

Rethinking Memory System Design (and the Platforms We Design Around It)

Onur Mutlu

omutlu@gmail.com

<https://people.inf.ethz.ch/omutlu>

December 4, 2017

INESC-ID Distinguished Lecture (Lisbon)



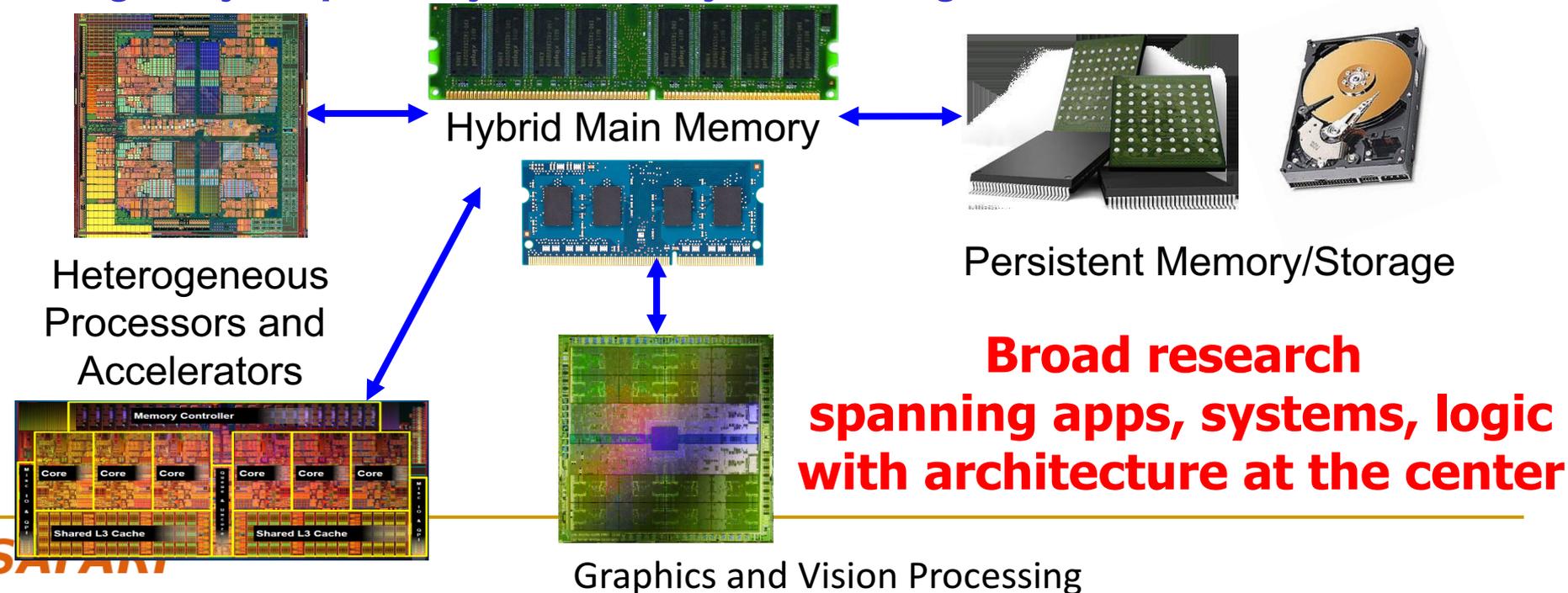
SAFARI

ETH zürich

Current Research Focus Areas

Research Focus: Computer architecture, HW/SW, bioinformatics

- **Memory and storage (DRAM, flash, emerging), interconnects**
- Heterogeneous & parallel systems, GPUs, systems for data analytics
- System/architecture interaction, new execution models, new interfaces
- Energy efficiency, fault tolerance, hardware security, performance
- Genome sequence analysis & assembly algorithms and architectures
- Biologically inspired systems & system design for bio/medicine



Four Key Current Directions

- Fundamentally **Secure/Reliable/Safe** Architectures
- Fundamentally **Energy-Efficient** Architectures
 - **Memory-centric** (Data-centric) Architectures
- Fundamentally **Low-Latency** Architectures
- Architectures for **Genomics, Medicine, Health**

In-Memory DNA Sequence Analysis

- Jeremie S. Kim, Damla Senol Cali, Hongyi Xin, Donghyuk Lee, Saugata Ghose, Mohammed Alser, Hasan Hassan, Oguz Ergin, Can Alkan, and Onur Mutlu, **"GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies"**
to appear in ***BMC Genomics***, 2018.
to also appear in *Proceedings of the 16th Asia Pacific Bioinformatics Conference (APBC)*, Yokohama, Japan, January 2018.
[arxiv.org Version \(pdf\)](#)

GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies

Jeremie S. Kim^{1,6*}, Damla Senol Cali¹, Hongyi Xin², Donghyuk Lee³, Saugata Ghose¹,
Mohammed Alser⁴, Hasan Hassan⁶, Oguz Ergin⁵, Can Alkan^{*4}, and Onur Mutlu^{*6,1}

Nanopore Sequencing Technology and Tools: Computational Analysis of the Current State, Bottlenecks, and Future Directions

**Damla Senol Cali^{1,*}, Jeremie Kim^{1,3}, Saugata Ghose¹, Can Alkan^{2*}
and Onur Mutlu^{3,1*}**

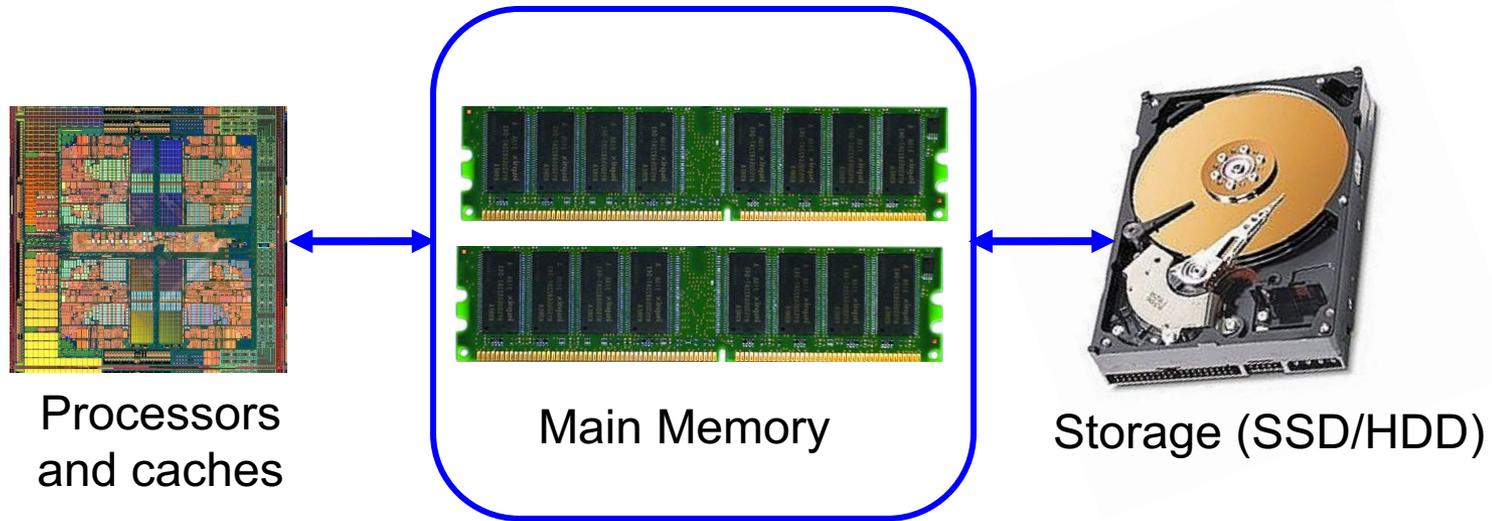
¹ Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

² Department of Computer Engineering, Bilkent University, Bilkent, Ankara, Turkey

³ Department of Computer Science, Systems Group, ETH Zürich, Zürich, Switzerland

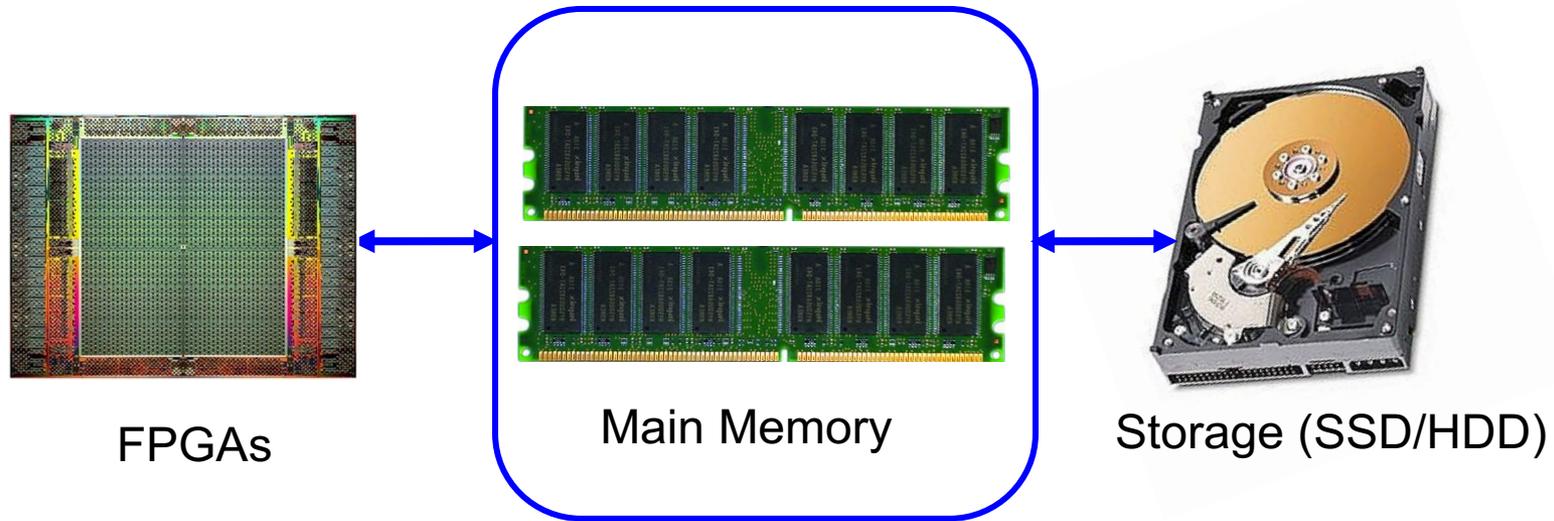
Rethinking Memory & Storage

The Main Memory System



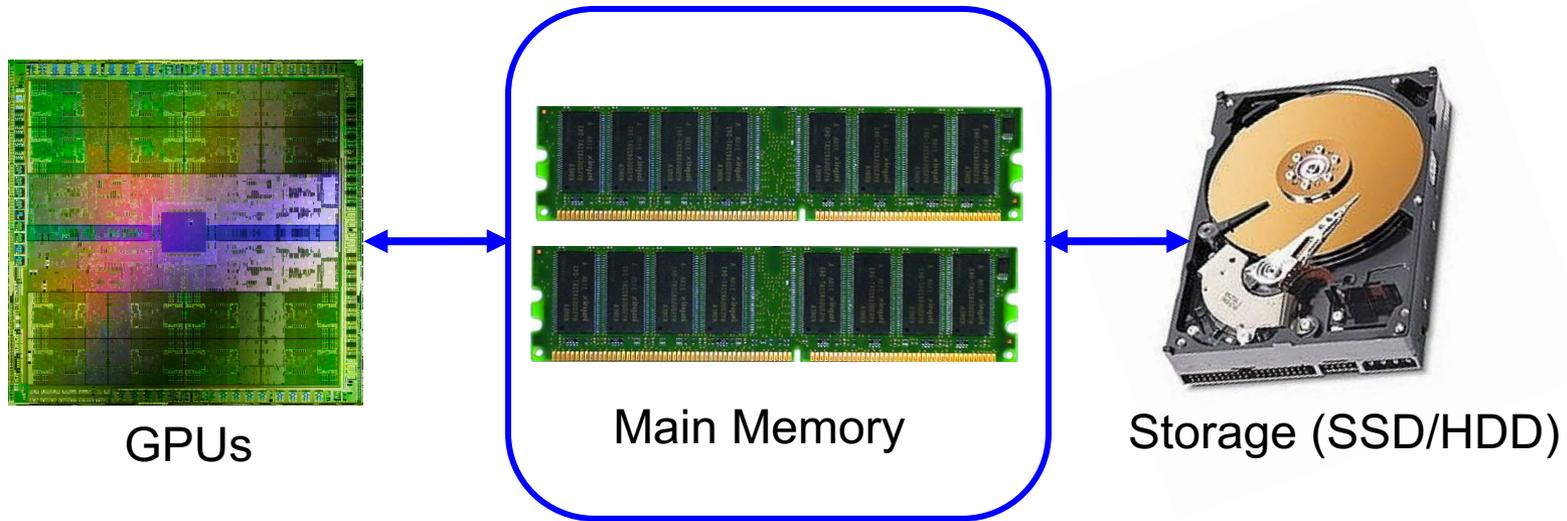
- **Main memory is a critical component of all computing systems:** server, mobile, embedded, desktop, sensor
- **Main memory system must scale** (in *size, technology, efficiency, cost, and management algorithms*) to maintain performance growth and technology scaling benefits

The Main Memory System



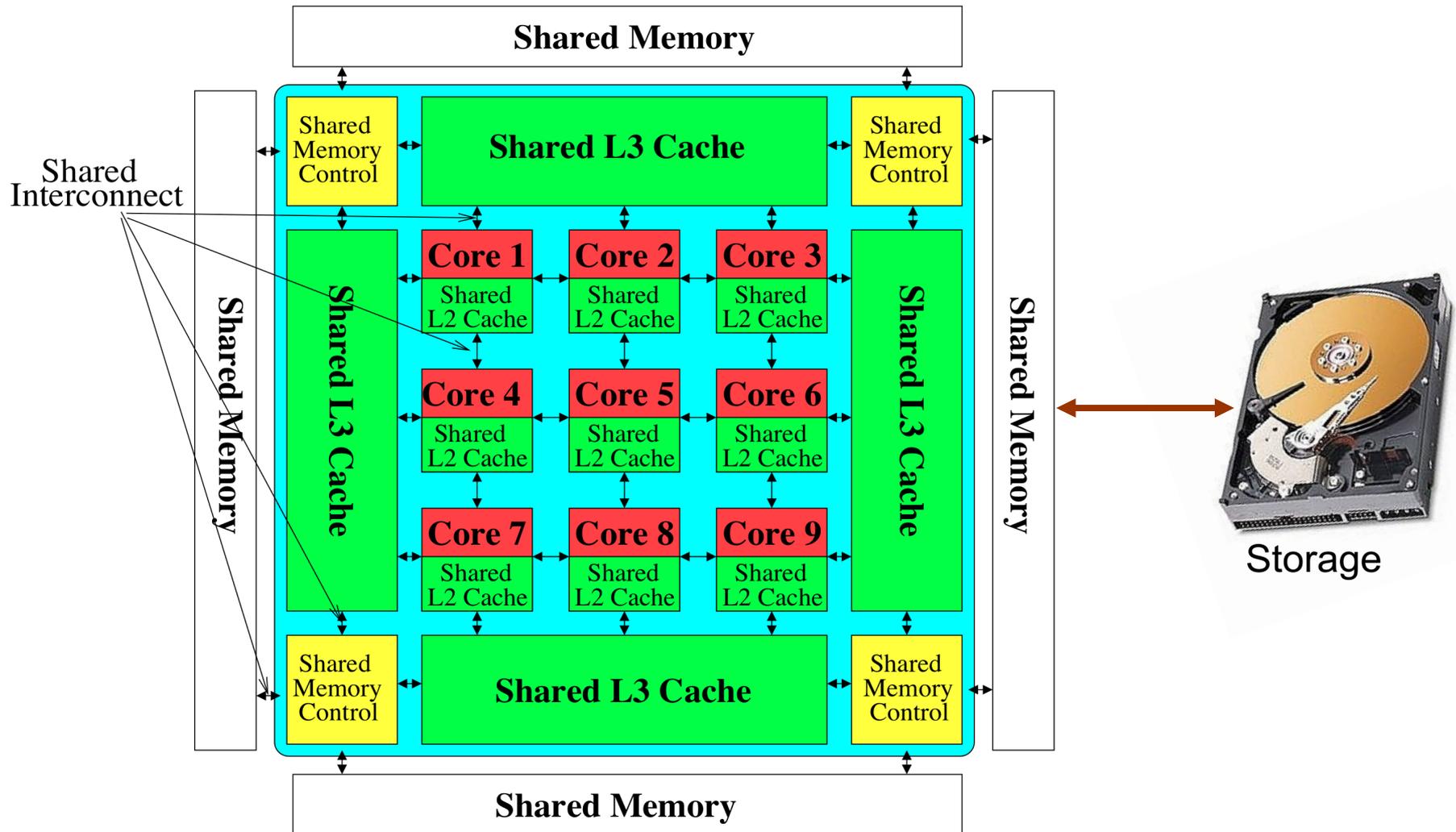
- Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor
- Main memory system must scale (in *size, technology, efficiency, cost, and management algorithms*) to maintain performance growth and technology scaling benefits

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



Most of the system is dedicated to storing and moving data

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

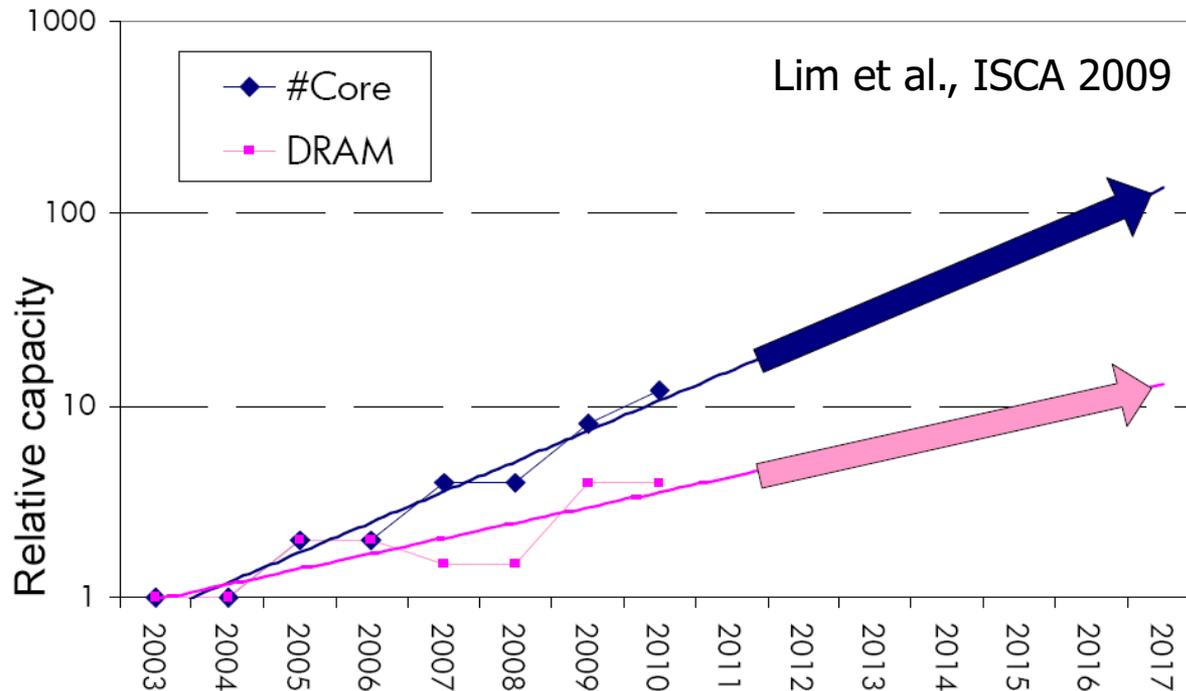
Major Trends Affecting Main Memory (I)

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

Example: The Memory Capacity Gap

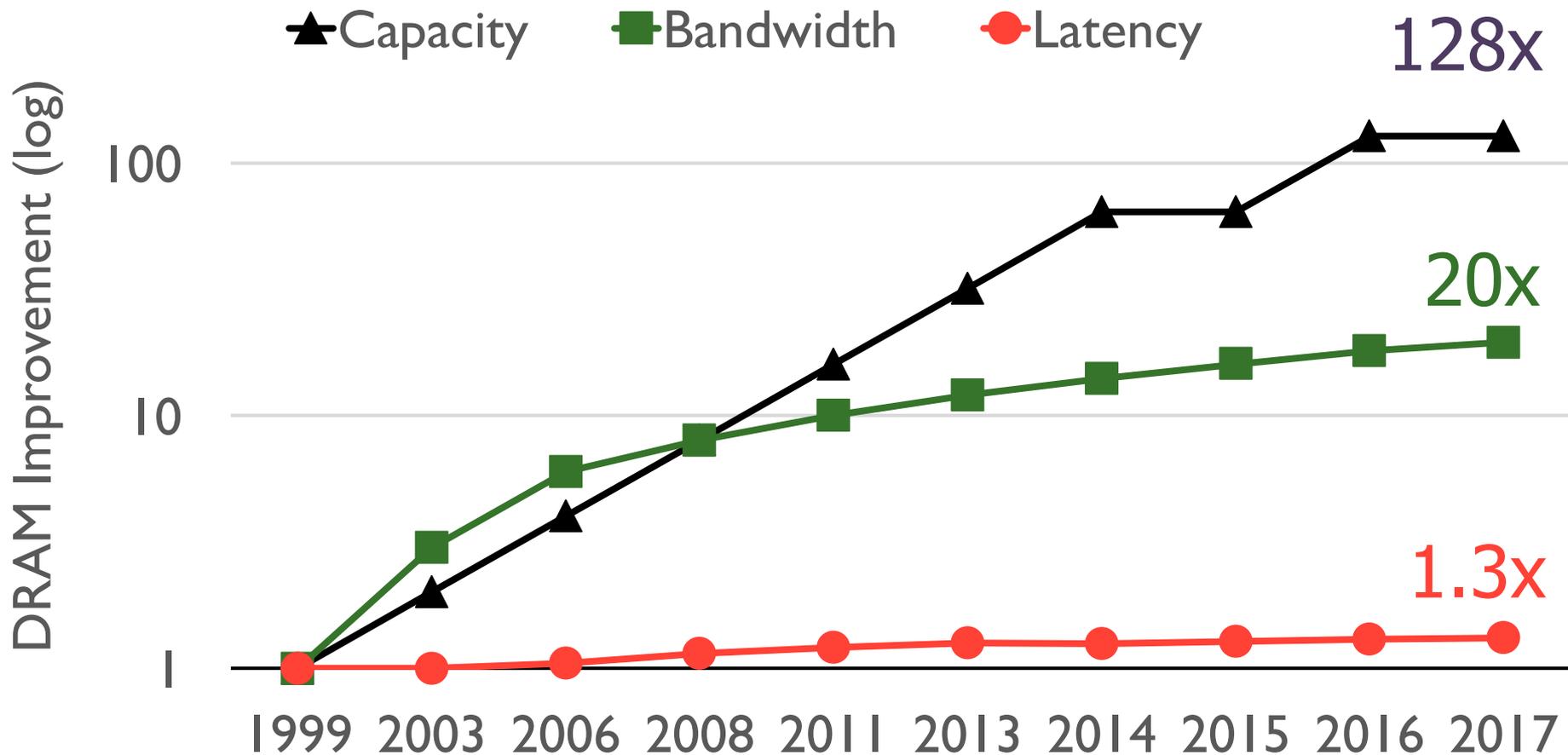
Core count doubling ~ every 2 years

DRAM DIMM capacity doubling ~ every 3 years



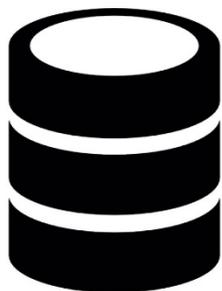
- *Memory capacity per core* expected to drop by 30% every two years
- Trends worse for *memory bandwidth per core*!

Example: Memory Bandwidth & Latency



Memory latency remains almost constant

DRAM Latency Is Critical for Performance



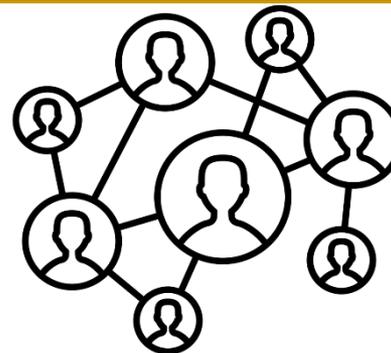
In-memory Databases

[Mao+, EuroSys'12;
Clapp+ (Intel), IISWC'15]



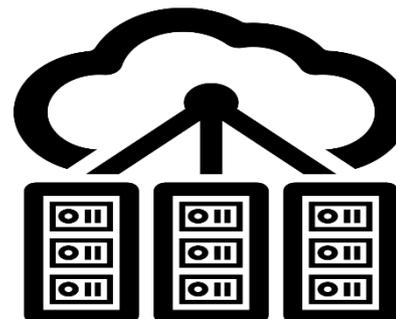
In-Memory Data Analytics

[Clapp+ (Intel), IISWC'15;
Awan+, BDCloud'15]



Graph/Tree Processing

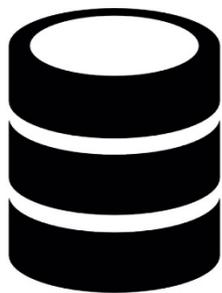
[Xu+, IISWC'12; Umuroglu+, FPL'15]



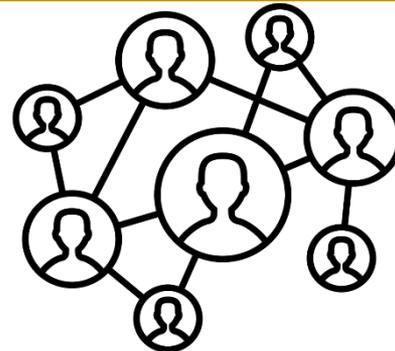
Datacenter Workloads

[Kanev+ (Google), ISCA'15]

DRAM Latency Is Critical for Performance



In-memory Databases



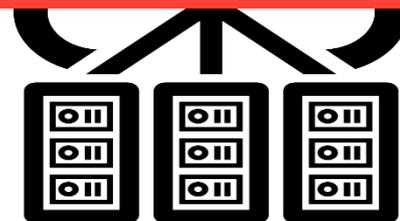
Graph/Tree Processing

Long memory latency → performance bottleneck



In-Memory Data Analytics

[Clapp+ (Intel), IISWC'15;
Awan+, BDCloud'15]



Datacenter Workloads

[Kanev+ (Google), ISCA'15]

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'03] >40% power in DRAM [Ware, HPCA'10][Paul,ISCA'15]
 - 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

Major Trends Affecting Main Memory (V)

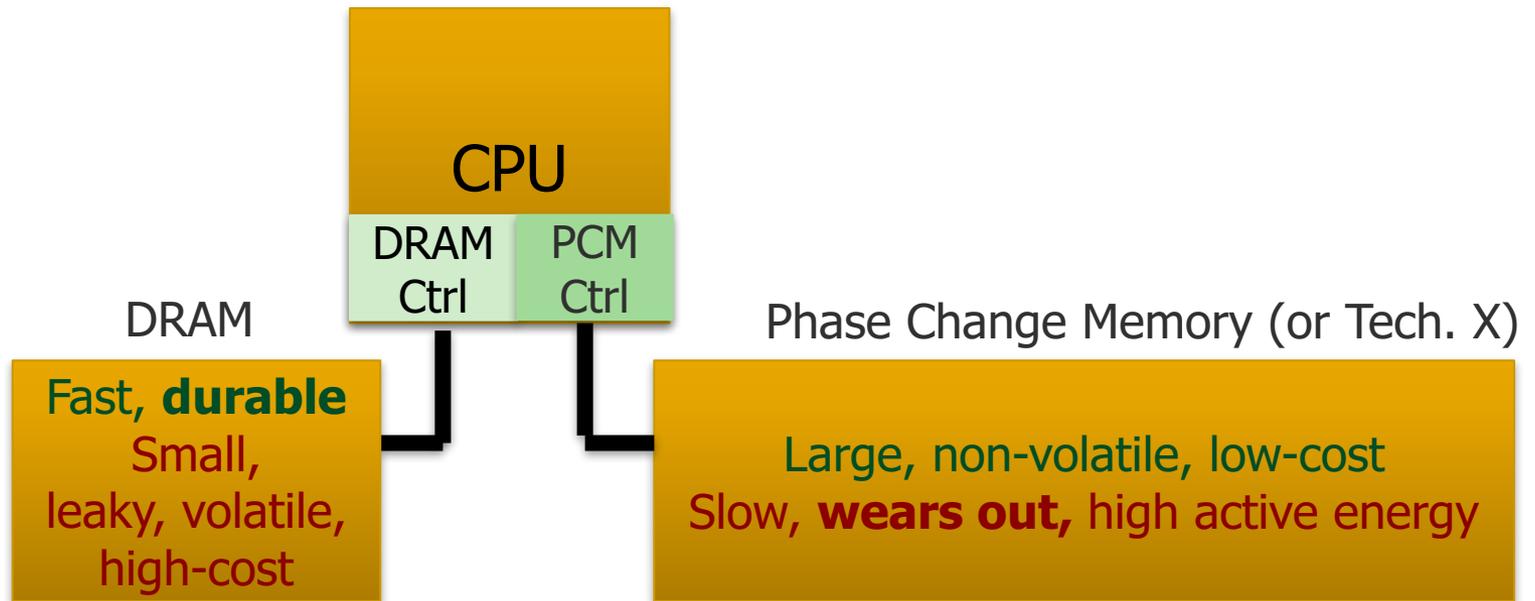
- DRAM scaling has already become increasingly difficult
 - Increasing cell leakage current, reduced cell reliability, increasing manufacturing difficulties [Kim+ ISCA 2014], [Liu+ ISCA 2013], [Mutlu IMW 2013], [Mutlu DATE 2017]
 - **Difficult to significantly improve capacity, energy**
- **Emerging memory technologies** are promising

Major Trends Affecting Main Memory (V)

- DRAM scaling has already become increasingly difficult
 - Increasing cell leakage current, reduced cell reliability, increasing manufacturing difficulties [Kim+ ISCA 2014], [Liu+ ISCA 2013], [Mutlu IMW 2013], [Mutlu DATE 2017]
 - **Difficult to significantly improve capacity, energy**
- **Emerging memory technologies** are promising

3D-Stacked DRAM	higher bandwidth	smaller capacity
Reduced-Latency DRAM (e.g., RL/TL-DRAM, FLY-RAM)	lower latency	higher cost
Low-Power DRAM (e.g., LPDDR3, LPDDR4, Voltron)	lower power	higher latency higher cost
Non-Volatile Memory (NVM) (e.g., PCM, STTRAM, ReRAM, 3D Xpoint)	larger capacity	higher latency higher dynamic power lower endurance

Major Trend: Hybrid Main Memory



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.

Main Memory Needs Intelligent Controllers

Agenda

- Major Trends Affecting Main Memory
- The Memory Scaling Problem and Solution Directions
 - New Memory Architectures
 - Enabling Emerging Technologies
- Cross-Cutting Principles
- Summary

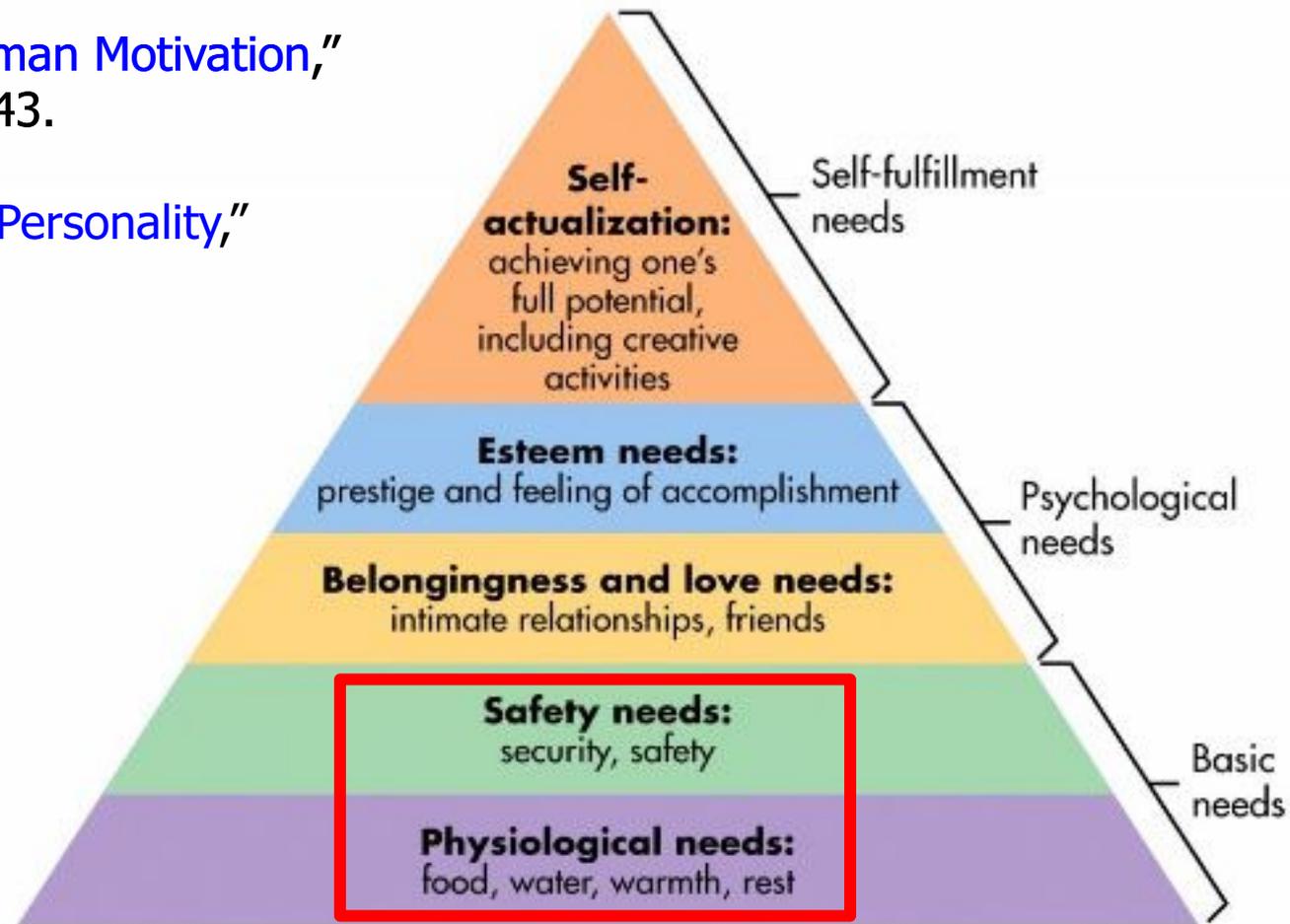
Three Key Issues in Future Platforms

- Fundamentally **Secure/Reliable/Safe** Architectures
- Fundamentally **Energy-Efficient** Architectures
 - **Memory-centric** (Data-centric) Architectures
- Fundamentally **Low Latency** Architectures

Maslow's (Human) Hierarchy of Needs

Maslow, "A Theory of Human Motivation,"
Psychological Review, 1943.

Maslow, "Motivation and Personality,"
Book, 1954-1970.



- We need to start with **reliability and security**...

How Reliable/Secure/Safe is This Bridge?



Collapse of the “Galloping Gertie”



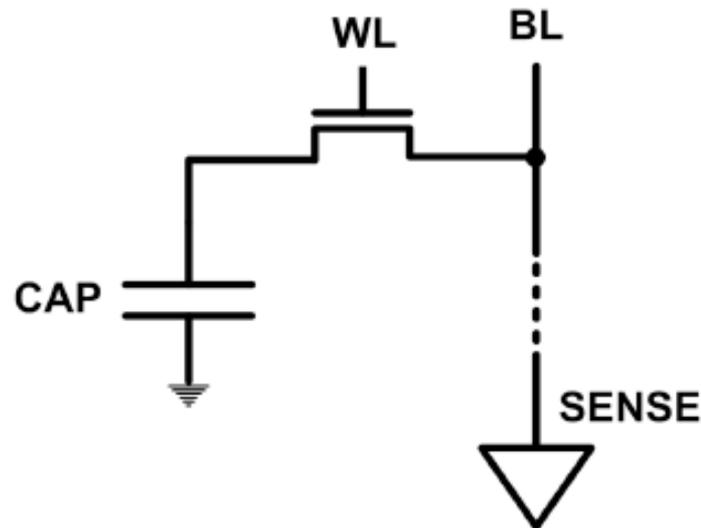
How Secure Are These People?



Security is about preventing unforeseen consequences

The DRAM Scaling Problem

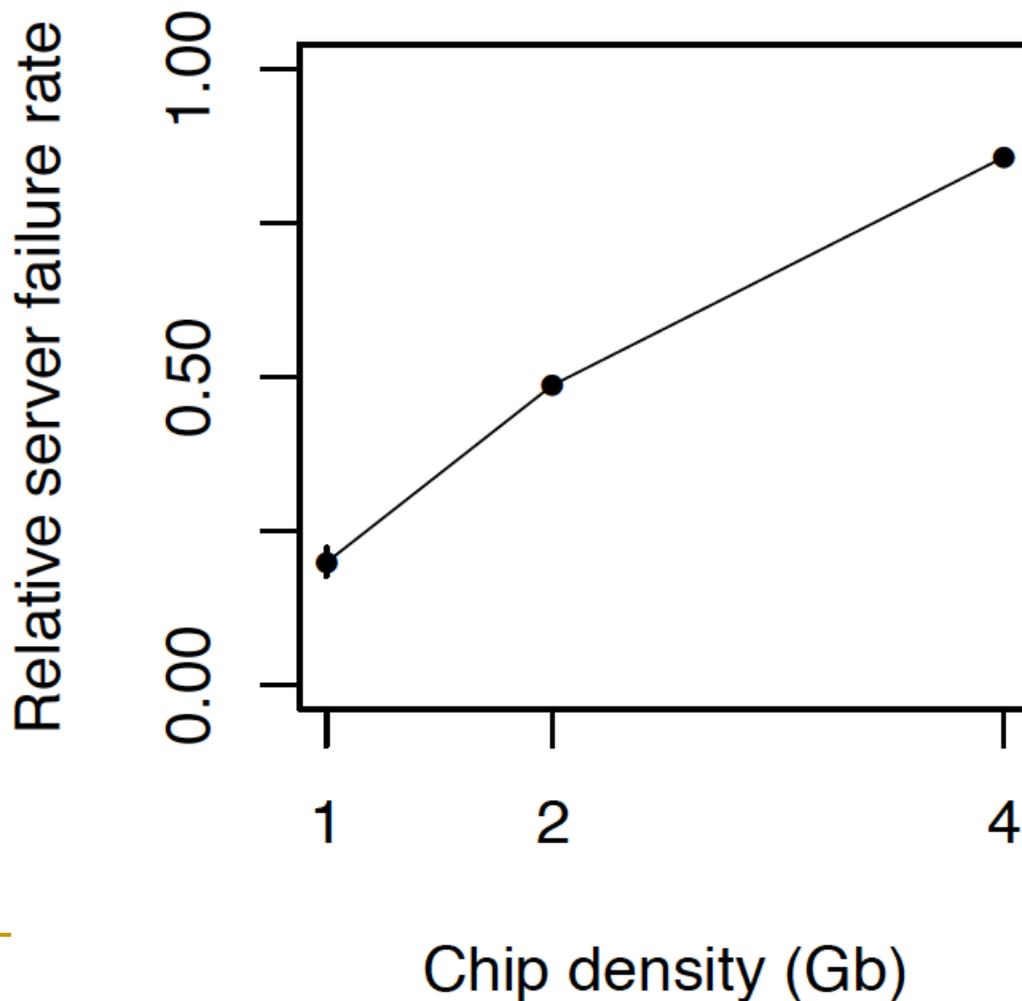
- DRAM stores charge in a capacitor (charge-based memory)
 - Capacitor must be large enough for reliable sensing
 - Access transistor should be large enough for low leakage and high retention time
 - Scaling beyond 40-35nm (2013) is challenging [ITRS, 2009]



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

As Memory Scales, It Becomes Unreliable

- Data from all of Facebook's servers worldwide
- Meza+, "Revisiting Memory Errors in Large-Scale Production Data Centers," DSN'15.



*Intuition:
quadratic
increase
in
capacity*

Large-Scale Failure Analysis of DRAM Chips

- Analysis and modeling of memory errors found in all of Facebook's server fleet
- 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](#)]

Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field

Justin Meza Qiang Wu* Sanjeev Kumar* Onur Mutlu
Carnegie Mellon University * Facebook, Inc.

Infrastructures to Understand Such Issues



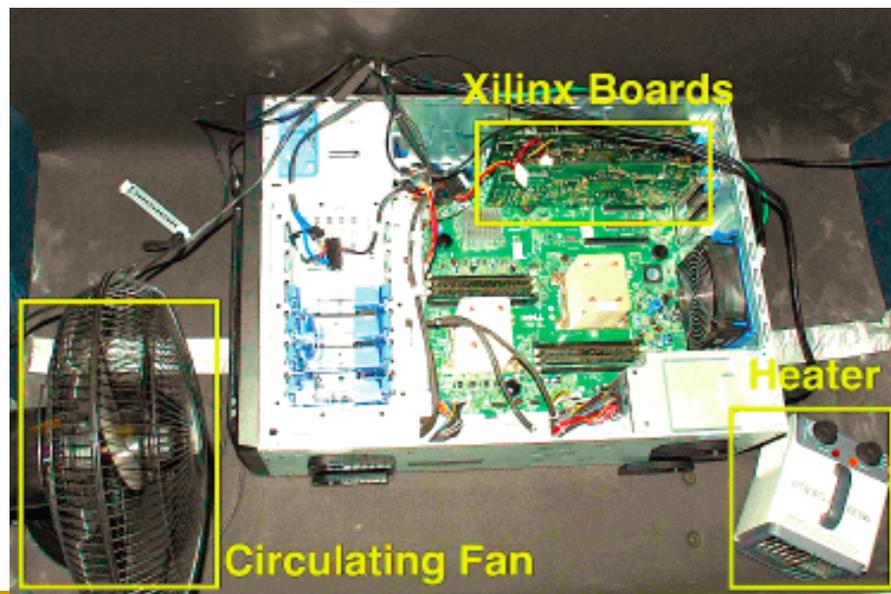
An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

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)



Infrastructures to Understand Such Issues

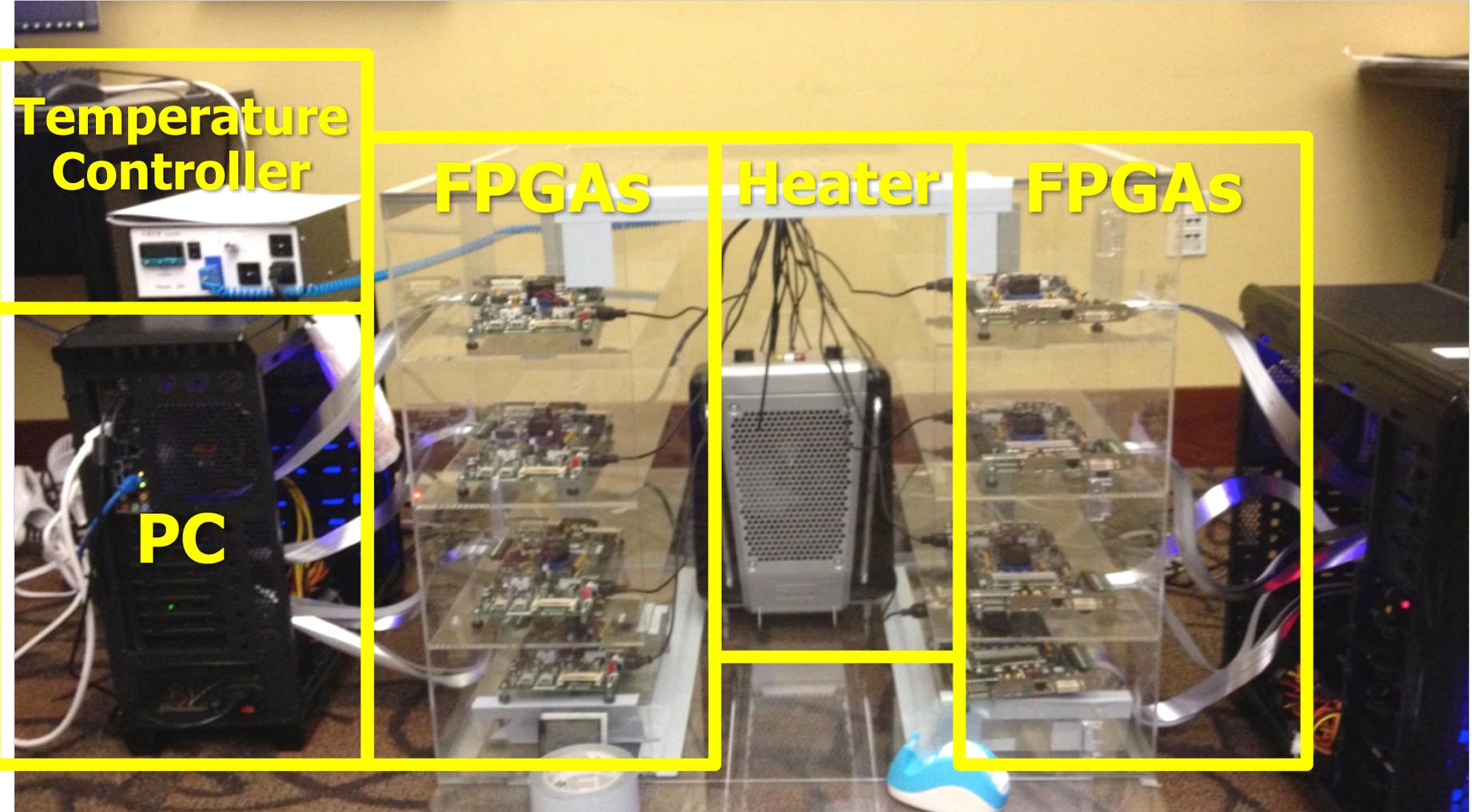
Temperature
Controller

FPGAs

Heater

FPGAs

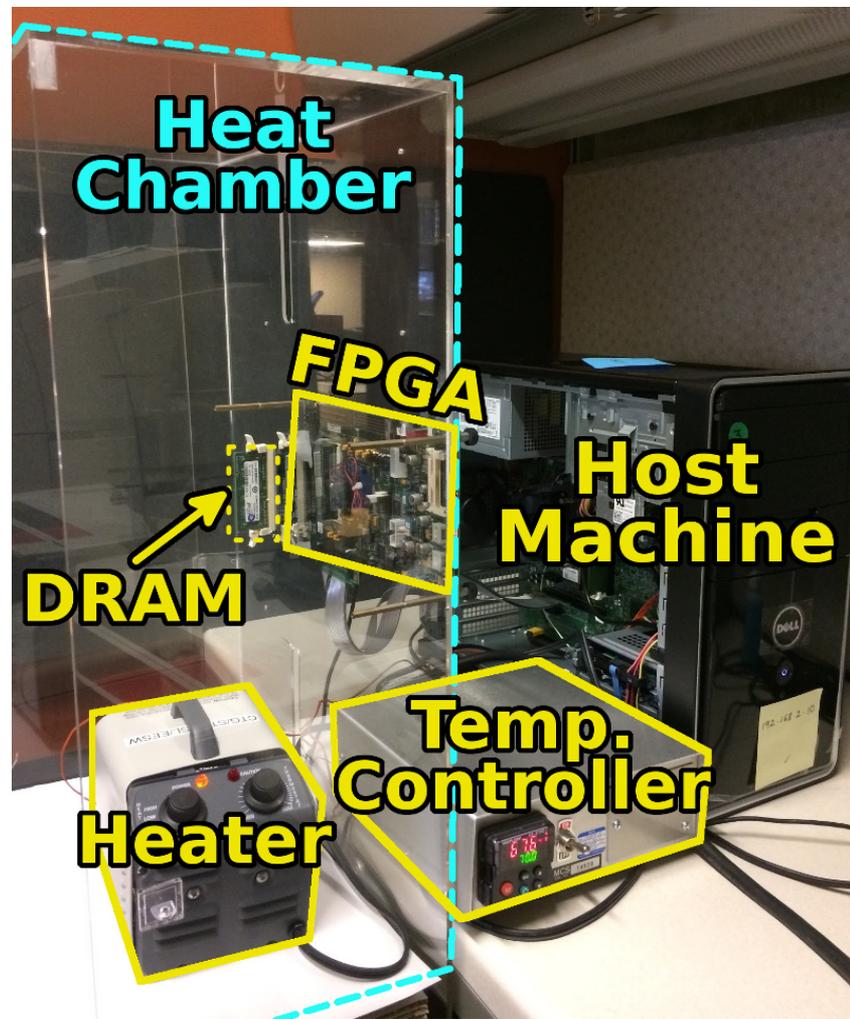
PC



SoftMC: Open Source DRAM Infrastructure

- Hasan Hassan et al., "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies," HPCA 2017.

- Flexible
- Easy to Use (C++ API)
- Open-source
github.com/CMU-SAFARI/SoftMC



- <https://github.com/CMU-SAFARI/SoftMC>

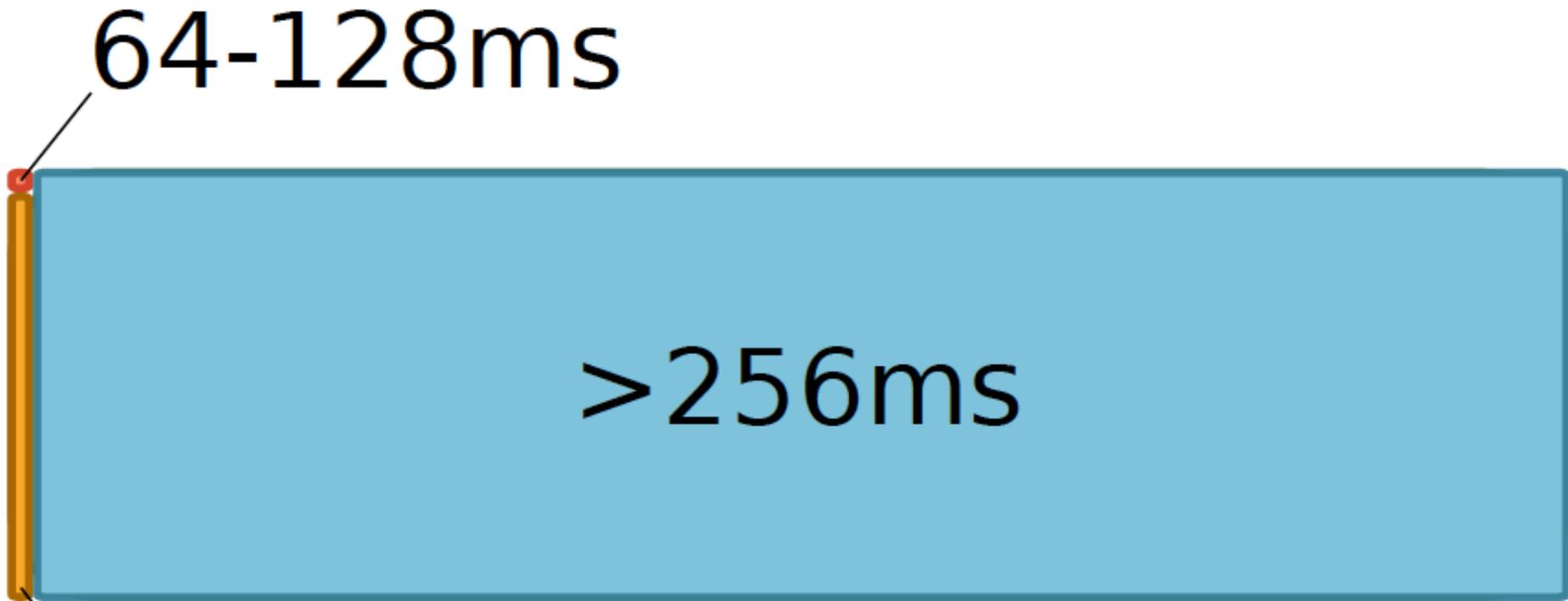
SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan^{1,2,3} Nandita Vijaykumar³ Samira Khan^{4,3} Saugata Ghose³ Kevin Chang³
Gennady Pekhimenko^{5,3} Donghyuk Lee^{6,3} Oguz Ergin² Onur Mutlu^{1,3}

¹*ETH Zürich* ²*TOBB University of Economics & Technology* ³*Carnegie Mellon University*
⁴*University of Virginia* ⁵*Microsoft Research* ⁶*NVIDIA Research*

Data Retention in Memory [Liu et al., ISCA 2013]

- Retention Time Profile of DRAM looks like this:



Location dependent
Stored value pattern dependent
Time dependent

A Curious Discovery [Kim et al., ISCA 2014]

One can
predictably induce errors
in most DRAM memory chips

DRAM RowHammer

A simple hardware failure mechanism
can create a widespread
system security vulnerability

WIRED

Forget Software—Now Hackers Are Exploiting Physics

BUSINESS

CULTURE

DESIGN

GEAR

SCIENCE

ANDY GREENBERG SECURITY 08.31.16 7:00 AM

SHARE



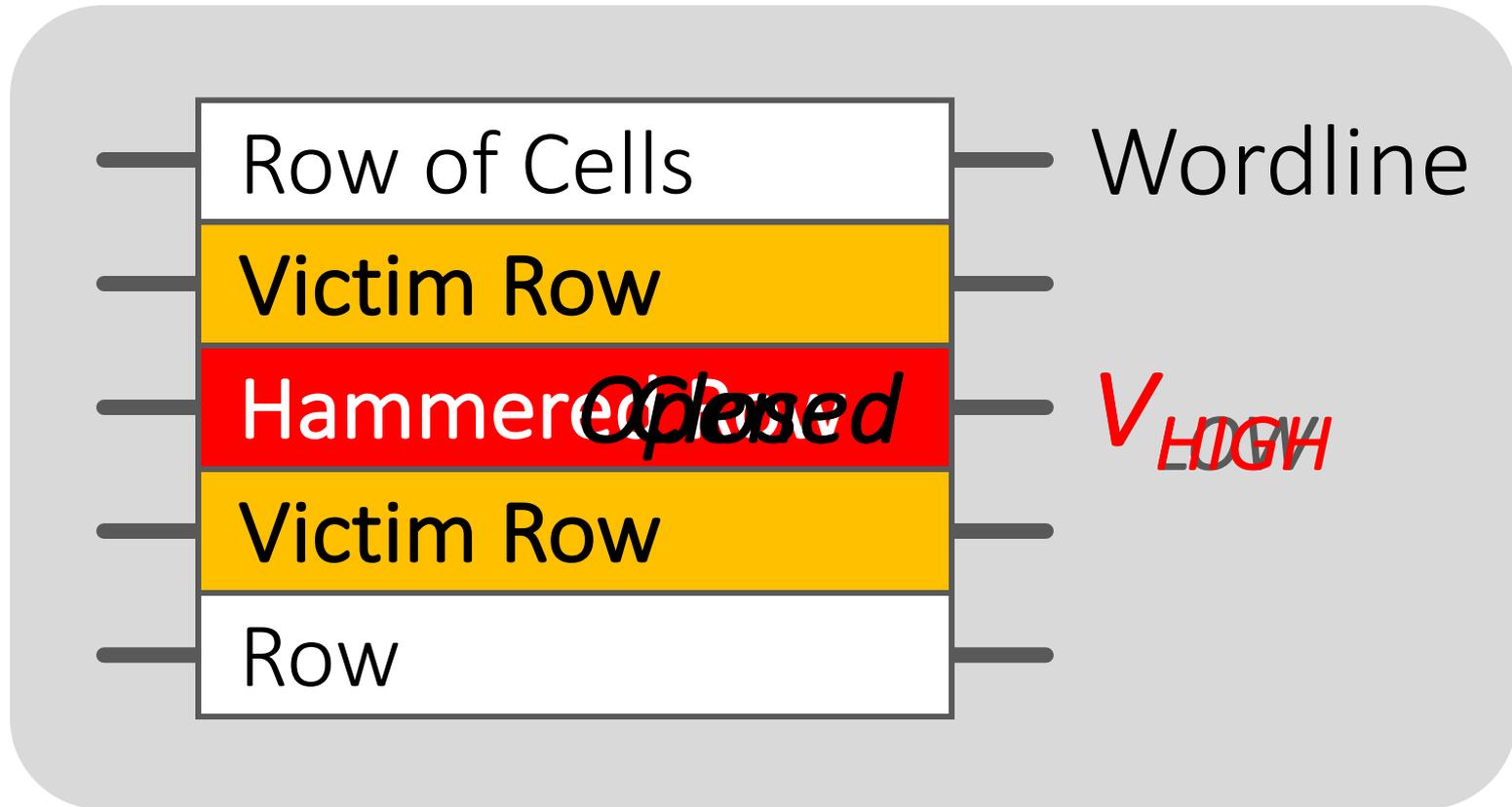
SHARE
18276



TWEET

FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS

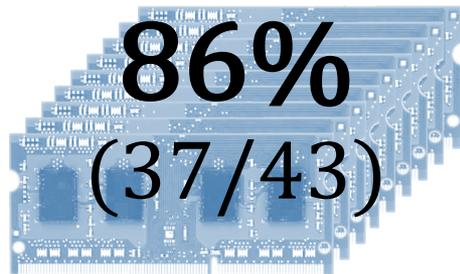
Modern DRAM is Prone to Disturbance Errors



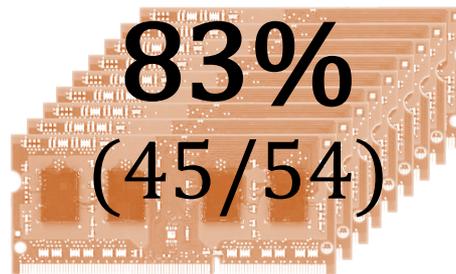
Repeatedly reading a row enough times (before memory gets refreshed) induces **disturbance errors** in adjacent rows in **most real DRAM chips you can buy today**

Most DRAM Modules Are at Risk

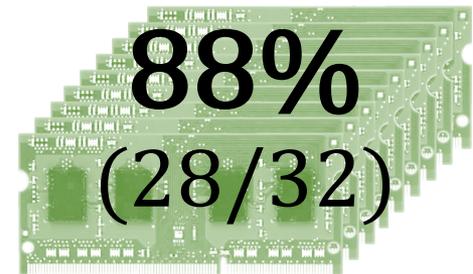
A company



B company



C company

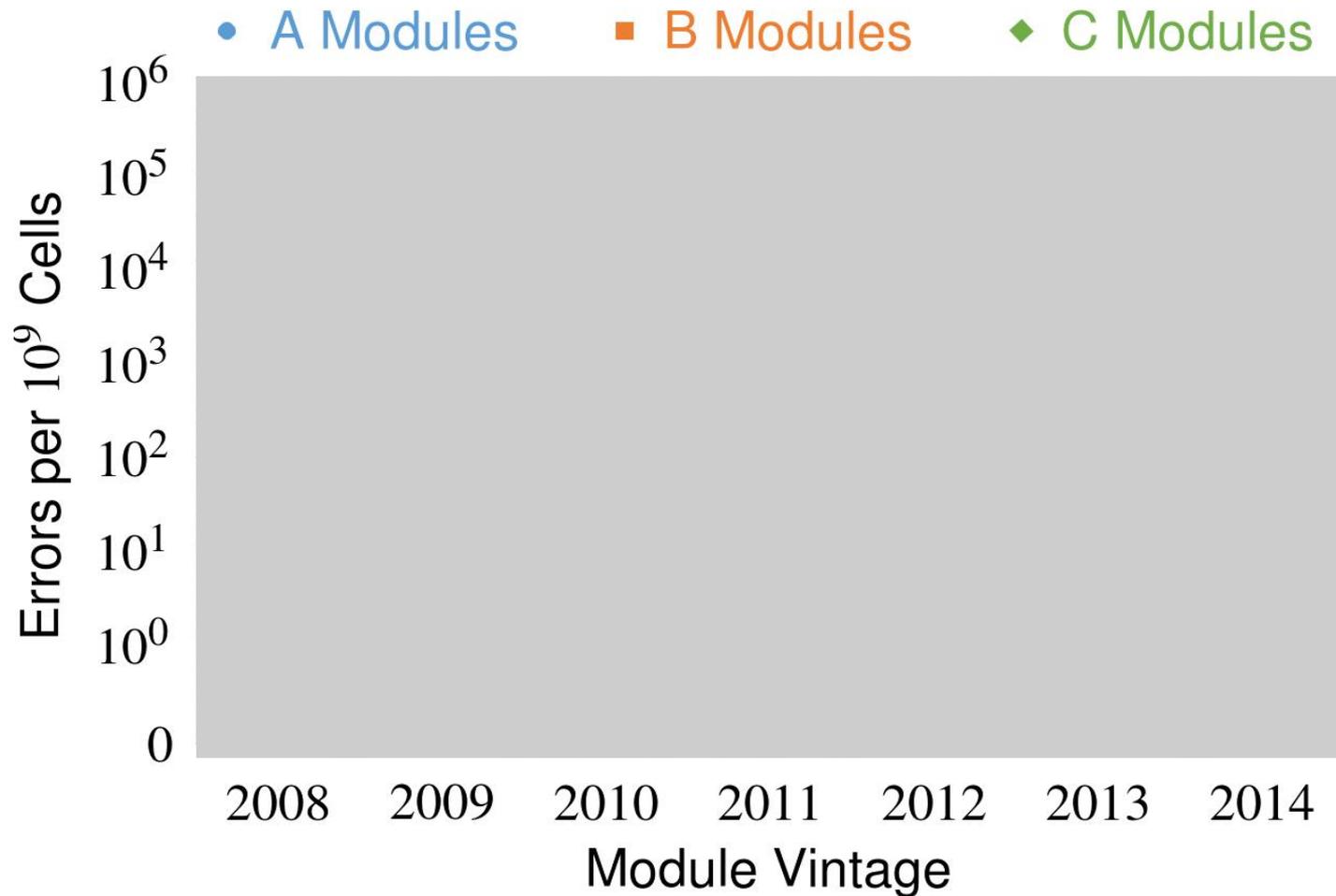


Up to
 1.0×10^7
errors

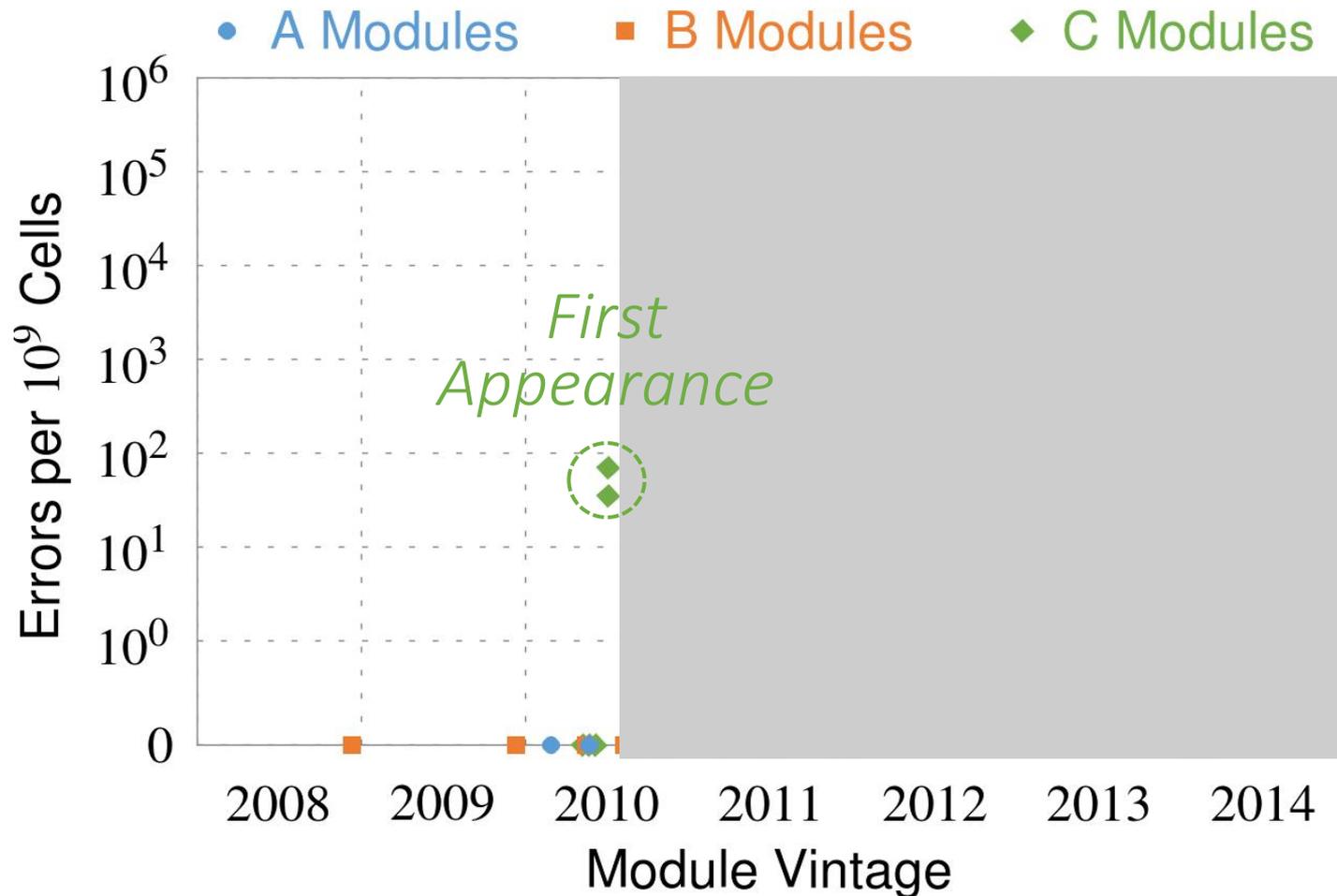
Up to
 2.7×10^6
errors

Up to
 3.3×10^5
errors

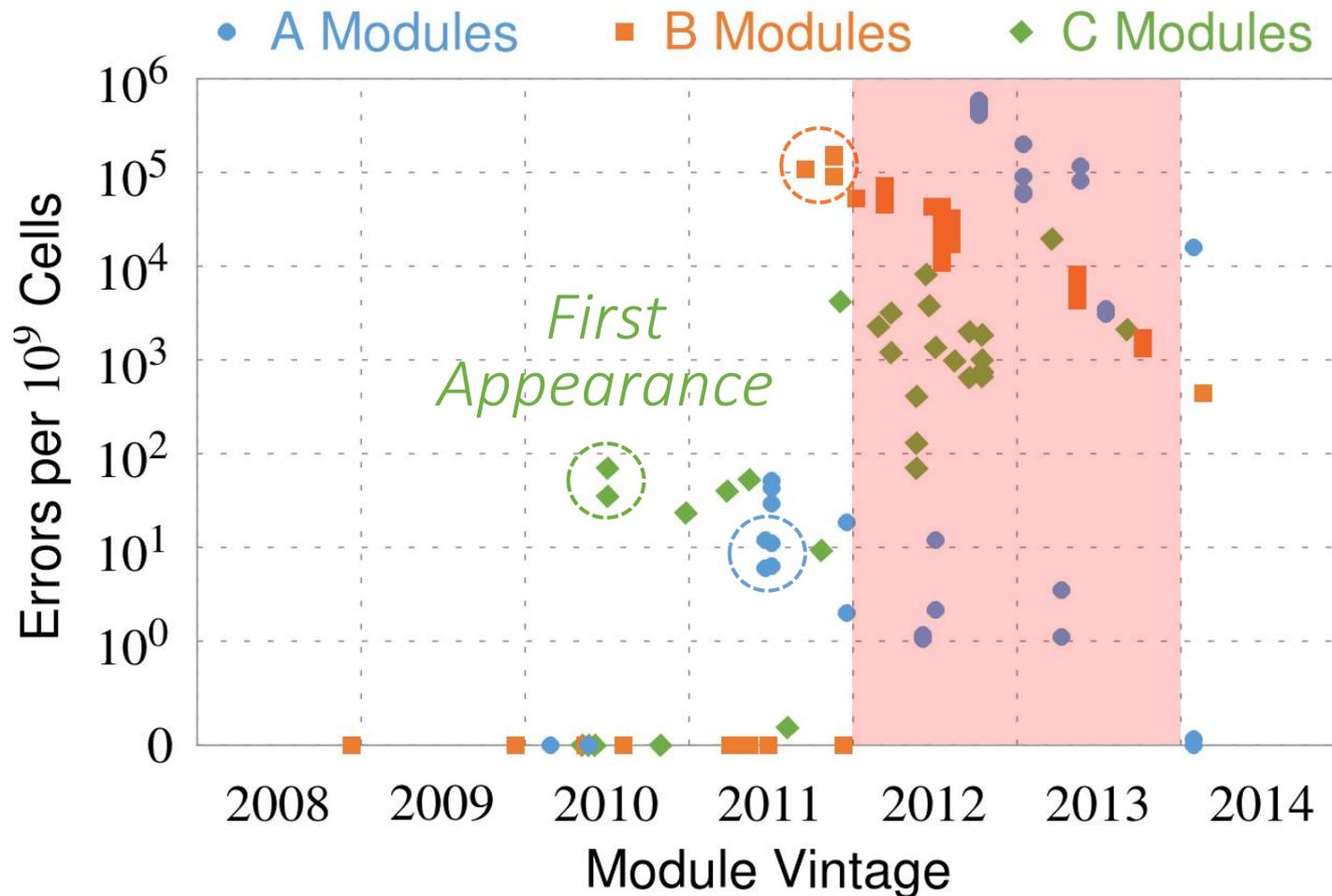
Recent DRAM Is More Vulnerable



Recent DRAM Is More Vulnerable

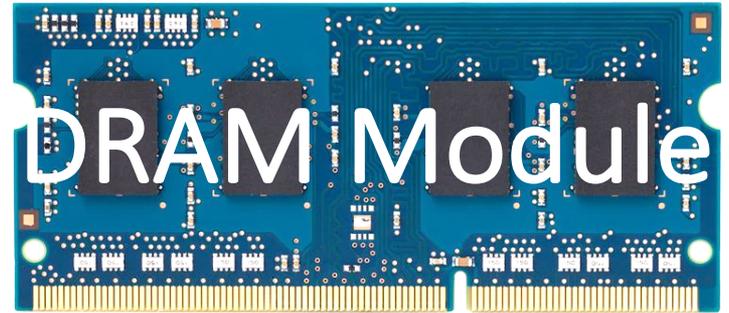
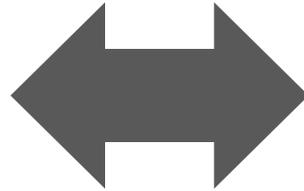
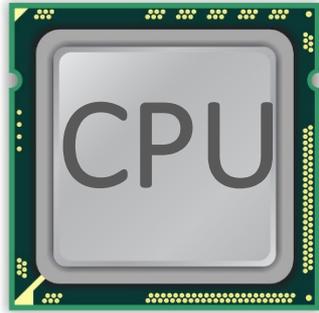


Recent DRAM Is More Vulnerable

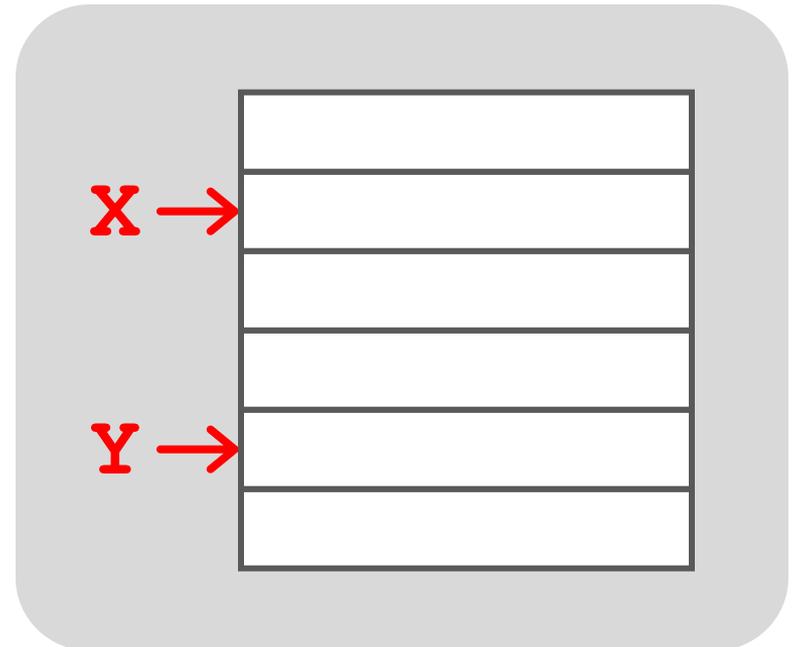


All modules from 2012–2013 are vulnerable

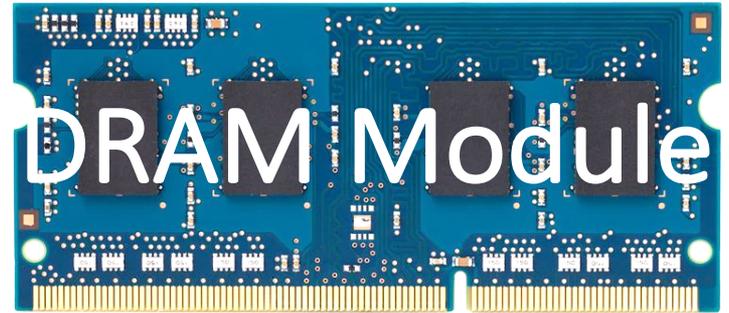
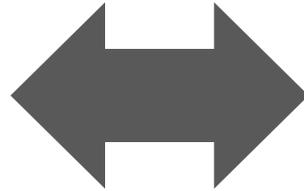
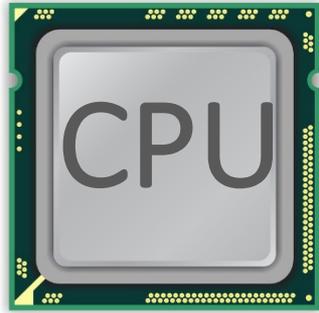
A Simple Program Can Induce Many Errors



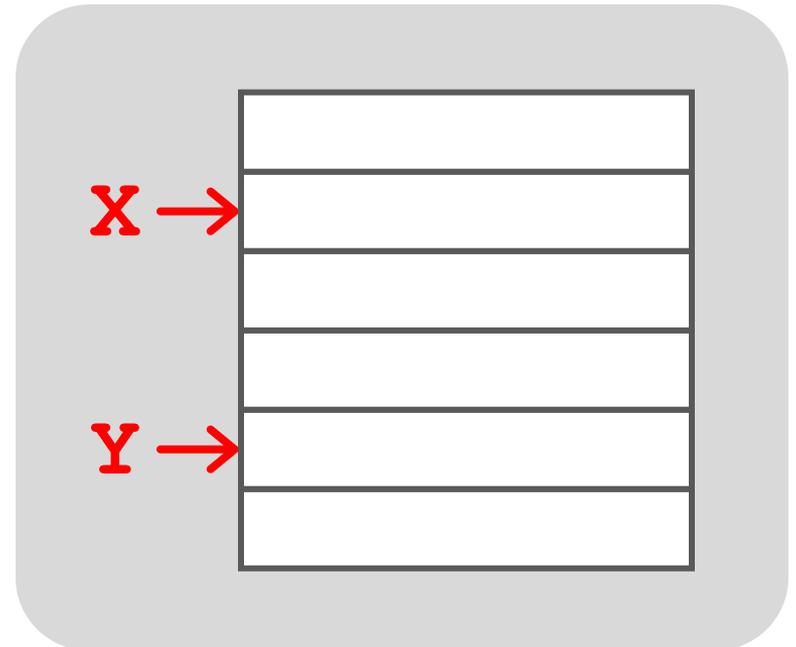
```
loop:  
  mov (X), %eax  
  mov (Y), %ebx  
  clflush (X)  
  clflush (Y)  
  mfence  
  jmp loop
```



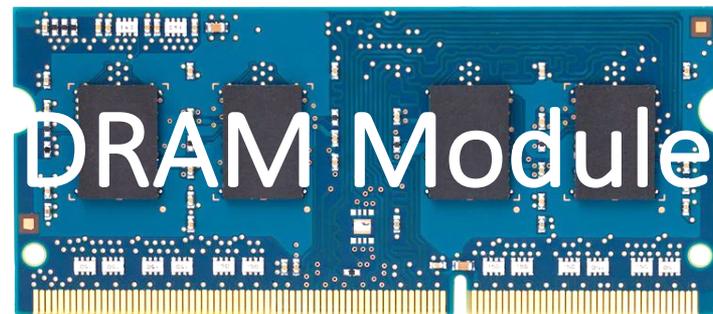
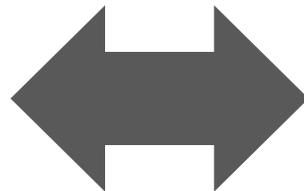
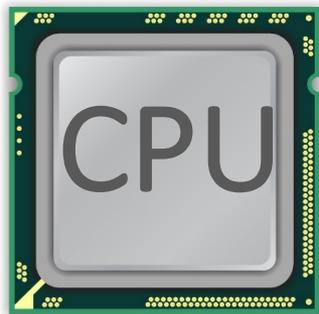
A Simple Program Can Induce Many Errors



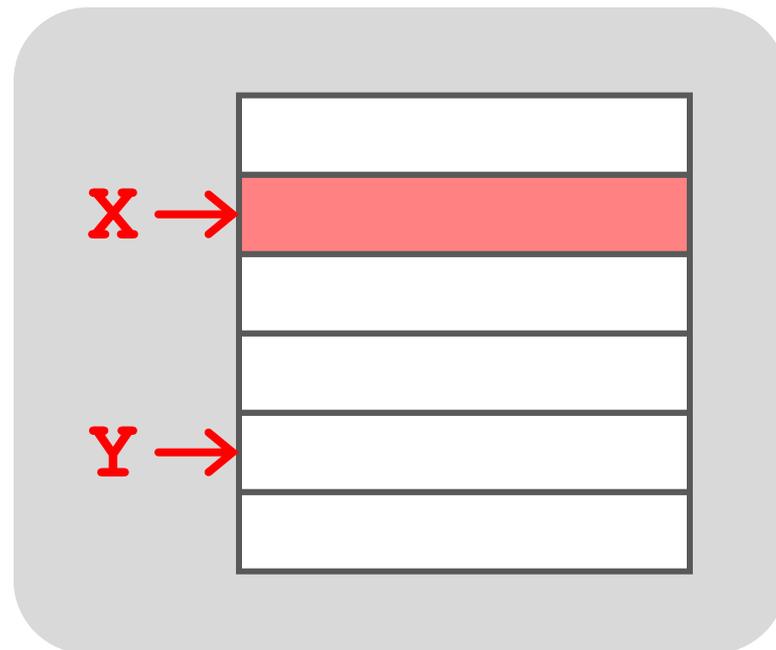
1. Avoid *cache hits*
 - Flush **X** from cache
2. Avoid *row hits* to **X**
 - Read **Y** in another row



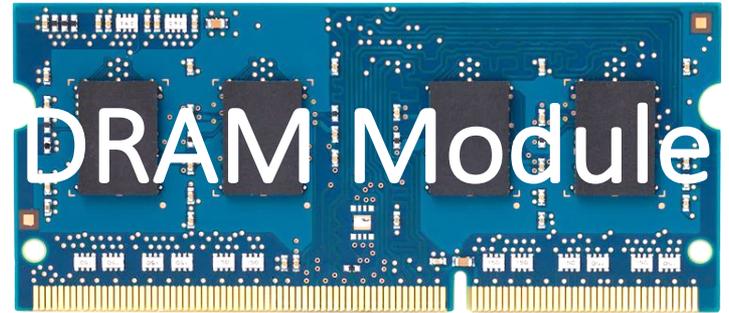
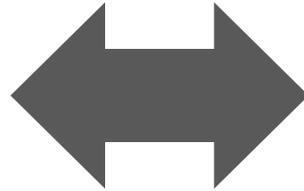
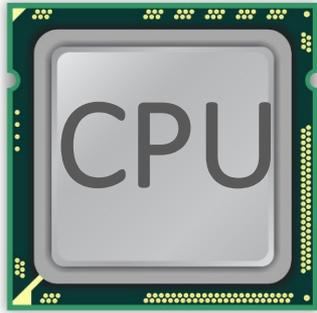
A Simple Program Can Induce Many Errors



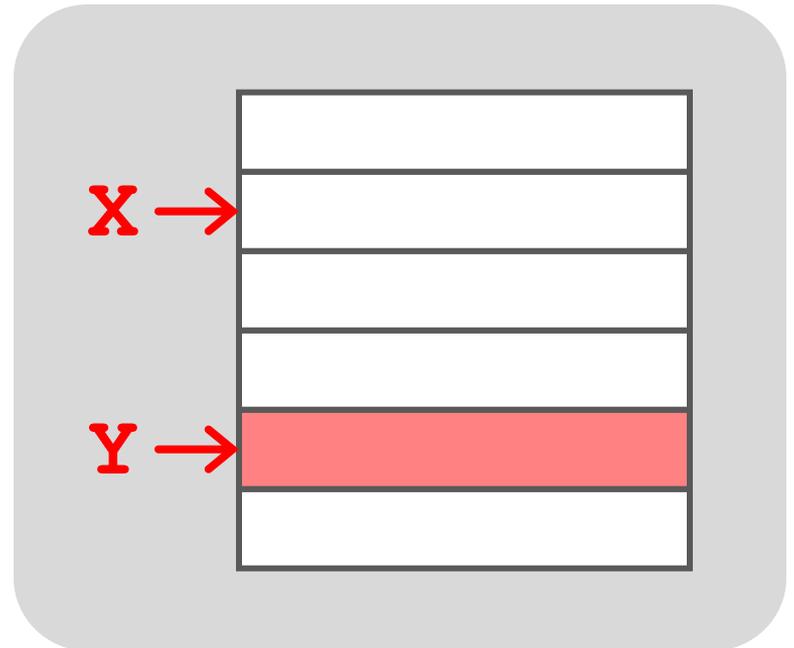
```
loop:  
  mov  (X), %eax  
  mov  (Y), %ebx  
  clflush (X)  
  clflush (Y)  
  mfence  
  jmp  loop
```



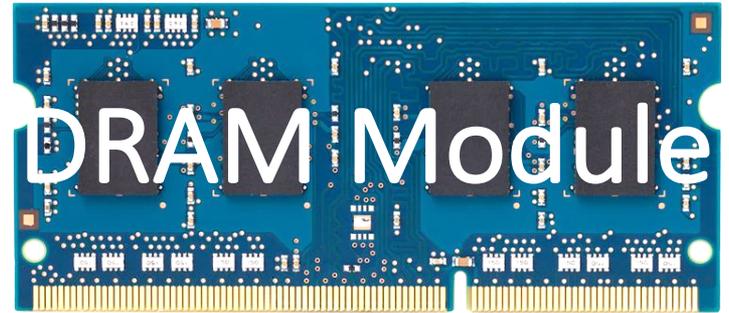
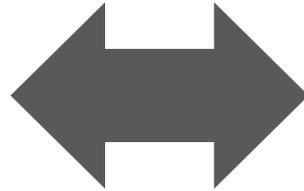
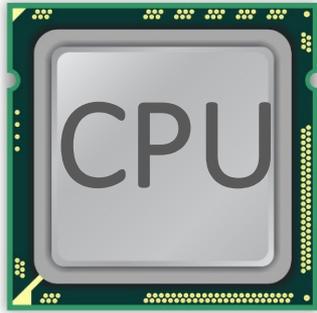
A Simple Program Can Induce Many Errors



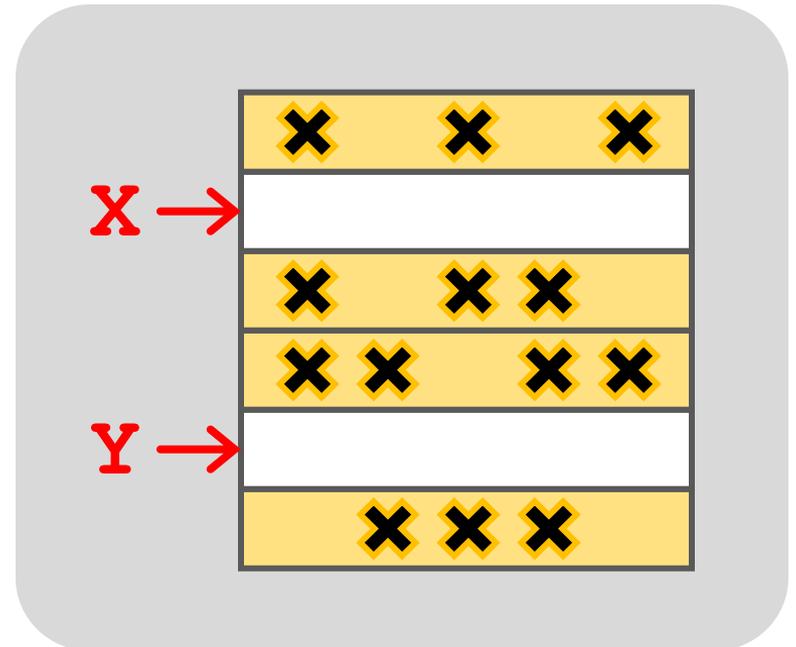
```
loop:  
  mov  (X), %eax  
  mov  (Y), %ebx  
  clflush (X)  
  clflush (Y)  
  mfence  
  jmp  loop
```



A Simple Program Can Induce Many Errors



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



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

One Can Take Over an Otherwise-Secure System

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

Project Zero

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

News and updates from the Project Zero team at Google

[Exploiting the DRAM rowhammer bug to
gain kernel privileges](#) (Seaborn+, 2015)

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

RowHammer Security Attack Example

- “Rowhammer” is a problem with some recent DRAM devices in which repeatedly accessing a row of memory can cause bit flips in adjacent rows (Kim et al., ISCA 2014).
 - Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)
- We tested a selection of laptops and found that a subset of them exhibited the problem.
- We built two working privilege escalation exploits that use this effect.
 - Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn+, 2015)
- One exploit uses rowhammer-induced bit flips to gain kernel privileges on x86-64 Linux when run as an unprivileged userland process.
- When run on a machine vulnerable to the rowhammer problem, the process was able to induce bit flips in page table entries (PTEs).
- It was able to use this to gain write access to its own page table, and hence gain read-write access to all of physical memory.

