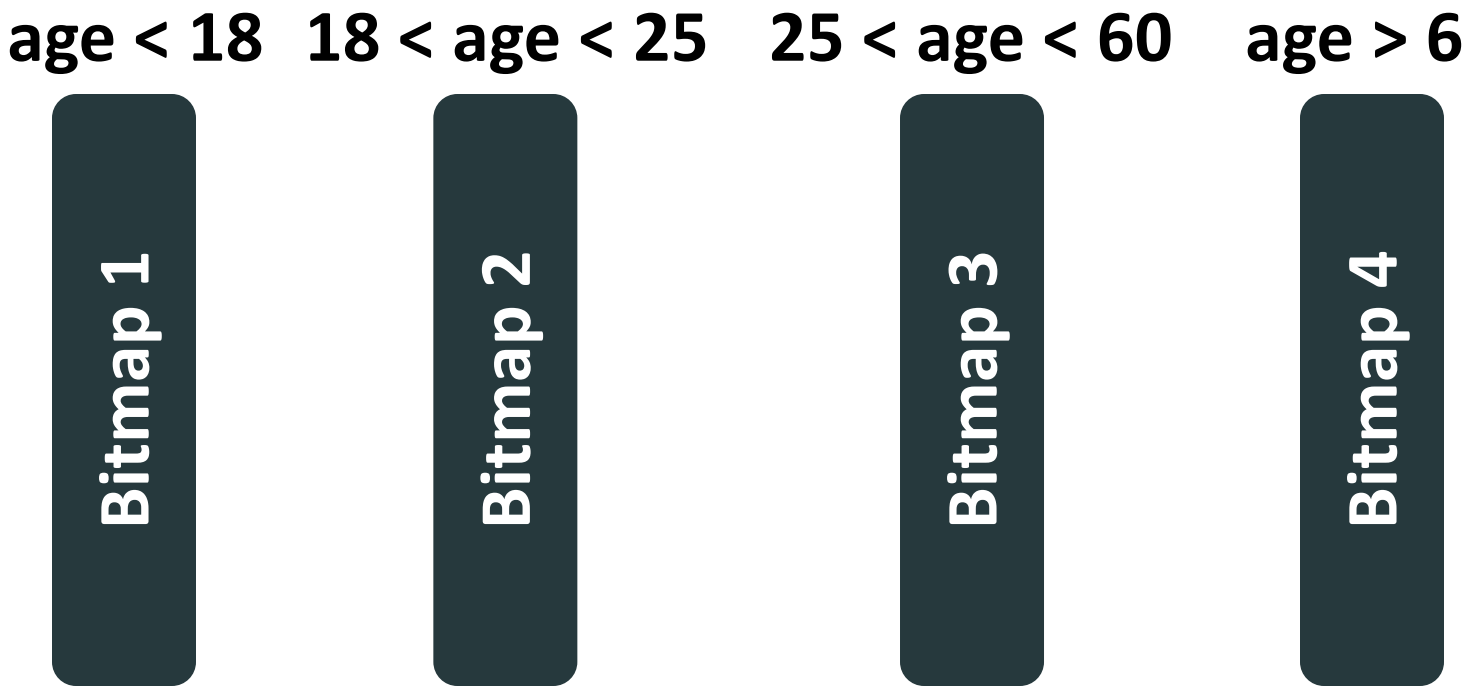
Required Operation: Perform a bitwise AND of two rows *A* and *B* and store the result in *C*

- *R0* – reserved zero row, *R1* – reserved one row
- *D1, D2, D3* – Designated rows for triple activation

