Memory Reliability, Security & Beyond

Three Key Issues in Modern Systems

Onur Mutlu
onur.mutlu@inf.ethz.ch
https://people.inf.ethz.ch/omutlu

June 18, 2017

Design Automation Summer School @ DAC 2017

SAFARI

ETH zürich

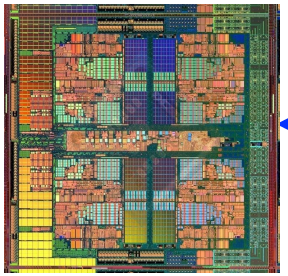
Brief Self Introduction

- Onur Mutlu
  - Full Professor @ ETH Zurich CS, since September'15, started May'16
  - 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)
  - Publications: 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
  - …

Current Research Focus Areas

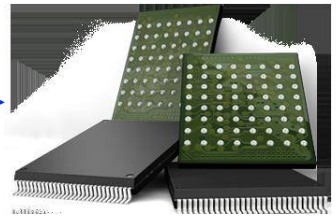
Research Focus: Computer architecture, HW/SW, bioinformatics

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

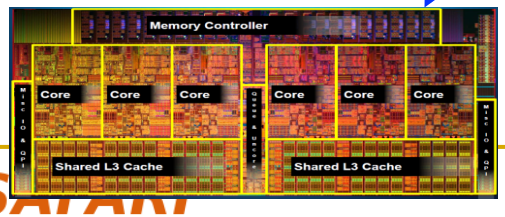


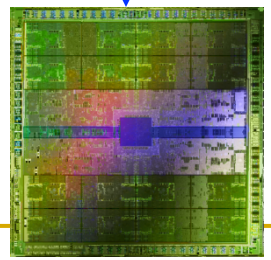
Heterogeneous Processors and Accelerators

Hybrid Main Memory



Persistent Memory/Storage





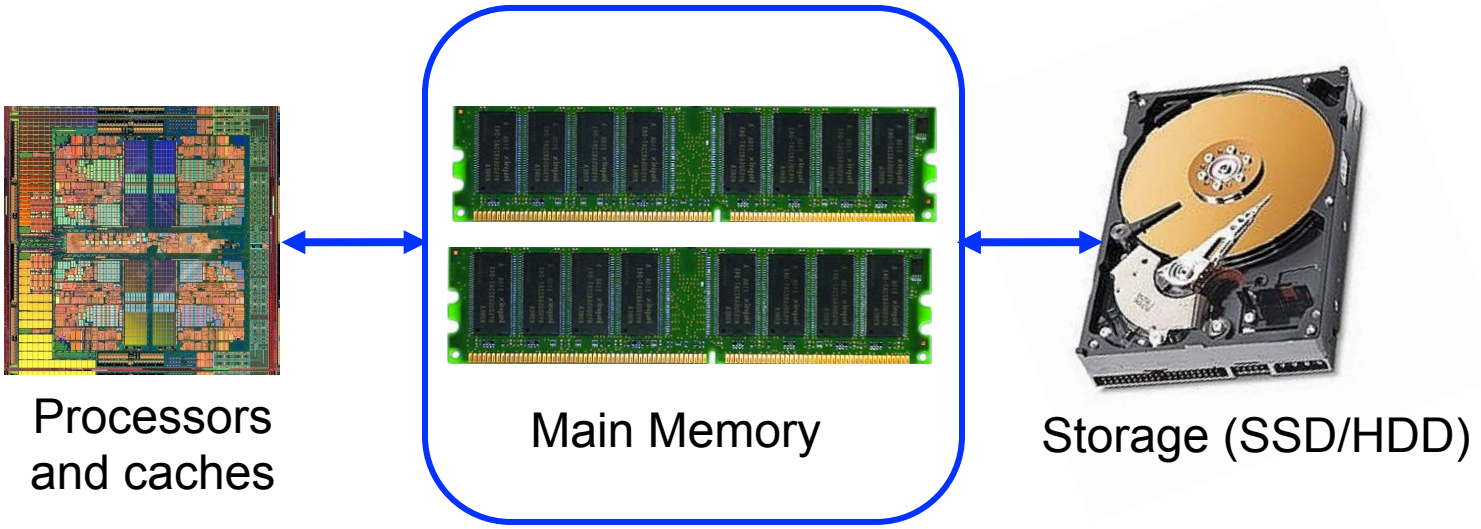
General Purpose GPUs

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

Current Research Focus Areas (II)

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

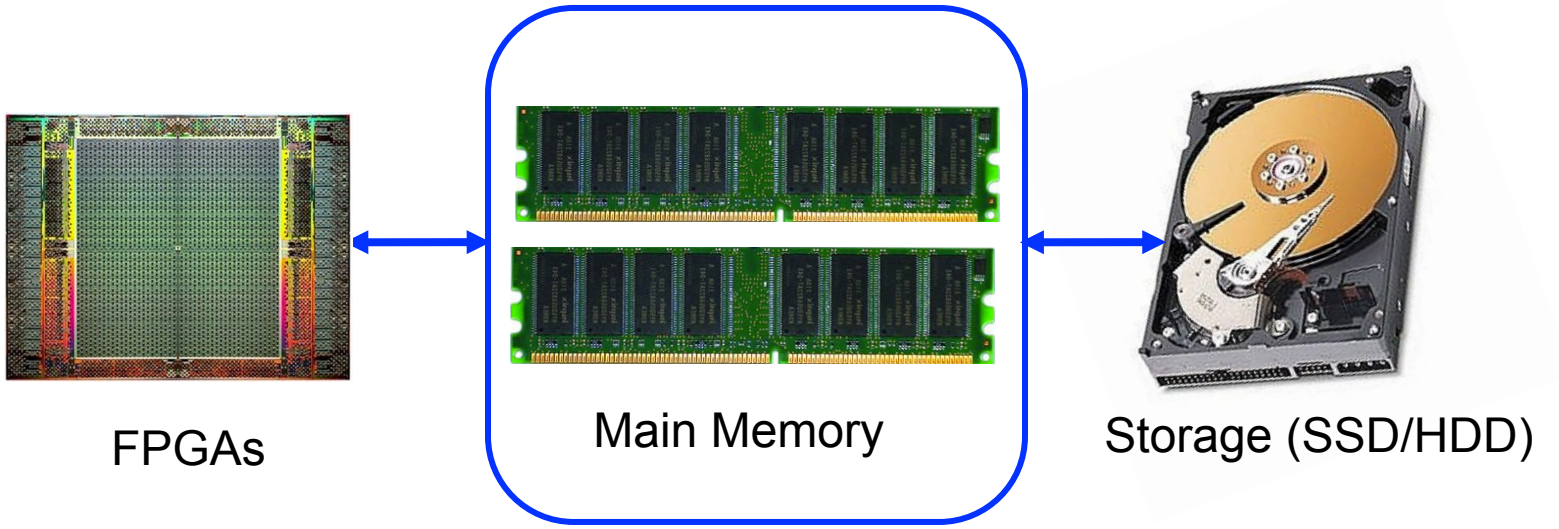
The Main Memory System



- 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

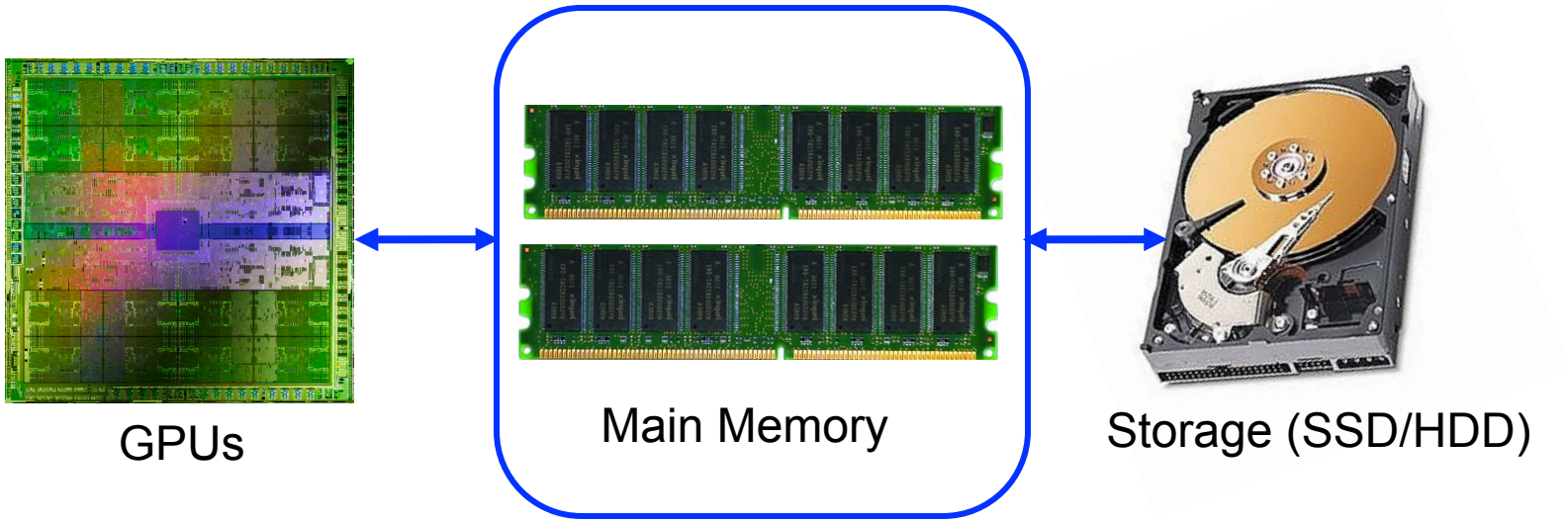
The Main Memory System



- 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

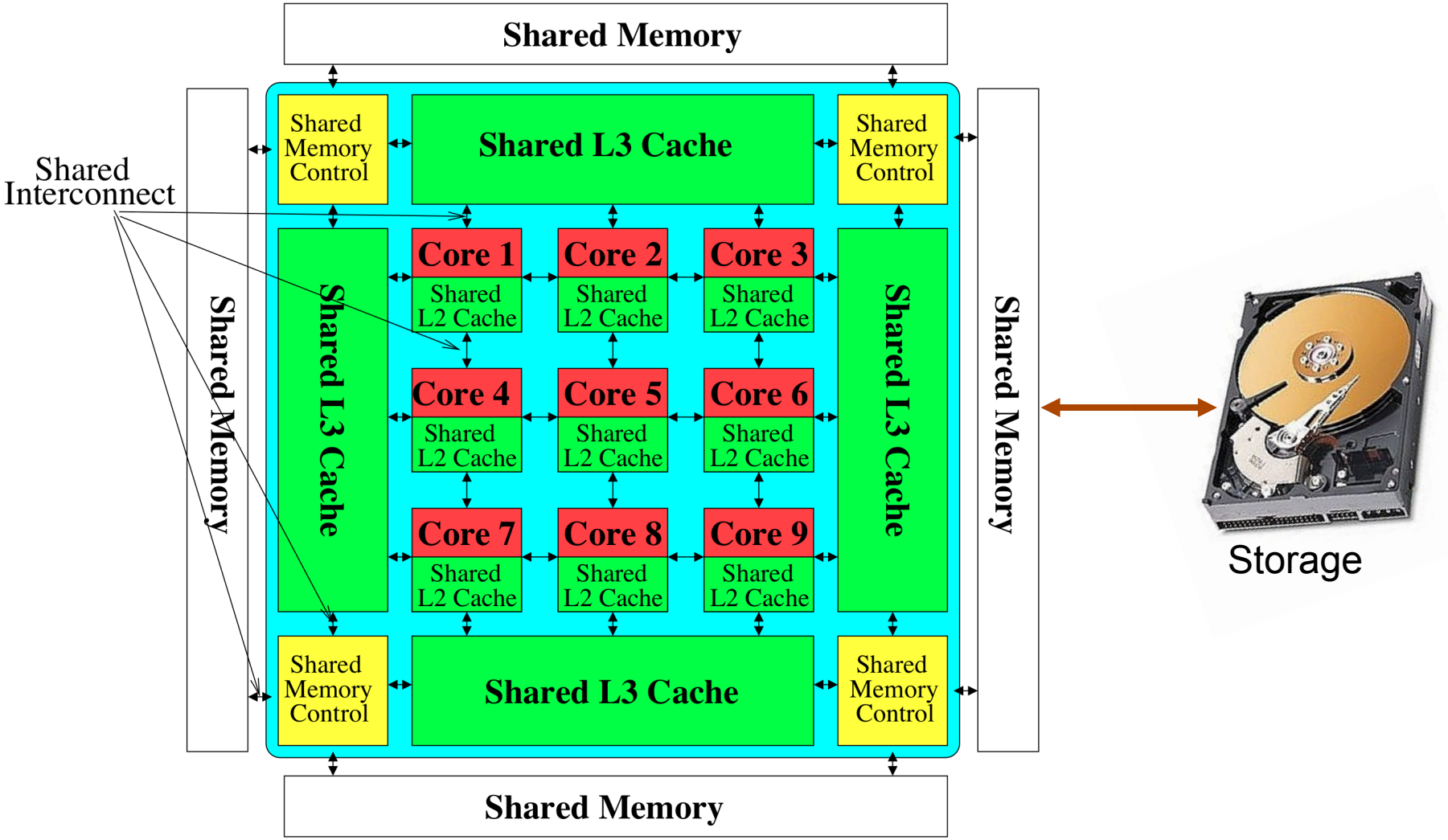
The Main Memory System



- 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

Agenda

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

Major Trends Affecting Main Memory (I)

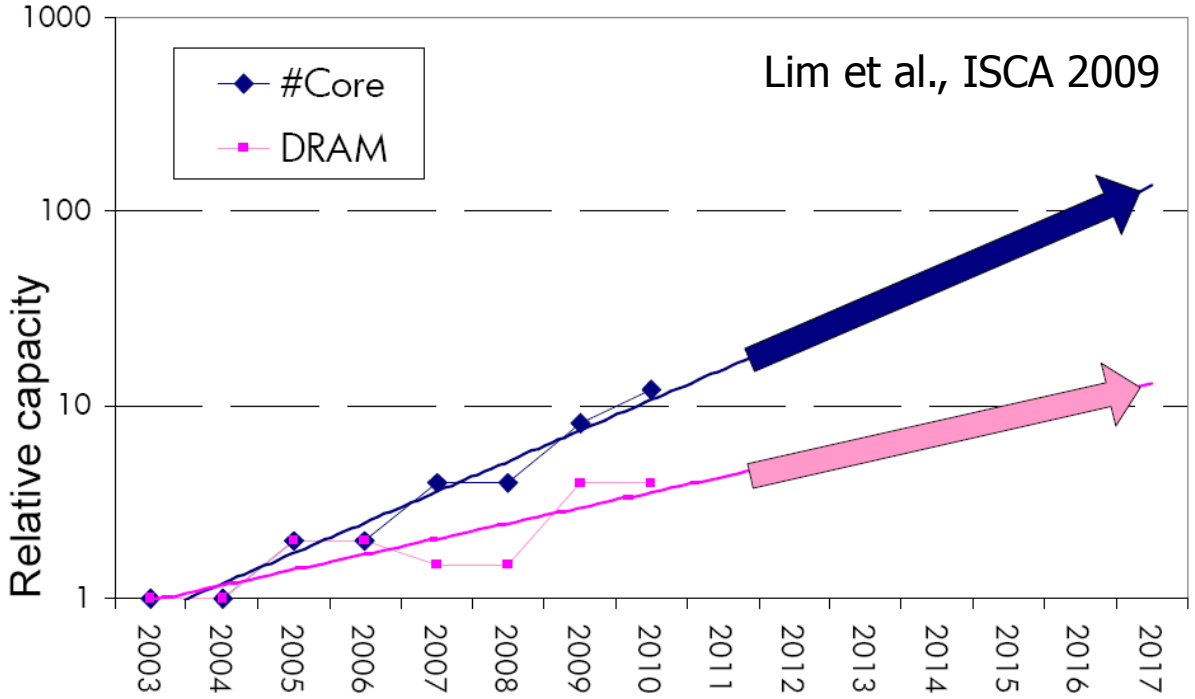
- Need for main memory capacity, bandwidth, QoS increasing
- Main memory energy/power is a key system design concern
- DRAM technology scaling is ending

Major Trends Affecting Main Memory (II)

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

Example: The Memory Capacity Gap

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



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

Major Trends Affecting Main Memory (III)

- Need for main memory capacity, bandwidth, QoS increasing

- Main memory energy/power is a key system design concern
  - ~40-50% energy spent in off-chip memory hierarchy [Lefurgy, IEEE Computer'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

Agenda

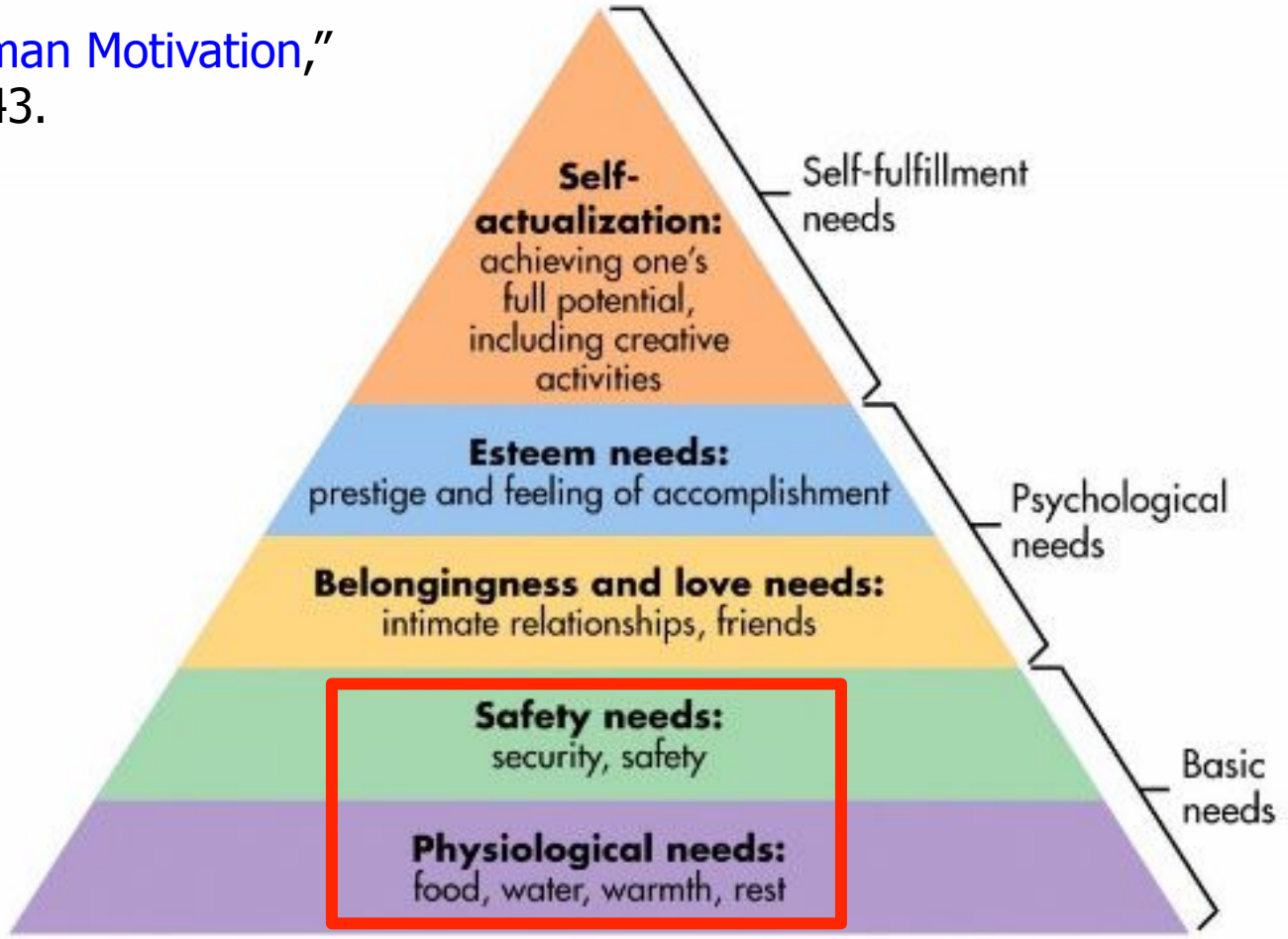
- Major Trends Affecting Main Memory
- The Memory Scaling Problem and Solution Directions
  - New Memory Architectures
  - Enabling Emerging Technologies
- Cross-Cutting Principles
- Summary

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.