Security Implications



Rowhammer

It's like breaking into an apartment by repeatedly slamming a neighbor's door until the vibrations open the door you were after

More Security Implications

“We can gain unrestricted access to systems of website visitors.”

www.iaik.tugraz.at ■

Not there yet, but ...



ROOT privileges for web apps!

29

Daniel Gruss (@lavados), Clémentine Maurice (@BloodyTangerine),
December 28, 2015 — 32c3, Hamburg, Germany



GATED
COMMUNITIES

Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript (DIMVA'16)

More Security Implications

"Can gain control of a smart phone deterministically"



Drammer: Deterministic Rowhammer
Attacks on Mobile Platforms, CCS'16 ⁵⁵

More Security Implications?



Apple's Patch for RowHammer

- <https://support.apple.com/en-gb/HT204934>

Available for: OS X Mountain Lion v10.8.5, OS X Mavericks v10.9.5

Impact: A malicious application may induce memory corruption to escalate privileges

Description: A disturbance error, also known as Rowhammer, exists with some DDR3 RAM that could have led to memory corruption. This issue was mitigated by increasing memory refresh rates.

CVE-ID

CVE-2015-3693 : Mark Seaborn and Thomas Dullien of Google, working from original research by Yoongu Kim et al (2014)

HP, Lenovo, and other vendors released similar patches

Our Solution to RowHammer

- PARA: *Probabilistic Adjacent Row Activation*
- Key Idea
 - After closing a row, we activate (i.e., refresh) one of its neighbors with a low probability: $p = 0.005$
- Reliability Guarantee
 - When $p=0.005$, errors in one year: 9.4×10^{-14}
 - By adjusting the value of p , we can vary the strength of protection against errors

More on RowHammer Analysis

- 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](#)]

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

Yoongu Kim¹ Ross Daly* Jeremie Kim¹ Chris Fallin* Ji Hye Lee¹
Donghyuk Lee¹ Chris Wilkerson² Konrad Lai Onur Mutlu¹

¹Carnegie Mellon University ²Intel Labs

Future of Memory Reliability

- Onur Mutlu,
"The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"
Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (DATE), Lausanne, Switzerland, March 2017.
[[Slides \(pptx\)](#) ([pdf](#))]

The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
<https://people.inf.ethz.ch/omutlu>

Industry Is Writing Papers About It, Too

DRAM Process Scaling Challenges

❖ Refresh

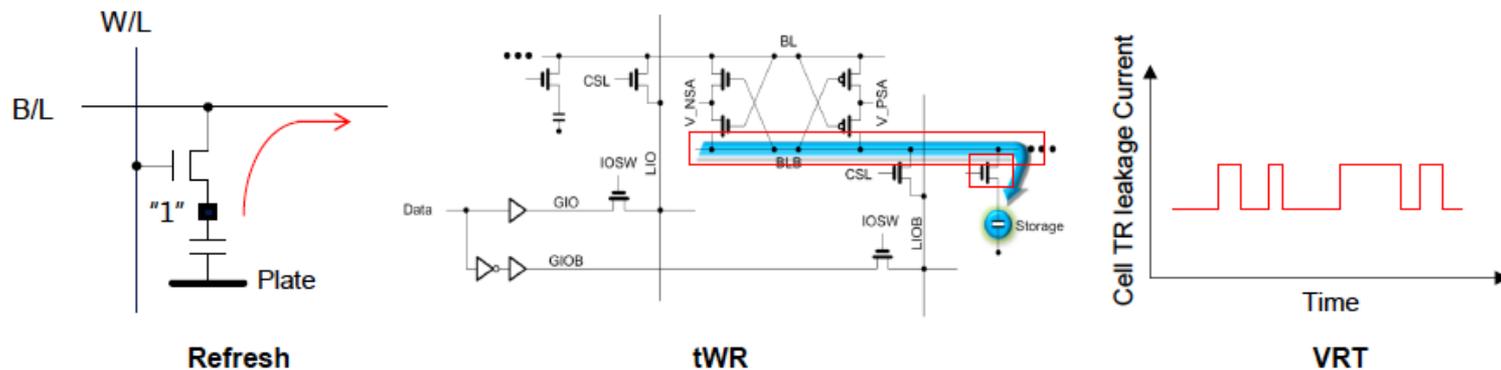
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance
- Leakage current of cell access transistors increasing

❖ t_{WR}

- Contact resistance between the cell capacitor and access transistor increasing
- On-current of the cell access transistor decreasing
- Bit-line resistance increasing

❖ VRT

- Occurring more frequently with cell capacitance decreasing



Call for Intelligent Memory Controllers

DRAM Process Scaling Challenges

❖ Refresh

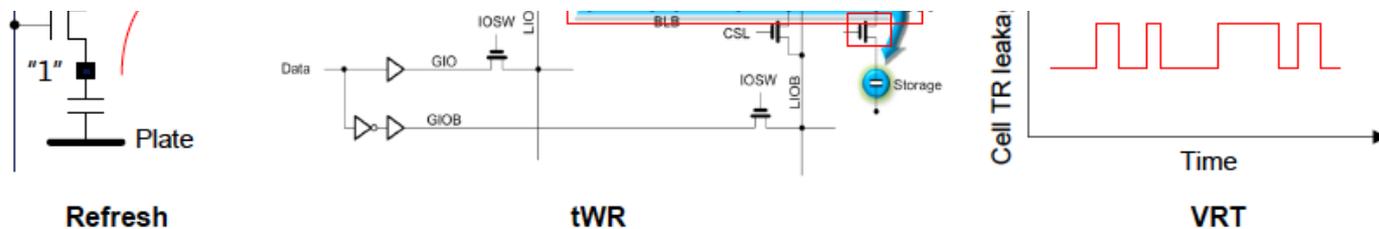
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

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*



Solution Direction: Principled Designs

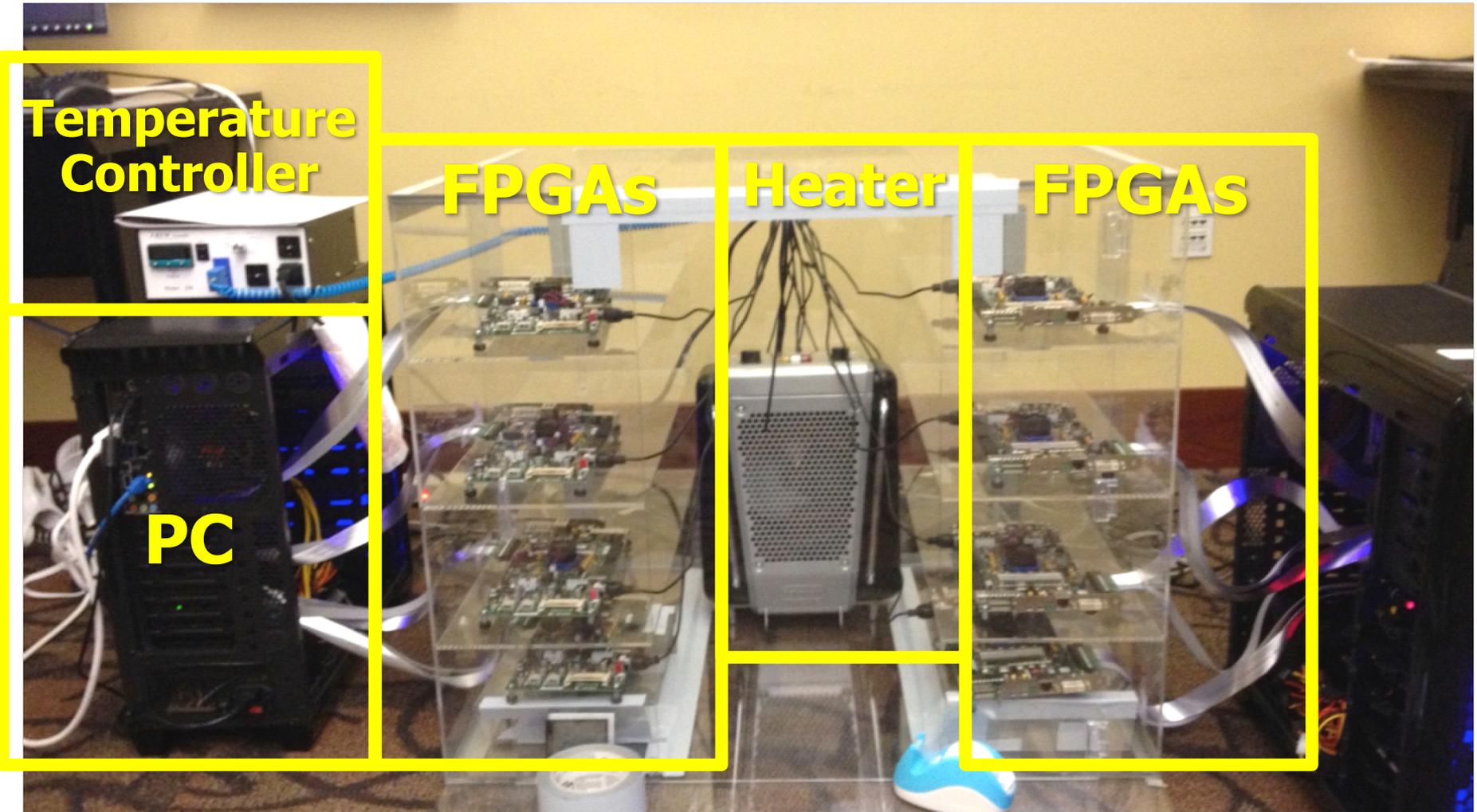
**Design fundamentally secure
computing architectures**

**Predict and prevent
such safety issues**

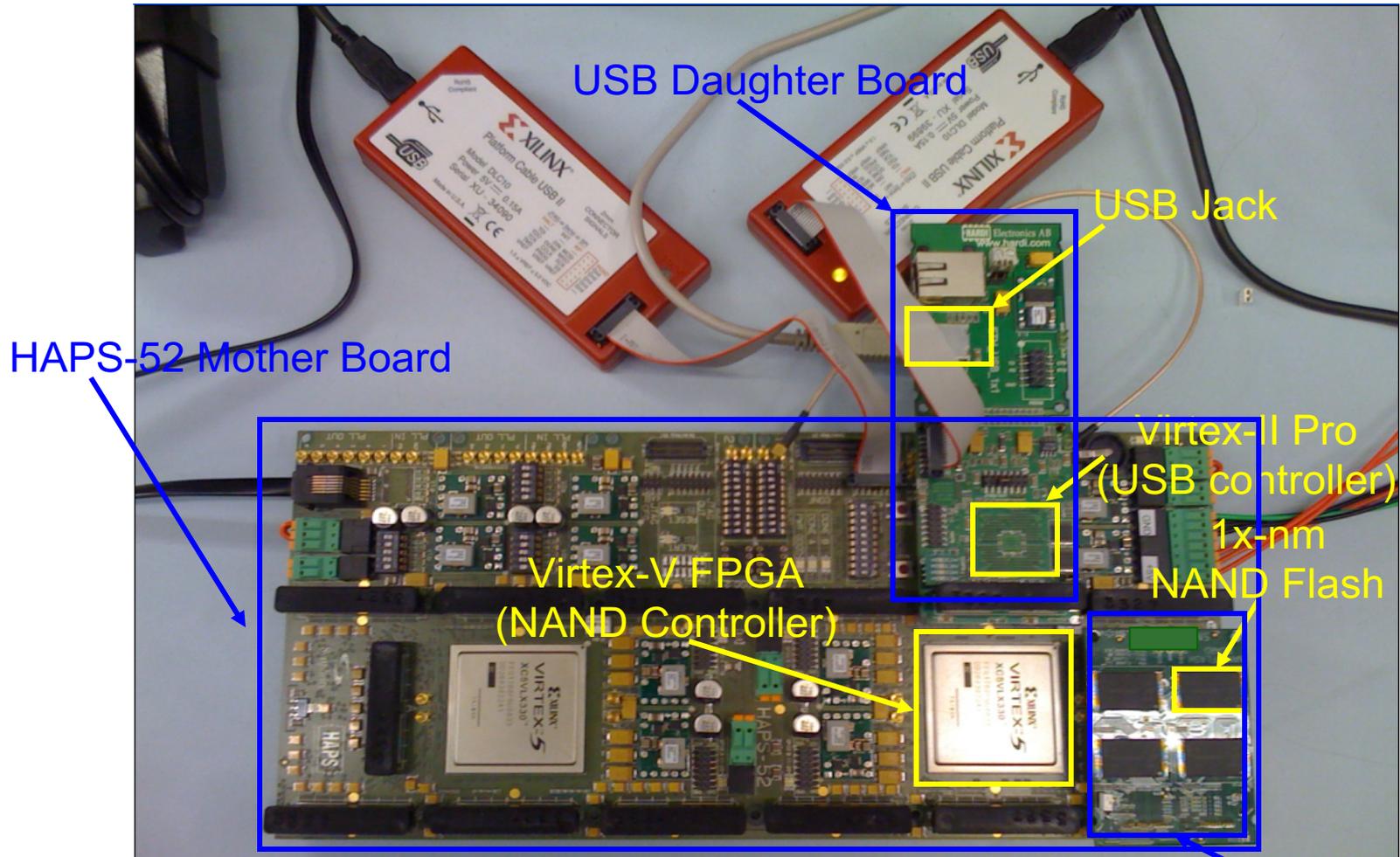
How Do We Keep Memory Secure?

- **Understand:** Methodologies for failure modeling and discovery
 - Modeling and prediction based on real (device) data
- **Architect:** Principled co-architecting of system and memory
 - Good partitioning of duties across the stack
- **Design & Test:** Principled design, automation, testing
 - High coverage and good interaction with system reliability methods

Understand and Model with Experiments (DRAM)



Understand and Model with Experiments (Flash)



[DATE 2012, ICCD 2012, DATE 2013, ITJ 2013, ICCD 2013, SIGMETRICS 2014, HPCA 2015, DSN 2015, MSST 2015, JSAC 2016, HPCA 2017, DFRWS 2017, PIEEE'17]

Another Talk: NAND Flash Reliability

- Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu, **"Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives"**

to appear in Proceedings of the IEEE, 2017.

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.

Meza+, "A Large-Scale Study of Flash Memory Errors in the Field," SIGMETRICS 2015.

Luo+, "Enabling Accurate and Practical Online Flash Channel Modeling for Modern MLC NAND Flash Memory," IEEE JSAC 2016.

Cai+, "Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques," HPCA 2017.

Fukami+, "Improving the Reliability of Chip-Off Forensic Analysis of NAND Flash Memory Devices," DFRWS EU 2017.

Cai+, "Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives," Proc. IEEE 2017.

NAND Flash Vulnerabilities

HPCA, Feb. 2017

Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques

Yu Cai[†] Saugata Ghose[†] Yixin Luo^{††} Ken Mai[†] Onur Mutlu^{§†} Erich F. Haratsch[‡]
[†]Carnegie Mellon University [‡]Seagate Technology [§]ETH Zürich

Modern NAND flash memory chips provide high density by storing two bits of data in each flash cell, called a multi-level cell (MLC). An MLC partitions the threshold voltage range of a flash cell into four voltage states. When a flash cell is programmed, a high voltage is applied to the cell. Due to parasitic capacitance coupling between flash cells that are physically close to each other, flash cell programming can lead to cell-to-cell program interference, which introduces errors into neighboring flash cells. In order to reduce the impact of cell-to-cell interference on the reliability of MLC NAND flash memory, flash manufacturers adopt a two-step programming method, which programs the MLC in two separate steps. First, the flash memory partially programs the least significant bit of the MLC to some intermediate threshold voltage. Second, it programs the most significant bit to bring the MLC up to its full voltage state.

In this paper, we demonstrate that two-step programming exposes new reliability and security vulnerabilities. We expe-

belongs to a different flash memory page (the unit of data programmed and read at the same time), which we refer to, respectively, as the least significant bit (LSB) page and the most significant bit (MSB) page [5].

*A flash cell is programmed by applying a large voltage on the control gate of the transistor, which triggers charge transfer into the floating gate, thereby increasing the threshold voltage. To precisely control the threshold voltage of the cell, the flash memory uses *incremental step pulse programming* (ISPP) [12, 21, 25, 41]. ISPP applies multiple short pulses of the programming voltage to the control gate, in order to increase the cell threshold voltage by some small voltage amount (V_{step}) after each step. Initial MLC designs programmed the threshold voltage in *one shot*, issuing all of the pulses back-to-back to program *both* bits of data at the same time. However, as flash memory scales down, the distance between neighboring flash cells decreases, which*

https://people.inf.ethz.ch/omutlu/pub/flash-memory-programming-vulnerabilities_hpca17.pdf



Proceedings of the IEEE, Sept. 2017



Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

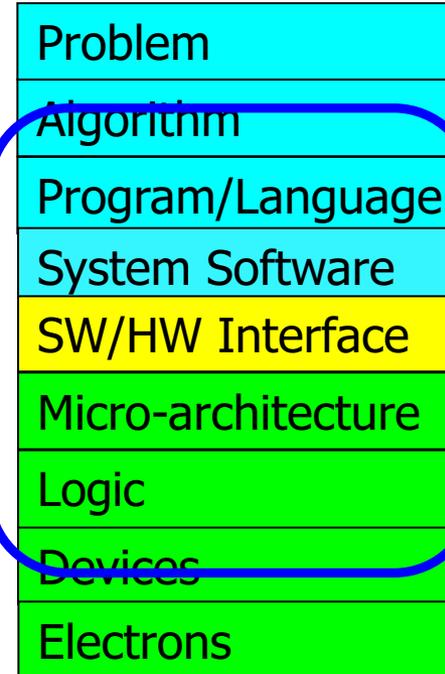
This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD's reliability and lifetime.

By YU CAI, SAUGATA GHOSE, ERICH F. HARATSCH, YIXIN LUO, AND ONUR MUTLU

<https://arxiv.org/pdf/1706.08642>

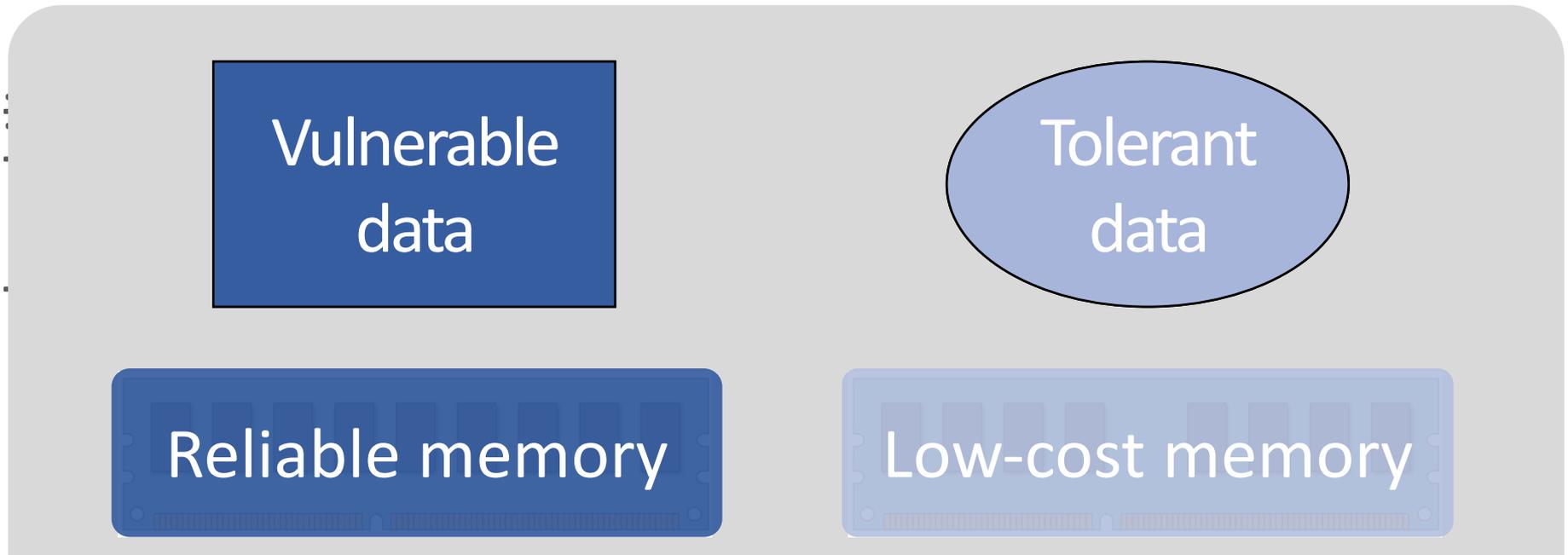
There are Two Other Solution Directions

- **New Technologies:** Replace or (more likely) augment DRAM with a different technology
 - Non-volatile memories
- **Embracing Un-reliability:**
Design memories with different reliability and store data intelligently across them
- ...



**Fundamental solutions to security
require co-design across the hierarchy**

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-Reliability Memory [DSN 2014]

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. [[Summary](#)] [[Slides \(pptx\)](#)] [[pdf](#)] [[Coverage on ZDNet](#)]

Characterizing Application Memory Error Vulnerability to Optimize Datacenter Cost via Heterogeneous-Reliability Memory

Yixin Luo Sriram Govindan* Bikash Sharma* Mark Santaniello* Justin Meza
Aman Kansal* Jie Liu* Badriddine Khessib* Kushagra Vaid* Onur Mutlu

Carnegie Mellon University, yixinluo@cs.cmu.edu, {meza, onur}@cmu.edu

*Microsoft Corporation, {srgovin, bsharma, marksan, kansal, jie.liu, bk Hessib, kvaid}@microsoft.com

Summary: Memory Reliability and Security

- **Memory reliability is reducing**
- Reliability issues open up security vulnerabilities
 - Very hard to defend against
- Rowhammer is an example
 - Its implications on system security research are tremendous & exciting

- **Good news: We have a lot more to do.**
- **Understand: Solid methodologies for failure modeling and discovery**
 - Modeling based on real device data – small scale and large scale
- **Architect: Principled co-architecting of system and memory**
 - Good partitioning of duties across the stack
- **Design & Test: Principled electronic design, automation, testing**
 - High coverage and good interaction with system reliability methods

Fundamentally Secure, Reliable, Safe Computing Architectures

One Important Takeaway

**Main Memory Needs
Intelligent Controllers**

Three Key Issues in Future Platforms

- Fundamentally **Secure/Reliable/Safe** Architectures
- Fundamentally **Energy-Efficient** Architectures
 - **Memory-centric** (Data-centric) Architectures
- Fundamentally **Low Latency** Architectures

Do We Want This?



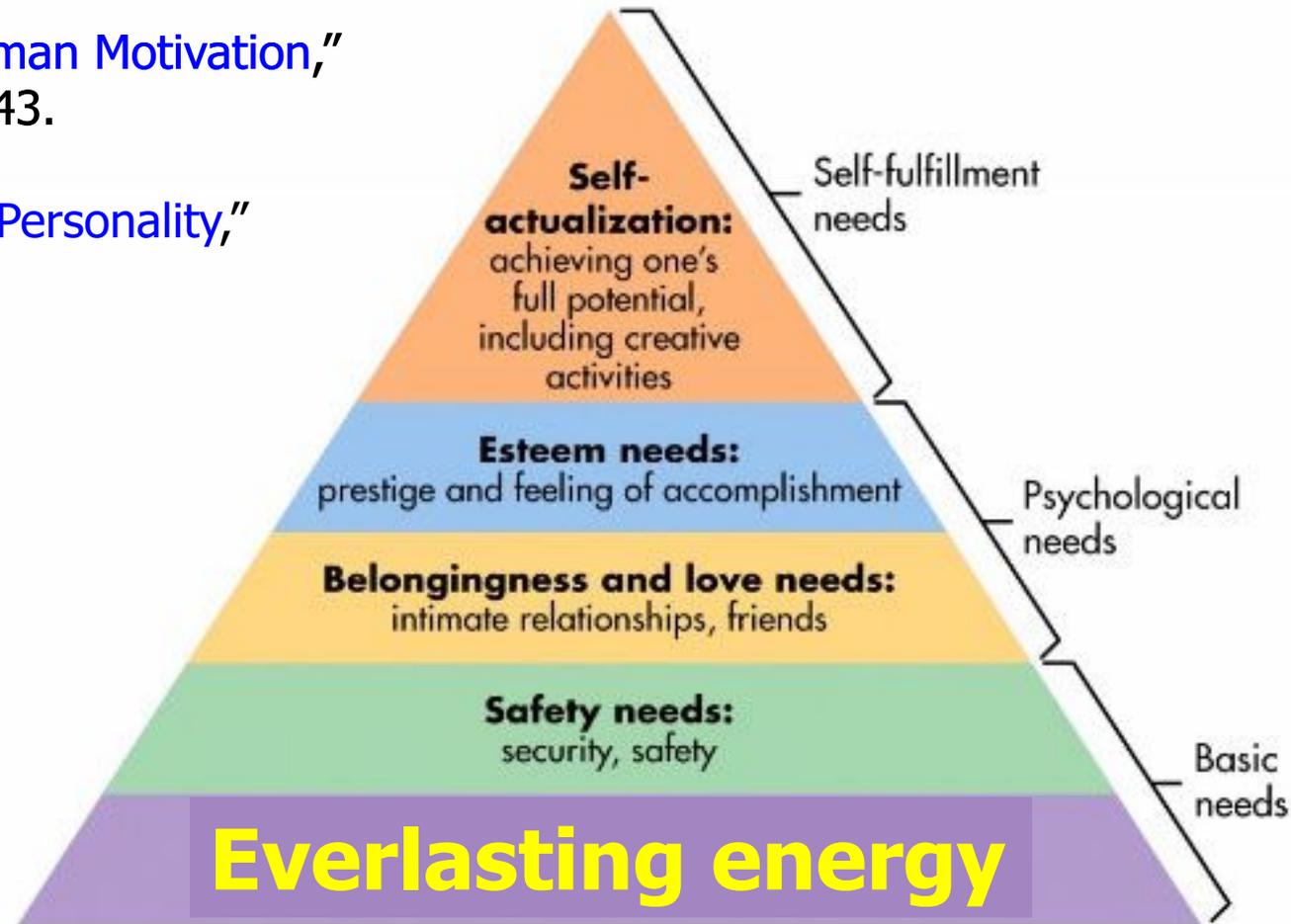
Or, This?



Maslow's (Human) Hierarchy of Needs, Revisited

Maslow, "A Theory of Human Motivation,"
Psychological Review, 1943.

Maslow, "Motivation and Personality,"
Book, 1954-1970.



Sustainable and Energy Efficient

Three Key Systems Trends

1. Data access is a major bottleneck

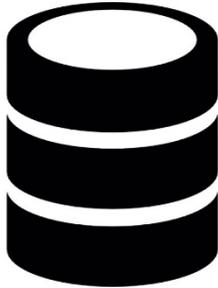
- Applications are increasingly data hungry

2. Energy consumption is a key limiter

3. Data movement energy dominates compute

- Especially true for off-chip to on-chip movement

The Need for More Memory Performance



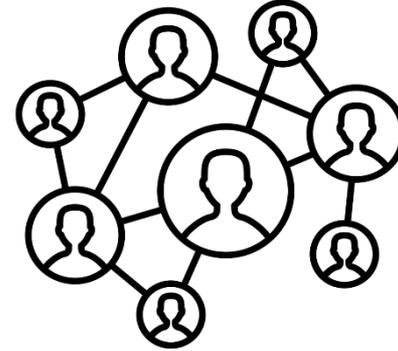
In-memory Databases

[Mao+, EuroSys'12;
Clapp+ (Intel), IISWC'15]



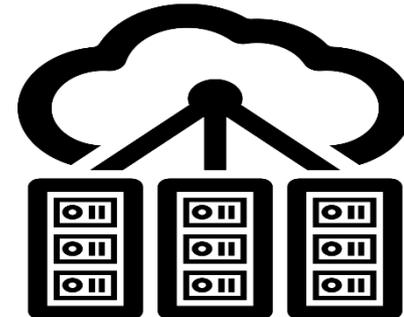
In-Memory Data Analytics

[Clapp+ (Intel), IISWC'15;
Awan+, BDCloud'15]



Graph/Tree Processing

[Xu+, IISWC'12; Umuroglu+, FPL'15]

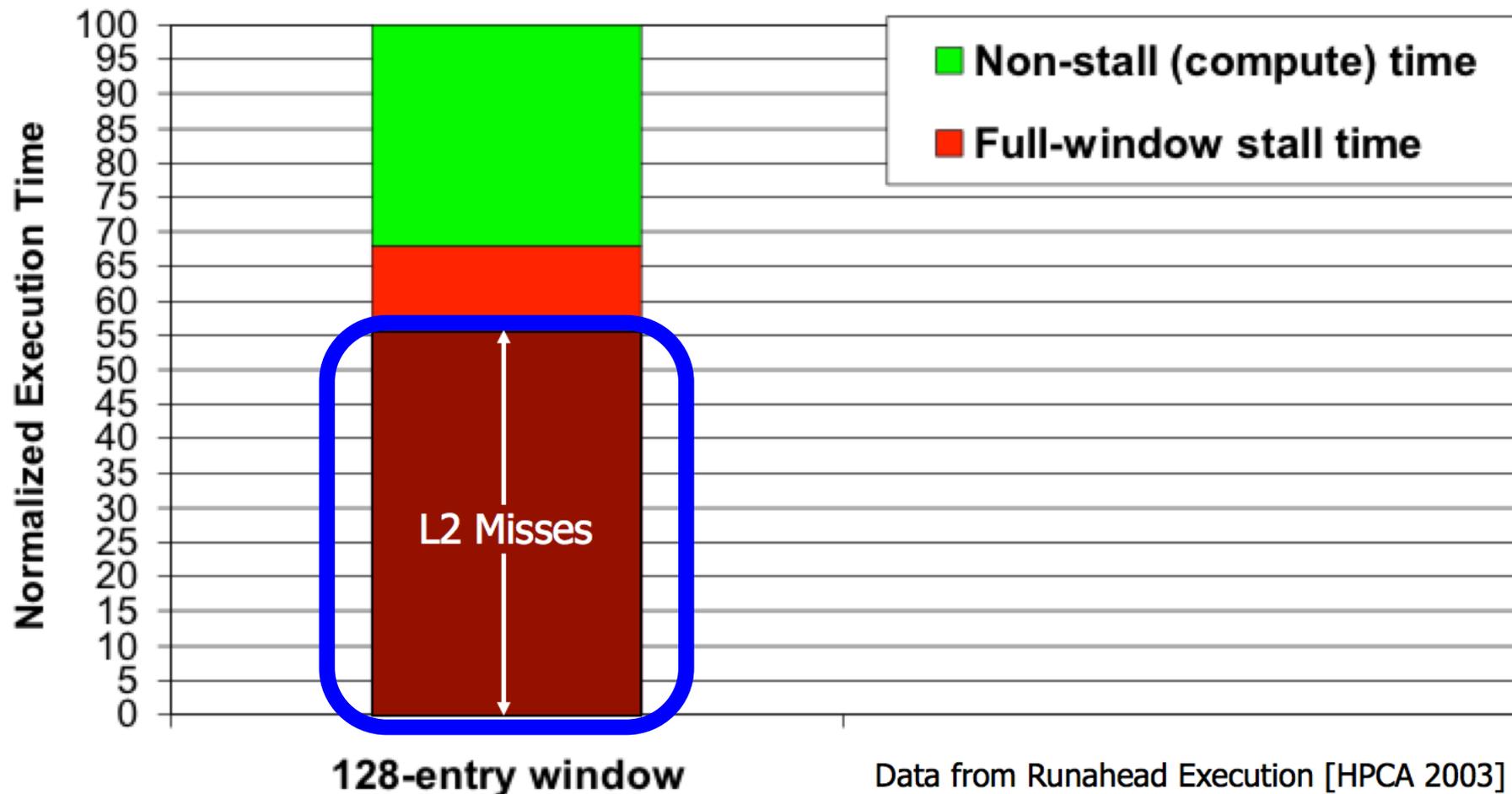


Datacenter Workloads

[Kanev+ (Google), ISCA'15]

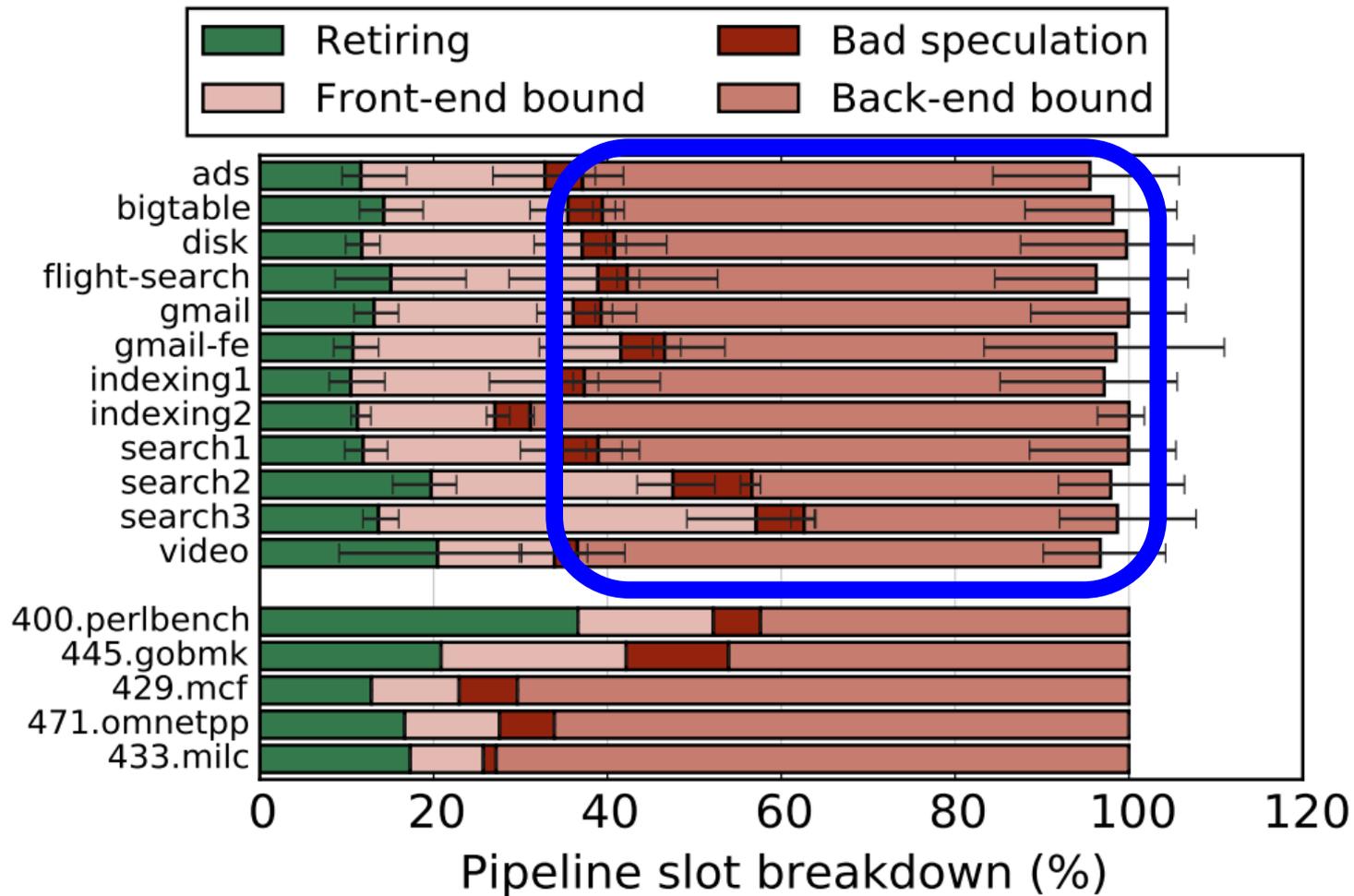
The Performance Perspective (1996-2005)

- **“It’s the Memory, Stupid!”** (Richard Sites, MPR, 1996)



The Performance Perspective (Today)

- All of Google's Data Center Workloads (2015):



The Performance Perspective (Today)

- All of Google's Data Center Workloads (2015):

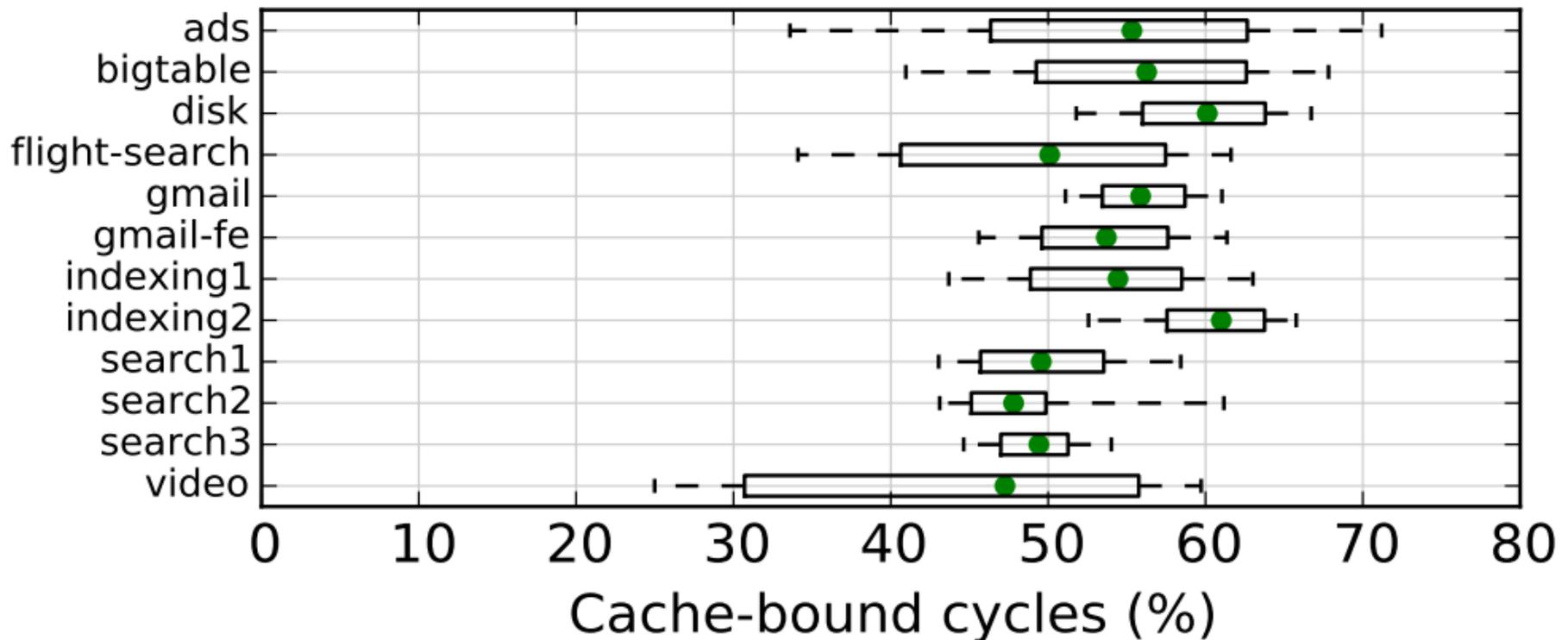


Figure 11: Half of cycles are spent stalled on caches.

The Performance Perspective

- Onur Mutlu, Jared Stark, Chris Wilkerson, and Yale N. Patt, **"Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors"**
Proceedings of the 9th International Symposium on High-Performance Computer Architecture (HPCA), pages 129-140, Anaheim, CA, February 2003. [Slides \(pdf\)](#)

Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors

Onur Mutlu § Jared Stark † Chris Wilkerson ‡ Yale N. Patt §

§ECE Department
The University of Texas at Austin
{onur,patt}@ece.utexas.edu

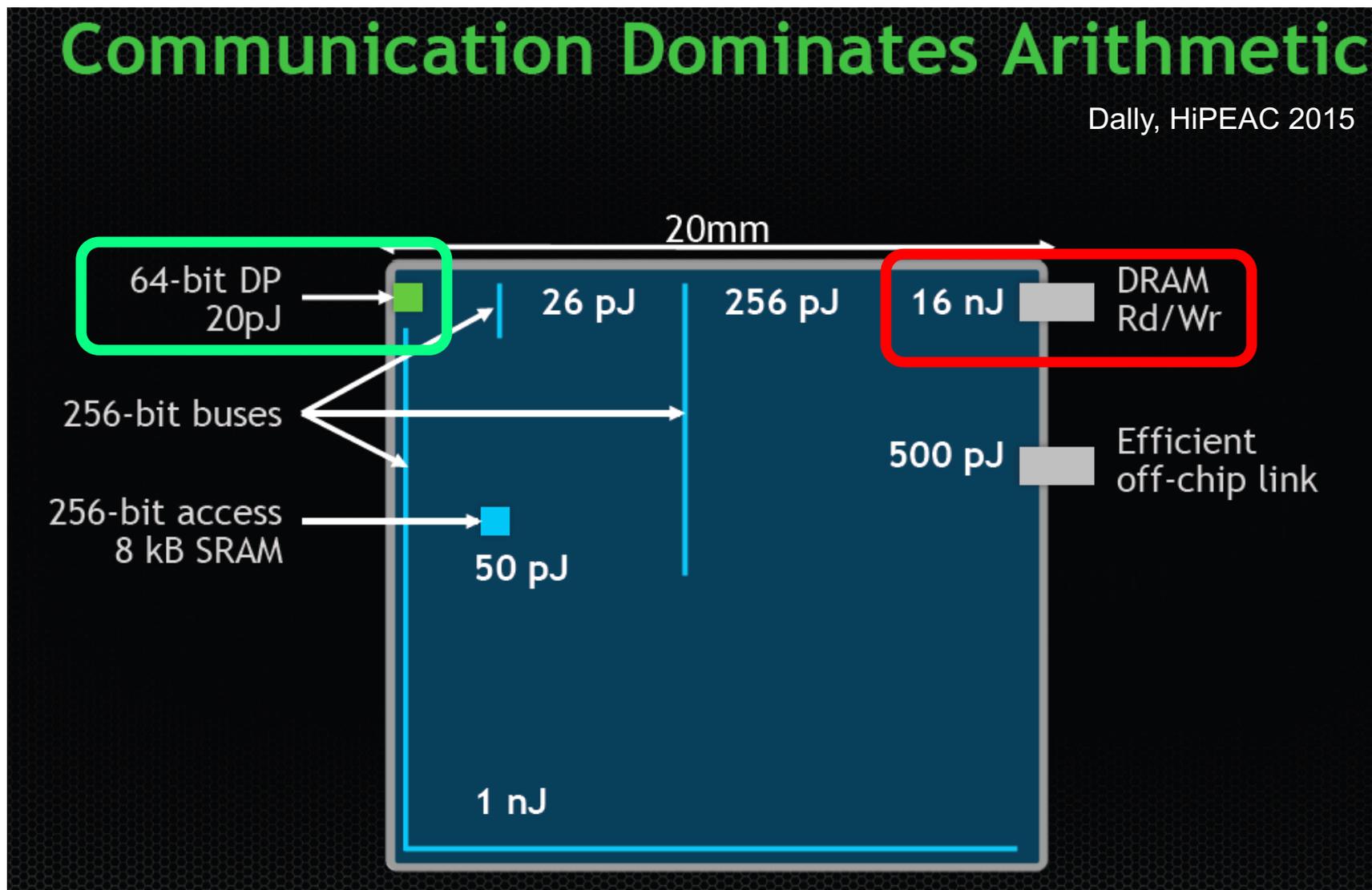
†Microprocessor Research
Intel Labs
jared.w.stark@intel.com

‡Desktop Platforms Group
Intel Corporation
chris.wilkerson@intel.com

The Energy Perspective

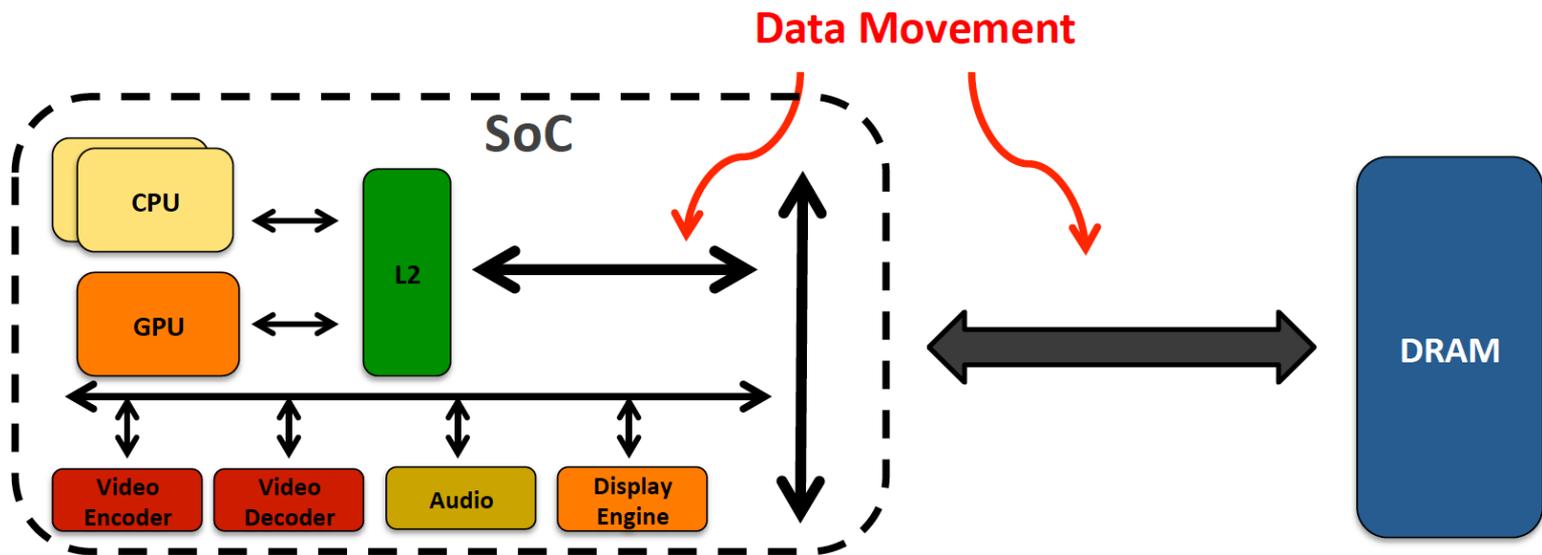
Communication Dominates Arithmetic

Dally, HiPEAC 2015



Data Movement vs. Computation Energy

- **Data movement** is a major system energy bottleneck
 - Comprises 41% of mobile system energy during web browsing [2]
 - Costs ~ 115 times as much energy as an ADD operation [1, 2]



[1]: Reducing data Movement Energy via Online Data Clustering and Encoding (MICRO'16)

[2]: Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms (IISWC'14)

High Performance and Energy Efficient

The Problem

Data access is the major performance and energy bottleneck

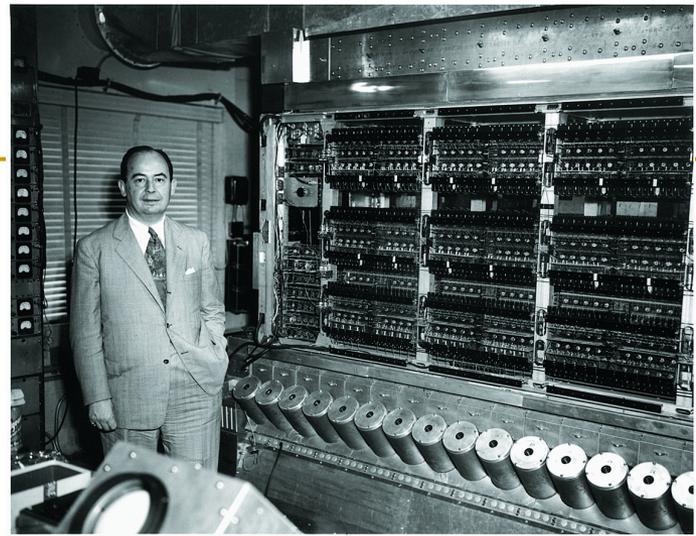
Our current
design principles
cause great energy waste
(and great performance loss)

The Problem

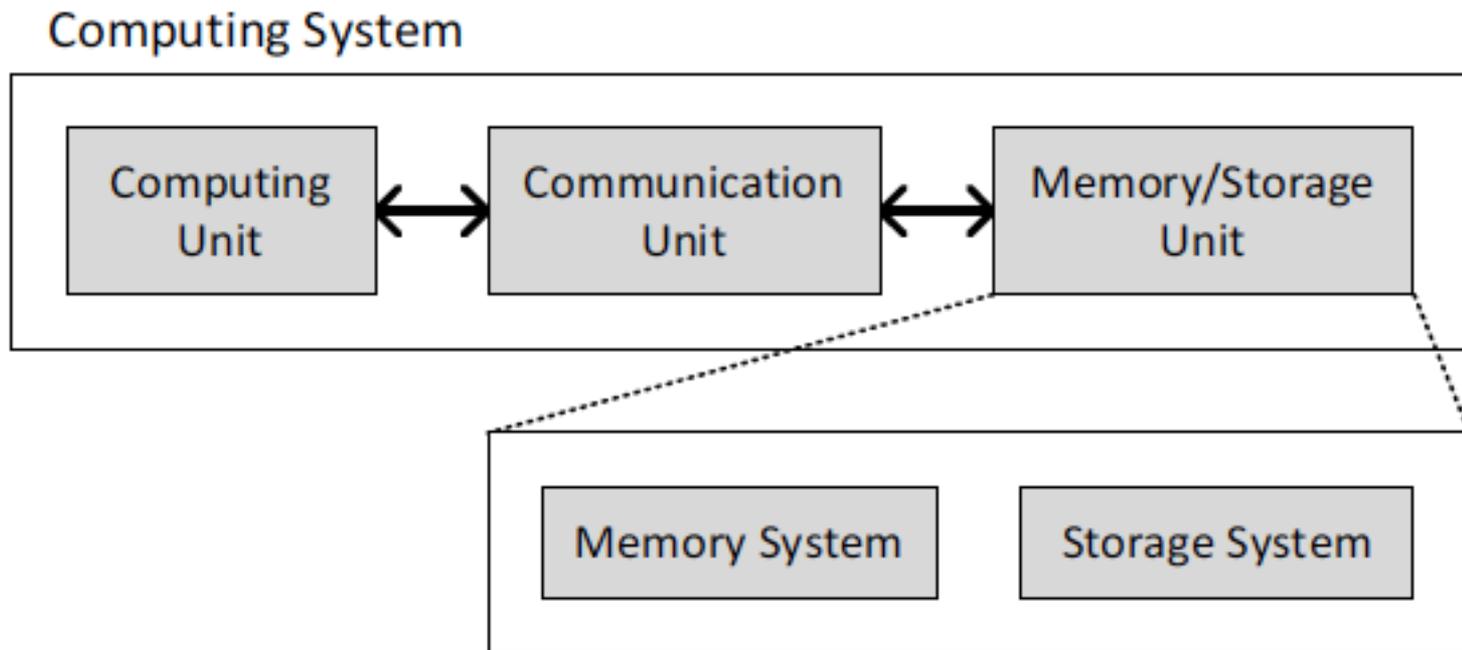
Processing of data
is performed
far away from the data

A Computing System

- Three key components
- Computation
- Communication
- Storage/memory

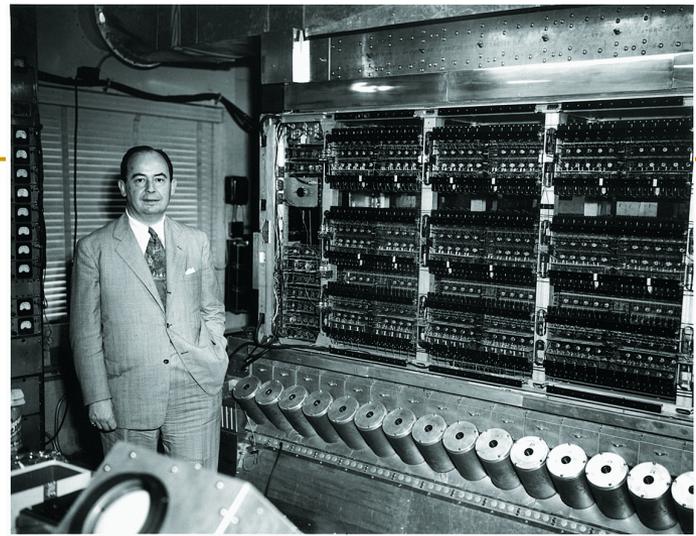


Burks, Goldstein, von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument," 1946.



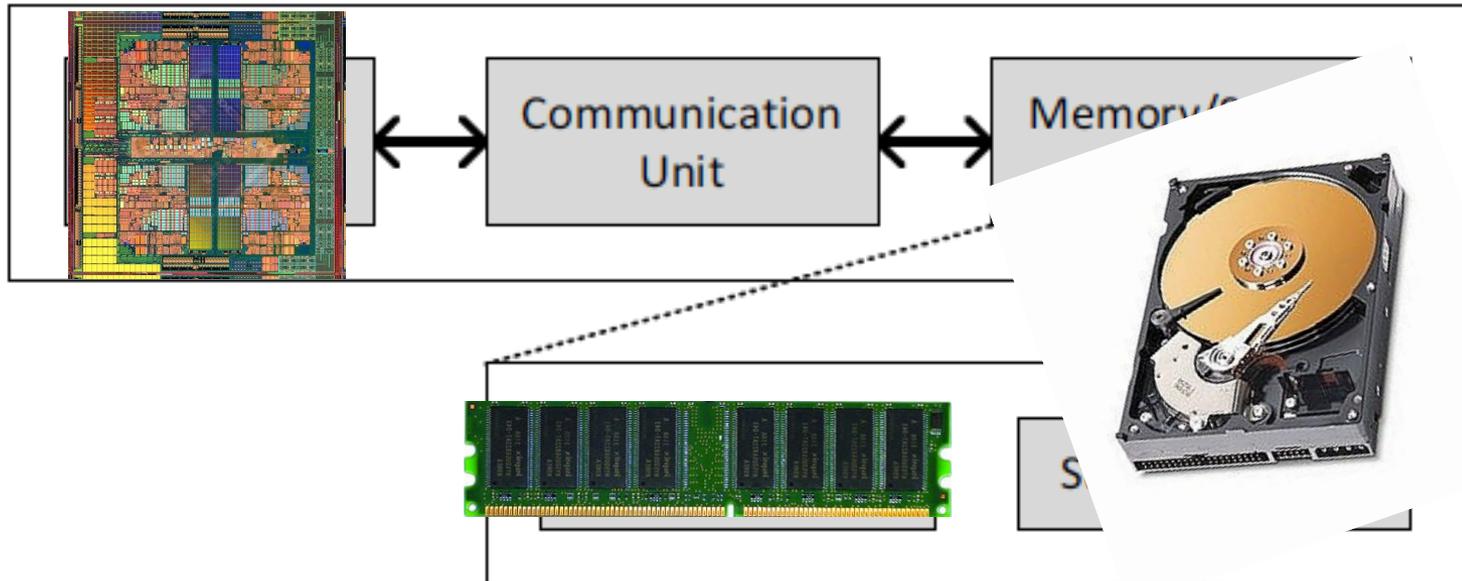
A Computing System

- Three key components
- Computation
- Communication
- Storage/memory



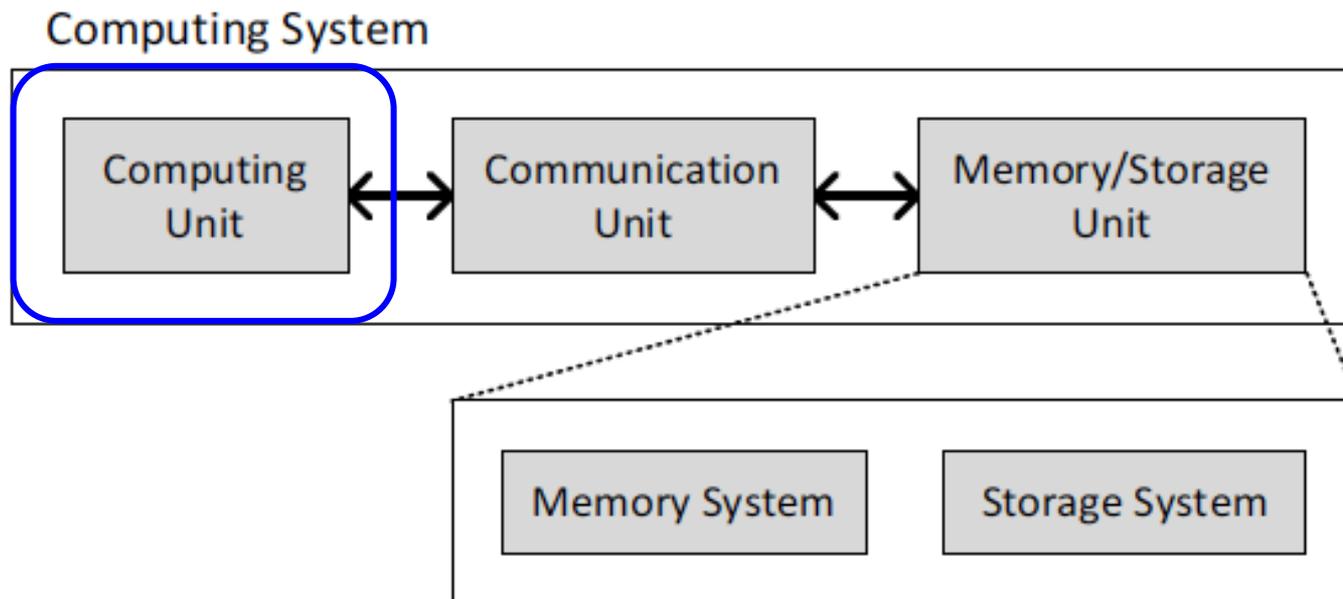
Burks, Goldstein, von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument," 1946.

Computing System



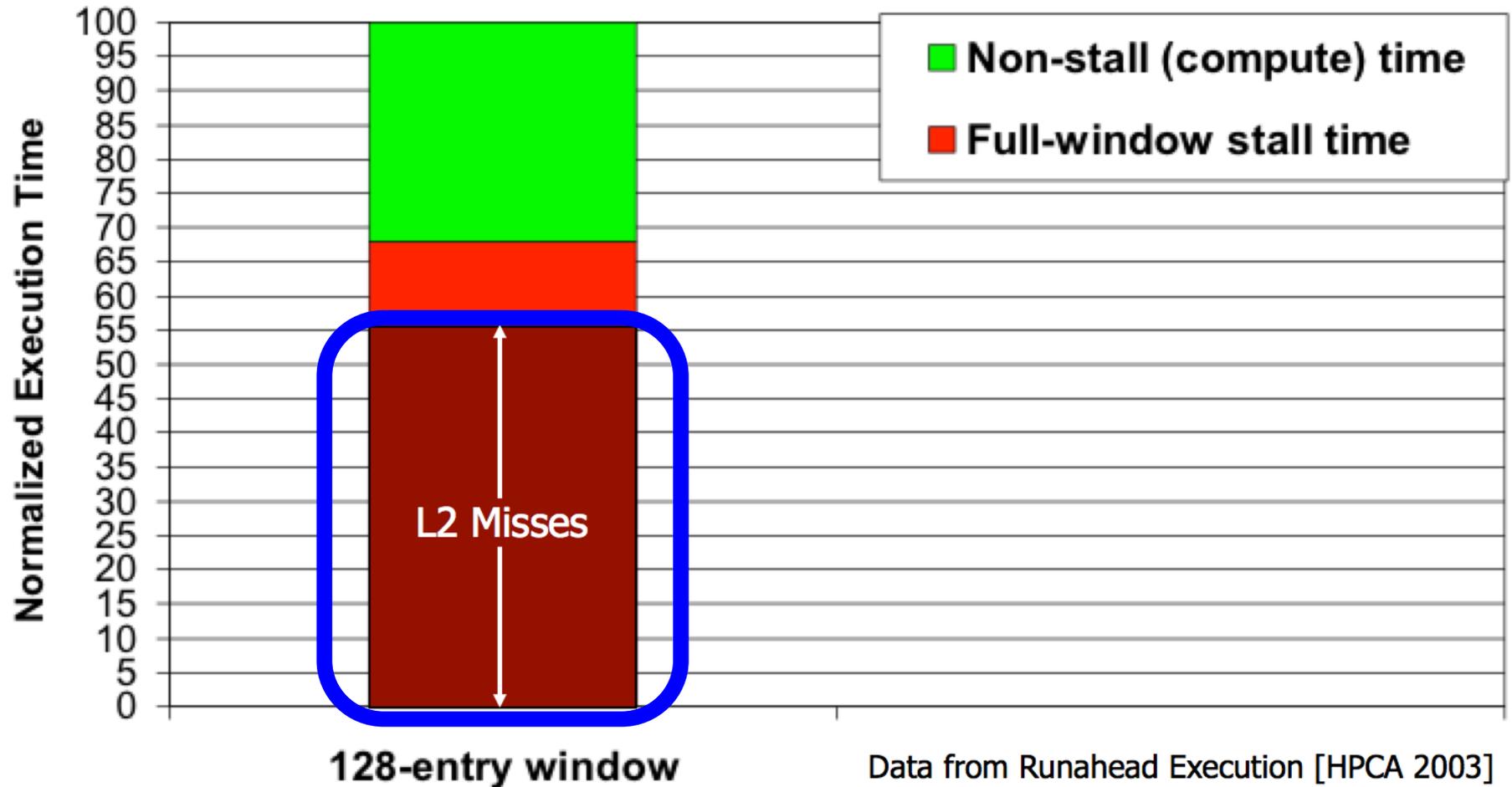
Today's Computing Systems

- Are overwhelmingly processor centric
- **All data processed in the processor** → at great system cost
- Processor is heavily optimized and is considered the master
- **Data storage units are dumb** and are largely unoptimized (except for some that are on the processor die)



Yet ...

- **“It’s the Memory, Stupid!”** (Richard Sites, MPR, 1996)

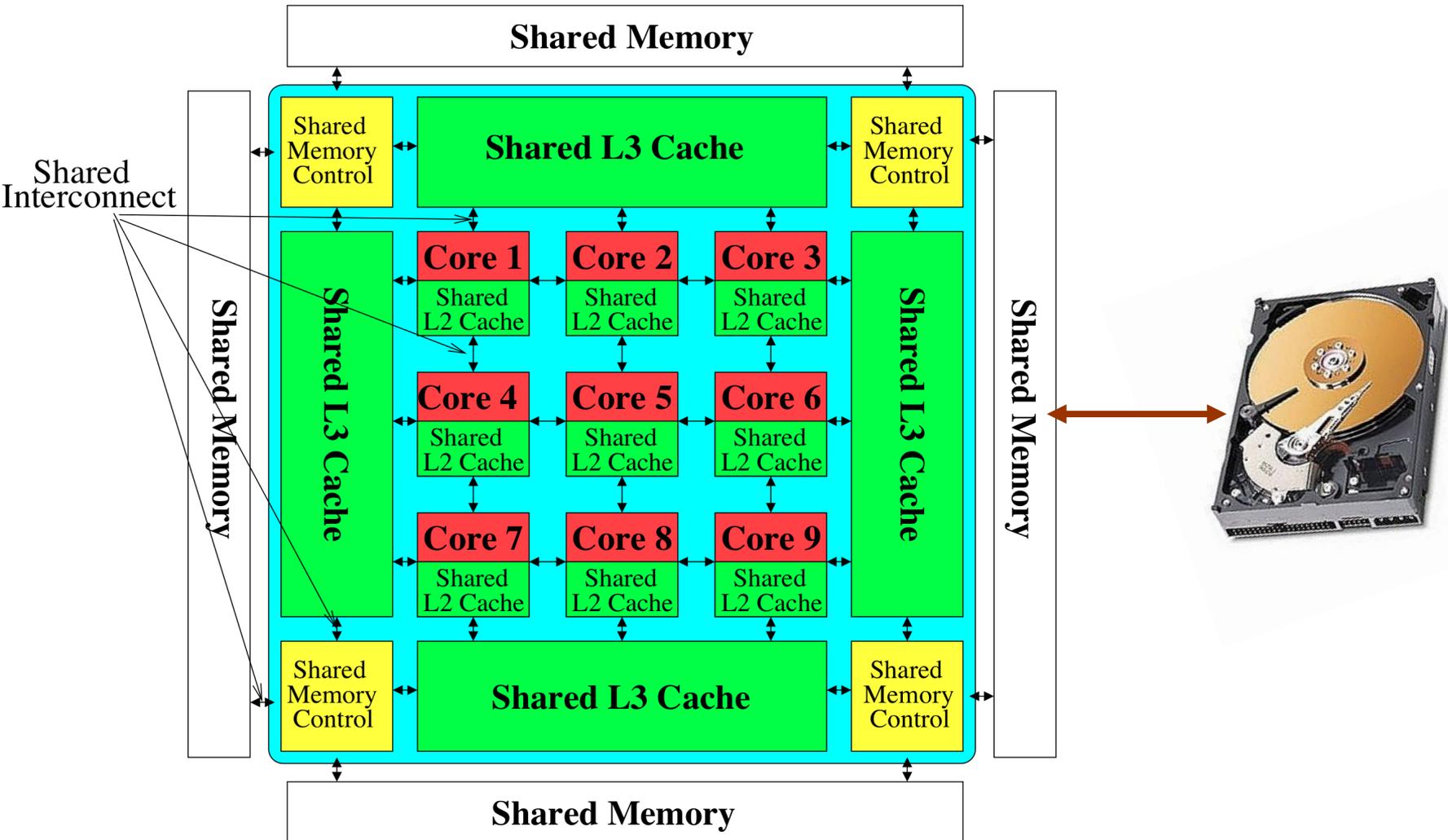


Perils of Processor-Centric Design

- **Grossly-imbalanced systems**
 - ❑ Processing done only in **one place**
 - ❑ Everything else just stores and moves data: **data moves a lot**
 - Energy inefficient
 - Low performance
 - Complex

- **Overly complex and bloated processor (and accelerators)**
 - ❑ To tolerate data access from memory
 - ❑ Complex hierarchies and mechanisms
 - Energy inefficient
 - Low performance
 - Complex

Perils of Processor-Centric Design

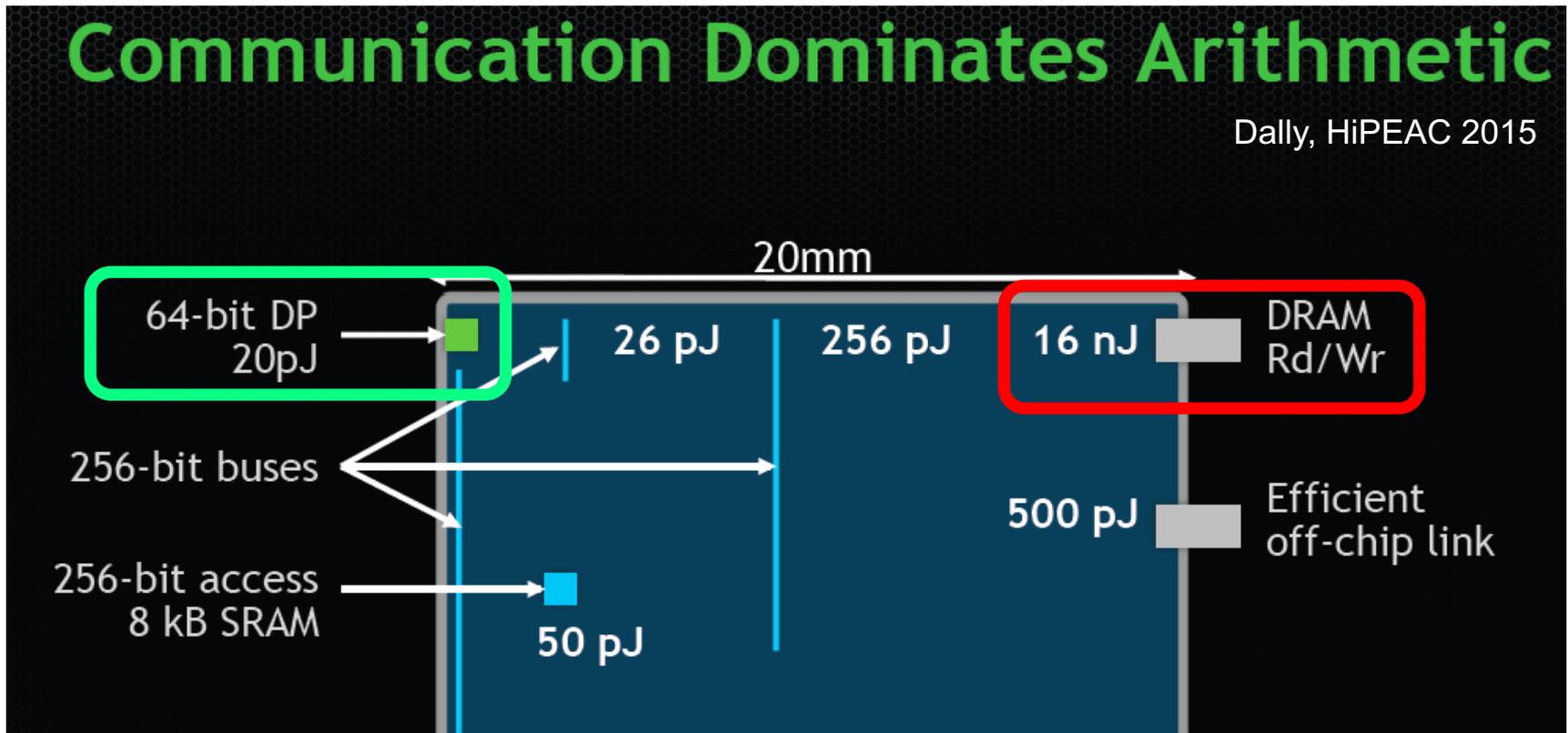


Most of the system is dedicated to storing and moving data

We Do Not Want to Move Data!

Communication Dominates Arithmetic

Dally, HiPEAC 2015

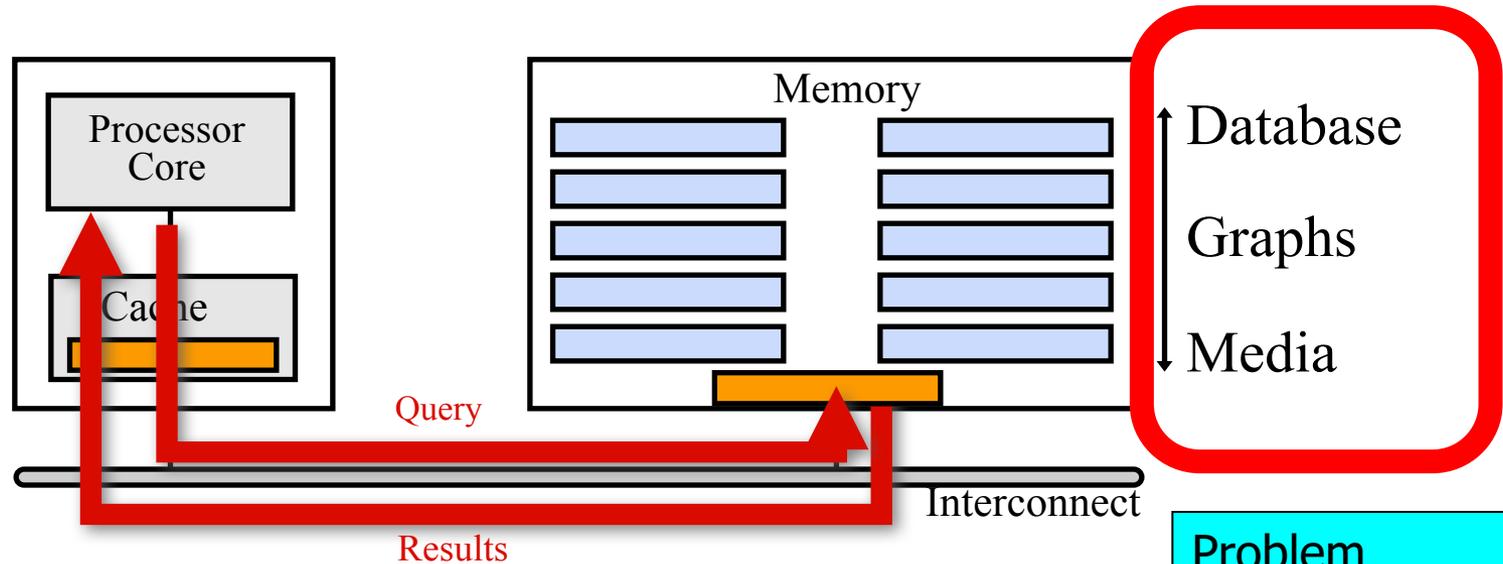


A memory access consumes $\sim 1000X$ the energy of a complex addition

We Need A Paradigm Shift To ...

- Enable computation with **minimal data movement**
- **Compute where it makes sense** (**where data resides**)
- Make computing architectures more **data-centric**

Goal: Processing Inside Memory



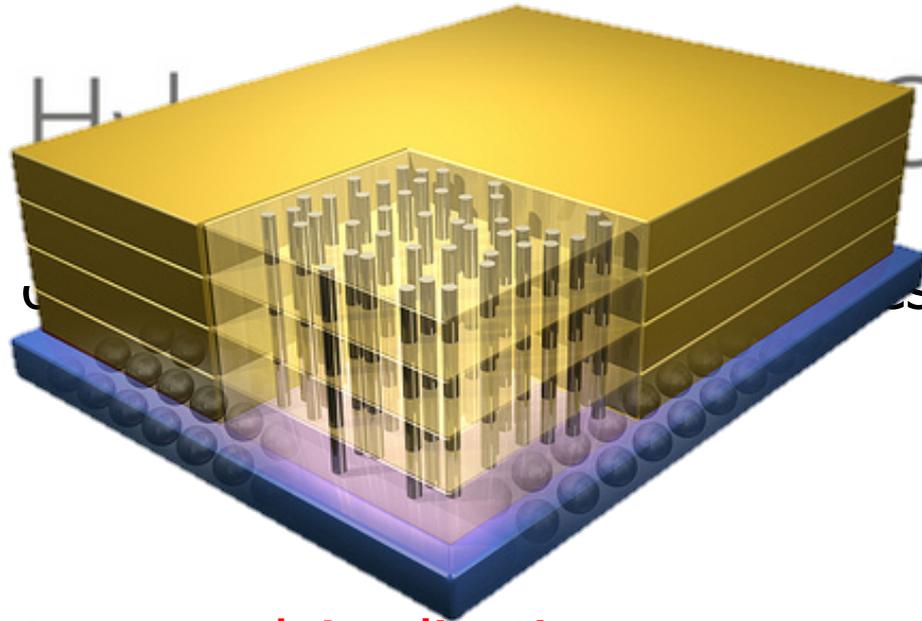
- Many questions ... How do we design the:
 - ❑ compute-capable memory & controllers?
 - ❑ processor chip?
 - ❑ software and hardware interfaces?
 - ❑ system software and languages?
 - ❑ algorithms?

Problem
Algorithm
Program/Language
System Software
SW/HW Interface
Micro-architecture
Logic
Devices
Electrons

Why In-Memory Computation Today?