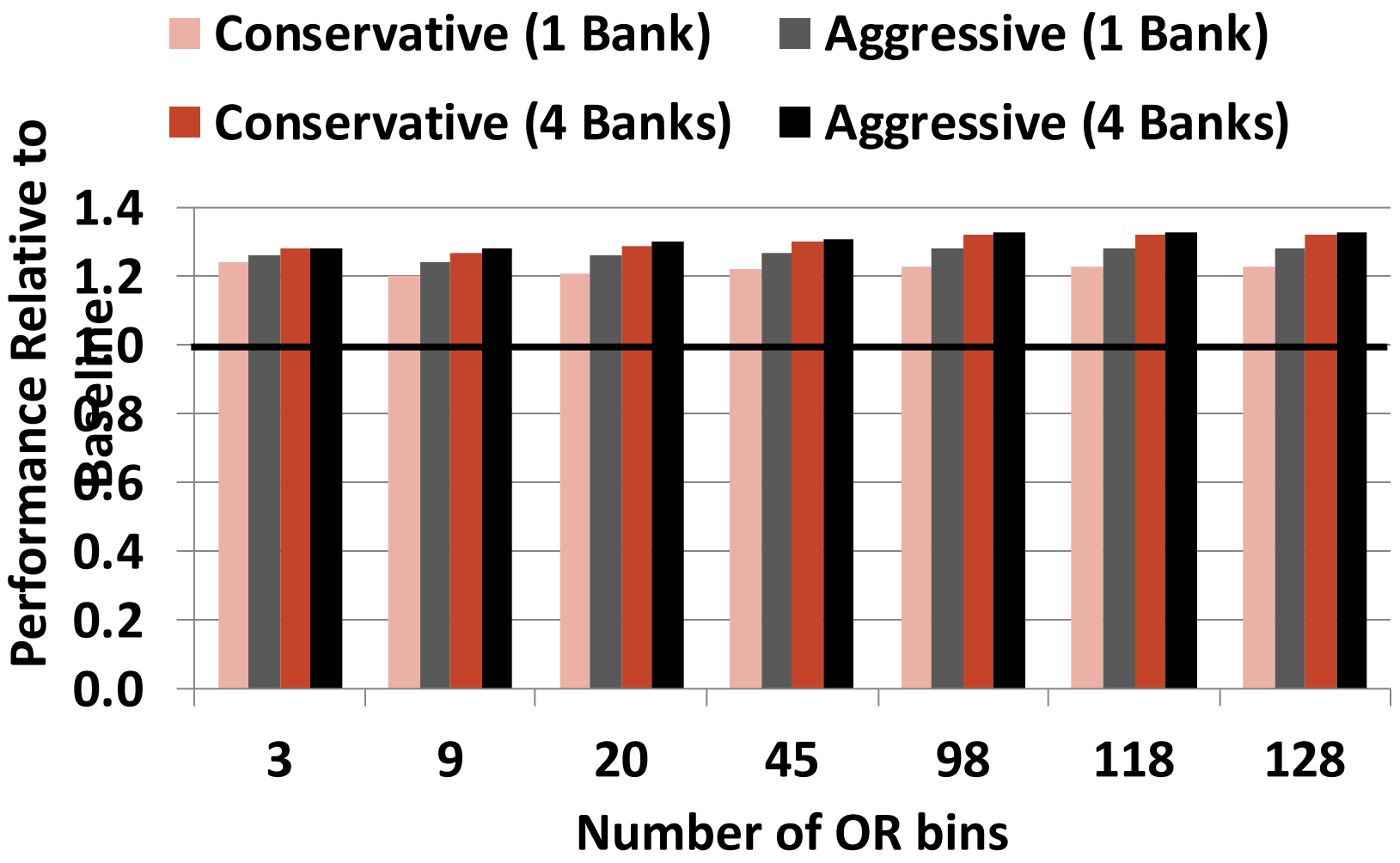
1. RowClone *A* into *D1*
2. RowClone *B* into *D2*
3. RowClone *R0* into *D3*
4. ACTIVATE *D1,D2,D3*
5. RowClone *Result* into *C*

Bitmap Index

- Alternative to B-tree and its variants
- Efficient for performing *range queries* and *joins*