- We need to start with reliability and security…

Source: https://www.simplypsychology.org/maslow.html

Source: http://www.technologystudent.com/struct1/tacom1.png

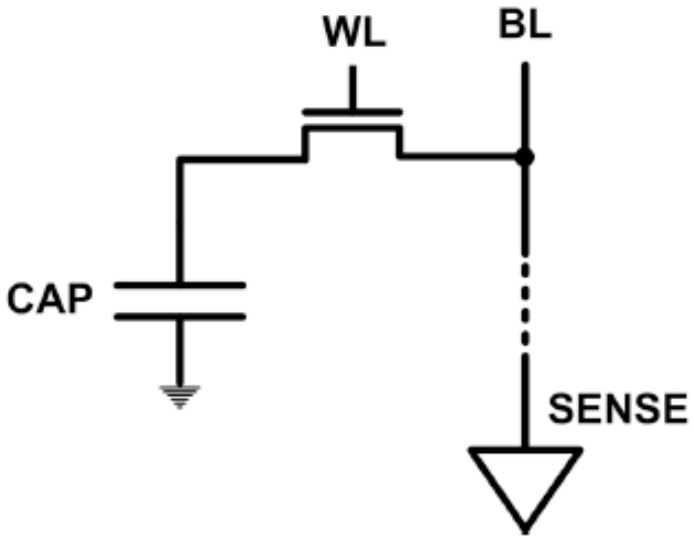
Source: AP

Security is about preventing unforeseen consequences

Source: https://s-media-cache-ak0.pinimg.com/originals/48/09/54/4809543a9c7700246a0cf8acdae27abf.jpg

The DRAM Scaling Problem

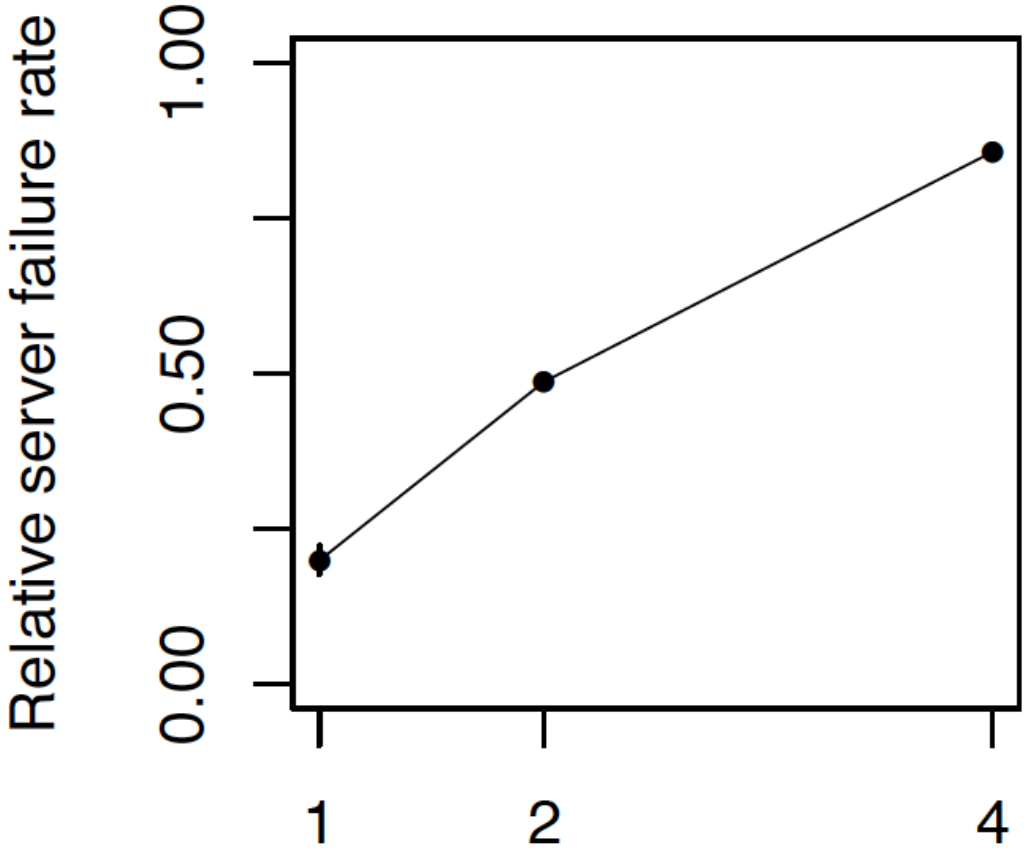
- DRAM stores charge in a capacitor (charge-based memory)
  - Capacitor must be large enough for reliable sensing
  - Access transistor should be large enough for low leakage and high retention time
  - Scaling beyond 40-35nm (2013) is challenging [ITRS, 2009]



- 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

Infrastructures to Understand Such Issues

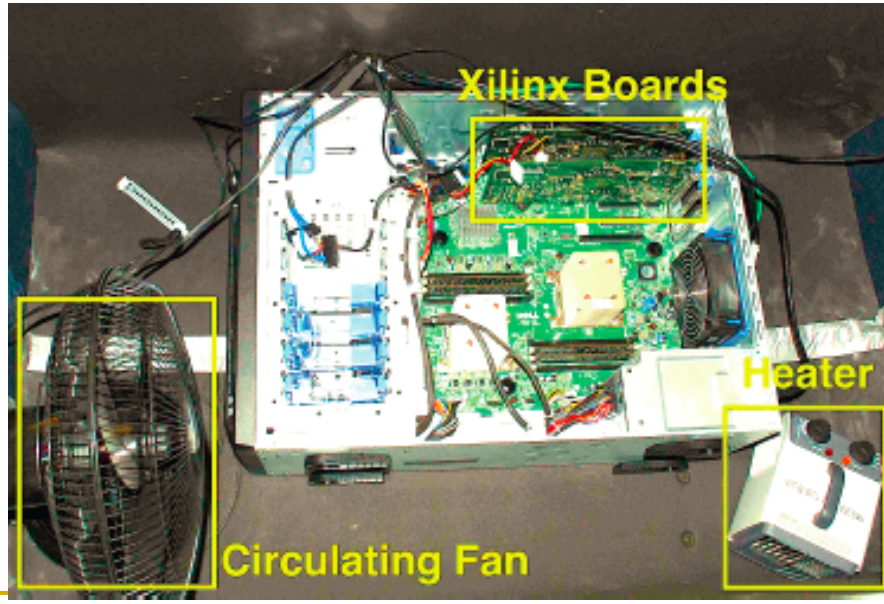
An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

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

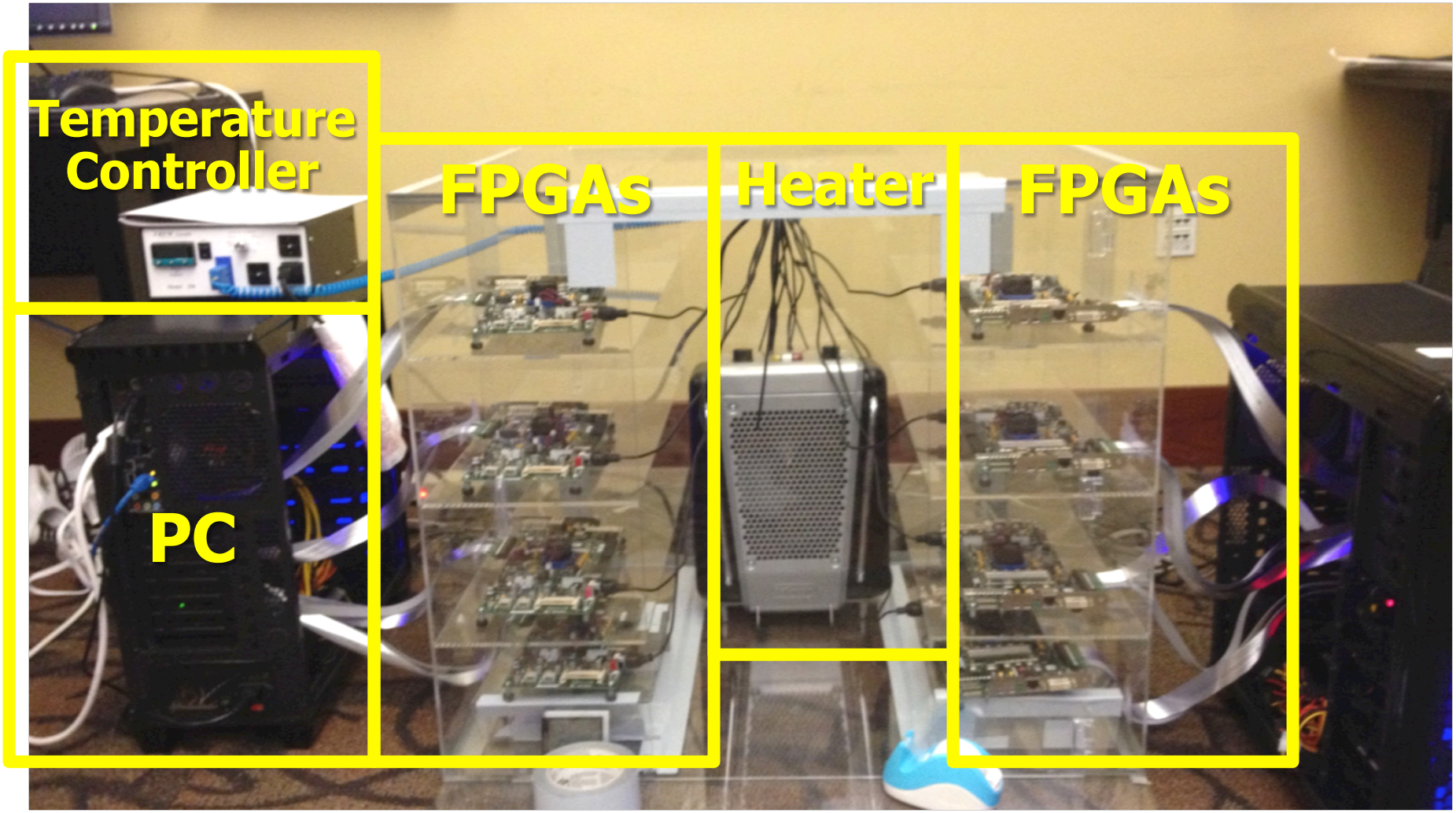
Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

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



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

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

SHARE

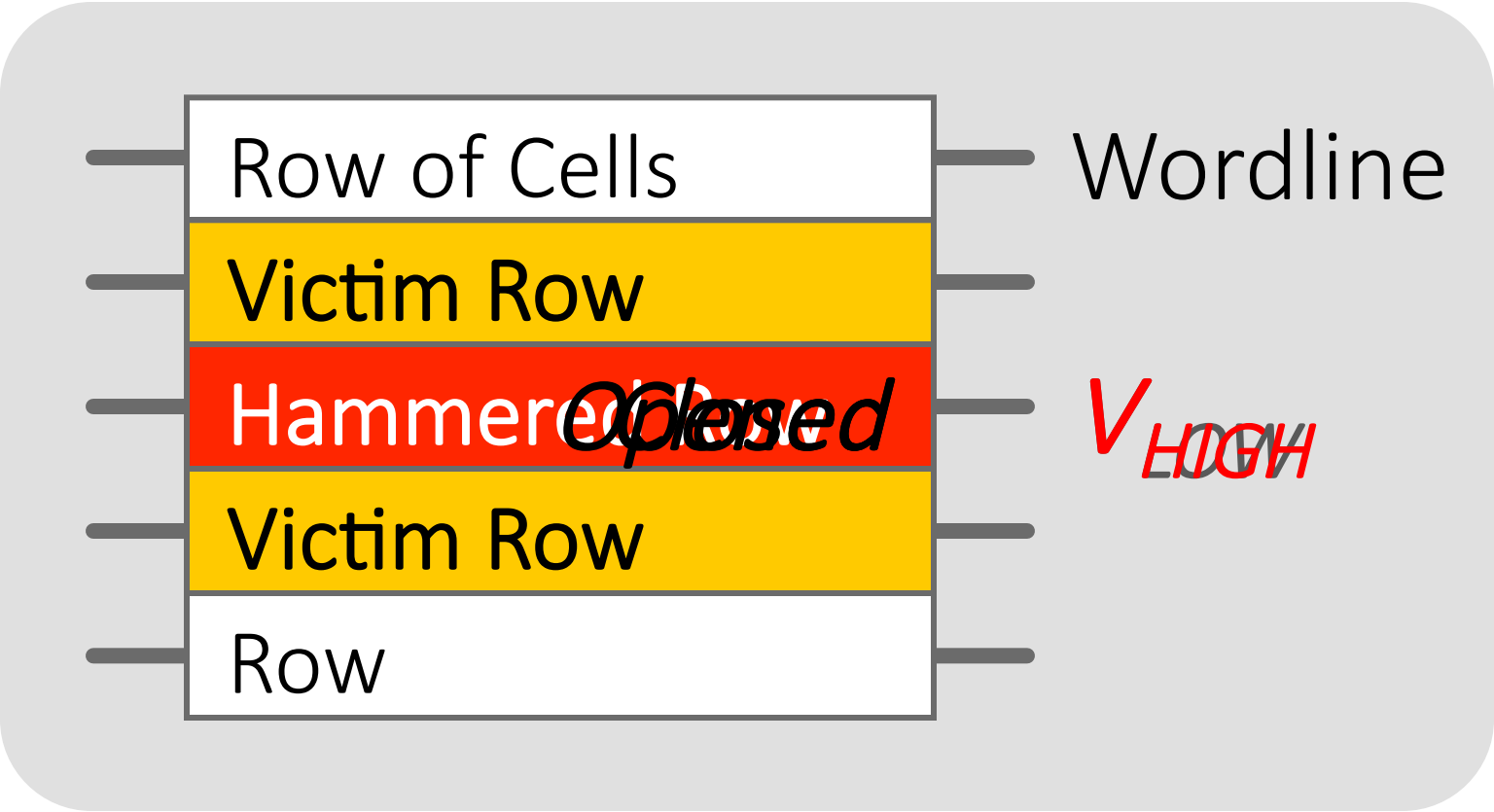
SHARE
18276

TWEET

ANDY GREENBERG SECURITY 08.31.16 7:00 AM

FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS

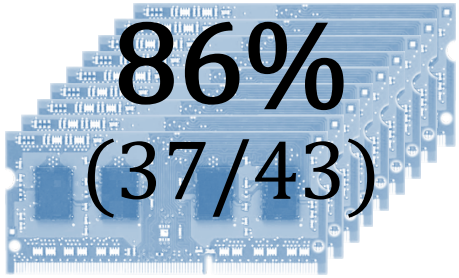
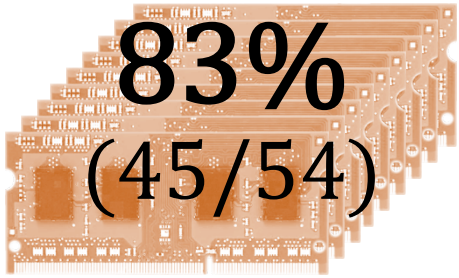
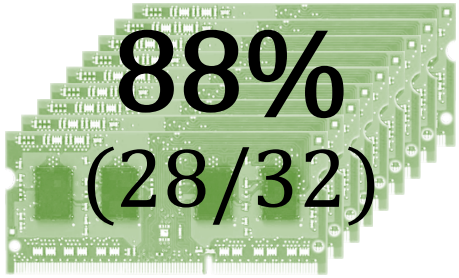
Modern DRAM is Prone to Disturbance Errors



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

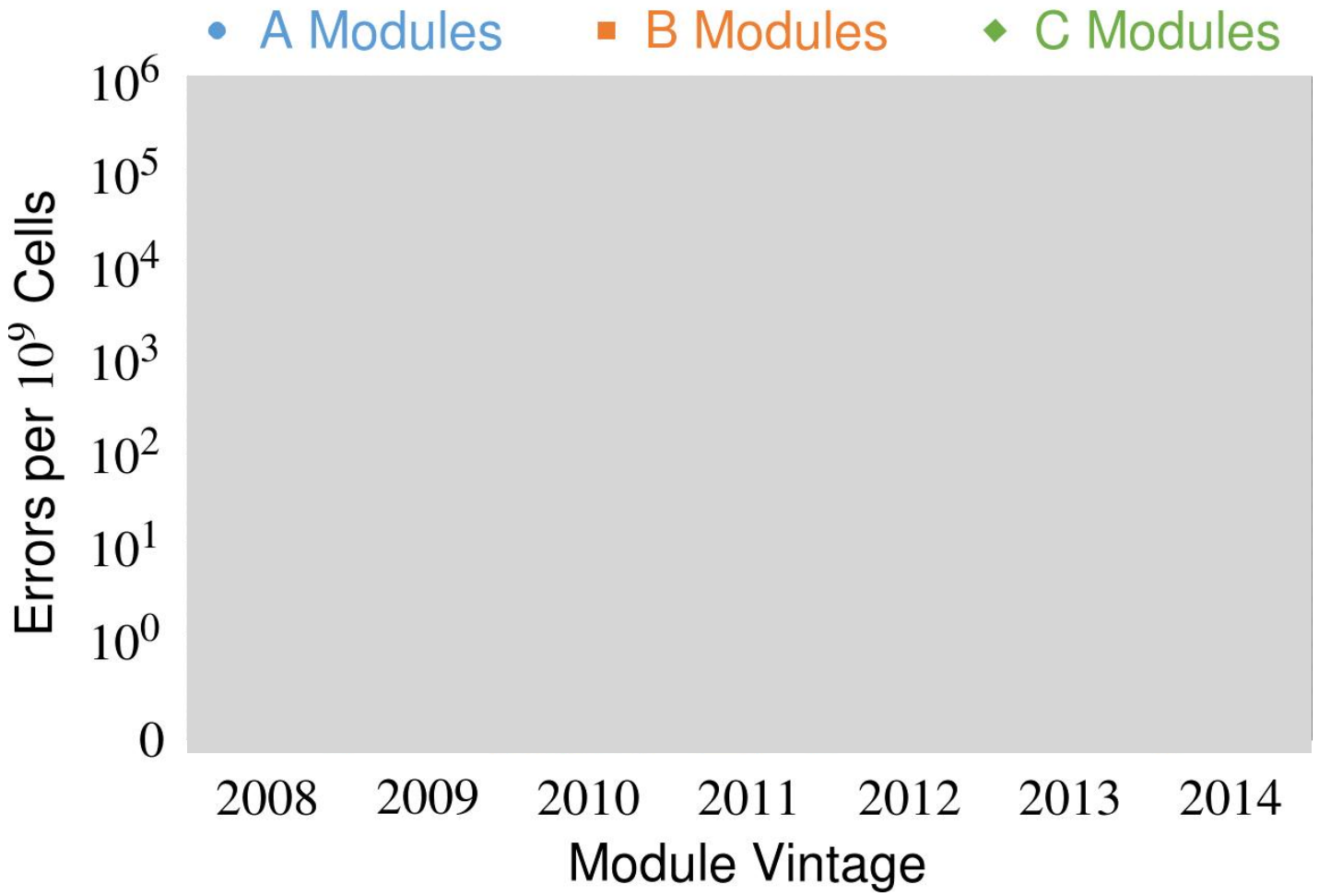
Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors, (Kim et al., ISCA 2014)

Most DRAM Modules Are at Risk

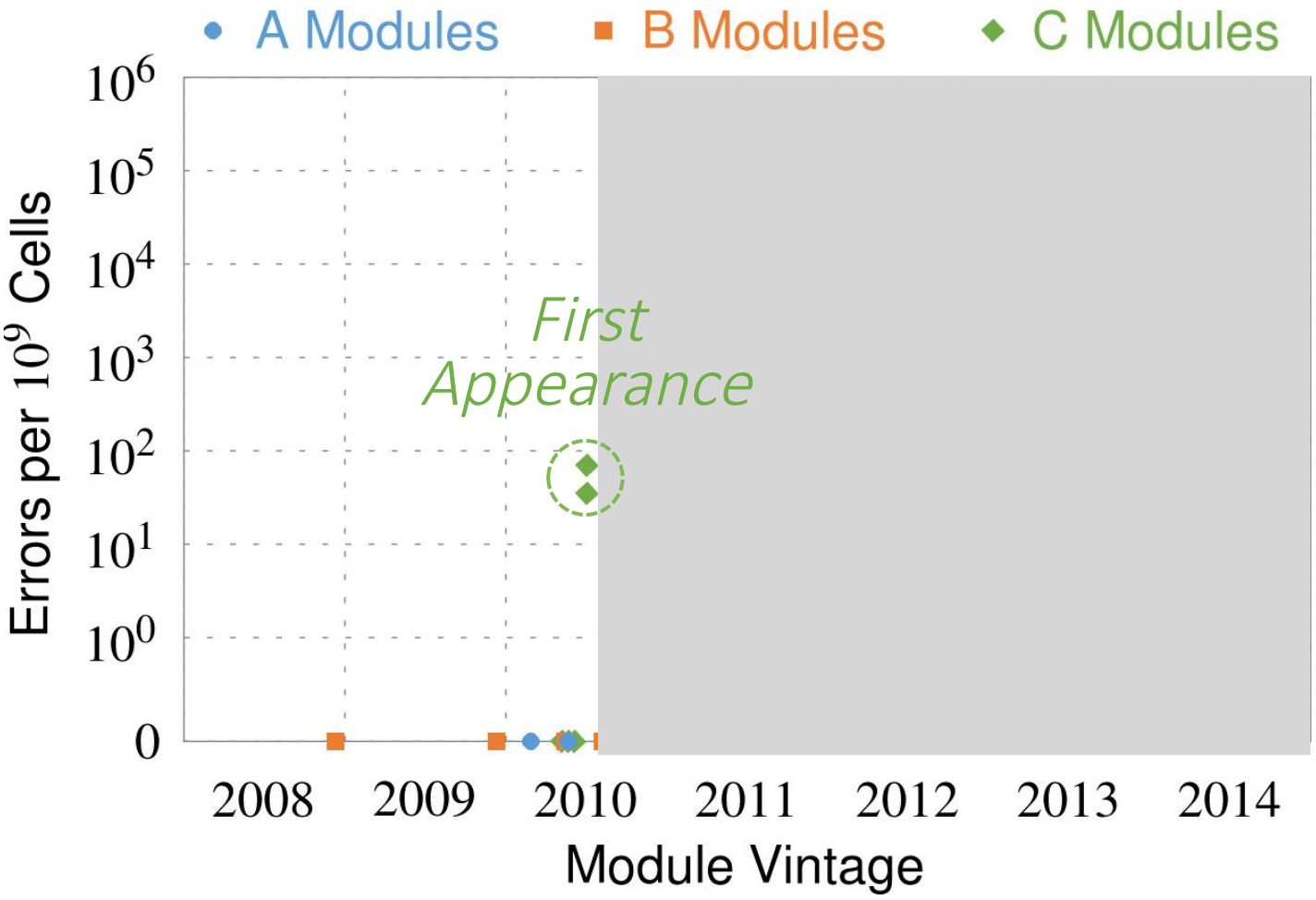
A company	B company	C company
 86% (37/43)	 83% (45/54)	 88% (28/32)
Up to 1.0×10^{7} errors	Up to 2.7×10^{6} errors	Up to 3.3×10^{5} errors