→ Industry C



- Pull from Systems and Applications
 - ❑ Data access is a major system and application bottleneck
 - ❑ Systems are energy limited
 - ❑ Data movement much more energy-hungry than computation

Processing in Memory: Two Approaches

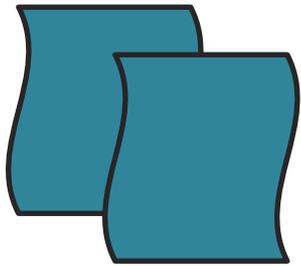
1. Minimally changing memory chips
2. Exploiting 3D-stacked memory

Approach 1: Minimally Changing DRAM

- DRAM has great capability to perform **bulk data movement and computation** internally with small changes
 - Can **exploit internal connectivity** to move data
 - Can **exploit analog computation capability**
 - ...
- Examples: RowClone, In-DRAM AND/OR, Gather/Scatter DRAM
 - 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)
 - Gather-Scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses (Seshadri et al., MICRO 2015)
 - "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology" (Seshadri et al., MICRO 2017)

Starting Simple: Data Copy and Initialization

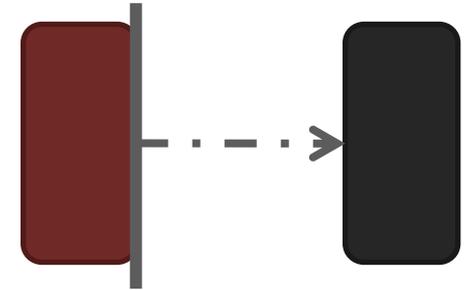
memcpy & memmove: 5% cycles in Google's datacenter [Kanev+ ISCA'15]



Forking



**Zero initialization
(e.g., security)**



Checkpointing



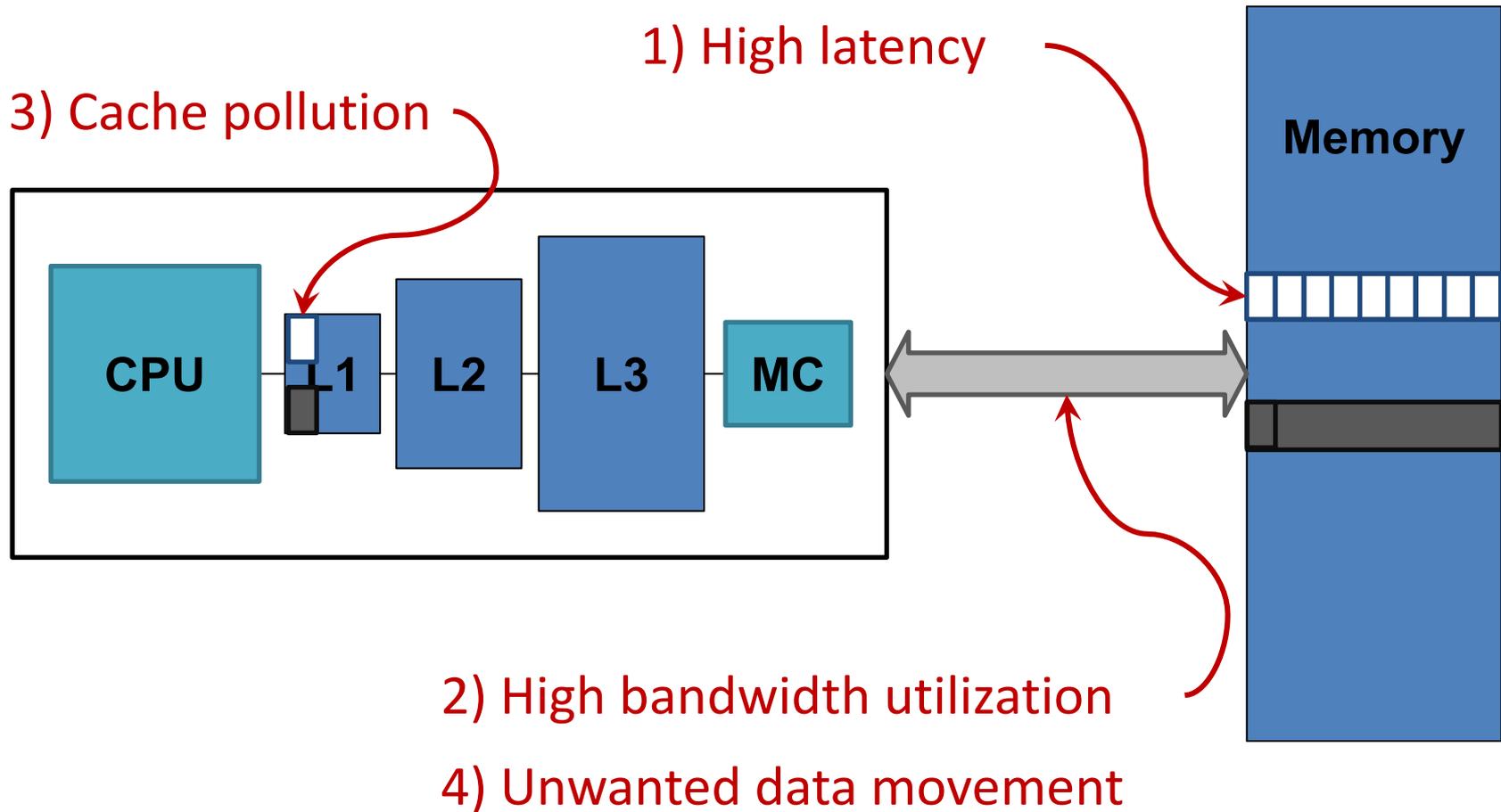
**VM Cloning
Deduplication**



Page Migration

•••
Many more

Today's Systems: Bulk Data Copy

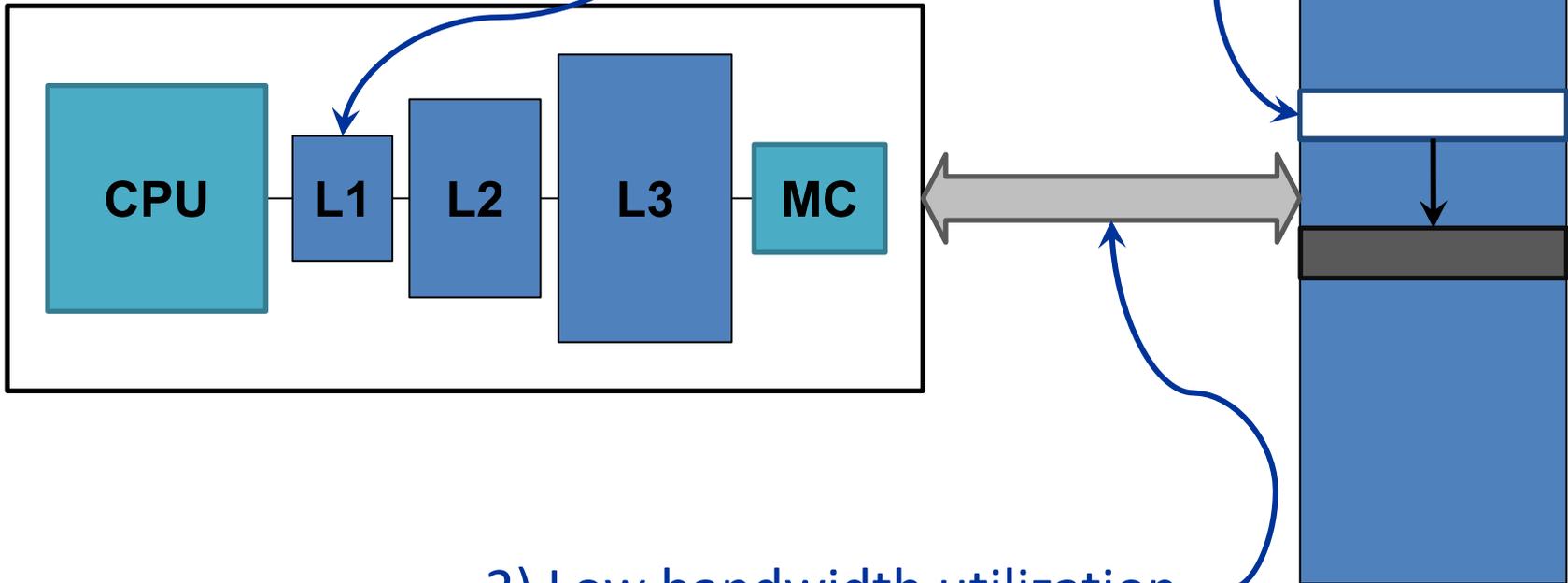


1046ns, 3.6uJ (for 4KB page copy via DMA)

Future Systems: In-Memory Copy

3) No cache pollution

1) Low latency



2) Low bandwidth utilization

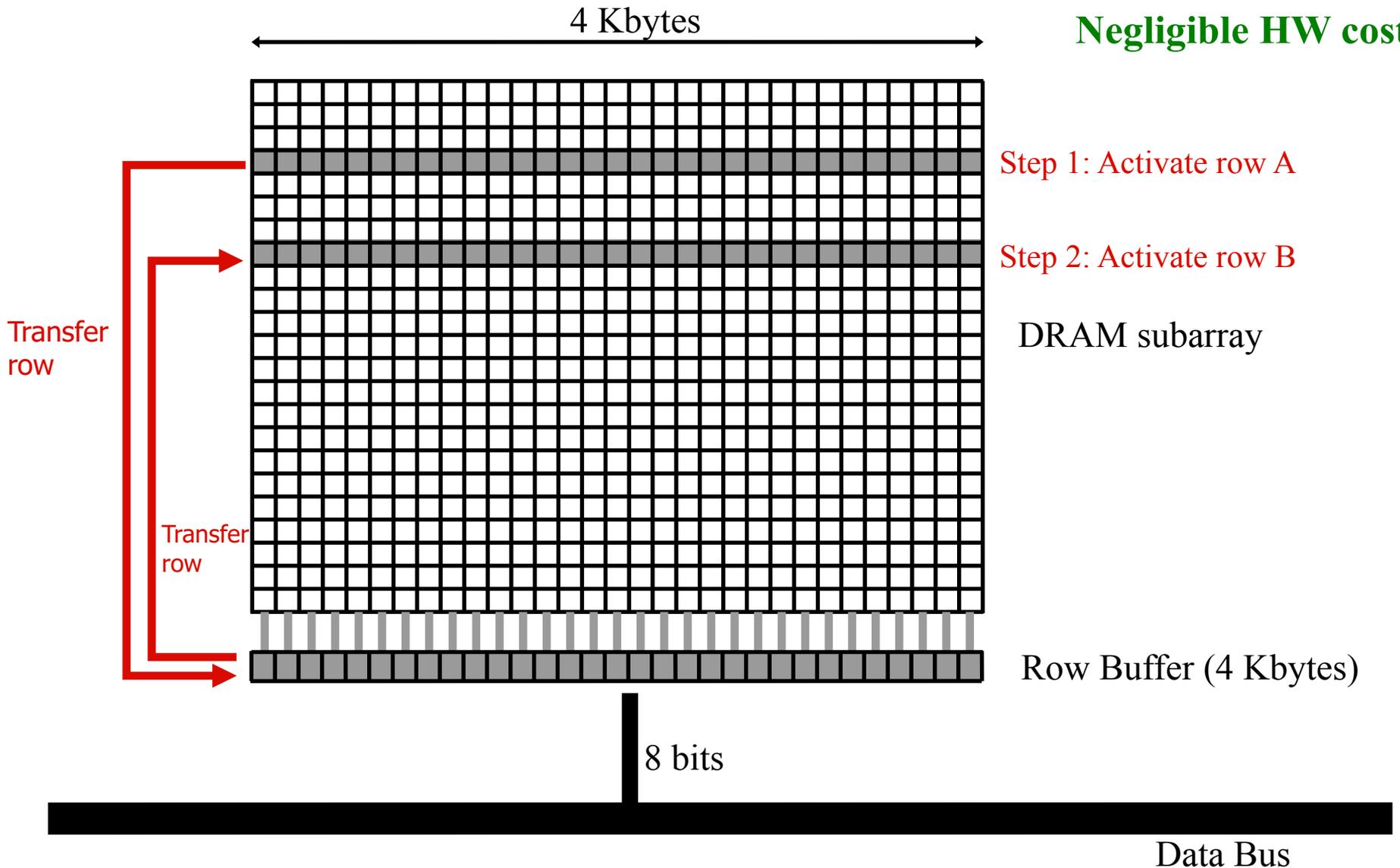
4) No unwanted data movement

1046ns, 3.6uJ

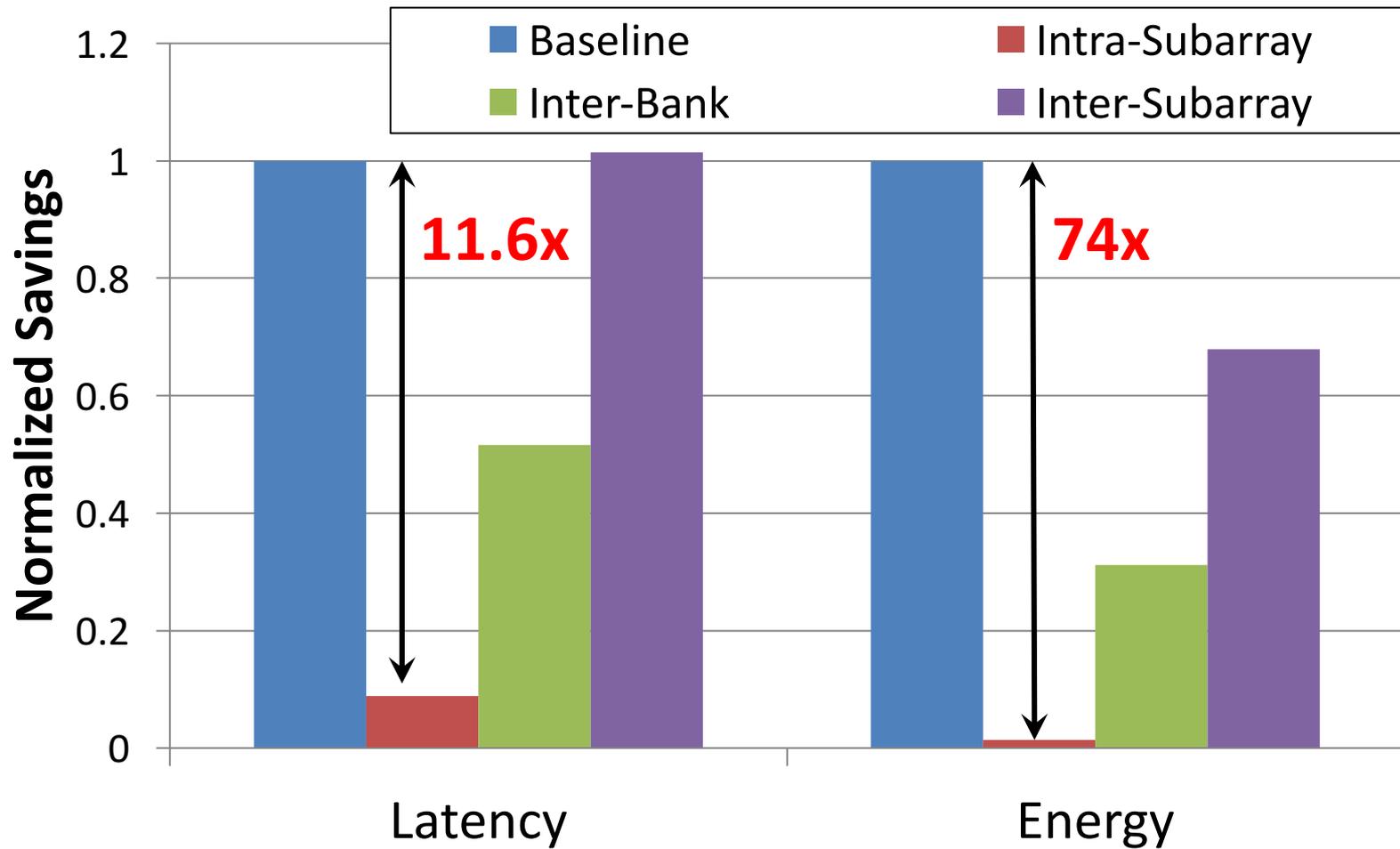
→ 90ns, 0.04uJ

RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates
Negligible HW cost



RowClone: Latency and Energy Savings



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

More on RowClone

- Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry,
"RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization"

Proceedings of the 46th International Symposium on Microarchitecture (MICRO), Davis, CA, December 2013. [[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Session Slides \(pptx\)](#)] [[pdf](#)] [[Poster \(pptx\)](#)] [[pdf](#)]

RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

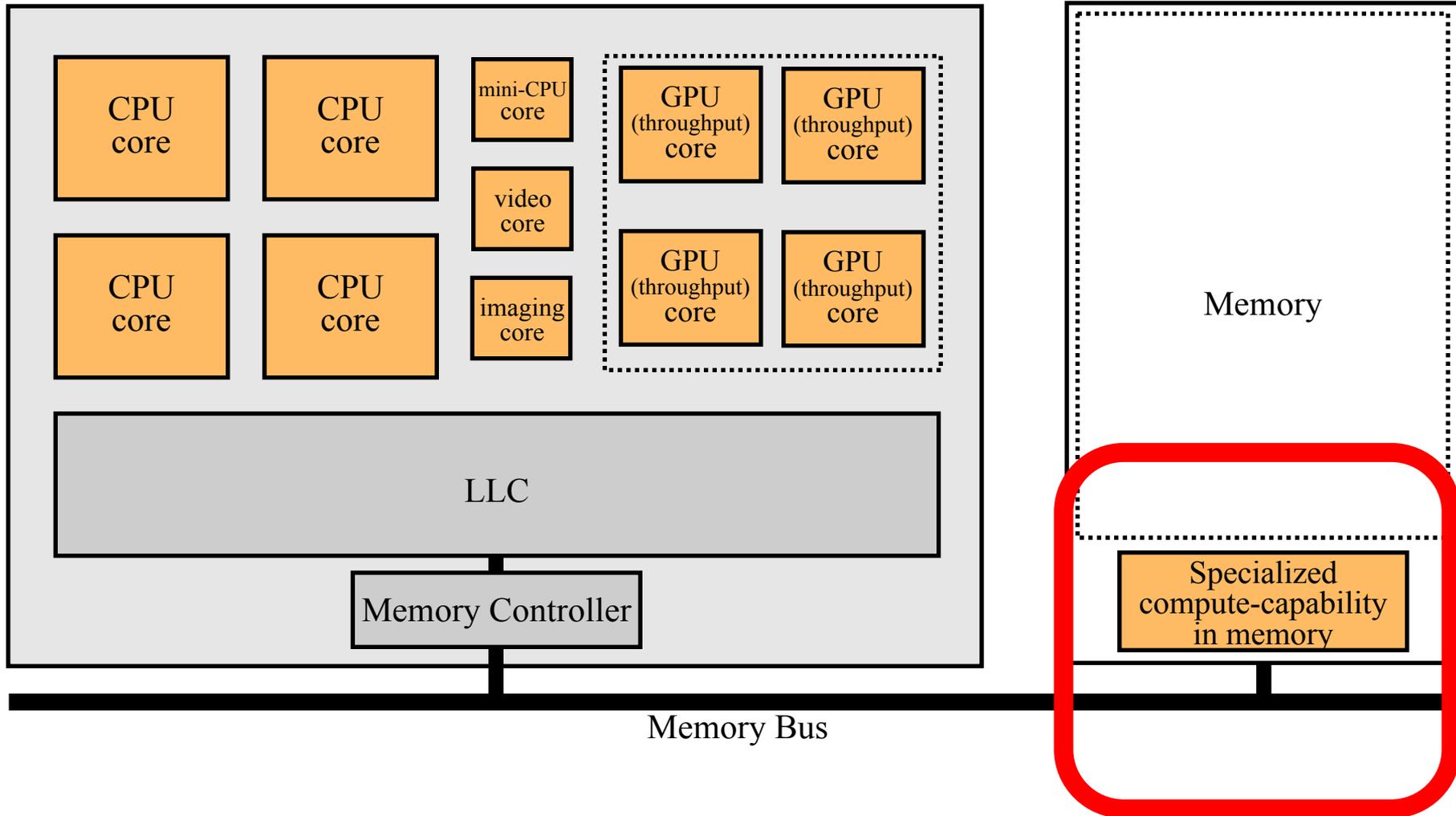
Vivek Seshadri Yoongu Kim Chris Fallin* Donghyuk Lee
vseshadr@cs.cmu.edu yoongukim@cmu.edu cfallin@c1f.net donghyuk1@cmu.edu

Rachata Ausavarungnirun Gennady Pekhimenko Yixin Luo
rachata@cmu.edu gpekhime@cs.cmu.edu yixinluo@andrew.cmu.edu

Onur Mutlu Phillip B. Gibbons† Michael A. Kozuch† Todd C. Mowry
onur@cmu.edu phillip.b.gibbons@intel.com michael.a.kozuch@intel.com tcm@cs.cmu.edu

Carnegie Mellon University †Intel Pittsburgh

Memory as an Accelerator



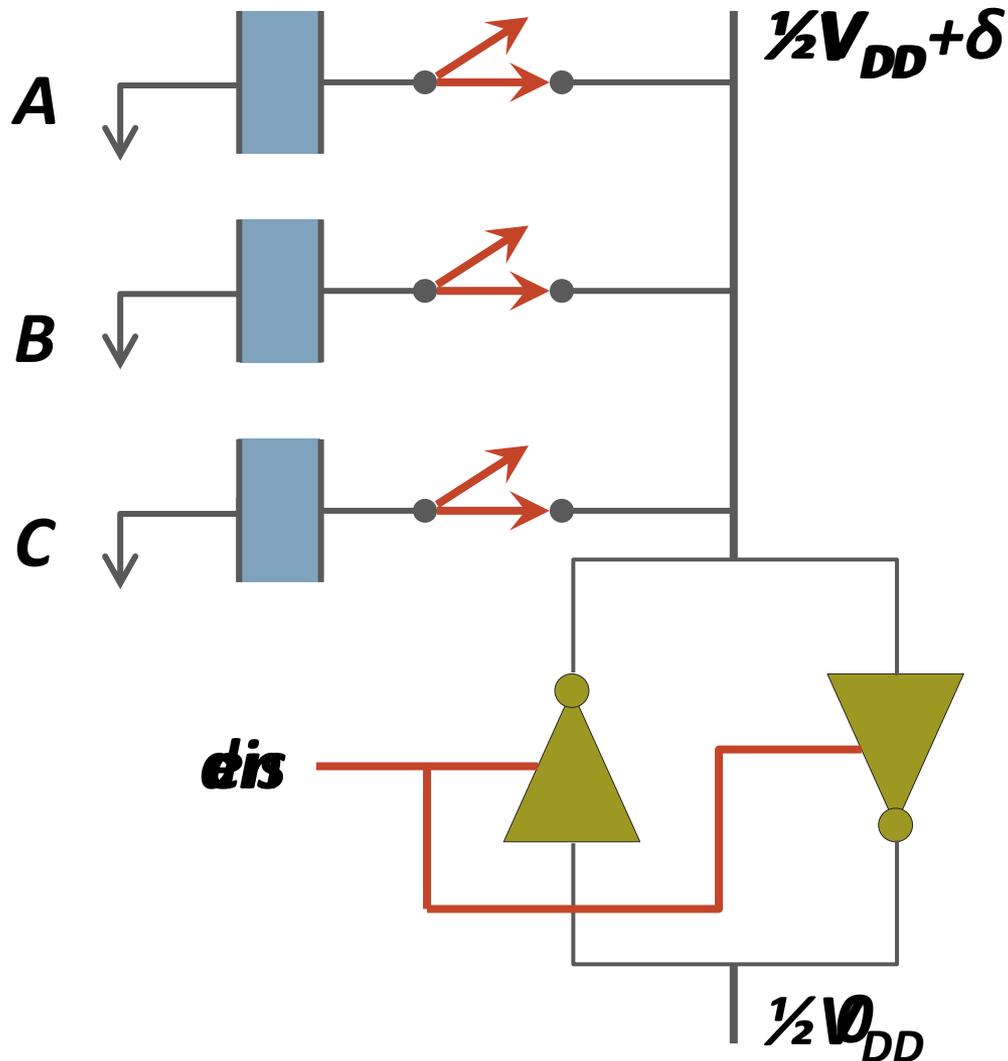
Memory similar to a "conventional" accelerator

In-Memory Bulk Bitwise Operations

- We can support **in-DRAM COPY, ZERO, AND, OR, NOT, MAJ**
- At low cost
- Using analog computation capability of DRAM
 - Idea: activating multiple rows performs computation
- **30-60X performance and energy improvement**
 - Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," MICRO 2017.

- **New memory technologies** enable even more opportunities
 - Memristors, resistive RAM, phase change mem, STT-MRAM, ...
 - Can operate on data **with minimal movement**

In-DRAM AND/OR: Triple Row Activation



Final State
 $AB + BC + AC$

$C(A + B) + \sim C(AB)$

In-DRAM NOT: Dual Contact Cell

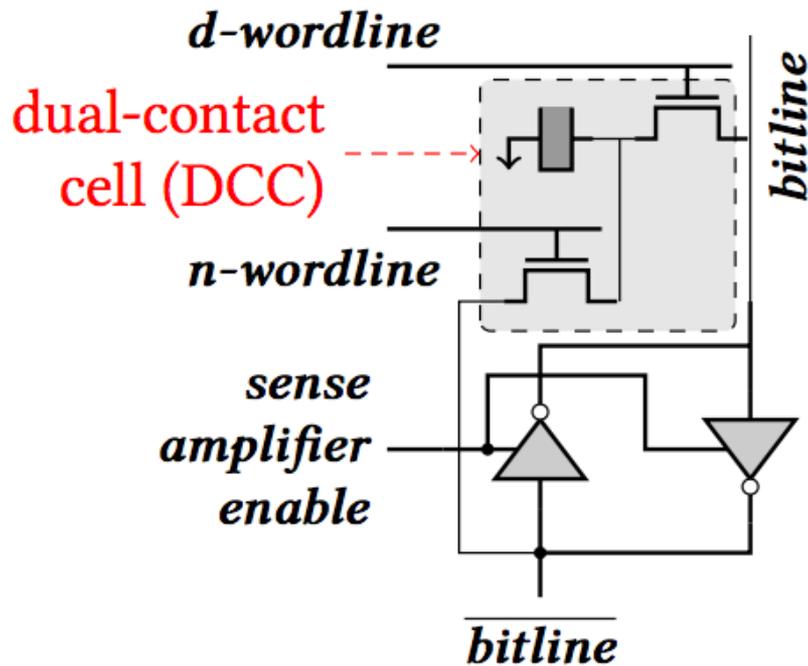


Figure 5: A dual-contact cell connected to both ends of a sense amplifier

Idea:
Feed the
negated value
in the sense amplifier
into a special row

Performance: In-DRAM Bitwise Operations

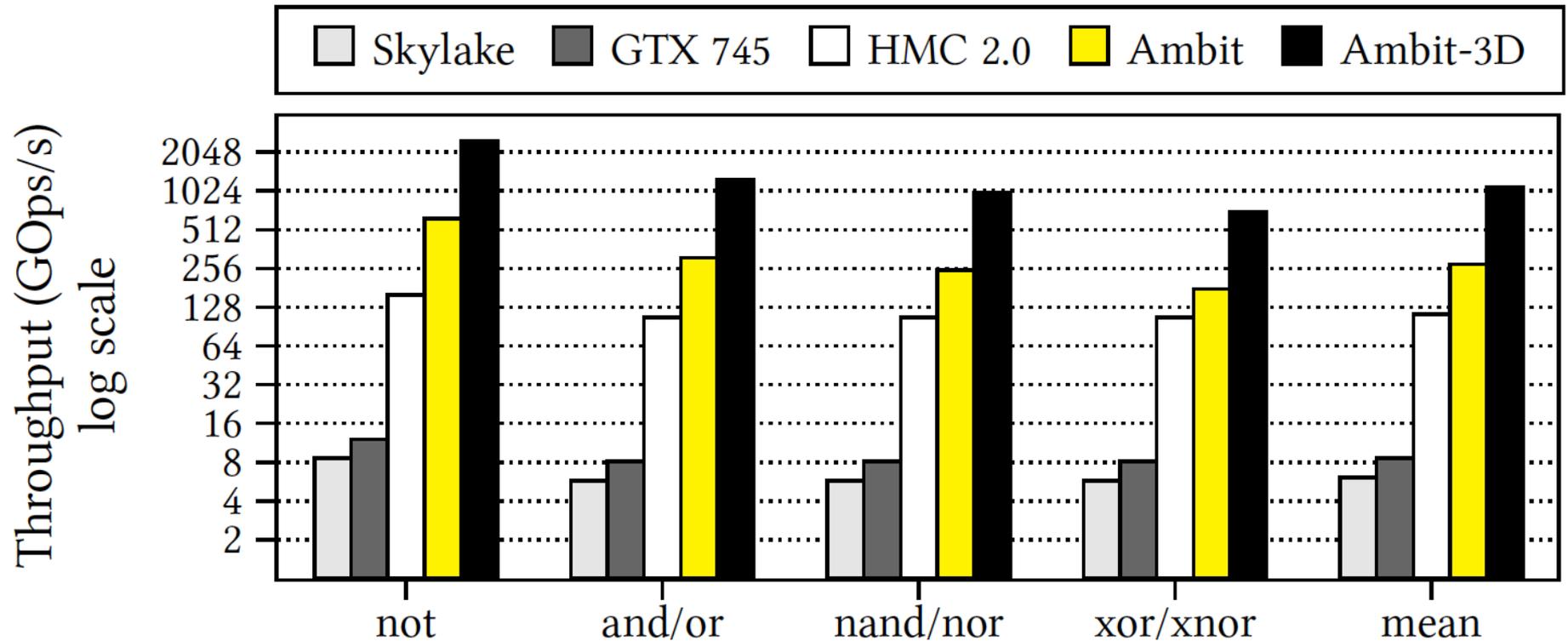


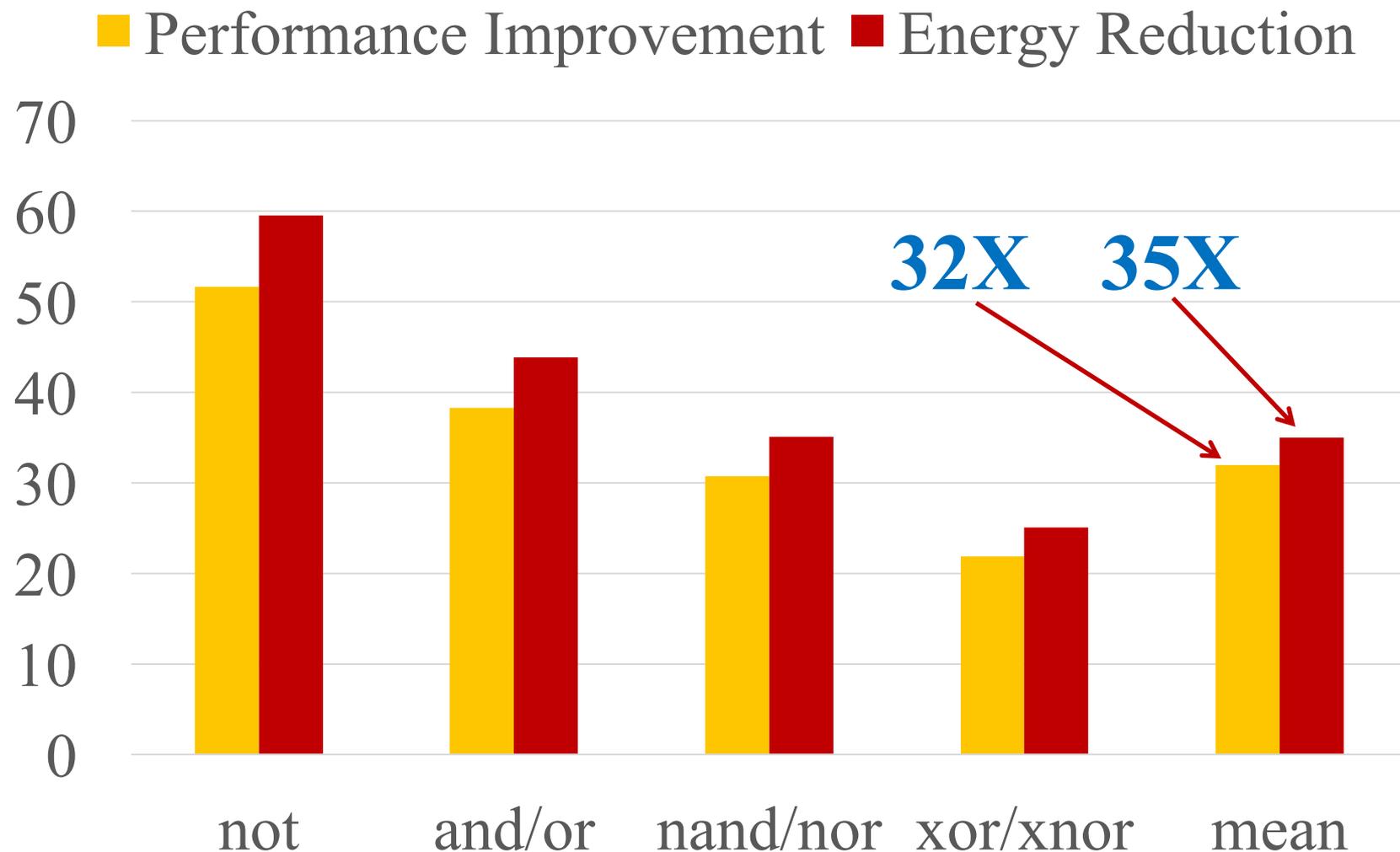
Figure 9: Throughput of bitwise operations on various systems.

Energy of In-DRAM Bitwise Operations

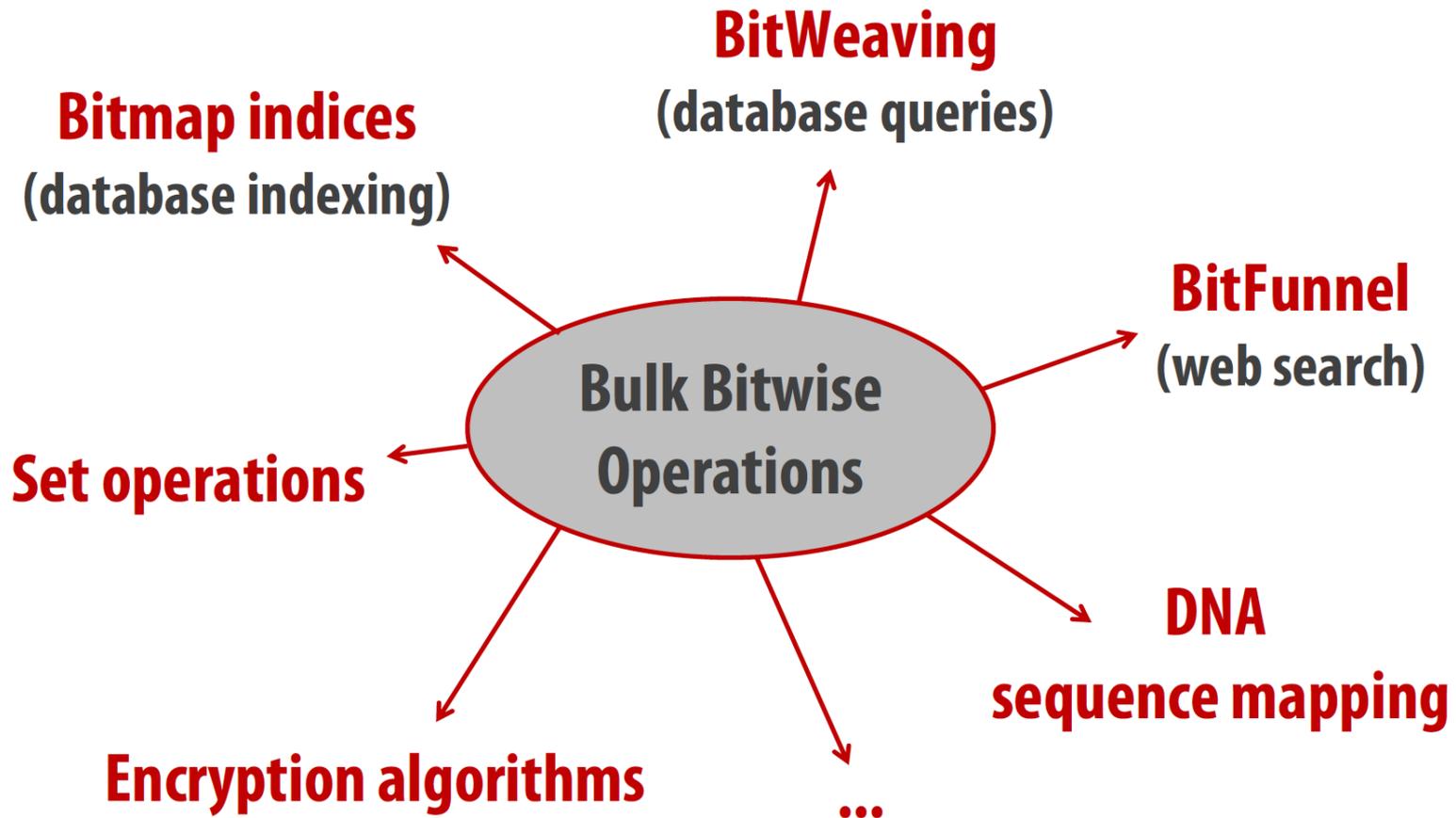
	Design	not	and/or	nand/nor	xor/xnor
DRAM &	DDR3	93.7	137.9	137.9	137.9
Channel Energy	Ambit	1.6	3.2	4.0	5.5
(nJ/KB)	(↓)	59.5X	43.9X	35.1X	25.1X

Table 3: Energy of bitwise operations. (↓) indicates energy reduction of Ambit over the traditional DDR3-based design.

Ambit vs. DDR3: Performance and Energy



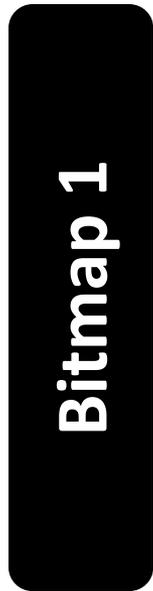
Bulk Bitwise Operations in Workloads



Example Data Structure: Bitmap Index

- Alternative to B-tree and its variants
- Efficient for performing *range queries* and *joins*
- **Many bitwise operations to perform a query**

age < 18 18 < age < 25 25 < age < 60 age > 60



Performance: Bitmap Index on Ambit

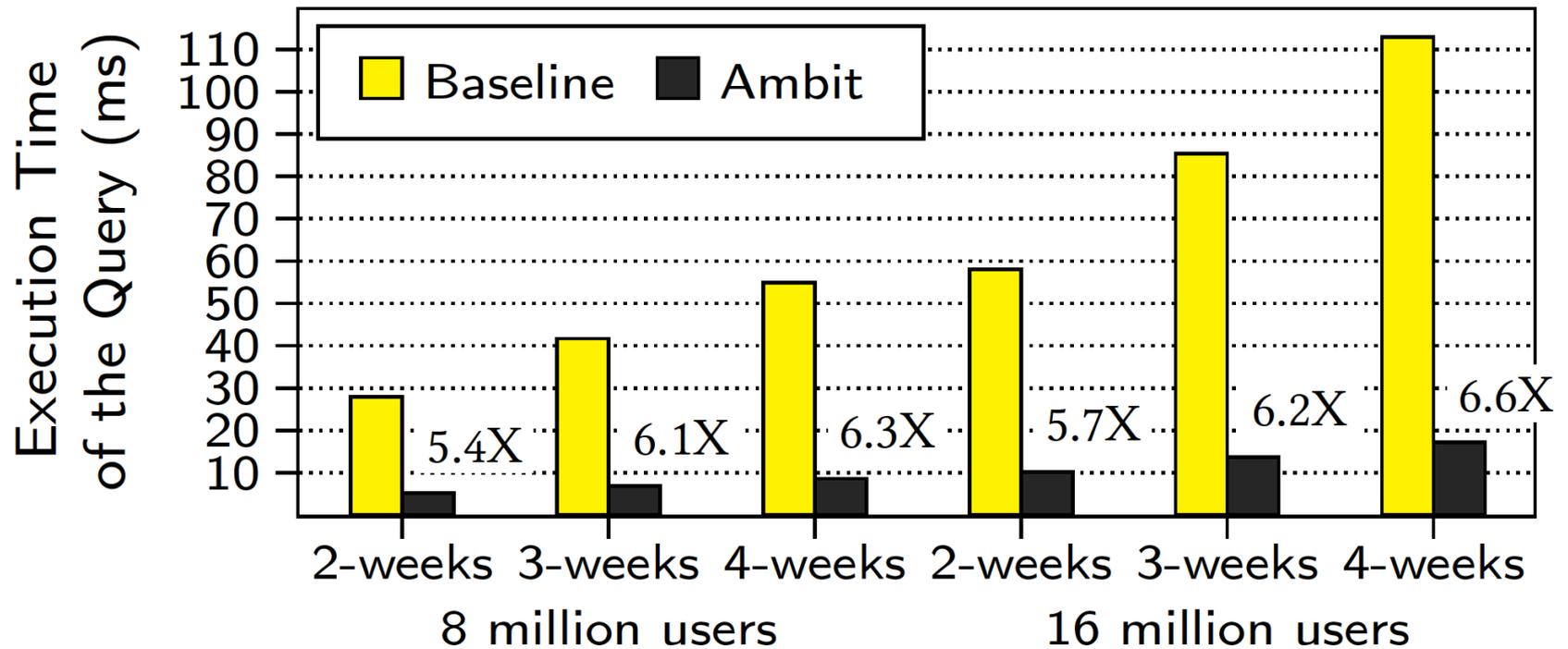


Figure 10: Bitmap index performance. The value above each bar indicates the reduction in execution time due to Ambit.

Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

Performance: BitWeaving on Ambit

```
'select count(*) from T where c1 <= val <= c2'
```

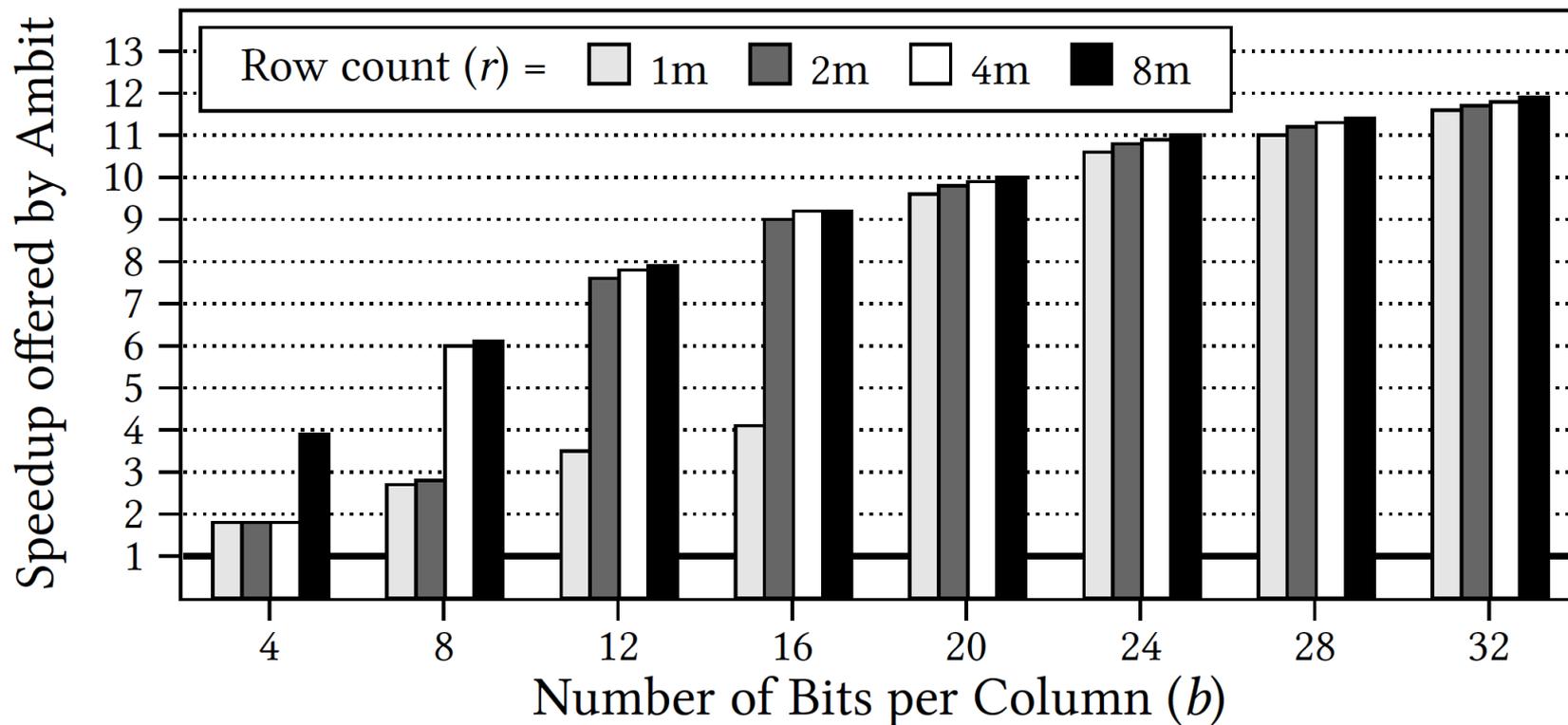


Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

More on In-DRAM Bulk AND/OR

- Vivek Seshadri, Kevin Hsieh, Amirali Boroumand, Donghyuk Lee, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
"Fast Bulk Bitwise AND and OR in DRAM"
IEEE Computer Architecture Letters (**CAL**), April 2015.

Fast Bulk Bitwise AND and OR in DRAM

Vivek Seshadri*, Kevin Hsieh*, Amirali Boroumand*, Donghyuk Lee*,
Michael A. Kozuch†, Onur Mutlu*, Phillip B. Gibbons†, Todd C. Mowry*

*Carnegie Mellon University †Intel Pittsburgh

More on Ambit

- Vivek Seshadri et al., “**Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology**,” MICRO 2017.

Ambit: In-Memory Accelerator for Bulk Bitwise Operations
Using Commodity DRAM Technology

Vivek Seshadri^{1,5} Donghyuk Lee^{2,5} Thomas Mullins^{3,5} Hasan Hassan⁴ Amirali Boroumand⁵
Jeremie Kim^{4,5} Michael A. Kozuch³ Onur Mutlu^{4,5} Phillip B. Gibbons⁵ Todd C. Mowry⁵

¹Microsoft Research India ²NVIDIA Research ³Intel ⁴ETH Zürich ⁵Carnegie Mellon University

Computing Architectures with Minimal Data Movement

Challenge: Intelligent Memory Device

Does **memory**
have to be
dumb?

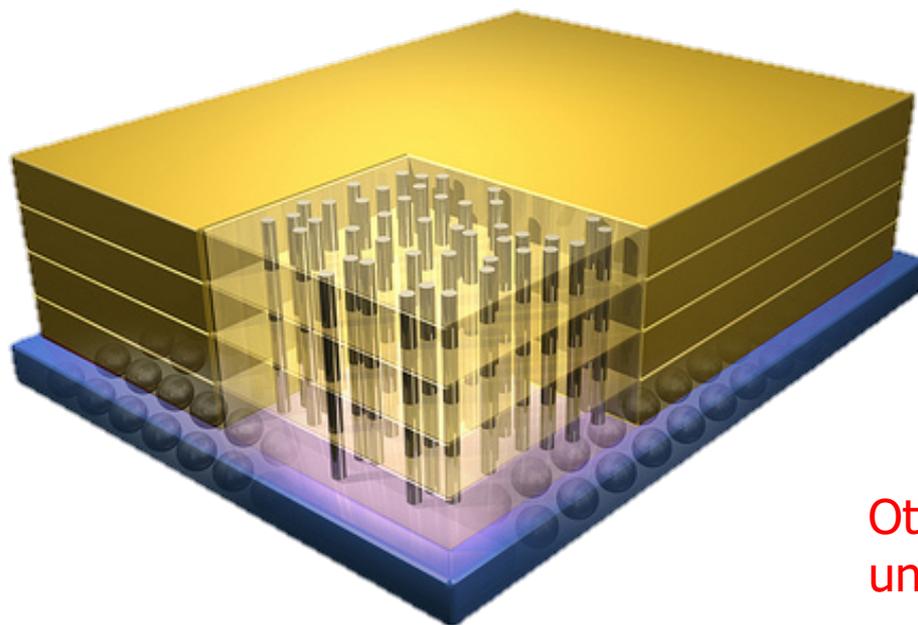
Processing in Memory: Two Approaches

1. Minimally changing memory chips
2. Exploiting 3D-stacked memory

Opportunity: 3D-Stacked Logic+Memory



Hybrid Memory Cube
C O N S O R T I U M



Memory

Logic

Other "True 3D" technologies
under development

DRAM Landscape (circa 2015)

<i>Segment</i>	<i>DRAM Standards & Architectures</i>
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLD RAM3 (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

Kim+, "Ramulator: A Flexible and Extensible DRAM Simulator", IEEE CAL 2015.

Two Key Questions in 3D-Stacked PIM

- 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?

- What is the **minimal processing-in-memory support** we can provide?
 - without changing the system significantly
 - while achieving significant benefits

Graph Processing

- Large graphs are everywhere (circa 2015)



36 Million
Wikipedia Pages



1.4 Billion
Facebook Users

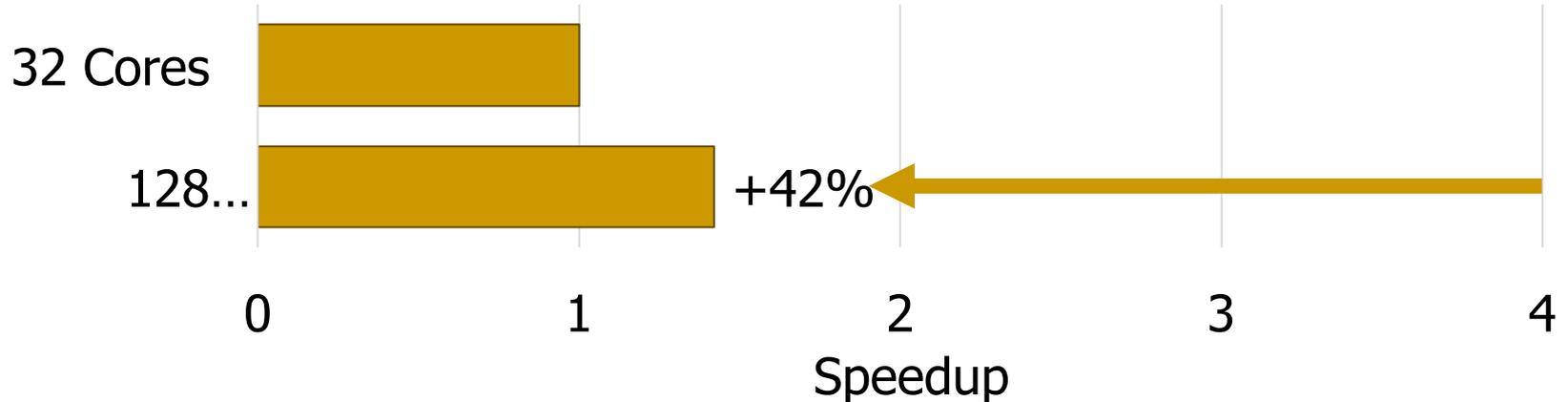


300 Million
Twitter Users



30 Billion
Instagram Photos

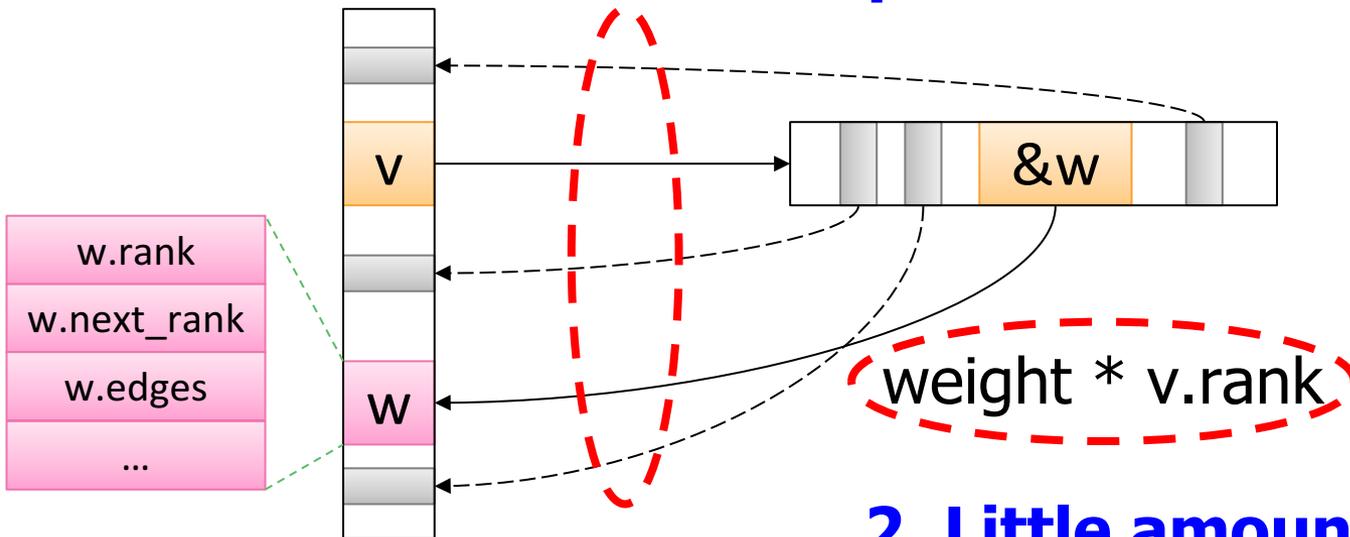
- Scalable large-scale graph processing is challenging



Key Bottlenecks in Graph Processing

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

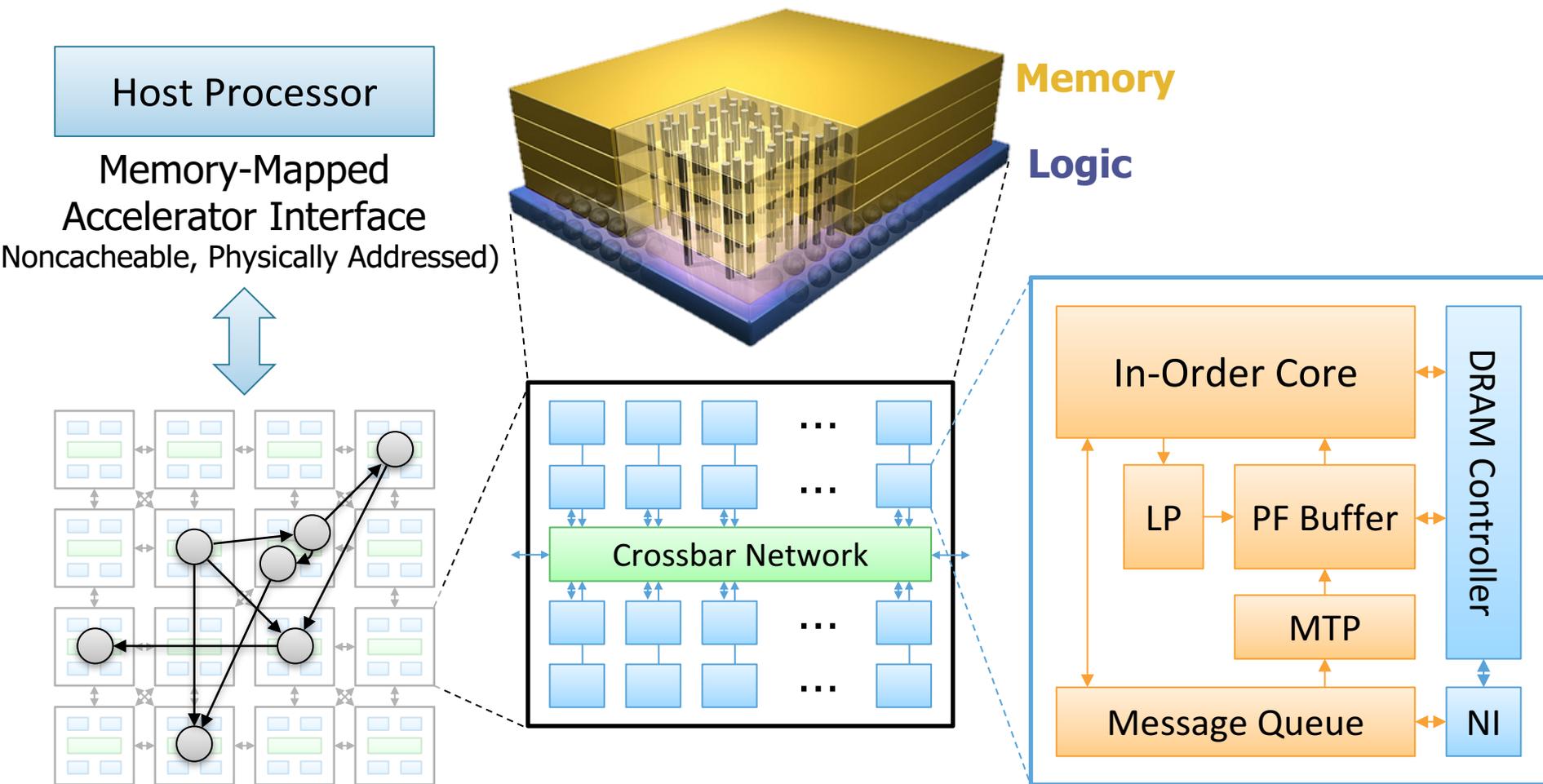
1. Frequent random memory accesses



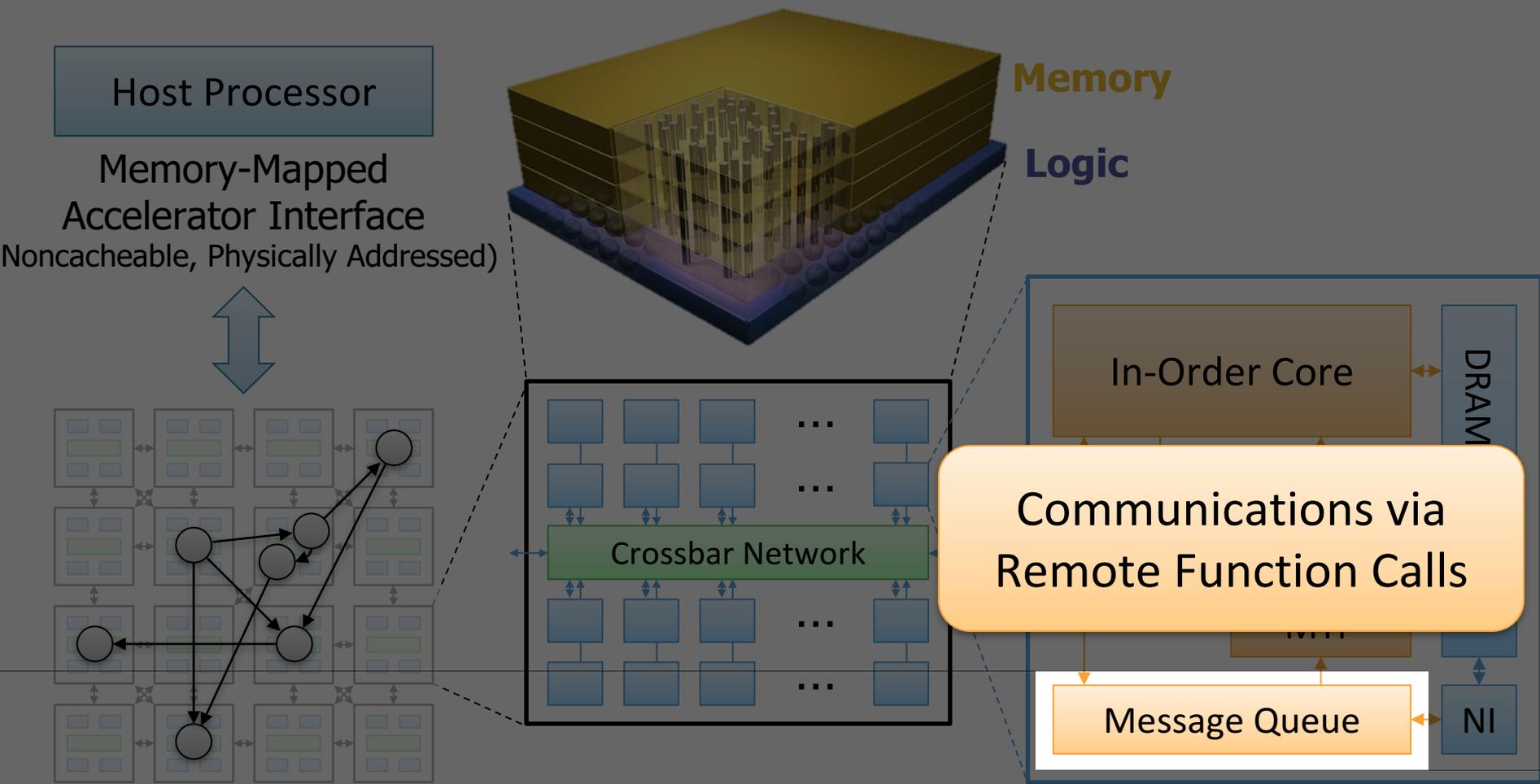
2. Little amount of computation

Tesseract System for Graph Processing

Interconnected set of 3D-stacked memory+logic chips with simple cores

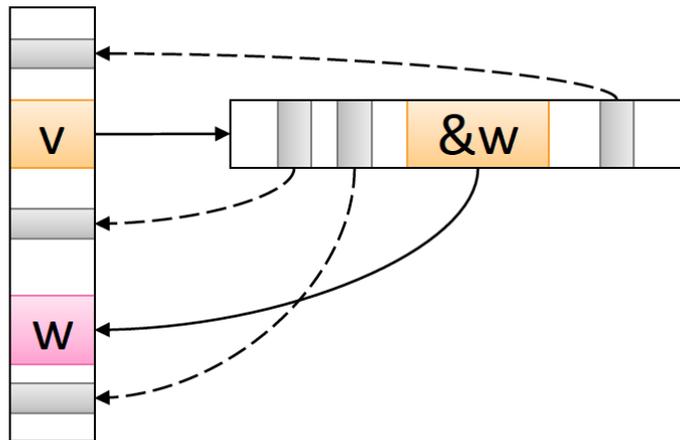


Tesseract System for Graph Processing



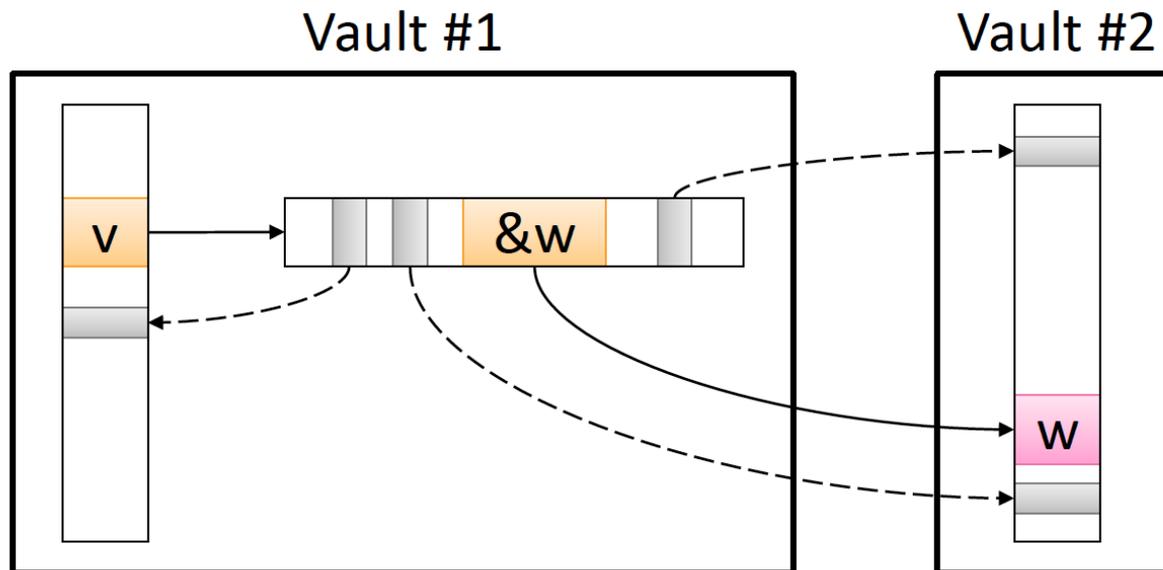
Communications In Tesseract (I)

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



Communications In Tesseract (II)

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

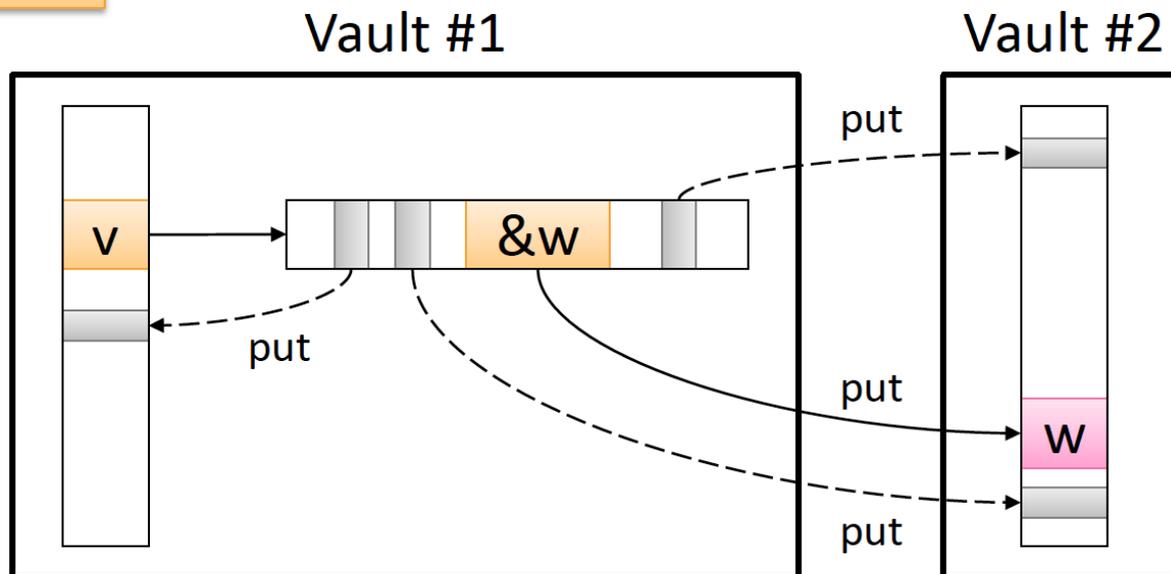


Communications In Tesseract (III)

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

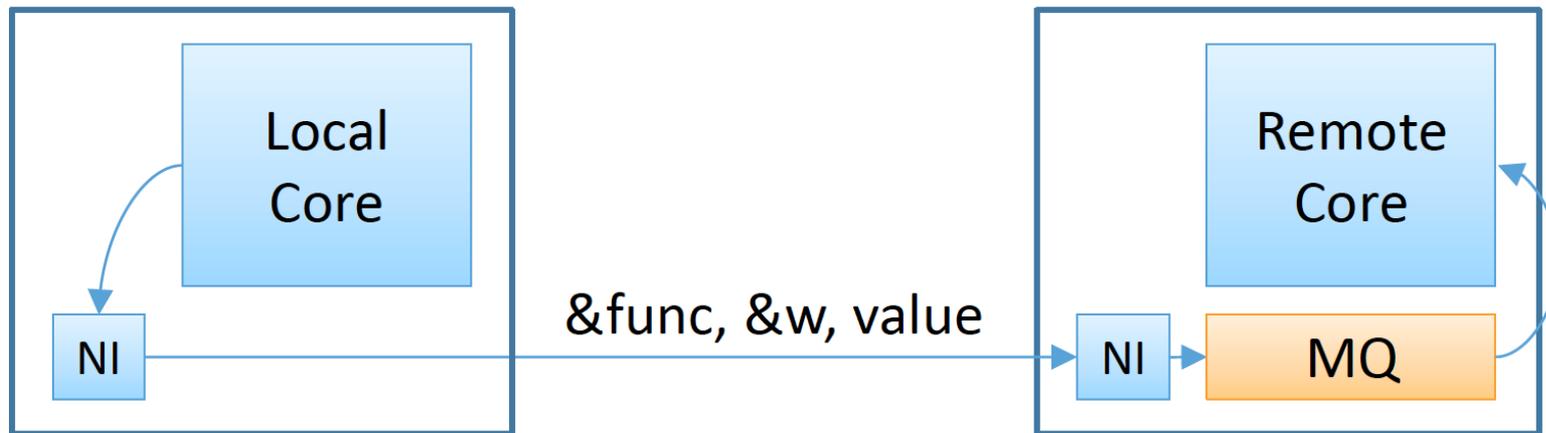
Non-blocking Remote Function Call

Can be **delayed** until the nearest barrier



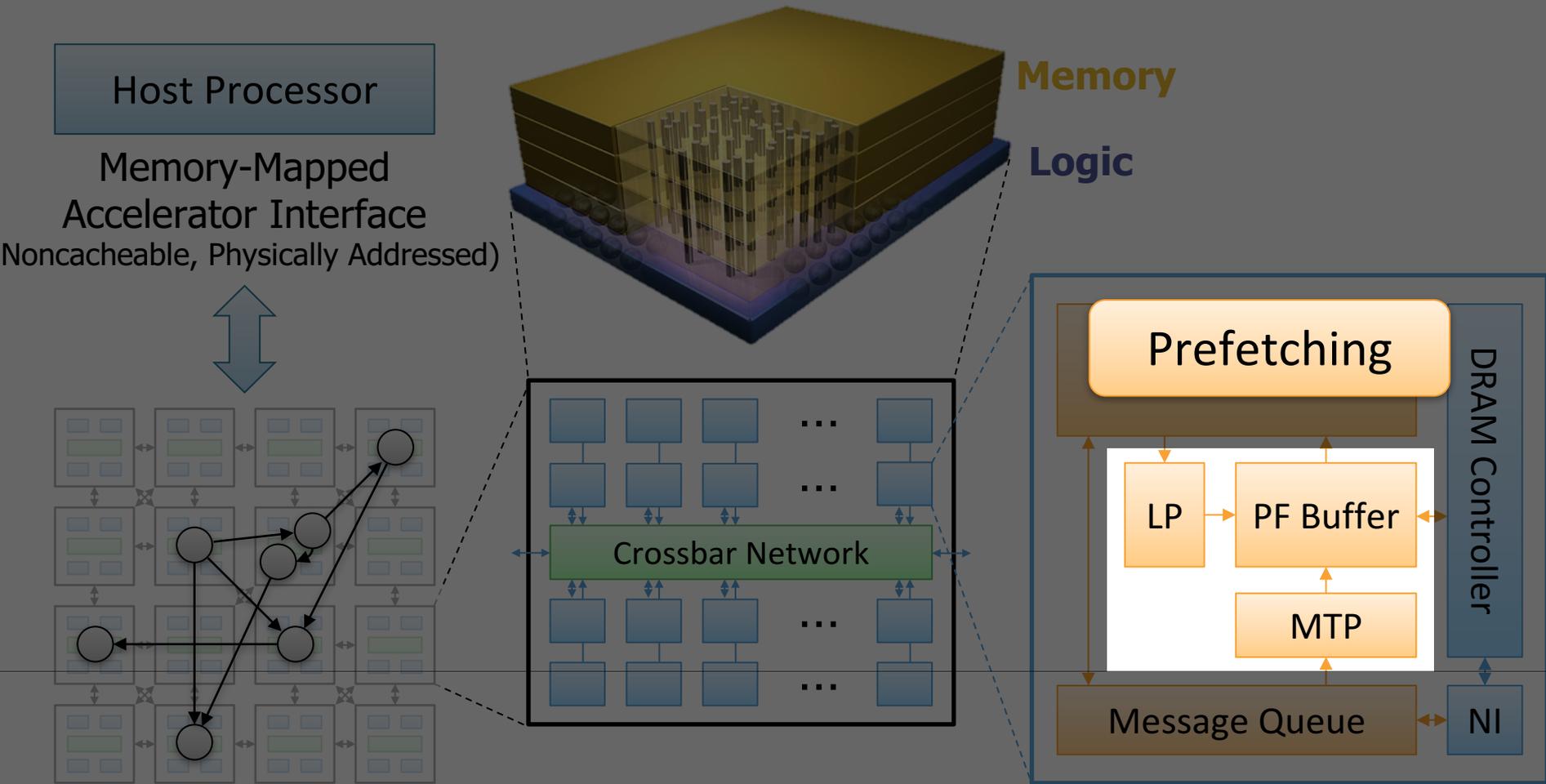
Remote Function Call (Non-Blocking)

1. Send function address & args to the remote core
2. Store the incoming message to the message queue
3. Flush the message queue when it is full or a synchronization barrier is reached



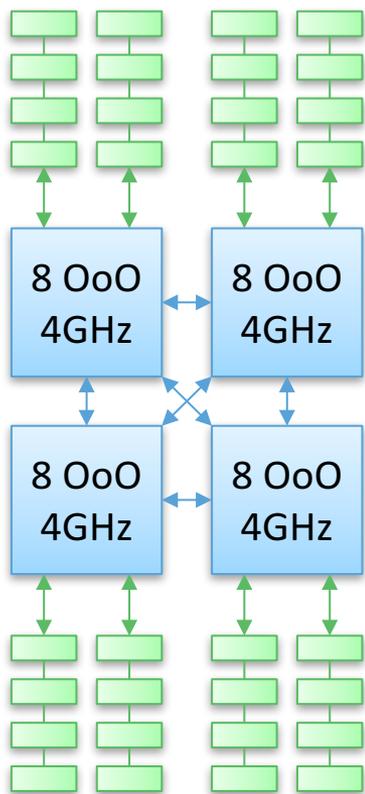
```
put(w.id, function() { w.next_rank += value; })
```

Tesseract System for Graph Processing



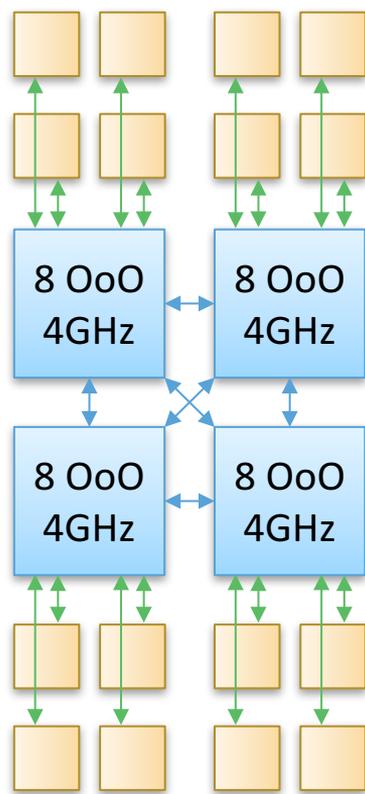
Evaluated Systems

DDR3-OoO



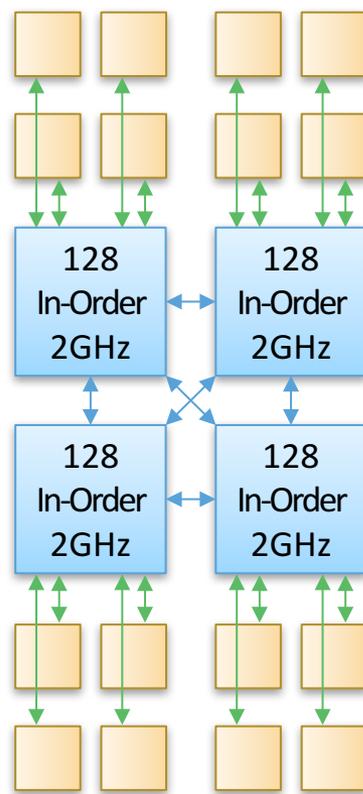
102.4GB/s

HMC-OoO



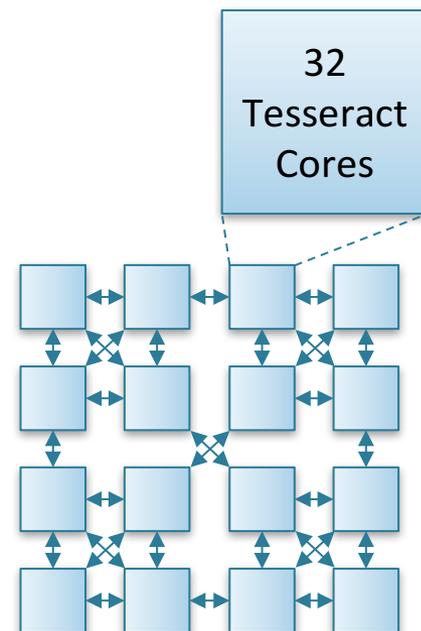
640GB/s

HMC-MC



640GB/s

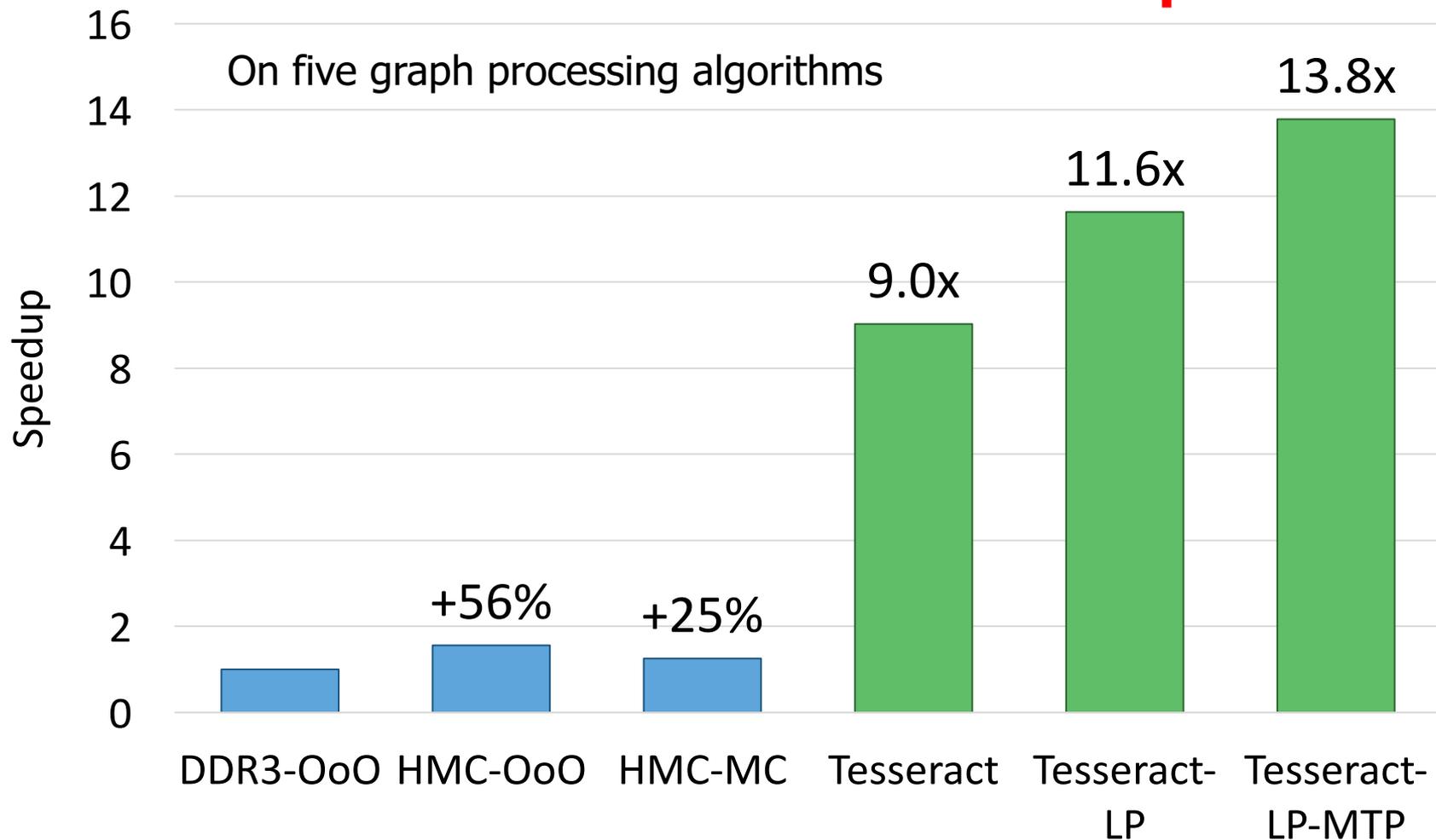
Tesseract



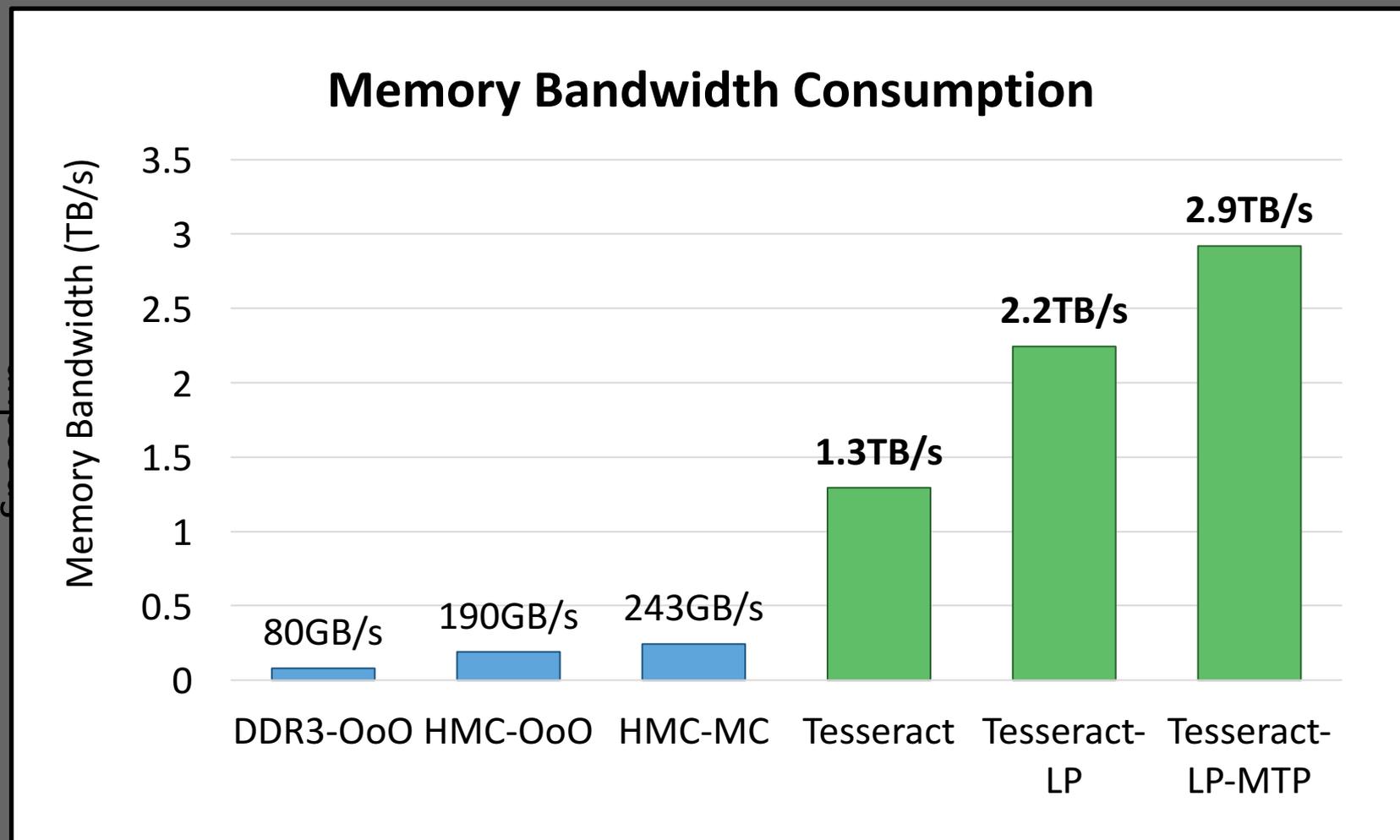
8TB/s

Tesseract Graph Processing Performance

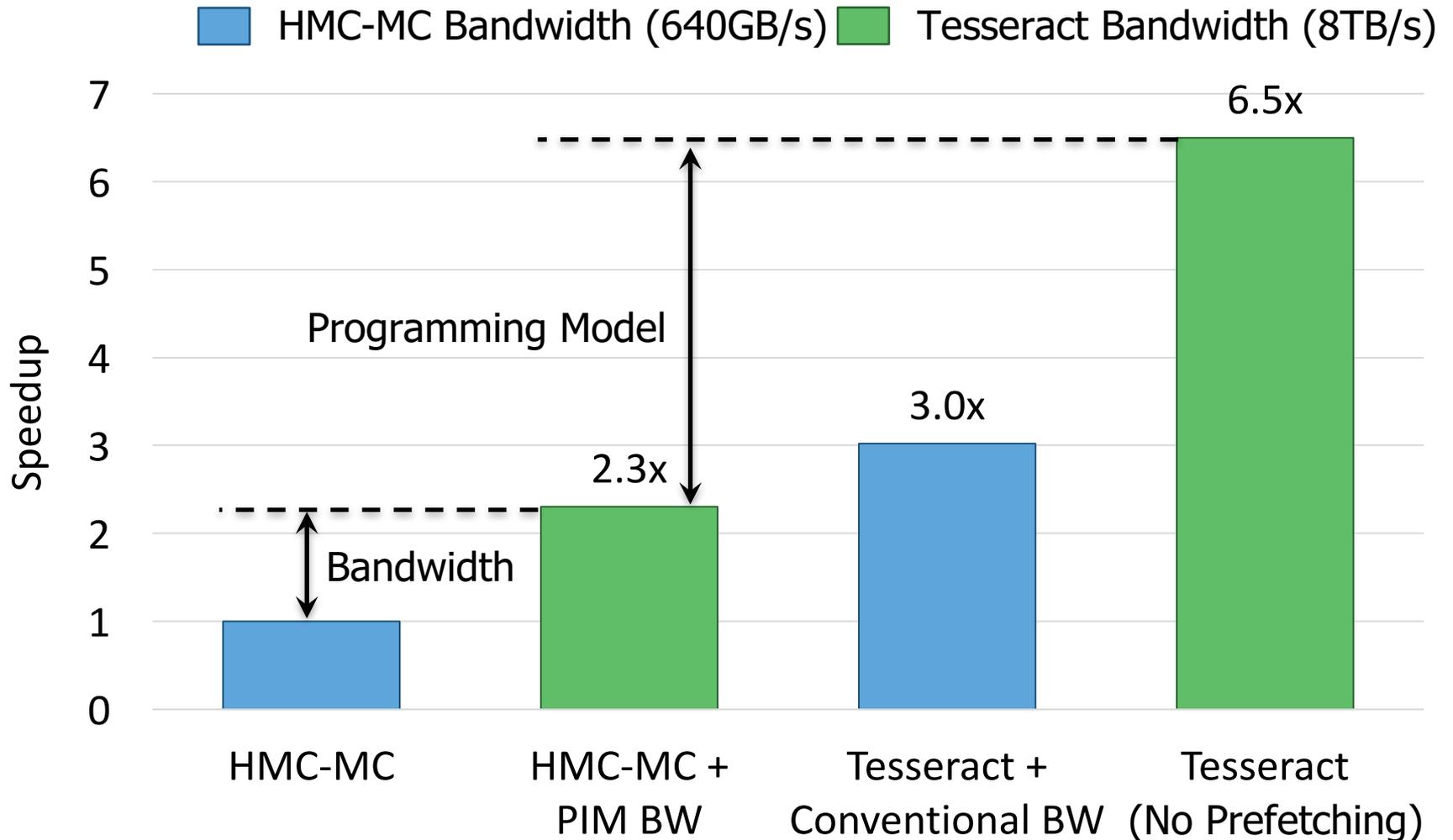
>13X Performance Improvement



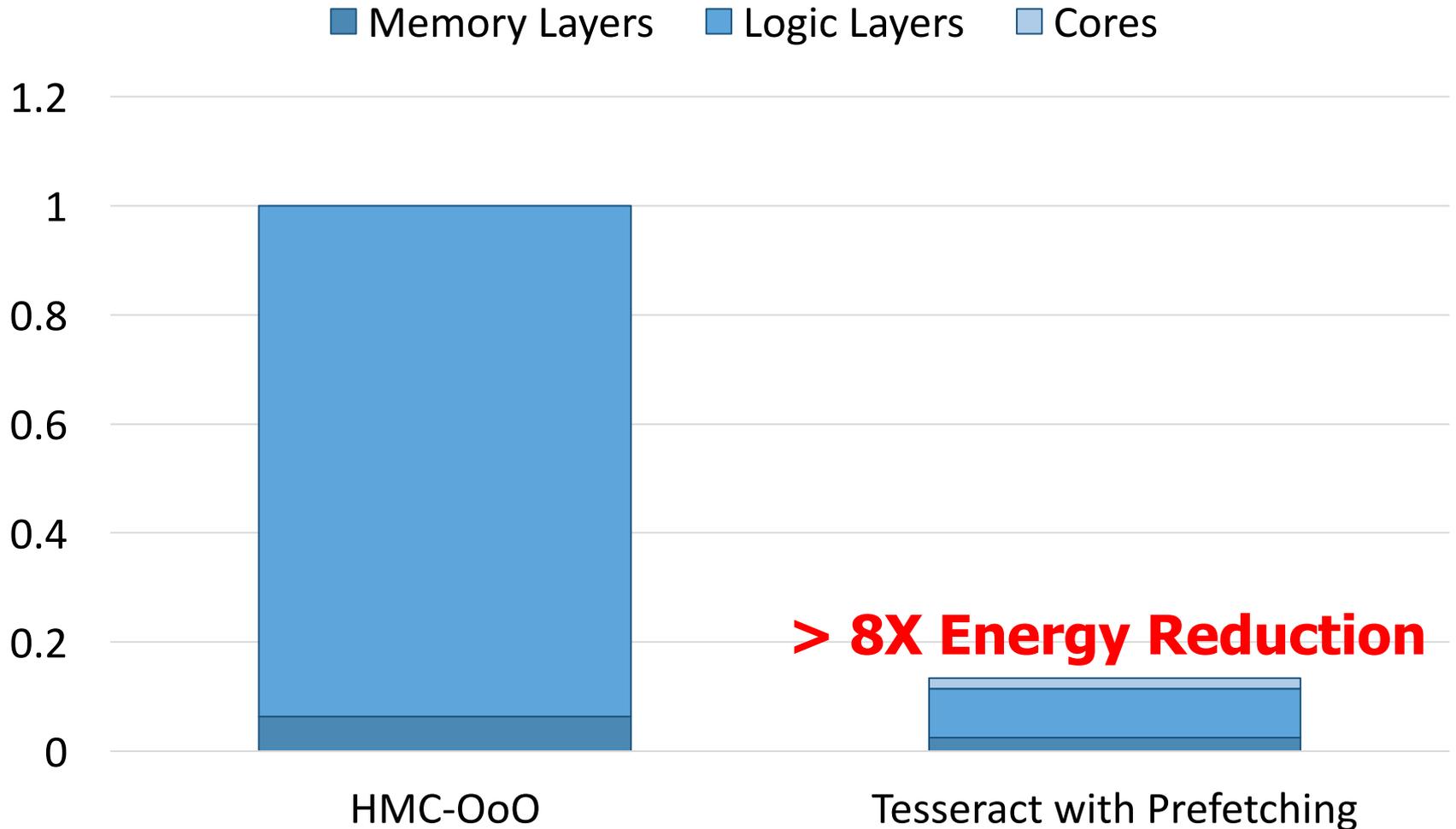
Tesseract Graph Processing Performance



Effect of Bandwidth & Programming Model



Tesseract Graph Processing System Energy



More on Tesseract

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi,
"A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing"
Proceedings of the 42nd International Symposium on Computer Architecture (ISCA), Portland, OR, June 2015.
[[Slides \(pdf\)](#)] [[Lightning Session Slides \(pdf\)](#)]

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

Junwhan Ahn Sungpack Hong[§] Sungjoo Yoo Onur Mutlu[†] Kiyoung Choi
junwhan@snu.ac.kr, sungpack.hong@oracle.com, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

[§]Oracle Labs

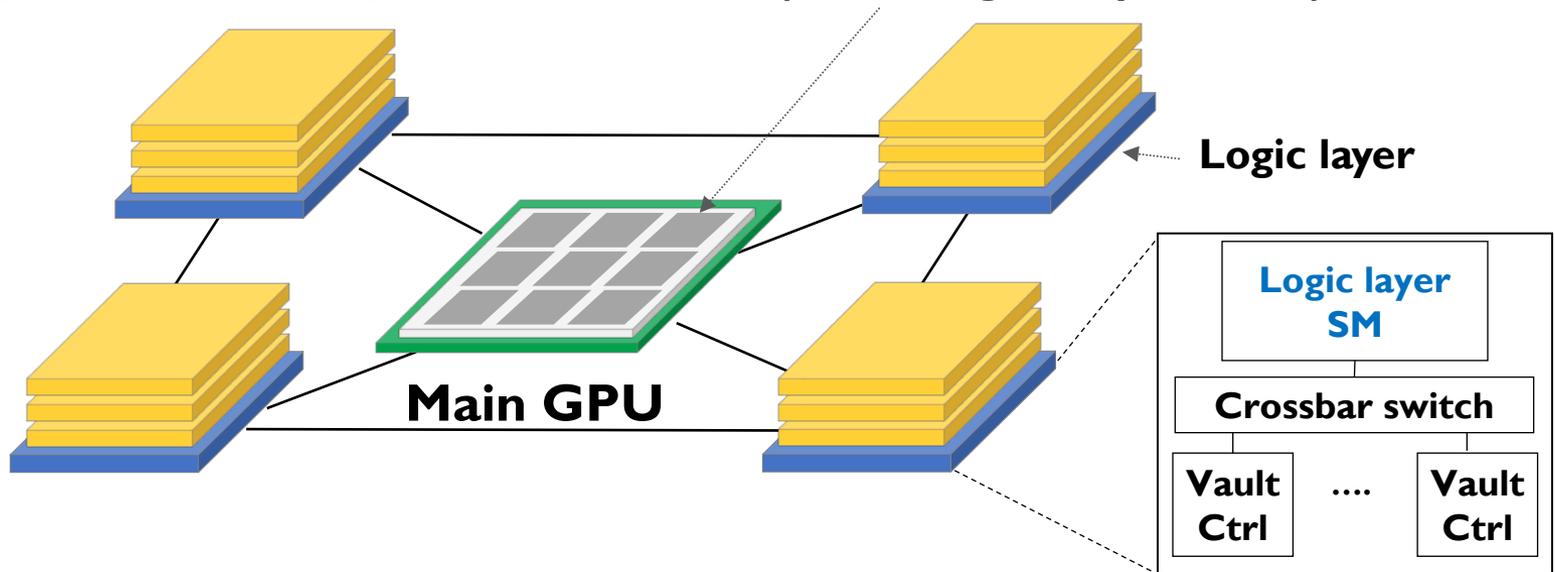
[†]Carnegie Mellon University

Truly Distributed GPU Processing with PIM?

```
__global__  
void applyScaleFactorsKernel( uint8_T * const out,  
                             uint8_T const * const in, const double *factor,  
                             size_t const numRows, size_t const numCols )  
{  
    // Work out which pixel we are working on.  
    const int rowIdx = blockIdx.x * blockDim.x + threadIdx.x;  
    const int colIdx = blockIdx.y;  
    const int sliceIdx = threadIdx.z;  
  
    // Check this thread isn't off the image  
    if( rowIdx >= numRows ) return;  
  
    // Compute the index of my element  
    size_t linearIdx = rowIdx + colIdx*numRows +  
                      sliceIdx*numRows*numCols;
```

**3D-stacked memory
(memory stack)**

SM (Streaming Multiprocessor)



Accelerating GPU Execution with PIM (I)

- Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler, **"Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"**

Proceedings of the 43rd International Symposium on Computer Architecture (ISCA), Seoul, South Korea, June 2016.

[[Slides \(pptx\)](#) ([pdf](#))]

[[Lightning Session Slides \(pptx\)](#) ([pdf](#))]

Transparent Offloading and Mapping (TOM):

Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh[‡] Eiman Ebrahimi[†] Gwangsun Kim* Niladrish Chatterjee[†] Mike O'Connor[†]
Nandita Vijaykumar[‡] Onur Mutlu^{§‡} Stephen W. Keckler[†]

[‡]Carnegie Mellon University [†]NVIDIA ^{*}KAIST [§]ETH Zürich

Accelerating GPU Execution with PIM (II)

- Ashutosh Pattnaik, Xulong Tang, Adwait Jog, Onur Kayiran, Asit K. Mishra, Mahmut T. Kandemir, Onur Mutlu, and Chita R. Das, **"Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities"**
Proceedings of the 25th International Conference on Parallel Architectures and Compilation Techniques (PACT), Haifa, Israel, September 2016.

Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities

Ashutosh Pattnaik¹ Xulong Tang¹ Adwait Jog² Onur Kayiran³
Asit K. Mishra⁴ Mahmut T. Kandemir¹ Onur Mutlu^{5,6} Chita R. Das¹

¹Pennsylvania State University ²College of William and Mary
³Advanced Micro Devices, Inc. ⁴Intel Labs ⁵ETH Zürich ⁶Carnegie Mellon University

Accelerating Linked Data Structures

- Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K. Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu,
"Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation"
Proceedings of the 34th IEEE International Conference on Computer Design (ICCD), Phoenix, AZ, USA, October 2016.

Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation

Kevin Hsieh[†] Samira Khan[‡] Nandita Vijaykumar[†]
Kevin K. Chang[†] Amirali Boroumand[†] Saugata Ghose[†] Onur Mutlu^{§†}
[†]*Carnegie Mellon University* [‡]*University of Virginia* [§]*ETH Zürich*

Two Key Questions in 3D-Stacked PIM

- 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?
- What is the **minimal processing-in-memory support** we can provide?
 - without changing the system significantly
 - while achieving significant benefits

PIM-Enabled Instructions

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi, **"PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture"** *Proceedings of the 42nd International Symposium on Computer Architecture (ISCA)*, Portland, OR, June 2015. [[Slides \(pdf\)](#)] [[Lightning Session Slides \(pdf\)](#)]

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

Junwhan Ahn Sungjoo Yoo Onur Mutlu[†] Kiyoung Choi

junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

[†]Carnegie Mellon University

Automatic Code and Data Mapping?

- Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler, **"Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"**

Proceedings of the 43rd International Symposium on Computer Architecture (ISCA), Seoul, South Korea, June 2016.

[[Slides \(pptx\)](#) ([pdf](#))]

[[Lightning Session Slides \(pptx\)](#) ([pdf](#))]

Transparent Offloading and Mapping (TOM):

Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh[‡] Eiman Ebrahimi[†] Gwangsun Kim* Niladrish Chatterjee[†] Mike O'Connor[†]
Nandita Vijaykumar[‡] Onur Mutlu^{§‡} Stephen W. Keckler[†]

[‡]Carnegie Mellon University [†]NVIDIA *KAIST §ETH Zürich

Fundamentally Energy-Efficient (Data-Centric)

Computing Architectures

Fundamentally

Low-Latency

(Data-Centric)

Computing Architectures

Three Key Issues in Future Platforms

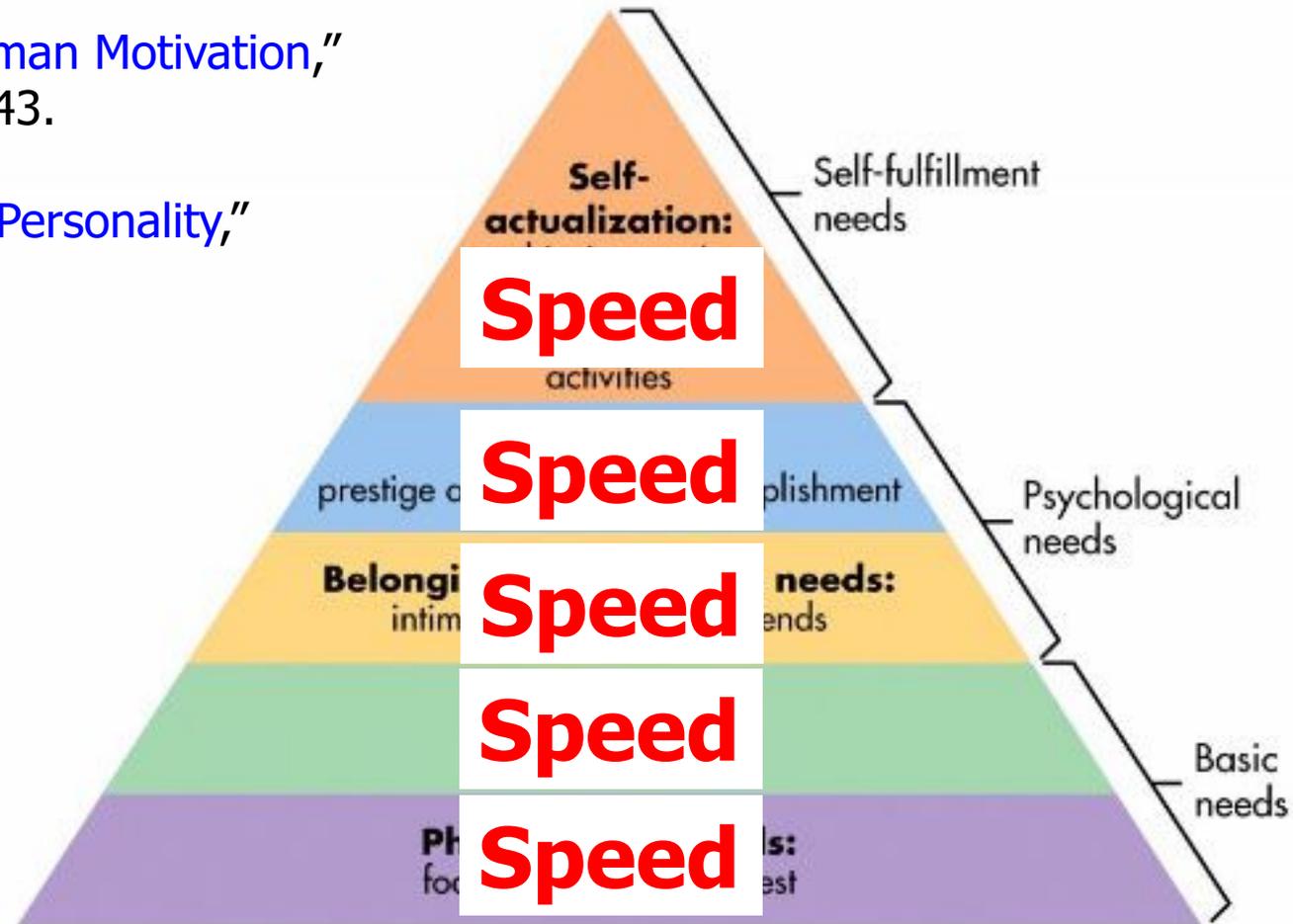
- Fundamentally **Secure/Reliable/Safe** Architectures
- Fundamentally **Energy-Efficient** Architectures
 - **Memory-centric** (Data-centric) Architectures
- Fundamentally **Low Latency** Architectures



Maslow's Hierarchy of Needs, A Third Time

Maslow, "A Theory of Human Motivation,"
Psychological Review, 1943.

Maslow, "Motivation and Personality,"
Book, 1954-1970.



See Backup Slides for Latency...

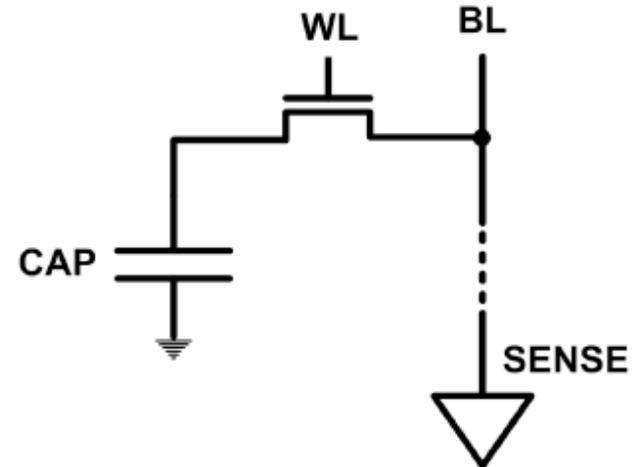
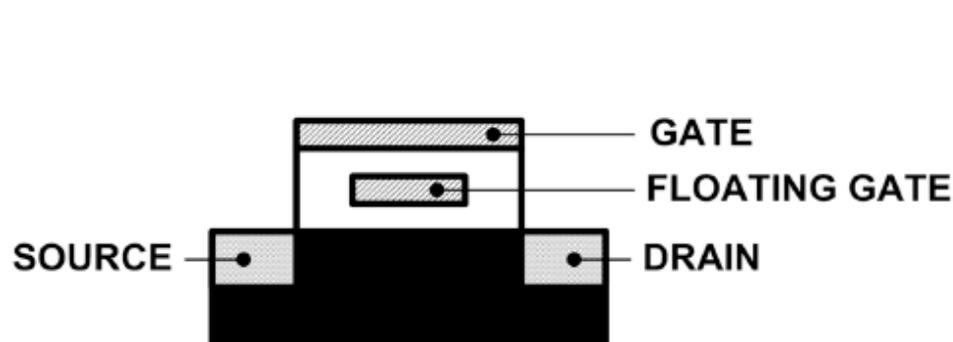
Fundamentally Low-Latency Computing Architectures

Agenda

- Major Trends Affecting Main Memory
- The Memory Scaling Problem and Solution Directions
 - New Memory Architectures
 - Enabling Emerging Technologies
- Cross-Cutting Principles
- Summary

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

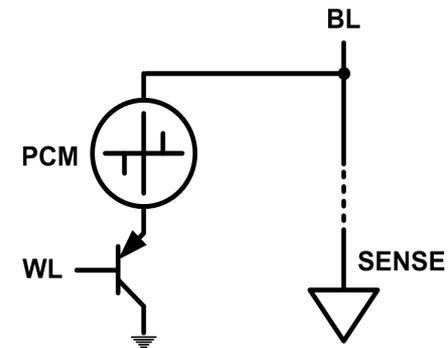


Emerging Memory Technologies

- 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?

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

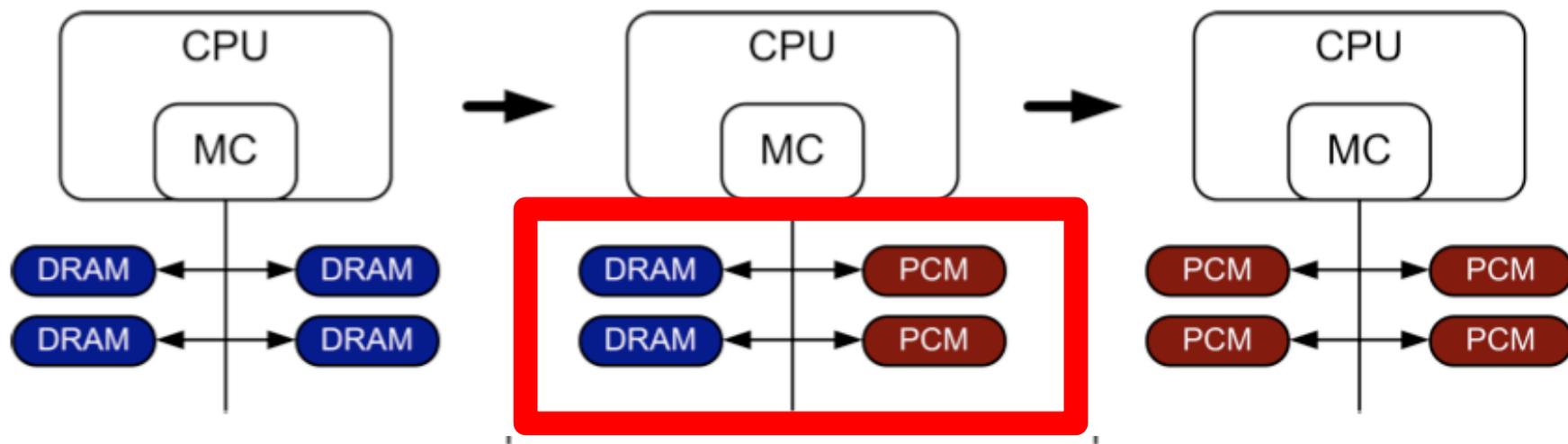
- Memristors/RRAM/ReRAM
 - Inject current to change **atomic structure**
 - Resistance determined by atom distance

Phase Change Memory: Pros and Cons

- Pros over DRAM
 - Better technology scaling (capacity and cost)
 - Non volatile → Persistent
 - 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 (I)

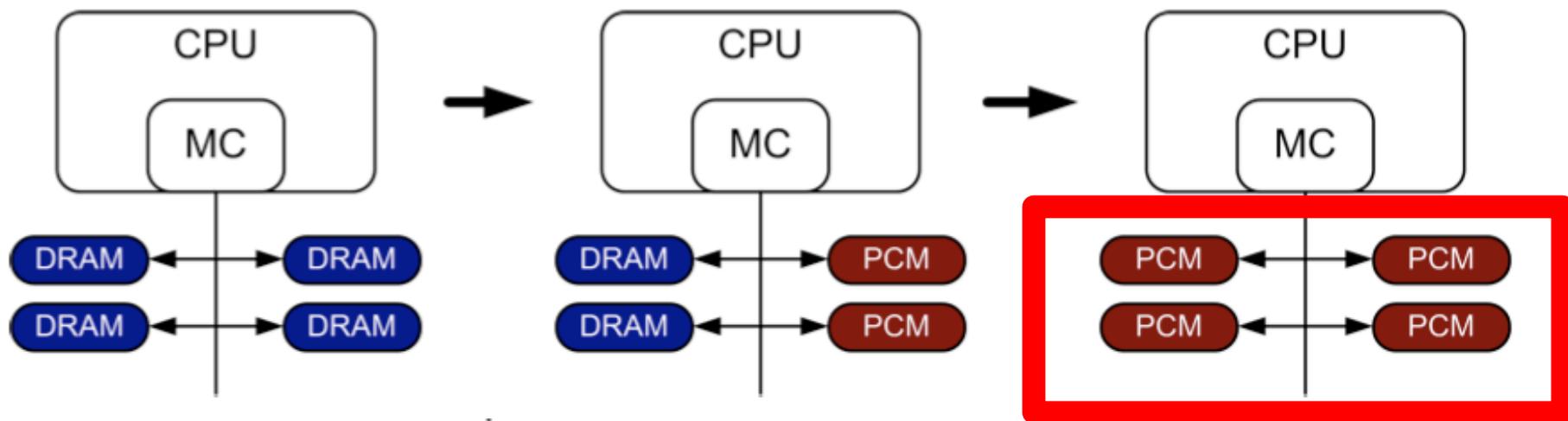
- How should PCM-based (main) memory be organized?



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

PCM-based Main Memory (II)

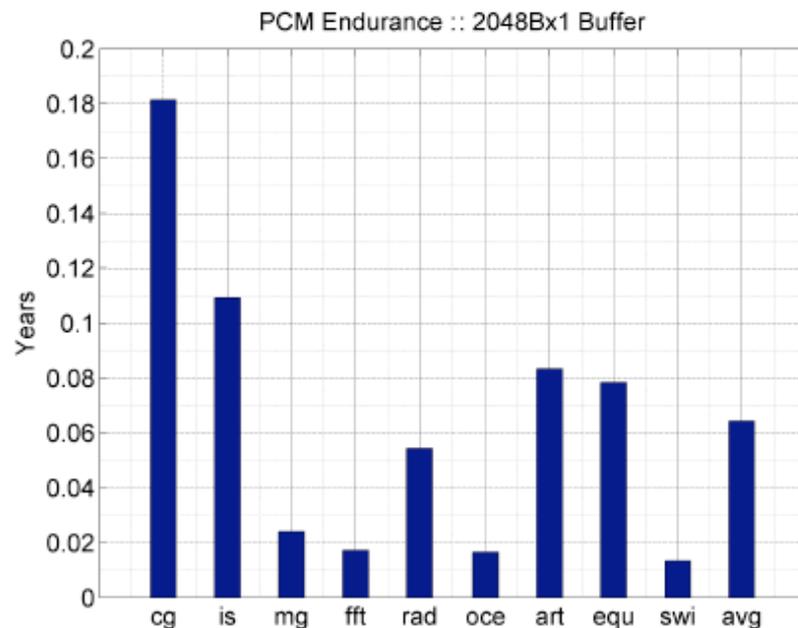
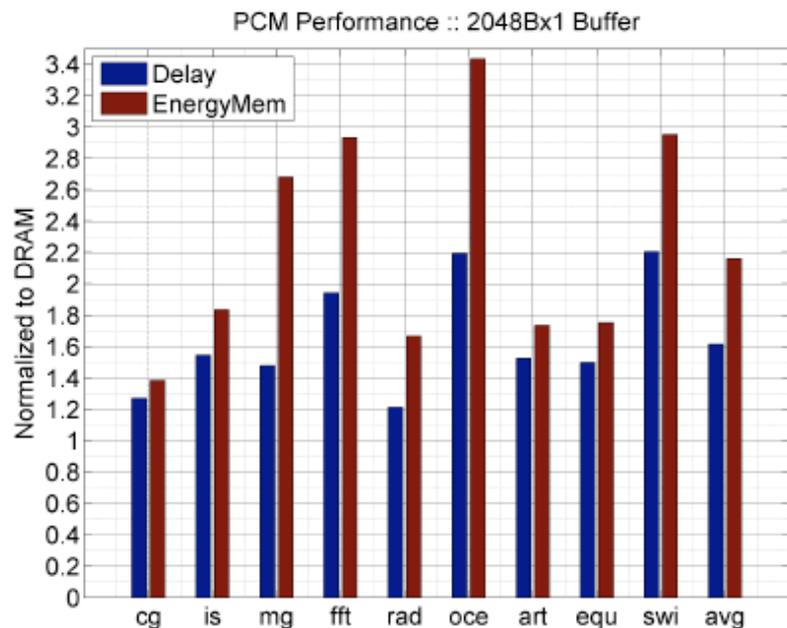
- How should PCM-based (main) memory be organized?



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

Results: Naïve Replacement of DRAM with PCM

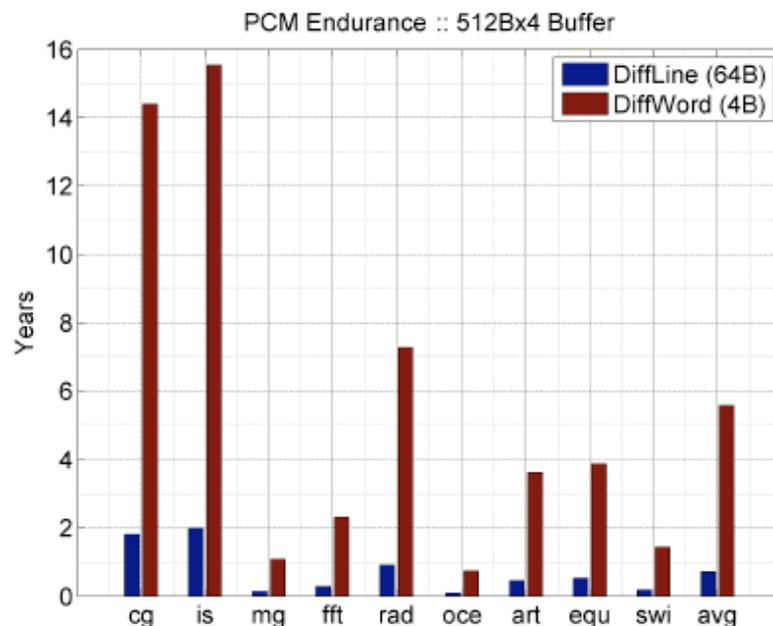
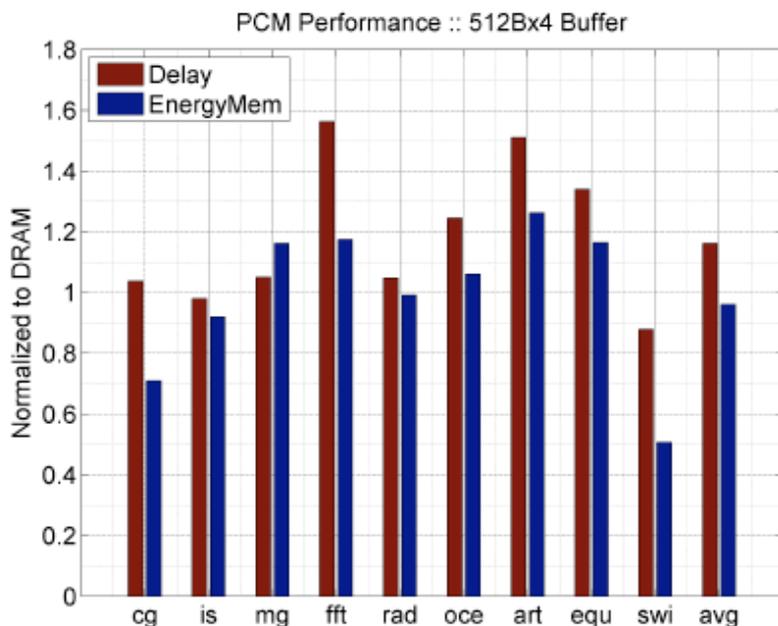
- Replace DRAM with PCM in a 4-core, 4MB L2 system
- PCM organized the same as DRAM: row buffers, banks, peripherals
- 1.6x delay, 2.2x energy, 500-hour average lifetime



- 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?

More on PCM As Main Memory

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

Architecting Phase Change Memory as a Scalable DRAM Alternative

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

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

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

More on PCM As Main Memory (II)

- Benjamin C. Lee, Ping Zhou, Jun Yang, Youtao Zhang, Bo Zhao, Engin Ipek, Onur Mutlu, and Doug Burger,
"Phase Change Technology and the Future of Main Memory"
IEEE Micro, Special Issue: Micro's Top Picks from 2009 Computer Architecture Conferences (MICRO TOP PICKS), Vol. 30, No. 1, pages 60-70, January/February 2010.

PHASE-CHANGE TECHNOLOGY AND THE FUTURE OF MAIN MEMORY

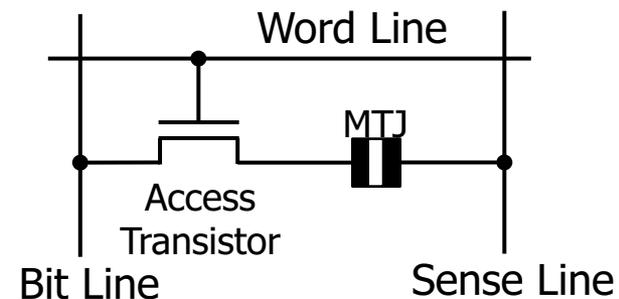
STT-MRAM as Main Memory

- Magnetic Tunnel Junction (MTJ) device
 - Reference layer: Fixed magnetic orientation
 - Free layer: Parallel or anti-parallel
- Magnetic orientation of the free layer determines logical state of device
 - High vs. low resistance
- Write: Push large current through MTJ to change orientation of free layer
- Read: Sense current flow
- Kultursay et al., "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," ISPASS 2013.

Logical 0



Logical 1



STT-MRAM: Pros and Cons

■ Pros over DRAM

- Better technology scaling (capacity and cost)
- Non volatile → Persistent
- Low idle power (no refresh)

■ Cons

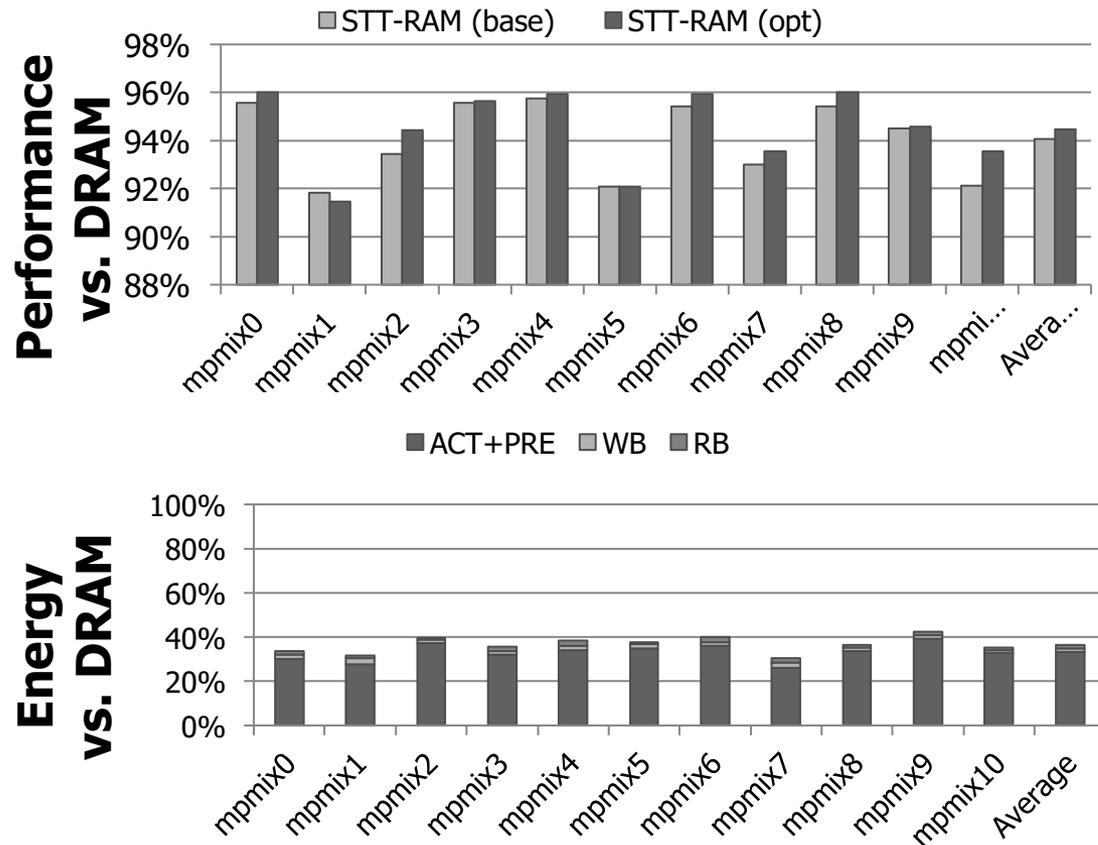
- Higher write latency
- Higher write energy
- Poor density (currently)
- Reliability?

■ Another level of freedom

- Can trade off non-volatility for lower write latency/energy (by reducing the size of the MTJ)

Architected STT-MRAM as Main Memory

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



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

More on STT-MRAM as Main Memory

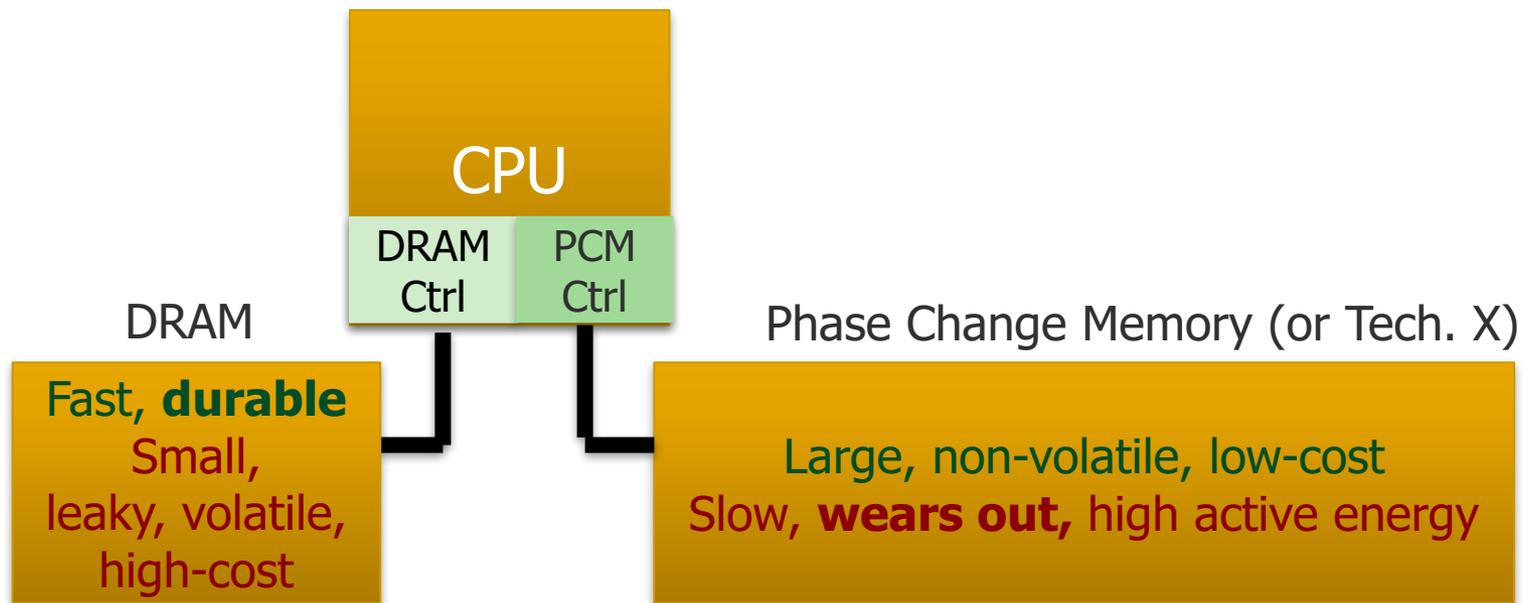
- Emre Kultursay, Mahmut Kandemir, Anand Sivasubramaniam, and Onur Mutlu,
"Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative"
Proceedings of the 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, TX, April 2013. Slides (pptx) (pdf)

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

Emre Kültürsay*, Mahmut Kandemir*, Anand Sivasubramaniam*, and Onur Mutlu†

*The Pennsylvania State University and †Carnegie Mellon University

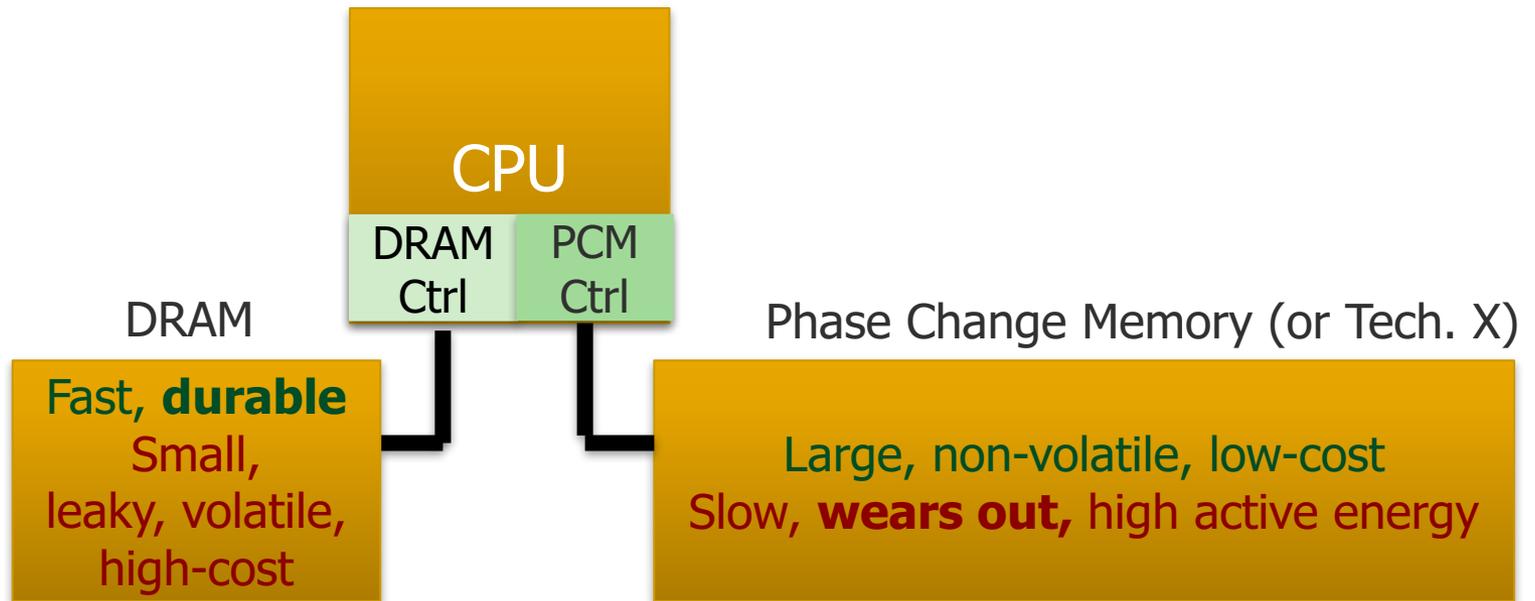
A More Viable Approach: Hybrid Memory Systems



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.

A More Viable Approach: 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.

Providing the Best of
Multiple Metrics

with

Multiple Memory Technologies

Challenge and Opportunity

Heterogeneous,
Configurable,
Programmable
Memory Systems

Hybrid Memory Systems: Issues

- Cache vs. Main Memory
- Granularity of Data Move/Management: Fine or Coarse
- Hardware vs. Software vs. HW/SW Cooperative
- When to migrate data?
- How to design a scalable and efficient large cache?
- ...

On Hybrid Memory Data Placement (I)

- HanBin Yoon, Justin Meza, Rachata Ausavarungnirun, Rachael Harding, and Onur Mutlu,
"Row Buffer Locality Aware Caching Policies for Hybrid Memories"

Proceedings of the 30th IEEE International Conference on Computer Design (ICCD), Montreal, Quebec, Canada, September 2012. Slides (pptx) (pdf)

Row Buffer Locality Aware Caching Policies for Hybrid Memories

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

On Hybrid Memory Data Placement (II)

- Yang Li, Saugata Ghose, Jongmoo Choi, Jin Sun, Hui Wang, and Onur Mutlu,
"Utility-Based Hybrid Memory Management"
*Proceedings of the 19th IEEE Cluster Conference (**CLUSTER**),
Honolulu, Hawaii, USA, September 2017.*
[[Slides \(pptx\)](#) ([pdf](#))]

Utility-Based Hybrid Memory Management

Yang Li[†] Saugata Ghose[†] Jongmoo Choi[‡] Jin Sun[†] Hui Wang^{*} Onur Mutlu^{††}
[†]*Carnegie Mellon University* [‡]*Dankook University* ^{*}*Beihang University* ^{††}*ETH Zürich*

On Large DRAM Cache Design (I)

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

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

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

On Large DRAM Cache Design (II)

- Xiangyao Yu, Christopher J. Hughes, Nadathur Satish, Onur Mutlu, and Srinivas Devadas,
"Banshee: Bandwidth-Efficient DRAM Caching via Software/Hardware Cooperation"
Proceedings of the 50th International Symposium on Microarchitecture (MICRO), Boston, MA, USA, October 2017.

Banshee: Bandwidth-Efficient DRAM Caching via Software/Hardware Cooperation

Xiangyao Yu¹ Christopher J. Hughes² Nadathur Satish² Onur Mutlu³ Srinivas Devadas¹
¹MIT ²Intel Labs ³ETH Zürich

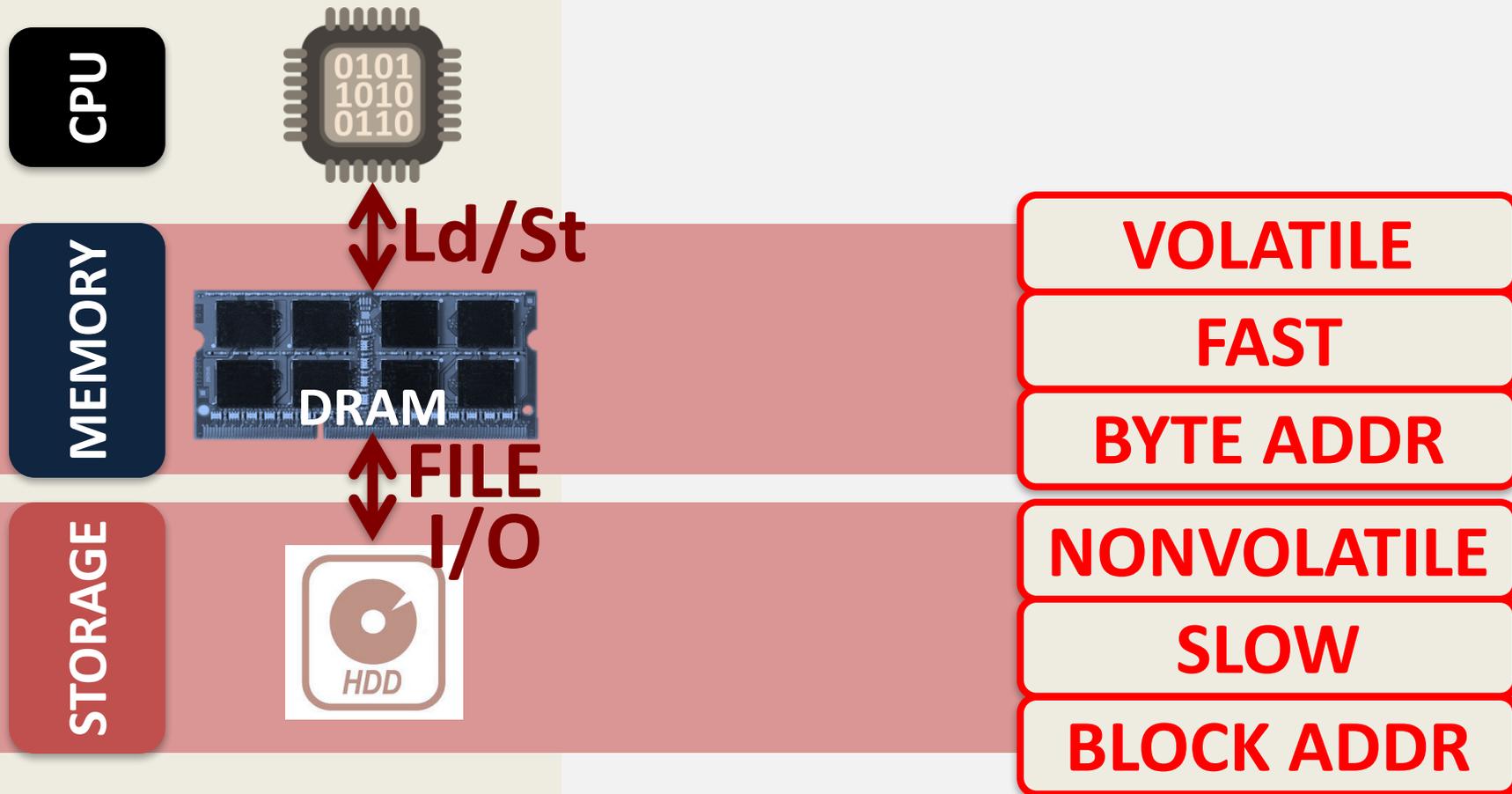
Enabling an Emerging Technology to Augment DRAM

Managing Hybrid Memories

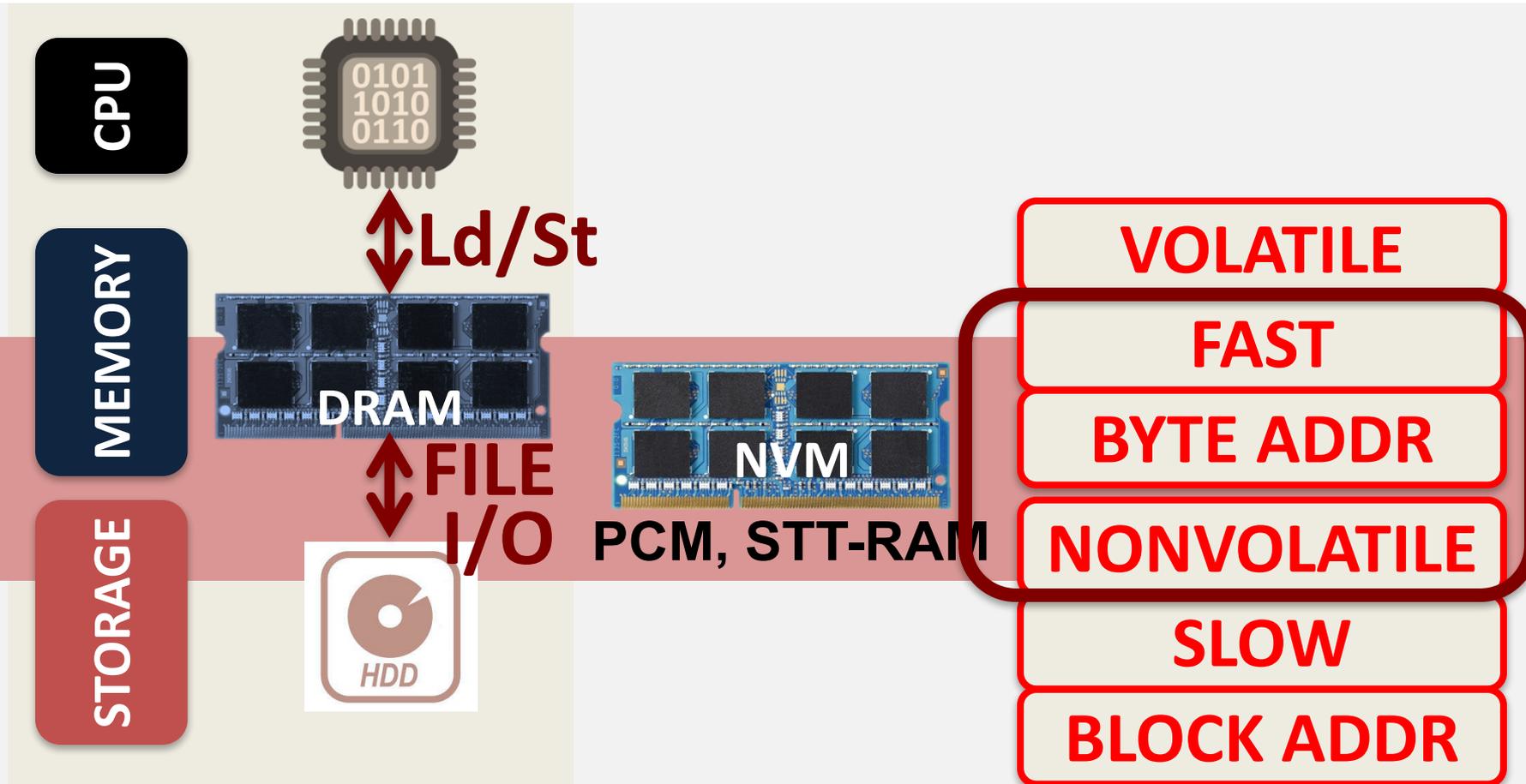
Other Opportunities with Emerging Technologies

- **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

TWO-LEVEL STORAGE MODEL



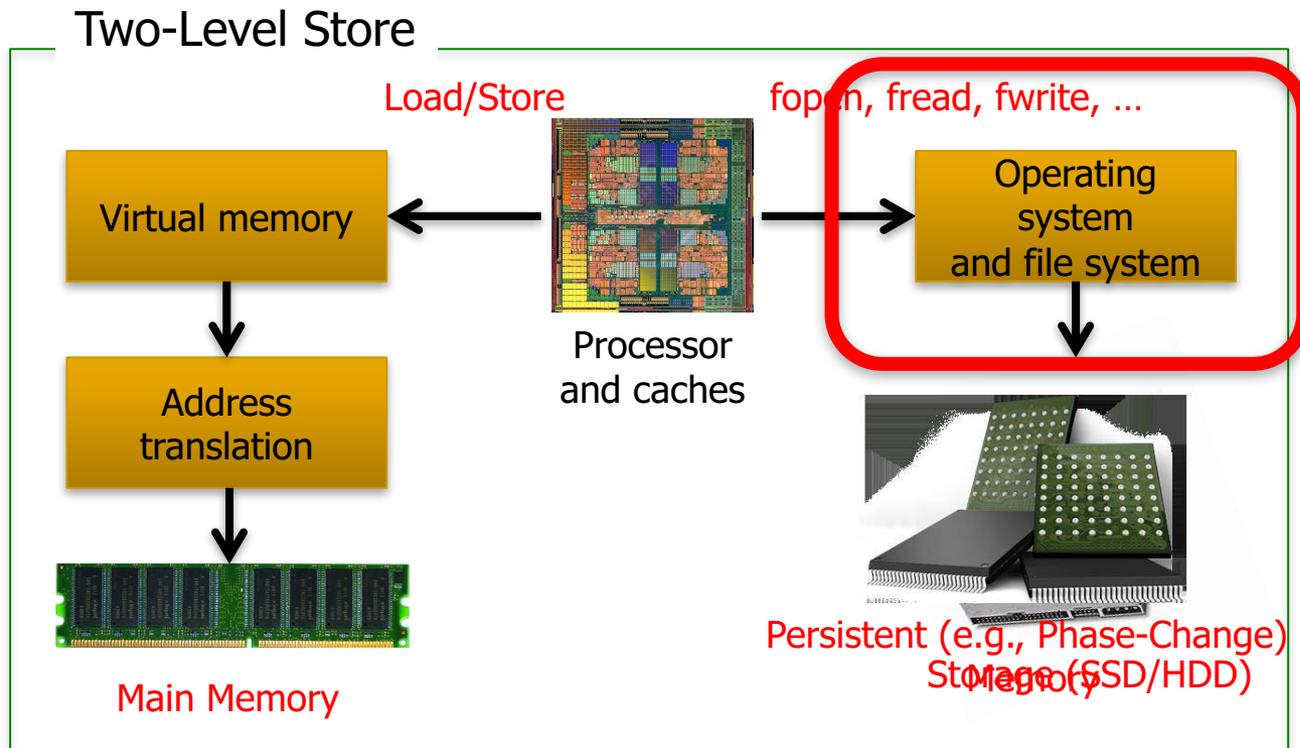
TWO-LEVEL STORAGE MODEL



Non-volatile memories combine characteristics of memory and storage

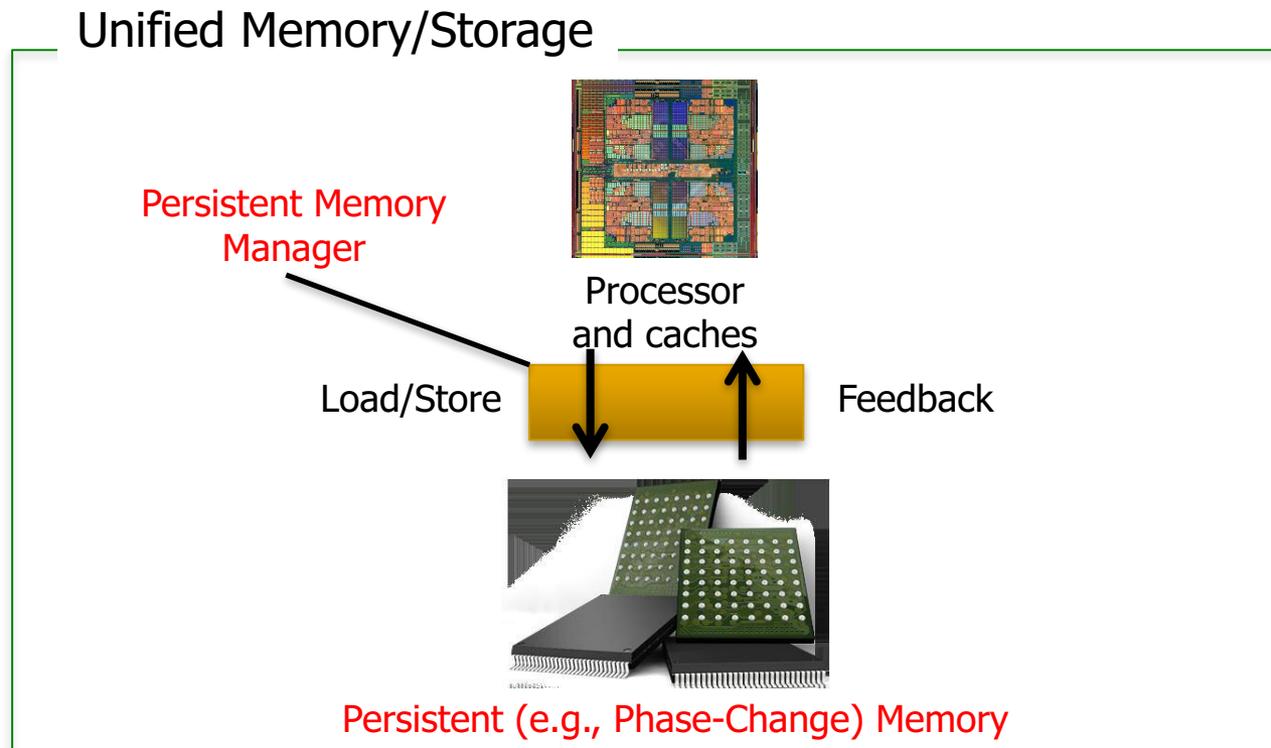
Two-Level Memory/Storage Model

- 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



Unified Memory and Storage with NVM

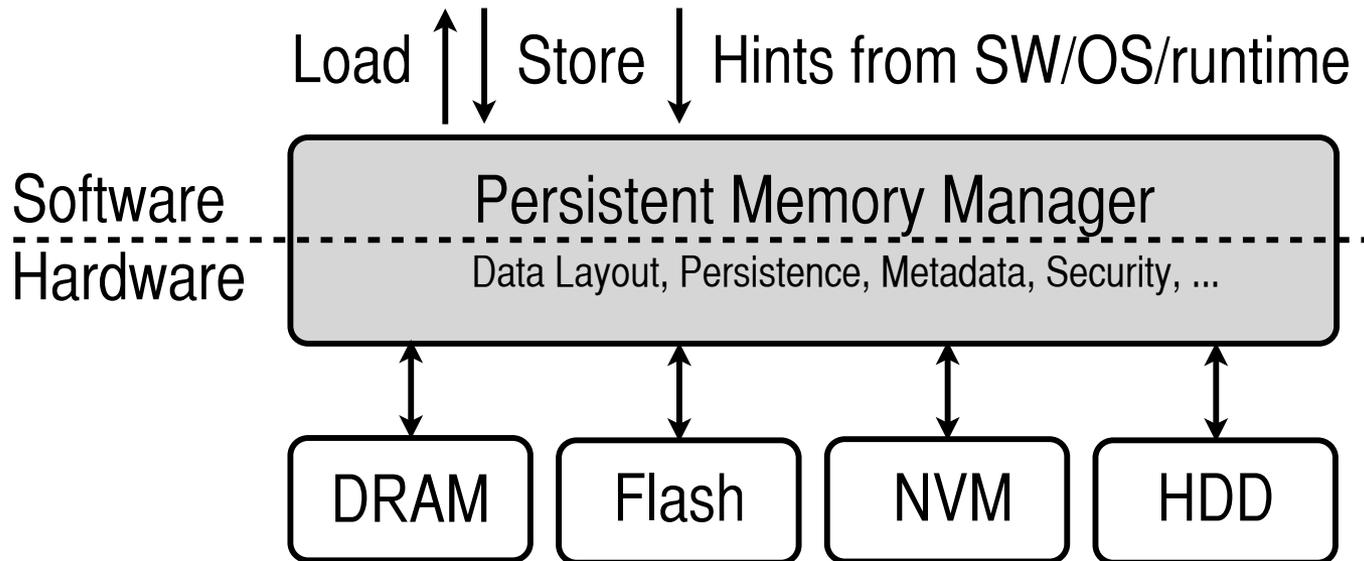
- Goal: Unify memory and storage management in a single unit to eliminate wasted work to locate, transfer, and translate data
 - Improves both energy and performance
 - Simplifies programming model as well



The Persistent Memory Manager (PMM)

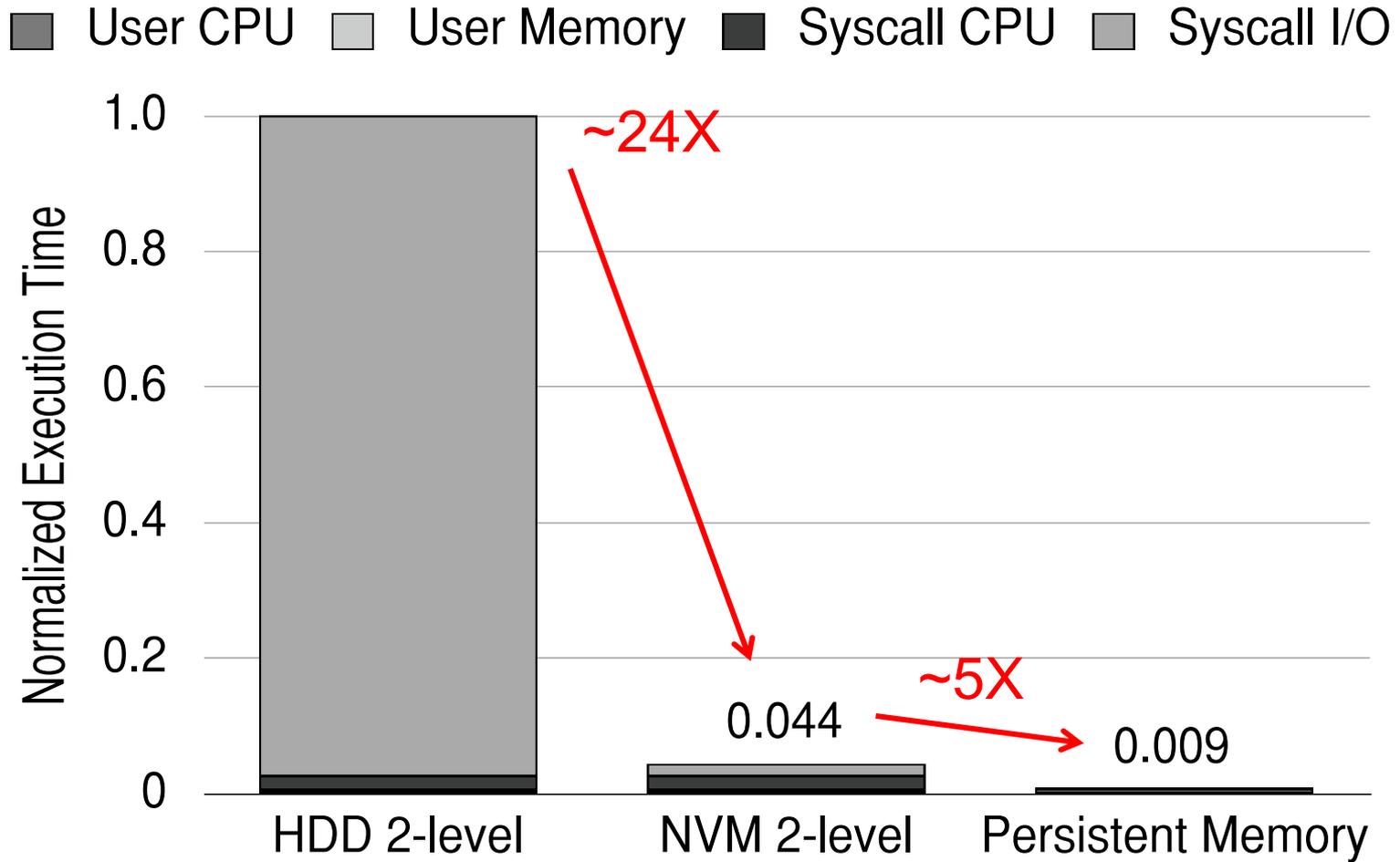
```
1 int main(void) {  
2     // data in file.dat is persistent  
3     FILE myData = "file.dat";  
4     myData = new int[64];  
5 }  
6 void updateValue(int n, int value) {  
7     FILE myData = "file.dat";  
8     myData[n] = value; // value is persistent  
9 }
```

Persistent objects

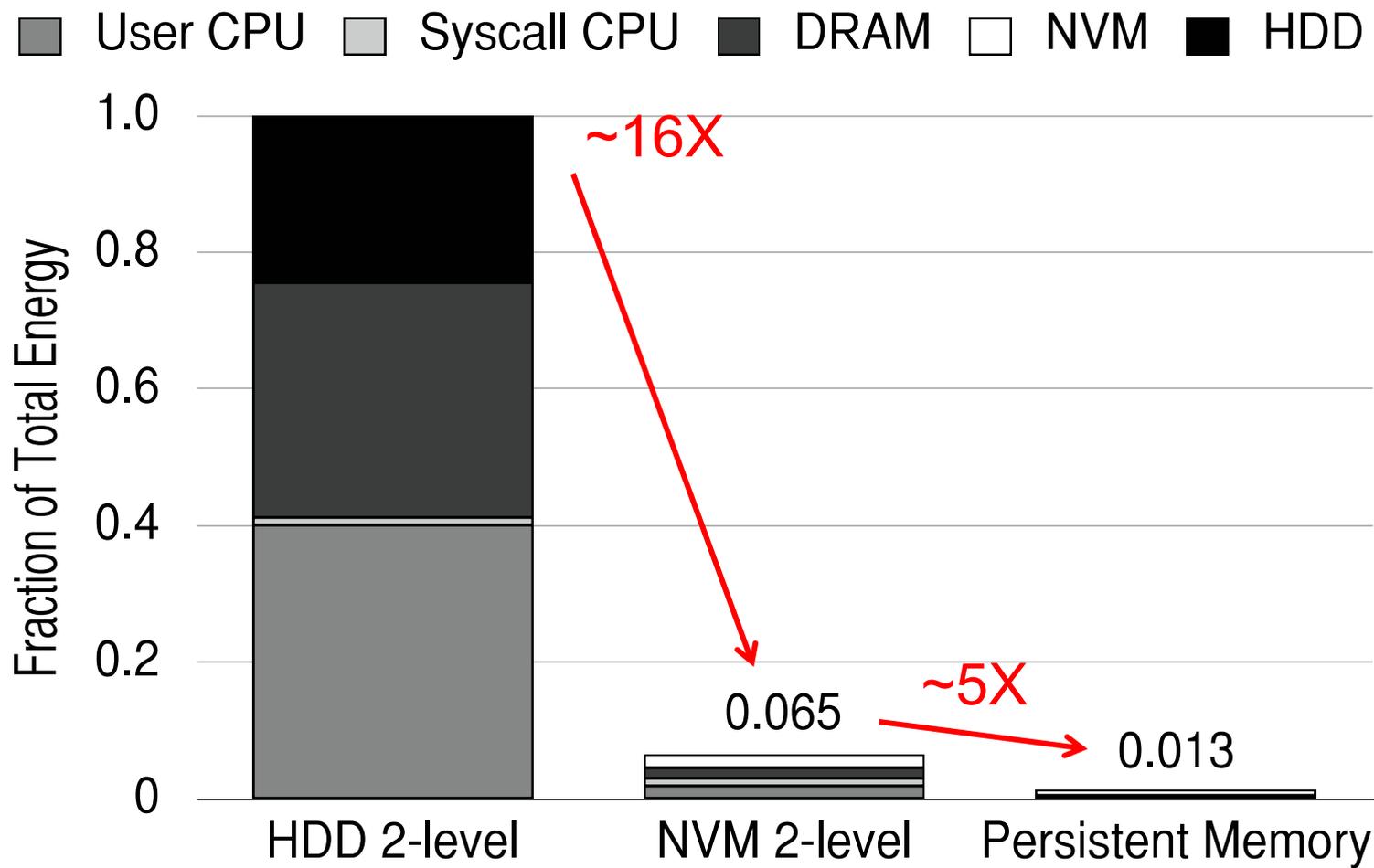


PMM uses access and hint information to allocate, locate, migrate and access data in the heterogeneous array of devices

Performance Benefits of a Single-Level Store



Energy Benefits of a Single-Level Store



On Persistent Memory Benefits & Challenges

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

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

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

Combined Memory & Storage

A Unified Interface to **All Data**

Programming Ease to Exploit Persistence

Hardware Support for Crash Consistency

- Jinglei Ren, Jishen Zhao, Samira Khan, Jongmoo Choi, Yongwei Wu, and Onur Mutlu,
"ThyNVM: Enabling Software-Transparent Crash Consistency in Persistent Memory Systems"
Proceedings of the 48th International Symposium on Microarchitecture (MICRO), Waikiki, Hawaii, USA, December 2015.
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
[Source Code]

ThyNVM: Enabling Software-Transparent Crash Consistency in Persistent Memory Systems

Jinglei Ren^{*†} Jishen Zhao[‡] Samira Khan^{†'} Jongmoo Choi^{+†} Yongwei Wu^{*} Onur Mutlu[†]

[†]Carnegie Mellon University ^{*}Tsinghua University

[‡]University of California, Santa Cruz [']University of Virginia ⁺Dankook University

Tools/Libraries to Help Programmers

- Himanshu Chauhan, Irina Calciu, Vijay Chidambaram, Eric Schkufza, Onur Mutlu, and Pratap Subrahmanyam, **"NVMove: Helping Programmers Move to Byte-Based Persistence"**

Proceedings of the 4th Workshop on Interactions of NVM/Flash with Operating Systems and Workloads (INFLOW), Savannah, GA, USA, November 2016.

[[Slides \(pptx\)](#) ([pdf](#))]

NVMOVE: Helping Programmers Move to Byte-Based Persistence

Himanshu Chauhan *
UT Austin

Irina Calciu
VMware Research Group

Vijay Chidambaram
UT Austin

Eric Schkufza
VMware Research Group

Onur Mutlu
ETH Zürich

Pratap Subrahmanyam
VMware

Data Structures for In-Memory Processing

- Zhiyu Liu, Irina Calciu, Maurice Herlihy, and Onur Mutlu,
"Concurrent Data Structures for Near-Memory Computing"
Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), Washington, DC, USA, July 2017.
[\[Slides \(pptx\) \(pdf\)\]](#)

Concurrent Data Structures for Near-Memory Computing

Zhiyu Liu

Computer Science Department
Brown University
zhiyu.liu@brown.edu

Irina Calciu

VMware Research Group
icalciu@vmware.com

Maurice Herlihy

Computer Science Department
Brown University
mph@cs.brown.edu

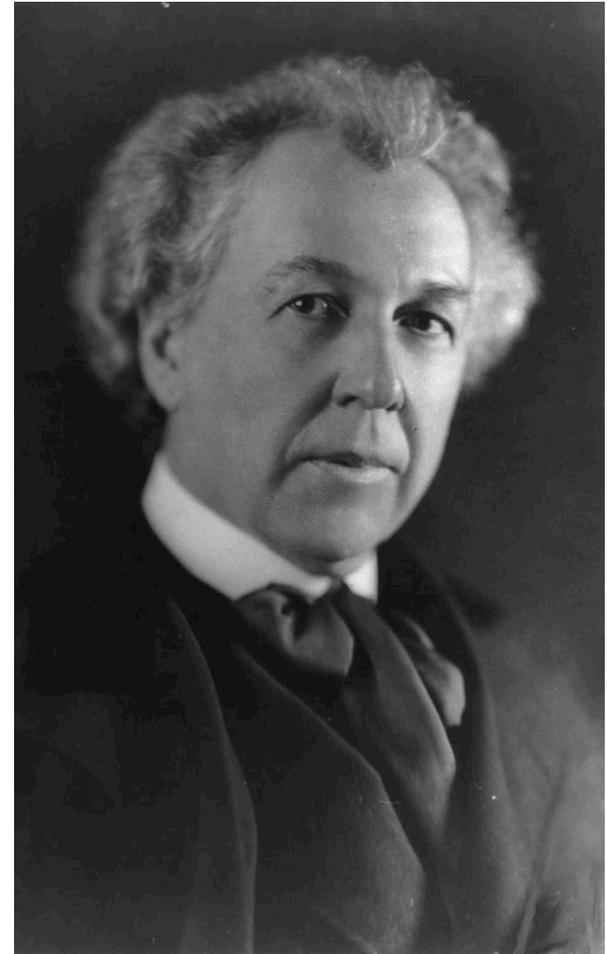
Onur Mutlu

Computer Science Department
ETH Zürich
onur.mutlu@inf.ethz.ch

Concluding Remarks

A Quote from A Famous Architect

- “architecture [...] based upon **principle**, and not upon **precedent**”



Precedent-Based Design?

- “architecture [...] based upon **principle**, and not upon **precedent**”



Principled Design

- “architecture [...] based upon **principle**, and not upon **precedent**”





The Overarching Principle

Organic architecture

From Wikipedia, the free encyclopedia

Organic architecture is a [philosophy](#) of [architecture](#) which promotes harmony between human habitation and the natural world through design approaches so sympathetic and well integrated with its site, that buildings, furnishings, and surroundings become part of a unified, interrelated composition.

A well-known example of organic architecture is [Fallingwater](#), the residence Frank Lloyd Wright designed for the Kaufmann family in rural Pennsylvania. Wright had many choices to locate a home on this large site, but chose to place the home directly over the waterfall and creek creating a close, yet noisy dialog with the rushing water and the steep site. The horizontal striations of stone masonry with daring [cantilevers](#) of colored beige concrete blend with native rock outcroppings and the wooded environment.

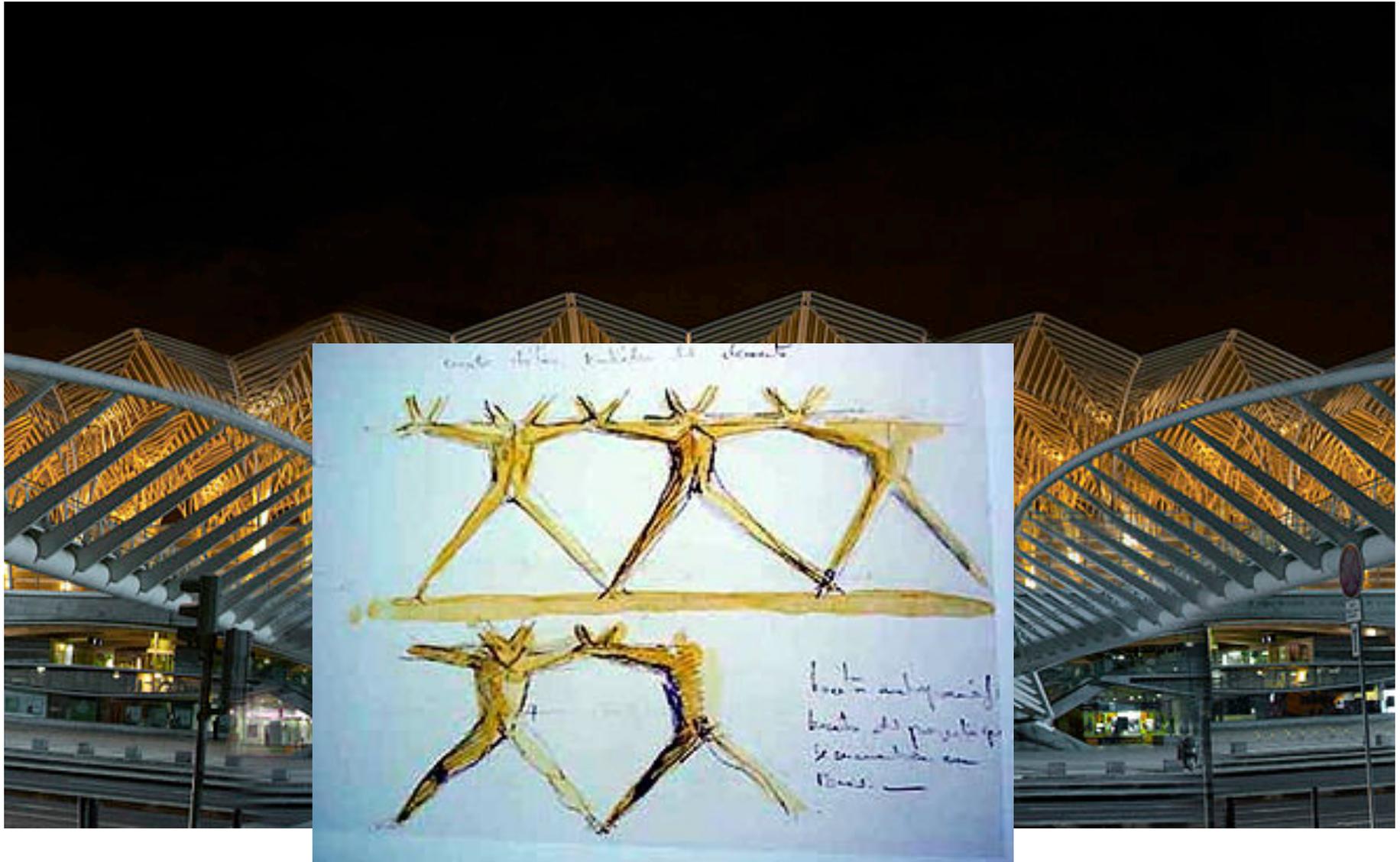
Another Example: Precedent-Based Design



Principled Design



Another Principled Design



Source: By Martín Gómez Tagle - Lisbon, Portugal, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=13764903>

Source: <http://www.arcspace.com/exhibitions/unsorted/santiago-calatrava/>

Principle Applied to Another Structure



Source: By 準建築人手札網站 Forgemind ArchiMedia - Flickr: IMG_2489.JPG, CC BY 2.0,

Source: <https://www.dezeen.com/2016/08/29/santiago-calatrava-oculus-world-trade-center-transportation-hub-new-york-photographs-hufton-crow/>

The Overarching Principle

Zoomorphic architecture

From Wikipedia, the free encyclopedia

Zoomorphic architecture is the practice of using animal forms as the inspirational basis and blueprint for architectural design. "While animal forms have always played a role adding some of the deepest layers of meaning in architecture, it is now becoming evident that a new strand of **biomorphism** is emerging where the meaning derives not from any specific representation but from a more general allusion to biological processes."^[1]

Some well-known examples of Zoomorphic architecture can be found in the [TWA Flight Center](#) building in [New York City](#), by [Eero Saarinen](#), or the [Milwaukee Art Museum](#) by [Santiago Calatrava](#), both inspired by the form of a bird's wings.^[3]

Overarching Principles for Computing?



Concluding Remarks

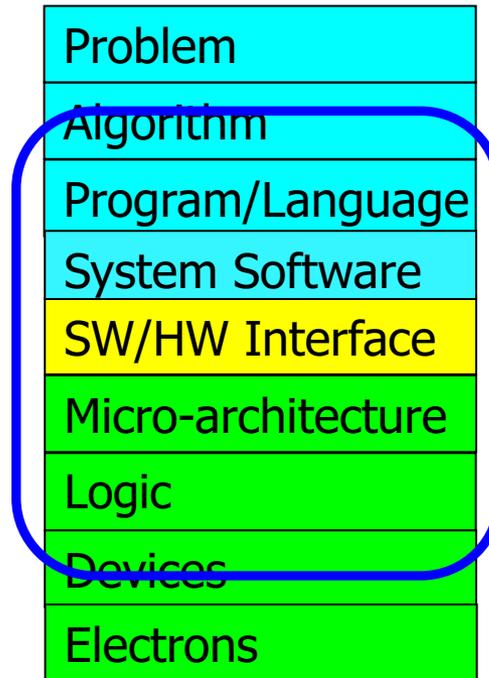
- It is time to design **principled system architectures** to solve the **memory scaling** problem
- **Discover design principles** for fundamentally secure and reliable computer architectures
- **Design complete systems** to be balanced and energy-efficient, i.e., **data-centric** (or **memory-centric**) and **low latency**
- **Enable new and emerging memory architectures**
- **This can**
 - Lead to **orders-of-magnitude** improvements
 - **Enable new applications & computing platforms**
 - ...

The Future of New Memory is Bright

- Regardless of challenges
 - in underlying technology and overlying problems/requirements

Can enable:

- Orders of magnitude improvements
- New applications and computing systems



Yet, we have to

- Think across the stack
- Design enabling systems

If In Doubt, See Other Doubtful Technologies

- A very “doubtful” emerging technology
 - for at least two decades



Proceedings of the IEEE, Sept. 2017

Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD's reliability and lifetime.

By YU CAI, SAUGATA GHOSE, ERICH F. HARATSCH, YIXIN LUO, AND ONUR MUTLU

Rethinking Memory System Design (and the Platforms We Design Around It)

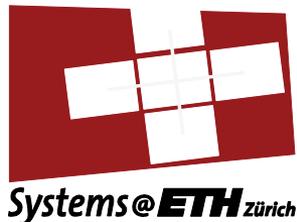
Onur Mutlu

omutlu@gmail.com

<https://people.inf.ethz.ch/omutlu>

December 4, 2017

INESC-ID Distinguished Lecture (Lisbon)



SAFARI

ETH zürich

Open Problems

For More Open Problems, See (I)

- Onur Mutlu and Lavanya Subramanian,
"Research Problems and Opportunities in Memory Systems"
Invited Article in Supercomputing Frontiers and Innovations (SUPERFRI), 2014/2015.

Research Problems and Opportunities in Memory Systems

Onur Mutlu¹, Lavanya Subramanian¹

For More Open Problems, See (II)

- Onur Mutlu,
"The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"
Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (DATE), Lausanne, Switzerland, March 2017.
[[Slides \(pptx\)](#) ([pdf](#))]

The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
<https://people.inf.ethz.ch/omutlu>

For More Open Problems, See (III)

- Onur Mutlu,

"Memory Scaling: A Systems Architecture Perspective"

*Technical talk at MemCon 2013 (MEMCON), Santa Clara, CA, August 2013. [Slides (pptx)] [pdf]
[Video] [Coverage on StorageSearch]*

Memory Scaling: A Systems Architecture Perspective

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu
<http://users.ece.cmu.edu/~omutlu/>

For More Open Problems, See (IV)

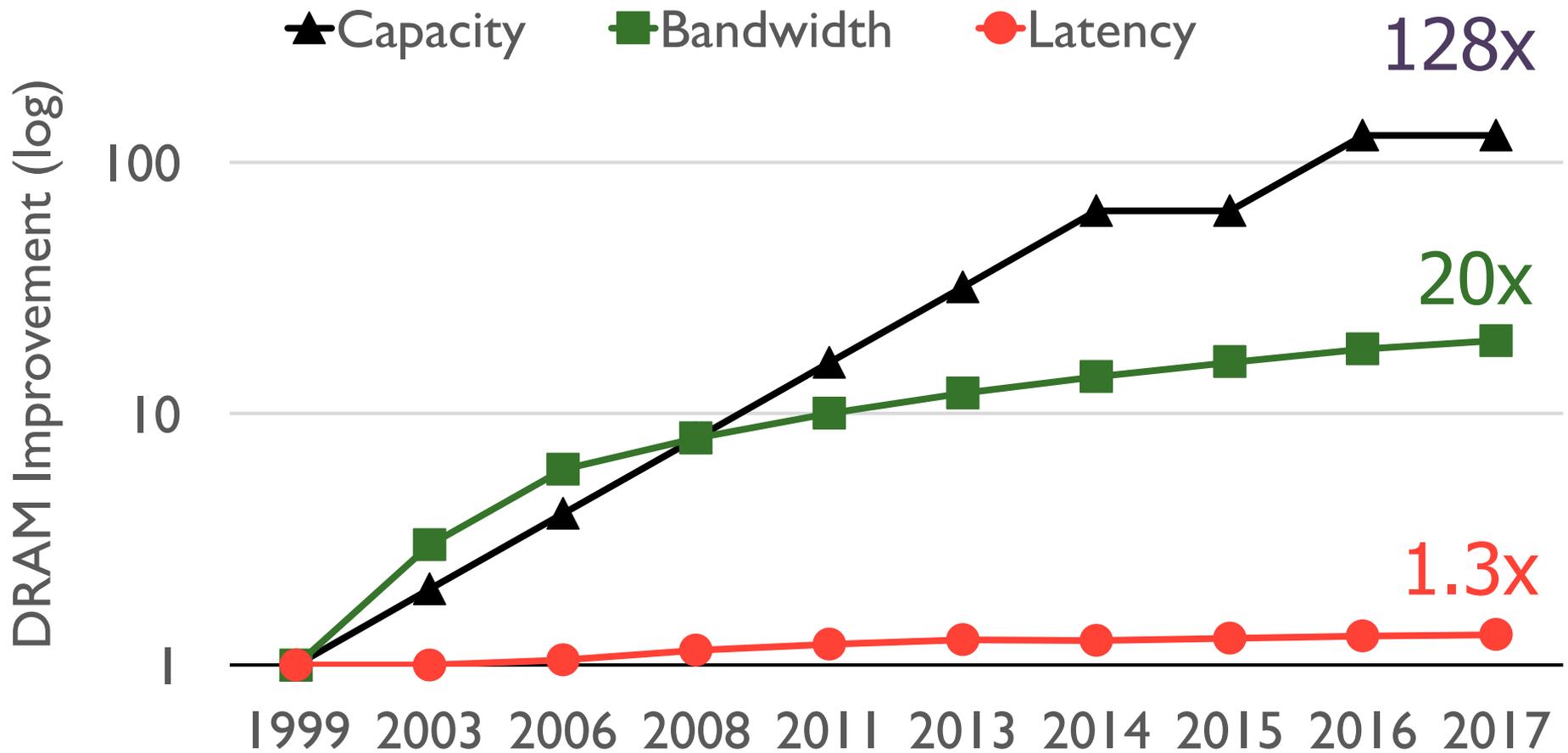
- Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu,
"Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives"
to appear in Proceedings of the IEEE, 2017.
[Preliminary arxiv.org version]

Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives

Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu

Reducing Memory Latency

Main Memory Latency Lags Behind



Memory latency remains almost constant

A Closer Look ...

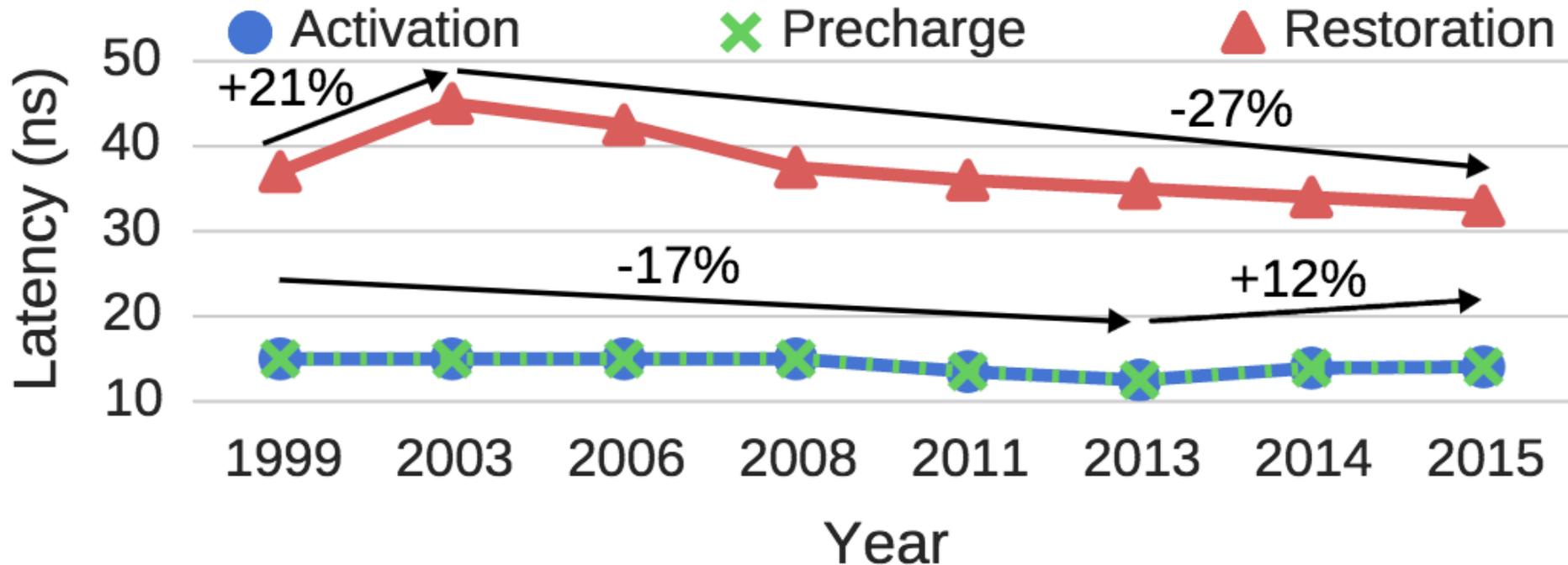
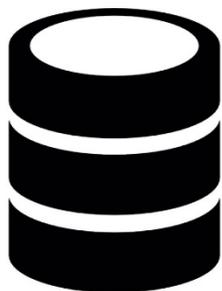


Figure 1: DRAM latency trends over time [20, 21, 23, 51].

Chang+, "[Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization](#)," SIGMETRICS 2016.

DRAM Latency Is Critical for Performance



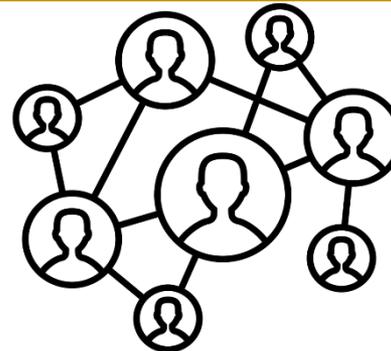
In-memory Databases

[Mao+, EuroSys'12;
Clapp+ (Intel), IISWC'15]



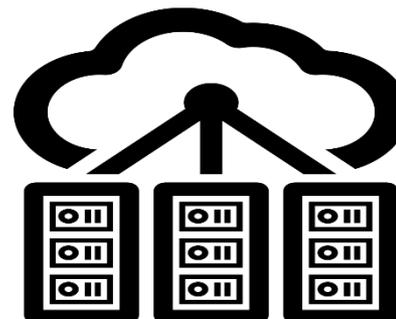
In-Memory Data Analytics

[Clapp+ (Intel), IISWC'15;
Awan+, BDCloud'15]



Graph/Tree Processing

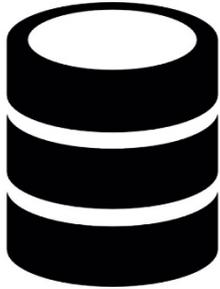
[Xu+, IISWC'12; Umuroglu+, FPL'15]



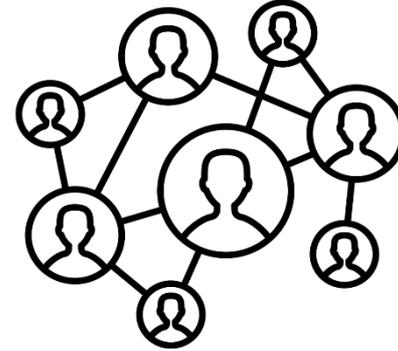
Datacenter Workloads

[Kanev+ (Google), ISCA'15]

DRAM Latency Is Critical for Performance



In-memory Databases



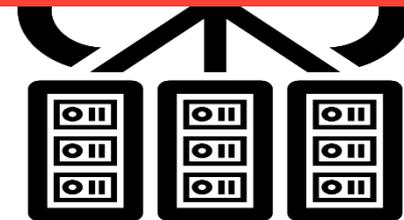
Graph/Tree Processing

Long memory latency → performance bottleneck



In-Memory Data Analytics

[Clapp+ (Intel), IISWC'15;
Awan+, BDCloud'15]



Datacenter Workloads

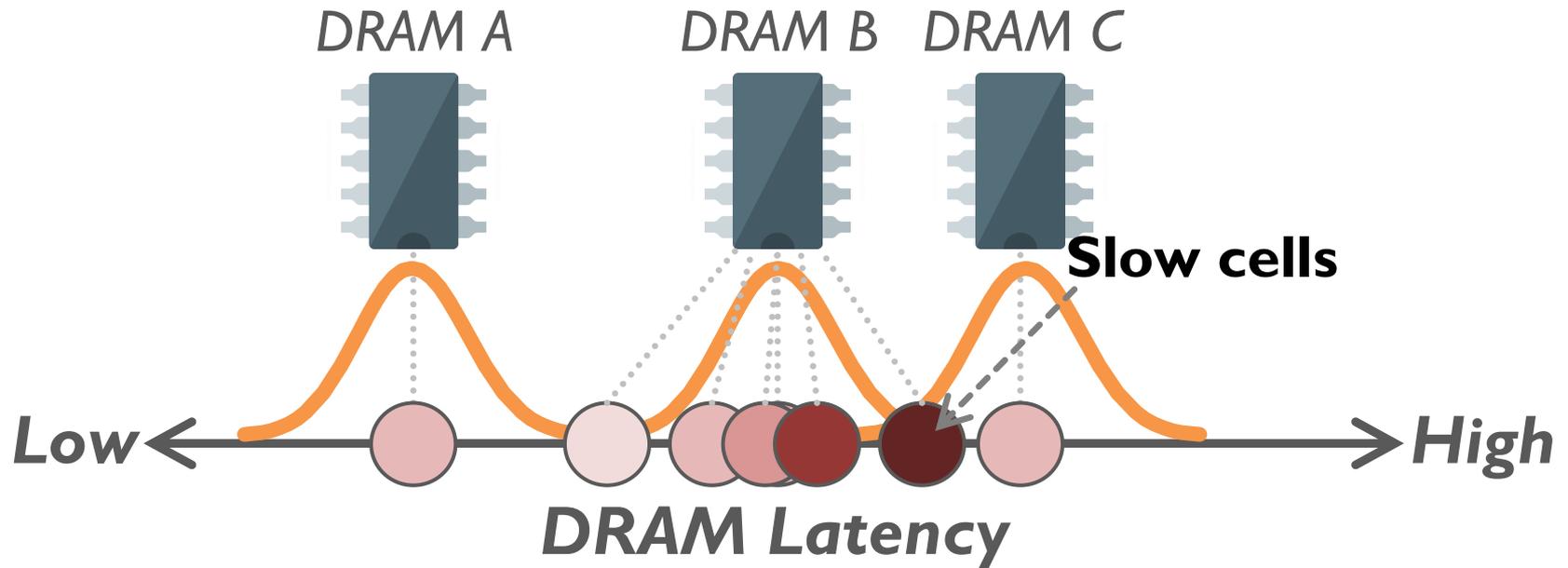
[Kanev+ (Google), ISCA'15]

Why the Long Latency?