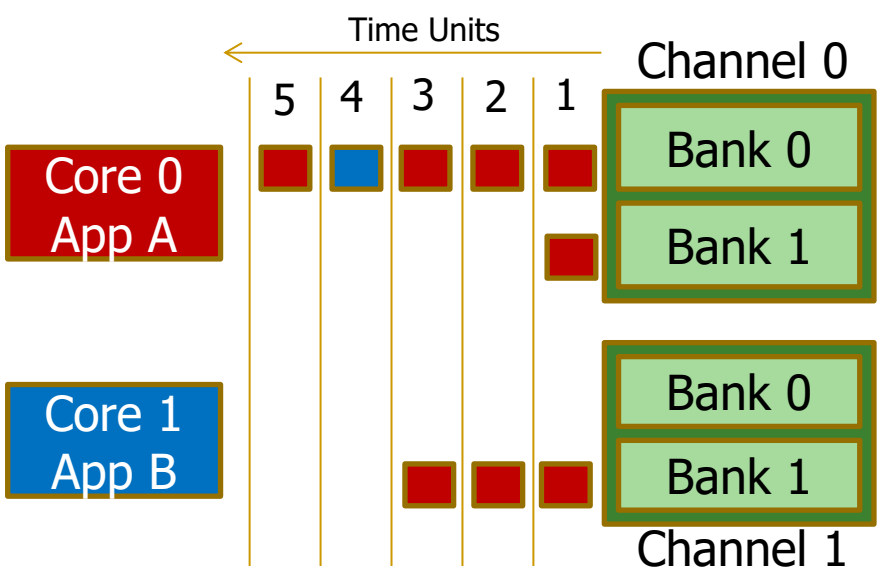


Performance Evaluation

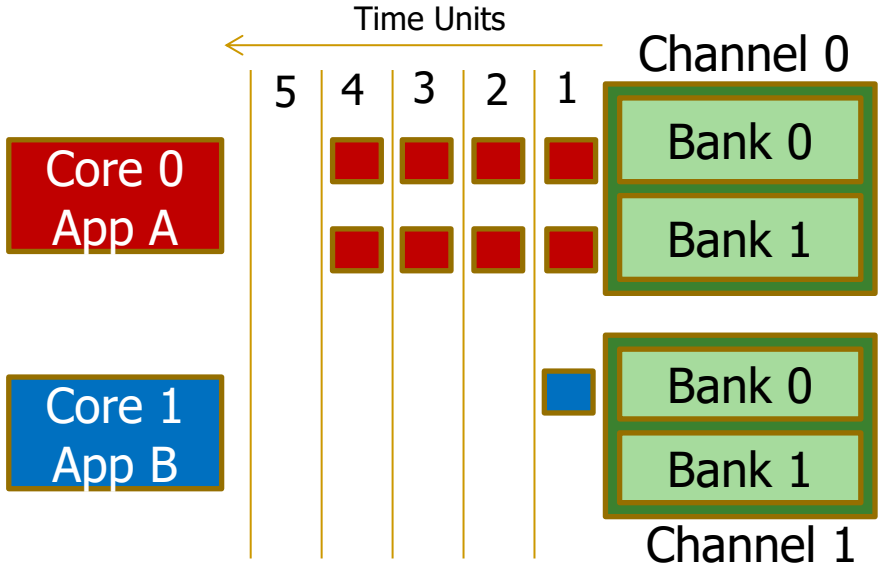


A Mechanism to Reduce Memory Interference

- Memory Channel Partitioning
  - Idea: System software maps badly-interfering applications' pages to different channels [Muralidhara+, MICRO'11]



Conventional Page Mapping



Channel Partitioning

- Separate data of low/high intensity and low/high row-locality applications
- Especially effective in reducing interference of threads with "medium" and "heavy" memory intensity
  - 11% higher performance over existing systems (200 workloads)