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

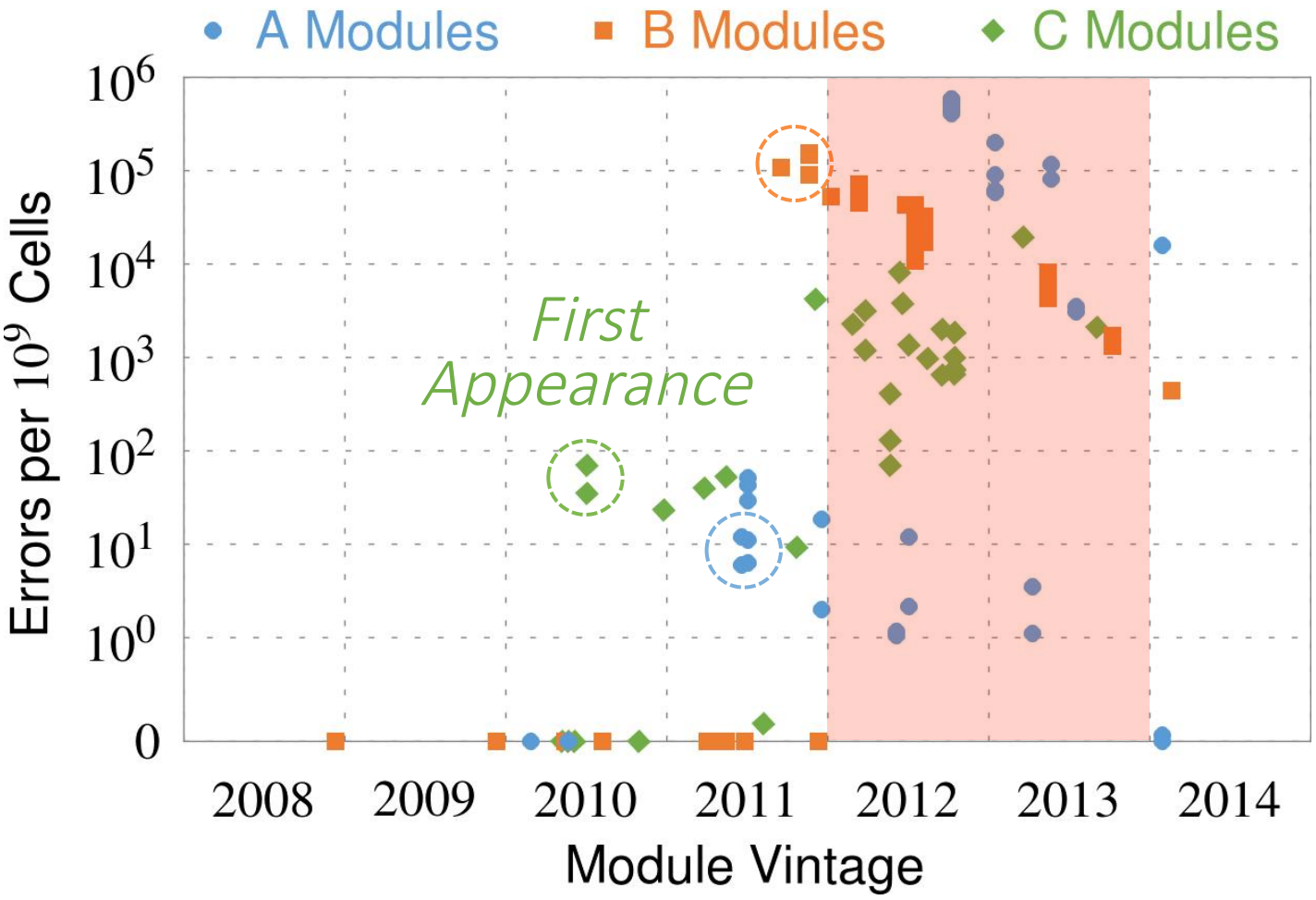
Recent DRAM Is More Vulnerable



Recent DRAM Is More Vulnerable

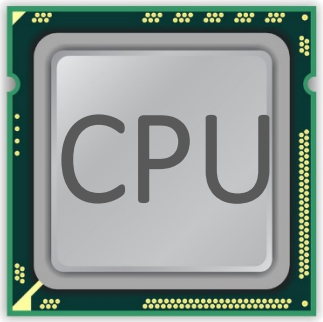


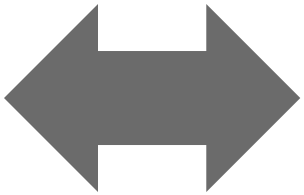
Recent DRAM Is More Vulnerable

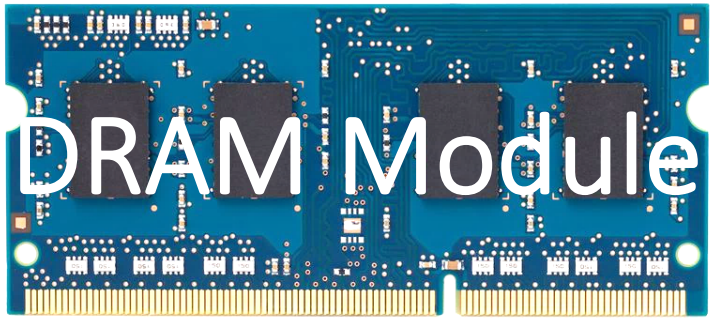


All modules from 2012–2013 are vulnerable

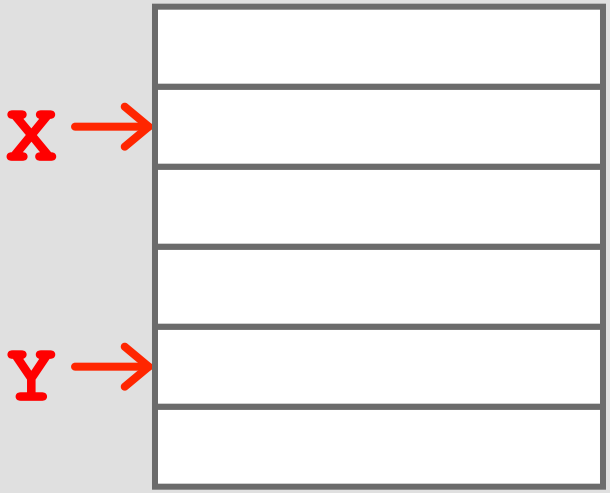
A Simple Program Can Induce Many Errors





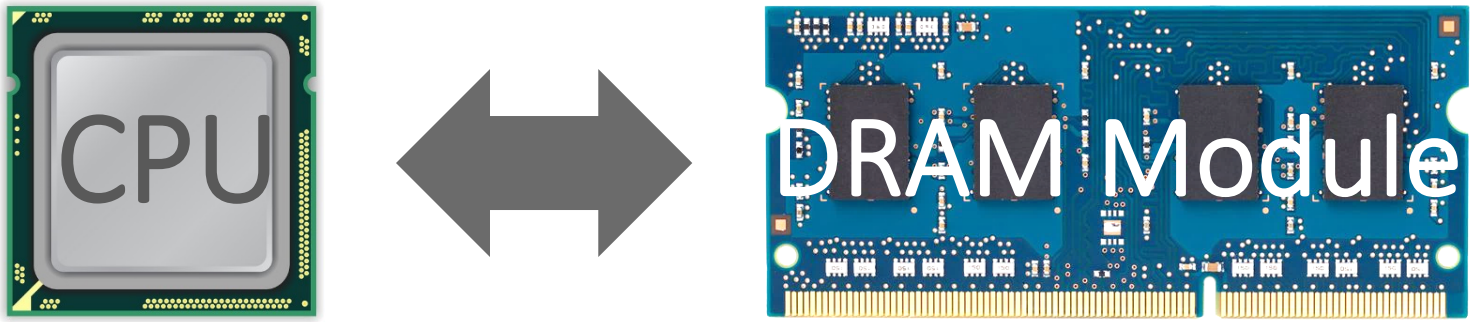



```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```



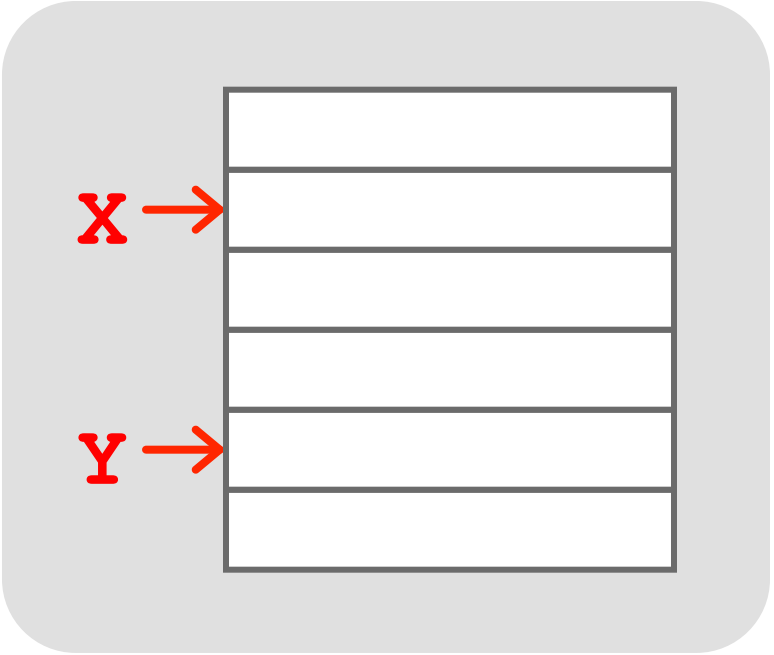
Download from: https://github.com/CMU-SAFARI/rowhammer

A Simple Program Can Induce Many Errors



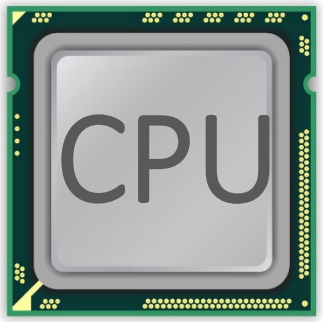
1. Avoid *cache hits*
   – Flush **X** from cache

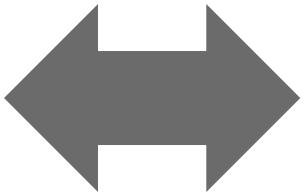
2. Avoid *row hits* to **X**
   – Read **Y** in another row

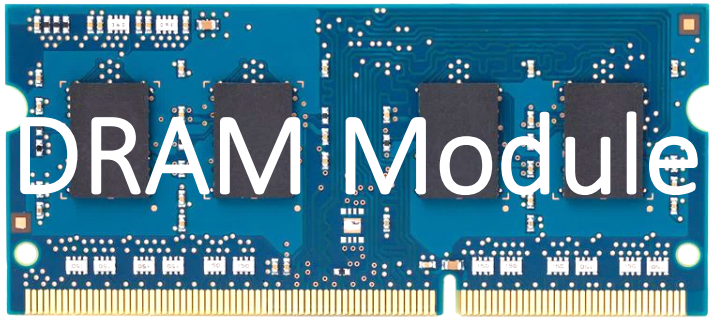


Download from: https://github.com/CMU-SAFARI/rowhammer

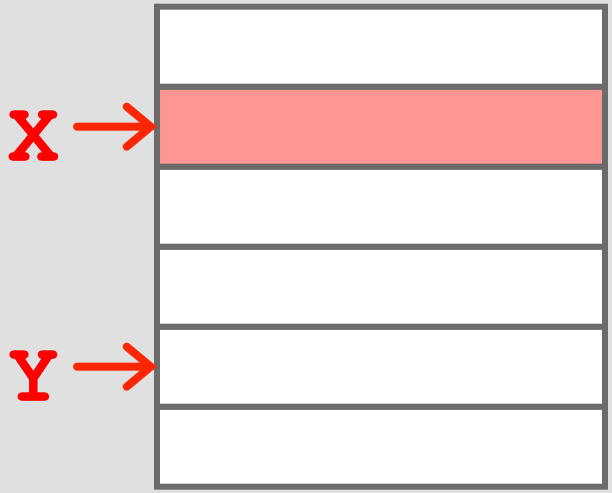
A Simple Program Can Induce Many Errors





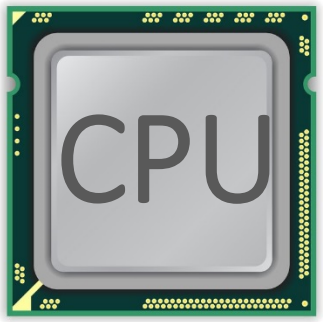


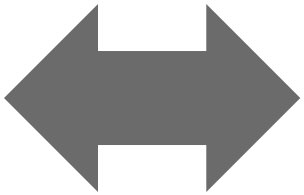