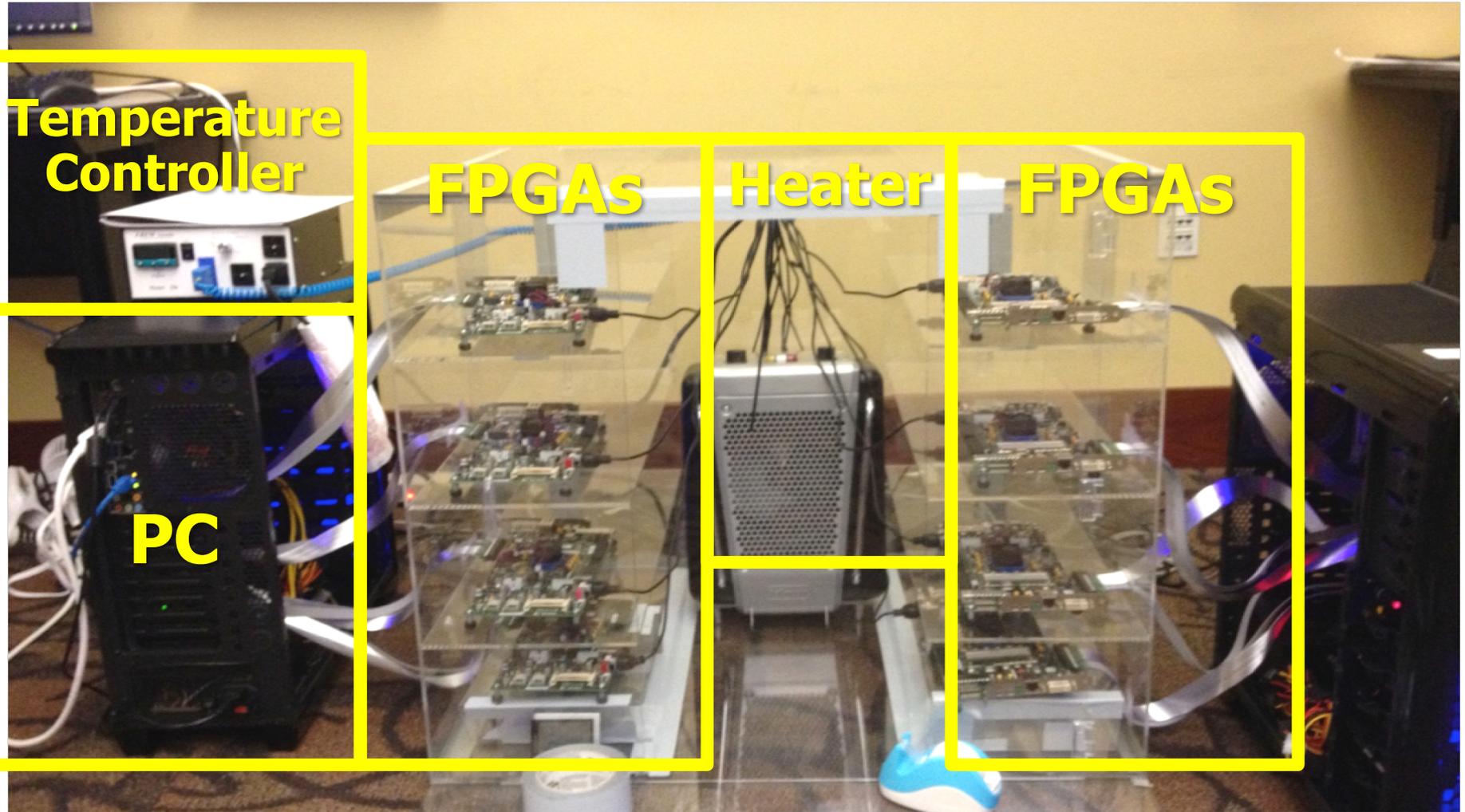
- Design of DRAM uArchitecture
 - Goal: Maximize capacity/area, not minimize latency
- “One size fits all” approach to latency specification
 - Same latency parameters for all temperatures
 - Same latency parameters for all DRAM chips (e.g., rows)
 - Same latency parameters for all parts of a DRAM chip
 - Same latency parameters for all supply voltage levels
 - Same latency parameters for all application data
 - ...

Latency Variation in Memory Chips

Heterogeneous manufacturing & operating conditions →
latency variation in timing parameters



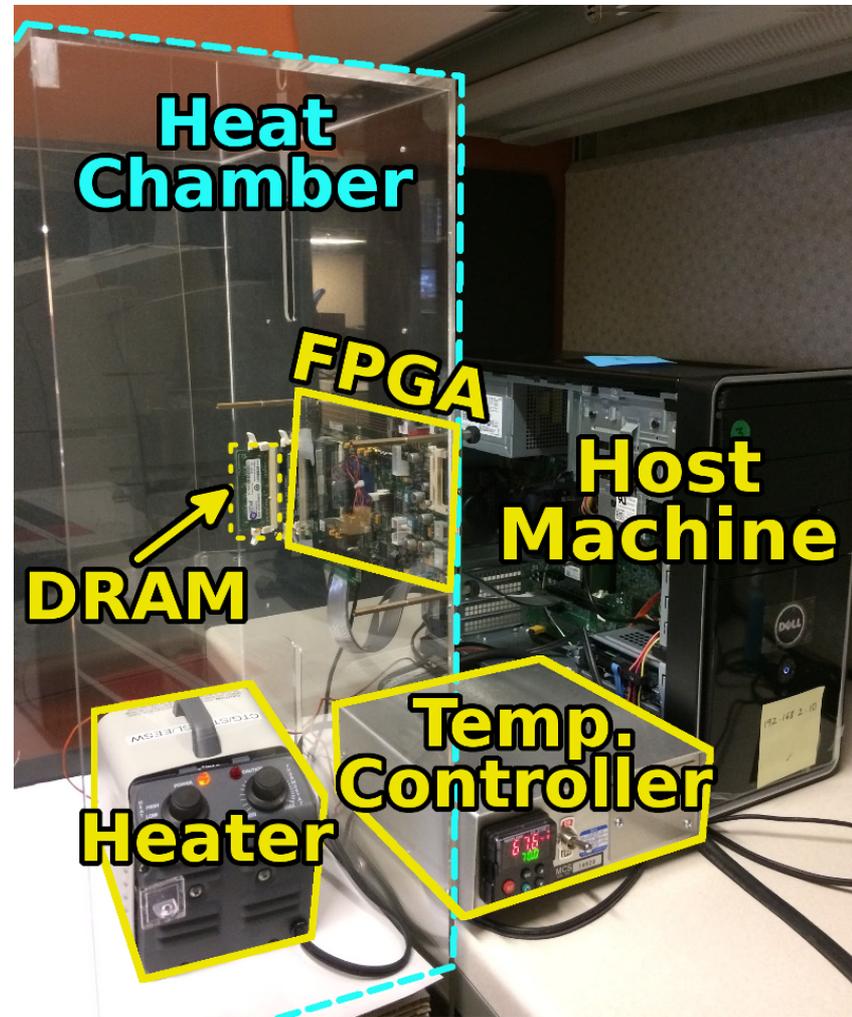
DRAM Characterization Infrastructure



DRAM Characterization Infrastructure

- Hasan Hassan et al., **SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies**, HPCA 2017.

- Flexible
- Easy to Use (C++ API)
- Open-source
github.com/CMU-SAFARI/SoftMC



SoftMC: Open Source DRAM Infrastructure

- <https://github.com/CMU-SAFARI/SoftMC>

SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan^{1,2,3} Nandita Vijaykumar³ Samira Khan^{4,3} Saugata Ghose³ Kevin Chang³
Gennady Pekhimenko^{5,3} Donghyuk Lee^{6,3} Oguz Ergin² Onur Mutlu^{1,3}

¹*ETH Zürich* ²*TOBB University of Economics & Technology* ³*Carnegie Mellon University*
⁴*University of Virginia* ⁵*Microsoft Research* ⁶*NVIDIA Research*

Tackling the Fixed Latency Mindset

- Reliable operation latency is actually very heterogeneous
 - Across temperatures, chips, parts of a chip, voltage levels, ...
- Idea: Dynamically find out and use the lowest latency one can reliably access a memory location with
 - Adaptive-Latency DRAM [HPCA 2015]
 - Flexible-Latency DRAM [SIGMETRICS 2016]
 - Design-Induced Variation-Aware DRAM [SIGMETRICS 2017]
 - Voltron [SIGMETRICS 2017]
 - ...
- We would like to find sources of latency heterogeneity and exploit them to minimize latency

Adaptive-Latency 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

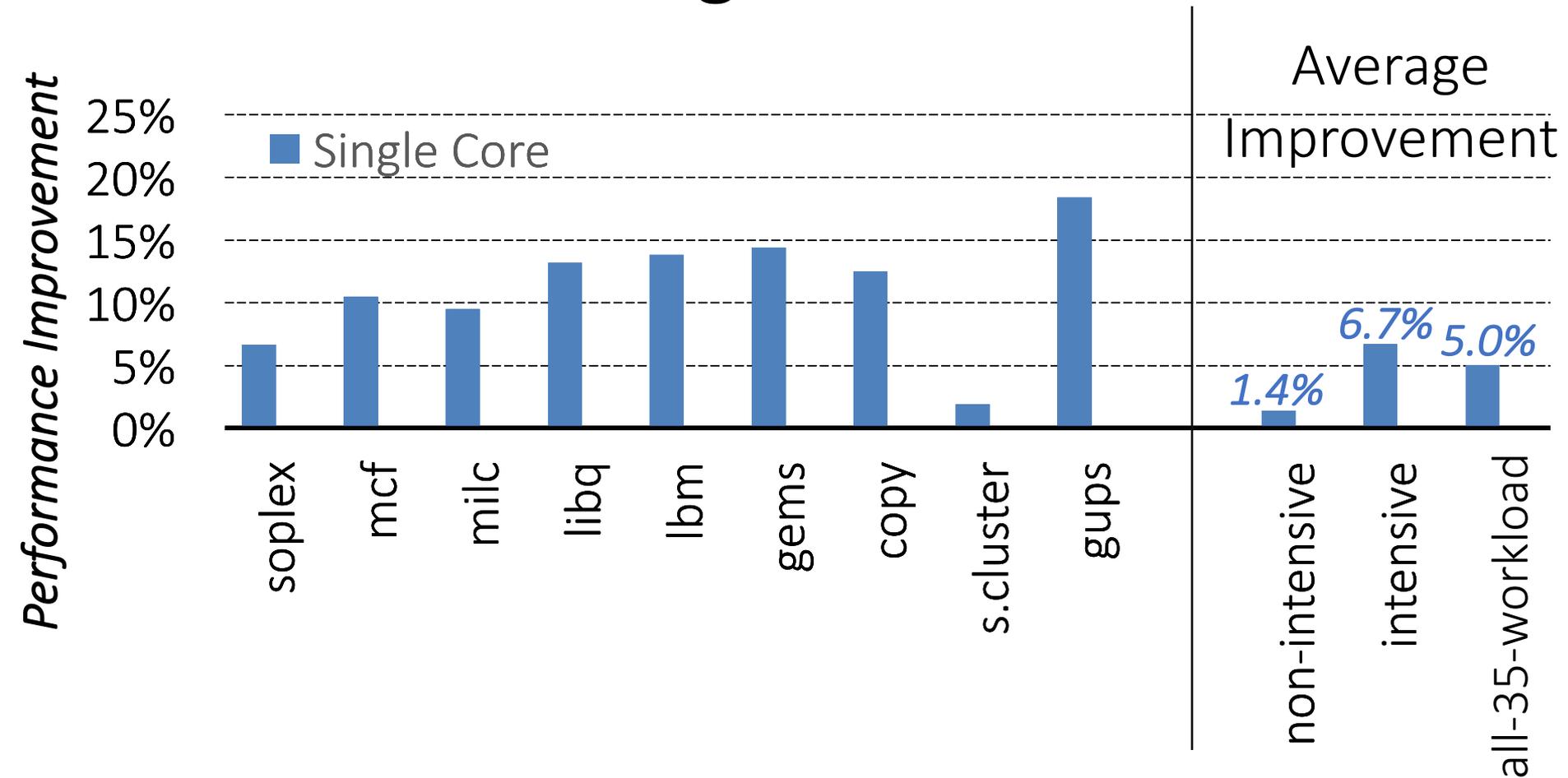
- *System*
 - CPU: AMD 4386 (8 Cores, 3.1GHz, 8MB LLC)

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

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

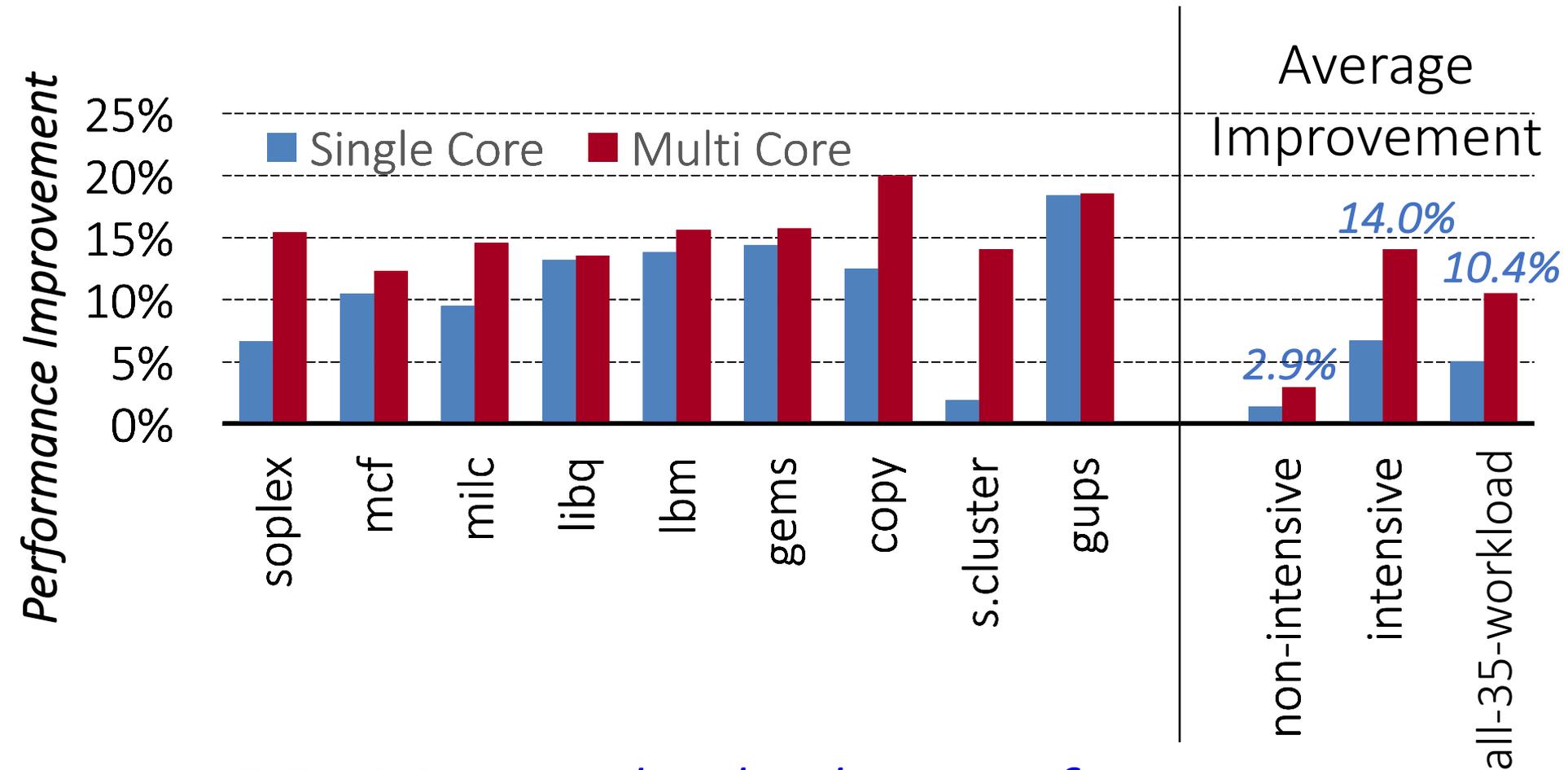
Bits	Description								
31:30	Reserved.								
29:24	Tras: row active strobe. Read-write. BIOS: See 2.9.7.5 [SPD ROM-Based Configuration]. Specifies the minimum time in memory clock cycles from an activate command to a precharge command, both to the same chip select bank. <table border="1"><thead><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><Tras> 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	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 single-core performance on a real system

AL-DRAM: Multi-Core Evaluation



AL-DRAM provides higher performance on multi-programmed & multi-threaded workloads

Reducing Latency Also Reduces Energy

- AL-DRAM reduces DRAM power consumption by 5.8%
- Major reason: reduction in row activation time

More on Adaptive-Latency DRAM

- 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.

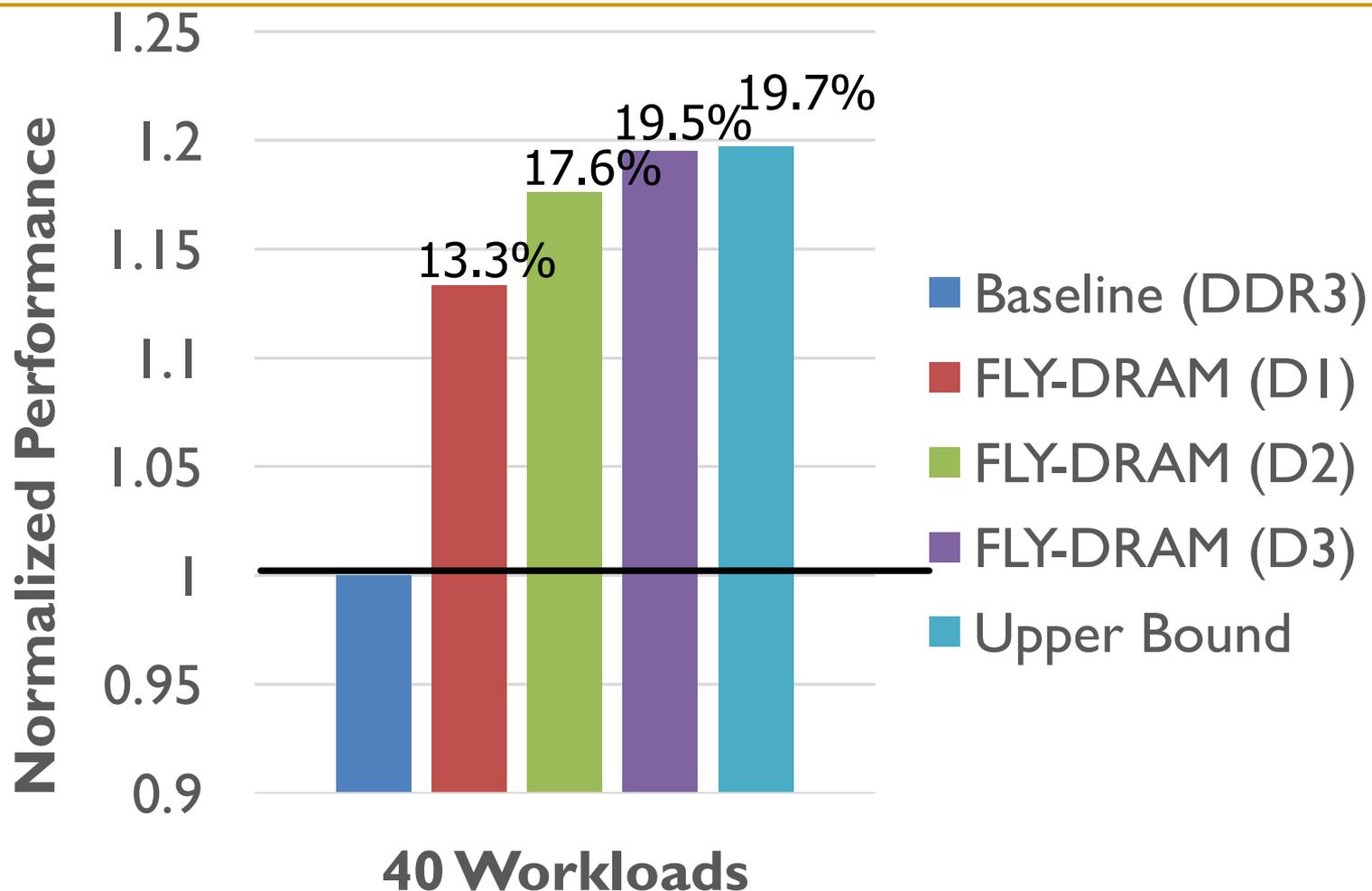
[[Slides \(pptx\)](#)] [[pdf](#)] [[Full data sets](#)]

Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case

Donghyuk Lee Yoongu Kim Gennady Pekhimenko
Samira Khan Vivek Seshadri Kevin Chang Onur Mutlu

Carnegie Mellon University

Heterogeneous Latency within A Chip



Chang+, "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization", SIGMETRICS 2016.

Analysis of Latency Variation in DRAM Chips

- Kevin Chang, Abhijith Kashyap, Hasan Hassan, Samira Khan, Kevin Hsieh, Donghyuk Lee, Saugata Ghose, Gennady Pekhimenko, Tianshi Li, and Onur Mutlu,

"Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization"

*Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**), Antibes Juan-Les-Pins, France, June 2016.*

[[Slides \(pptx\)](#) ([pdf](#))]

[[Source Code](#)]

Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization

Kevin K. Chang¹

Abhijith Kashyap¹

Hasan Hassan^{1,2}

Saugata Ghose¹

Kevin Hsieh¹

Donghyuk Lee¹

Tianshi Li^{1,3}

Gennady Pekhimenko¹

Samira Khan⁴

Onur Mutlu^{5,1}

¹Carnegie Mellon University

²TOBB ETÜ

³Peking University

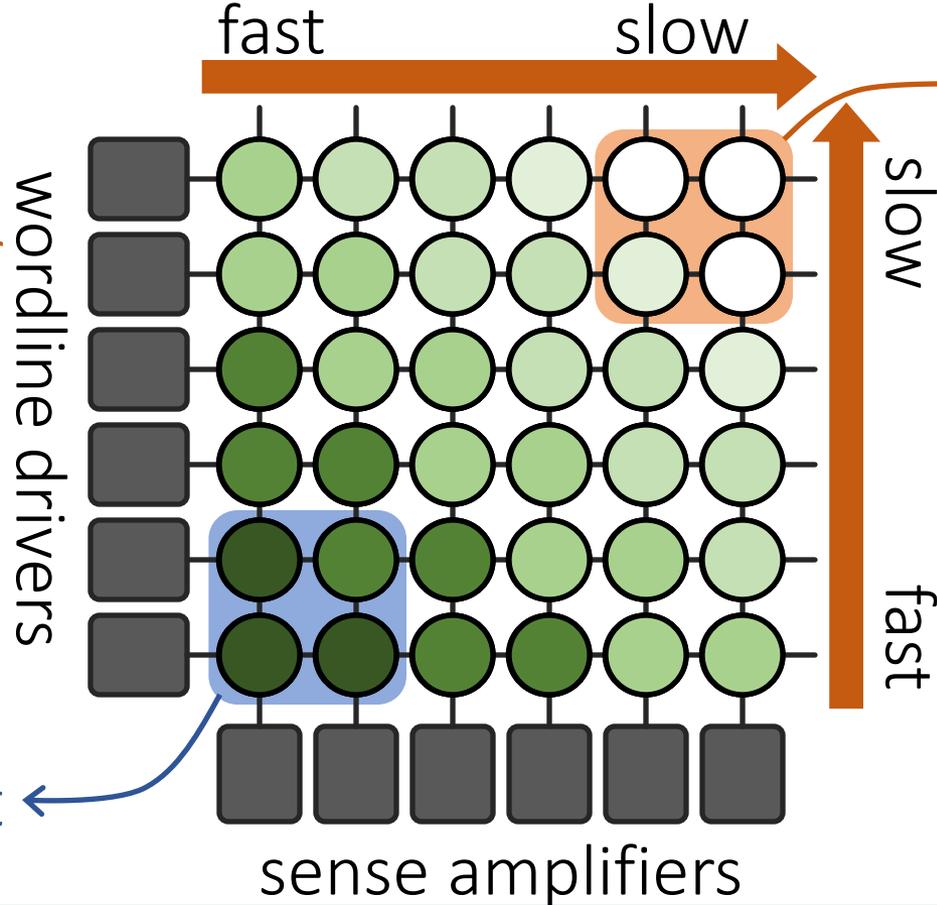
⁴University of Virginia

⁵ETH Zürich

What Is Design-Induced Variation?

across column

distance from
wordline driver



inherently slow

across row

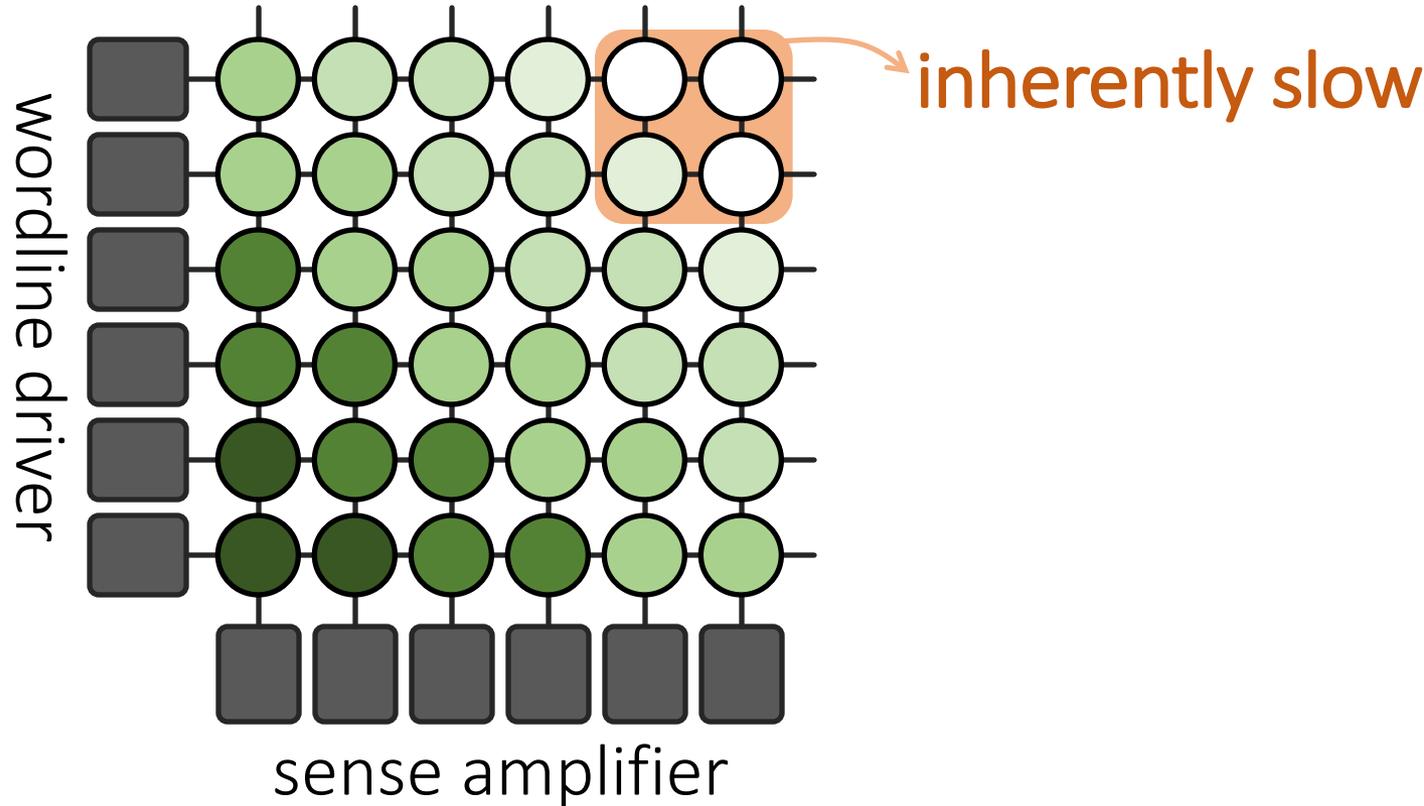
distance from
sense amplifier

Inherently fast

Systematic variation in cell access times
caused by the ***physical organization*** of DRAM

DIVA Online Profiling

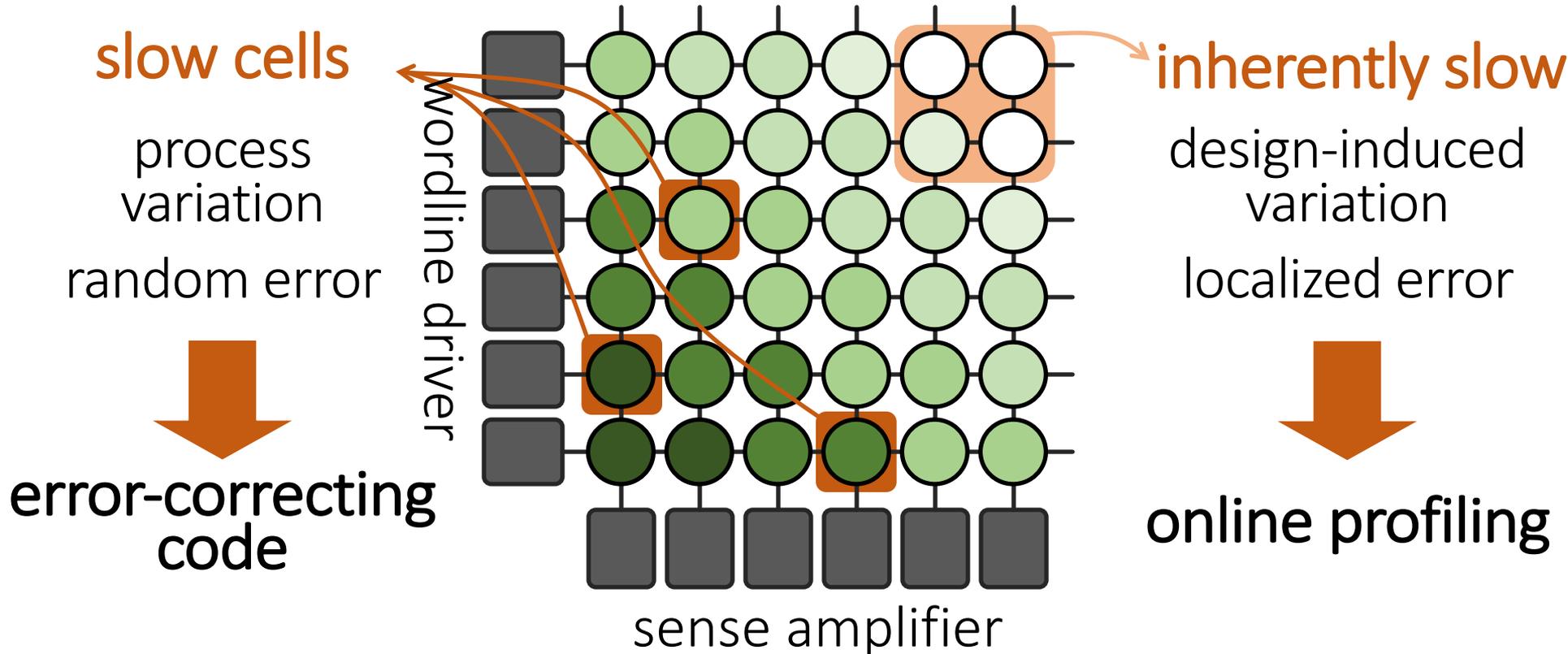
Design-Induced-Variation-Aware



Profile *only slow regions* to determine min. latency
→ *Dynamic* & *low cost* latency optimization

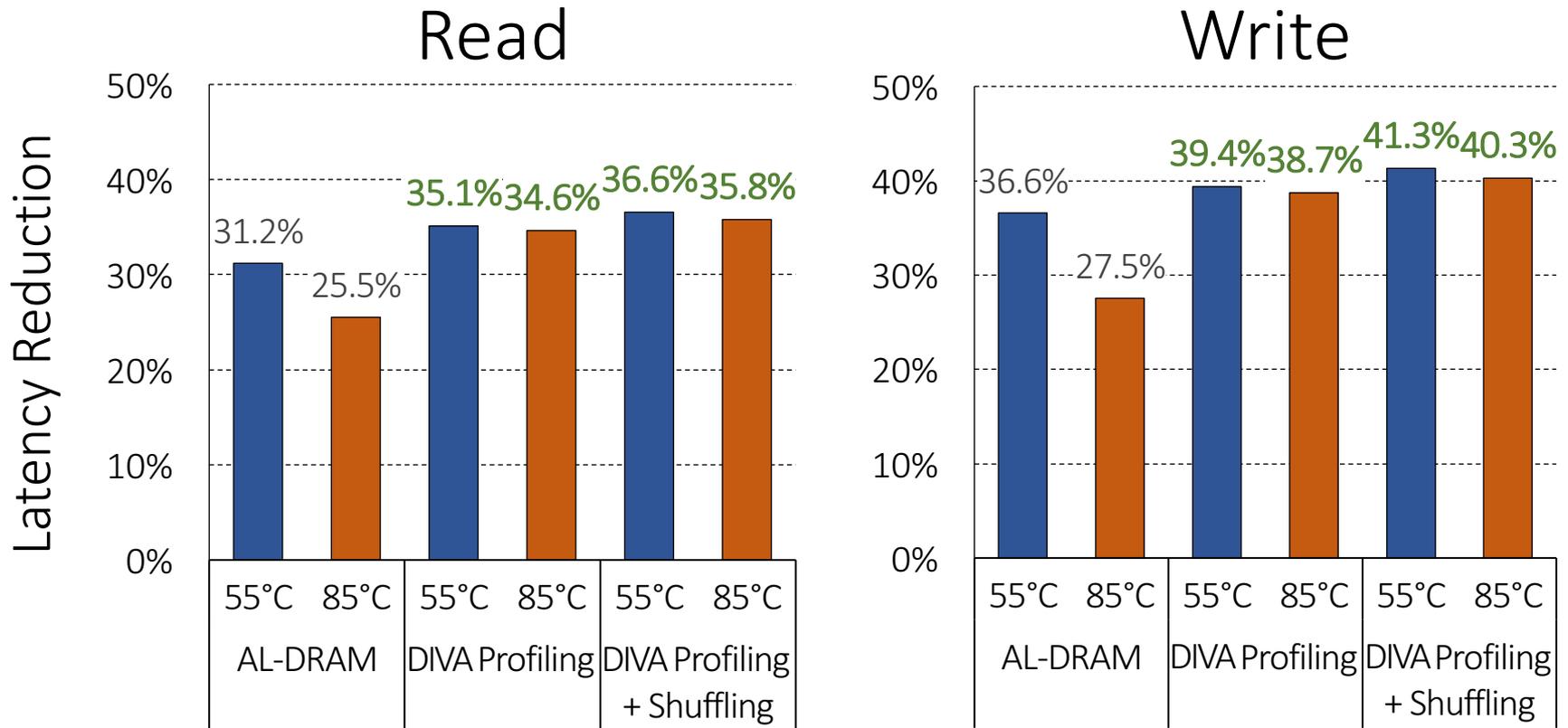
DIVA Online Profiling

Design-Induced-Variation-Aware



Combine **error-correcting codes** & **online profiling**
→ **Reliably** reduce DRAM latency

DIVA-DRAM Reduces Latency



DIVA-DRAM *reduces latency more aggressively* and uses ECC to correct random slow cells

Design-Induced Latency Variation in DRAM

- Donghyuk Lee, Samira Khan, Lavanya Subramanian, Saugata Ghose, Rachata Ausavarungnirun, Gennady Pekhimenko, Vivek Seshadri, and Onur Mutlu,
"Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms"
Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), Urbana-Champaign, IL, USA, June 2017.

Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms

Donghyuk Lee, NVIDIA and Carnegie Mellon University

Samira Khan, University of Virginia

Lavanya Subramanian, Saugata Ghose, Rachata Ausavarungnirun, Carnegie Mellon University

Gennady Pekhimenko, Vivek Seshadri, Microsoft Research

Onur Mutlu, ETH Zürich and Carnegie Mellon University

Voltron: Exploiting the Voltage-Latency-Reliability Relationship

Executive Summary

- **DRAM (memory) power is significant in today's systems**
 - Existing low-voltage DRAM reduces voltage **conservatively**
- Goal: Understand and exploit the reliability and latency behavior of real DRAM chips under **aggressive reduced-voltage operation**
- Key experimental observations:
 - Huge voltage margin -- Errors occur beyond some voltage
 - Errors exhibit **spatial locality**
 - Higher operation latency mitigates voltage-induced errors
- Voltron: A new DRAM energy reduction mechanism
 - Reduce DRAM voltage **without introducing errors**
 - Use a **regression model** to select voltage that does not degrade performance beyond a chosen target → **7.3% system energy reduction**

Analysis of Latency-Voltage in DRAM Chips

- Kevin Chang, A. Giray Yaglikci, Saugata Ghose, Aditya Agrawal, Niladrish Chatterjee, Abhijith Kashyap, Donghyuk Lee, Mike O'Connor, Hasan Hassan, and Onur Mutlu,

"Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms"

*Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**), Urbana-Champaign, IL, USA, June 2017.*

Understanding Reduced-Voltage Operation in Modern DRAM Chips: Characterization, Analysis, and Mechanisms

Kevin K. Chang[†] Abdullah Giray Yağlıkçı[†] Saugata Ghose[†] Aditya Agrawal[¶] Niladrish Chatterjee[¶]
Abhijith Kashyap[†] Donghyuk Lee[¶] Mike O'Connor^{¶,‡} Hasan Hassan[§] Onur Mutlu^{§,†}

[†]Carnegie Mellon University

[¶]NVIDIA

[‡]The University of Texas at Austin

[§]ETH Zürich

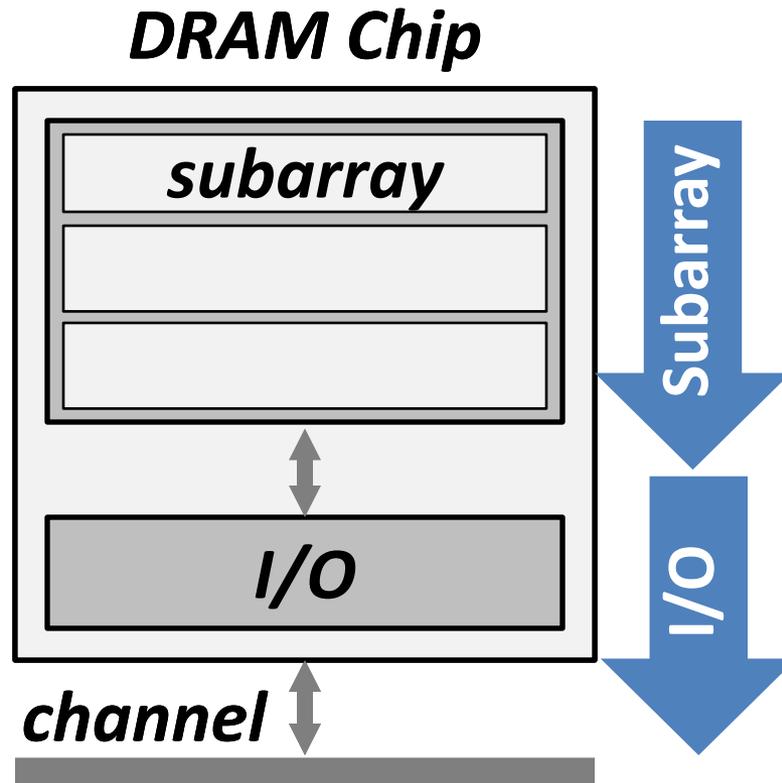
And, What If ...

- ... we can sacrifice reliability of some data to access it with even lower latency?

Fundamentally Low Latency Computing Architectures

Tiered Latency DRAM

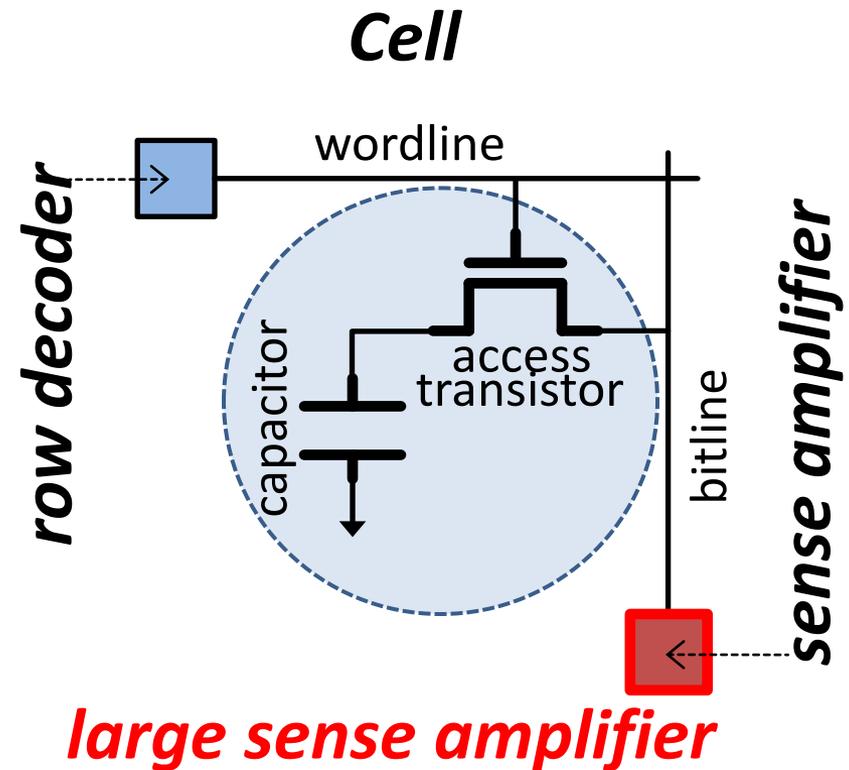
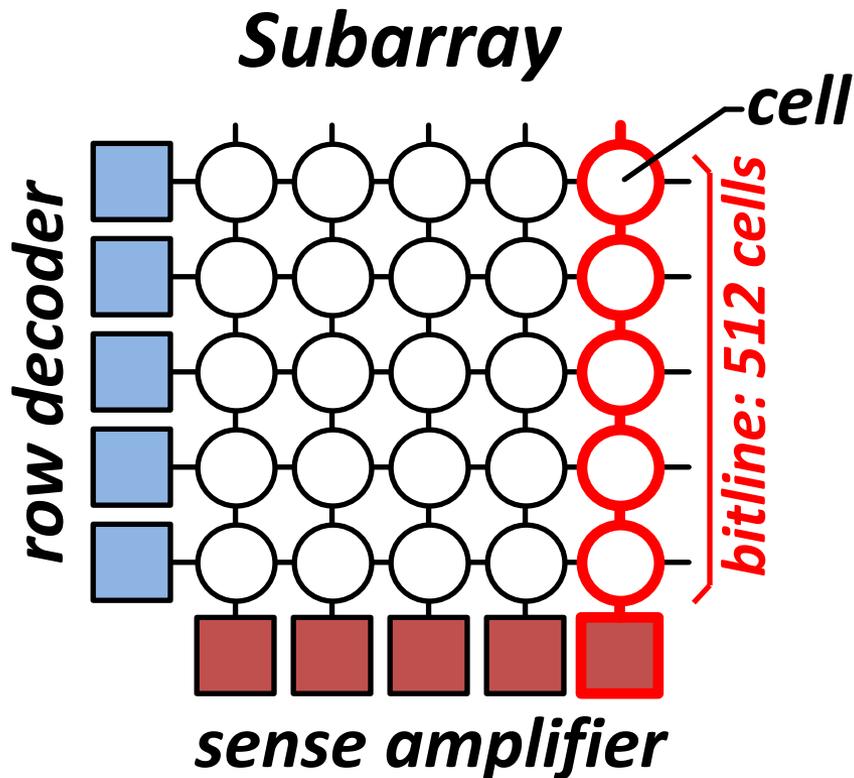
What Causes the Long Latency?



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

Dominant

Why is the Subarray So Slow?

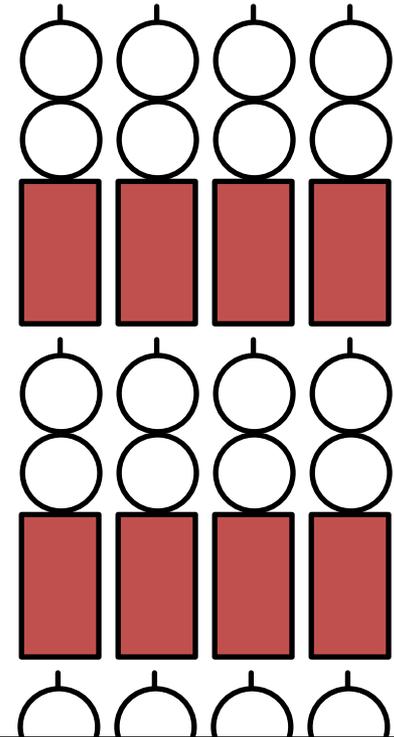
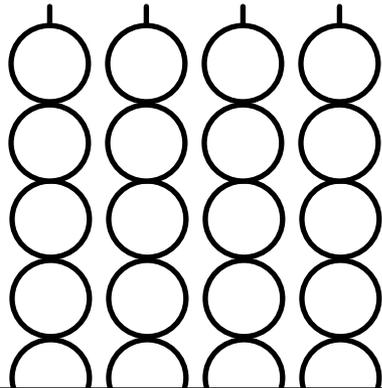


- Long bitline
 - Amortizes sense amplifier cost → Small area
 - Large bitline capacitance → High latency & power

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

Long Bitline

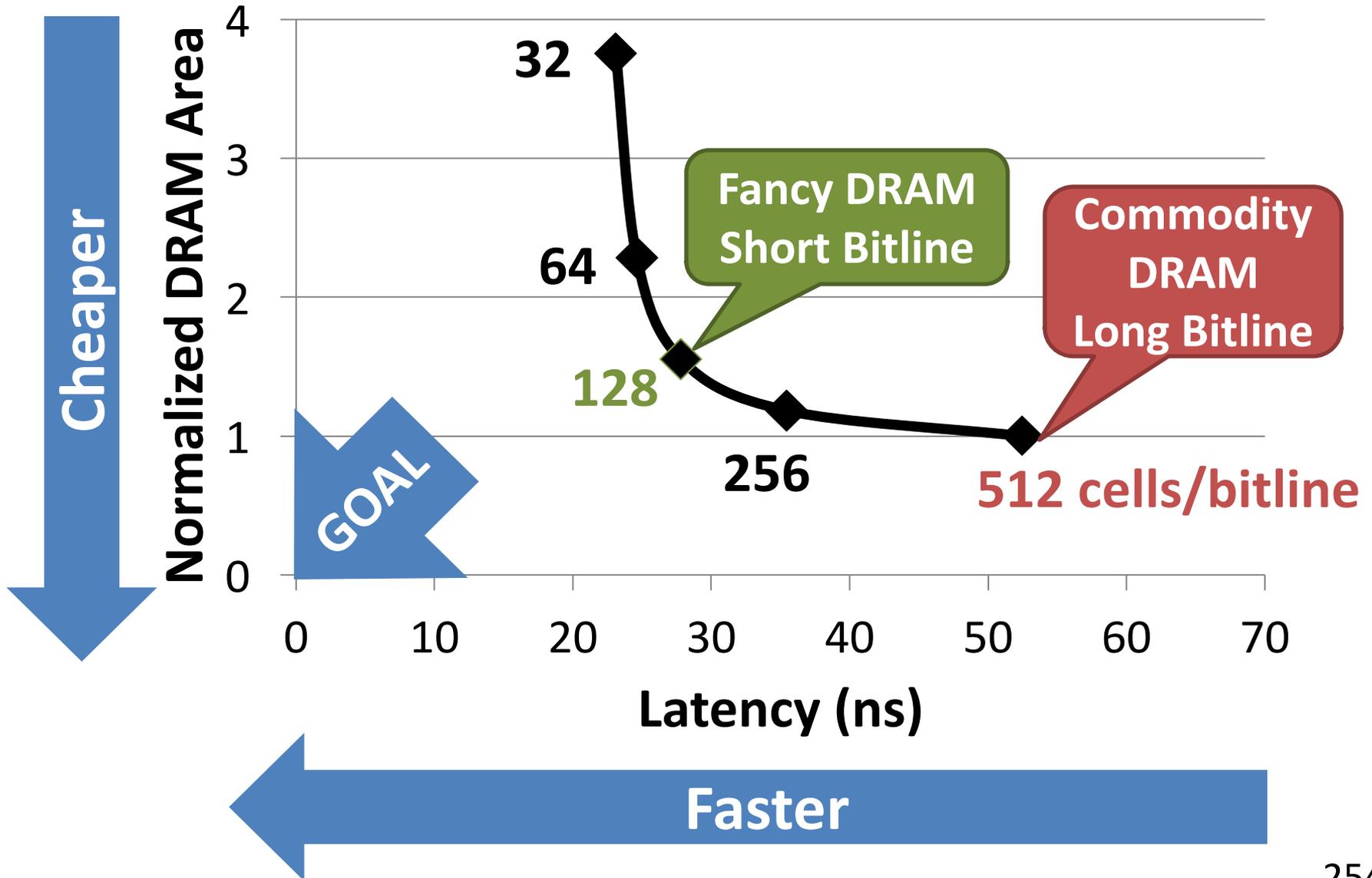
Short Bitline



Trade-Off: Area vs. Latency



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

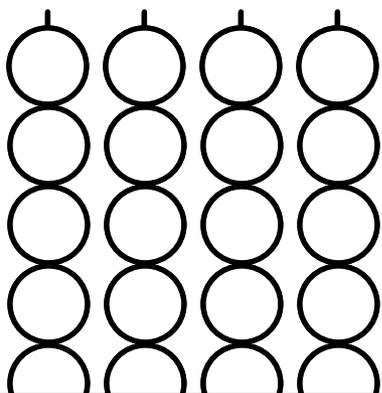


Approximating the Best of Both Worlds

Long Bitline

Small Area

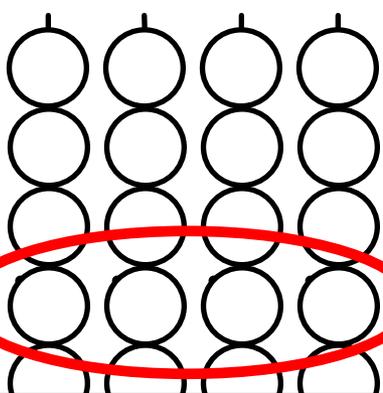
~~High Latency~~



Need Isolation

Our Proposal

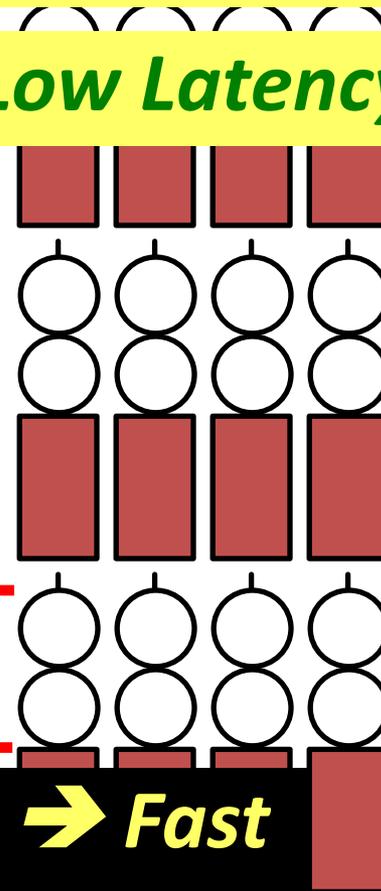
Add Isolation Transistors



Short Bitline

~~Large Area~~

Low Latency



tline → Fast

Approximating the Best of Both Worlds

Long Bitline Tiered-Latency DRAM Short Bitline

Small Area

Small Area

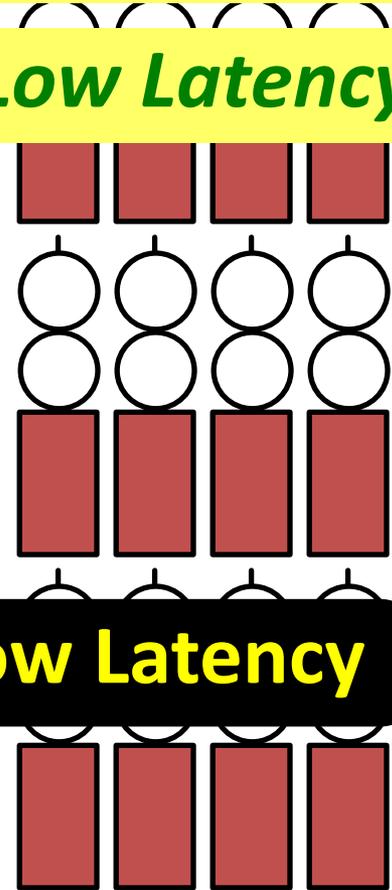
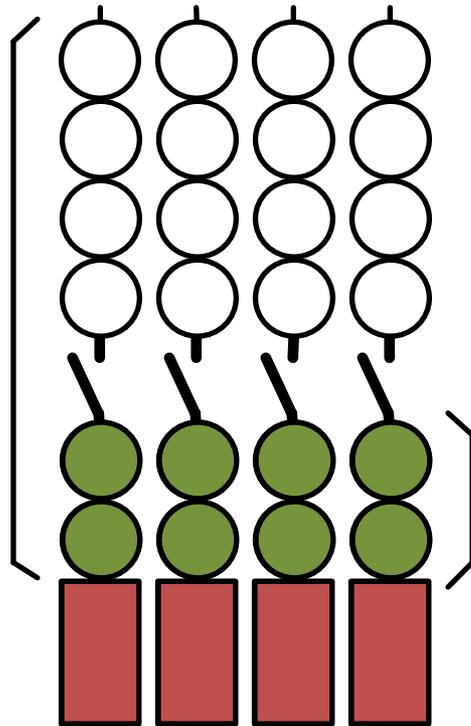
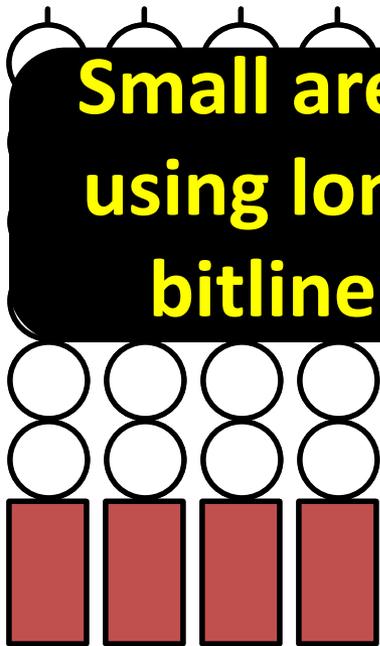
~~Large Area~~

~~High Latency~~

Low Latency

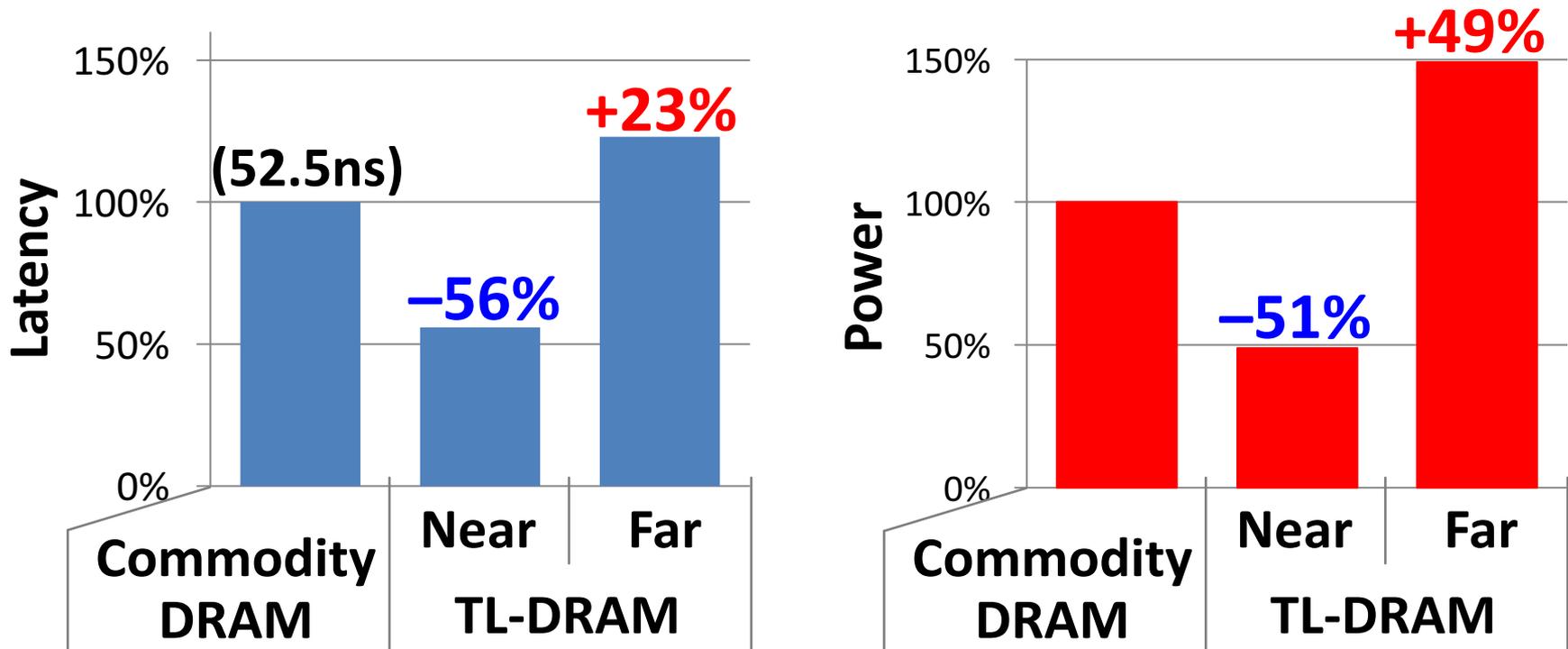
Low Latency

**Small area
using long
bitline**



Commodity DRAM vs. TL-DRAM [HPCA 2013]

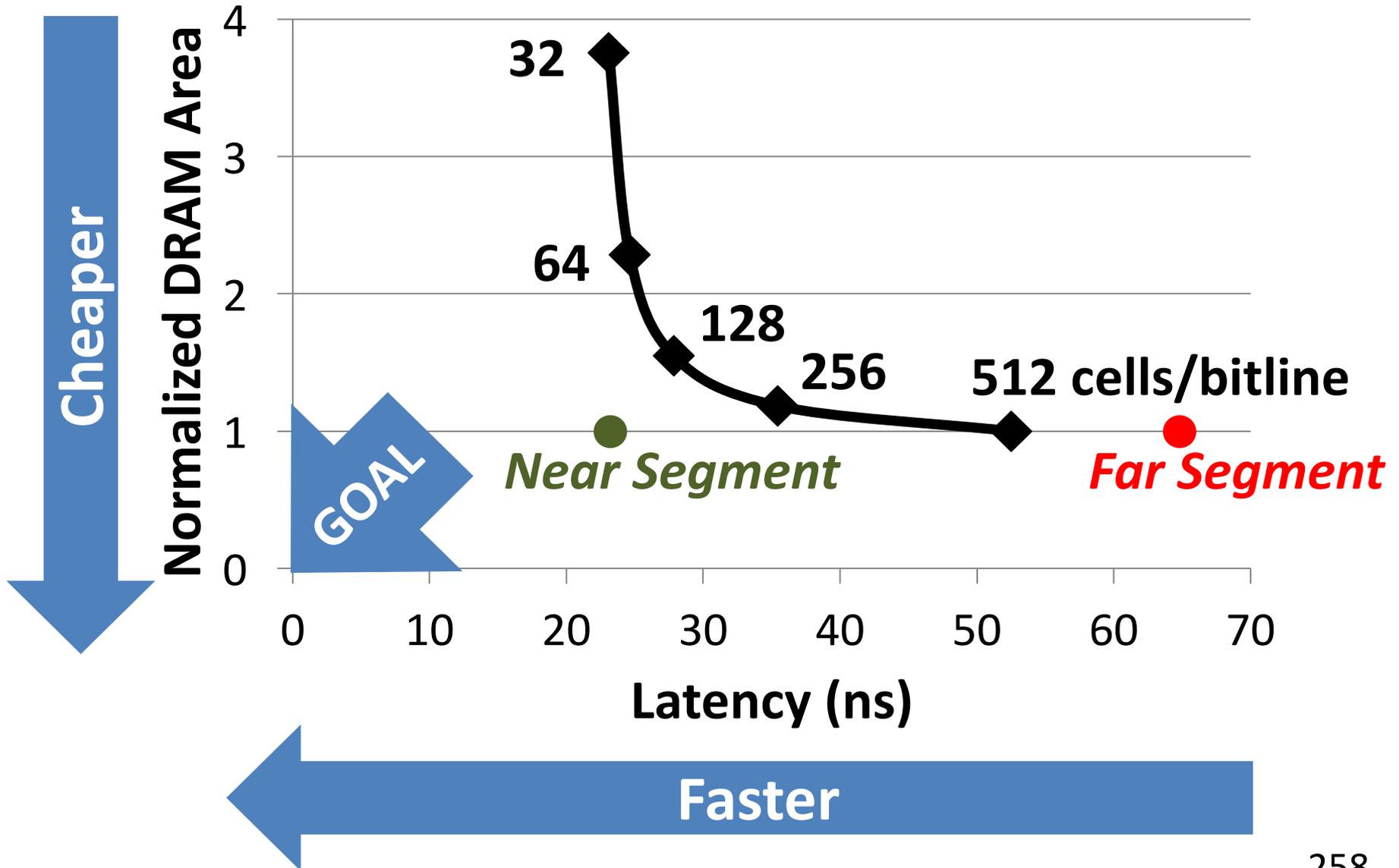
- DRAM Latency (tRC) • DRAM Power



- DRAM Area Overhead

~3%: mainly due to the isolation transistors

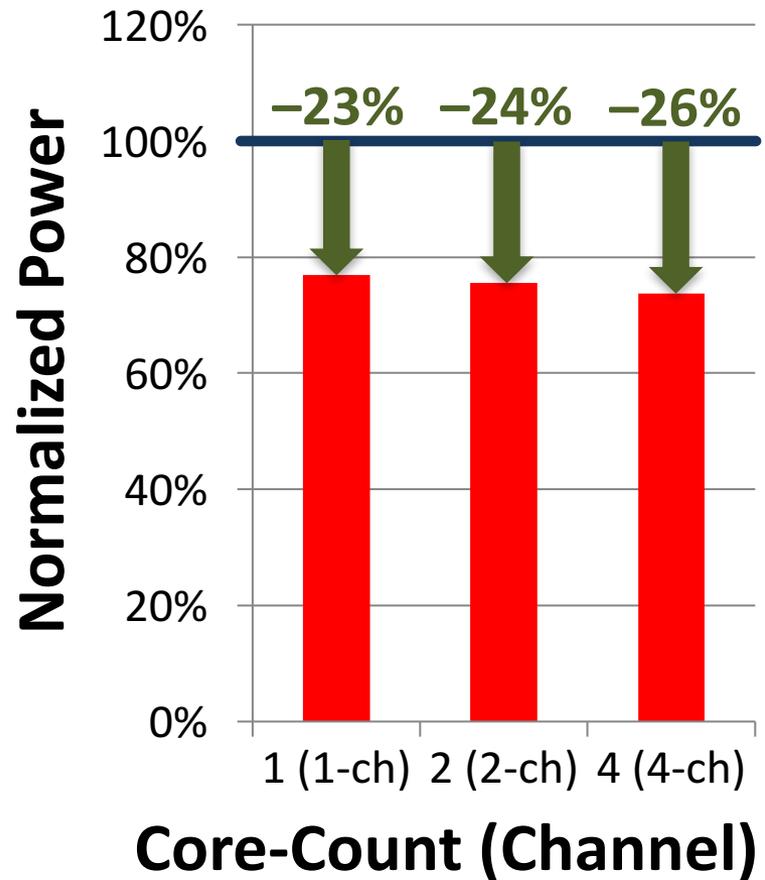
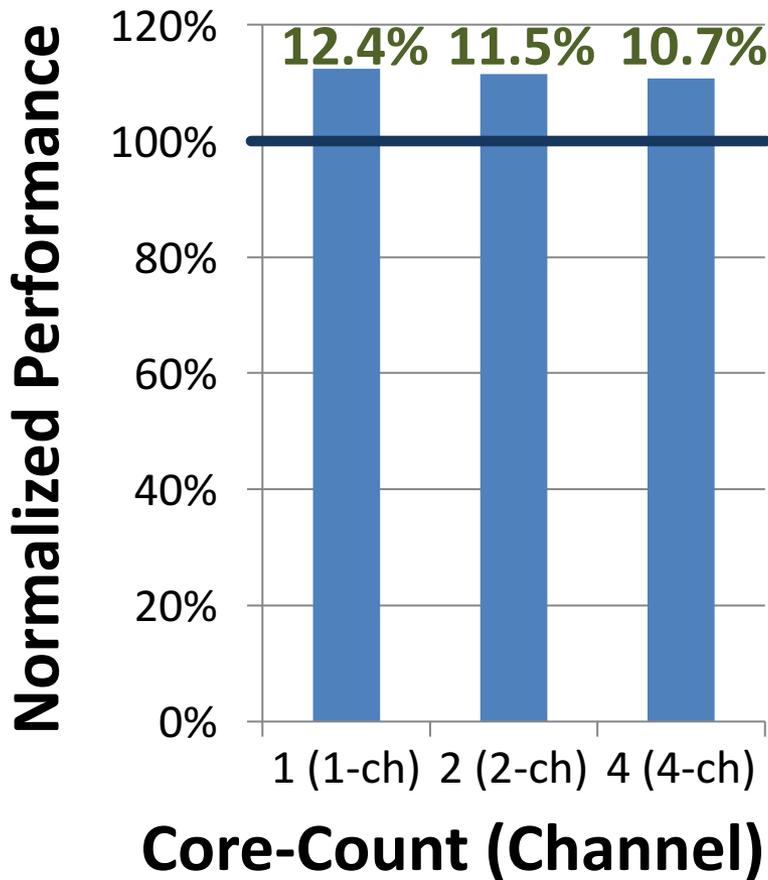
Trade-Off: Area (Die-Area) vs. Latency



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

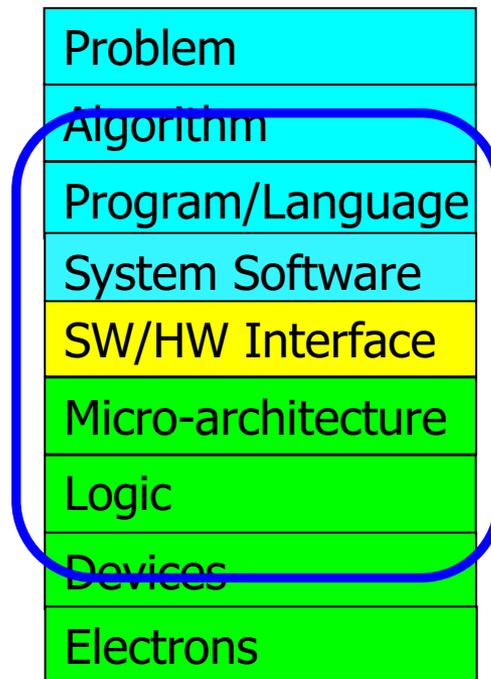
More on PIM

How to Enable Adoption of Processing in Memory

Barriers to Adoption of PIM

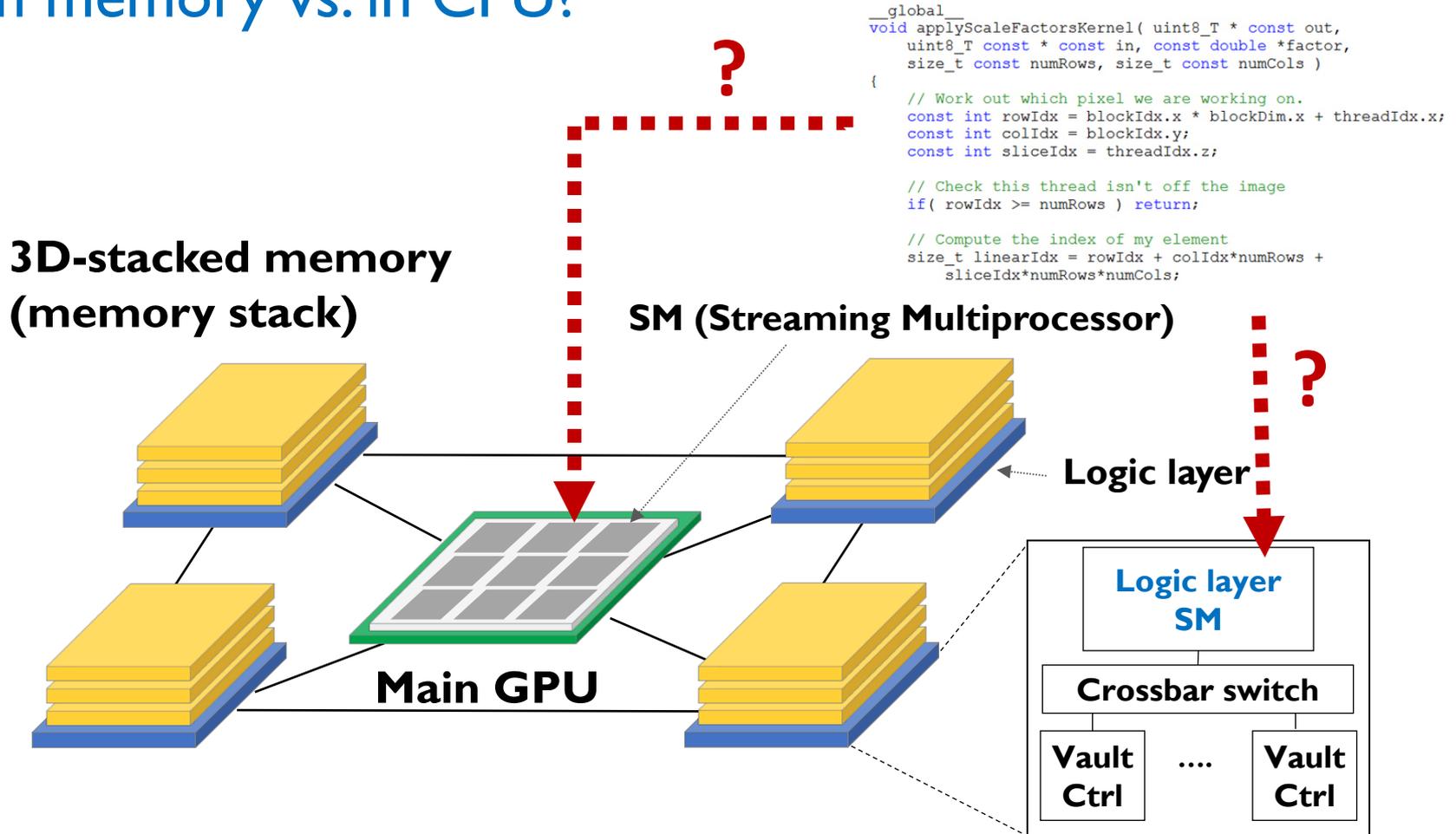
1. Functionality of and applications for PIM
2. Ease of programming (interfaces and compiler/HW support)
3. System support: coherence & virtual memory
4. Runtime systems for adaptive scheduling, data mapping, access/sharing control
5. Infrastructures to assess benefits and feasibility

We Need to Revisit the Entire Stack



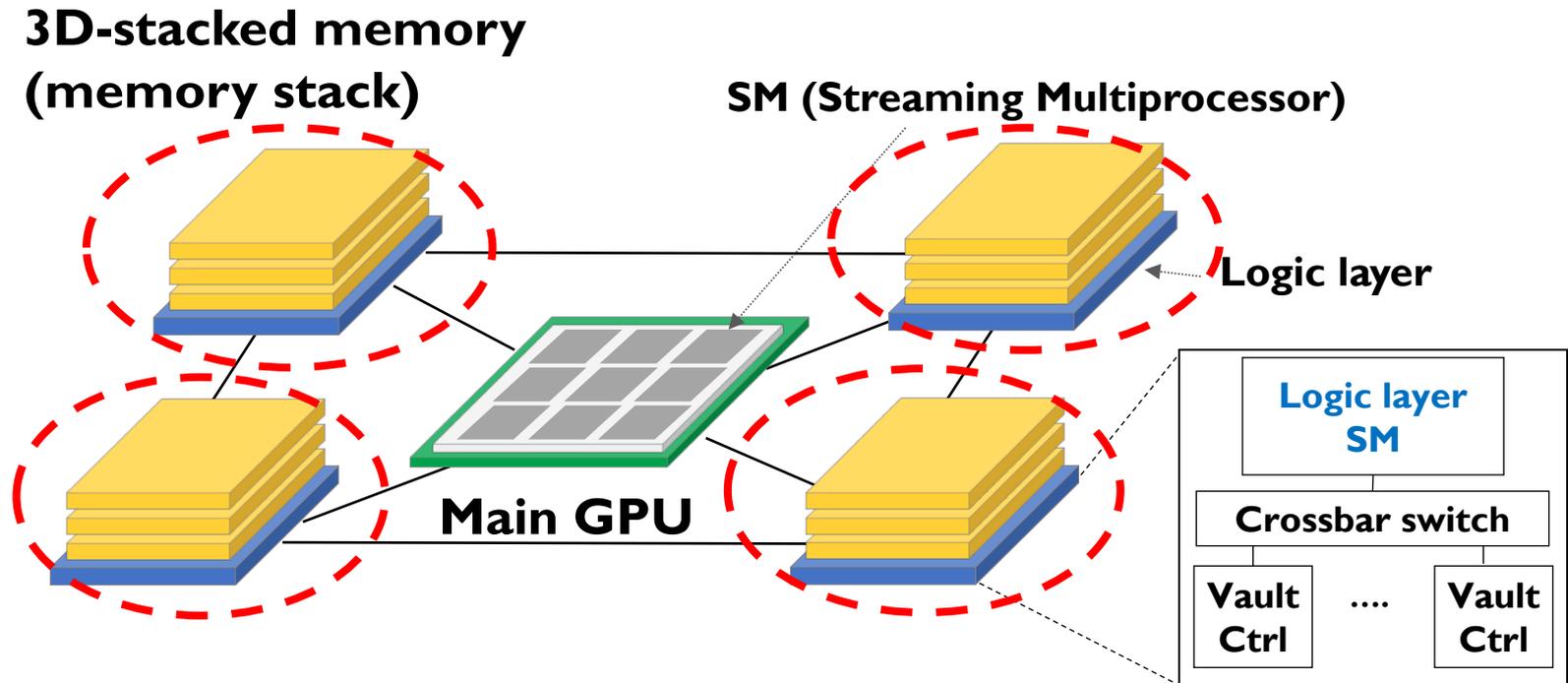
Key Challenge 1: Code Mapping

- **Challenge 1: Which operations should be executed in memory vs. in CPU?**



Key Challenge 2: Data Mapping

- **Challenge 2:** How should data be mapped to different 3D memory stacks?



How to Do the Code and Data Mapping?

- Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler, **"Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"**

Proceedings of the 43rd International Symposium on Computer Architecture (ISCA), Seoul, South Korea, June 2016.

[[Slides \(pptx\)](#) ([pdf](#))]

[[Lightning Session Slides \(pptx\)](#) ([pdf](#))]

Transparent Offloading and Mapping (TOM):

Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh[‡] Eiman Ebrahimi[†] Gwangsun Kim* Niladrish Chatterjee[†] Mike O'Connor[†]
Nandita Vijaykumar[‡] Onur Mutlu^{§‡} Stephen W. Keckler[†]

[‡]Carnegie Mellon University [†]NVIDIA ^{*}KAIST [§]ETH Zürich

How to Schedule Code?

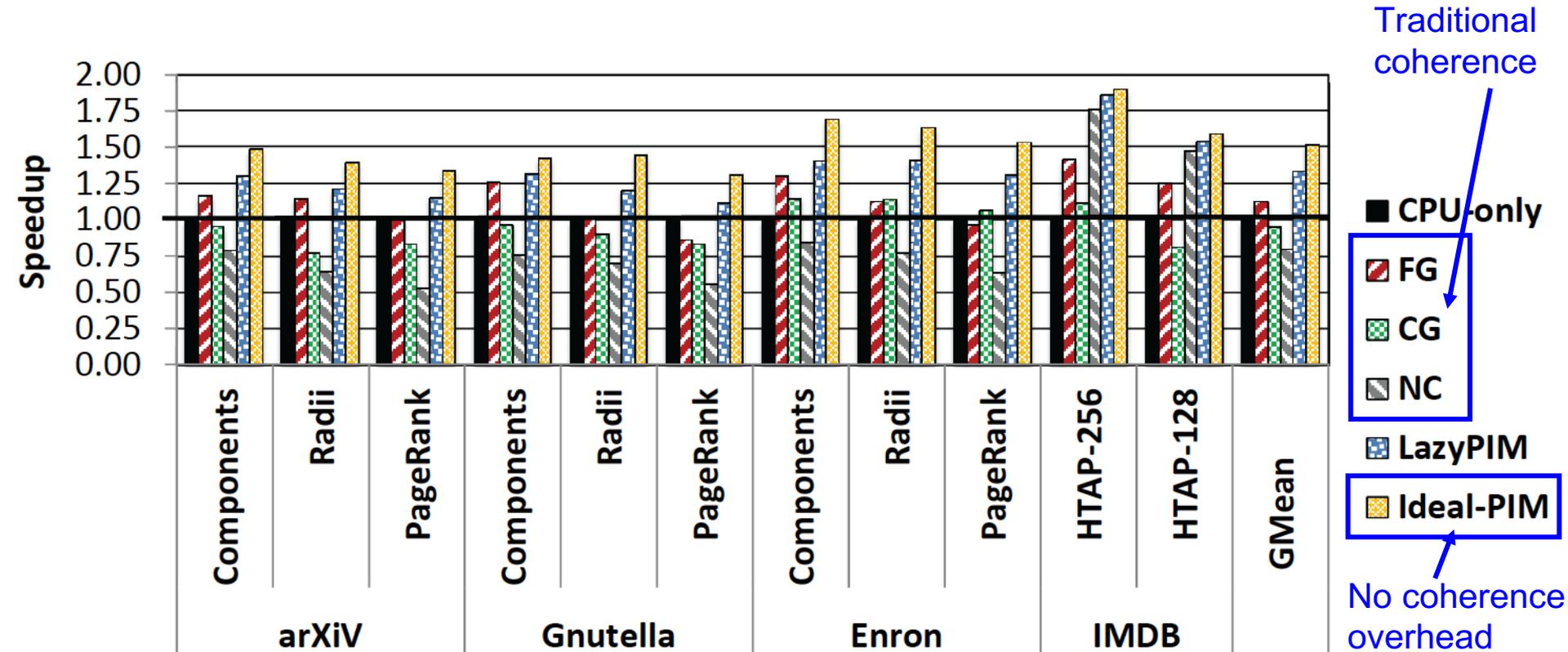
- Ashutosh Pattnaik, Xulong Tang, Adwait Jog, Onur Kayiran, Asit K. Mishra, Mahmut T. Kandemir, Onur Mutlu, and Chita R. Das, **"Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities"**
Proceedings of the 25th International Conference on Parallel Architectures and Compilation Techniques (PACT), Haifa, Israel, September 2016.

Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities

Ashutosh Pattnaik¹ Xulong Tang¹ Adwait Jog² Onur Kayiran³
Asit K. Mishra⁴ Mahmut T. Kandemir¹ Onur Mutlu^{5,6} Chita R. Das¹

¹Pennsylvania State University ²College of William and Mary
³Advanced Micro Devices, Inc. ⁴Intel Labs ⁵ETH Zürich ⁶Carnegie Mellon University

Challenge: Coherence for Hybrid CPU-PIM Apps



How to Maintain Coherence?

- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,
"LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory"
IEEE Computer Architecture Letters (**CAL**), June 2016.

LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory

Amirali Boroumand[†], Saugata Ghose[†], Minesh Patel[†], Hasan Hassan^{†§}, Brandon Lucia[†],
Kevin Hsieh[†], Krishna T. Malladi^{*}, Hongzhong Zheng^{*}, and Onur Mutlu^{‡†}

[†] *Carnegie Mellon University* ^{*} *Samsung Semiconductor, Inc.* [§] *TOBB ETÜ* [‡] *ETH Zürich*

How to Support Virtual Memory?

- Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K. Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu,
"Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation"
Proceedings of the 34th IEEE International Conference on Computer Design (ICCD), Phoenix, AZ, USA, October 2016.

Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation

Kevin Hsieh[†] Samira Khan[‡] Nandita Vijaykumar[†]

Kevin K. Chang[†] Amirali Boroumand[†] Saugata Ghose[†] Onur Mutlu^{§†}

[†]*Carnegie Mellon University* [‡]*University of Virginia* [§]*ETH Zürich*

How to Design Data Structures for PIM?

- Zhiyu Liu, Irina Calciu, Maurice Herlihy, and Onur Mutlu,
"Concurrent Data Structures for Near-Memory Computing"
Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), Washington, DC, USA, July 2017.
[\[Slides \(pptx\) \(pdf\)\]](#)

Concurrent Data Structures for Near-Memory Computing

Zhiyu Liu

Computer Science Department
Brown University
zhiyu.liu@brown.edu

Irina Calciu

VMware Research Group
icalciu@vmware.com

Maurice Herlihy

Computer Science Department
Brown University
mph@cs.brown.edu

Onur Mutlu

Computer Science Department
ETH Zürich
onur.mutlu@inf.ethz.ch

Simulation Infrastructures for PIM

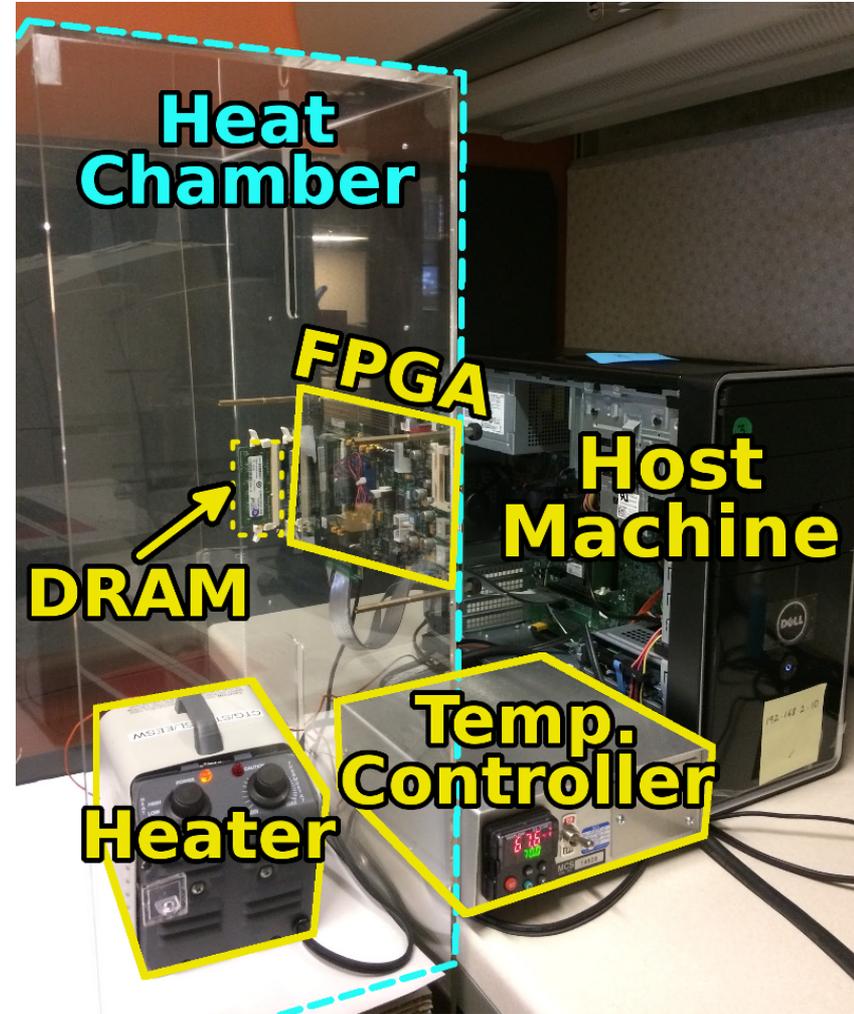
- **Ramulator** extended for PIM
 - Flexible and extensible DRAM simulator
 - Can model many different memory standards and proposals
 - Kim+, “**Ramulator: A Flexible and Extensible DRAM Simulator**”, IEEE CAL 2015.
 - <https://github.com/CMU-SAFARI/ramulator>

Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim¹ Weikun Yang^{1,2} Onur Mutlu¹
¹Carnegie Mellon University ²Peking University

An FPGA-based Test-bed for PIM?

- Hasan Hassan et al., **SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies** HPCA 2017.
- Flexible
- Easy to Use (C++ API)
- Open-source
github.com/CMU-SAFARI/SoftMC



Some PIM Applications

Goals

- Understand the primitives, architectures, and benefits of PIM by carefully examining many important workloads
- Develop a common workload suite for PIM research

Genome Read In-Memory (GRIM) Filter: Fast Location Filtering in DNA Read Mapping with Emerging Memory Technologies

Jeremie Kim,

Damla Senol, Hongyi Xin, Donghyuk Lee,
Saugata Ghose, Mohammed Alser, Hasan Hassan,
Oguz Ergin, Can Alkan, and Onur Mutlu

Carnegie Mellon



ETH zürich

- **Genome Read Mapping** is a very important problem and is the first step in many types of genomic analysis
 - Could lead to improved health care, medicine, quality of life
- Read mapping is an **approximate string matching** problem
 - Find the best fit of 100 character strings into a 3 billion character dictionary
 - **Alignment** is currently the best method for determining the similarity between two strings, but is **very expensive**
- We propose an in-memory processing algorithm **GRIM-Filter** for accelerating read mapping, by reducing the number of required alignments
- We implement GRIM-Filter using **in-memory processing** within **3D-stacked memory** and show up to **3.7x speedup**.