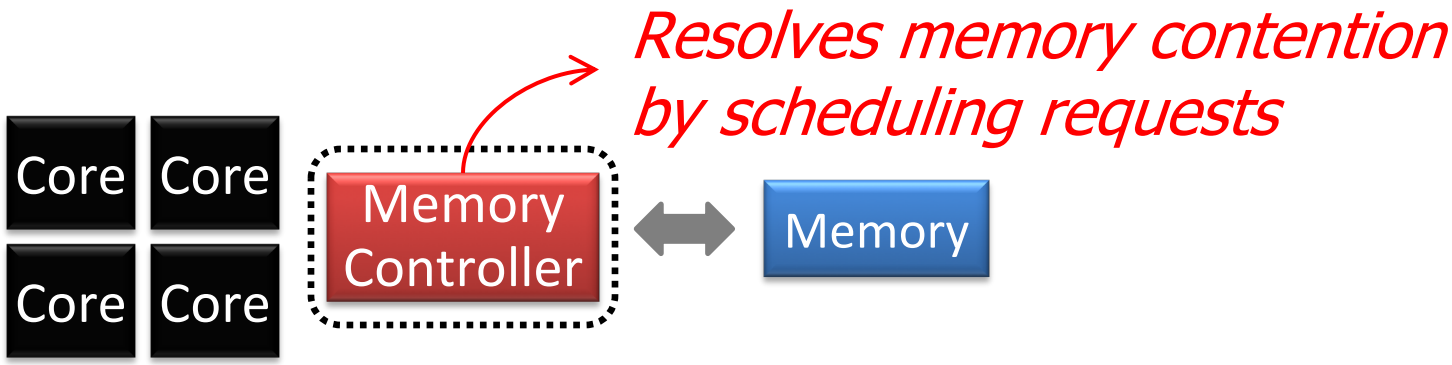
More on Memory Channel Partitioning

- Sai Prashanth Muralidhara, Lavanya Subramanian, Onur Mutlu, Mahmut Kandemir, and Thomas Moscibroda,
  "Reducing Memory Interference in Multicore Systems via Application-Aware Memory Channel Partitioning"
  *Proceedings of the 44th International Symposium on Microarchitecture (**MICRO**)*, Porto Alegre, Brazil, December 2011. Slides (pptx)

Designing QoS-Aware Memory Systems: Approaches

- Smart resources: Design each shared resource to have a configurable interference control/reduction mechanism
  - QoS-aware memory controllers [Mutlu+ MICRO'07] [Moscibroda+, Usenix Security'07] [Mutlu+ ISCA'08, Top Picks'09] [Kim+ HPCA'10] [Kim+ MICRO'10, Top Picks'11] [Ebrahimi+ ISCA'11, MICRO'11] [Ausavarungnirun+, ISCA'12][Subramanian+, HPCA'13] [Subramanian+, ICCD'14]
  - QoS-aware interconnects [Das+ MICRO'09, ISCA'10, Top Picks '11] [Grot+ MICRO'09, ISCA'11, Top Picks '12]
  - QoS-aware caches
- Dumb resources: Keep each resource free-for-all, but reduce/control interference by injection control or data mapping
  - Source throttling to control access to memory system [Ebrahimi+ ASPLOS'10, ISCA'11, TOCS'12] [Ebrahimi+ MICRO'09] [Nychis+ HotNets'10] [Nychis+ SIGCOMM'12]
  - QoS-aware data mapping to memory controllers [Muralidhara+ MICRO'11]
  - QoS-aware thread scheduling to cores [Das+ HPCA'13]

QoS-Aware Memory Scheduling



- How to schedule requests to provide
  - High system performance
  - High fairness to applications
  - Configurability to system software
- Memory controller needs to be aware of threads

More on DRAM Disturbance Errors

- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu,
  "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"
  *Proceedings of the 41st International Symposium on Computer Architecture (**ISCA**)*, Minneapolis, MN, June 2014. Slides (pptx) (pdf) Lightning Session Slides (pptx) (pdf) Source Code and Data

- Source Code
  - https://github.com/CMU-SAFARI/rowhammer

More on Heterogeneous-Reliability Memory

- Yixin Luo, Sriram Govindan, Bikash Sharma, Mark Santaniello, Justin Meza, Aman Kansal, Jie Liu, Badriddine Khessib, Kushagra Vaid, and Onur Mutlu,
  "Characterizing Application Memory Error Vulnerability to Optimize Data Center Cost via Heterogeneous-Reliability Memory"
  *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**)*, Atlanta, GA, June 2014.
  Slides (pptx) (pdf) Coverage on ZDNet

How good does memory need to be?

Summary: *Main memory is all the same. But why? All data is not created equal, so why is memory? Another reason the cloud is winning.*

By Robin Harris for Storage Bits | July 22, 2014 -- 11:20 GMT (04:20 PDT)

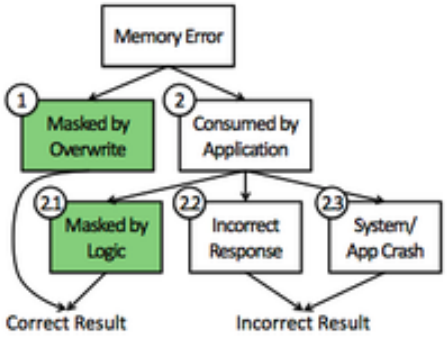
Follow @StorageMojo

Get the ZDNet Big Data newsletter now

Comments 3 Votes 2 Share 44 Tweet 26 Share more +

There is money to be saved

Main memory treats all data the same. In servers, which typically use some form of error correcting code (ECC) to detect and correct errors, the added cost can be significant with today's large-memory servers.