```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```

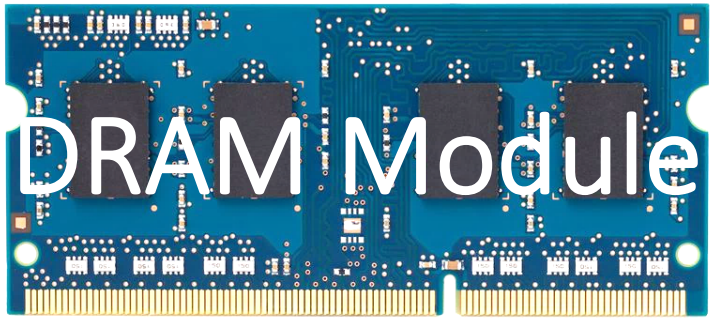


Download from: https://github.com/CMU-SAFARI/rowhammer

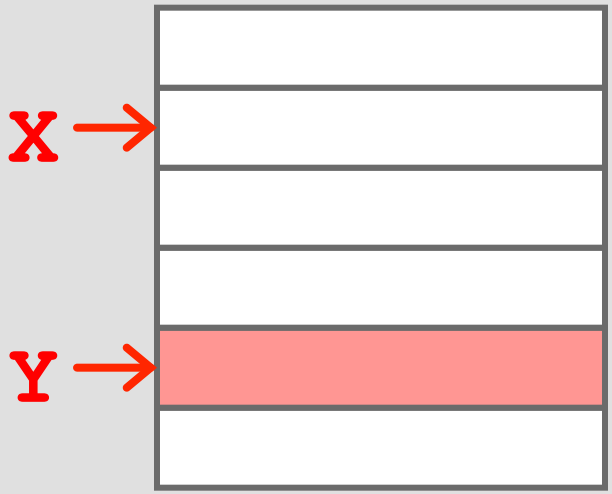
A Simple Program Can Induce Many Errors





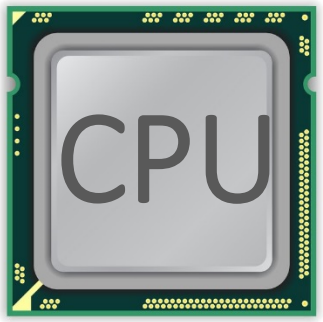


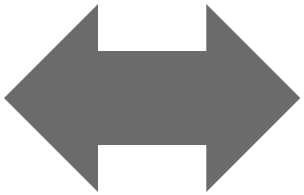

```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```

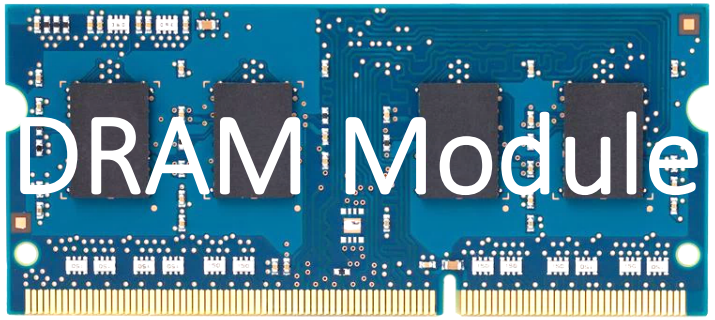


Download from: https://github.com/CMU-SAFARI/rowhammer

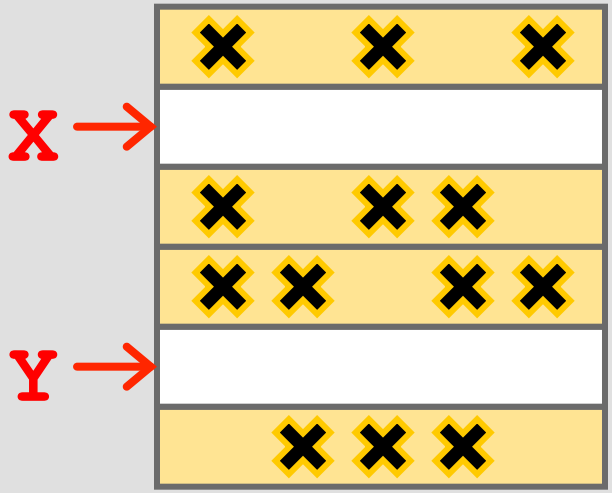
A Simple Program Can Induce Many Errors








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



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

Observed Errors in Real Systems

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

A real reliability & security issue

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

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

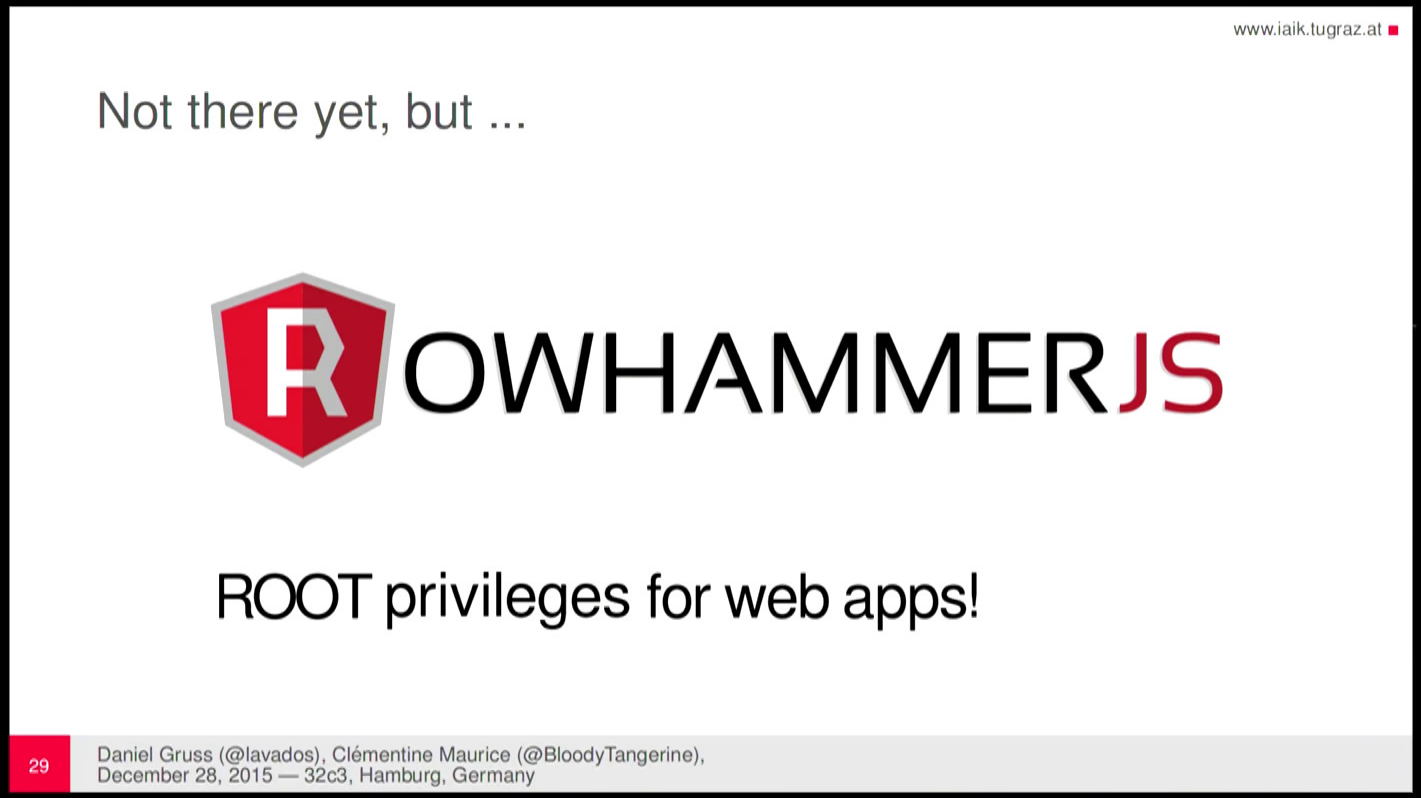
- "Rowhammer" is a problem with some recent DRAM devices in which repeatedly accessing a row of memory can cause bit flips in adjacent rows (Kim et al., ISCA 2014).
  - Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)
- We tested 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

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."



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

Source: https://lab.dsst.io/32c3-slides/7197.html

More Security Implications

"Can gain control of a smart phone deterministically"

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

Source: https://fossbytes.com/drammer-rowhammer-attack-android-root-devices/

More Security Implications?

Selected Readings on RowHammer (I)

- Our first detailed study: Rowhammer analysis and solutions (June 2014)
  - 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]

- Our Source Code to Induce Errors in Modern DRAM Chips (June 2014)
  - https://github.com/CMU-SAFARI/rowhammer

- Google Project Zero's Attack to Take Over a System (March 2015)
  - Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn+, 2015)
  - https://github.com/google/rowhammer-test
  - **Double-sided Rowhammer**

Selected Readings on RowHammer (II)

- Remote RowHammer Attacks via JavaScript (July 2015)
  - http://arxiv.org/abs/1507.06955
  - https://github.com/IAIK/rowhammerjs
  - Gruss et al., DIMVA 2016.
  - **CLFLUSH-free Rowhammer**
  - "A fully automated attack that requires nothing but a website with JavaScript to **trigger faults on remote hardware**."
  - "We can gain unrestricted access to systems of website visitors."

- ANVIL: Software-Based Protection Against Next-Generation Rowhammer Attacks (March 2016)
  - http://dl.acm.org/citation.cfm?doid=2872362.2872390
  - Aweke et al., ASPLOS 2016