GRIM-Filter in 3D-stacked DRAM

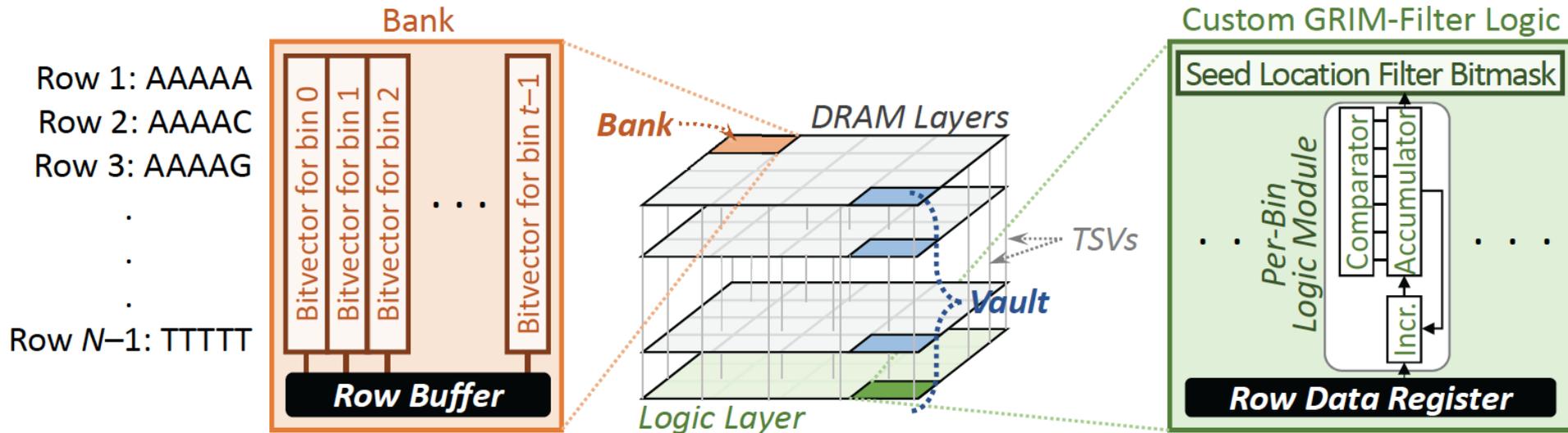


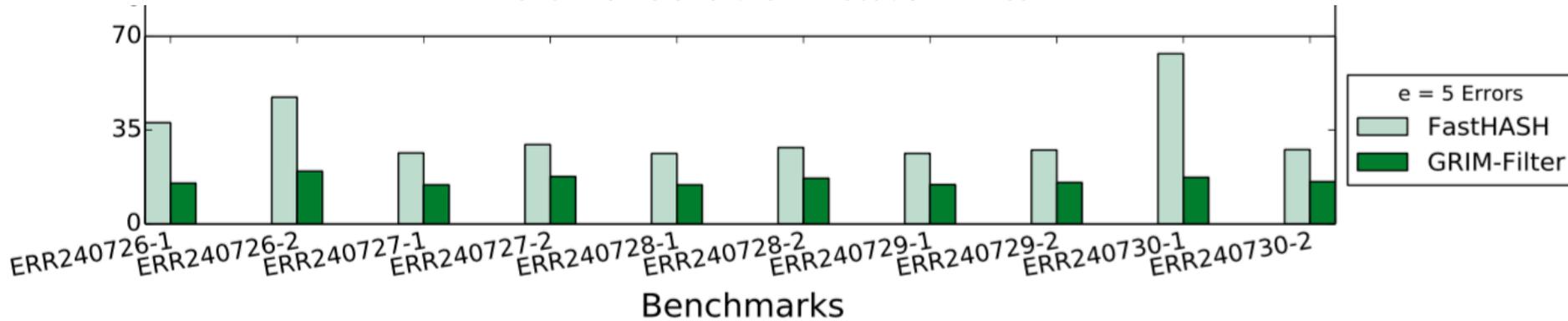
Figure 7: *Left block:* GRIM-Filter bitvector layout within a DRAM bank. *Center block:* 3D-stacked DRAM with tightly integrated logic layer stacked underneath with TSVs for a high intra-DRAM data transfer bandwidth. *Right block:* Custom GRIM-Filter logic placed in the logic layer.

- The layout of bit vectors in a bank enables filtering many bins in parallel
- Customized logic for accumulation and comparison per genome segment
 - Low area overhead, simple implementation

GRIM-Filter Performance

Time (x1000 seconds)

Benchmarks and their Execution Times

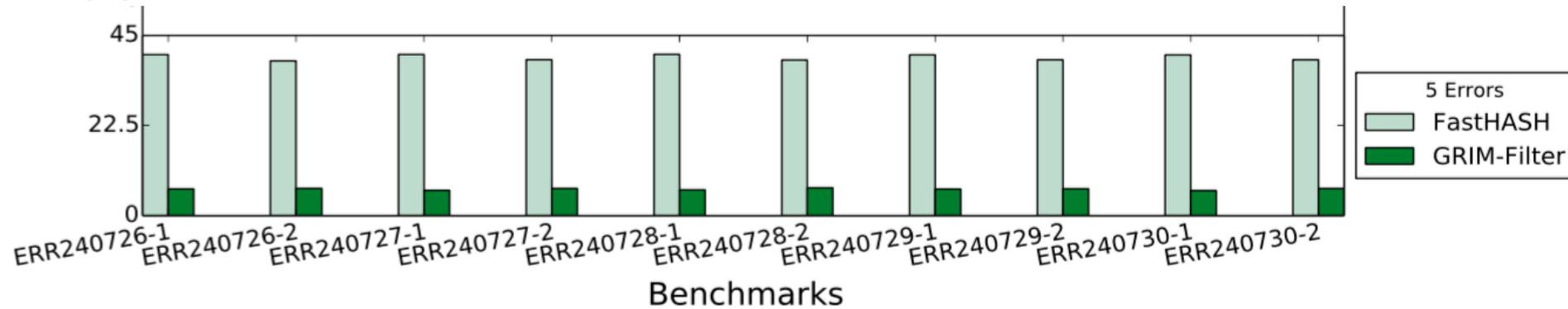


1.8x-3.7x performance benefit across real data sets

GRIM-Filter False Positive Rate

False Positive
Rate (%)

Benchmarks and their False Positive Rates



5.6x-6.4x False Positive reduction across real data sets

- We propose an **in memory filter algorithm** to **accelerate end-to-end genome read mapping** by reducing the number of required alignments
- Compared to the previous best filter
 - We observed **1.8x-3.7x speedup**
 - We observed **5.6x-6.4x fewer false positives**
- **GRIM-Filter is a universal filter** that can be applied to any genome read mapper

PIM-Based DNA Sequence Analysis

- Jeremie Kim, Damla Senol, Hongyi Xin, Donghyuk Lee, Mohammed Alser, Hasan Hassan, Oguz Ergin, Can Alkan, and Onur Mutlu, **"Genome Read In-Memory (GRIM) Filter: Fast Location Filtering in DNA Read Mapping Using Emerging Memory Technologies"** *Pacific Symposium on Biocomputing (PSB) Poster Session*, Hawaii, January 2017.
[[Poster \(pdf\) \(pptx\)](#)] [[Abstract \(pdf\)](#)]
- **To Appear in APBC 2018 and BMC Genomics 2018.**

GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies

Jeremie S. Kim^{1,6*}, Damla Senol Cali¹, Hongyi Xin², Donghyuk Lee³, Saugata Ghose¹, Mohammed Alser⁴, Hasan Hassan⁶, Oguz Ergin⁵, Can Alkan^{*4}, and Onur Mutlu^{*6,1}

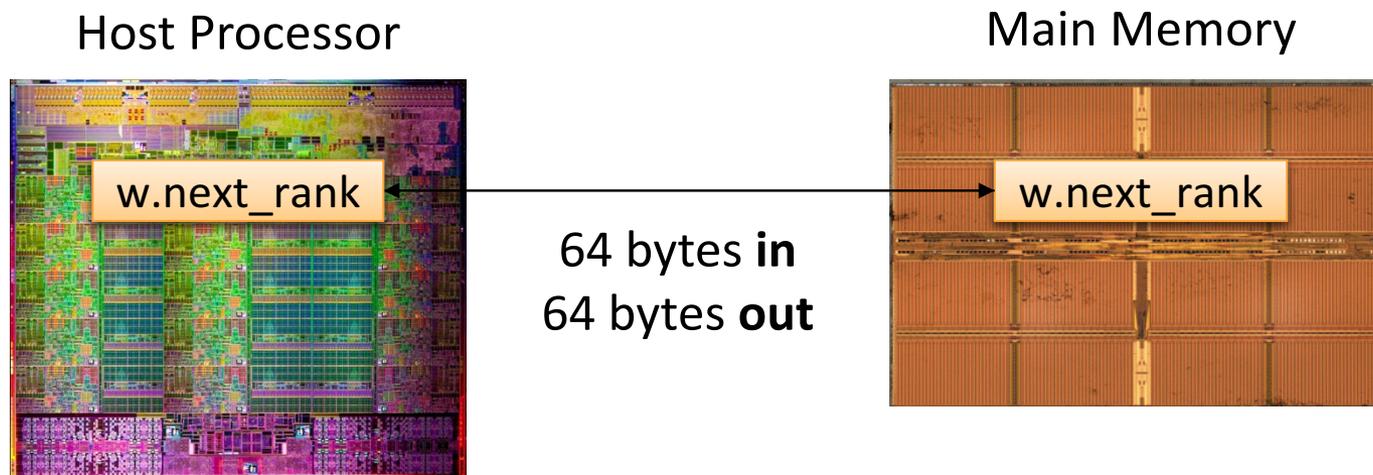
PIM-Enabled Instructions

PEI: PIM-Enabled Instructions (Ideas)

- **Goal:** Develop mechanisms to get the most out of near-data processing with **minimal cost, minimal changes to the system, no changes to the programming model**
- **Key Idea 1:** Expose each PIM operation as a **cache-coherent, virtually-addressed host processor instruction** (called PEI) that operates on **only a single cache block**
 - e.g., `__pim_add(&w.next_rank, value) → pim.add r1, (r2)`
 - No changes sequential execution/programming model
 - No changes to virtual memory
 - Minimal changes to cache coherence
 - No need for data mapping: Each PEI restricted to a single memory module
- **Key Idea 2:** **Dynamically decide where to execute a PEI** (i.e., the host processor or PIM accelerator) based on simple locality characteristics and simple hardware predictors
 - Execute each operation at the location that provides the best performance

Simple PIM Operations as ISA Extensions (I)

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



Conventional Architecture

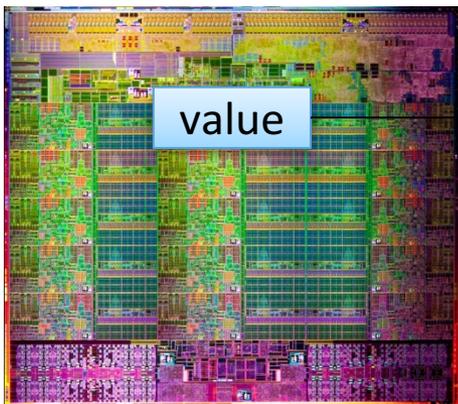
Simple PIM Operations as ISA Extensions (II)

```
for (v: graph.vertices) {  
    value = weight * v.rank;  
    for (w: v.successors) {  
        __pim_add(&w.next_rank, value);  
    }  
}
```

pim.add r1, (r2)

__pim_add(&w.next_rank, value);

Host Processor



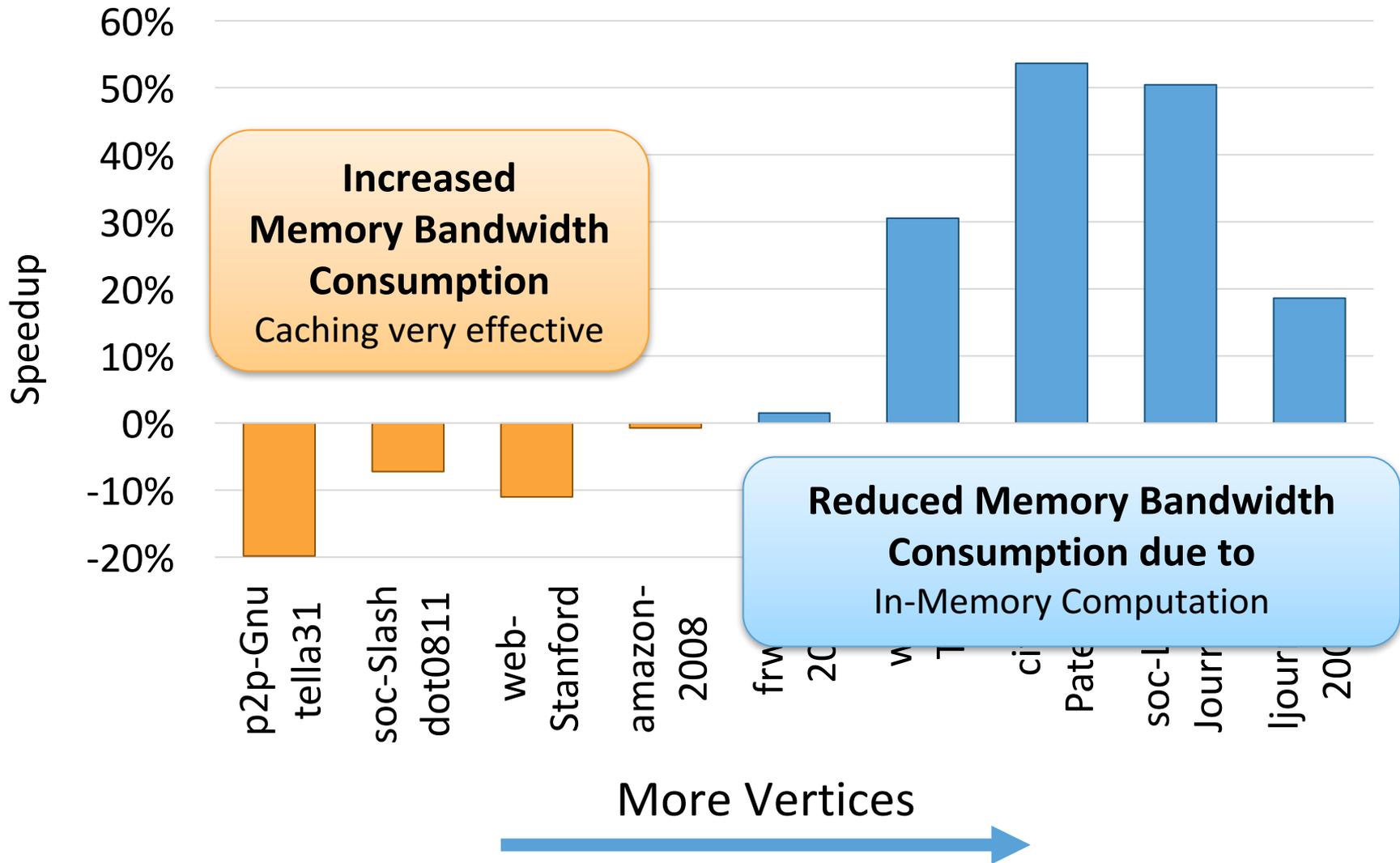
Main Memory



8 bytes in
0 bytes out

In-Memory Addition

Always Executing in Memory? Not A Good Idea



PEI: PIM-Enabled Instructions: Examples

Table 1: Summary of Supported PIM Operations

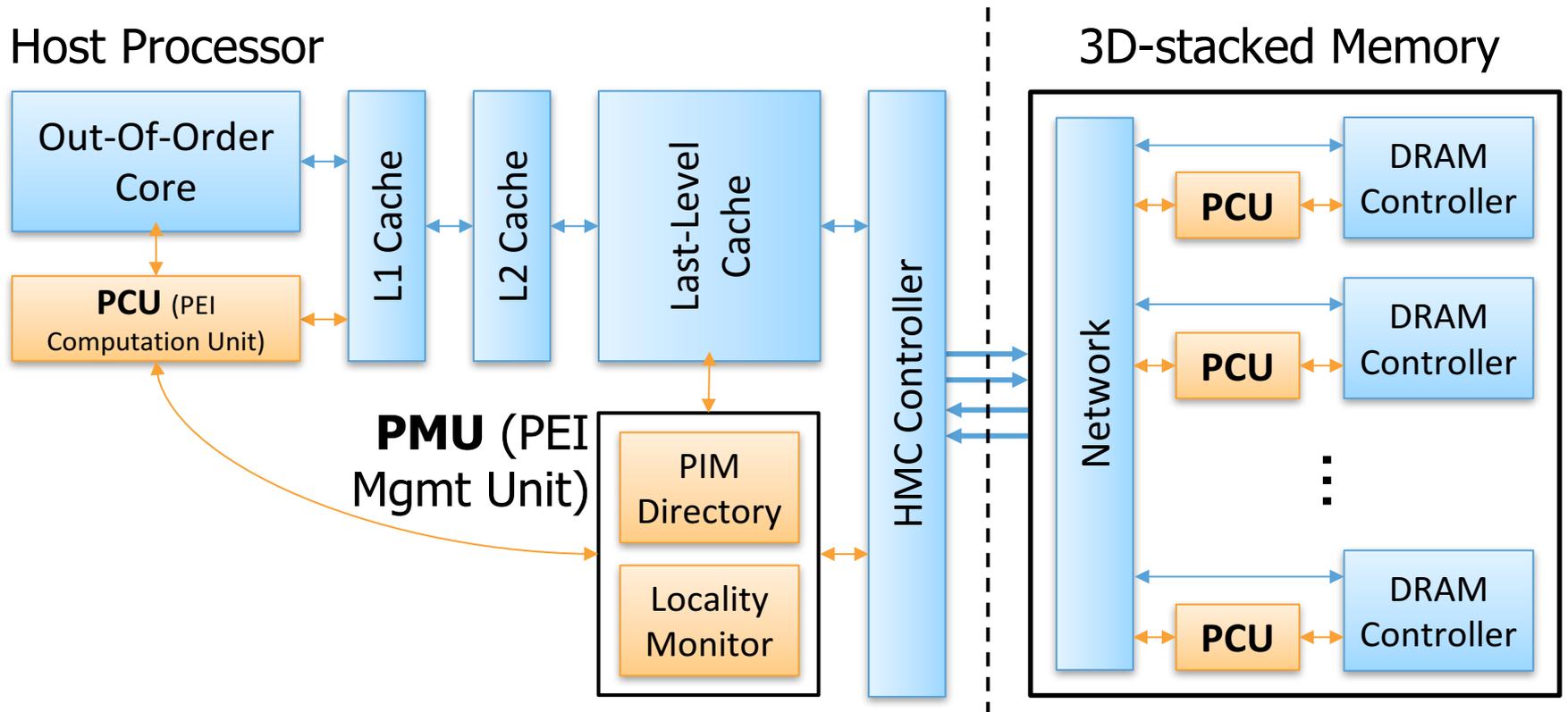
Operation	R	W	Input	Output	Applications
8-byte integer increment	O	O	0 bytes	0 bytes	AT
8-byte integer min	O	O	8 bytes	0 bytes	BFS, SP, WCC
Floating-point add	O	O	8 bytes	0 bytes	PR
Hash table probing	O	X	8 bytes	9 bytes	HJ
Histogram bin index	O	X	1 byte	16 bytes	HG, RP
Euclidean distance	O	X	64 bytes	4 bytes	SC
Dot product	O	X	32 bytes	8 bytes	SVM

- Executed either in memory or in the processor: dynamic decision
 - Low-cost locality monitoring for a single instruction
- Cache-coherent, virtually-addressed, single cache block only
- Atomic between different PEIs
- *Not* atomic with normal instructions (use *pfence* for ordering)

PIM-Enabled Instructions

- Key to practicality: **single-cache-block restriction**
 - **Each PEI can access *at most one last-level cache block***
 - Similar restrictions exist in atomic instructions
- Benefits
 - **Localization:** each PEI is bounded to one memory module
 - **Interoperability:** easier support for cache coherence and virtual memory
 - **Simplified locality monitoring:** data locality of PEIs can be identified simply by the cache control logic

Example PEI Microarchitecture



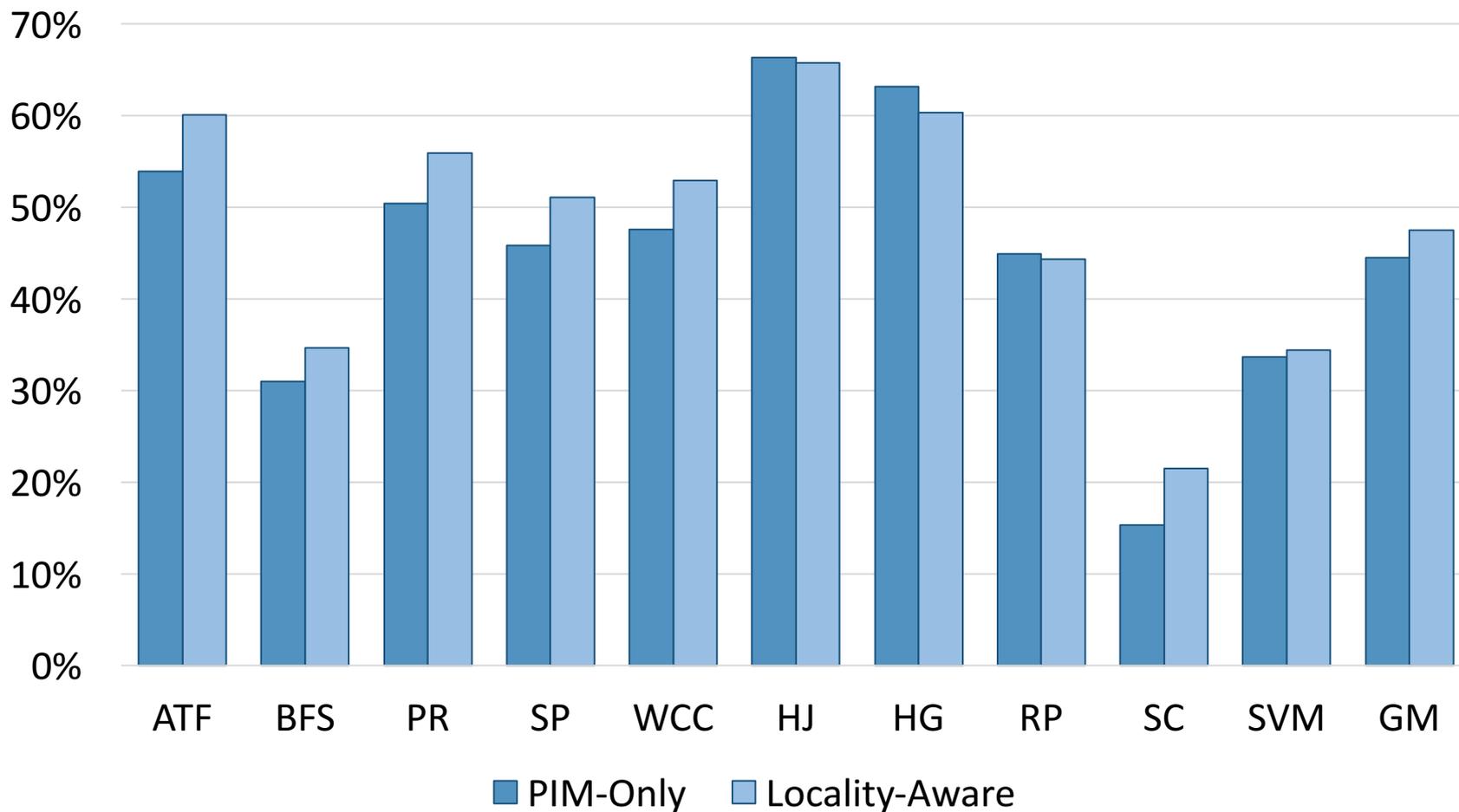
Example PEI uArchitecture

Evaluated Data-Intensive Applications

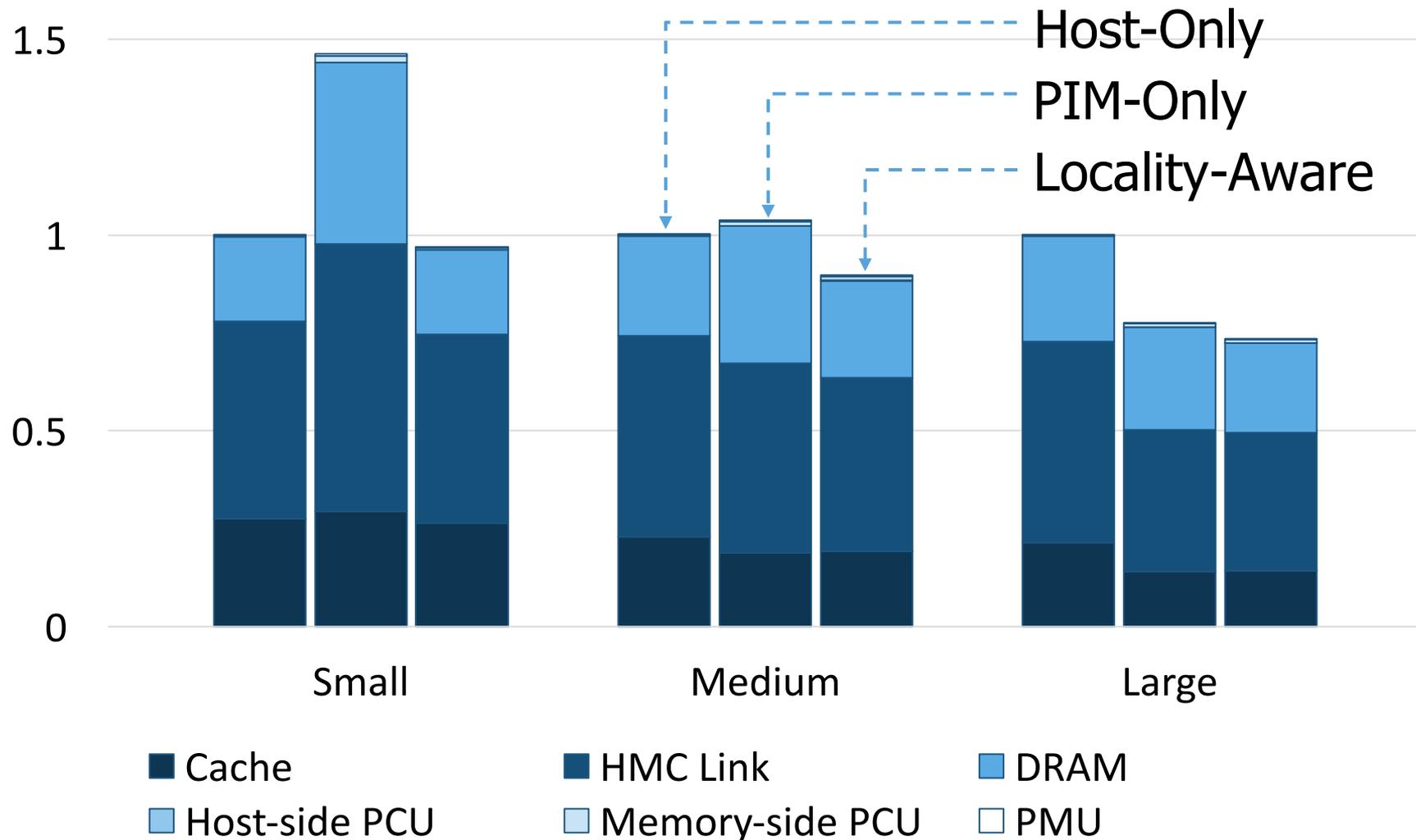
- **Ten emerging data-intensive workloads**
 - **Large-scale graph processing**
 - Average teenage follower, BFS, PageRank, single-source shortest path, weakly connected components
 - **In-memory data analytics**
 - Hash join, histogram, radix partitioning
 - **Machine learning and data mining**
 - Streamcluster, SVM-RFE
- Three input sets (small, medium, large) for each workload to show the impact of data locality

PEI Performance Delta: Large Data Sets

(Large Inputs, Baseline: Host-Only)



PEI Energy Consumption



More on PIM-Enabled Instructions

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi, **"PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture"** *Proceedings of the 42nd International Symposium on Computer Architecture (ISCA)*, Portland, OR, June 2015. [[Slides \(pdf\)](#)] [[Lightning Session Slides \(pdf\)](#)]

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

Junwhan Ahn Sungjoo Yoo Onur Mutlu[†] Kiyoung Choi

junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

[†]Carnegie Mellon University

More on RowHammer and Memory Reliability

A Deeper Dive into DRAM Reliability Issues

Root Causes of Disturbance Errors

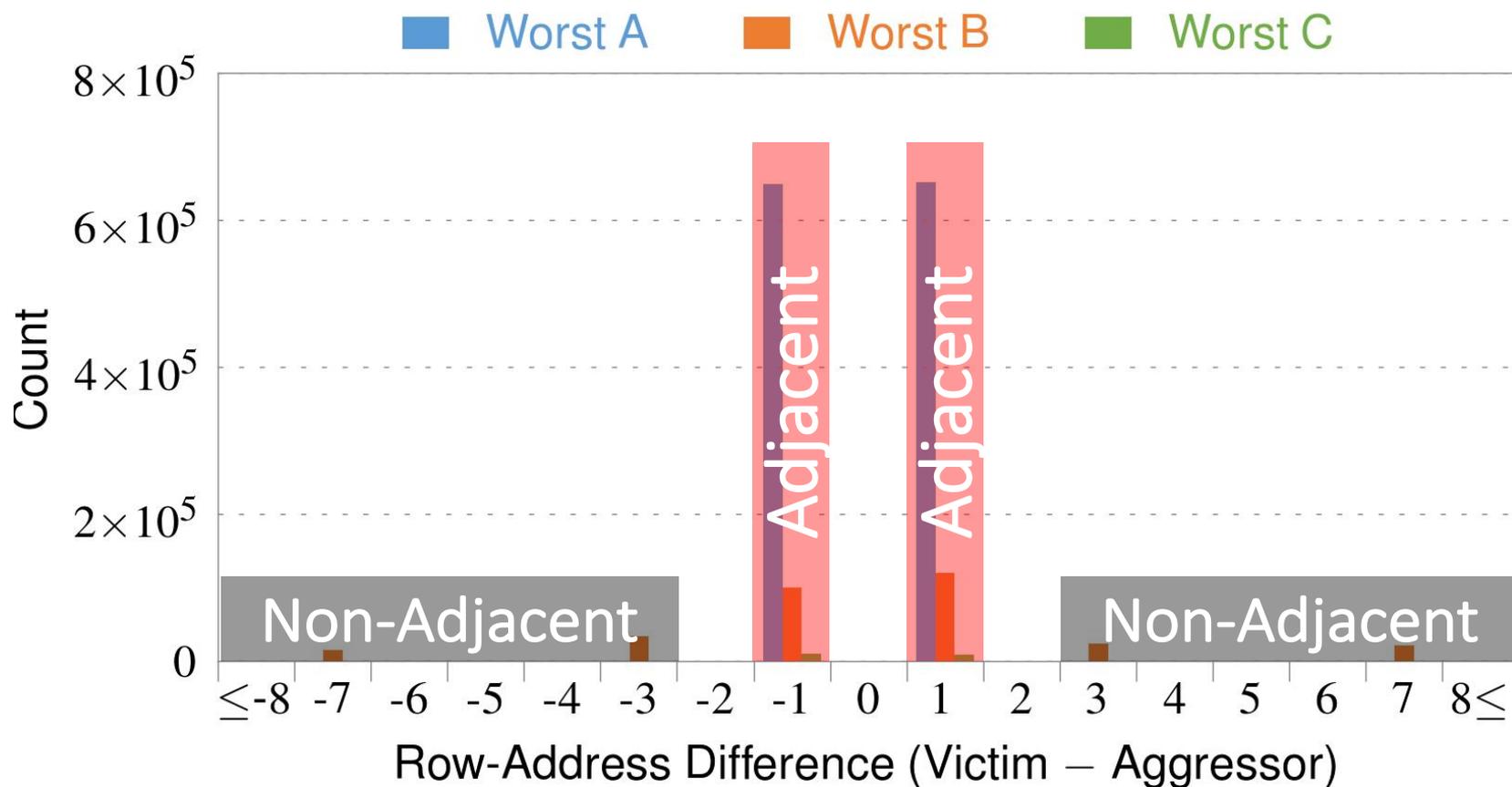
- *Cause 1: Electromagnetic coupling*
 - Toggling the wordline voltage briefly increases the voltage of adjacent wordlines
 - Slightly opens adjacent rows → Charge leakage
- *Cause 2: Conductive bridges*
- *Cause 3: Hot-carrier injection*

Confirmed by at least one manufacturer

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

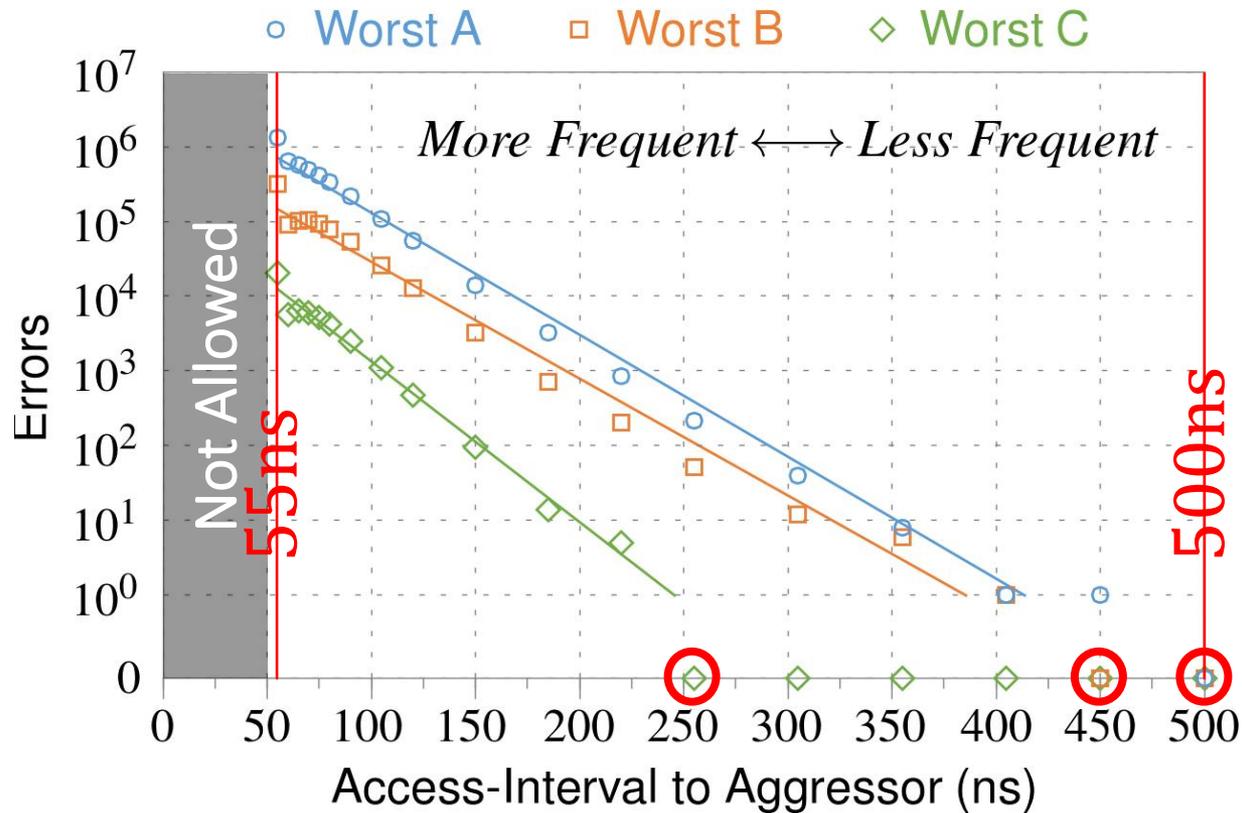
4. Adjacency: Aggressor & Victim



Note: For three modules with the most errors (only first bank)

Most aggressors & victims are adjacent

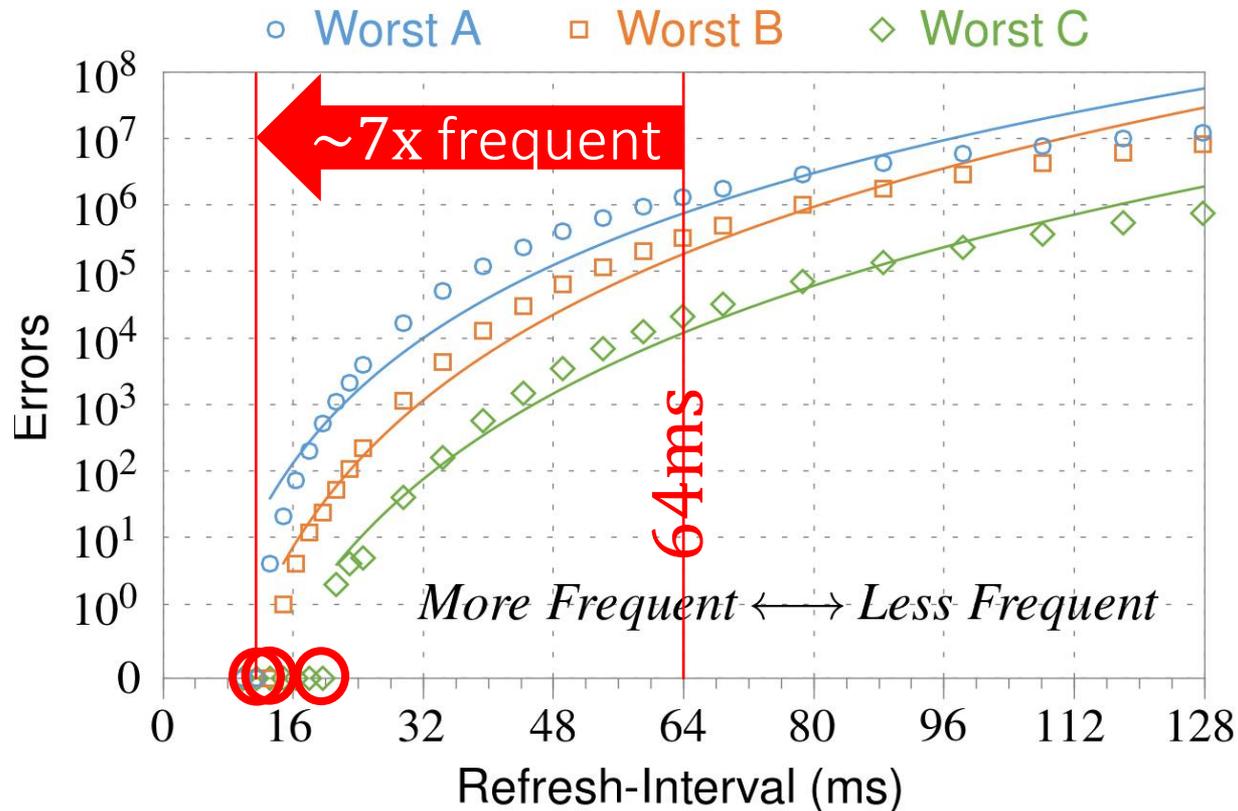
1 Access Interval (Aggressor)



Note: For three modules with the most errors (only first bank)

Less frequent accesses → Fewer errors

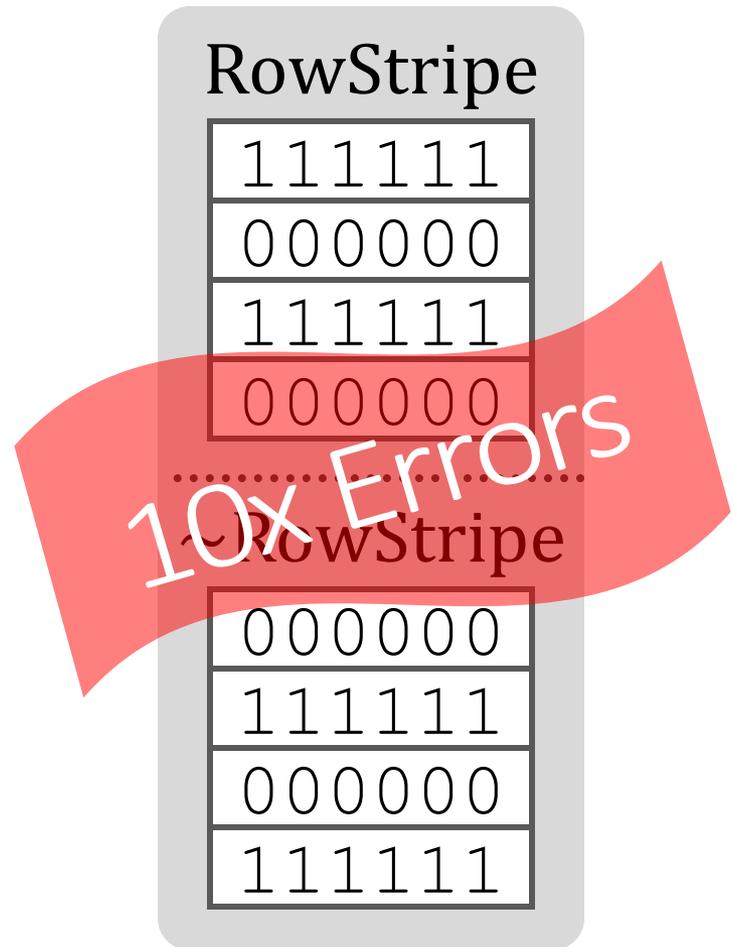
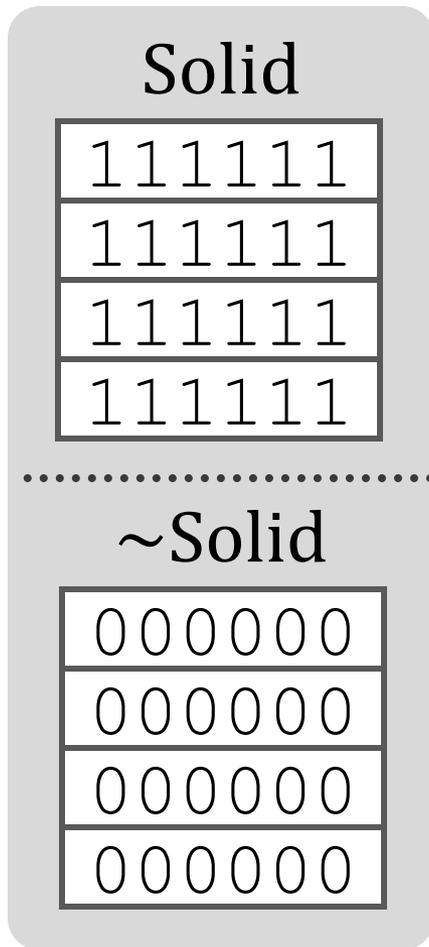
2 Refresh Interval



Note: Using three modules with the most errors (only first bank)

More frequent refreshes → Fewer errors

3 Data Pattern



Errors affected by data stored in other cells

6. Other Results (in Paper)

- *Victim Cells \neq Weak Cells (i.e., leaky cells)*
 - Almost no overlap between them
- *Errors not strongly affected by temperature*
 - Default temperature: 50°C
 - At 30°C and 70°C, number of errors changes <15%
- *Errors are repeatable*
 - Across ten iterations of testing, >70% of victim cells had errors in every iteration

6. Other Results (in Paper) cont'd

- *As many as 4 errors per cache-line*
 - Simple ECC (e.g., SECDED) cannot prevent all errors
- *Number of cells & rows affected by aggressor*
 - Victims cells per aggressor: ≤ 110
 - Victims rows per aggressor: ≤ 9
- *Cells affected by two aggressors on either side*
 - Very small fraction of victim cells (< 100) have an error when either one of the aggressors is toggled

Some Potential Solutions

- Make better DRAM chips

Cost

- Refresh frequently

Power, Performance

- Sophisticated ECC

Cost, Power

- Access counters

Cost, Power, Complexity

Naive Solutions

1 *Throttle accesses to same row*

- Limit access-interval: $\geq 500\text{ns}$
- Limit number of accesses: $\leq 128\text{K}$ (=64ms/500ns)

2 *Refresh more frequently*

- Shorten refresh-interval by $\sim 7\text{x}$

Both naive solutions introduce significant overhead in performance and power

Apple's Patch for RowHammer

- <https://support.apple.com/en-gb/HT204934>

Available for: OS X Mountain Lion v10.8.5, OS X Mavericks v10.9.5

Impact: A malicious application may induce memory corruption to escalate privileges

Description: A disturbance error, also known as Rowhammer, exists with some DDR3 RAM that could have led to memory corruption. This issue was mitigated by increasing memory refresh rates.

CVE-ID

CVE-2015-3693 : Mark Seaborn and Thomas Dullien of Google, working from original research by Yoongu Kim et al (2014)

HP and Lenovo released similar patches

Our Solution to RowHammer

- PARA: *Probabilistic Adjacent Row Activation*
- Key Idea
 - After closing a row, we activate (i.e., refresh) one of its neighbors with a low probability: $p = 0.005$
- Reliability Guarantee
 - When $p=0.005$, errors in one year: 9.4×10^{-14}
 - By adjusting the value of p , we can vary the strength of protection against errors

Advantages of PARA

- *PARA refreshes rows infrequently*
 - Low power
 - Low performance-overhead
 - Average slowdown: **0.20%** (for 29 benchmarks)
 - Maximum slowdown: **0.75%**
- *PARA is stateless*
 - Low cost
 - Low complexity
- *PARA is an effective and low-overhead solution to prevent disturbance errors*

Requirements for PARA

- If implemented in **DRAM chip**
 - Enough slack in timing parameters
 - Plenty of slack today:
 - Lee et al., “**Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common Case**,” HPCA 2015.
 - Chang et al., “**Understanding Latency Variation in Modern DRAM Chips**,” SIGMETRICS 2016.
 - Lee et al., “**Design-Induced Latency Variation in Modern DRAM Chips**,” SIGMETRICS 2017.
 - Chang et al., “**Understanding Reduced-Voltage Operation in Modern DRAM Devices**,” SIGMETRICS 2017.
- If implemented in **memory controller**
 - Better coordination between memory controller and DRAM
 - Memory controller should know which rows are physically adjacent

More on RowHammer Analysis

- 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 \(pptx\) \(pdf\)\]](#) [\[Source Code and Data\]](#)

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

Yoongu Kim¹ Ross Daly* Jeremie Kim¹ Chris Fallin* Ji Hye Lee¹
Donghyuk Lee¹ Chris Wilkerson² Konrad Lai Onur Mutlu¹

¹Carnegie Mellon University ²Intel Labs

Retrospective on RowHammer & Future

- Onur Mutlu,
"The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"
Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (DATE), Lausanne, Switzerland, March 2017.
[[Slides \(pptx\)](#) ([pdf](#))]

The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
<https://people.inf.ethz.ch/omutlu>

Fundamentally Secure, Reliable, Safe Computing Architectures

Future of Main Memory

- DRAM is becoming less reliable → more vulnerable

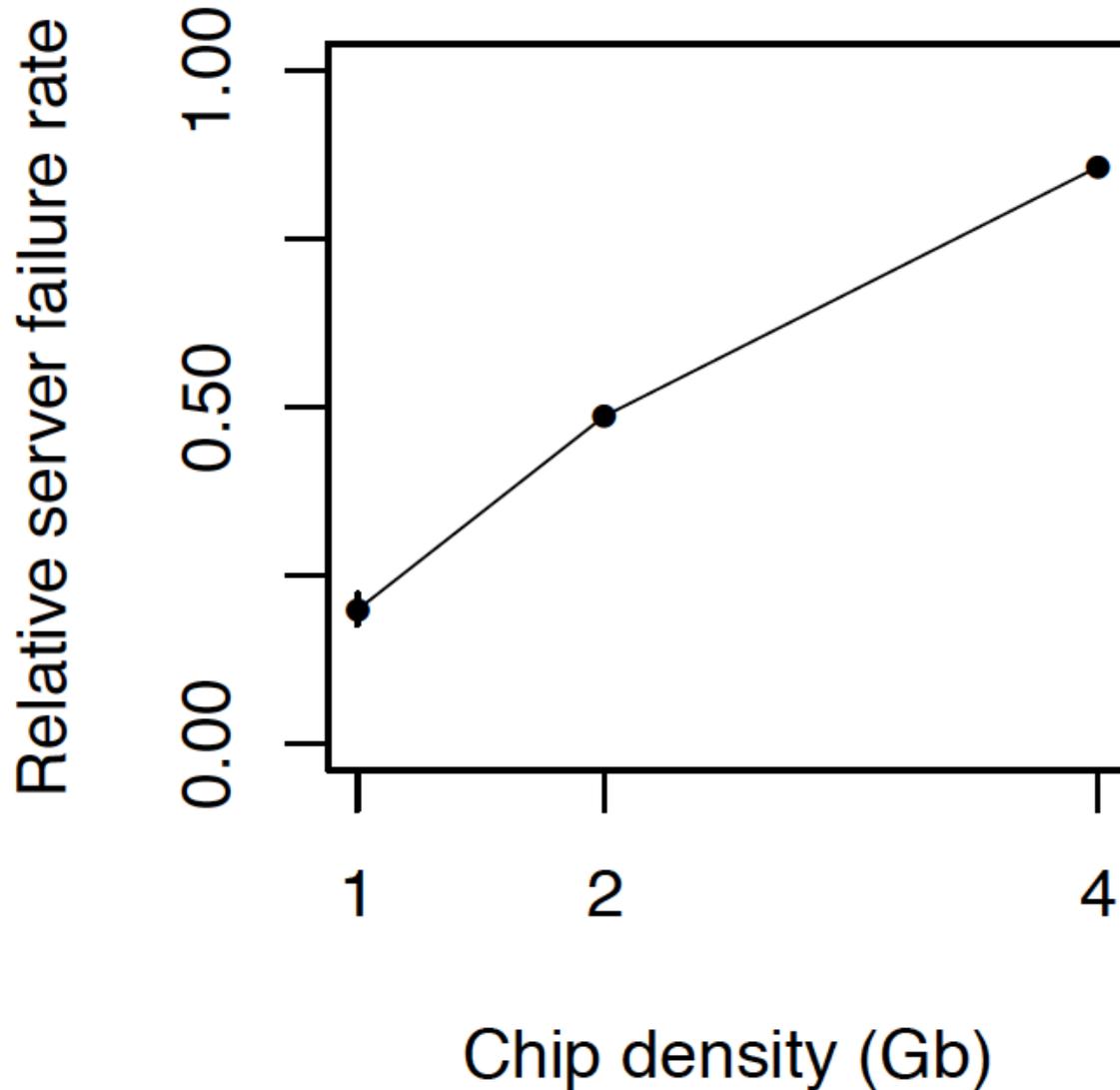
Large-Scale Failure Analysis of DRAM Chips

- Analysis and modeling of memory errors found in all of Facebook's server fleet
- 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](#)]

Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field

Justin Meza Qiang Wu* Sanjeev Kumar* Onur Mutlu
Carnegie Mellon University * Facebook, Inc.

DRAM Reliability Reducing



*Intuition:
quadratic
increase in
capacity*

Aside: SSD Error Analysis in the Field

- First large-scale field study of flash memory errors
- 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.
[[Slides \(pptx\)](#)] [[pdf](#)] [[Coverage at ZDNet](#)]

A Large-Scale Study of Flash Memory Failures in the Field

Justin Meza
Carnegie Mellon University
meza@cmu.edu

Qiang Wu
Facebook, Inc.
qwu@fb.com

Sanjeev Kumar
Facebook, Inc.
skumar@fb.com

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu

Future of Main Memory

- DRAM is becoming less reliable → more vulnerable
- Due to difficulties in DRAM scaling, other problems may also appear (or they may be going unnoticed)
- Some errors may already be slipping into the field
 - Read disturb errors (Rowhammer)
 - Retention errors
 - Read errors, write errors
 - ...
- These errors can also pose security vulnerabilities

DRAM Data Retention Time Failures

- Determining the data retention time of a cell/row is getting more difficult
- Retention failures may already be slipping into the field

Analysis of Retention Failures [ISCA'13]

- Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu,
"An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms"
Proceedings of the 40th International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013. [Slides \(ppt\)](#) [Slides \(pdf\)](#)

An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms

Jamie Liu^{*}

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
jamiel@alumni.cmu.edu

Ben Jaiyen^{*}

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
bjaiyen@alumni.cmu.edu

Yoongu Kim

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
yoonguk@ece.cmu.edu

Chris Wilkerson

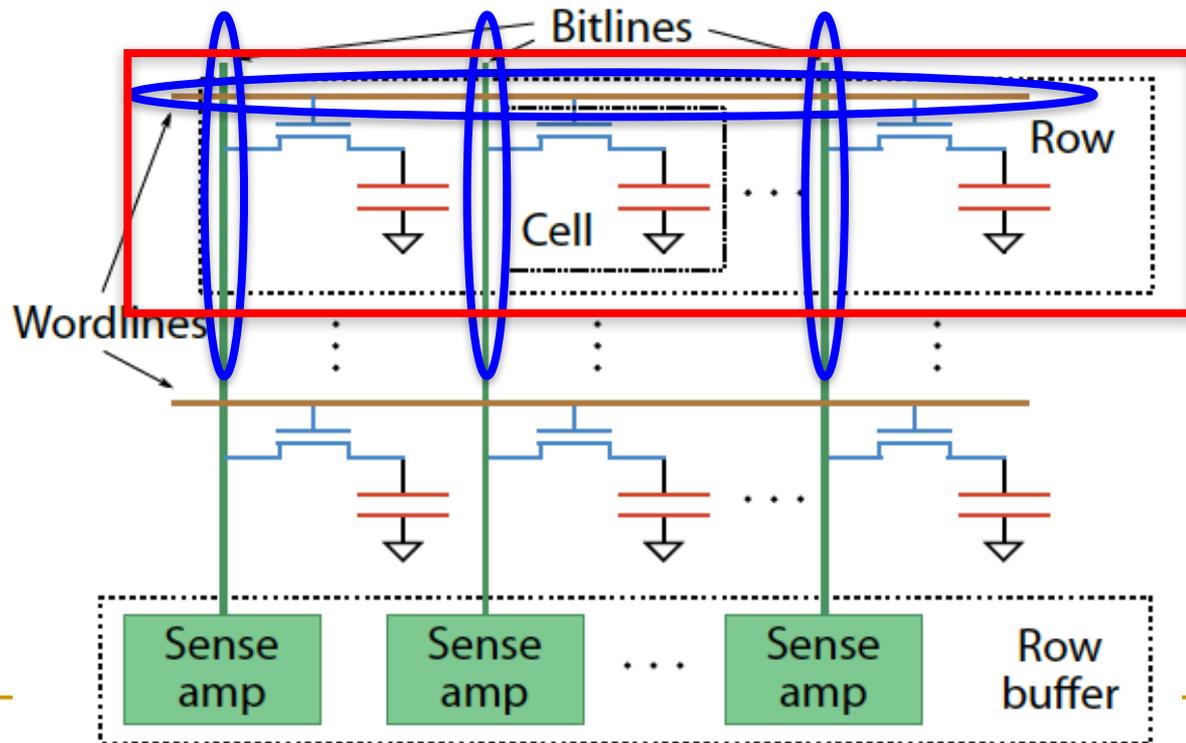
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
chris.wilkerson@intel.com

Onur Mutlu

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
onur@cmu.edu

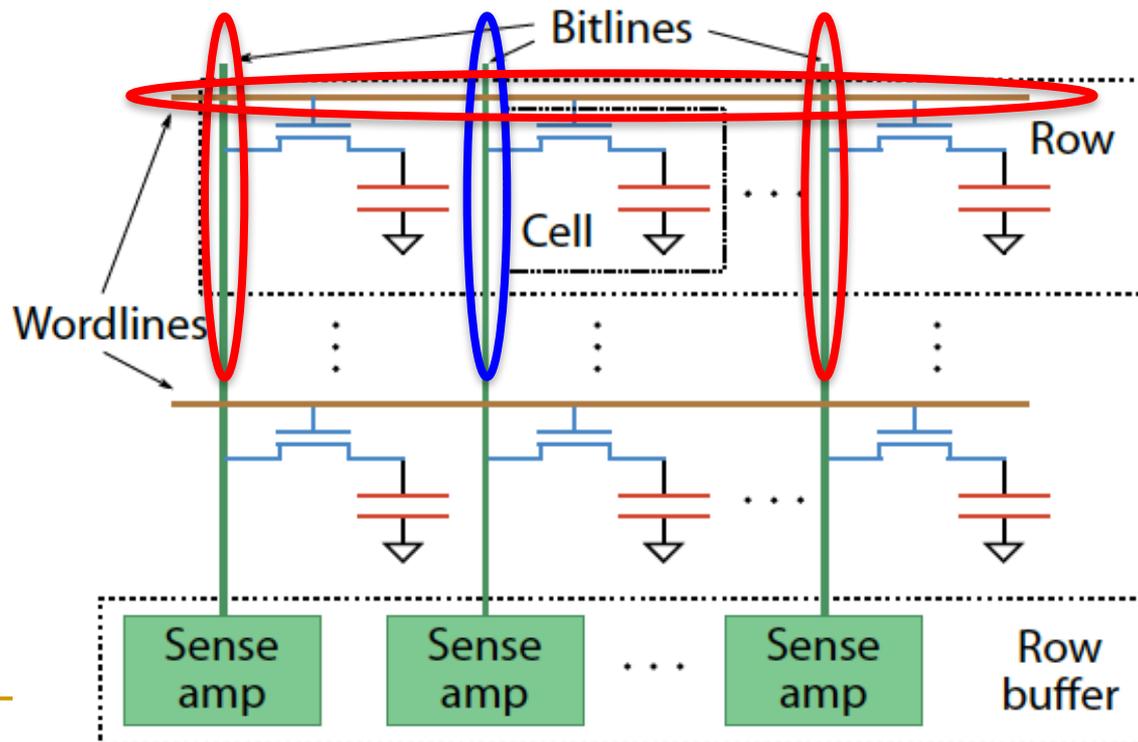
Two Challenges to Retention Time Profiling

- **Challenge 1: Data Pattern Dependence (DPD)**
 - Retention time of a DRAM cell depends on its value and the values of cells nearby it
 - When a row is activated, all bitlines are perturbed simultaneously



Data Pattern Dependence

- Electrical noise on the bitline affects reliable sensing of a DRAM cell
- The magnitude of this noise is affected by values of nearby cells via
 - Bitline-bitline coupling → electrical coupling between adjacent bitlines
 - Bitline-wordline coupling → electrical coupling between each bitline and the activated wordline



Data Pattern Dependence

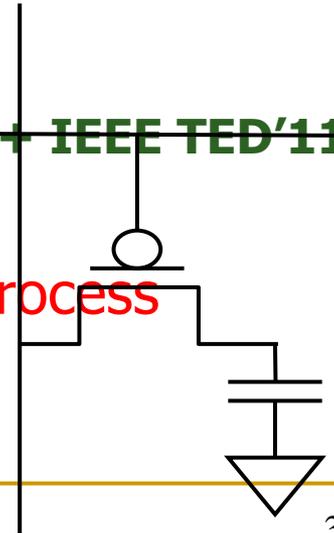
- Electrical noise on the bitline affects reliable sensing of a DRAM cell
- The magnitude of this noise is affected by values of nearby cells via
 - Bitline-bitline coupling → electrical coupling between adjacent bitlines
 - Bitline-wordline coupling → electrical coupling between each bitline and the activated wordline

- Retention time of a cell depends on data patterns stored in nearby cells
 - need to find the worst data pattern to find worst-case retention time
 - this pattern is location dependent

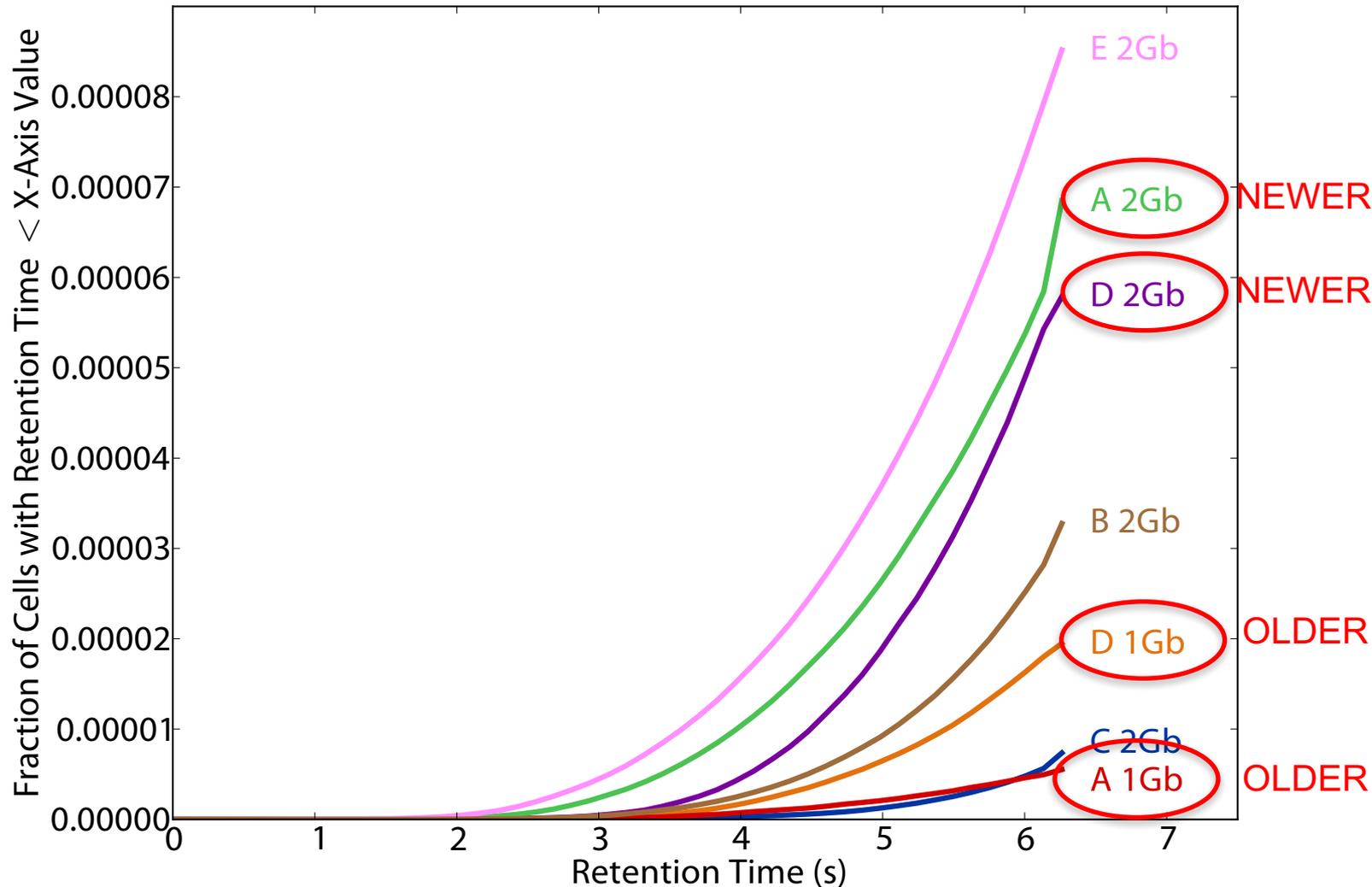
Two Challenges to Retention Time Profiling

■ Challenge 2: Variable Retention Time (VRT)

- Retention time of a DRAM cell changes randomly over time
 - a cell alternates between multiple retention time states
- Leakage current of a cell changes sporadically due to a charge trap in the gate oxide of the DRAM cell access transistor
- When the trap becomes occupied, charge leaks more readily from the transistor's drain, leading to a short retention time
 - Called *Trap-Assisted Gate-Induced Drain Leakage*
- This process appears to be a random process [Kim+ IEEE TED'11]
- Worst-case retention time depends on a random process
→ need to find the worst case despite this

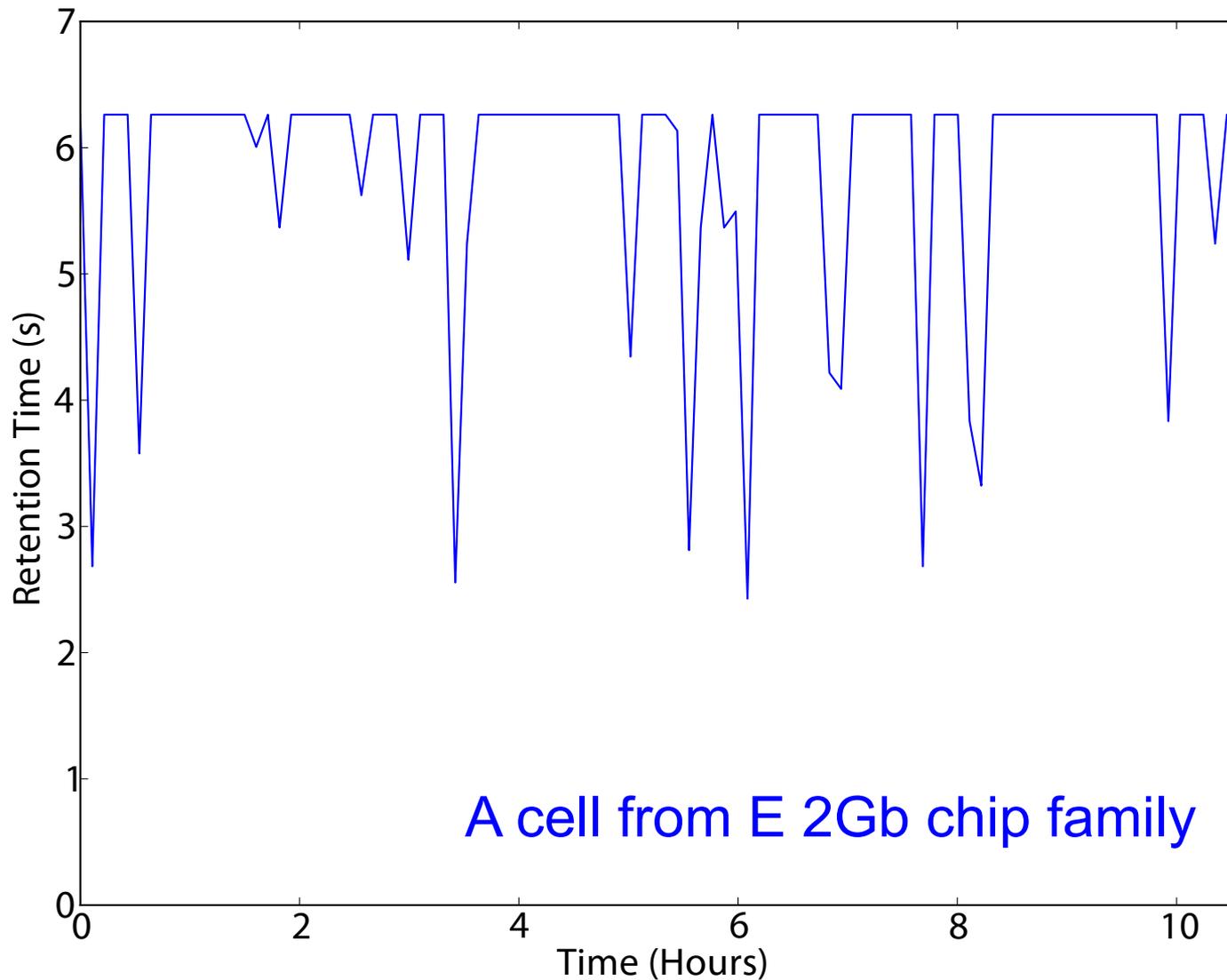


Modern DRAM Retention Time Distribution



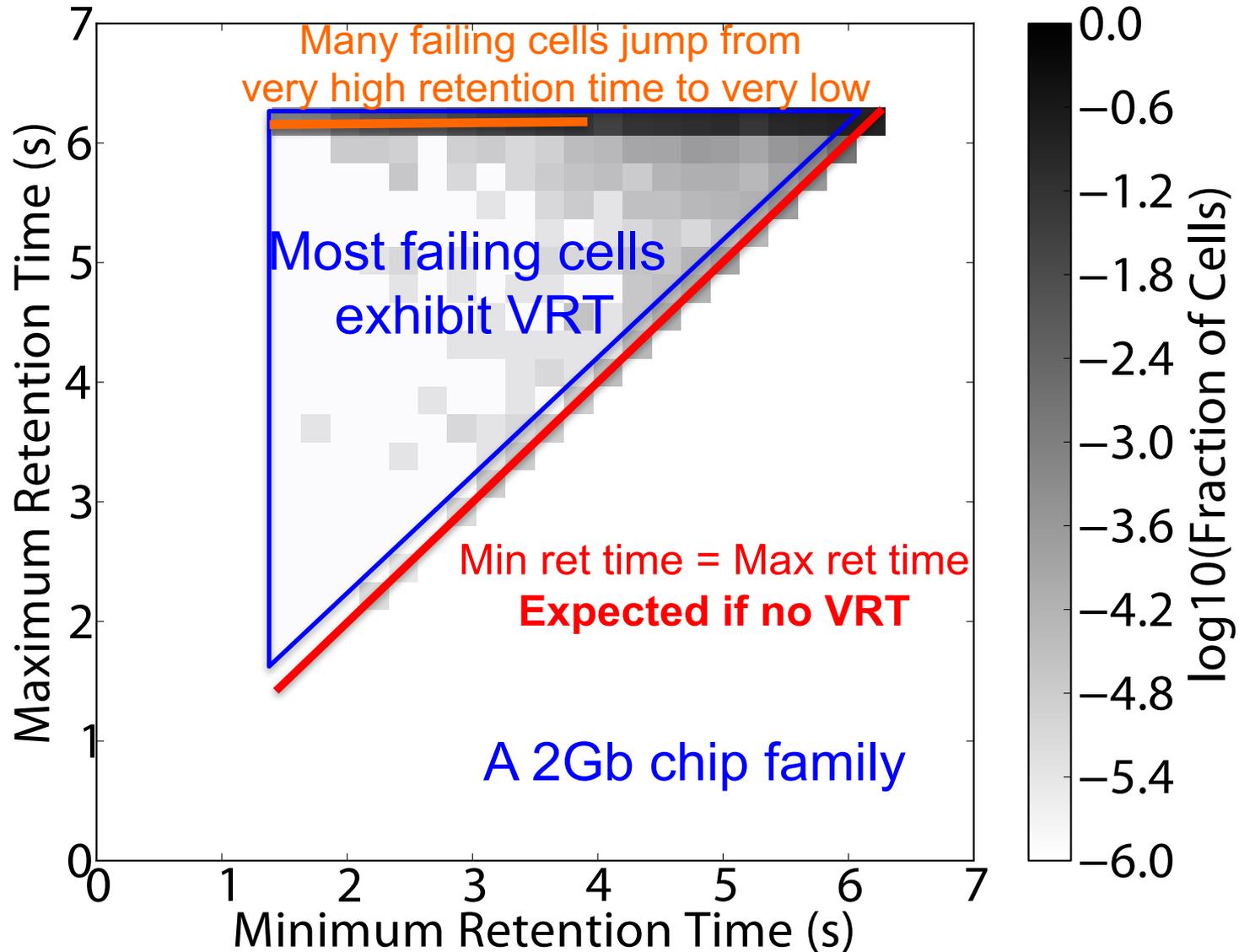
**Newer device families have more weak cells than older ones
Likely a result of technology scaling**

An Example VRT Cell



A cell from E 2Gb chip family

Variable Retention Time



More on DRAM Retention Analysis

- Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu,
"An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms"
Proceedings of the 40th International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013. [Slides \(ppt\)](#) [Slides \(pdf\)](#)

An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms

Jamie Liu*

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
jamiel@alumni.cmu.edu

Ben Jaiyen*

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
bjaiyen@alumni.cmu.edu

Yoongu Kim

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
yoonguk@ece.cmu.edu

Chris Wilkerson

Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
chris.wilkerson@intel.com

Onur Mutlu

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
onur@cmu.edu

Industry Is Writing Papers About It, Too

DRAM Process Scaling Challenges

❖ Refresh

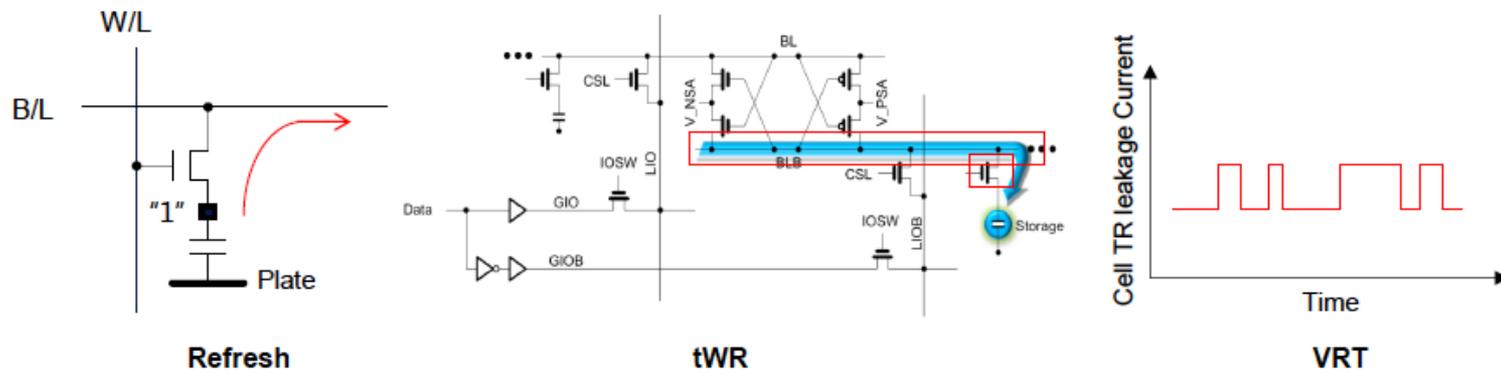
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance
- Leakage current of cell access transistors increasing

❖ t_{WR}

- Contact resistance between the cell capacitor and access transistor increasing
- On-current of the cell access transistor decreasing
- Bit-line resistance increasing

❖ VRT

- Occurring more frequently with cell capacitance decreasing



Industry Is Writing Papers About It, Too

DRAM Process Scaling Challenges

❖ Refresh

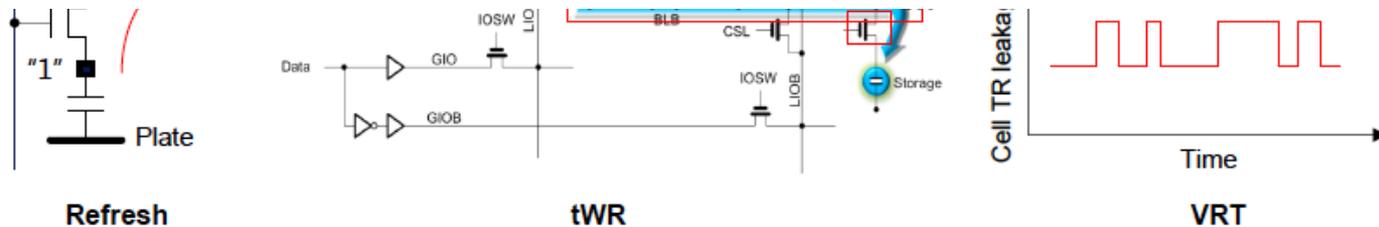
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

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*



Mitigation of Retention Issues [SIGMETRICS'14]

- Samira Khan, Donghyuk Lee, Yoongu Kim, Alaa Alameldeen, Chris Wilkerson, and Onur Mutlu,
"The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study"
Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), Austin, TX, June 2014. [[Slides \(pptx\)](#)] [[pdf](#)] [[Poster \(pptx\)](#)] [[pdf](#)] [[Full data sets](#)]

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

Handling Data-Dependent Failures [DSN'16]

- Samira Khan, Donghyuk Lee, Yoongu Kim, Alaa Alameldeen, Chris Wilkerson, and Onur Mutlu,
"The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study"
Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), Austin, TX, June 2014. [[Slides \(pptx\)](#)] [[pdf](#)] [[Poster \(pptx\)](#)] [[pdf](#)] [[Full data sets](#)]

PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM

Samira Khan^{*}

^{*}University of Virginia

Donghyuk Lee^{†‡}

[†]Carnegie Mellon University

Onur Mutlu^{*†}

[‡]Nvidia

^{*}ETH Zürich

Handling Data-Dependent Failures [CAL'16]

- Samira Khan, Chris Wilkerson, Donghyuk Lee, Alaa R. Alameldeen, and Onur Mutlu,
"A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM"
IEEE Computer Architecture Letters (**CAL**), November 2016.

A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM

Samira Khan^{*}, Chris Wilkerson[†], Donghyuk Lee[‡], Alaa R. Alameldeen[†], Onur Mutlu^{*‡}

^{*}University of Virginia [†]Intel Labs [‡]Carnegie Mellon University ^{*}ETH Zürich

Handling Variable Retention Time [DSN'15]

- Moinuddin Qureshi, Dae Hyun Kim, Samira Khan, Prashant Nair, and Onur Mutlu,
"AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems"
Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, June 2015.
[\[Slides \(pptx\) \(pdf\)\]](#)

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems

Moinuddin K. Qureshi[†] Dae-Hyun Kim[†] Samira Khan[‡] Prashant J. Nair[†] Onur Mutlu[‡]
[†]Georgia Institute of Technology [‡]Carnegie Mellon University
{moin, dhkim, pnair6}@ece.gatech.edu {samirakhan, onur}@cmu.edu

Handling Both DPD and VRT [ISCA'17]

- Minesh Patel, Jeremie S. Kim, and Onur Mutlu,
"The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions"
Proceedings of the 44th International Symposium on Computer Architecture (ISCA), Toronto, Canada, June 2017.
- First experimental analysis of (mobile) LPDDR4 chips
- Analyzes the complex tradeoff space of retention time profiling
- Key idea: enable fast and robust profiling at higher refresh intervals & temp.

The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions

Minesh Patel^{§‡} Jeremie S. Kim^{‡§} Onur Mutlu^{§‡}
§ETH Zürich ‡Carnegie Mellon University

If Time Permits: NAND Flash Vulnerabilities

- Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu, **"Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives"**

to appear in Proceedings of the IEEE, 2017.

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.

Meza+, "A Large-Scale Study of Flash Memory Errors in the Field," SIGMETRICS 2015.

Luo+, "Enabling Accurate and Practical Online Flash Channel Modeling for Modern MLC NAND Flash Memory," IEEE JSAC 2016.

Cai+, "Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques," HPCA 2017.

Fukami+, "Improving the Reliability of Chip-Off Forensic Analysis of NAND Flash Memory Devices," DFRWS EU 2017.

Cai+, "Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives," Proc. IEEE 2017.

Overview Paper on Flash Reliability

- Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu,
"Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives"
to appear in Proceedings of the IEEE, 2017.

Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives

Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu

NAND Flash Memory

Reliability and Security



Proceedings of the IEEE, Sept. 2017



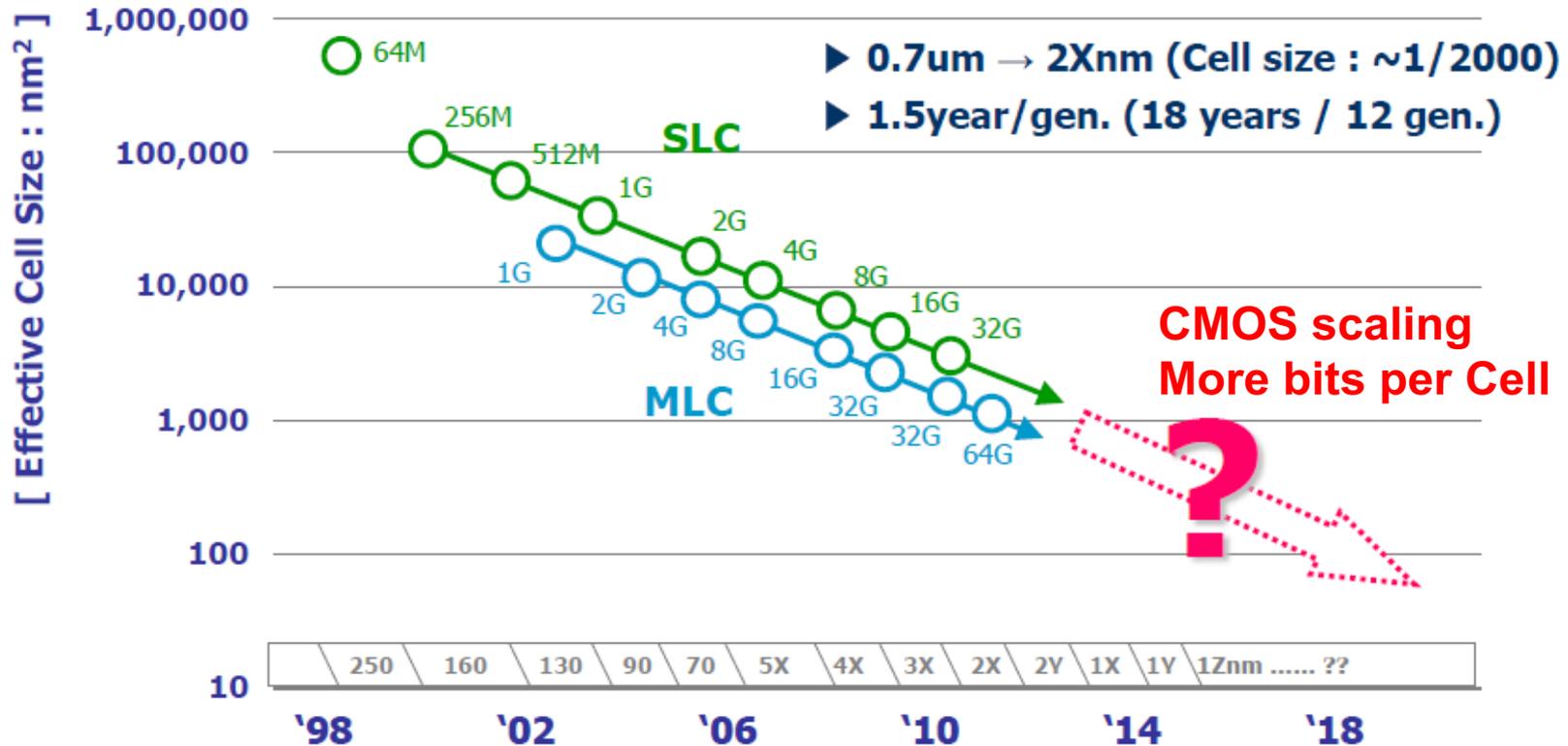
Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD's reliability and lifetime.