Memory error outcomes.

Researchers at Microsoft and Carnegie Mellon University are studying the issue. Finding that ≈57 percent of data center TCO is capital cost - most of which is server cost - and that processors and memory are about 60 percent of server cost, it's clear that reducing memory costs could materially improve data center capital efficiency.

ECC also slows down systems and, due to added logic and RAM, increases power and cooling costs. It's a double whammy.

The researchers wanted to know if applications all need the level of care that ECC provides and, if they don't, how much could be saved through hetrogenous memory systems. The key is to understand how vulnerable a given workload is to memory errors.

How Do We Solve The Problem?

- Tolerate it: Make DRAM and controllers more intelligent
  - New interfaces, functions, architectures: system-DRAM codesign
- Eliminate or minimize it: Replace or (more likely) augment DRAM with a different technology
  - New technologies and system-wide rethinking of memory & storage
- Embrace it: Design heterogeneous-reliability memories that map error-tolerant data to less reliable portions
  - New models for execution and maybe usage

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

More on Memory Scheduling

- Lavanya Subramanian, Vivek Seshadri, Yoongu Kim, Ben Jaiyen, and Onur Mutlu, **"MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems"**
  *Proceedings of the 19th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Shenzhen, China, February 2013. Slides (pptx)

- Lavanya Subramanian, Donghyuk Lee, Vivek Seshadri, Harsha Rastogi, and Onur Mutlu,
  "The Blacklisting Memory Scheduler: Achieving High Performance and Fairness at Low Cost"
  *Proceedings of the 32nd IEEE International Conference on Computer Design (**ICCD**)*, Seoul, South Korea, October 2014. Slides (pptx) (pdf)

- Yoongu Kim, Michael Papamichael, Onur Mutlu, and Mor Harchol-Balter,
  "Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior"
  *Proceedings of the 43rd International Symposium on Microarchitecture (**MICRO**)*, pages 65-76, Atlanta, GA, December 2010. Slides (pptx) (pdf)

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

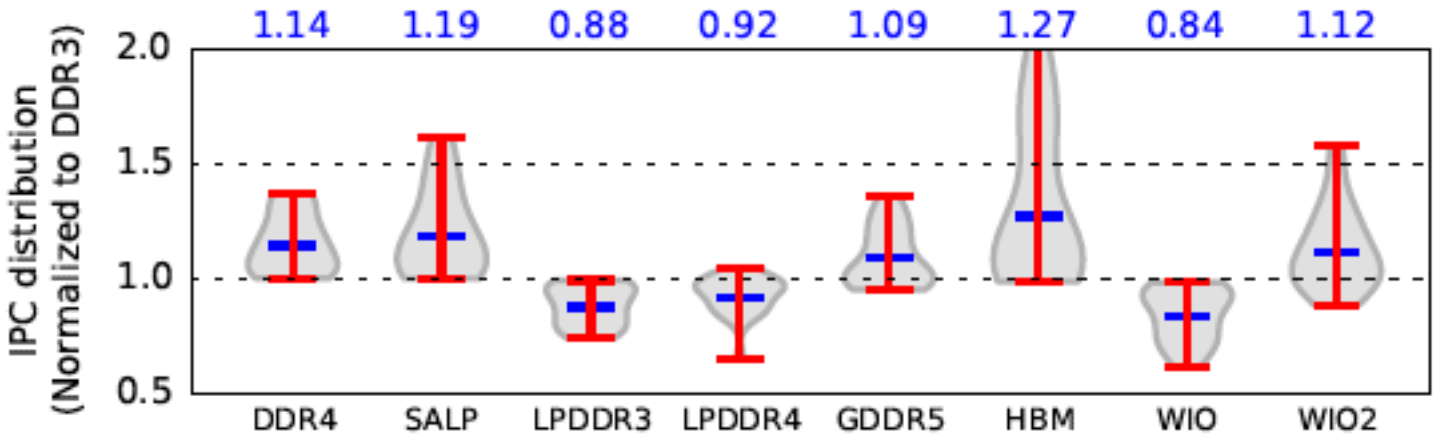


Figure 2. Performance comparison of DRAM standards

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