  - **CLFLUSH-free Rowhammer**
  - Software based monitoring for rowhammer detection

Selected Readings on RowHammer (III)

- Flip Feng Shui: Hammering a Needle in the Software Stack (August 2016)
  - https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_razavi.pdf
  - Razavi et al., USENIX Security 2016.
  - Combines memory deduplication and RowHammer
  - "**A malicious VM can gain unauthorized access to a co-hosted VM running OpenSSH**."
  - Breaks OpenSSH public key authentication

- Drammer: Deterministic Rowhammer Attacks on Mobile Platforms (October 2016)
  - http://dl.acm.org/citation.cfm?id=2976749.2978406
  - Van Der Veen et al., CCS 2016
  - **Can take over an ARM-based Android system deterministically**
  - Exploits predictable physical memory allocator behavior
    - Can deterministically place security-sensitive data (e.g., page table) in an attacker-chosen, vulnerable location in memory

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

Better Solution Directions: Principled Designs

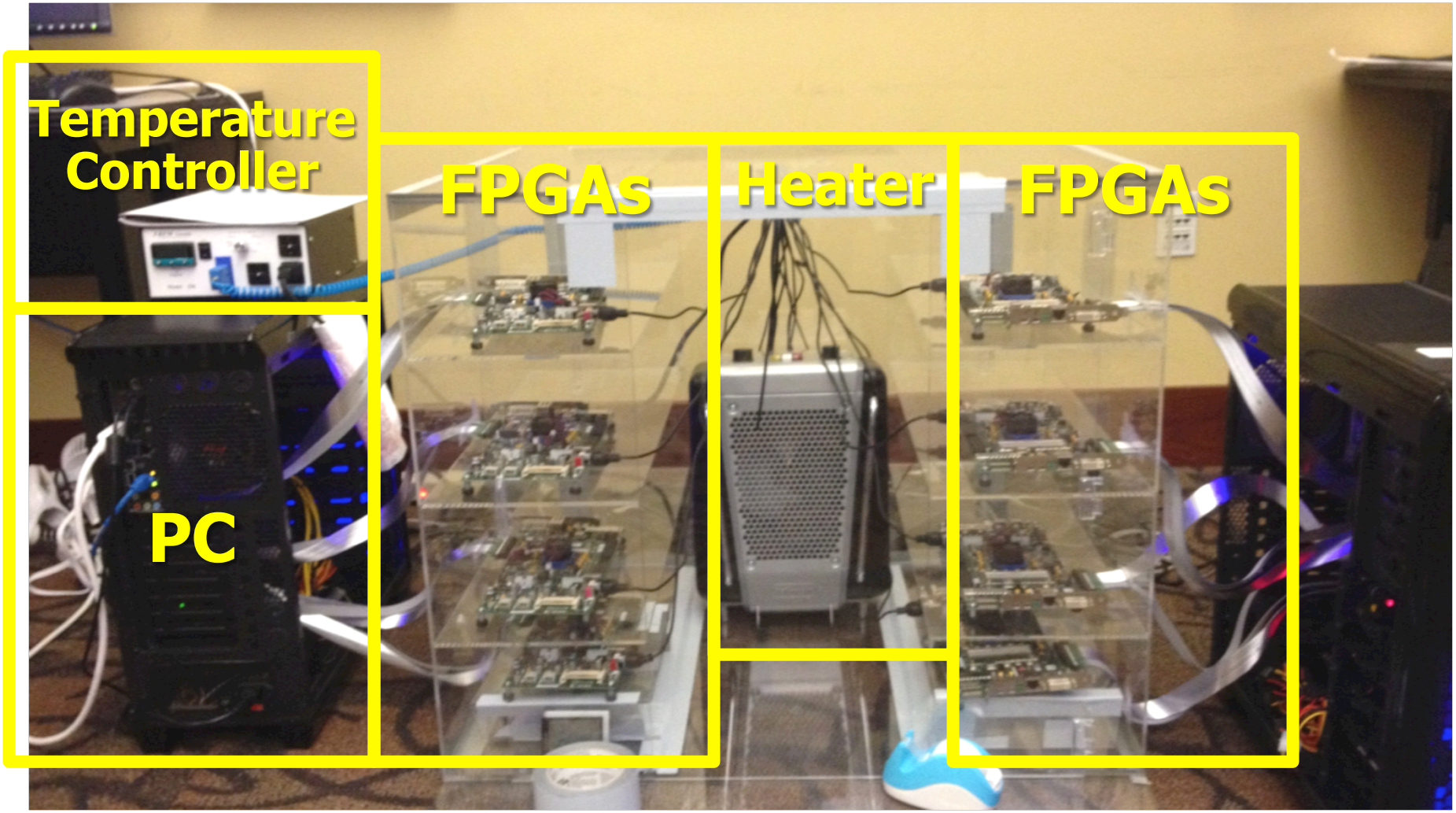
Design fundamentally secure computing architectures

Predict and prevent such safety issues

How Do We Keep Memory Secure?

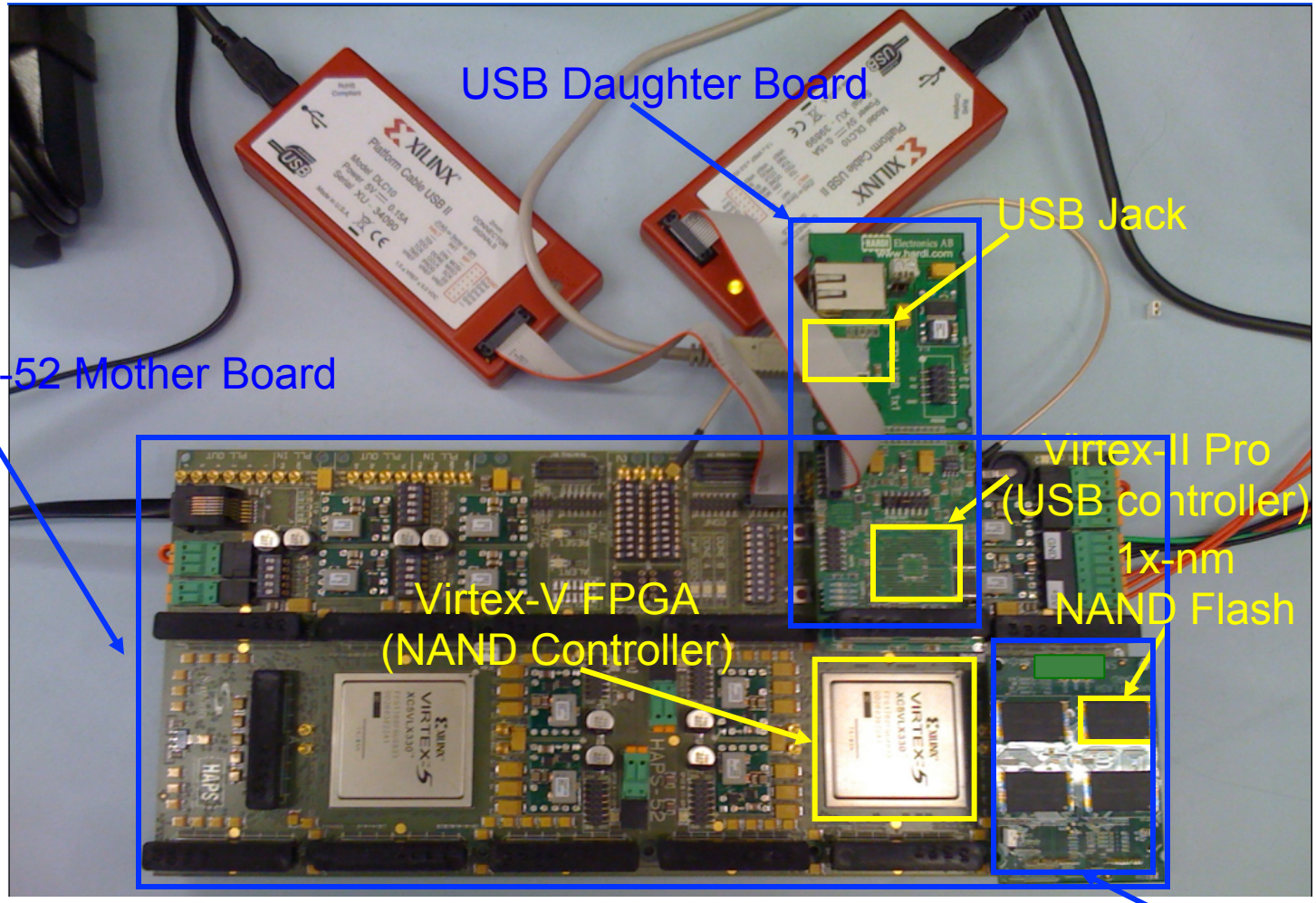
- **Understand**: Methodologies for failure modeling and discovery
  - Modeling and prediction based on real (device) data
- **Architect**: Principled co-architecting of system and memory
  - Good partitioning of duties across the stack
- **Design & Test**: Principled design, automation, testing
  -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

Understand and Model with Experiments (DRAM)



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

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]

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

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.

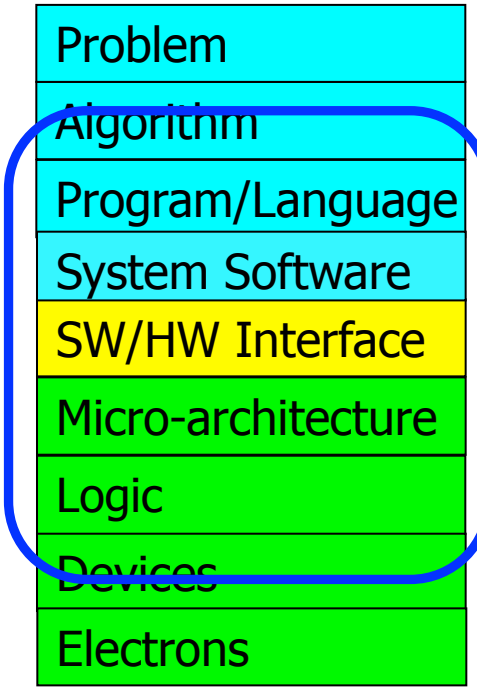
Fukami+, "Improving the Reliability of Chip-Off Forensic Analysis of NAND Flash Memory Devices," DFRWS EU 2017.

There are Two Other Solution Directions

- **New Technologies:** Replace or (more likely) augment DRAM with a different technology
  - Non-volatile memories

- **Embracing Un-reliability:**

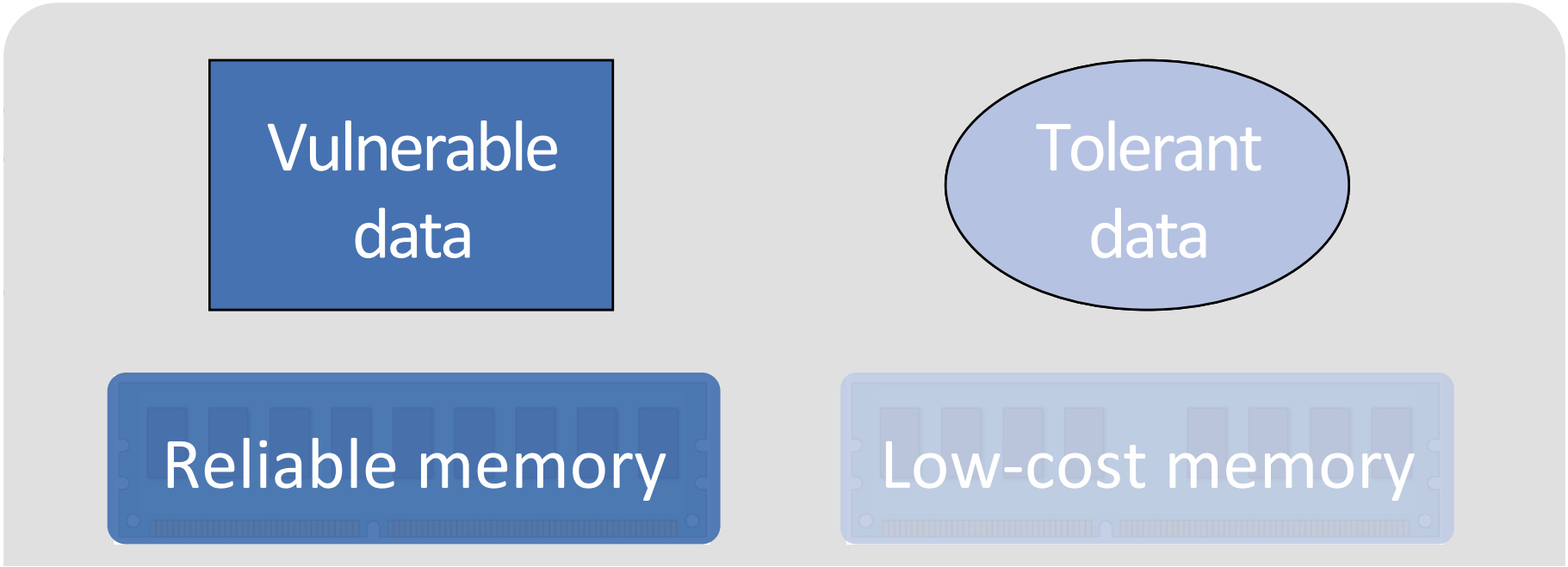
  Design memories with different reliability and store data intelligently across them



- …

Fundamental solutions to security require co-design across the hierarchy

Exploiting Memory Error Tolerance with Hybrid Memory Systems



On Microsoft's Web Search workload

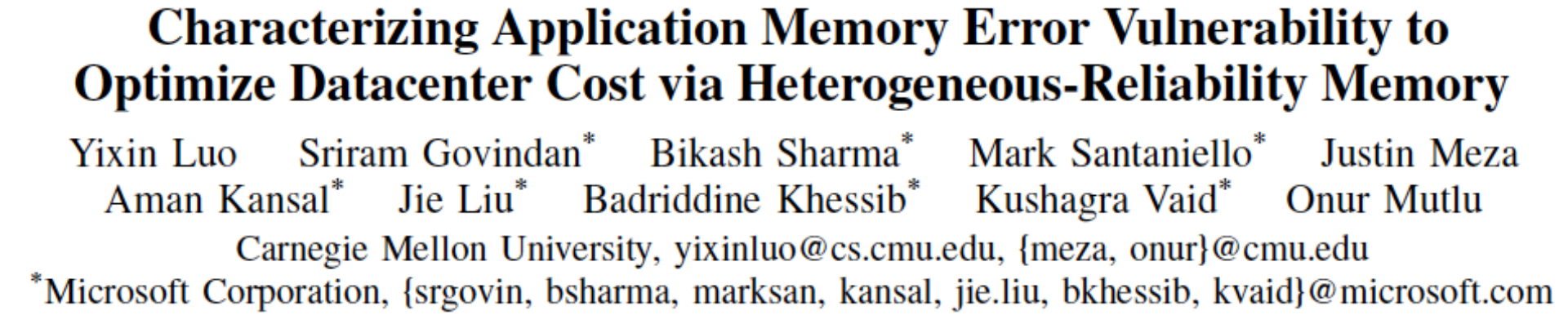
Reduces server hardware cost by 4.7 %

Achieves single server availability target of 99.90 %

Heterogeneous-**R**eliability **M**emory [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, bkhessib, kvaid}@microsoft.com