By YU CAI, SAUGATA GHOSE, ERICH F. HARATSCH, YIXIN LUO, AND ONUR MUTLU

<https://arxiv.org/pdf/1706.08642>

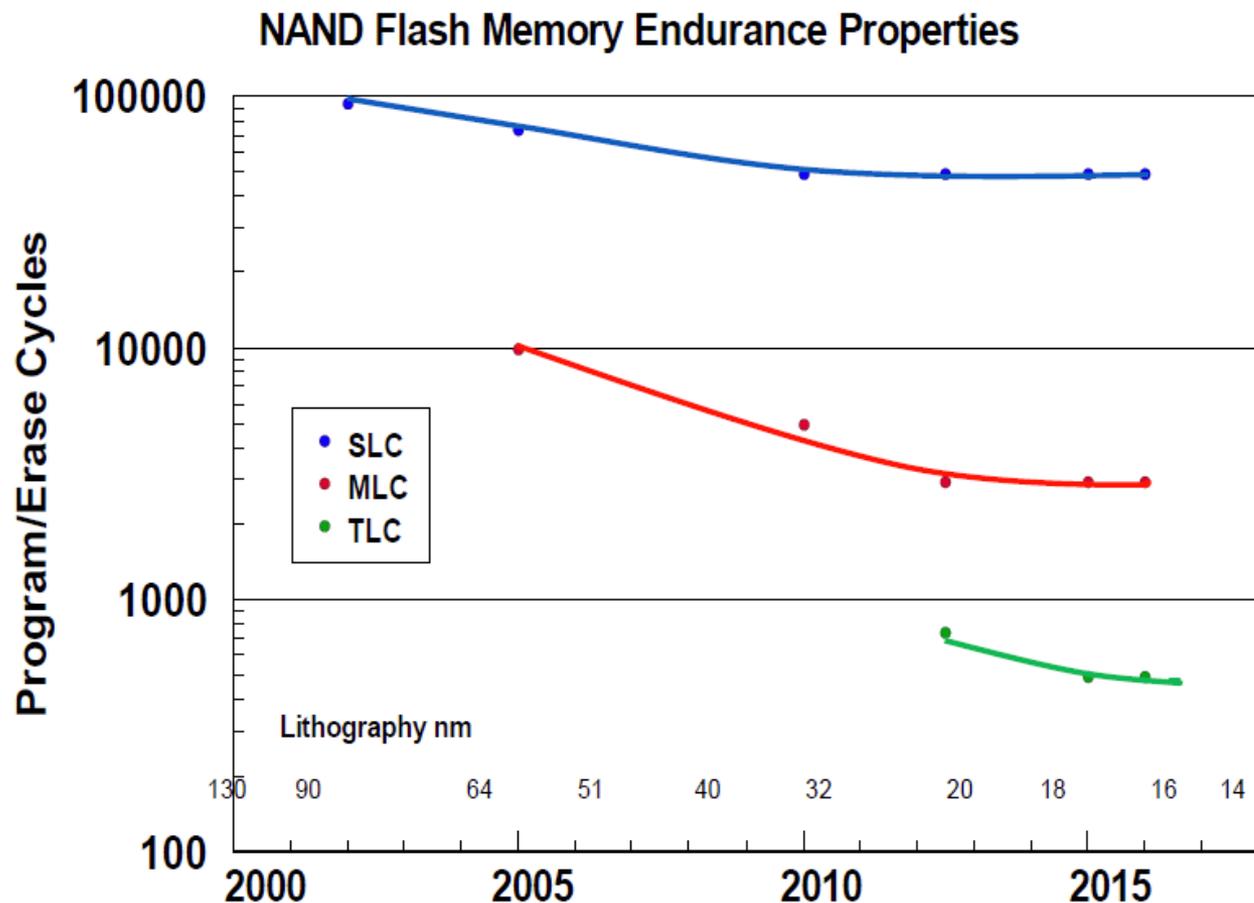
Evolution of NAND Flash Memory



Seaung Suk Lee, "Emerging Challenges in NAND Flash Technology", Flash Summit 2011 (Hynix)

- Flash memory is widening its range of applications
 - Portable consumer devices, laptop PCs and enterprise servers

Flash Challenges: Reliability and Endurance



- **P/E cycles (provided)**

A few thousand

- **P/E cycles (required)**

Writing the full capacity of the drive
10 times per day
for 5 years
(STEC)

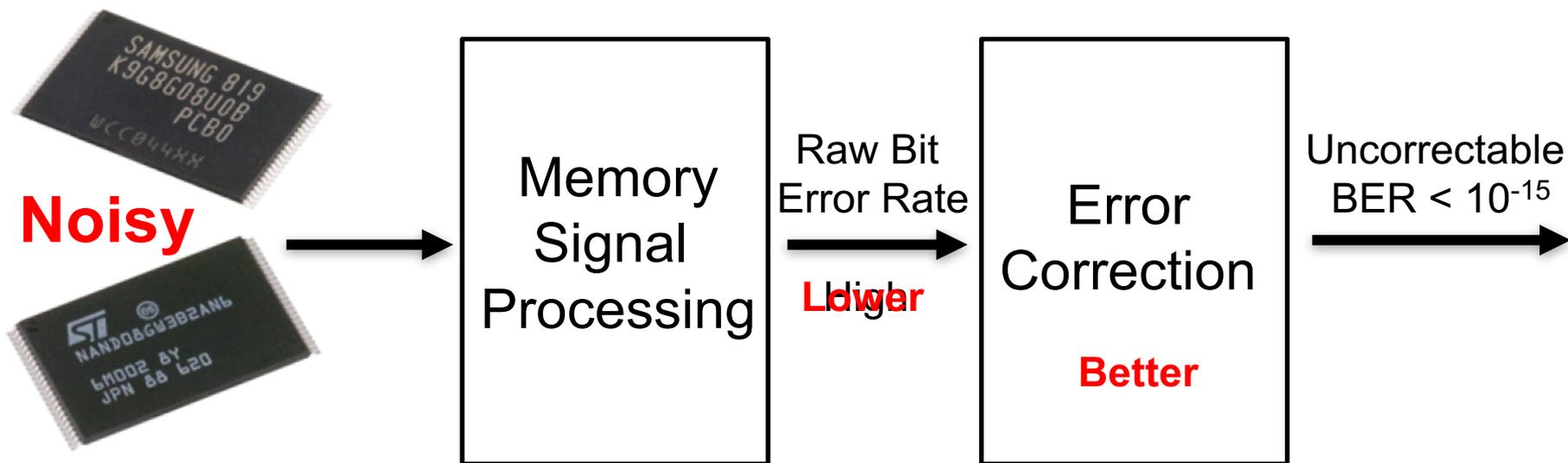
> 50k P/E cycles

E. Grochowski et al., "Future technology challenges for NAND flash and HDD products", Flash Memory Summit 2012

NAND Flash Memory is Increasingly Noisy



Future NAND Flash-based Storage Architecture



Our Goals:

Build reliable error models for NAND flash memory

Design efficient reliability mechanisms based on the model

NAND Flash Error Model



Experimentally characterize and model dominant errors

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

Luo et al., "Enabling Accurate and Practical Online Flash Channel Modeling for Modern MLC NAND Flash Memory", **JSAC 2016**



Cai et al., "Threshold voltage distribution in MLC NAND Flash Memory: Characterization, Analysis, and Modeling", **DATE 2013**

Cai et al., "Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques", **HPCA 2017**

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

Cai et al., "Neighbor-Cell Assisted Error Correction in MLC NAND Flash Memories", **SIGMETRICS 2014**

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

Cai et al., "Flash Correct-and-Refresh: Retention-aware error management for increased flash memory lifetime", **ICCD 2012**

Cai et al., "Error Analysis and Retention-Aware Error Management for NAND Flash Memory", **ITJ 2013**

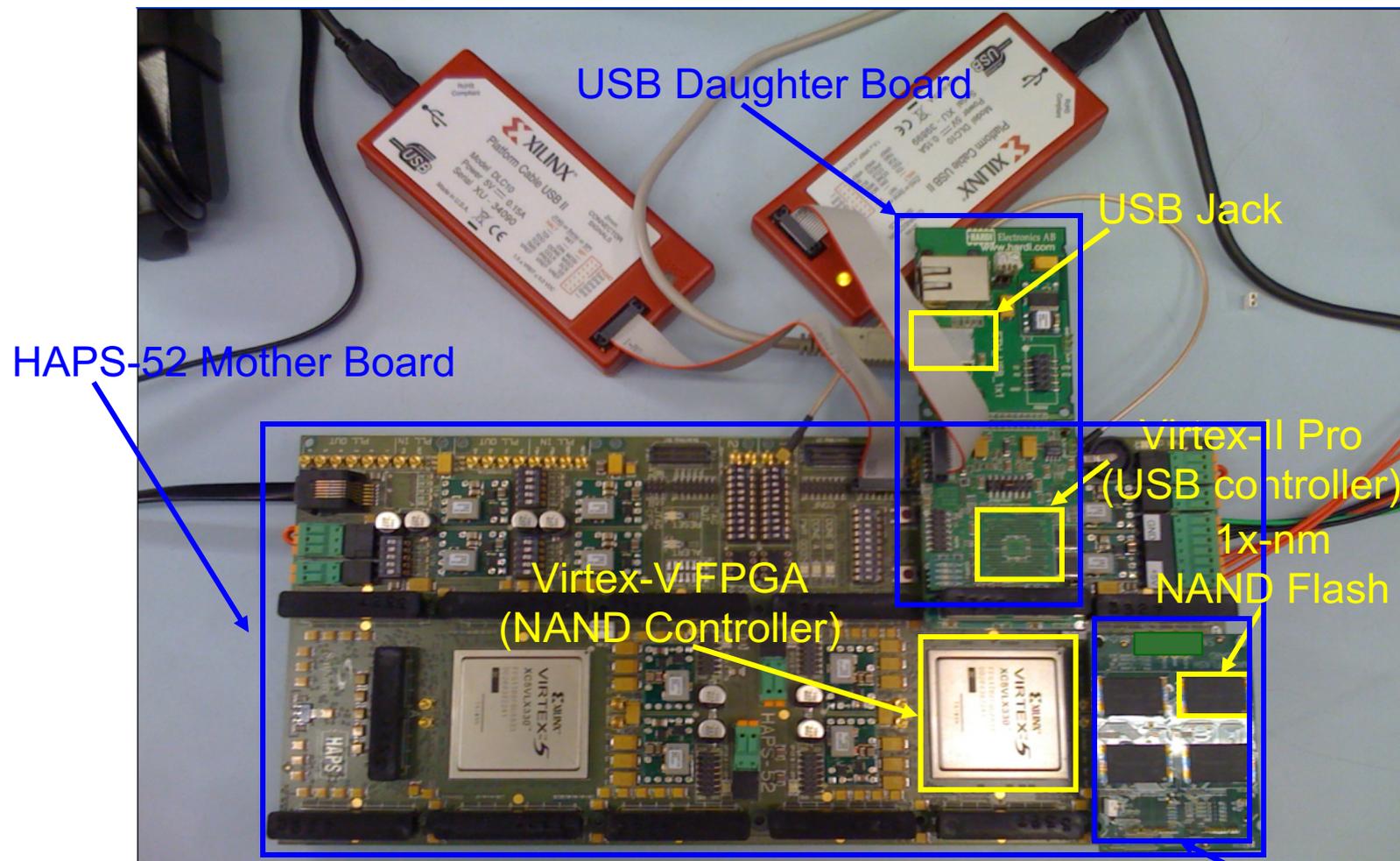
Cai et al., "Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery", **HPCA 2015**

Our Goals and Approach

- Goals:
 - Understand error mechanisms and develop reliable predictive models for MLC NAND flash memory errors
 - Develop efficient error management techniques to mitigate errors and improve flash reliability and endurance

- Approach:
 - Solid experimental analyses of errors in real MLC NAND flash memory → drive the understanding and models
 - Understanding, models, and creativity → drive the new techniques

Experimental Testing Platform



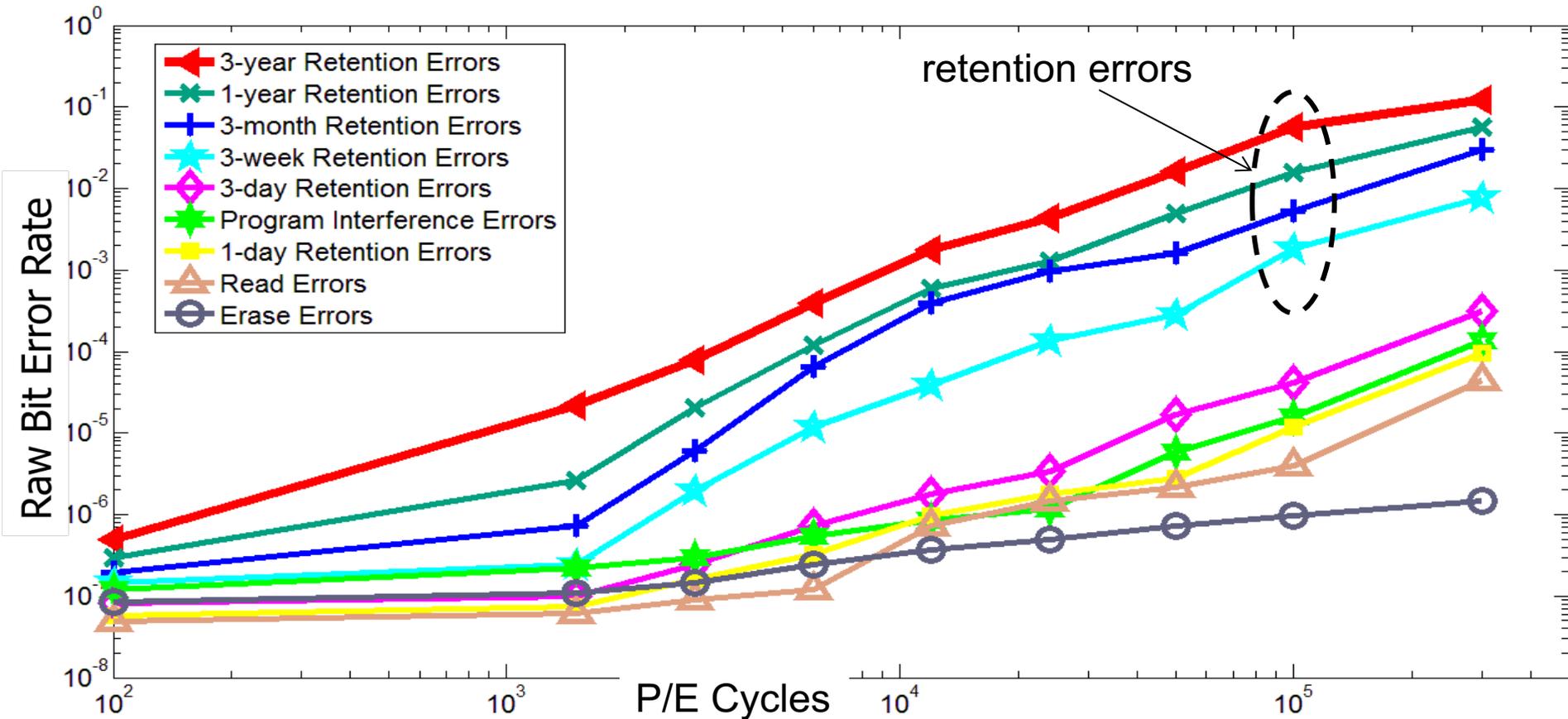
[DATE 2012, ICCD 2012, DATE 2013, ITJ 2013, ICCD 2013, SIGMETRICS 2014, HPCA 2015, DSN 2015, MSST 2015, JSAC 2016, HPCA 2017, DFRWS 2017]

NAND Daughter Board

NAND Flash Error Types

- Four types of errors [Cai+, DATE 2012]
- Caused by **common flash operations**
 - **Read** errors
 - **Erase** errors
 - **Program** (interference) errors
- Caused by flash **cell losing charge over time**
 - **Retention** errors
 - Whether an error happens depends on required retention time
 - Especially problematic in MLC flash because threshold voltage window to determine stored value is smaller

Observations: Flash Error Analysis



- Raw bit error rate increases exponentially with P/E cycles
- Retention errors are dominant (>99% for 1-year ret. time)
- Retention errors increase with retention time requirement

More on Flash Error Analysis

- Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai, **"Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis"** *Proceedings of the Design, Automation, and Test in Europe Conference (DATE)*, Dresden, Germany, March 2012. [Slides \(ppt\)](#)

Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis

Yu Cai¹, Erich F. Haratsch², Onur Mutlu¹ and Ken Mai¹

¹Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA

²LSI Corporation, 1110 American Parkway NE, Allentown, PA

¹{yucai, onur, kenmai}@andrew.cmu.edu, ²erich.haratsch@lsi.com

Solution to Retention Errors

- Refresh periodically
- Change the period based on P/E cycle wearout
 - Refresh more often at higher P/E cycles
- Use a combination of **in-place** and **remapping-based** refresh

Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime

Yu Cai¹, Gulay Yalcin², Onur Mutlu¹, Erich F. Haratsch³, Adrian Cristal², Osman S. Unsal² and Ken Mai¹

¹DSSC, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA

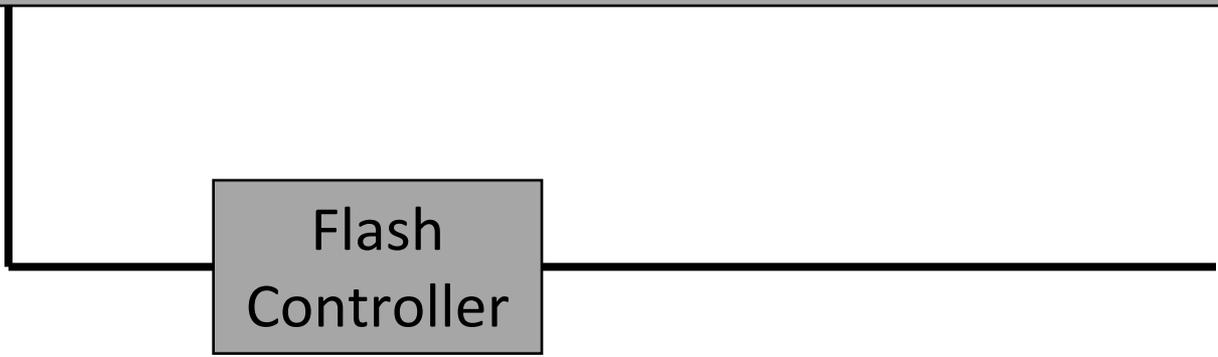
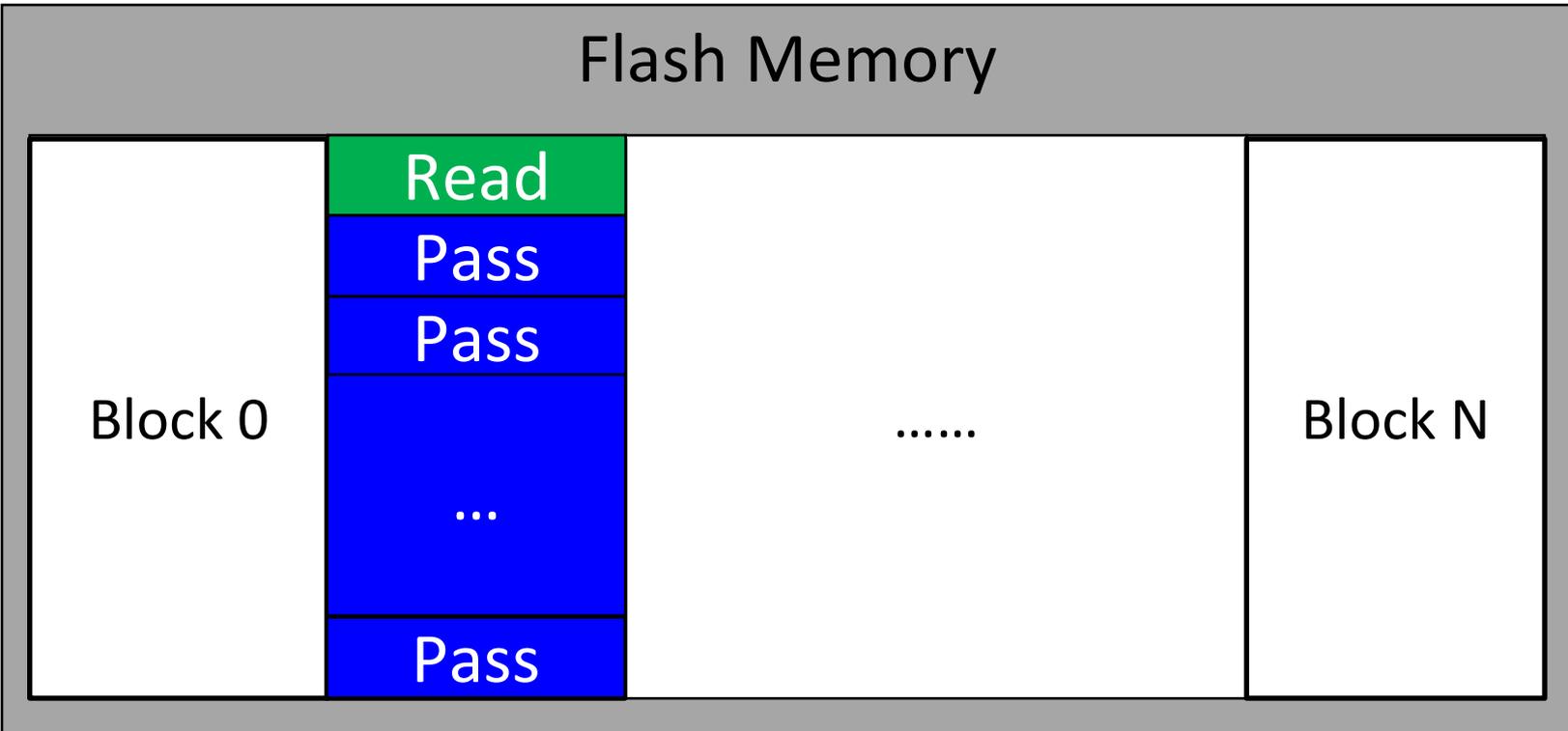
²Barcelona Supercomputing Center, C/Jordi Girona 29, Barcelona, Spain

³LSI Corporation, 1110 American Parkway NE, Allentown, PA

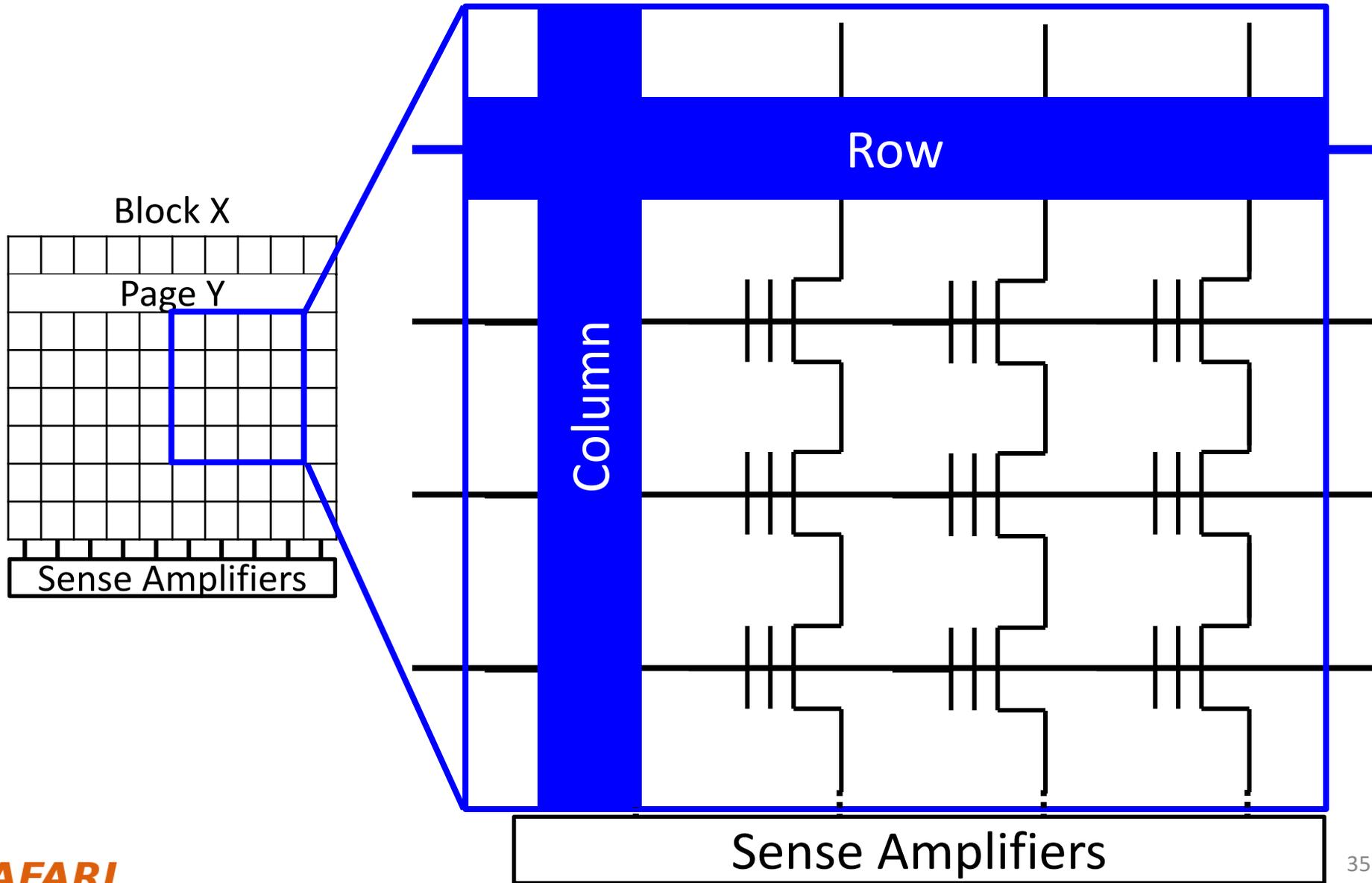
One Issue: Read Disturb in Flash Memory

- All scaled memories are prone to read disturb errors

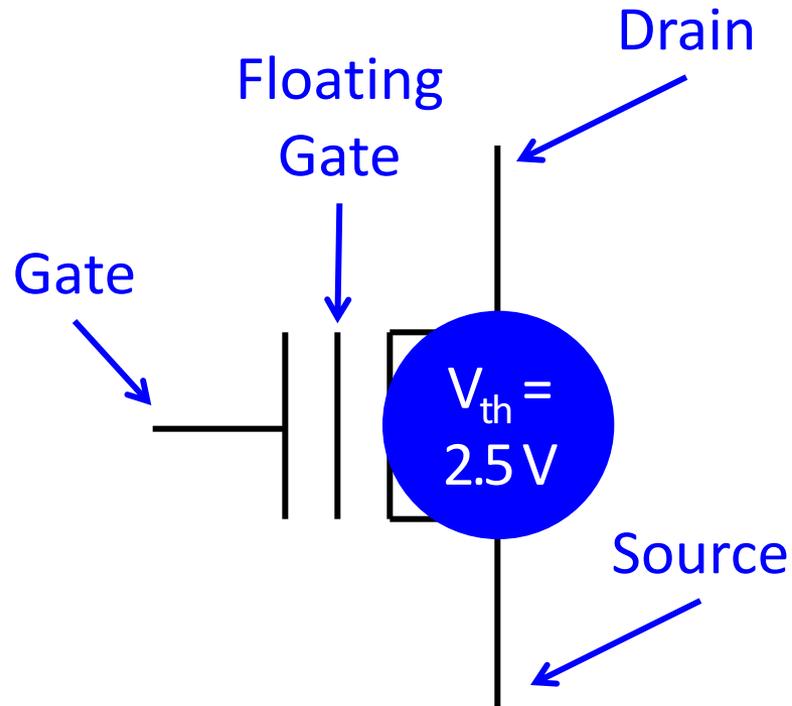
NAND Flash Memory Background



Flash Cell Array

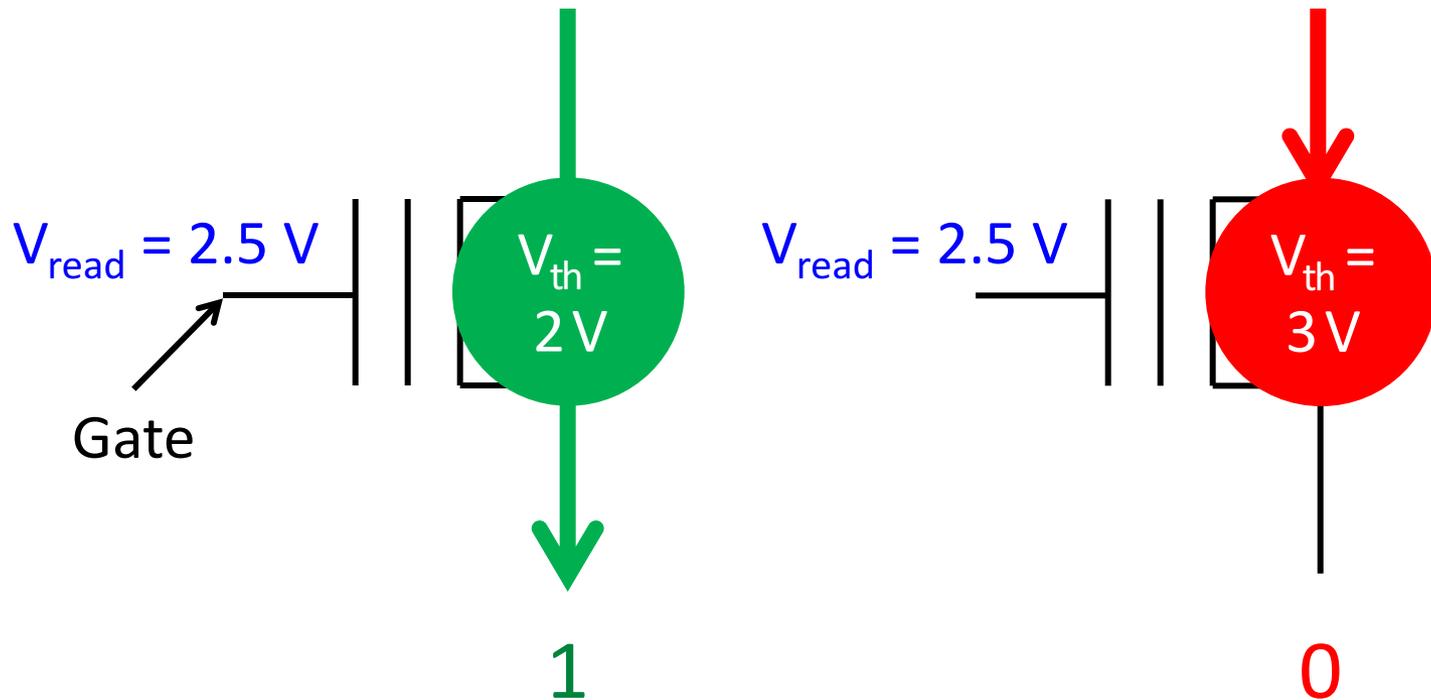


Flash Cell

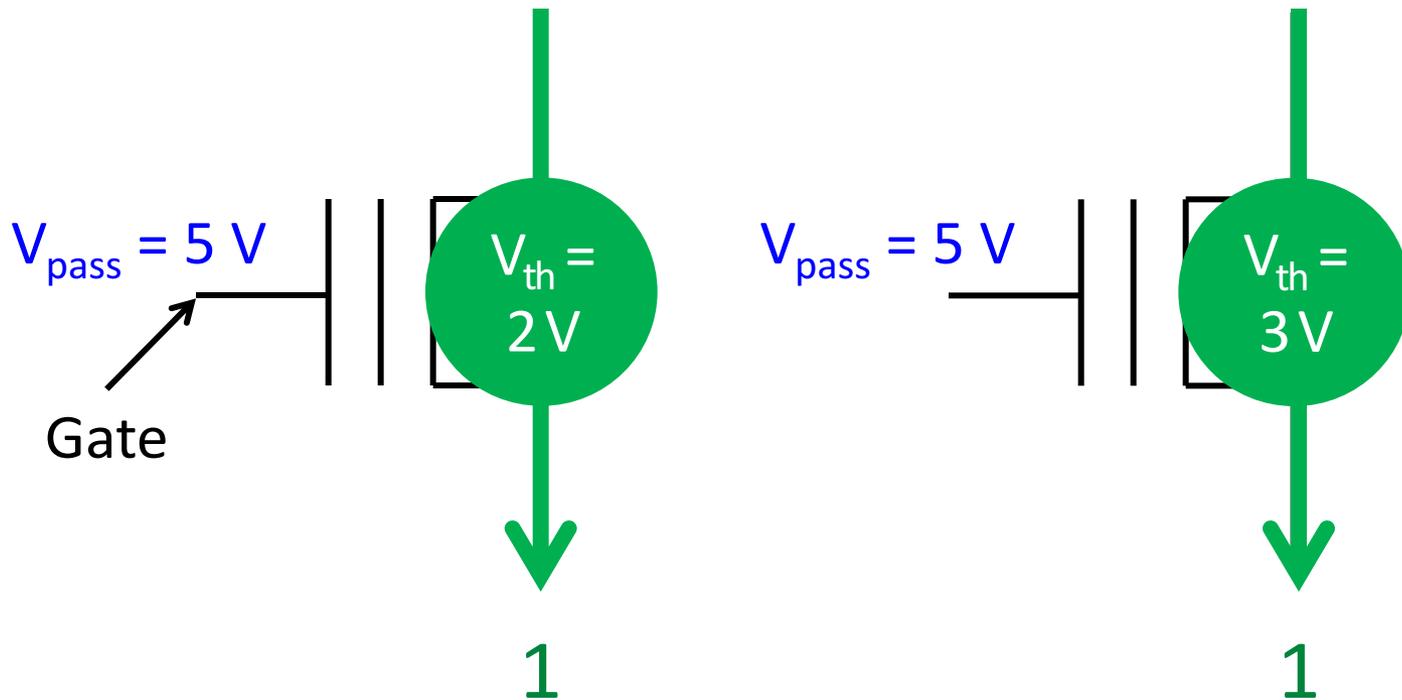


Floating Gate Transistor
(Flash Cell)

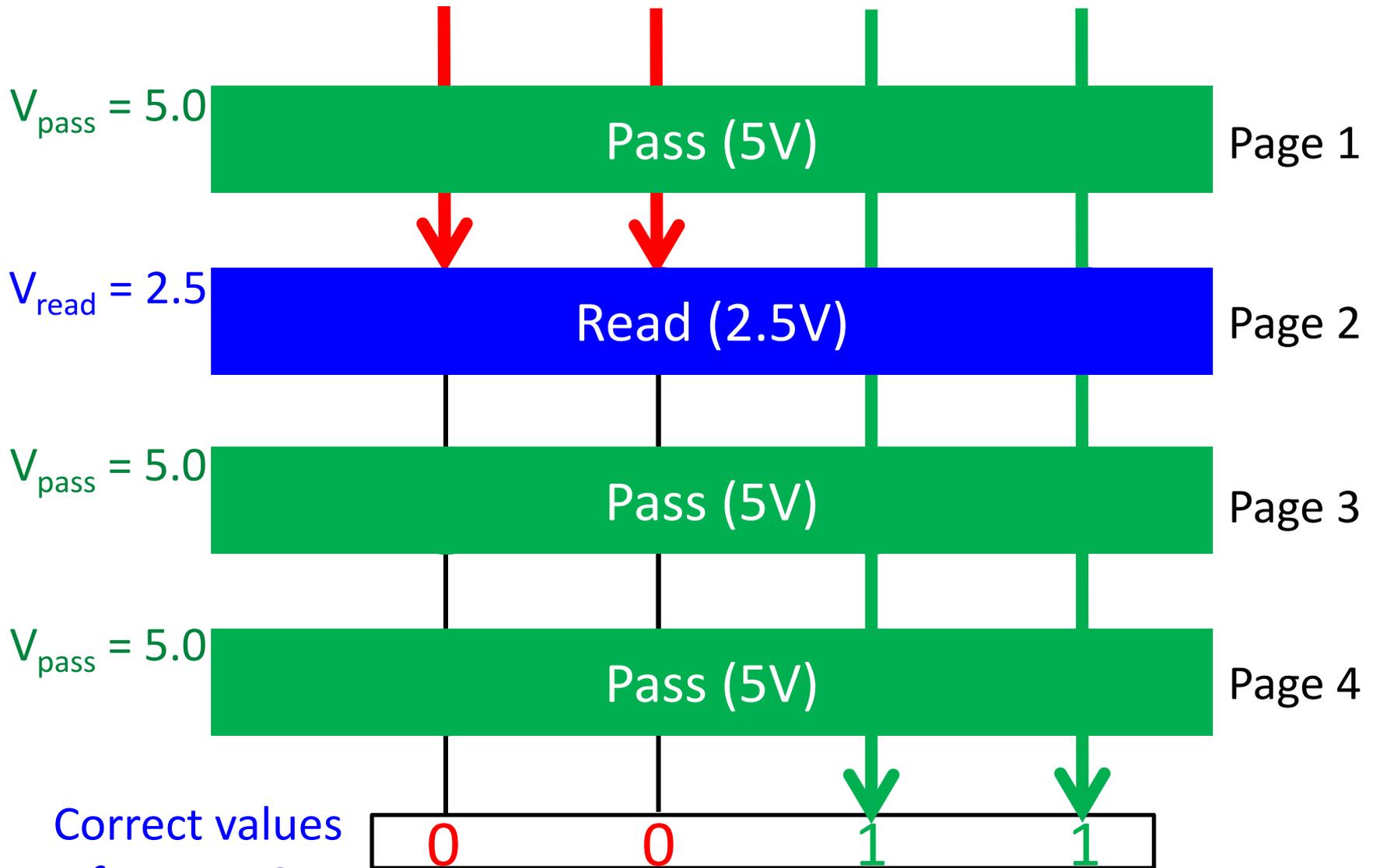
Flash Read



Flash Pass-Through

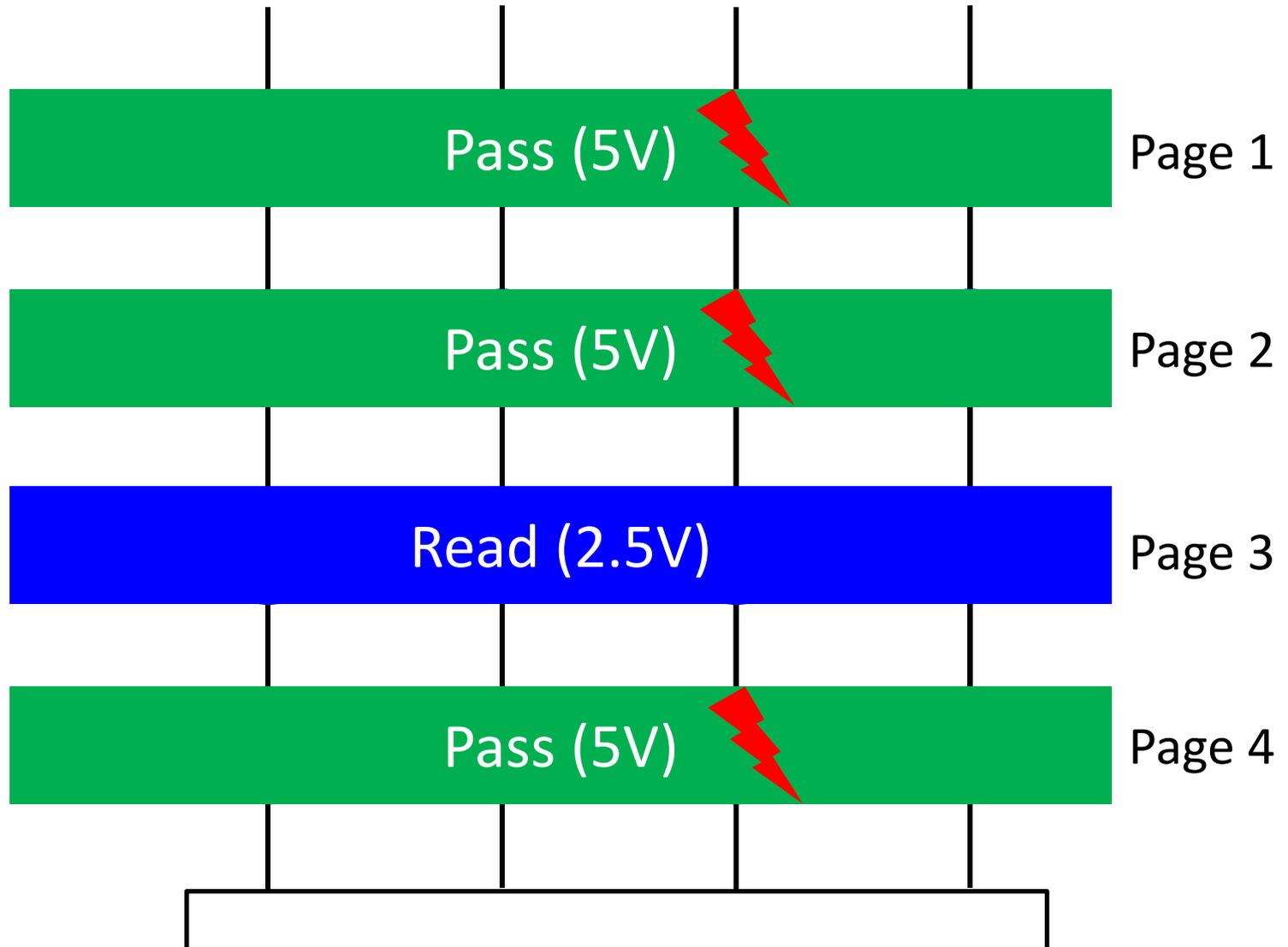


Read from Flash Cell Array



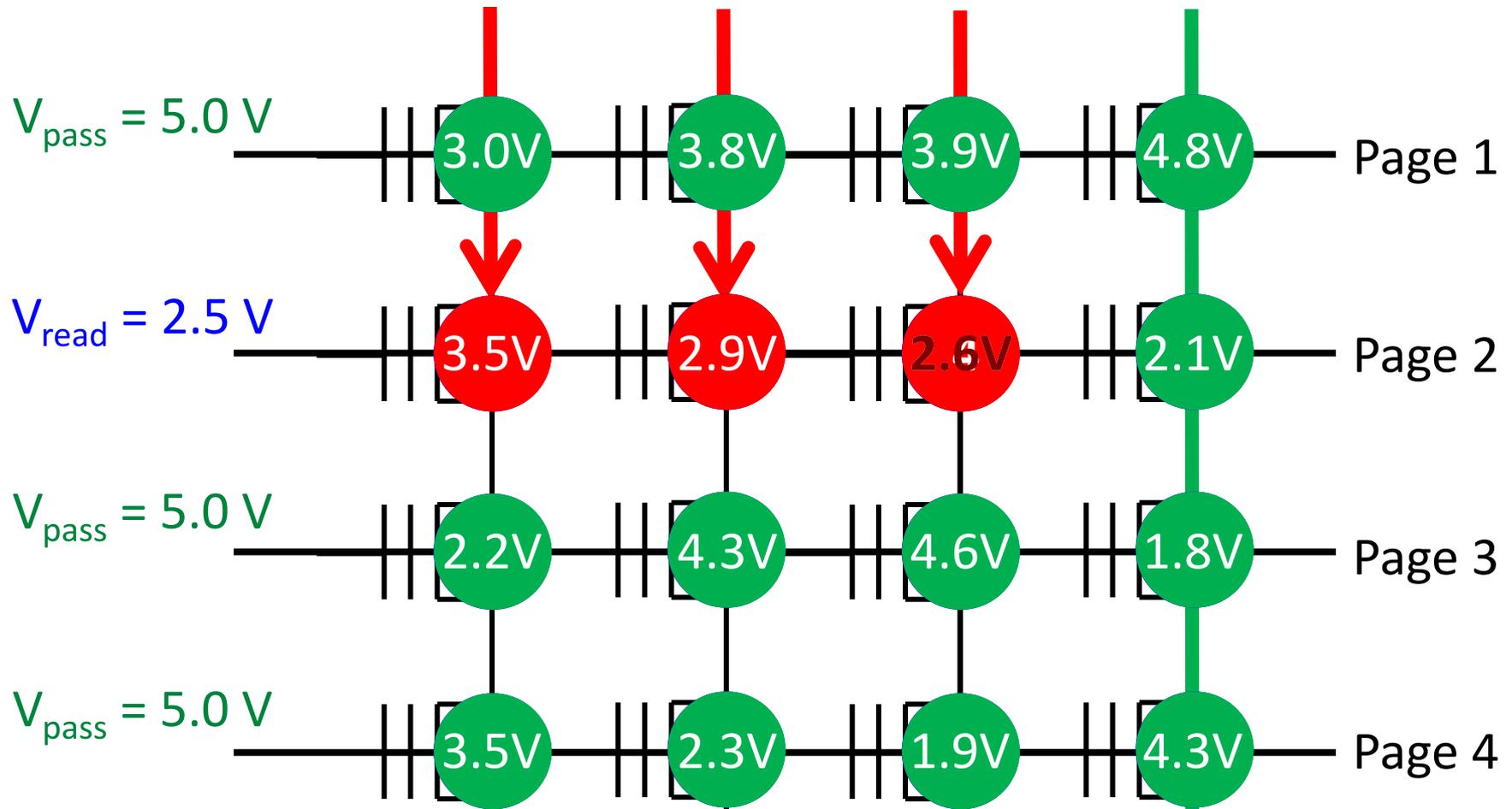
Correct values
for page 2:

Read Disturb Problem: “Weak Programming” Effect



SAFARI Repeatedly read page 3 (or any page other than page 2)

Read Disturb Problem: "Weak Programming" Effect



Incorrect values

from page 2:



High pass-through voltage induces "weak-programming" effect

Executive Summary



- **Read disturb errors** limit flash memory lifetime today
 - Apply a *high pass-through voltage* (V_{pass}) to multiple pages on a read
 - Repeated application of V_{pass} can alter stored values in unread pages
- We **characterize read disturb** on real NAND flash chips
 - Slightly lowering V_{pass} greatly reduces read disturb errors
 - Some flash cells are more prone to read disturb
- **Technique 1: Mitigate** read disturb errors online
 - V_{pass} **Tuning** dynamically finds and applies a lowered V_{pass} per block
 - Flash memory **lifetime improves by 21%**
- **Technique 2: Recover** after failure to prevent data loss
 - **Read Disturb Oriented Error Recovery** (RDR) selectively corrects cells more susceptible to read disturb errors
 - **Reduces raw bit error rate (RBER) by up to 36%**

More on Flash Read Disturb Errors

- Yu Cai, Yixin Luo, Saugata Ghose, Erich F. Haratsch, Ken Mai, and Onur Mutlu,
"Read Disturb Errors in MLC NAND Flash Memory: Characterization and Mitigation"
Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, June 2015.

Read Disturb Errors in MLC NAND Flash Memory: Characterization, Mitigation, and Recovery

Yu Cai, Yixin Luo, Saugata Ghose, Erich F. Haratsch*, Ken Mai, Onur Mutlu
Carnegie Mellon University, *Seagate Technology
yucaicai@gmail.com, {yixinluo, ghose, kenmai, onur}@cmu.edu

Large-Scale Flash SSD Error Analysis

- First large-scale field study of flash memory errors
- 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.*
[[Slides \(pptx\)](#)] [[pdf](#)] [[Coverage at ZDNet](#)] [[Coverage on The Register](#)]
[[Coverage on TechSpot](#)] [[Coverage on The Tech Report](#)]

A Large-Scale Study of Flash Memory Failures in the Field

Justin Meza
Carnegie Mellon University
meza@cmu.edu

Qiang Wu
Facebook, Inc.
qwu@fb.com

Sanjeev Kumar
Facebook, Inc.
skumar@fb.com

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu

Another Time: NAND Flash Vulnerabilities

- 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.

Meza+, "A Large-Scale Study of Flash Memory Errors in the Field," SIGMETRICS 2015.

Luo+, "Enabling Accurate and Practical Online Flash Channel Modeling for Modern MLC NAND Flash Memory," IEEE JSAC 2016.

Cai+, "Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques," HPCA 2017.

Fukami+, "Improving the Reliability of Chip-Off Forensic Analysis of NAND Flash Memory Devices," DFRWS EU 2017.

Flash Memory Programming Vulnerabilities

- Yu Cai, Saugata Ghose, Yixin Luo, Ken Mai, Onur Mutlu, and Erich F. Haratsch,
"Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques"
Proceedings of the 23rd International Symposium on High-Performance Computer Architecture (HPCA) Industrial Session, Austin, TX, USA, February 2017.
[\[Slides \(pptx\) \(pdf\)\]](#) [\[Lightning Session Slides \(pptx\) \(pdf\)\]](#)

Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques

Yu Cai[†] Saugata Ghose[†] Yixin Luo^{‡†} Ken Mai[†] Onur Mutlu^{§†} Erich F. Haratsch[‡]
[†]*Carnegie Mellon University* [‡]*Seagate Technology* [§]*ETH Zürich*

Other Works on Flash Memory

NAND Flash Error Model



Experimentally characterize and model dominant errors

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

Luo et al., "Enabling Accurate and Practical Online Flash Channel Modeling for Modern MLC NAND Flash Memory", **JSAC 2016**



Cai et al., "Threshold voltage distribution in MLC NAND Flash Memory: Characterization, Analysis, and Modeling", **DATE 2013**

Cai et al., "Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques", **HPCA 2017**

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

Cai et al., "Neighbor-Cell Assisted Error Correction in MLC NAND Flash Memories", **SIGMETRICS 2014**

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

Cai et al., "Flash Correct-and-Refresh: Retention-aware error management for increased flash memory lifetime", **ICCD 2012**

Cai et al., "Error Analysis and Retention-Aware Error Management for NAND Flash Memory", **ITJ 2013**

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

Threshold Voltage Distribution

- Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai, **"Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis and Modeling"** *Proceedings of the Design, Automation, and Test in Europe Conference (DATE)*, Grenoble, France, March 2013. Slides (ppt)

Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis, and Modeling

Yu Cai¹, Erich F. Haratsch², Onur Mutlu¹ and Ken Mai¹

¹DSSC, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA

²LSI Corporation, 1110 American Parkway NE, Allentown, PA

¹{yucai, onur, kenmai}@andrew.cmu.edu, ²erich.haratsch@lsi.com

Program Interference and Vref Prediction

- Yu Cai, Onur Mutlu, Erich F. Haratsch, and Ken Mai, **"Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation"**
Proceedings of the 31st IEEE International Conference on Computer Design (ICCD), Asheville, NC, October 2013.
[Slides \(pptx\)](#) [\(pdf\)](#) [Lightning Session Slides \(pdf\)](#)

Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation

Yu Cai¹, Onur Mutlu¹, Erich F. Haratsch² and Ken Mai¹

1. Data Storage Systems Center, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA

2. LSI Corporation, San Jose, CA

yucaicai@gmail.com, {omutlu, kenmai}@andrew.cmu.edu

Neighbor-Assisted Error Correction

- Yu Cai, Gulay Yalcin, Onur Mutlu, Eric Haratsch, Osman Unsal, Adrian Cristal, and Ken Mai,

"Neighbor-Cell Assisted Error Correction for MLC NAND Flash Memories"

Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems

(SIGMETRICS), Austin, TX, June 2014. [Slides \(ppt\)](#) [\(pdf\)](#)

Neighbor-Cell Assisted Error Correction for MLC NAND Flash Memories

Yu Cai¹, Gulay Yalcin², Onur Mutlu¹, Erich F. Haratsch⁴,
Osman Unsal², Adrian Cristal^{2,3}, and Ken Mai¹

¹Electrical and Computer Engineering Department, Carnegie Mellon University

²Barcelona Supercomputing Center, Spain

³IIIA – CSIC – Spain National Research Council

⁴LSI Corporation

yucaicai@gmail.com, {omutlu, kenmai}@ece.cmu.edu, {gulay.yalcin, adrian.cristal, osman.unsal}@bsc.es

Data Retention

- Yu Cai, Yixin Luo, Erich F. Haratsch, Ken Mai, and Onur Mutlu, **"Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery"**
Proceedings of the 21st International Symposium on High-Performance Computer Architecture (HPCA), Bay Area, CA, February 2015.
[[Slides \(pptx\)](#)] [[pdf](#)]

Data Retention in MLC NAND Flash Memory: Characterization, Optimization, and Recovery

Yu Cai, Yixin Luo, Erich F. Haratsch*, Ken Mai, Onur Mutlu
Carnegie Mellon University, *LSI Corporation

yucaicai@gmail.com, yixinluo@cs.cmu.edu, erich.haratsch@lsi.com, {[kenmai](mailto:kenmai@ece.cmu.edu), [omutlu](mailto:omutlu@ece.cmu.edu)}@ece.cmu.edu

SSD Error Analysis in the Field

- First large-scale field study of flash memory errors
- 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.*
[\[Slides \(pptx\) \(pdf\)\]](#) [\[Coverage at ZDNet\]](#) [\[Coverage on The Register\]](#) [\[Coverage on TechSpot\]](#) [\[Coverage on The Tech Report\]](#)

A Large-Scale Study of Flash Memory Failures in the Field

Justin Meza
Carnegie Mellon University
meza@cmu.edu

Qiang Wu
Facebook, Inc.
qwu@fb.com

Sanjeev Kumar
Facebook, Inc.
skumar@fb.com

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu

Flash Memory Programming Vulnerabilities

- Yu Cai, Saugata Ghose, Yixin Luo, Ken Mai, Onur Mutlu, and Erich F. Haratsch,
"Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques"
Proceedings of the 23rd International Symposium on High-Performance Computer Architecture (HPCA) Industrial Session, Austin, TX, USA, February 2017.
[\[Slides \(pptx\) \(pdf\)\]](#) [\[Lightning Session Slides \(pptx\) \(pdf\)\]](#)

Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques

Yu Cai[†] Saugata Ghose[†] Yixin Luo^{‡†} Ken Mai[†] Onur Mutlu^{§†} Erich F. Haratsch[‡]
[†]*Carnegie Mellon University* [‡]*Seagate Technology* [§]*ETH Zürich*

Accurate and Online Channel Modeling

- Yixin Luo, Saugata Ghose, Yu Cai, Erich F. Haratsch, and Onur Mutlu, **"Enabling Accurate and Practical Online Flash Channel Modeling for Modern MLC NAND Flash Memory"** to appear in *IEEE Journal on Selected Areas in Communications (JSAC)*, 2016.

Enabling Accurate and Practical Online Flash Channel Modeling for Modern MLC NAND Flash Memory

Yixin Luo, Saugata Ghose, Yu Cai, Erich F. Haratsch, Onur Mutlu

More on DRAM Refresh

Tackling Refresh: Solutions

- Parallelize refreshes with accesses [Chang+ HPCA'14]
- Eliminate unnecessary refreshes [Liu+ ISCA'12]
 - Exploit device characteristics
 - Exploit data and application characteristics
- Reduce refresh rate and detect+correct errors that occur [Khan+ SIGMETRICS'14]
- Understand retention time behavior in DRAM [Liu+ ISCA'13]

Summary: Refresh-Access Parallelization

- DRAM refresh interferes with memory accesses
 - Degrades system performance and energy efficiency
 - Becomes exacerbated as DRAM density increases
- Goal: Serve memory accesses in parallel with refreshes to reduce refresh interference on demand requests
- Our mechanisms:
 - 1. Enable more parallelization between refreshes and accesses across different banks with [new per-bank refresh scheduling algorithms](#)
 - 2. Enable serving accesses concurrently with refreshes in the same bank by [exploiting parallelism across DRAM subarrays](#)
- Improve system performance and energy efficiency for a wide variety of different workloads and DRAM densities
 - 20.2% and 9.0% for 8-core systems using 32Gb DRAM at low cost
 - Very close to the ideal scheme without refreshes

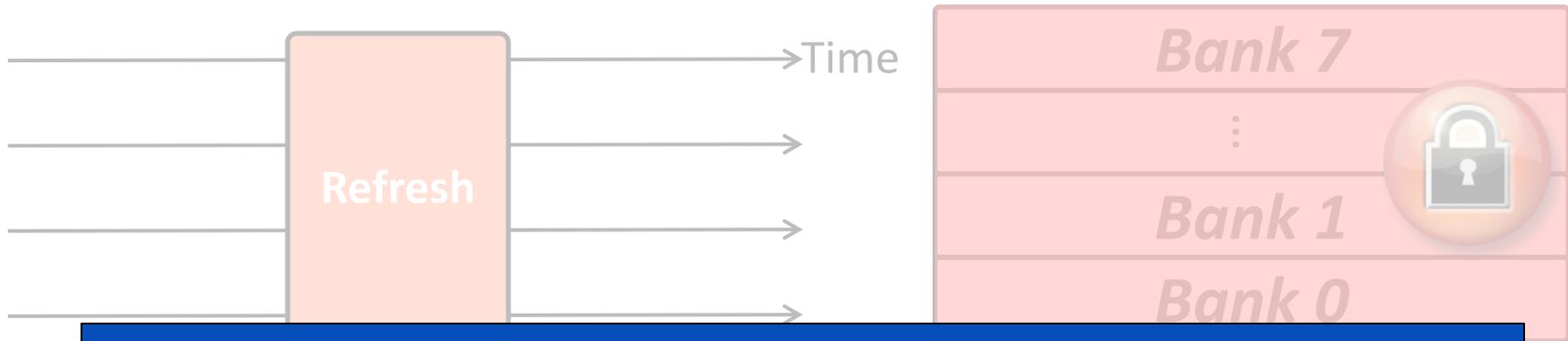
Refresh Penalty



Refresh delays requests by 100s of ns

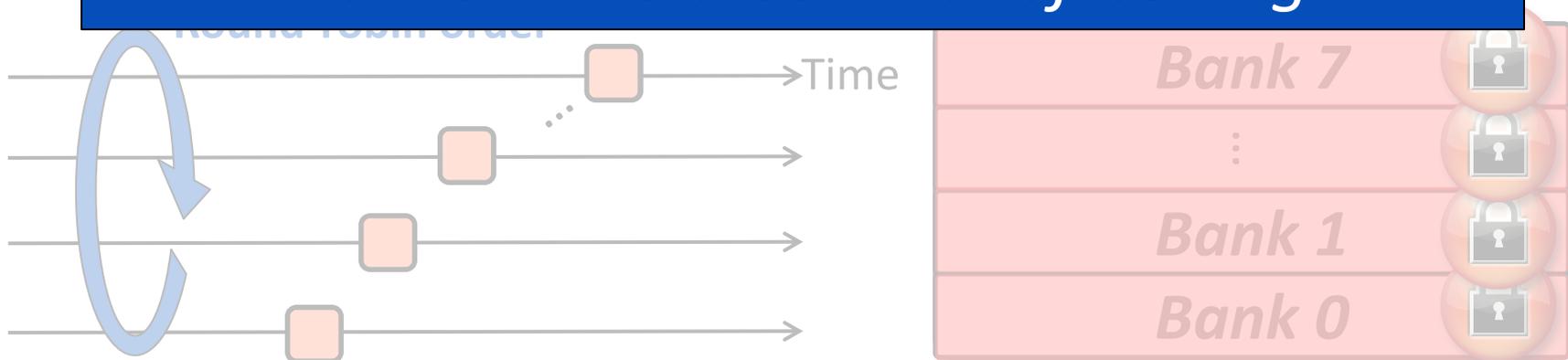
Existing Refresh Modes

All-bank refresh in commodity DRAM (DDR_x)



Per-bank refresh allows accesses to other banks while a bank is refreshing

Per



Shortcomings of Per-Bank Refresh

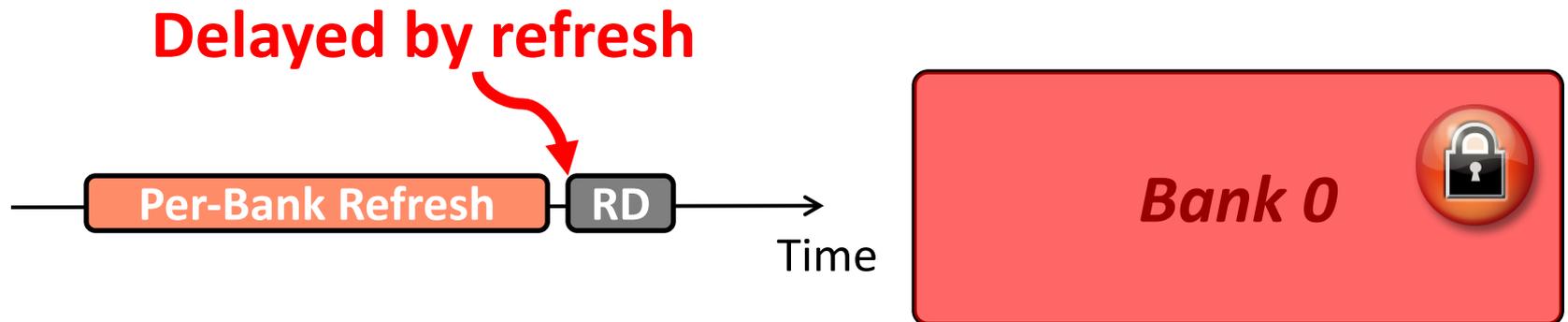
- Problem 1: Refreshes to different banks are scheduled in a **strict round-robin order**
 - The static ordering is hardwired into DRAM chips
 - Refreshes busy banks with many queued requests when other banks are idle
- Key idea: Schedule per-bank refreshes to idle banks opportunistically in a dynamic order

Our First Approach: DARP

- **Dynamic Access-Refresh Parallelization (DARP)**
 - An improved scheduling policy for **per-bank refreshes**
 - Exploits **refresh scheduling flexibility** in DDR DRAM
- **Component 1: Out-of-order per-bank refresh**
 - Avoids poor static scheduling decisions
 - Dynamically issues per-bank refreshes to idle banks
- **Component 2: Write-Refresh Parallelization**
 - Avoids refresh interference on latency-critical reads
 - Parallelizes refreshes with a **batch of writes**

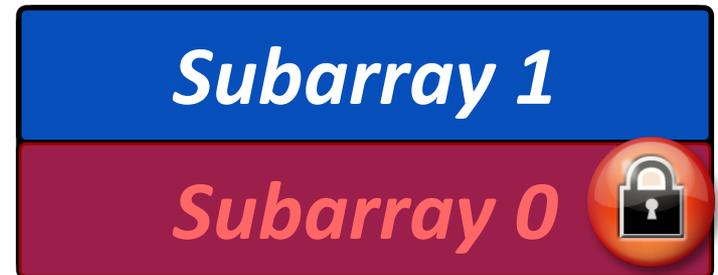
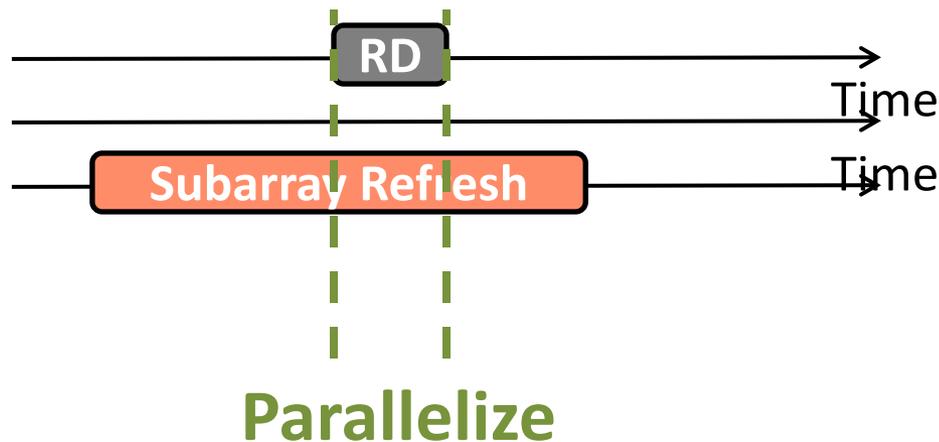
Shortcomings of Per-Bank Refresh

- Problem 2: Banks that are being refreshed cannot concurrently serve memory requests



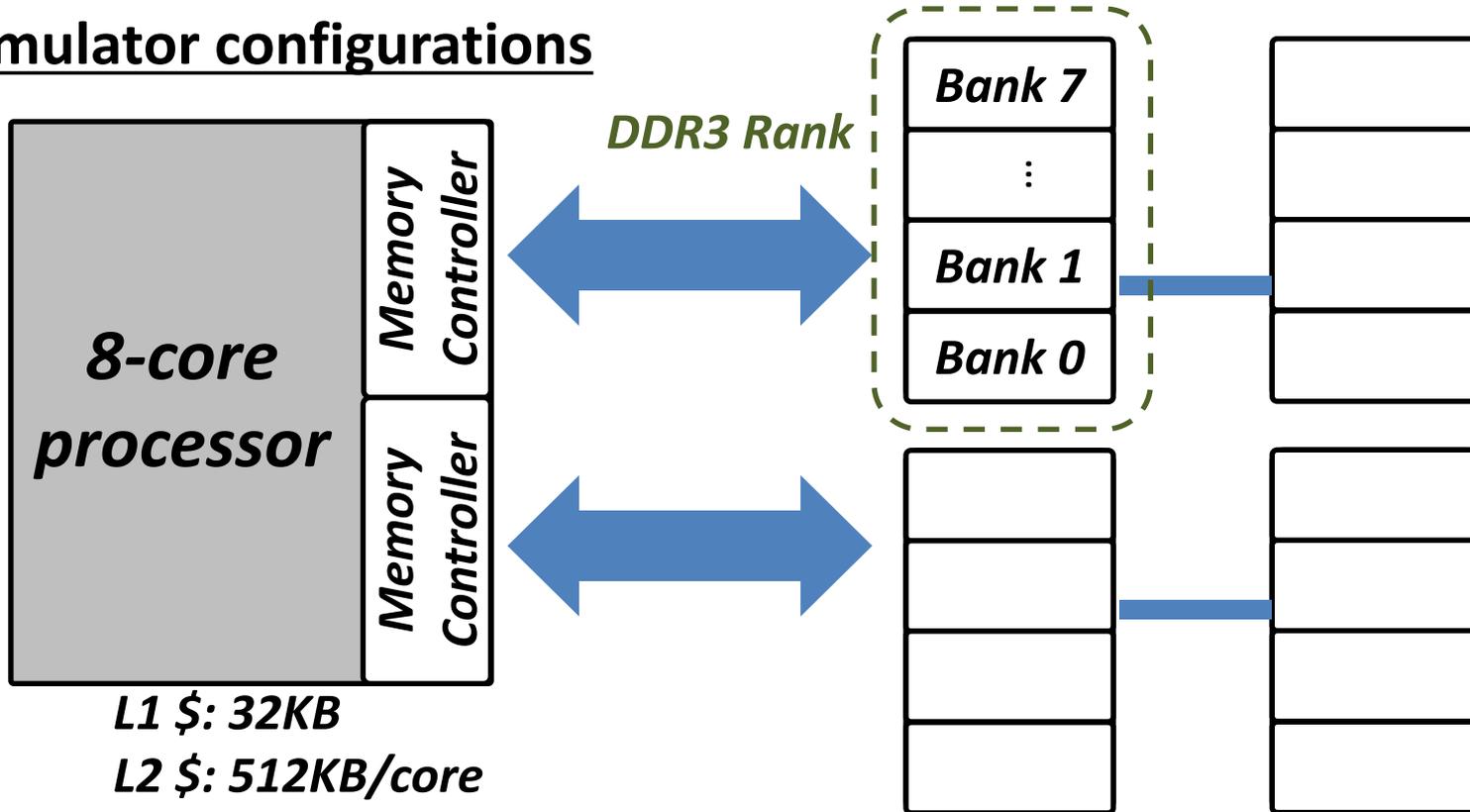
Shortcomings of Per-Bank Refresh

- Problem 2: Refreshing banks cannot concurrently serve memory requests
- Key idea: Exploit **subarrays** within a bank to parallelize refreshes and accesses across **subarrays**



Methodology

Simulator configurations

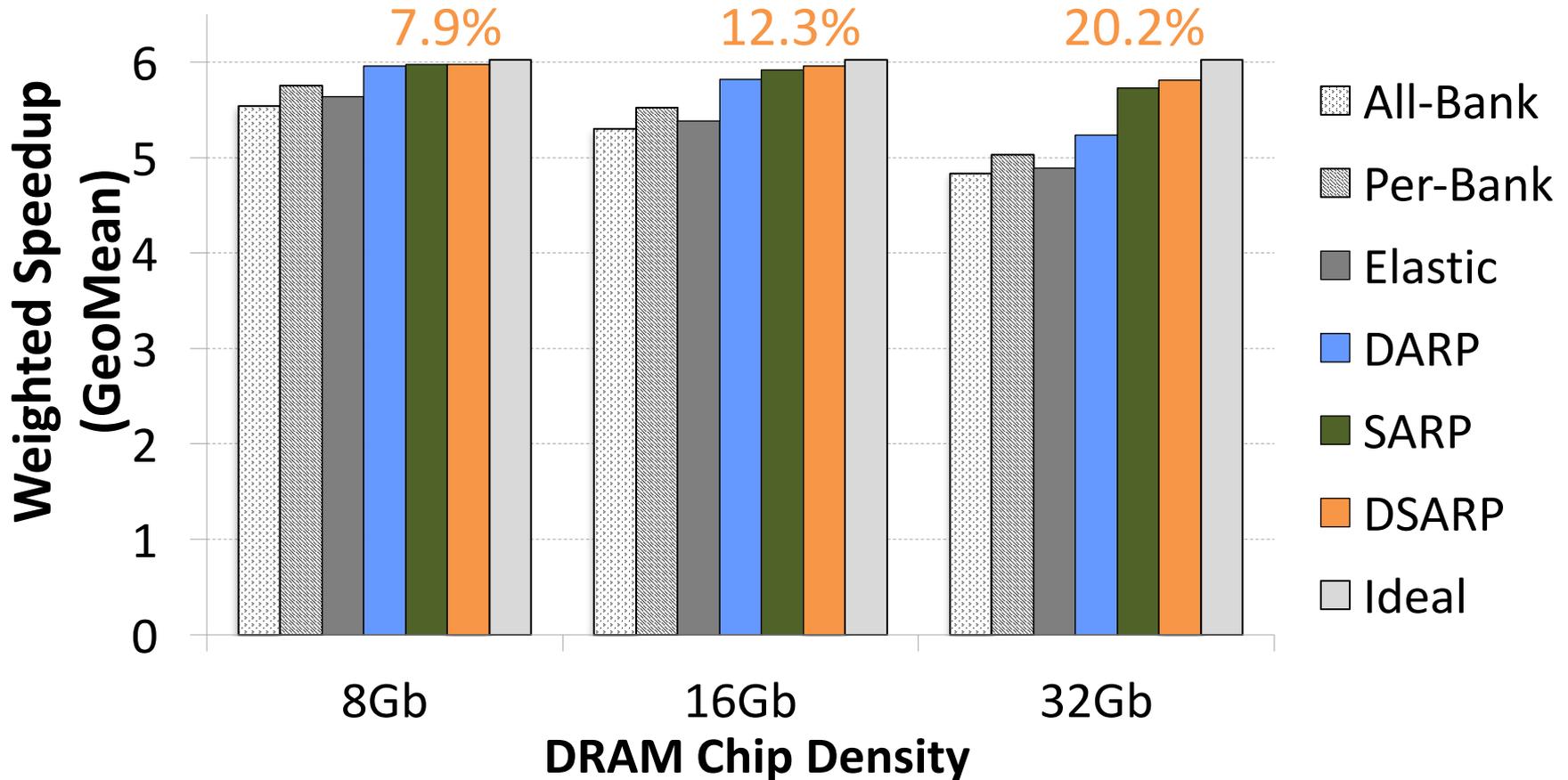


- **100 workloads**: SPEC CPU2006, STREAM, TPC-C/H, random access
- **System performance metric**: *Weighted speedup*

Comparison Points

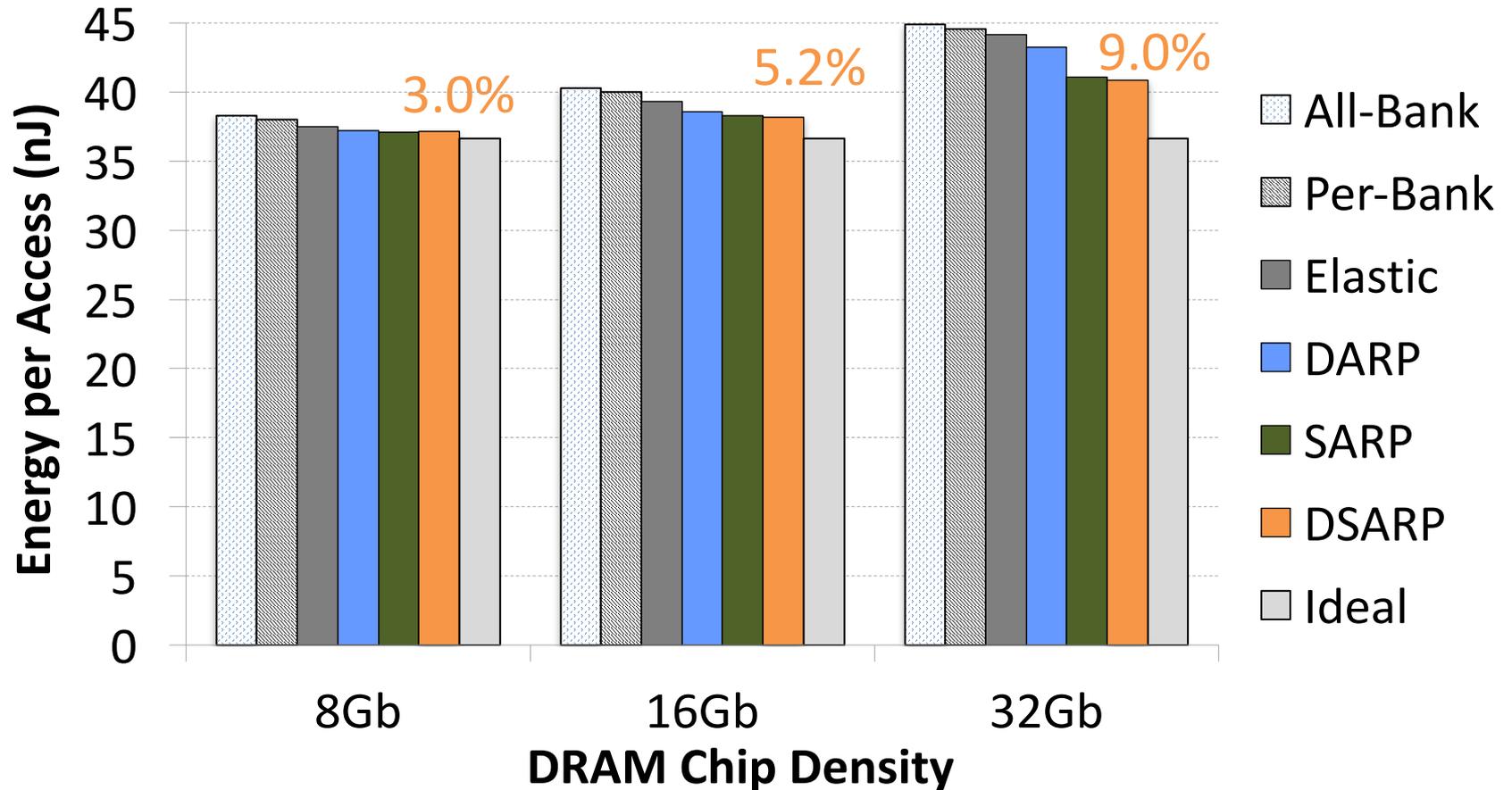
- **All-bank refresh** [DDR3, LPDDR3, ...]
- **Per-bank refresh** [LPDDR3]
- **Elastic refresh** [Stuecheli et al., MICRO '10]:
 - Postpones refreshes by a time delay based on the predicted rank idle time to avoid interference on memory requests
 - Proposed to schedule all-bank refreshes without exploiting per-bank refreshes
 - Cannot parallelize refreshes and accesses within a rank
- **Ideal (no refresh)**

System Performance



2. Consistent system performance improvement across DRAM densities (within 0.9%, 1.2%, and 3.8% of ideal)

Energy Efficiency



Consistent reduction on energy consumption

More Information on Refresh-Access Parallelization

- Kevin Chang, Donghyuk Lee, Zeshan Chishti, Alaa Alameldeen, Chris Wilkerson, Yoongu Kim, and Onur Mutlu,
"Improving DRAM Performance by Parallelizing Refreshes with Accesses"
Proceedings of the 20th International Symposium on High-Performance Computer Architecture (HPCA), Orlando, FL, February 2014.
[[Summary](#)] [[Slides \(pptx\)](#)] [[pdf](#)]

Reducing Performance Impact of DRAM Refresh by Parallelizing Refreshes with Accesses

Kevin Kai-Wei Chang Donghyuk Lee Zeshan Chishti†

Alaa R. Alameldeen† Chris Wilkerson† Yoongu Kim Onur Mutlu

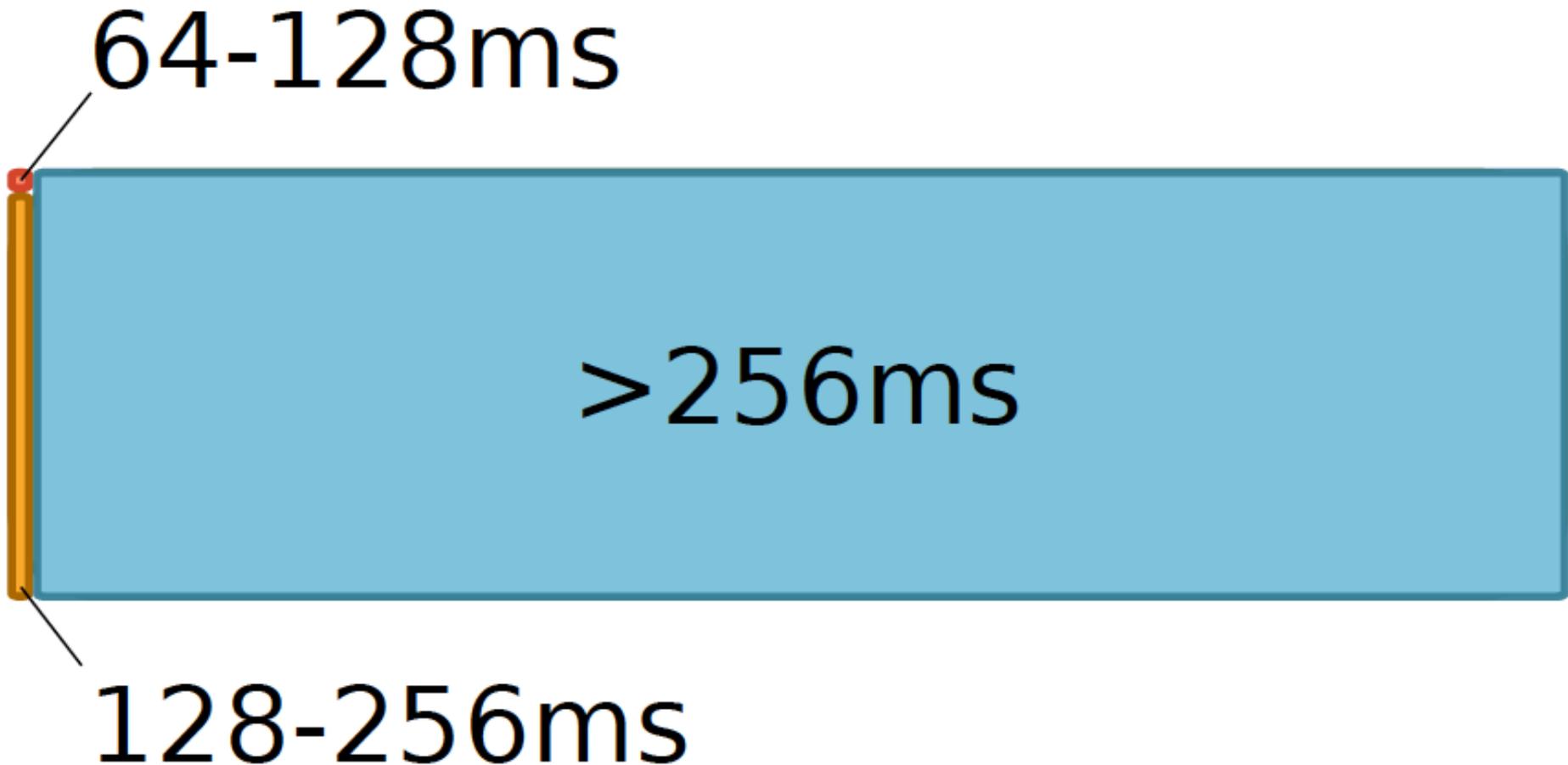
Carnegie Mellon University †Intel Labs

Tackling Refresh: Solutions

- Parallelize refreshes with accesses [Chang+ HPCA'14]
- Eliminate unnecessary refreshes [Liu+ ISCA'12]
 - Exploit device characteristics
 - Exploit data and application characteristics
- Reduce refresh rate and detect+correct errors that occur [Khan+ SIGMETRICS'14]
- Understand retention time behavior in DRAM [Liu+ ISCA'13]

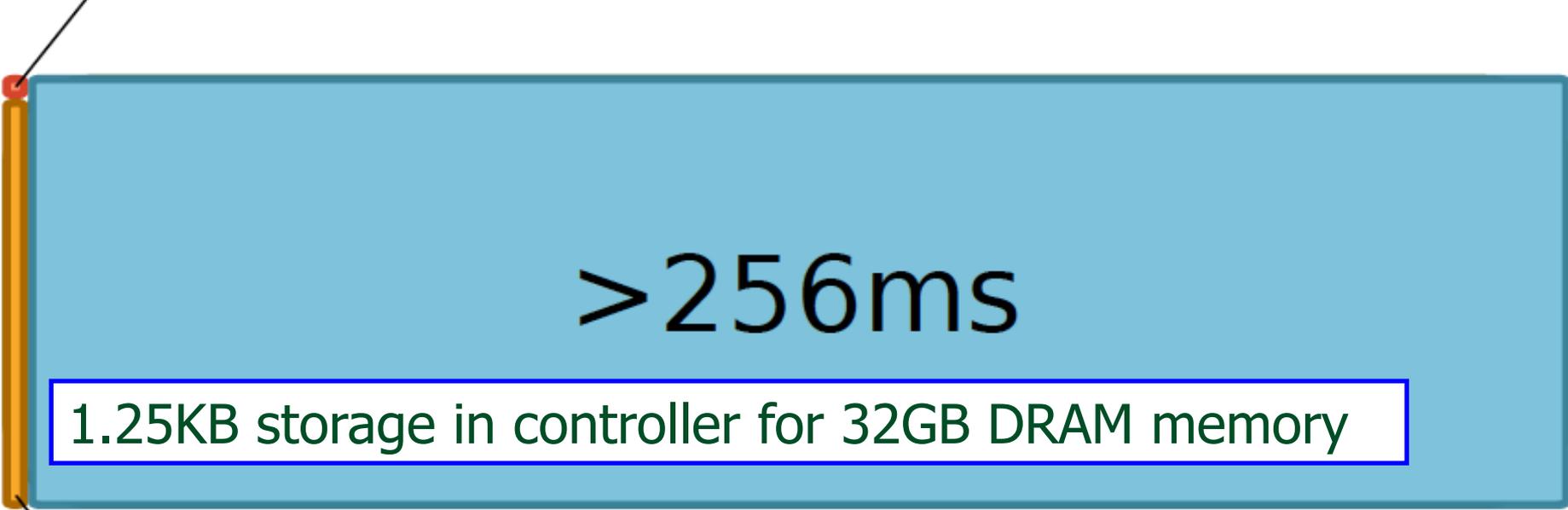
Most Refreshes Are Unnecessary

- Retention Time Profile of DRAM looks like this:



RAIDR: Eliminating Unnecessary Refreshes

64-128ms



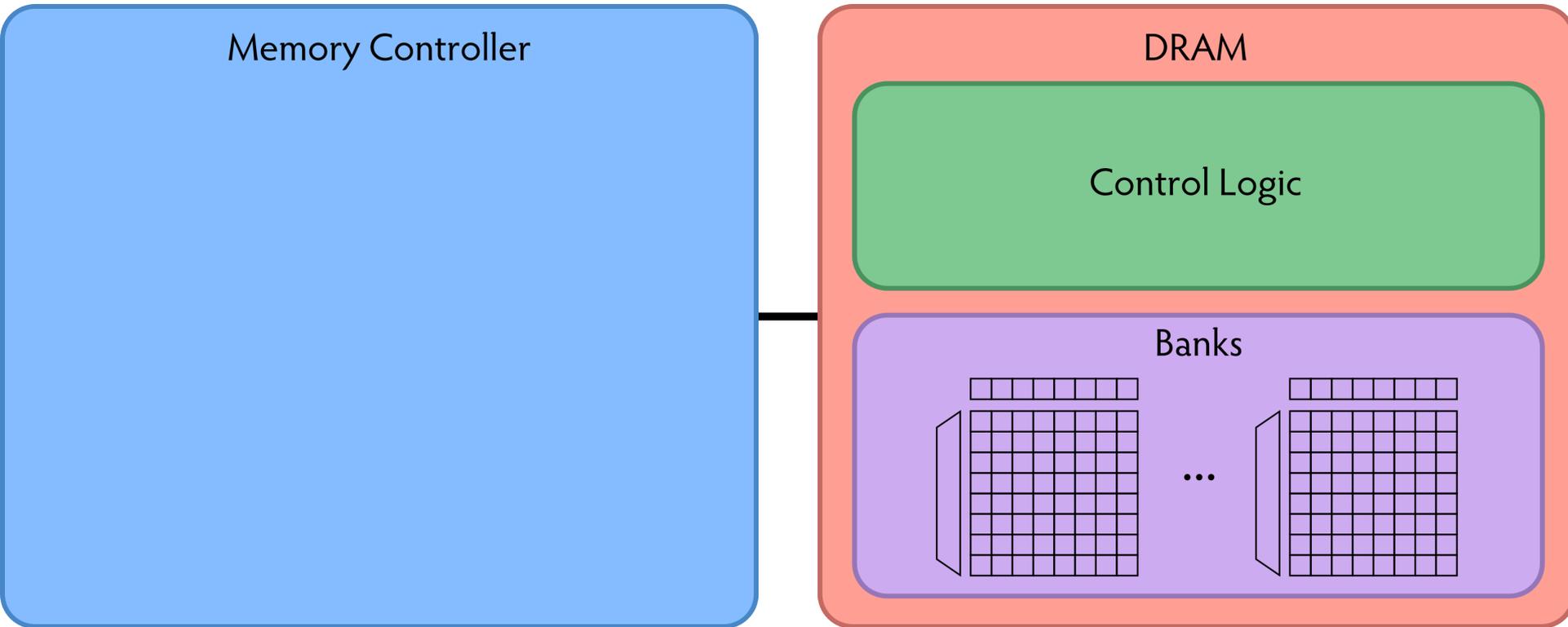
>256ms

1.25KB storage in controller for 32GB DRAM memory

128-256ms

Can reduce refreshes by $\sim 75\%$
→ reduces energy consumption and improves performance

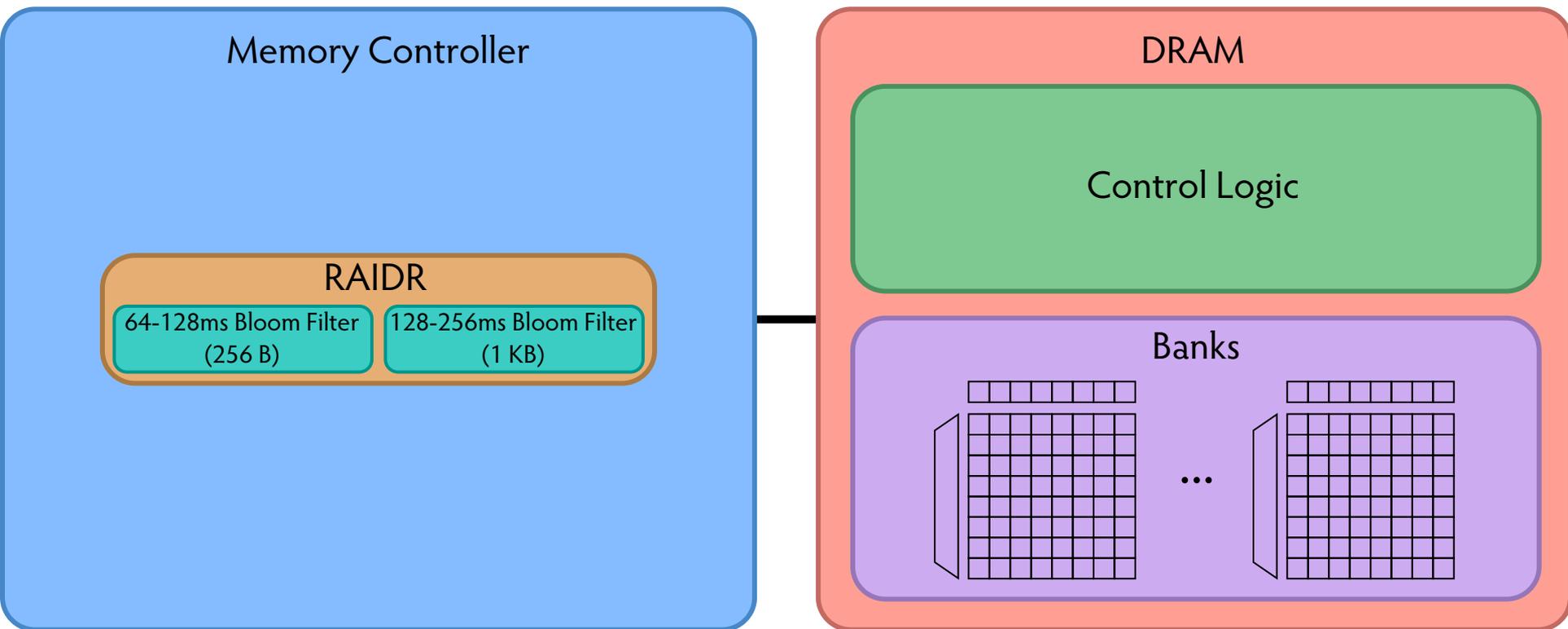
RAIDR: Baseline Design



Refresh control is in DRAM in today's auto-refresh systems

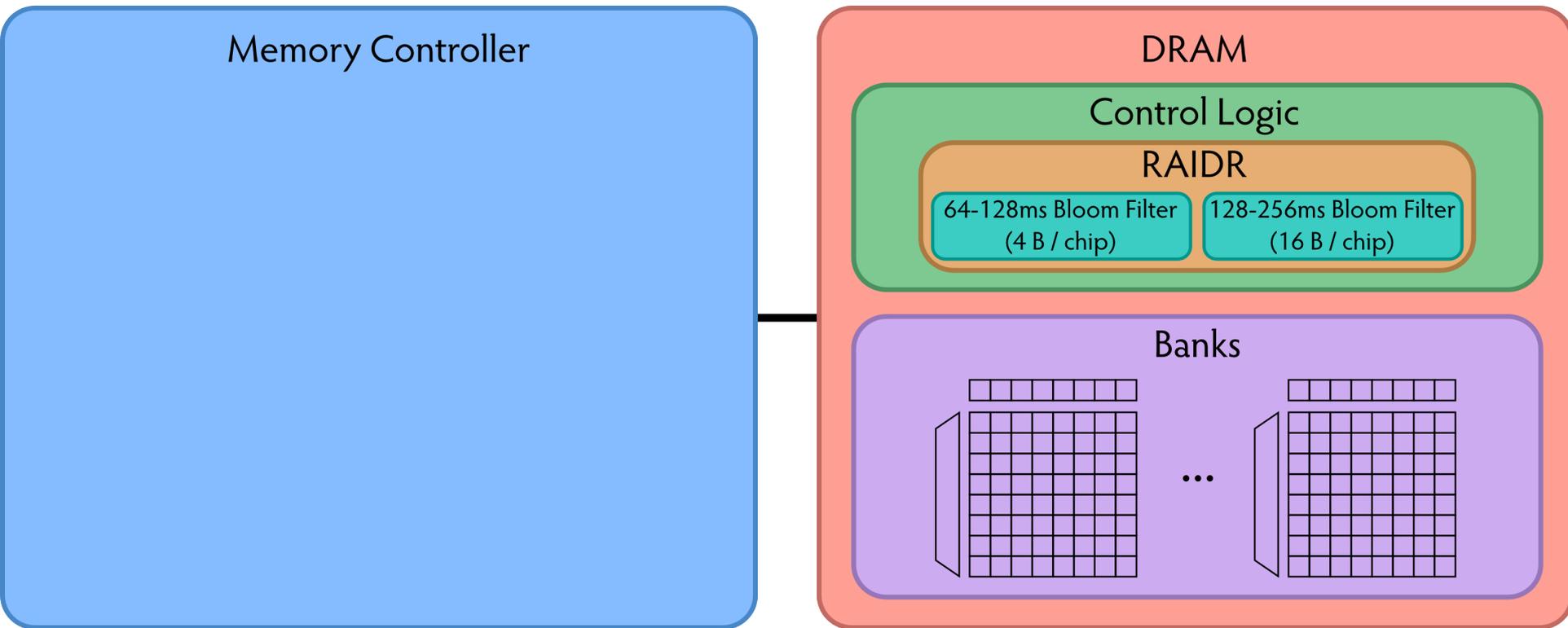
RAIDR can be implemented in either the controller or DRAM