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*

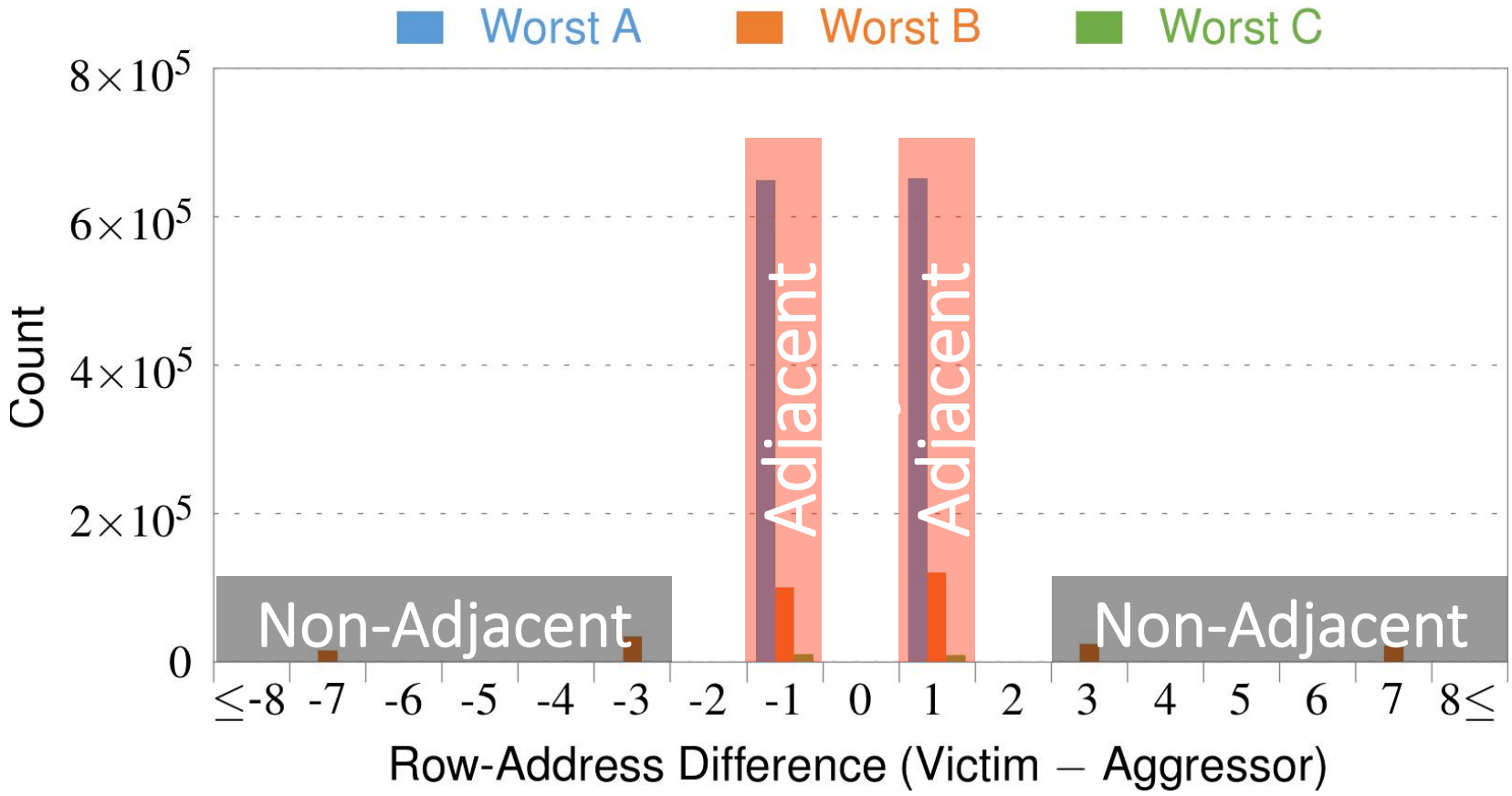
Confirmed by at least one manufacturer

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

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

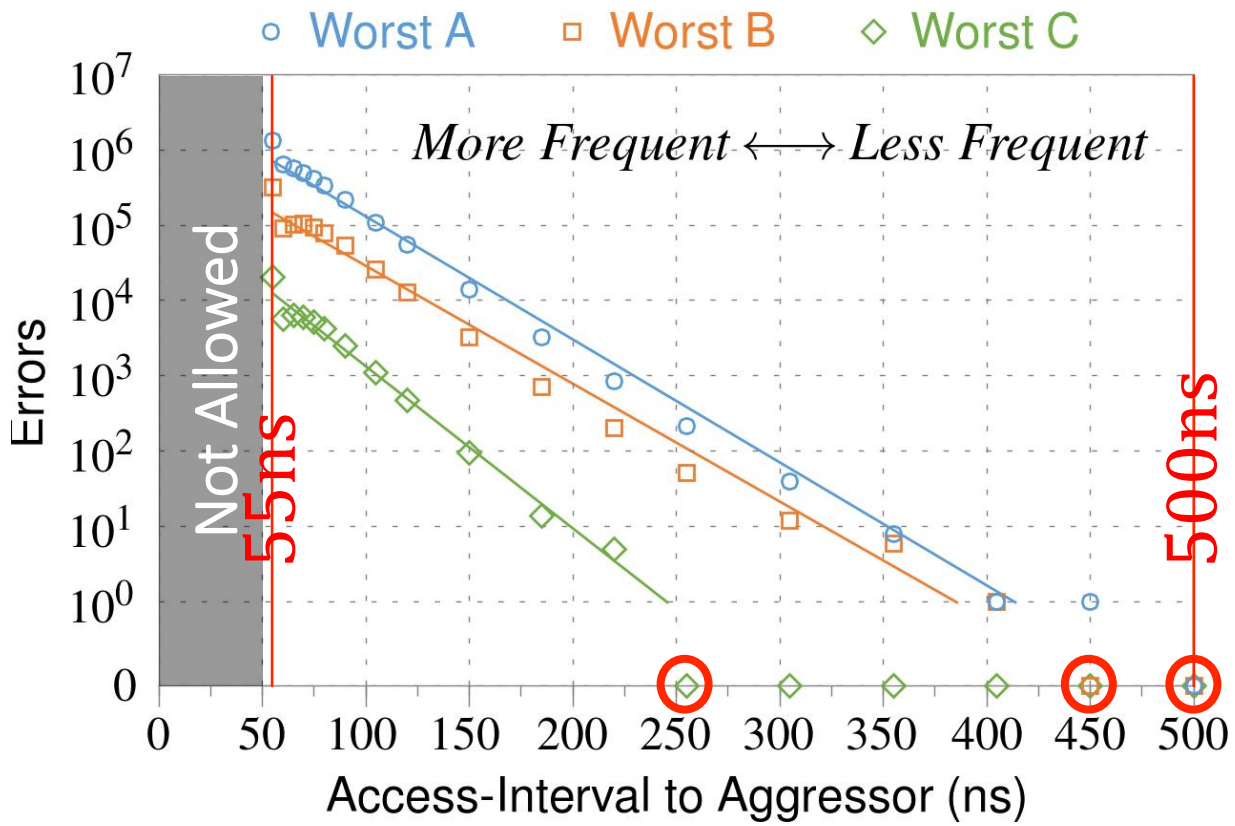
4. Adjacency: Aggressor & Victim



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

Most aggressors & victims are adjacent

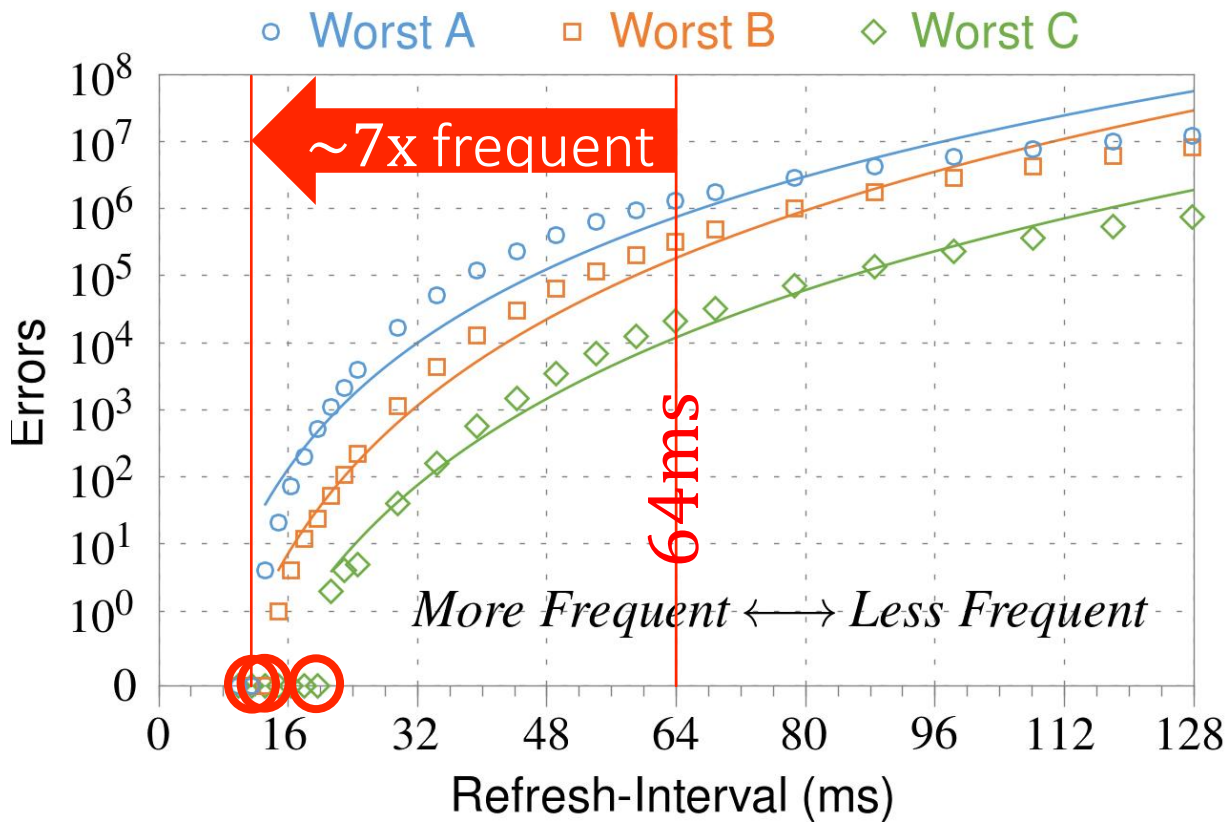
❶ Access Interval (Aggressor)



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

Less frequent accesses → Fewer errors

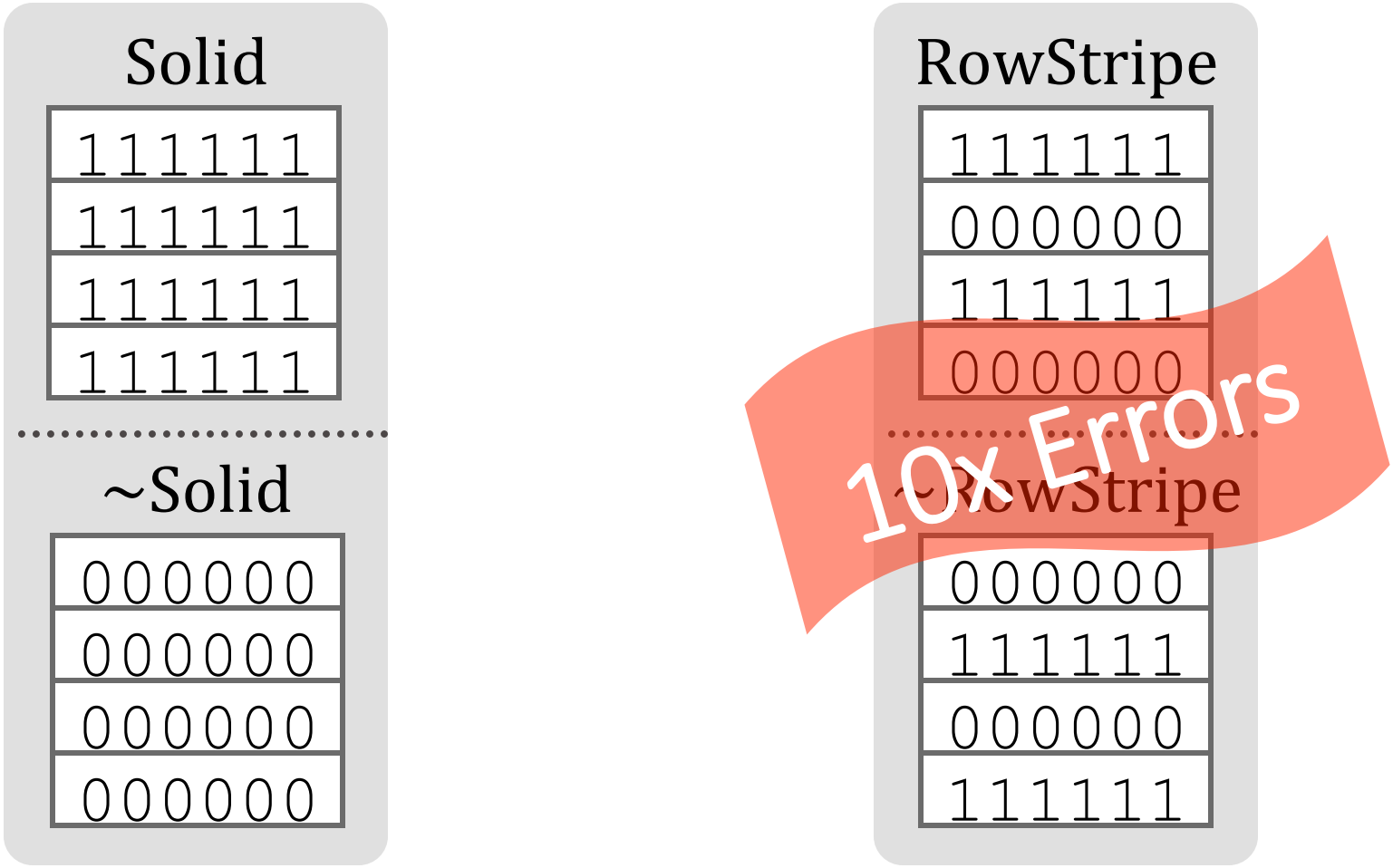
❷ Refresh Interval



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

More frequent refreshes → Fewer errors

❸ Data Pattern



Errors affected by data stored in other cells

6. Other Results (in Paper)

- *Victim Cells ≠ 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

❶ *Throttle accesses to same row*

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

❷ *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

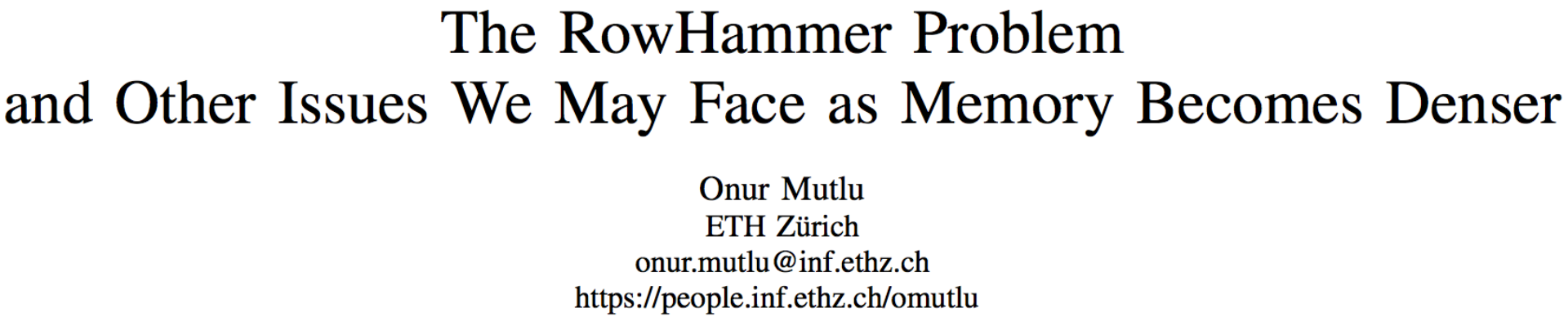
- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu,
  "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"
  Proceedings of the
  41st International Symposium on Computer Architecture (***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 Kim[1] Ross Daly* Jeremie Kim[1] Chris Fallin* Ji Hye Lee[1]
Donghyuk Lee[1] Chris Wilkerson[2] Konrad Lai Onur Mutlu[1]
[1]Carnegie Mellon University [2]Intel 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

Challenge and Opportunity for Future

Fundamentally

Secure, Reliable, Safe

Computing Architectures

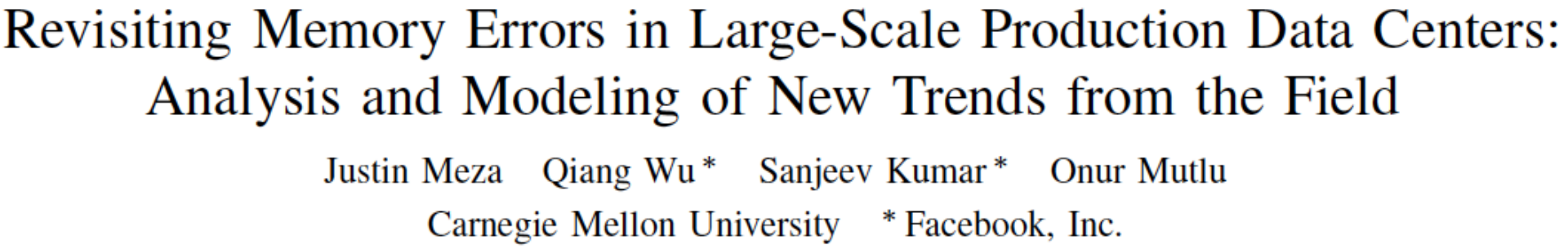
Future of Main Memory

- 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

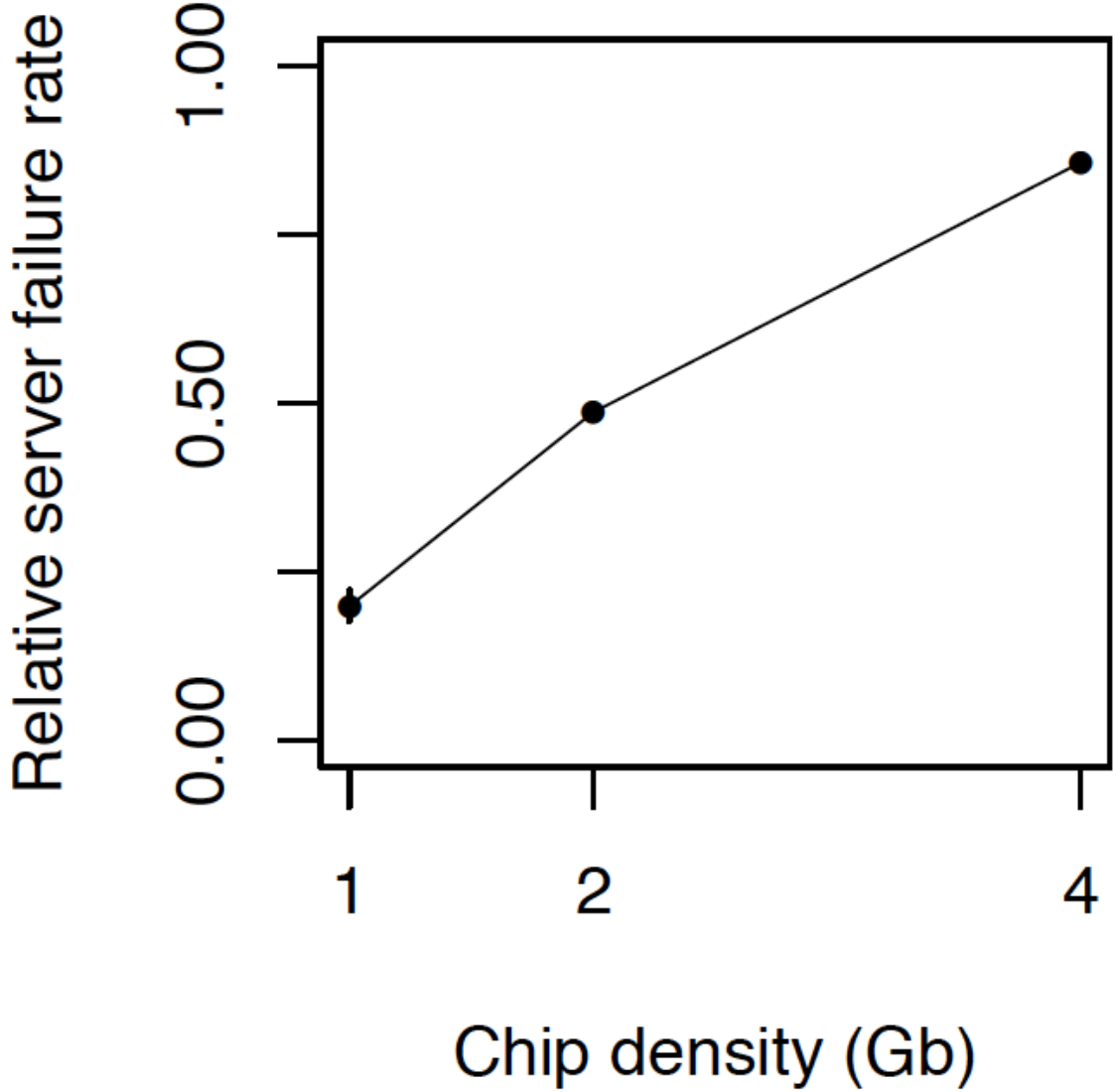
- Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu, **"Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field"** *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (***DSN***), Rio de 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

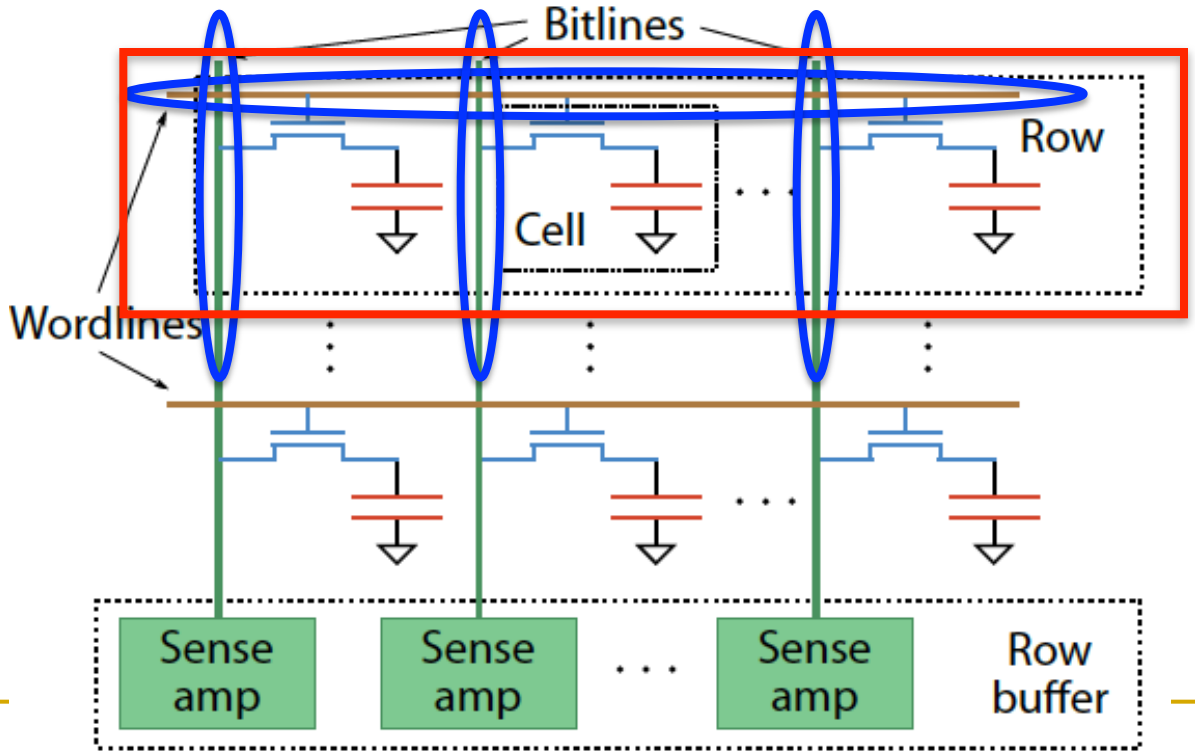
Two Challenges to Retention Time Profiling

- Data Pattern Dependence (DPD) of retention time

- Variable Retention Time (VRT) phenomenon

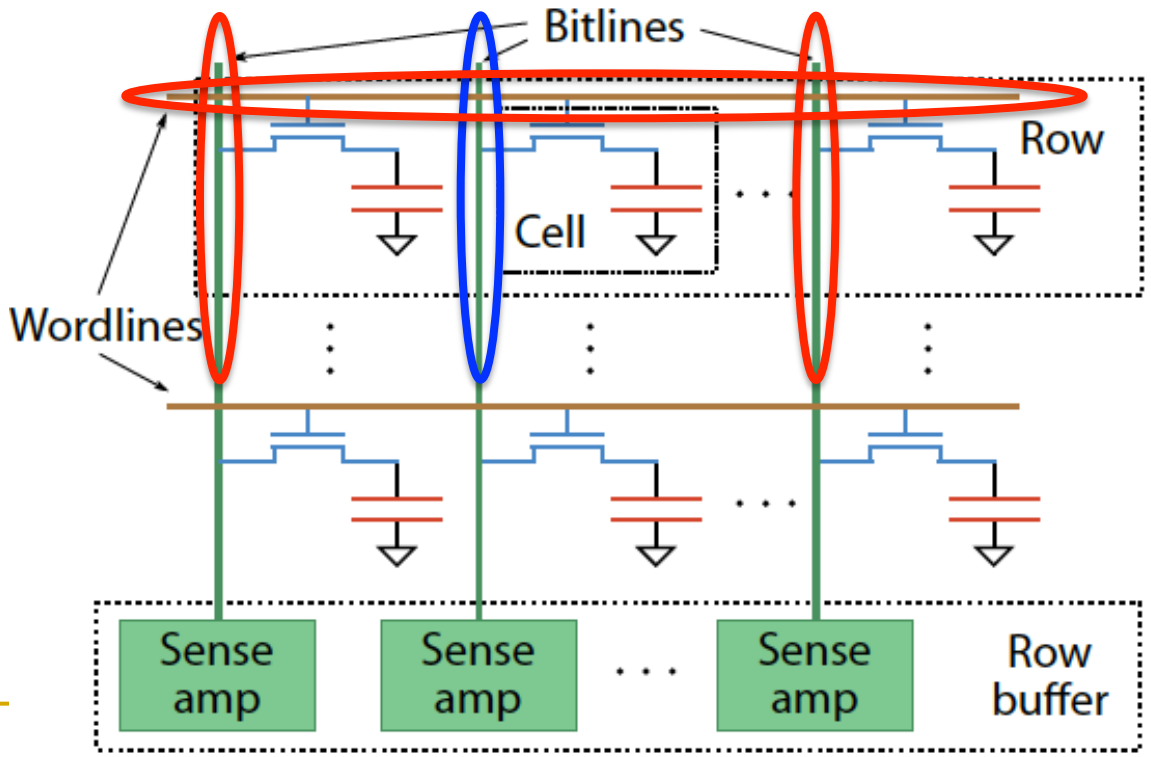
Two Challenges to Retention Time Profiling

- 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

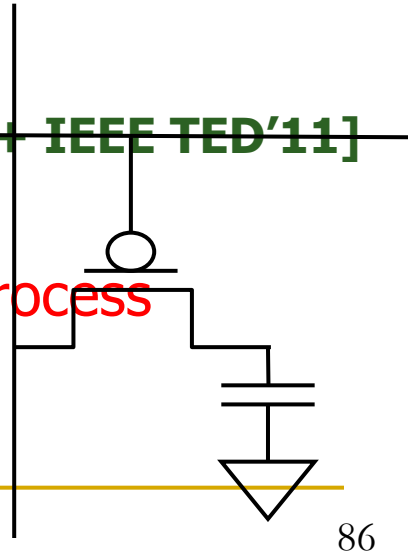
- Retention time of a cell depends on data patterns stored in nearby cells

  → need to find the worst data pattern to find worst-case retention time

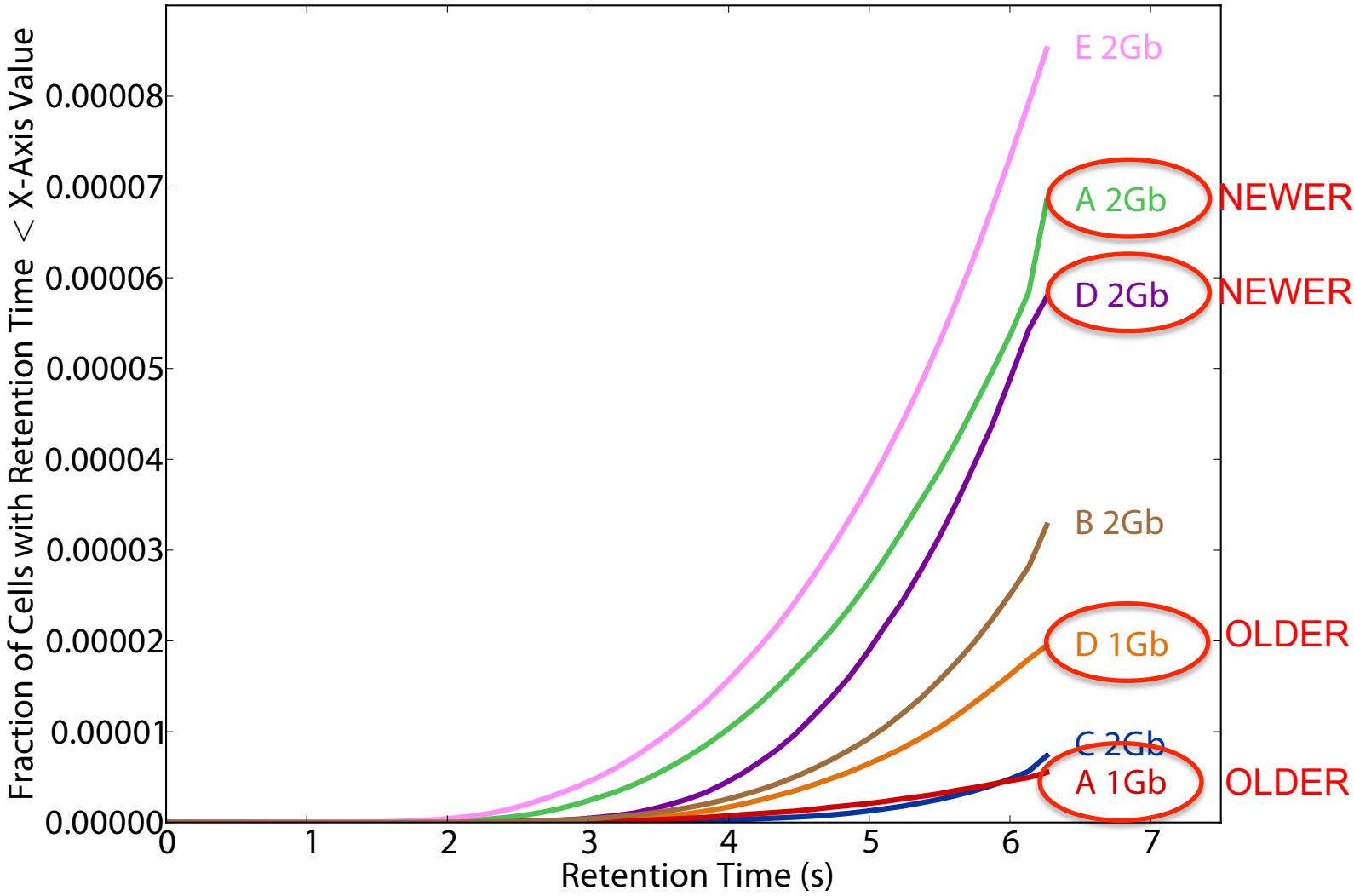
  → this pattern is location dependent