RAIDR in Memory Controller: Option 1



Overhead of RAIDR in DRAM controller:
1.25 KB Bloom Filters, 3 counters, additional commands
issued for per-row refresh (all accounted for in evaluations)

RAIDR in DRAM Chip: Option 2



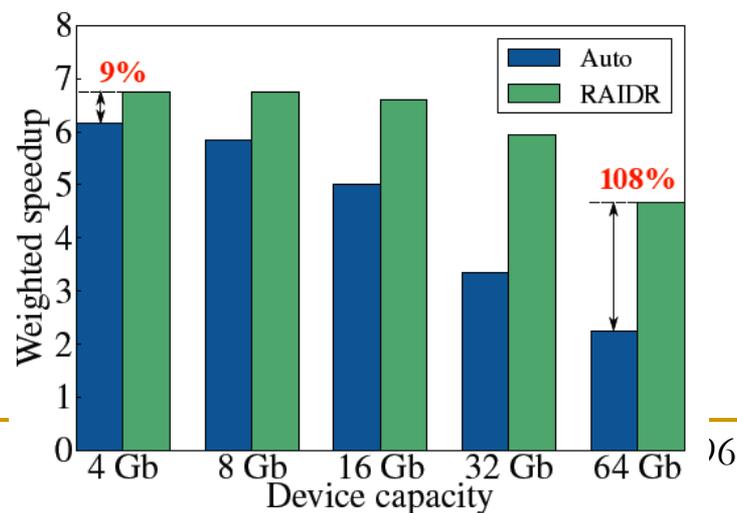
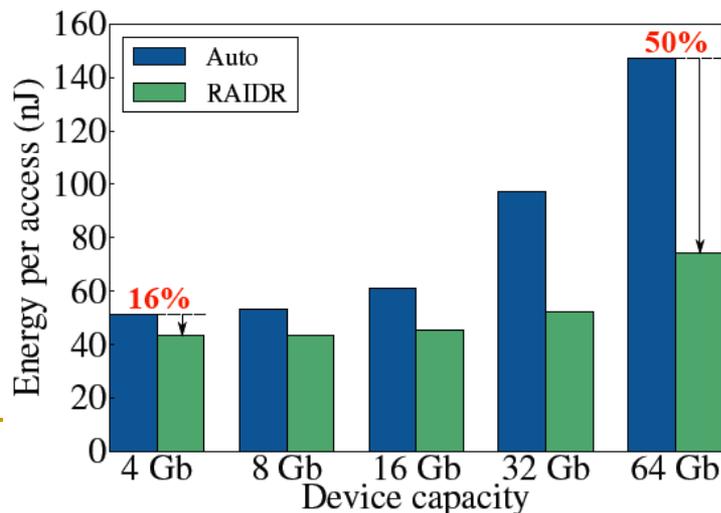
Overhead of RAIDR in DRAM chip:

Per-chip overhead: 20B Bloom Filters, 1 counter (4 Gbit chip)

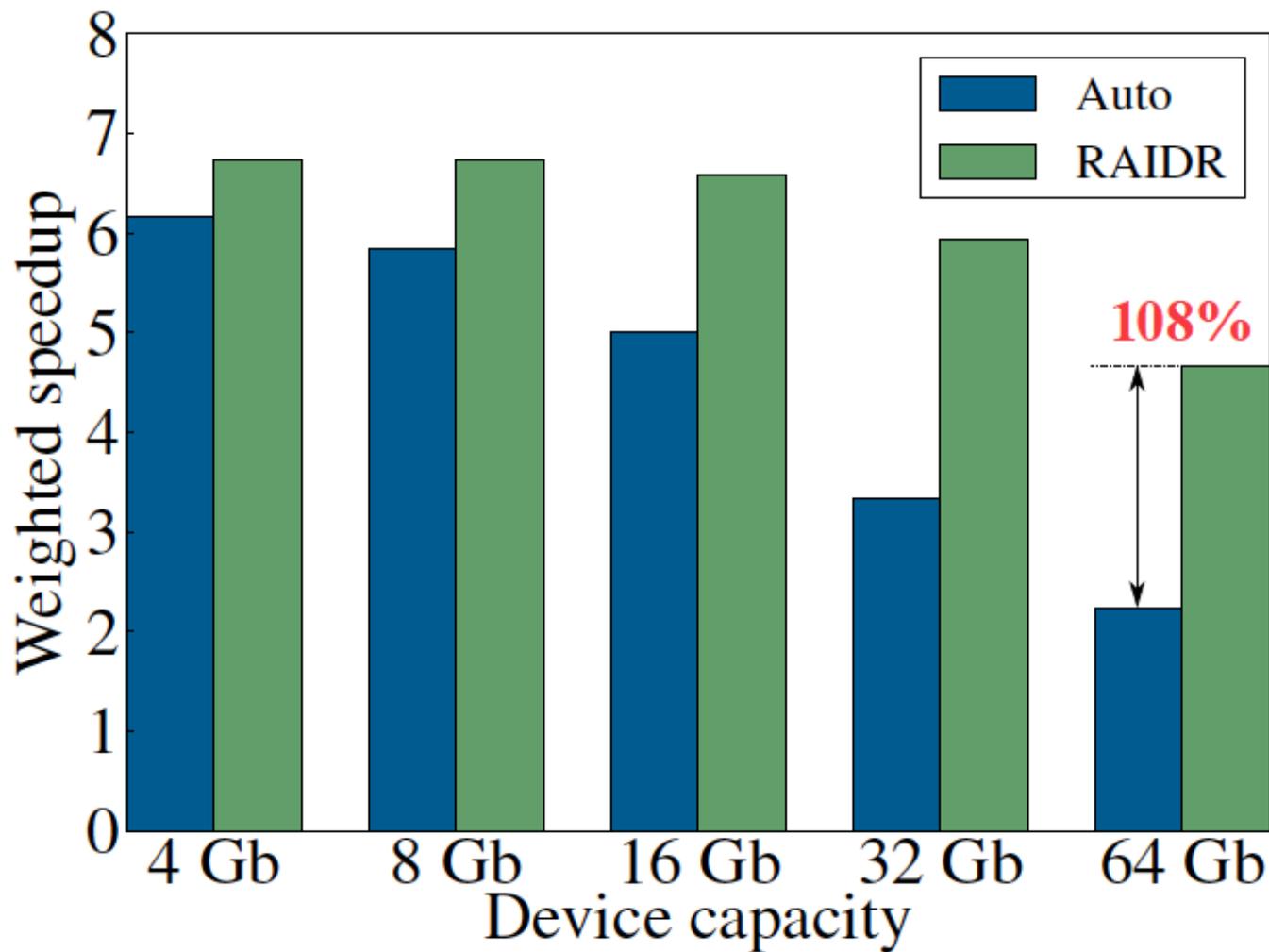
Total overhead: 1.25KB Bloom Filters, 64 counters (32 GB DRAM)

RAIDR: Results and Takeaways

- System: 32GB DRAM, 8-core; SPEC, TPC-C, TPC-H workloads
- RAIDR hardware cost: 1.25 kB (2 Bloom filters)
- Refresh reduction: 74.6%
- Dynamic DRAM energy reduction: 16%
- Idle DRAM power reduction: 20%
- Performance improvement: 9%
- Benefits increase as DRAM scales in density

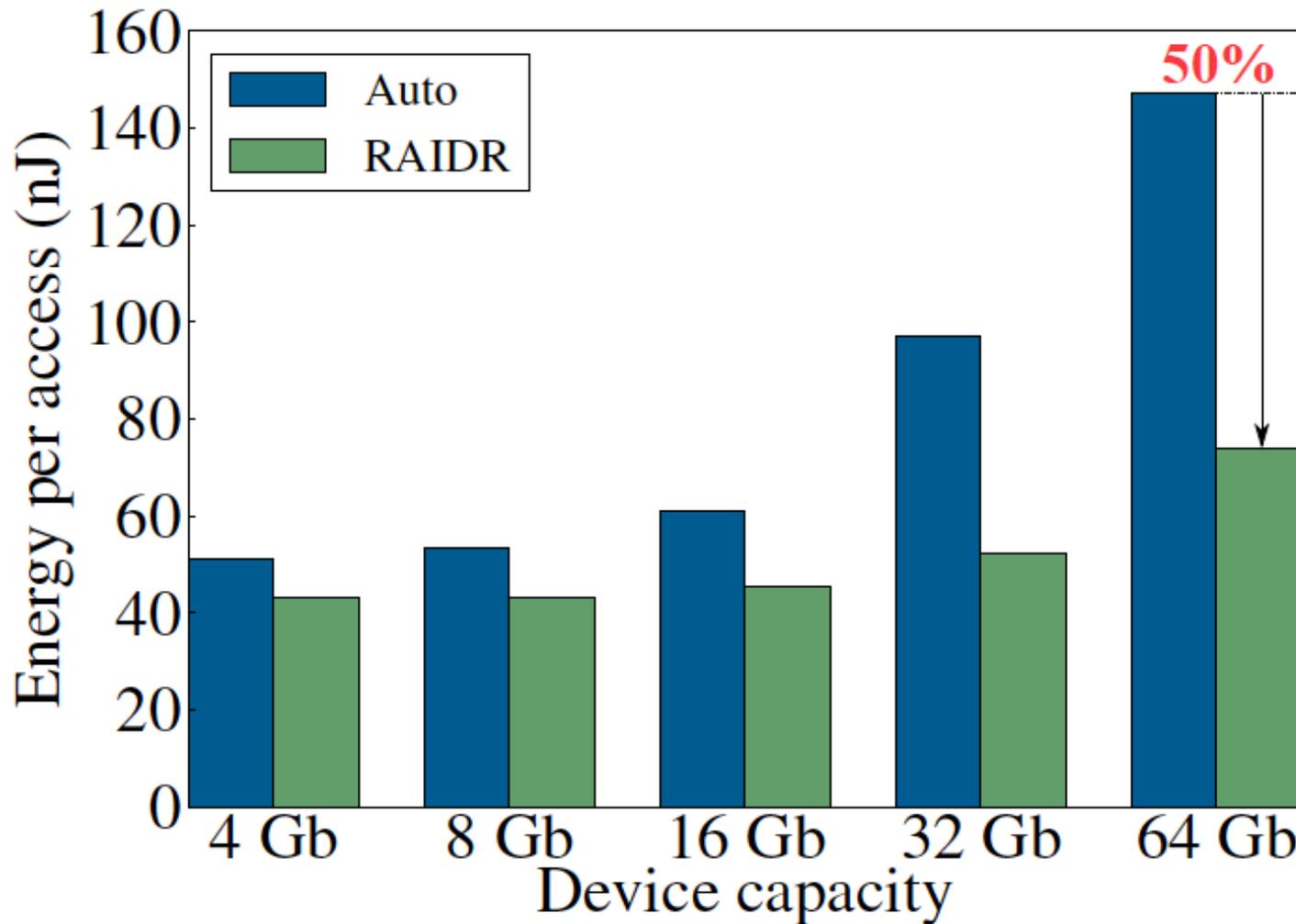


DRAM Device Capacity Scaling: Performance



RAIDR performance benefits increase with DRAM chip capacity

DRAM Device Capacity Scaling: Energy



RAIDR energy benefits increase with DRAM chip capacity

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

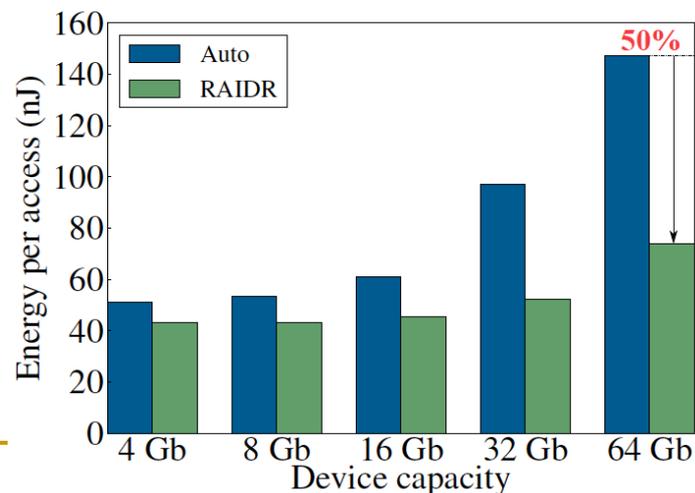
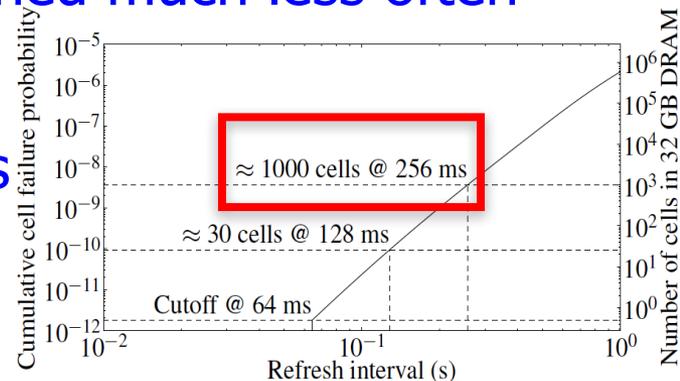
2. Binning: Store rows into bins by retention time in memory controller

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



More on RAIDR

- Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu, **"RAIDR: Retention-Aware Intelligent DRAM Refresh"**
Proceedings of the 39th International Symposium on Computer Architecture (ISCA), Portland, OR, June 2012.
Slides (pdf)

RAIDR: Retention-Aware Intelligent DRAM Refresh

Jamie Liu Ben Jaiyen Richard Veras Onur Mutlu
Carnegie Mellon University

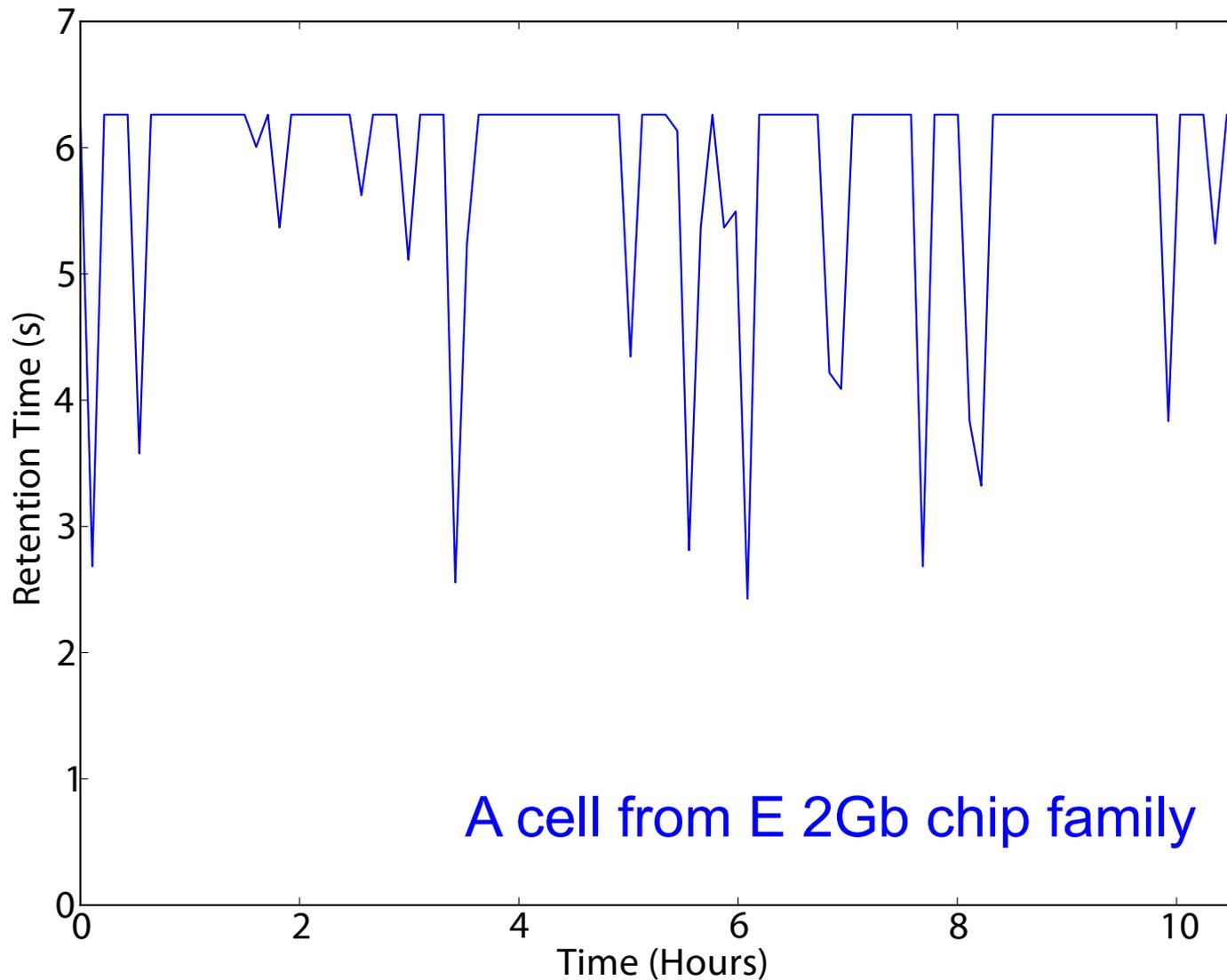
Tackling Refresh: Solutions

- Parallelize refreshes with accesses [Chang+ HPCA'14]
- Eliminate unnecessary refreshes [Liu+ ISCA'12]
 - Exploit device characteristics
 - Exploit data and application characteristics
- Reduce refresh rate and detect+correct errors that occur [Khan+ SIGMETRICS'14]
- Understand retention time behavior in DRAM [Liu+ ISCA'13]

Motivation: Understanding Retention

- Past works require **accurate and reliable measurement of retention time of each DRAM row**
 - To maintain data integrity while reducing refreshes
- **Assumption: worst-case retention time of each row can be determined and stays the same at a given temperature**
 - Some works propose writing all 1's and 0's to a row, and measuring the time before data corruption
- **Question:**
 - Can we reliably and accurately determine retention times of all DRAM rows?

An Example VRT Cell



VRT: Implications on Profiling Mechanisms

- Problem 1: There does not seem to be a way of determining if a cell exhibits VRT without actually observing a cell exhibiting VRT
 - VRT is a memoryless random process [Kim+ JJAP 2010]
- Problem 2: VRT complicates retention time profiling by DRAM manufacturers
 - Exposure to very high temperatures can induce VRT in cells that were not previously susceptible
 - can happen during soldering of DRAM chips
 - manufacturer's retention time profile may not be accurate
- One option for future work: Use ECC to continuously profile DRAM online while aggressively reducing refresh rate
 - Need to keep ECC overhead in check

More on DRAM Retention Analysis

- Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu,
"An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms"
Proceedings of the 40th International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013. [Slides \(ppt\)](#) [Slides \(pdf\)](#)

An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms

Jamie Liu*

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
jamiel@alumni.cmu.edu

Ben Jaiyen*

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
bjaiyen@alumni.cmu.edu

Yoongu Kim

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
yoonguk@ece.cmu.edu

Chris Wilkerson

Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
chris.wilkerson@intel.com

Onur Mutlu

Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
onur@cmu.edu

Tackling Refresh: Solutions

- Parallelize refreshes with accesses [Chang+ HPCA'14]
- Eliminate unnecessary refreshes [Liu+ ISCA'12]
 - Exploit device characteristics
 - Exploit data and application characteristics
- Reduce refresh rate and detect+correct errors that occur [Khan+ SIGMETRICS'14]
- Understand retention time behavior in DRAM [Liu+ ISCA'13]

Towards an Online Profiling System

Key Observations:

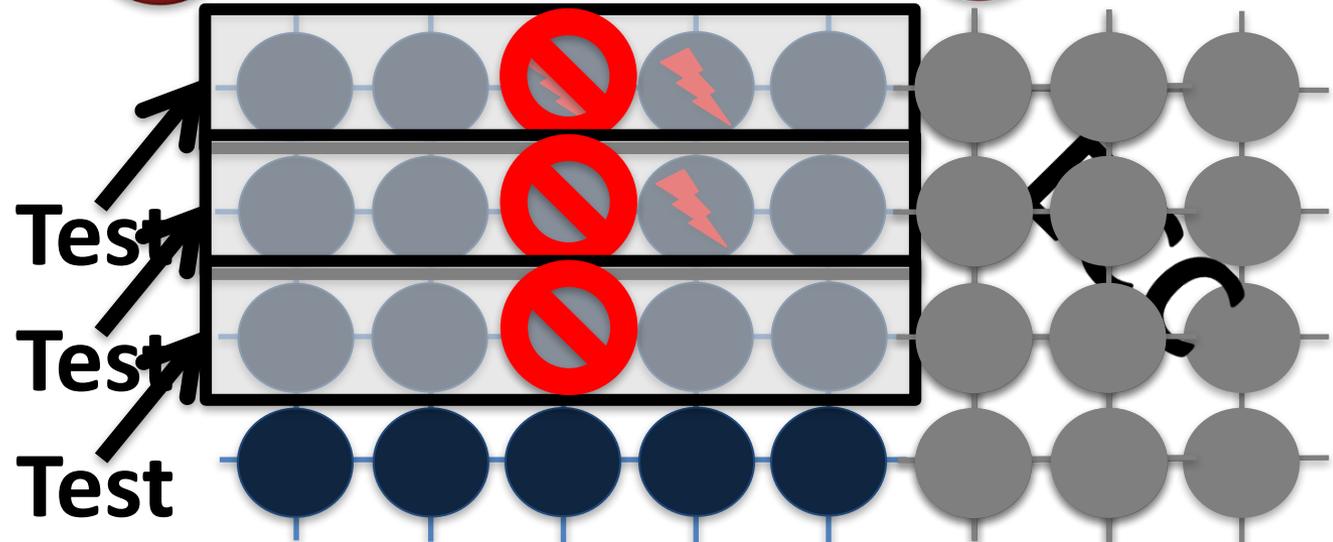
- **Testing** alone **cannot detect** all possible failures
- **Combination** of ECC and other mitigation techniques is much more **effective**
 - **But degrades performance**
- **Testing** can help to reduce the **ECC strength**
 - Even when starting with a **higher strength ECC**

Towards an Online Profiling System

Initially Protect DRAM with Strong ECC **1**

Periodically Test Parts of DRAM **2**

Mitigate errors and reduce ECC **3**



Run tests periodically after a short interval at smaller regions of memory

More on Online Profiling of DRAM

- Samira Khan, Donghyuk Lee, Yoongu Kim, Alaa Alameldeen, Chris Wilkerson, and Onur Mutlu,
"The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study"
Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), Austin, TX, June 2014. [[Slides \(pptx\) \(pdf\)](#)] [[Poster \(pptx\) \(pdf\)](#)] [[Full data sets](#)]

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

How Do We Make RAIDR Work in the
Presence of the VRT Phenomenon?

Making RAIDR Work w/ Online Profiling & ECC

- Moinuddin Qureshi, Dae Hyun Kim, Samira Khan, Prashant Nair, and Onur Mutlu,
"AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems"
Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, June 2015.
[\[Slides \(pptx\) \(pdf\)\]](#)

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems

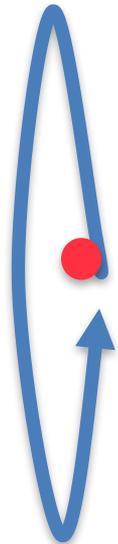
Moinuddin K. Qureshi[†] Dae-Hyun Kim[†] Samira Khan[‡] Prashant J. Nair[†] Onur Mutlu[‡]
[†]Georgia Institute of Technology [‡]Carnegie Mellon University
{moin, dhkim, pnair6}@ece.gatech.edu {samirakhan, onur}@cmu.edu

AVATAR

Insight: Avoid retention failures → Upgrade row on ECC error

Observation: Rate of VRT \gg Rate of soft error (50x-2500x)

Scrub
(15 min)



DRAM Rows

ECC	A
ECC	B
ECC	C
ECC	D
ECC	E
ECC	F
ECC	G
ECC	H

Ref. Rate Table

0
0
1
0
0
0
0
1
1

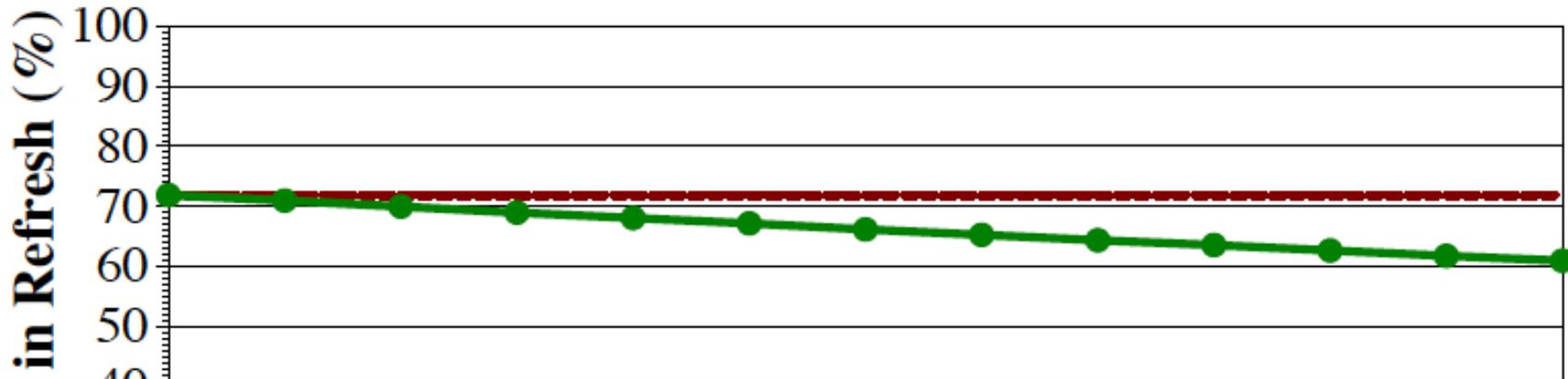
Weak Cell

RETENTION
PROFILING

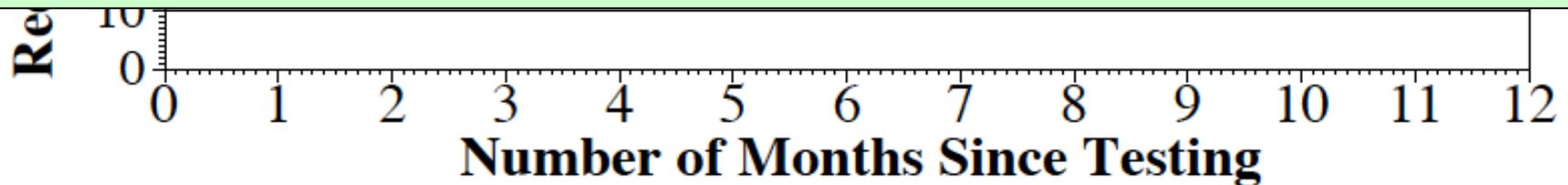
Row protected from
future
retention failures

AVATAR mitigates VRT by increasing refresh rate on error

RESULTS: REFRESH SAVINGS

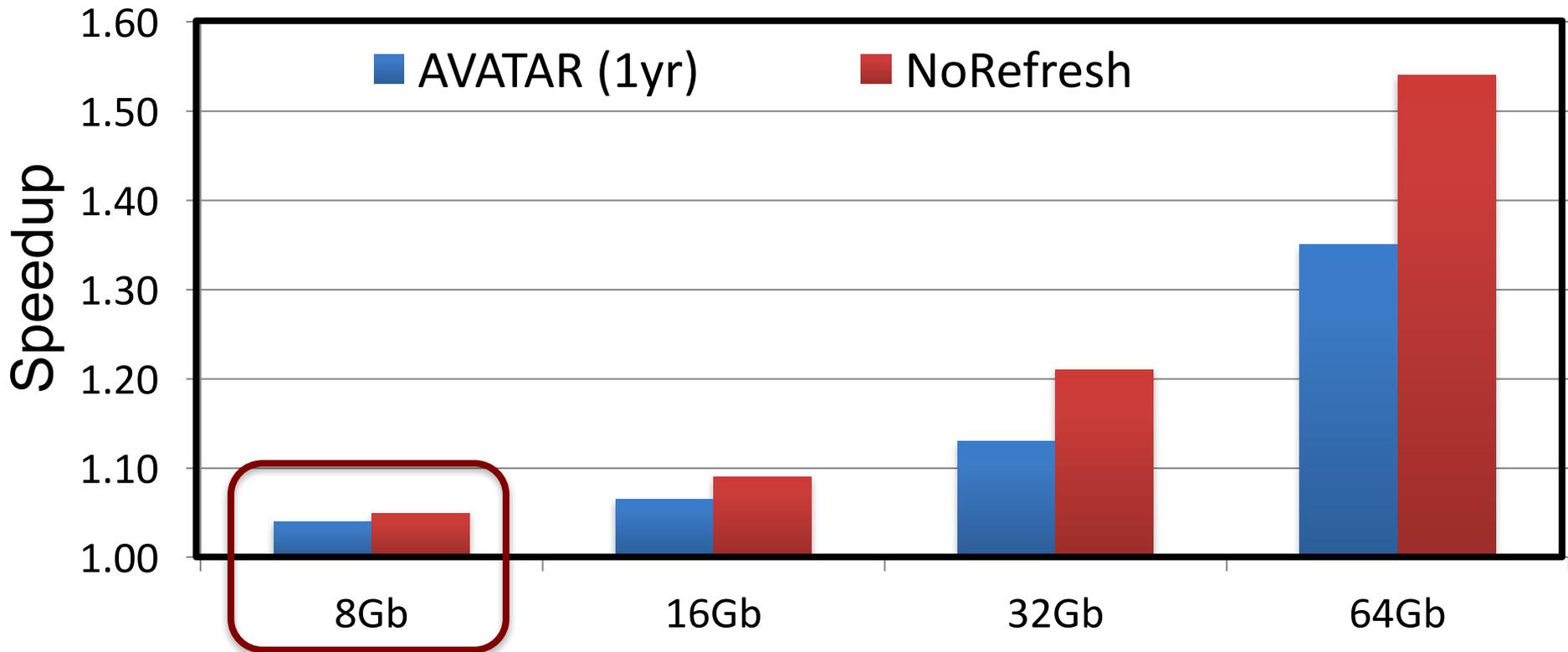


Retention Testing Once a Year can revert refresh saving from 60% to 70%



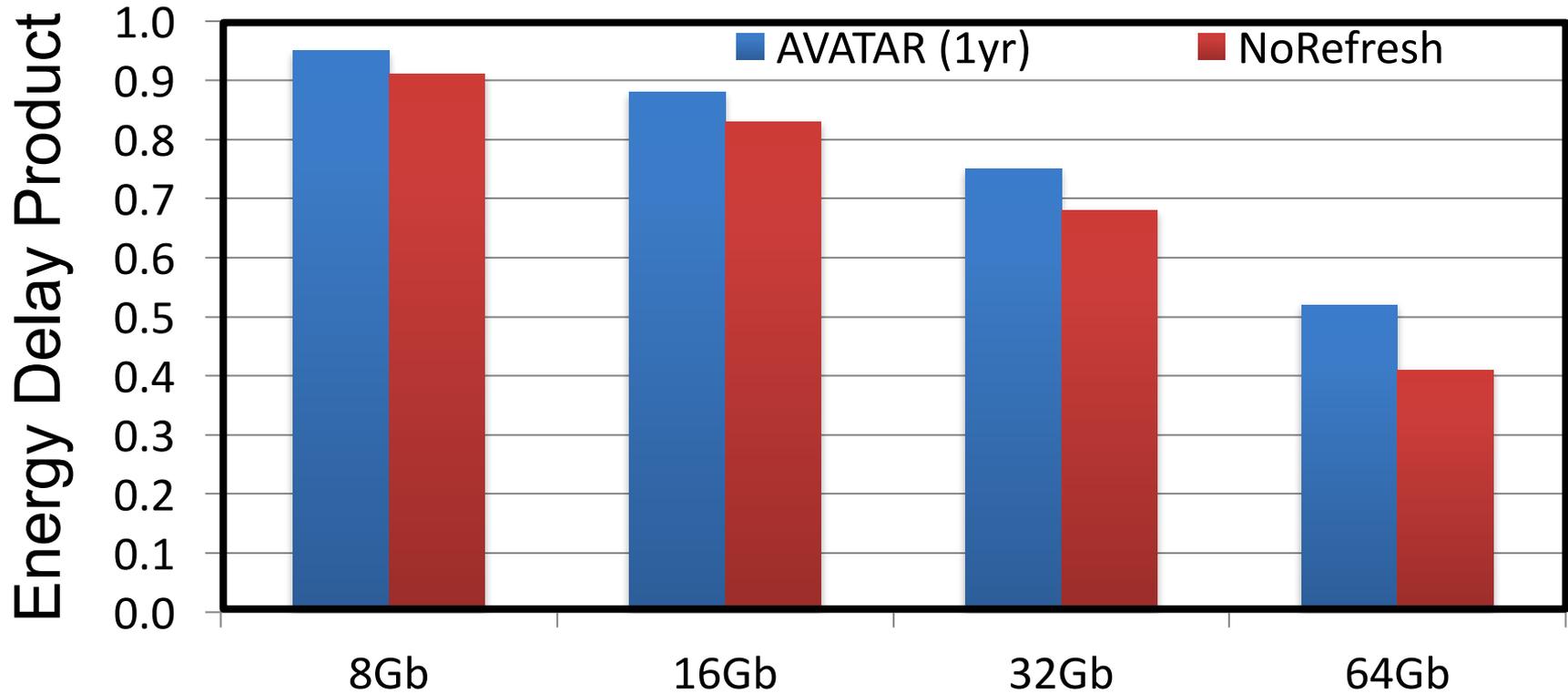
AVATAR reduces refresh by 60%-70%, similar to multi rate refresh but with VRT tolerance

SPEEDUP



AVATAR gets 2/3rd the performance of NoRefresh. More gains at higher capacity nodes

ENERGY DELAY PRODUCT



**AVATAR reduces EDP,
Significant reduction at higher capacity nodes**

Making RAIDR Work w/ Online Profiling & ECC

- Moinuddin Qureshi, Dae Hyun Kim, Samira Khan, Prashant Nair, and Onur Mutlu,
"AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems"
Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, June 2015.
[[Slides \(pptx\)](#)] [[pdf](#)]

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems

Moinuddin K. Qureshi[†] Dae-Hyun Kim[†] Samira Khan[‡] Prashant J. Nair[†] Onur Mutlu[‡]
[†]Georgia Institute of Technology [‡]Carnegie Mellon University
{*moin, dhkim, pnair6*}@ece.gatech.edu {*samirakhan, onur*}@cmu.edu

DRAM Refresh: Summary and Conclusions

- **DRAM refresh is a critical challenge**
 - in scaling DRAM technology efficiently to higher capacities
- **Discussed several promising solution directions**
 - Parallelize refreshes with accesses [Chang+ HPCA'14]
 - Eliminate unnecessary refreshes [Liu+ ISCA'12]
 - Reduce refresh rate and detect+correct errors that occur [Khan+ SIGMETRICS'14]
- **Examined properties of retention time behavior** [Liu+ ISCA'13]
 - Enable realistic VRT-Aware refresh techniques [Qureshi+ DSN'15]
- **Many avenues for overcoming DRAM refresh challenges**
 - Handling DPD/VRT phenomena
 - Enabling online retention time profiling and error mitigation
 - Exploiting application behavior

Other Backup Slides

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

- Can Alkan, Chita Das, Phil Gibbons, Sriram Govindan, Norm Jouppi, Mahmut Kandemir, Mike Kozuch, Konrad Lai, Ken Mai, Todd Mowry, 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

Summary

Business as Usual	Opportunity
RowHammer	Memory controller anticipates and fixes errors
Fixed, frequent refreshes	Heterogeneous refresh rate across memory
Fixed, high latency	Heterogeneous latency in time and space
Slow page copy & initialization	Exploit internal connectivity in memory to move data
Fixed reliability mechanisms	Heterogeneous reliability across time and space
Memory as a dumb device	Memory as an accelerator and autonomous agent
DRAM-only main memory	Emerging memory technologies and hybrid memories
Two-level data storage model	Unified interface to all data
Large timing and error margins	Online adaptation of timing and error margins
Poor performance guarantees	Strong service guarantees and configurable QoS
Fixed policies in controllers	Configurable and programmable memory controllers
...	...

Some Open Source Tools

- Rowhammer
 - <https://github.com/CMU-SAFARI/rowhammer>
- Ramulator – Fast and Extensible DRAM Simulator
 - <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>

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

<i>Segment</i>	<i>DRAM Standards & Architectures</i>
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLD RAM3 (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

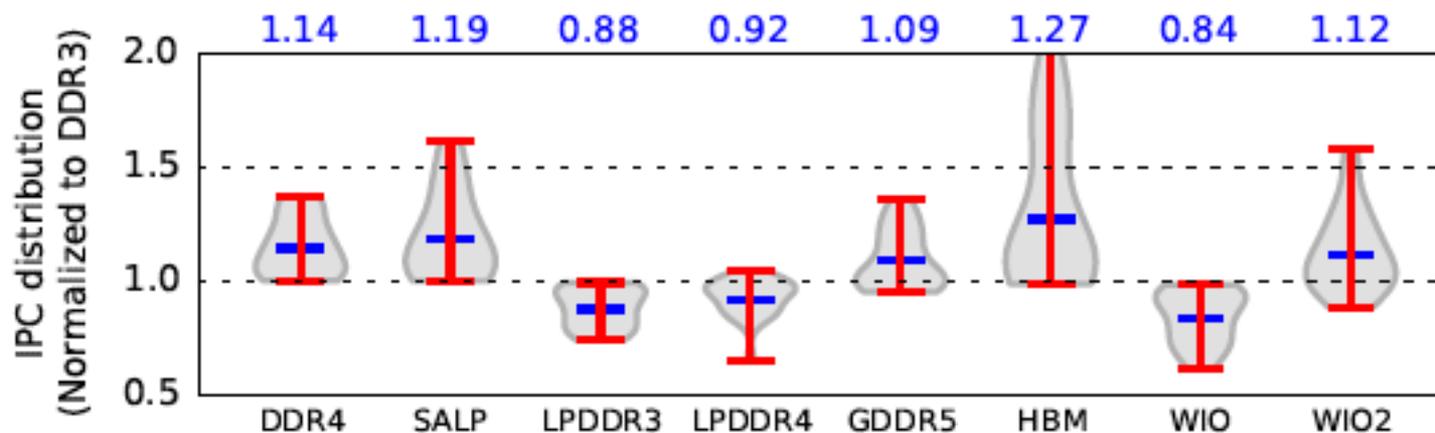
- Provides out-of-the box support for many DRAM standards:
 - DDR3/4, LPDDR3/4, GDDR5, WIO1/2, HBM, plus new proposals (SALP, AL-DRAM, TLDRAM, RowClone, and SARP)
- ~2.5X faster than fastest open-source simulator
- Modular and extensible to different standards

<i>Simulator</i> (clang -O3)	<i>Cycles (10⁶)</i>		<i>Runtime (sec.)</i>		<i>Req/sec (10³)</i>		<i>Memory</i> (MB)
	<i>Random</i>	<i>Stream</i>	<i>Random</i>	<i>Stream</i>	<i>Random</i>	<i>Stream</i>	
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

<i>Standard</i>	<i>Rate (MT/s)</i>	<i>Timing (CL-RCD-RP)</i>	<i>Data-Bus (Width×Chan.)</i>	<i>Rank-per-Chan</i>	<i>BW (GB/s)</i>
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



Across 22 workloads, simple CPU model

Figure 2. Performance comparison of DRAM standards

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>

Rethinking Memory Architecture

- Compute Capable Memory
- Refresh
- Reliability
- Latency
- Bandwidth
- Energy
- Memory Compression

Large DRAM Power in Modern Systems



>40% in POWER7 (Ware+, HPCA'10)



>40% in GPU (Paul+, ISCA'15)

Why Is Power Large?

- Design of DRAM uArchitecture
 - A lot of waste (granularity, latency, ...)
- High Voltage
 - Can we scale it down reliably?
- High Frequency
 - Can we scale it down with low performance impact?
- DRAM Refresh
- ...

Memory Dynamic Voltage/Freq. Scaling

- Howard David, Chris Fallin, Eugene Gorbатов, Ulf R. Hanebutte, and Onur Mutlu,
"Memory Power Management via Dynamic Voltage/Frequency Scaling"
Proceedings of the 8th International Conference on Autonomic Computing (ICAC), Karlsruhe, Germany, June 2011. Slides (pptx) (pdf)

Memory Power Management via Dynamic Voltage/Frequency Scaling

Howard David[†], Chris Fallin[§], Eugene Gorbатов[†], Ulf R. Hanebutte[†], Onur Mutlu[§]

[†]Intel Corporation
{howard.david,eugene.gorbatov,
ulf.r.hanebutte}@intel.com

[§]Carnegie Mellon University
{cfallin,onur}@cmu.edu

New Memory Architectures

- Compute Capable Memory
- Refresh
- Reliability
- Latency
- Bandwidth
- Energy
- Memory Compression

Readings on Memory Compression (I)

- Gennady Pekhimenko, Vivek Seshadri, Onur Mutlu, Philip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry,
"Base-Delta-Immediate Compression: Practical Data Compression for On-Chip Caches"
Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT), Minneapolis, MN, September 2012. [Slides \(pptx\)](#) [Source Code](#)

Base-Delta-Immediate Compression: Practical Data Compression for On-Chip Caches

Gennady Pekhimenko[†]
gpekhime@cs.cmu.edu

Vivek Seshadri[†]
vseshadr@cs.cmu.edu

Onur Mutlu[†]
onur@cmu.edu

Michael A. Kozuch^{*}
michael.a.kozuch@intel.com

Phillip B. Gibbons^{*}
phillip.b.gibbons@intel.com

Todd C. Mowry[†]
tcm@cs.cmu.edu

Readings on Memory Compression (II)

- Gennady Pekhimenko, Vivek Seshadri, Yoongu Kim, Hongyi Xin, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry, **"Linearly Compressed Pages: A Low-Complexity, Low-Latency Main Memory Compression Framework"**
Proceedings of the 46th International Symposium on Microarchitecture (MICRO), Davis, CA, December 2013. [[Slides \(pptx\) \(pdf\)](#)] [[Lightning Session Slides \(pptx\) \(pdf\)](#)] [[Poster \(pptx\) \(pdf\)](#)]

Linearly Compressed Pages: A Low-Complexity, Low-Latency Main Memory Compression Framework

Gennady Pekhimenko[†]
gpekhime@cs.cmu.edu

Vivek Seshadri[†]
vseshadr@cs.cmu.edu

Yoongu Kim[†]
yoongukim@cmu.edu

Hongyi Xin[†]
hxin@cs.cmu.edu

Onur Mutlu[†]
onur@cmu.edu

Phillip B. Gibbons^{*}
phillip.b.gibbons@intel.com

Michael A. Kozuch^{*}
michael.a.kozuch@intel.com

Todd C. Mowry[†]
tcm@cs.cmu.edu

[†]Carnegie Mellon University

^{*}Intel Labs Pittsburgh

Readings on Memory Compression (III)

- Gennady Pekhimenko, Tyler Huberty, Rui Cai, Onur Mutlu, Phillip P. Gibbons, Michael A. Kozuch, and Todd C. Mowry,
"Exploiting Compressed Block Size as an Indicator of Future Reuse"
Proceedings of the 21st International Symposium on High-Performance Computer Architecture (HPCA), Bay Area, CA, February 2015.
[[Slides \(pptx\)](#)] [[pdf](#)]

Exploiting Compressed Block Size as an Indicator of Future Reuse

Gennady Pekhimenko[†]
gpekhime@cs.cmu.edu

Tyler Huberty[†]
thuberty@alumni.cmu.edu

Rui Cai[†]
rcai@alumni.cmu.edu

Onur Mutlu[†]
onur@cmu.edu

Phillip B. Gibbons*
phillip.b.gibbons@intel.com

Michael A. Kozuch*
michael.a.kozuch@intel.com

Todd C. Mowry[†]
tcm@cs.cmu.edu

[†]Carnegie Mellon University

*Intel Labs Pittsburgh

Readings on Memory Compression (IV)

- Gennady Pekhimenko, Evgeny Bolotin, Nandita Vijaykumar, Onur Mutlu, Todd C. Mowry, and Stephen W. Keckler,
"A Case for Toggle-Aware Compression for GPU Systems"
Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (HPCA), Barcelona, Spain, March 2016.
[[Slides \(pptx\)](#)] [[pdf](#)]

A Case for Toggle-Aware Compression for GPU Systems

Gennady Pekhimenko[†], Evgeny Bolotin^{*}, Nandita Vijaykumar[†],
Onur Mutlu[†], Todd C. Mowry[†], Stephen W. Keckler^{*#}

[†]Carnegie Mellon University

^{*}NVIDIA

[#]University of Texas at Austin

Readings on Memory Compression (V)

- Nandita Vijaykumar, Gennady Pekhimenko, Adwait Jog, Abhishek Bhowmick, Rachata Ausavarungnirun, Chita Das, Mahmut Kandemir, Todd C. Mowry, and Onur Mutlu,
"A Case for Core-Assisted Bottleneck Acceleration in GPUs: Enabling Flexible Data Compression with Assist Warps"
Proceedings of the 42nd International Symposium on Computer Architecture (ISCA), Portland, OR, June 2015.
[[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Session Slides \(pptx\)](#)] [[pdf](#)]

A Case for Core-Assisted Bottleneck Acceleration in GPUs: Enabling Flexible Data Compression with Assist Warps

Nandita Vijaykumar Gennady Pekhimenko Adwait Jog[†] Abhishek Bhowmick
Rachata Ausavarungnirun Chita Das[†] Mahmut Kandemir[†] Todd C. Mowry Onur Mutlu

Carnegie Mellon University

[†] Pennsylvania State University

{nandita, abhowmick, rachata, onur}@cmu.edu

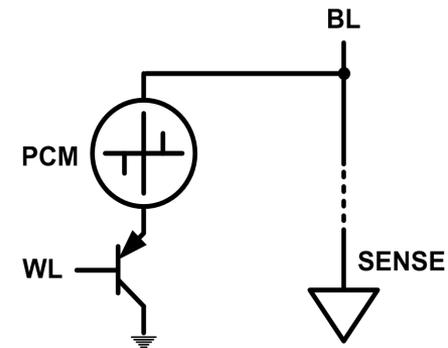
{gpekhime, tcm}@cs.cmu.edu

{adwait, das, kandemir}@cse.psu.edu

Emerging Technologies and Hybrid Memories

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 2009])
 - ❑ 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?**



Solution 2: Emerging Memory Technologies

- Lee+, “Architecting Phase Change Memory as a Scalable DRAM Alternative,” ISCA’09, CACM’10, IEEE 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,” TACO 2014.
- Ren+, “ThyNVM: Enabling Software-Transparent Crash Consistency in Persistent Memory Systems,” MICRO 2015.
- Chauhan+, “NVMove: Helping Programmers Move to Byte-Based Persistence,” INFLOW 2016.
- Li+, “Utility-Based Hybrid Memory Management,” CLUSTER 2017.
- Yu+, “Banshee: Bandwidth-Efficient DRAM Caching via Software/Hardware Cooperation,” MICRO 2017.

Promising Resistive Memory Technologies

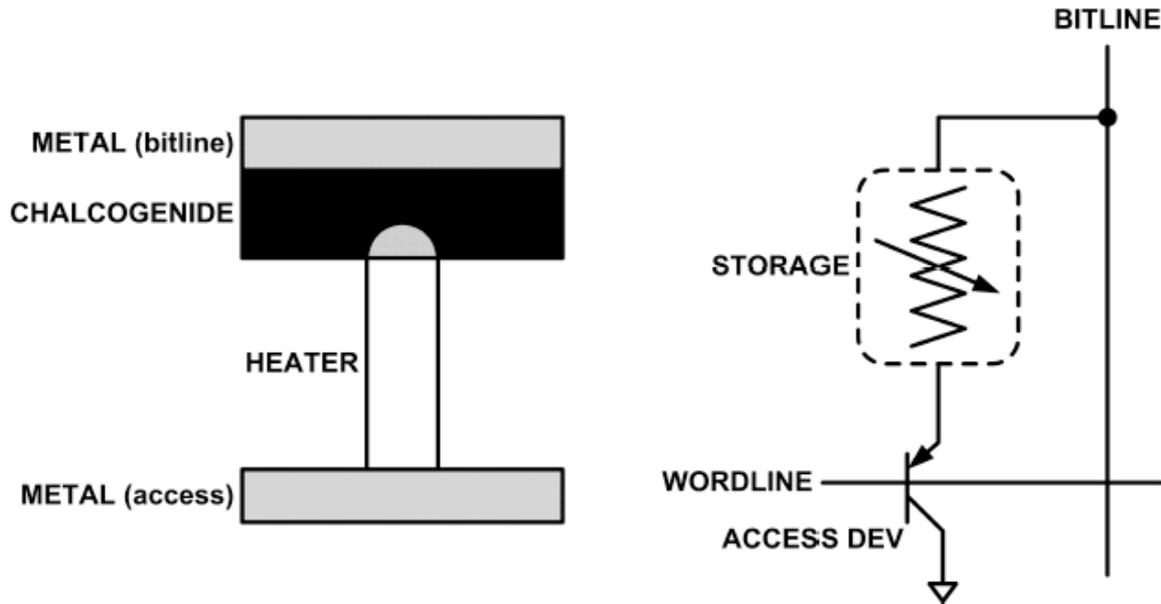
- PCM
 - Inject current to change **material phase**
 - Resistance determined by phase

- 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?

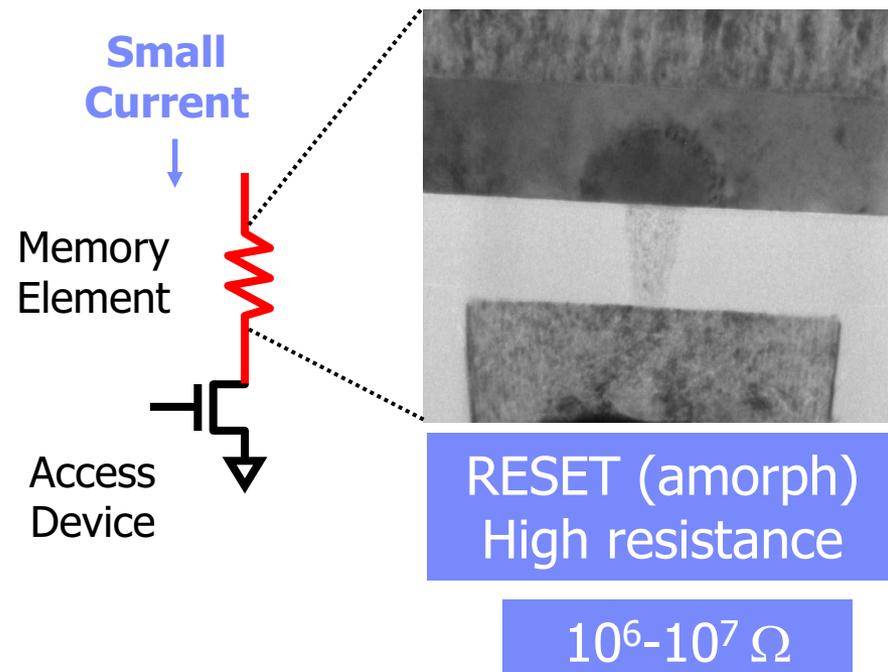
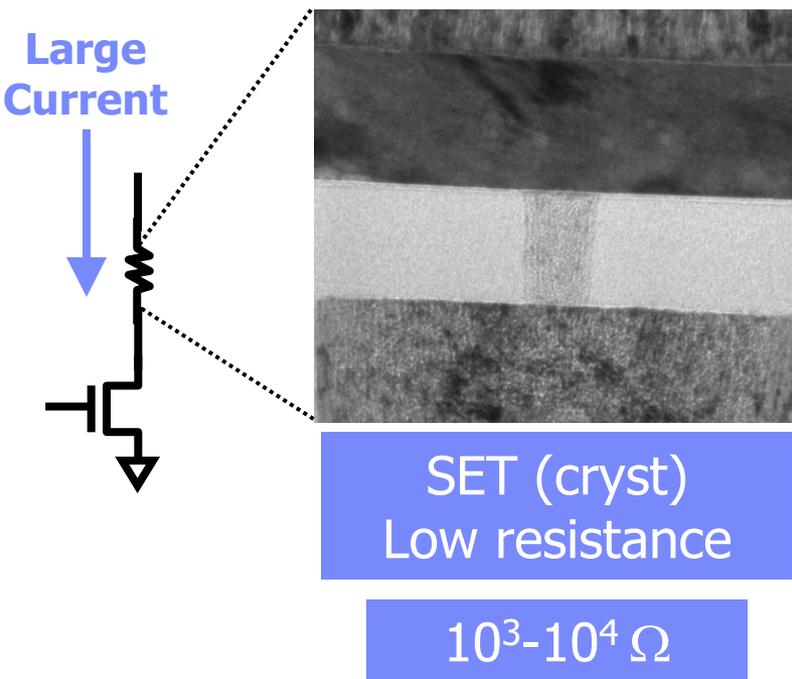
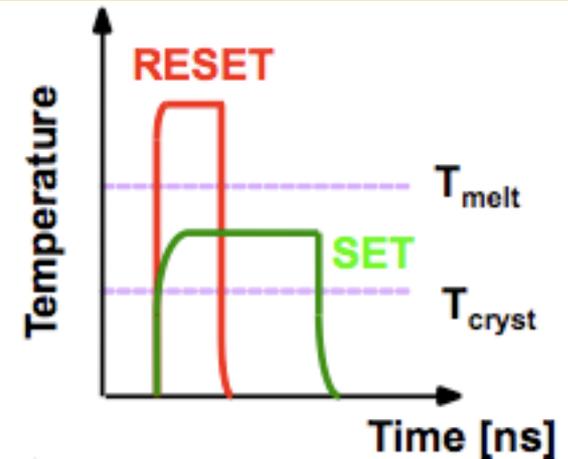
- Phase change material (chalcogenide glass) exists in two states:
 - Amorphous: Low optical reflexivity and high electrical resistivity
 - Crystalline: High optical reflexivity and low electrical resistivity



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.
- Lee et al., “Phase Change Technology and the Future of Main Memory,” IEEE Micro Top Picks 2010.

Table 1. Technology survey.

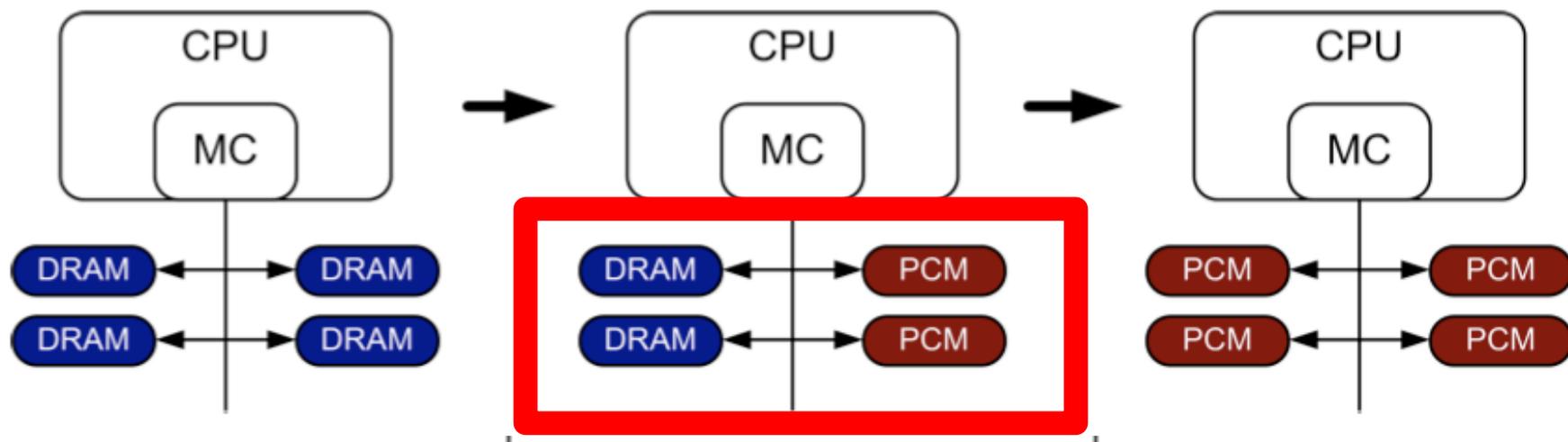
Parameter*	Published prototype									
	Horri ⁶	Ahn ¹²	Bedeschi ¹³	Oh ¹⁴	Pellizer ¹⁵	Chen ⁵	Kang ¹⁶	Bedeschi ⁹	Lee ¹⁰	Lee ²
Year	2003	2004	2004	2005	2006	2006	2006	2008	2008	**
Process, F (nm)	**	120	180	120	90	**	100	90	90	90
Array size (Mbytes)	**	64	8	64	**	**	256	256	512	**
Material	GST, N-d	GST, 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: Ge₂Sb₂Te₅; MLC: multilevel cells; N-d: nitrogen doped.

** This information is not available in the publication cited.

PCM-based Main Memory (I)

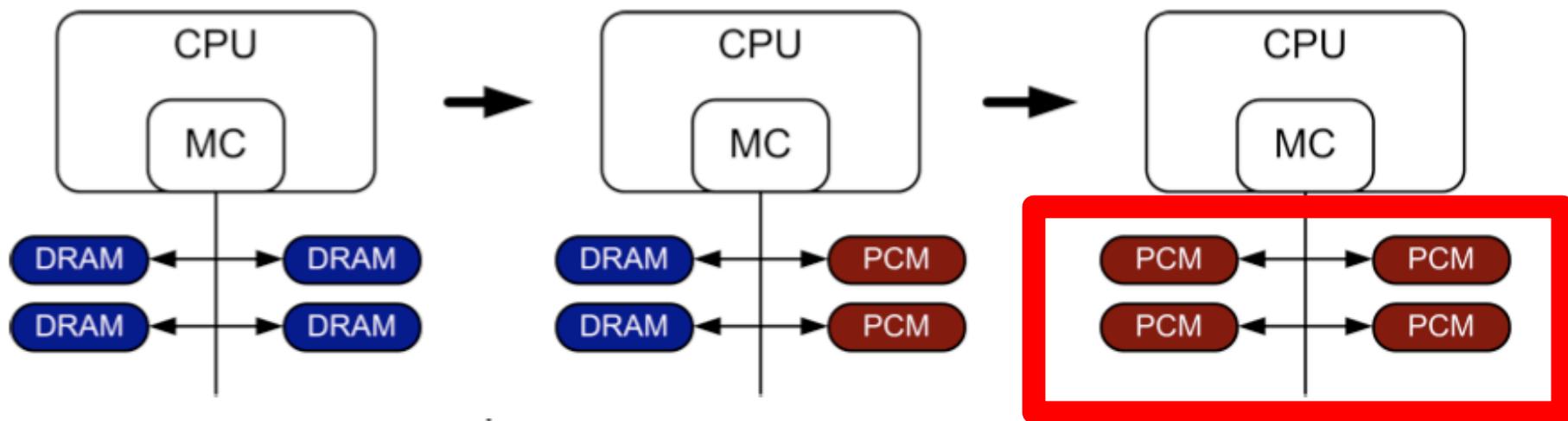
- How should PCM-based (main) memory be organized?



- **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 - 12 F^2 using BJT
- ▷ 1.5× DRAM

Latency

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

Endurance

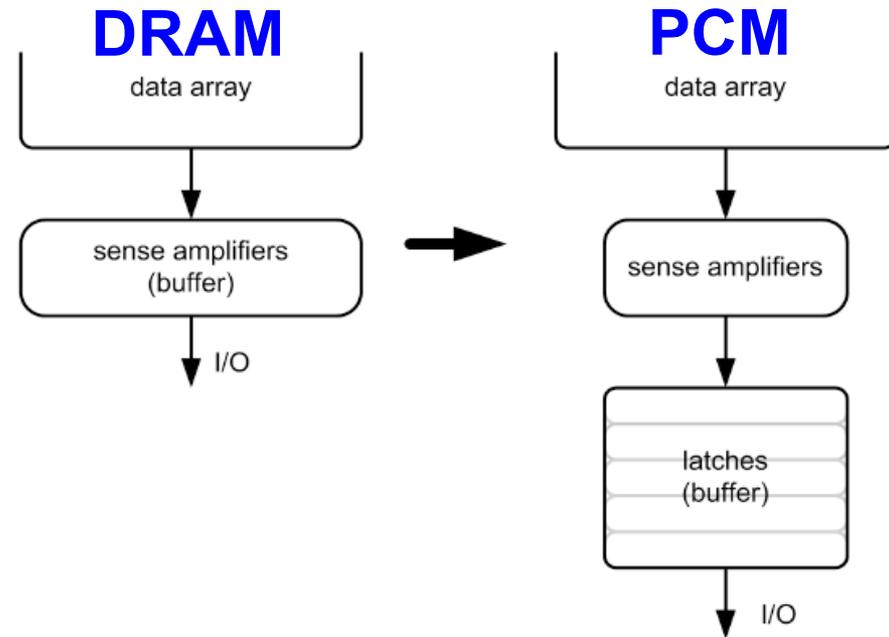
- ▷ 1E+08 writes
- ▷ 1E-08× DRAM

Energy

- ▷ 40 μ A Rd, 150 μ A Wr
- ▷ 2×, 43× DRAM

Architecting PCM to Mitigate Shortcomings

- Idea 1: Use multiple narrow row buffers in each PCM chip
→ Reduces array reads/writes → better endurance, latency, energy
- Idea 2: Write into array at cache block or word granularity
→ Reduces unnecessary wear



More on PCM As Main Memory

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

Architecting Phase Change Memory as a Scalable DRAM Alternative

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

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

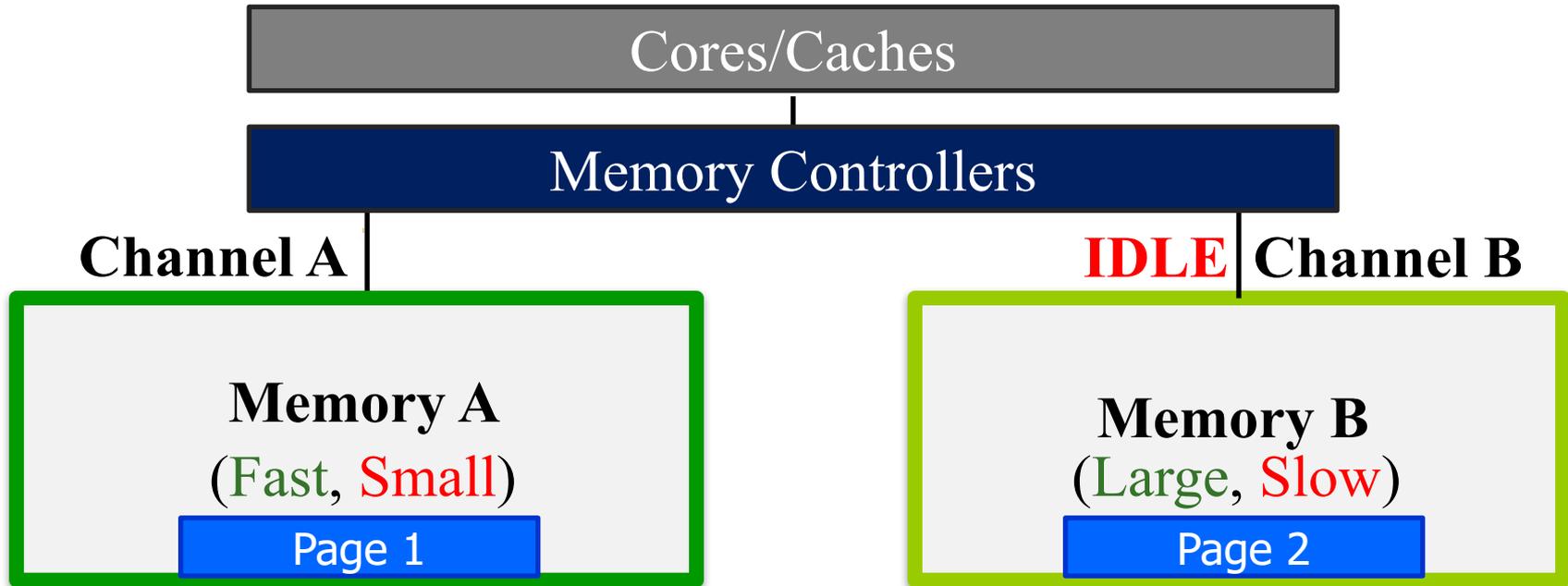
‡Computer Architecture Laboratory
Carnegie Mellon University
Pittsburgh, PA
onur@cmu.edu

More on PCM As Main Memory (II)

- Benjamin C. Lee, Ping Zhou, Jun Yang, Youtao Zhang, Bo Zhao, Engin Ipek, Onur Mutlu, and Doug Burger,
"Phase Change Technology and the Future of Main Memory"
IEEE Micro, Special Issue: Micro's Top Picks from 2009 Computer Architecture Conferences (MICRO TOP PICKS), Vol. 30, No. 1, pages 60-70, January/February 2010.

PHASE-CHANGE TECHNOLOGY AND THE FUTURE OF MAIN MEMORY

Data Placement in Hybrid Memory



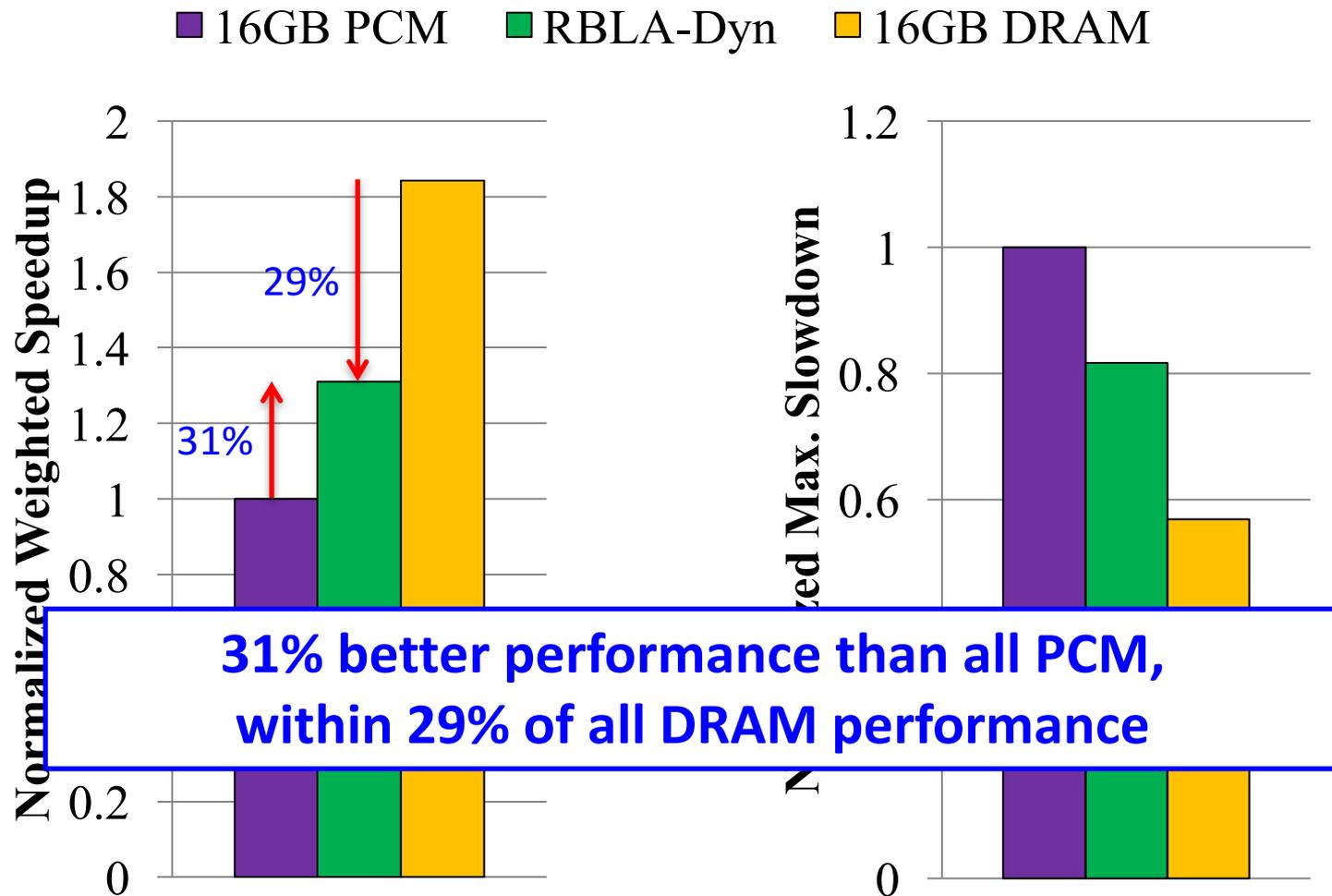
Which memory do we place each page in, to maximize system performance?

- Memory A is fast, but small
- Load should be balanced on both channels
- Page migrations have performance and energy overhead

Data Placement Between DRAM and PCM

- Idea: Characterize data access patterns and guide data placement in hybrid memory
- Streaming accesses: As fast in PCM as in DRAM
- Random accesses: Much faster in DRAM
- Idea: Place random access data with some reuse in DRAM; streaming data in PCM
- Yoon+, "Row Buffer Locality-Aware Data Placement in Hybrid Memories," ICCD 2012 Best Paper Award.

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



More on Hybrid Memory Data Placement

- HanBin Yoon, Justin Meza, Rachata Ausavarungnirun, Rachael Harding, and Onur Mutlu,
"Row Buffer Locality Aware Caching Policies for Hybrid Memories"

Proceedings of the 30th IEEE International Conference on Computer Design (ICCD), Montreal, Quebec, Canada, September 2012. Slides (pptx) (pdf)

Row Buffer Locality Aware Caching Policies for Hybrid Memories

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

Weaknesses of Existing Solutions

- They are all **heuristics** that consider only a ***limited part of memory access behavior***
- **Do not *directly* capture the overall system performance impact** of data placement decisions
- Example: None capture **memory-level parallelism** (MLP)
 - Number of ***concurrent memory requests*** from the same application when a page is accessed
 - Affects how much page migration helps performance

Importance of Memory-Level Parallelism

Before migration:

requests to Page 1 

After migration:

requests to Page 1 

T

time **Migrating one page**
reduces stall time by T

Before migration:

requests to Page 2 

requests to Page 3 

After migration:

requests to Page 2 

requests to Page 3 

time **Must migrate two pages**
to reduce stall time by T:
migrating one page alone
does not help

Page migration decisions **need to consider MLP**

Our Goal [CLUSTER 2017]

A **generalized** mechanism that

1. Directly estimates the **performance benefit of migrating a page** between **any two types of memory**
2. Places **only** the **performance-critical data** in the fast memory

Utility-Based Hybrid Memory Management

- A memory manager that works for *any* hybrid memory
 - e.g., DRAM-NVM, DRAM-RLDRAM
- **Key Idea**
 - For each page, use **comprehensive** characteristics to calculate estimated **utility** (i.e., performance impact) of migrating page from one memory to the other in the system
 - **Migrate only pages with the highest utility** (i.e., pages that improve system performance the most when migrated)
- Li+, “Utility-Based Hybrid Memory Management”, CLUSTER 2017.

Key Mechanisms of UH-MEM

- For each page, estimate **utility** using a **performance model**
 - **Application stall time reduction**

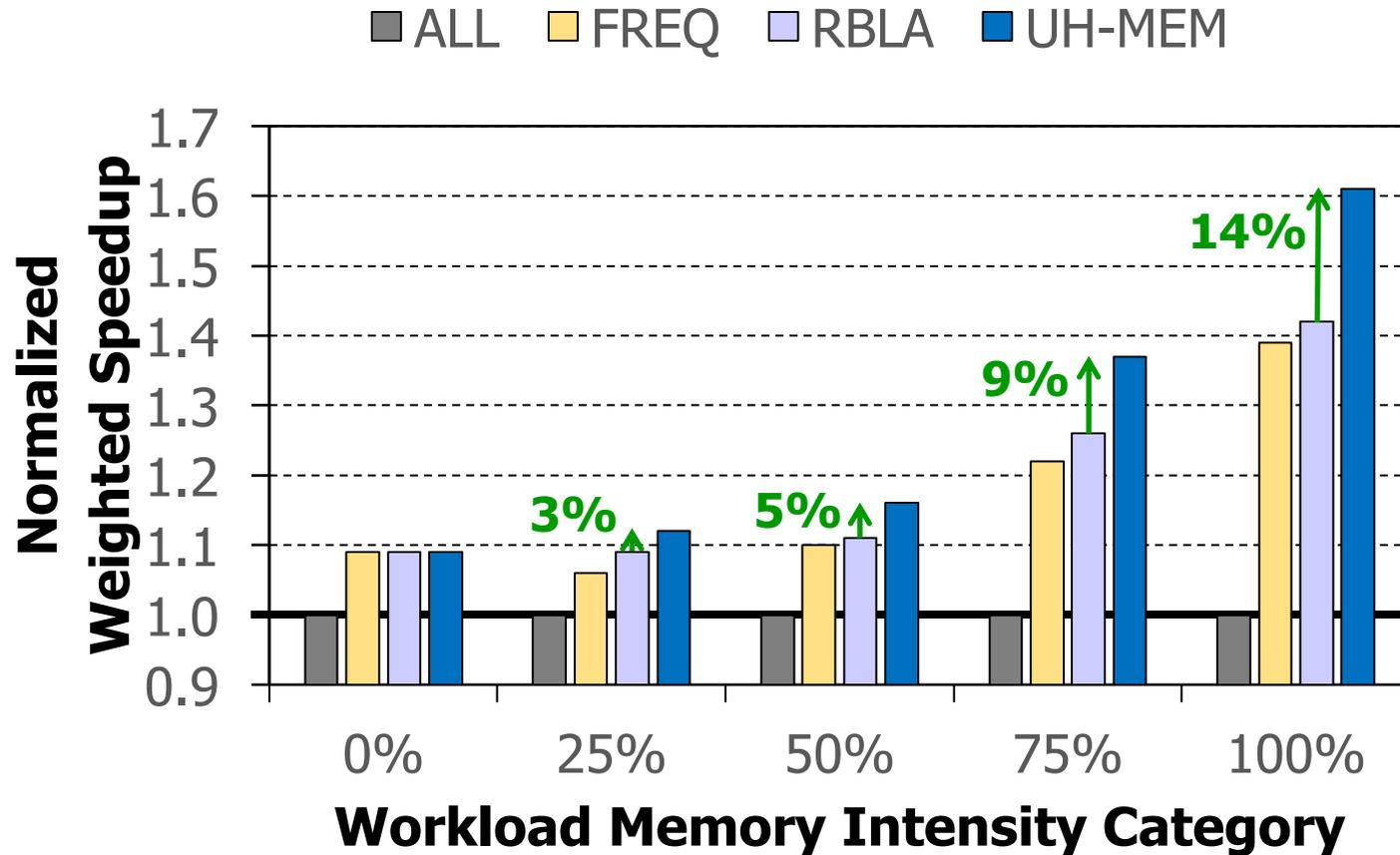
How much would migrating a page benefit the performance of the application that the page belongs to?
 - **Application performance sensitivity**

How much does the improvement of a single application's performance increase the *overall* system performance?

$$Utility = \Delta StallTime_i \times Sensitivity_i$$

- **Migrate** only pages whose utility **exceed the migration threshold** from slow memory to fast memory
- Periodically **adjust migration threshold**

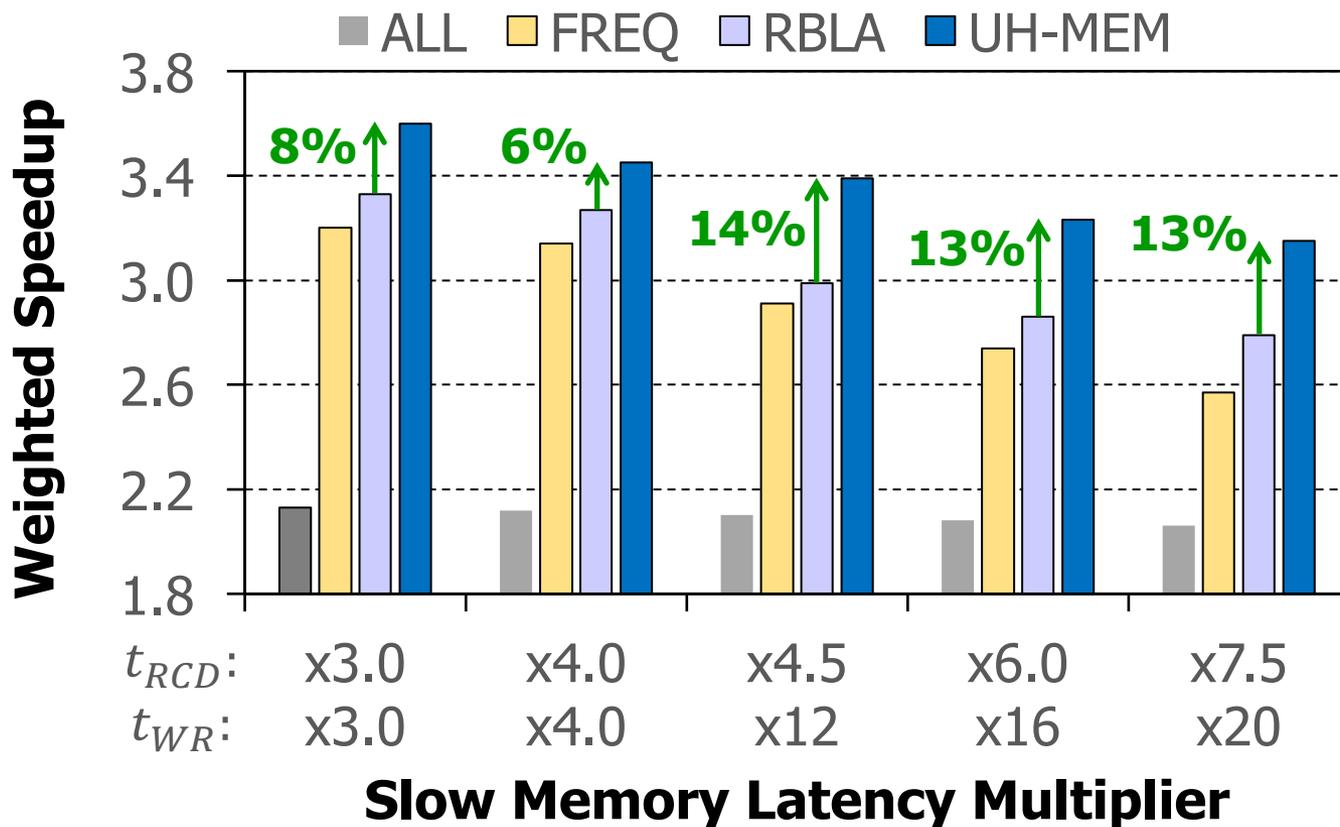
Results: System Performance



UH-MEM improves system performance
over the best state-of-the-art hybrid memory manager

Results: Sensitivity to Slow Memory Latency

- We vary t_{RCD} and t_{WR} of the slow memory



UH-MEM improves system performance for a wide variety of hybrid memory systems

Crash Consistency

One Key Challenge in Persistent Memory

- How to ensure consistency of system/data if all memory is persistent?
- Two extremes
 - Programmer transparent: Let the system handle it
 - Programmer only: Let the programmer handle it
- Many alternatives in-between...

CHALLENGE: CRASH CONSISTENCY

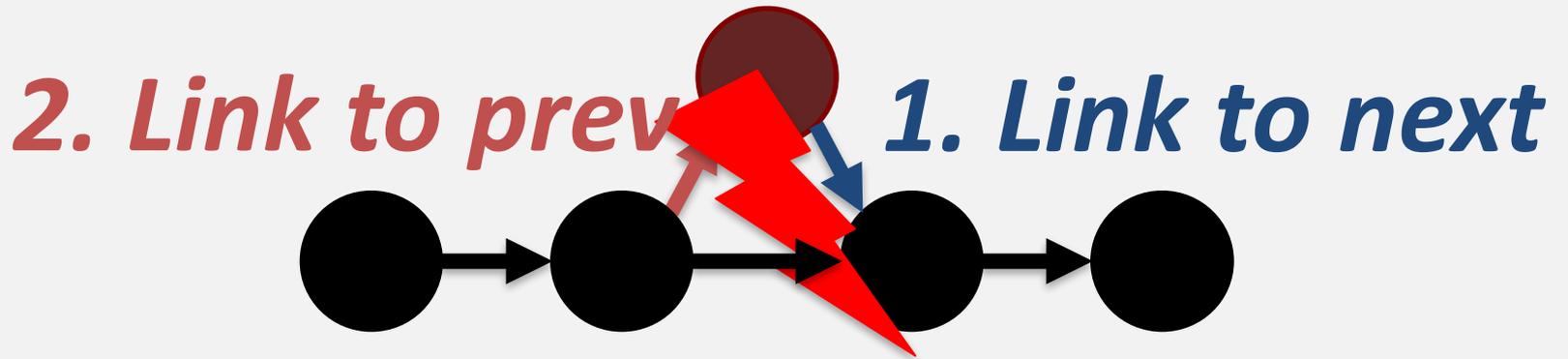


Persistent Memory System

System crash can result in permanent data corruption in NVM

CRASH CONSISTENCY PROBLEM

Example: Add a node to a linked list



**System crash can result in
inconsistent memory state**

CURRENT SOLUTIONS

Explicit interfaces to manage consistency

– NV-Heaps [ASPLOS'11], BPFS [SOSP'09], Mnemosyne [ASPLOS'11]

```
AtomicBegin {  
    Insert a new node;  
} AtomicEnd;
```

Limits adoption of NVM

Have to rewrite code with clear partition
between volatile and non-volatile data

Burden on the programmers

OUR APPROACH: ThyNVM

Goal:
**Software transparent consistency in
persistent memory systems**

ThyNVM: Summary

A new hardware-based checkpointing mechanism

- **Checkpoints** at *multiple granularities* to reduce both checkpointing latency and metadata overhead
- **Overlaps** *checkpointing* and *execution* to reduce checkpointing latency
- **Adapts** to *DRAM and NVM* characteristics

Performs within **4.9%** of an *idealized DRAM* with zero cost consistency

End of Backup Slides

Brief Self Introduction



■ Onur Mutlu

- ❑ Full Professor @ ETH Zurich CS, since September 2015
- ❑ Strecker Professor @ Carnegie Mellon University ECE/CS, 2009-2016, 2016-...
- ❑ PhD from UT-Austin, worked @ Google, VMware, Microsoft Research, Intel, AMD
- ❑ <https://people.inf.ethz.ch/omutlu/>
- ❑ omutlu@gmail.com (Best way to reach me)
- ❑ <https://people.inf.ethz.ch/omutlu/projects.htm>

■ Research, Education, Consulting in

- ❑ Computer architecture and systems, bioinformatics
- ❑ Memory and storage systems, emerging technologies
- ❑ Many-core systems, heterogeneous systems, core design
- ❑ Interconnects
- ❑ Hardware/software interaction and co-design (PL, OS, Architecture)
- ❑ Predictable and QoS-aware systems
- ❑ Hardware fault tolerance and security
- ❑ Algorithms and architectures for genome analysis