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

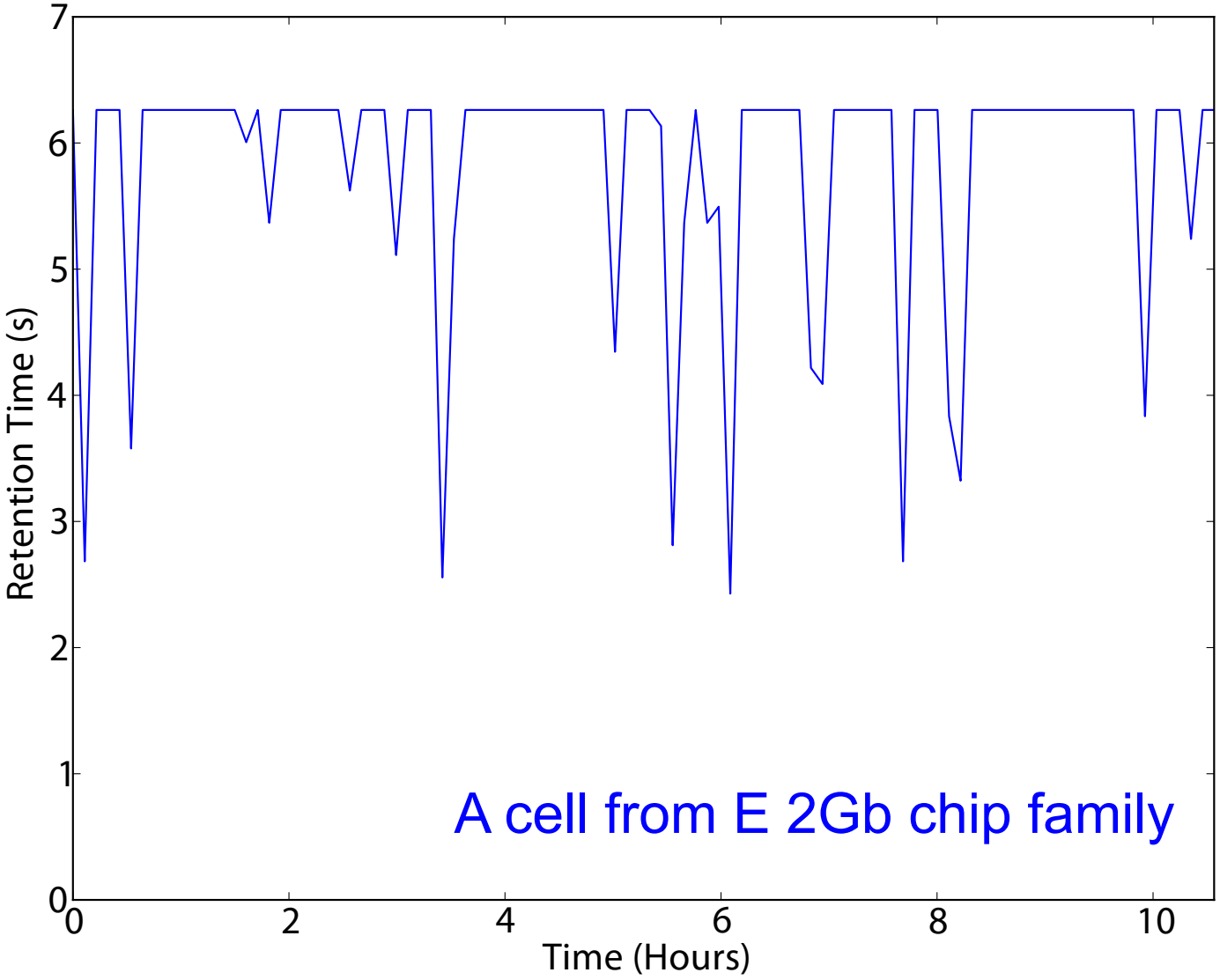


Modern DRAM Retention Time Distribution

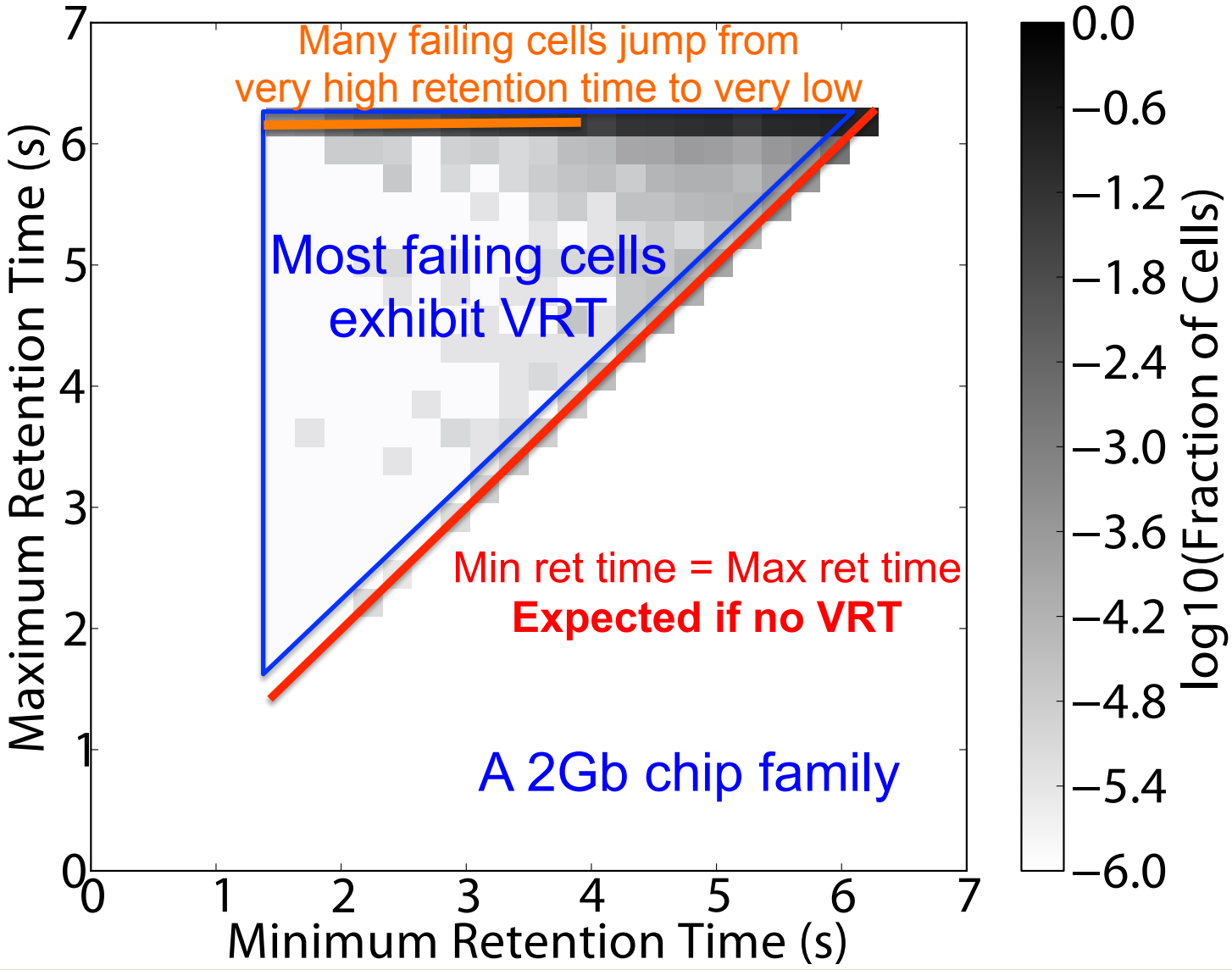


Newer device families have more weak cells than older ones
Likely a result of technology scaling

An Example VRT Cell



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

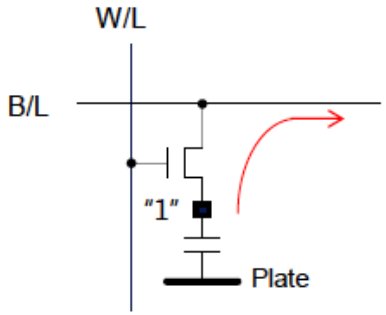
Chris Wilkerson
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
chris.wilkerson@intel.com

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

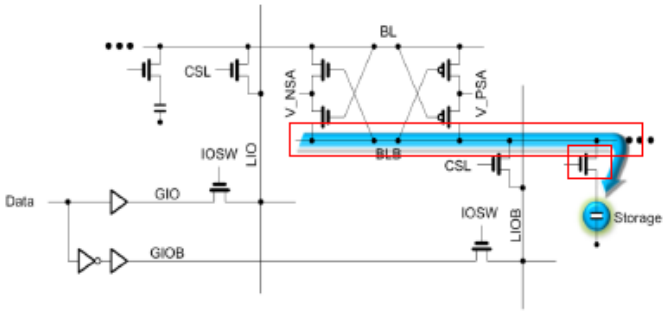
Industry Is Writing Papers About It, Too

DRAM Process Scaling Challenges

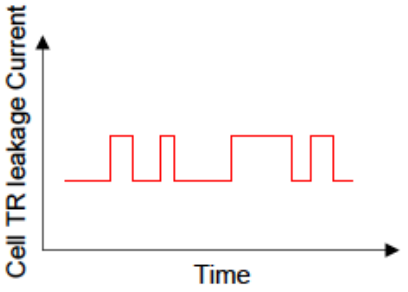
- **Refresh**
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance
  - Leakage current of cell access transistors increasing
- **tWR**
  - Contact resistance between the cell capacitor and access transistor increasing
  - On-current of the cell access transistor decreasing
  - Bit-line resistance increasing
- **VRT**
  - Occurring more frequently with cell capacitance decreasing



Refresh



tWR



VRT

Industry Is Writing Papers About It, Too

DRAM Process Scaling Challenges

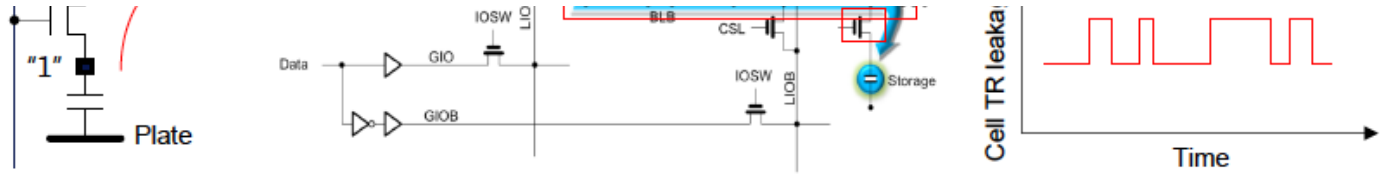
- **Refresh**
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

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*



Refresh **tWR** **VRT**

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]

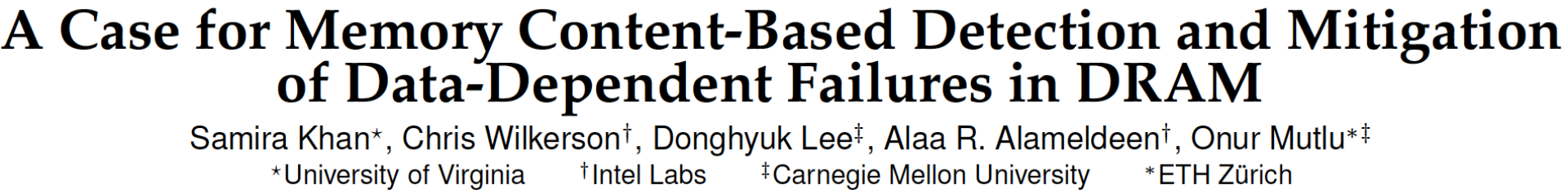
- Samira Khan, Donghyuk Lee, Yoongu Kim, Alaa Alameldeen, Chris Wilkerson, and Onur Mutlu,
  "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study"
  *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**)*, 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* Donghyuk Lee†‡ Onur Mutlu*†
*University of Virginia †Carnegie Mellon University ‡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.



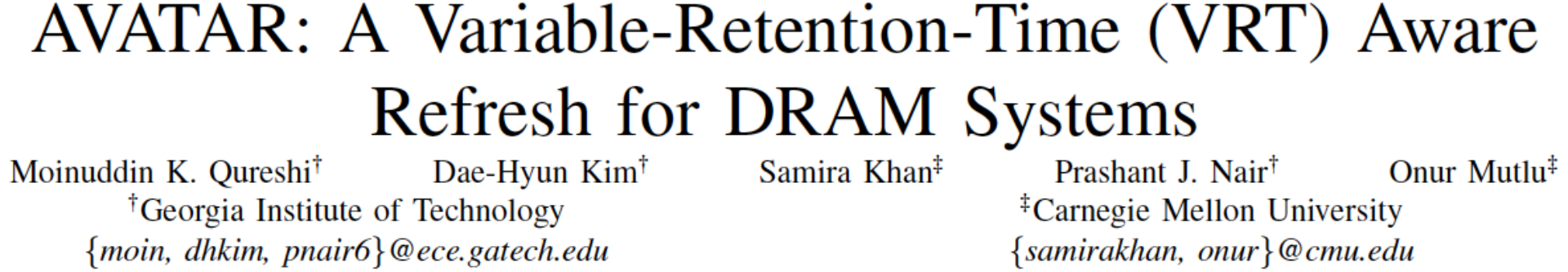
A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM

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
{moin, dhkim, pnair6}@ece.gatech.edu
‡Carnegie Mellon University
{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.

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

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.

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

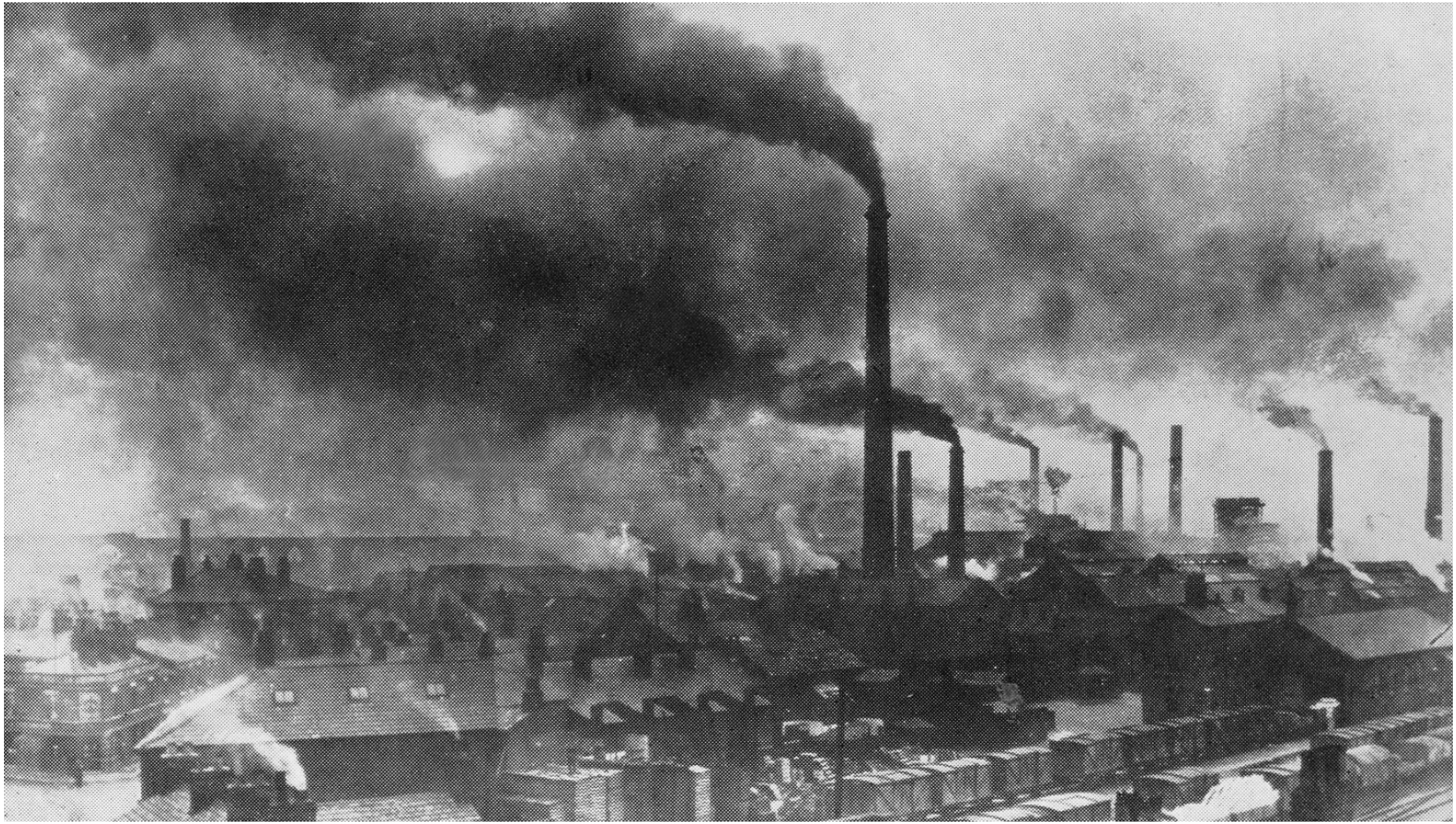
Challenge and Opportunity for Future

Fundamentally Secure, Reliable, Safe 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

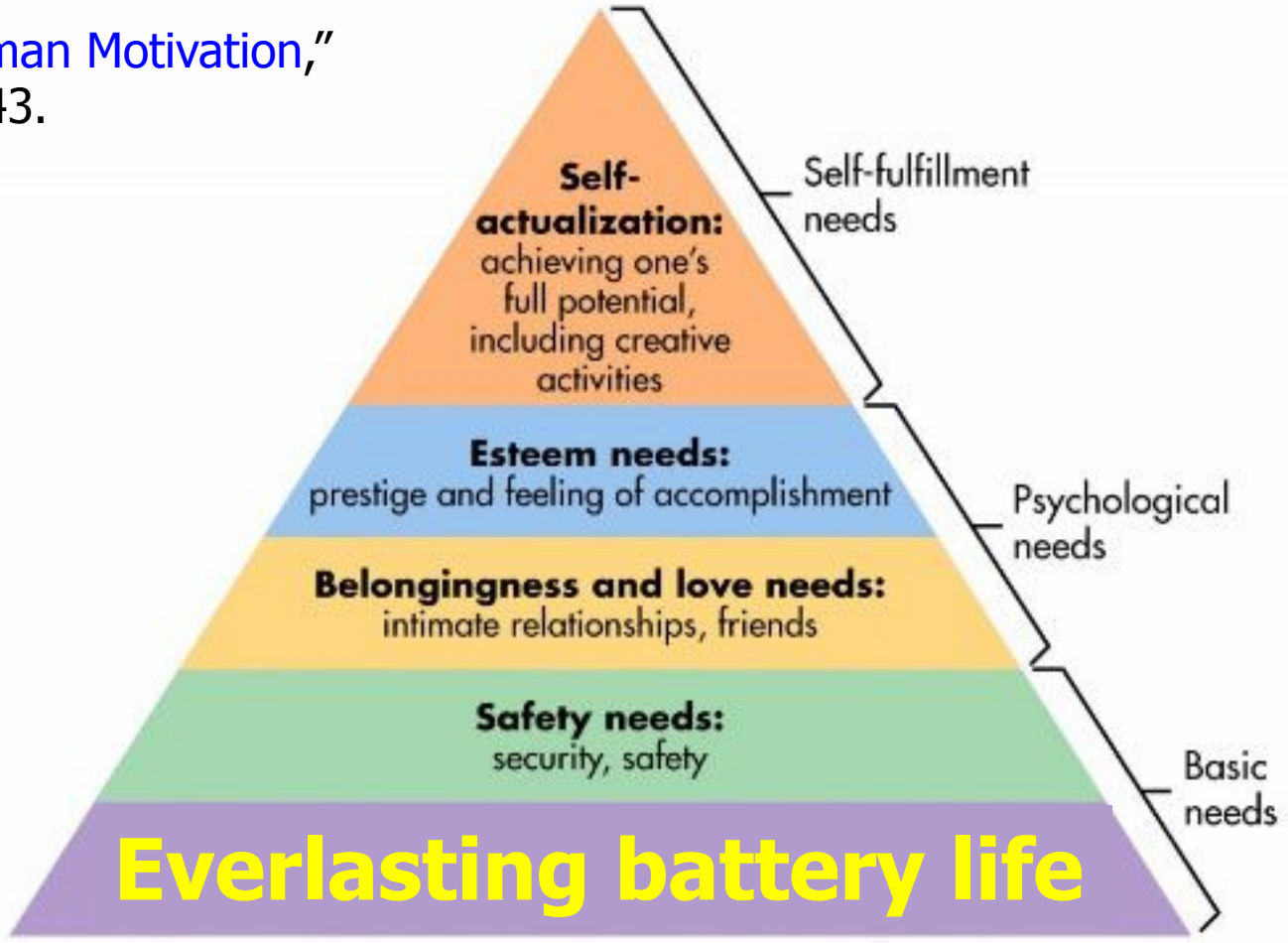
Source: V. Milutinovic



Source: V. Milutinovic

Maslow's (Human) Hierarchy of Needs, Revisited

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



Source: https://www.simplypsychology.org/maslow.html

Challenge and Opportunity for Future

Sustainable

and

Energy Efficient

The Problem

Data access is the major performance and energy bottleneck

Our current

design principles

cause great energy waste

The Problem

Processing of data

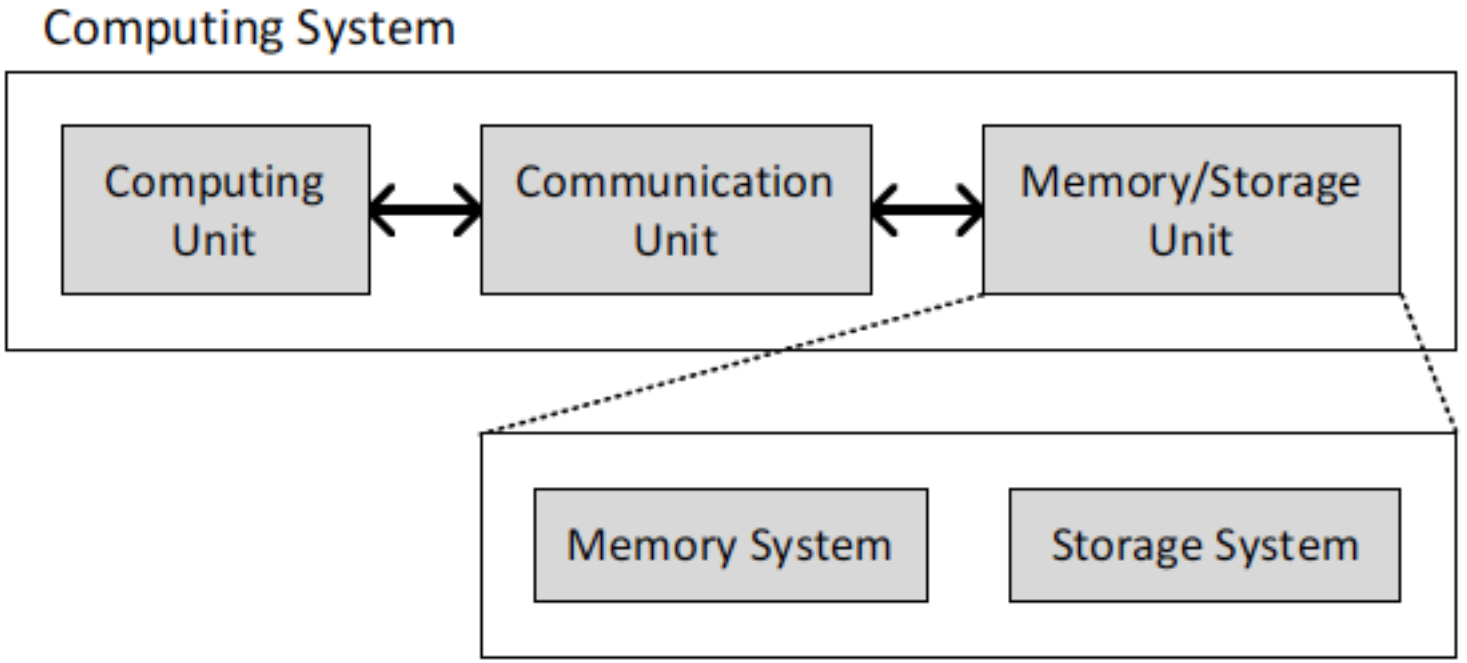
is performed

far away from the data

A Computing System

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

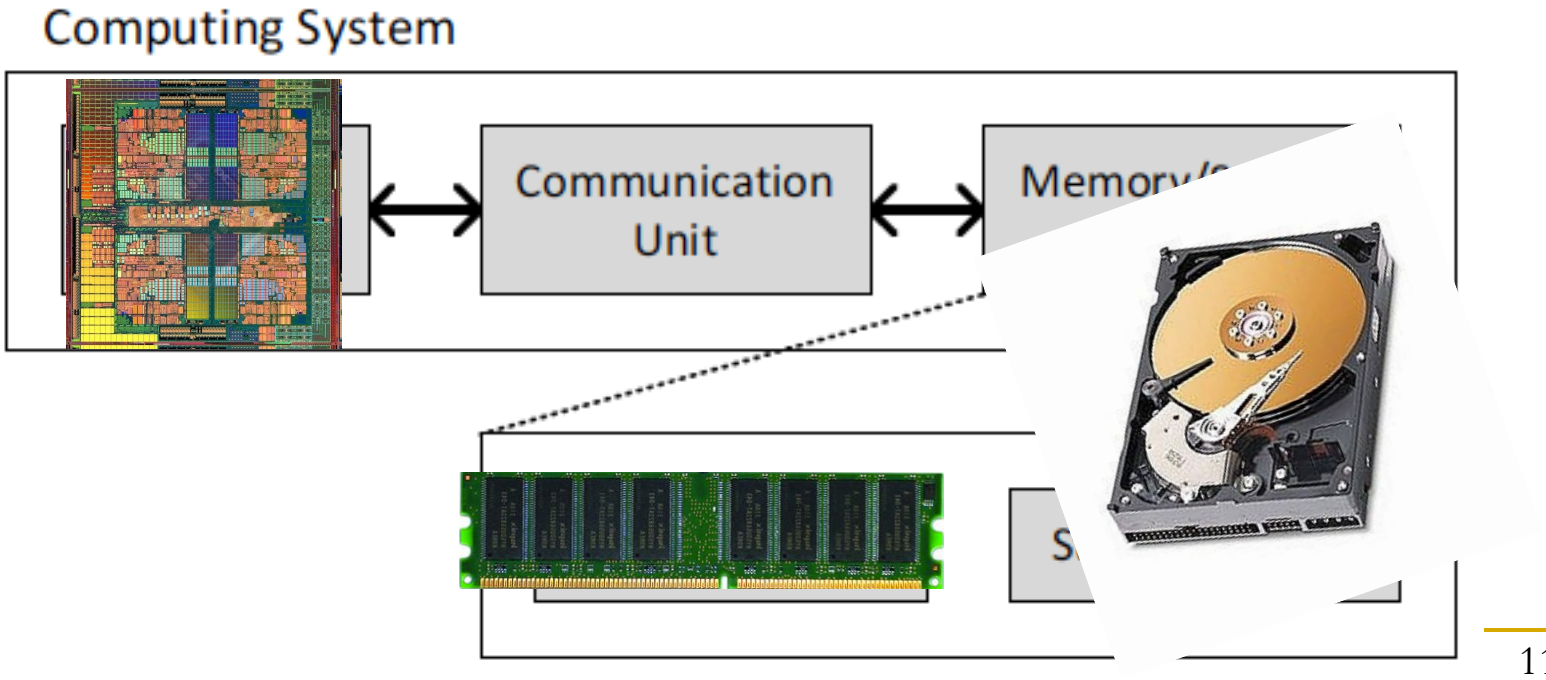
Burks, Goldstein, von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument," 1946.



A Computing System

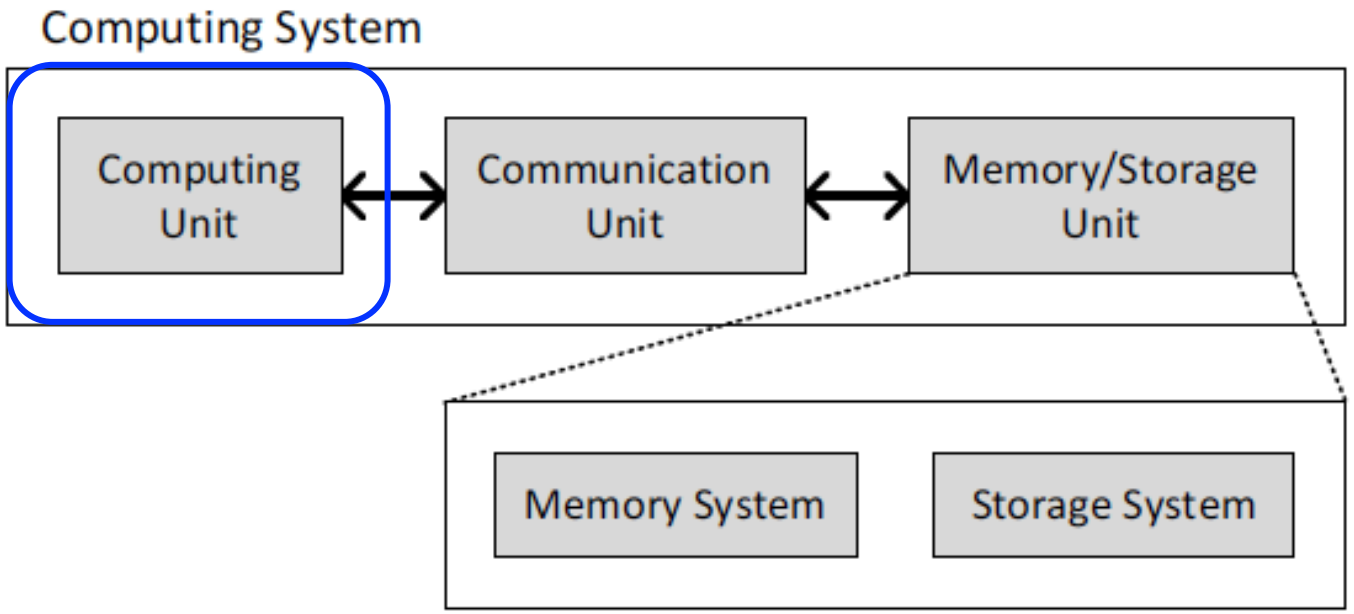
- Three key components
- Computation
- Communication
- Storage/memory

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



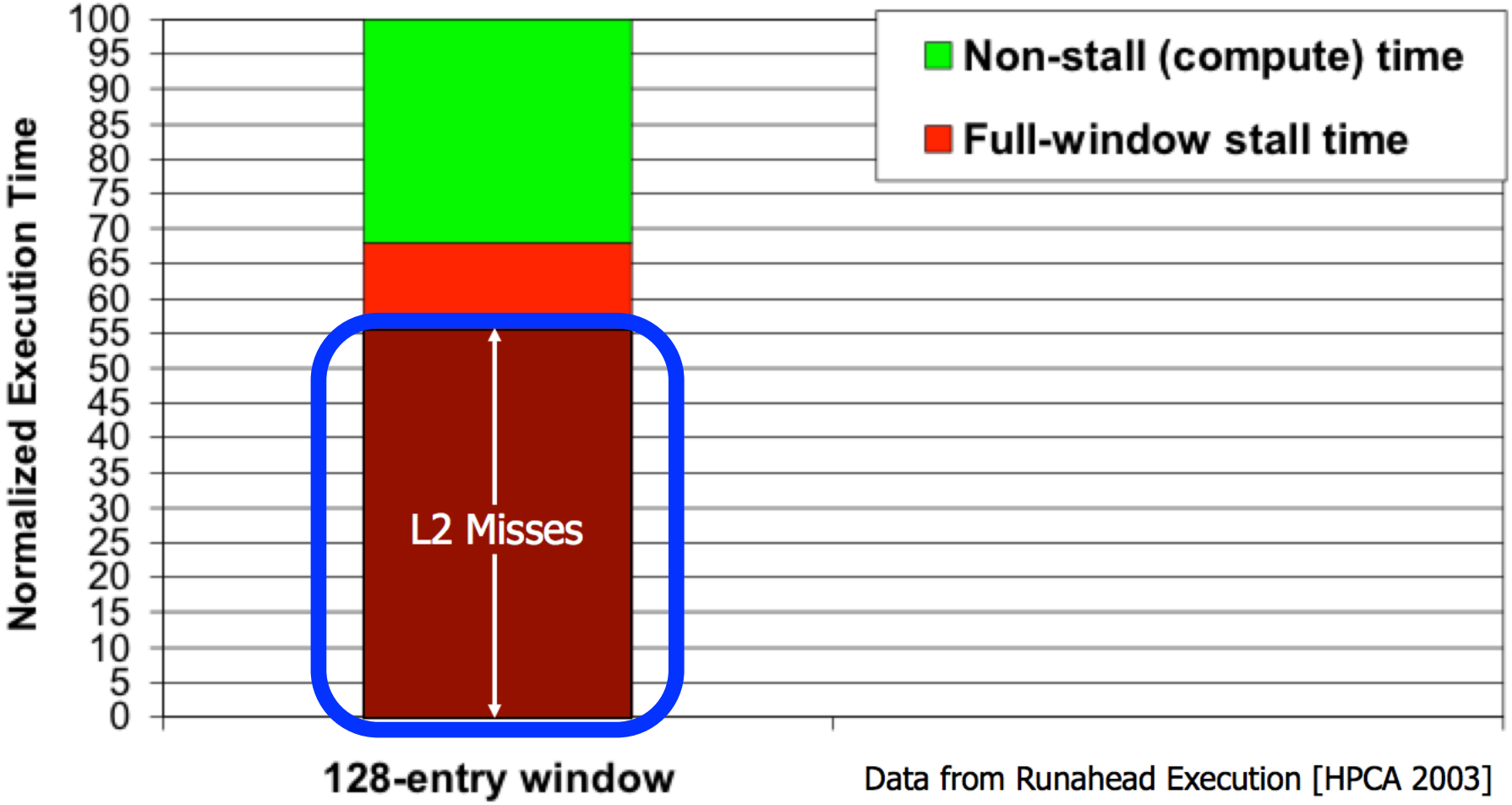
Today's Computing Systems

- 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 slaves and are largely unoptimized (except for some that are on the processor die)



Yet …

- "**It's the Memory, Stupid!**" (Richard Sites, MPR, 1996)



Mutlu+, "Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-Order Processors," HPCA 2003.

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

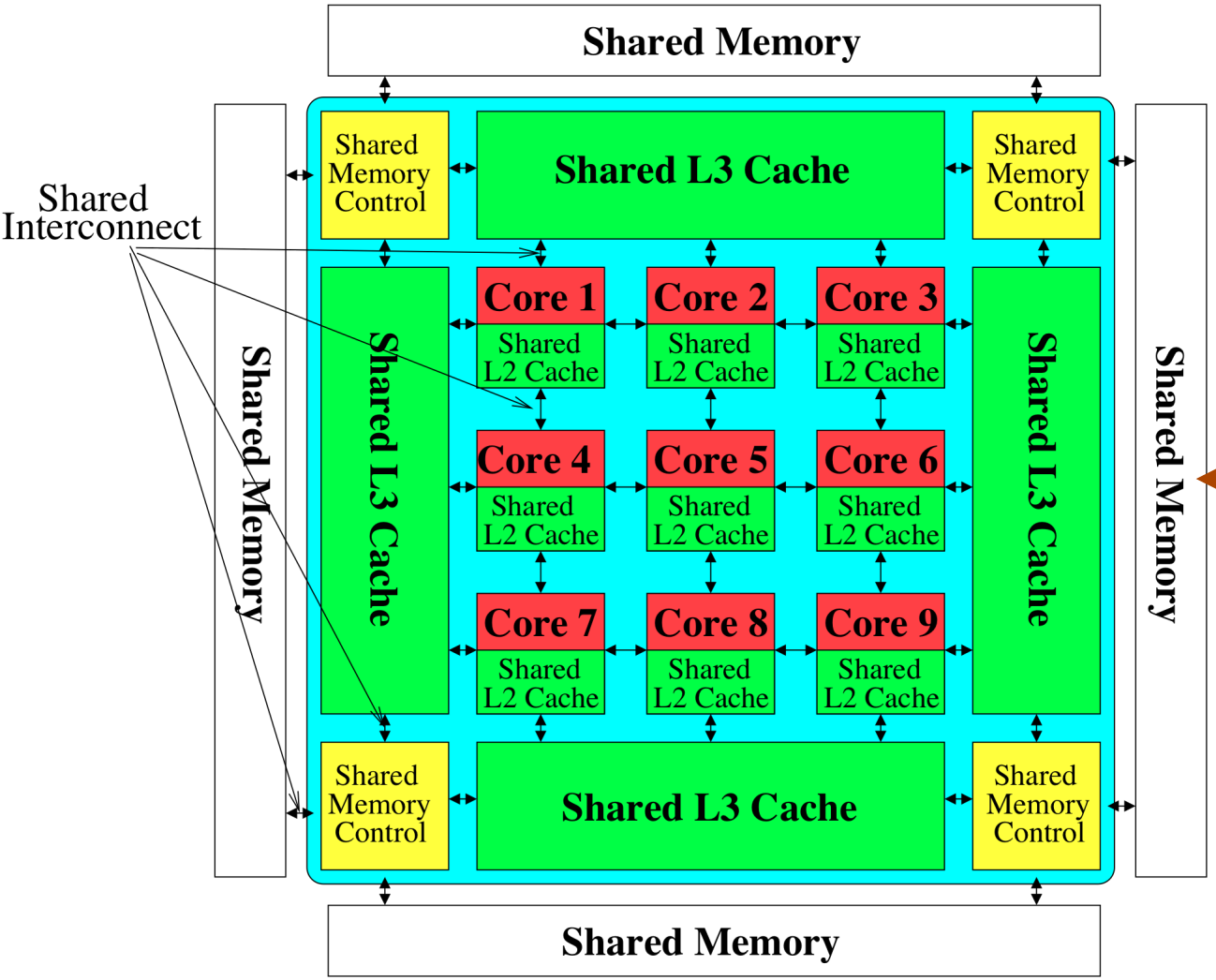
- Overly complex and bloated processor (and accelerators)
  - To tolerate data access from memory
  - Complex hierarchies and mechanisms

  → Energy inefficient

  → Low performance

  → Complex

Perils of Processor-Centric Design



Most of the system is dedicated to storing and moving data

Three Key Systems Trends

1. Data access is a major bottleneck

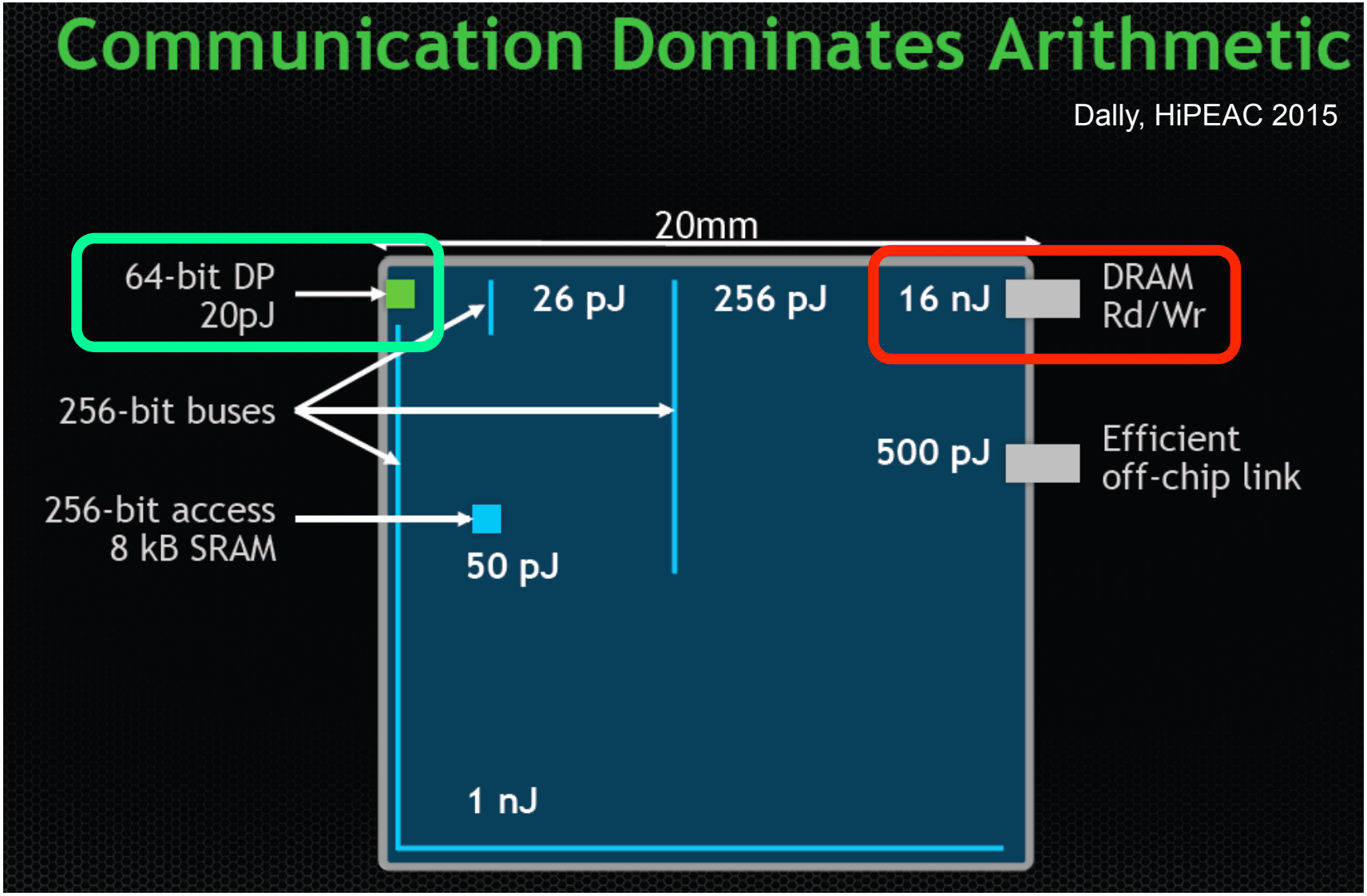
- Applications are increasingly data hungry

2. Energy consumption is a key limiter

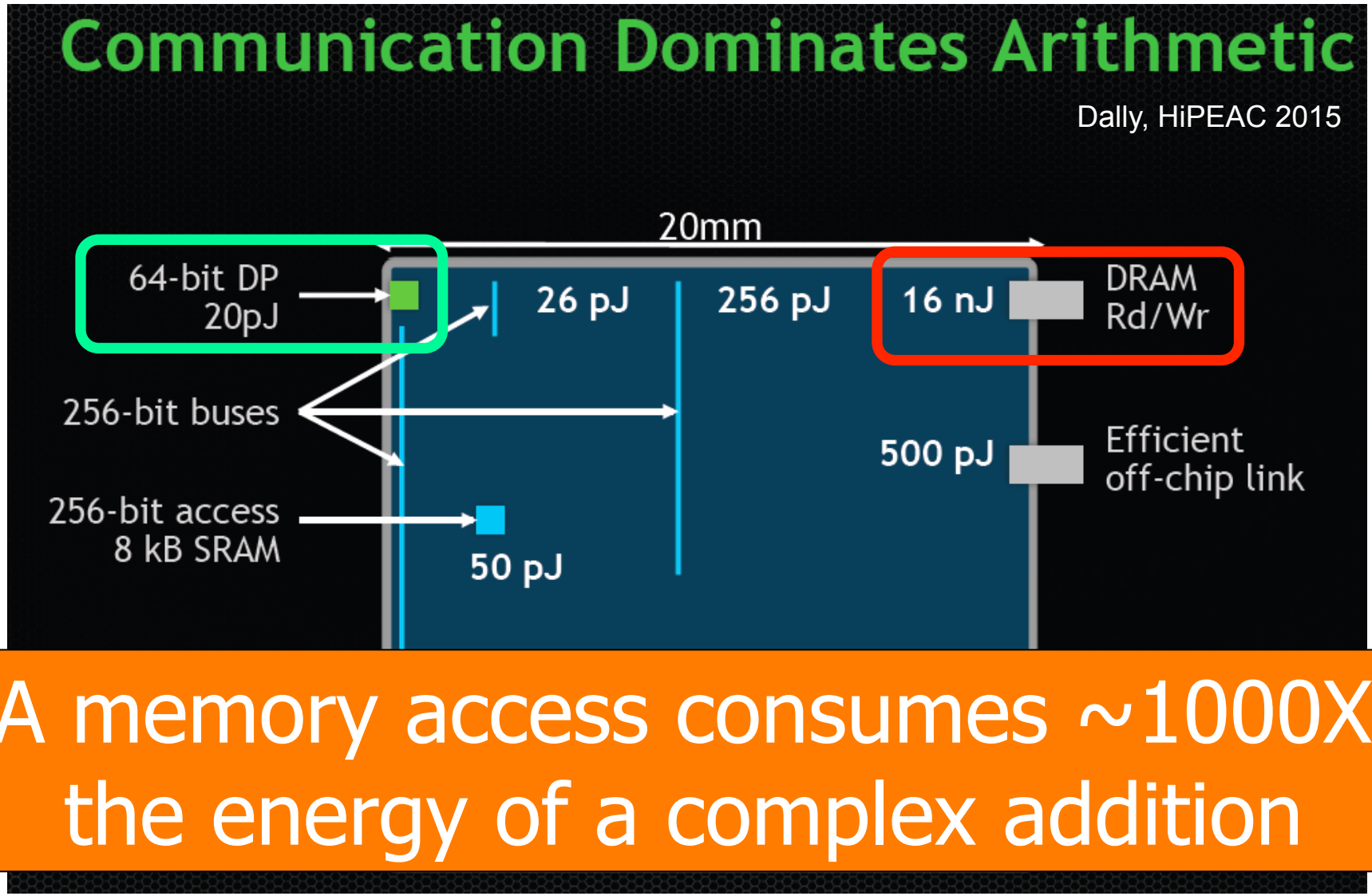
3. Data movement energy dominates compute

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

Data Movement vs. Computation Energy



Data Movement vs. Computation Energy

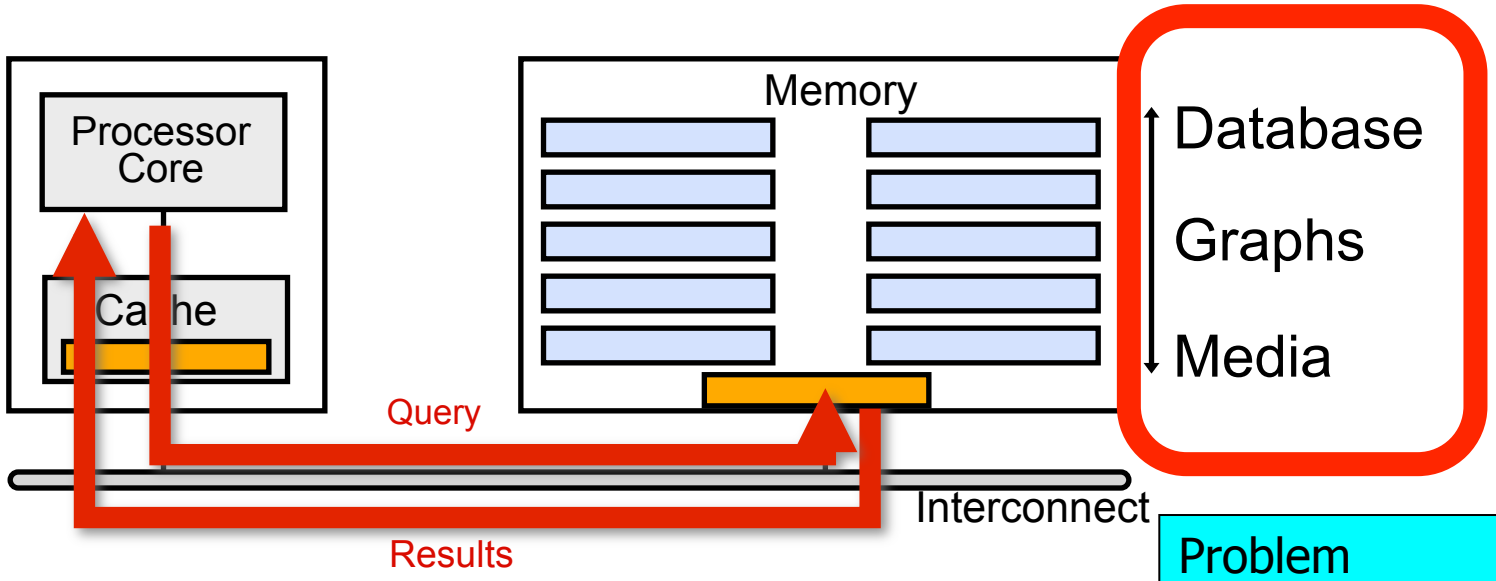


A memory access consumes ~1000X the energy of a complex addition

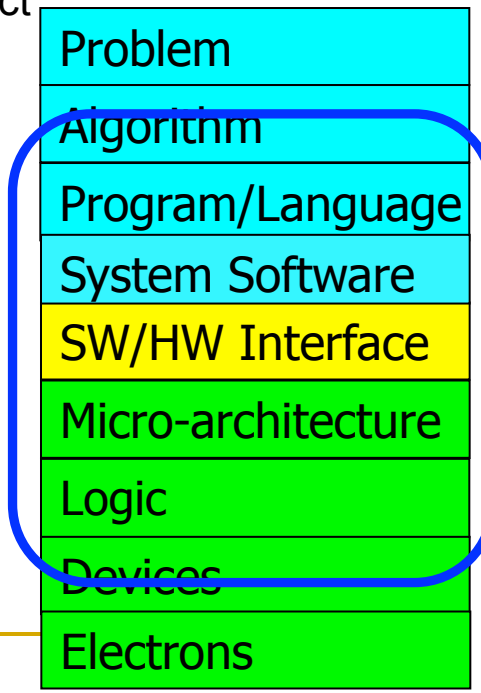
We Need A Paradigm Shift To …

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

Goal: In-Memory Computation Engine

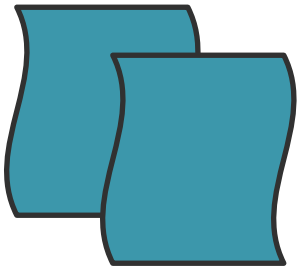


- Many questions … How do we design the:
  - compute-capable memory?
  - processor chip?
  - software interface?
  - system software and languages?
  - algorithms?

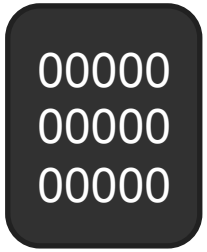


Starting Simple: Data Copy and Initialization

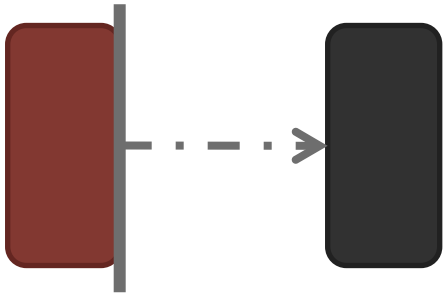
memmove & memcpy: 5% cycles in Google's datacenter [Kanev+ ISCA'15]



Forking

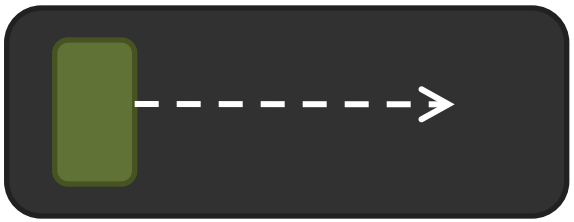


Zero initialization (e.g., security)



Checkpointing

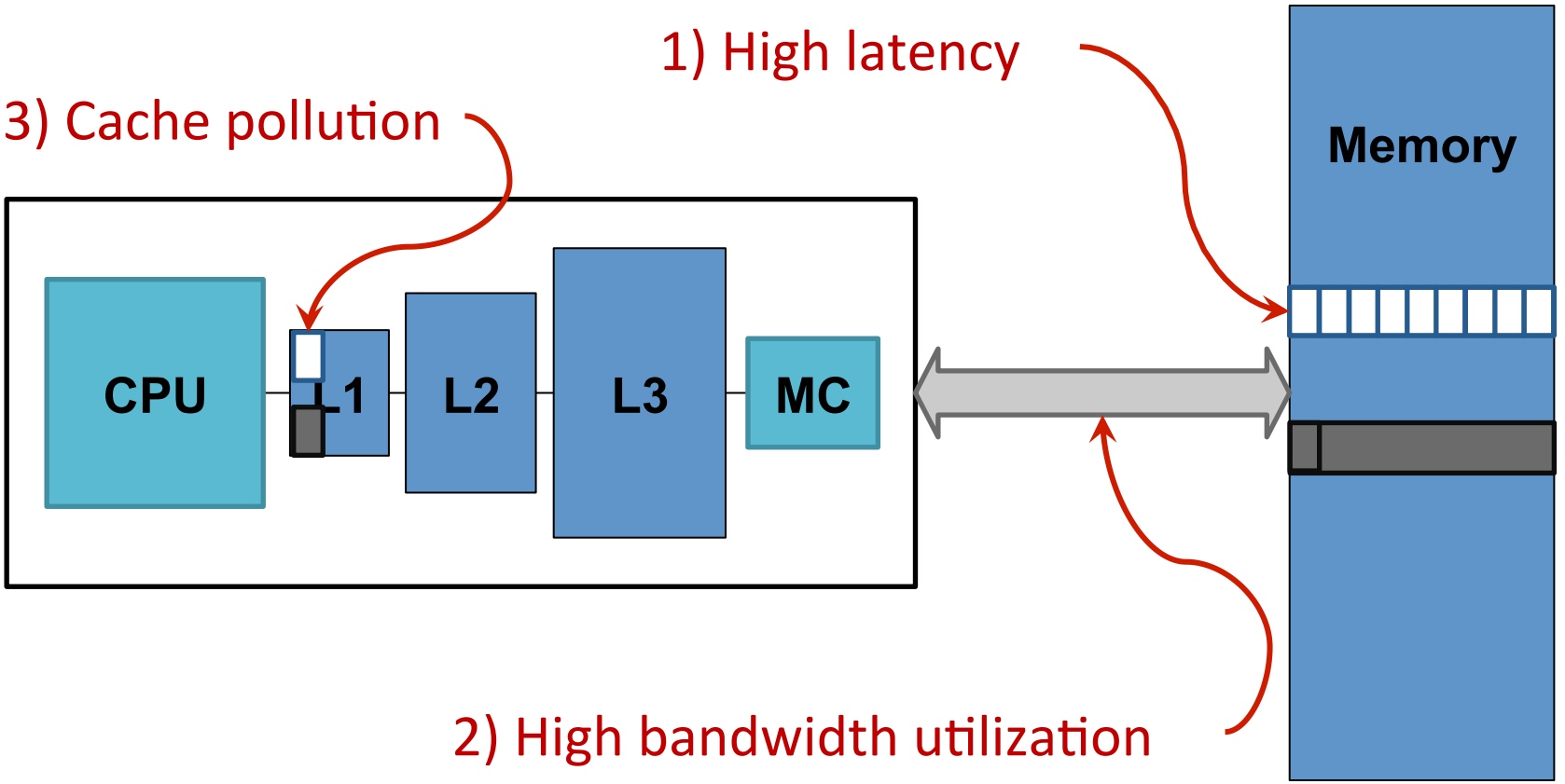
VM Cloning
Deduplication



Page Migration

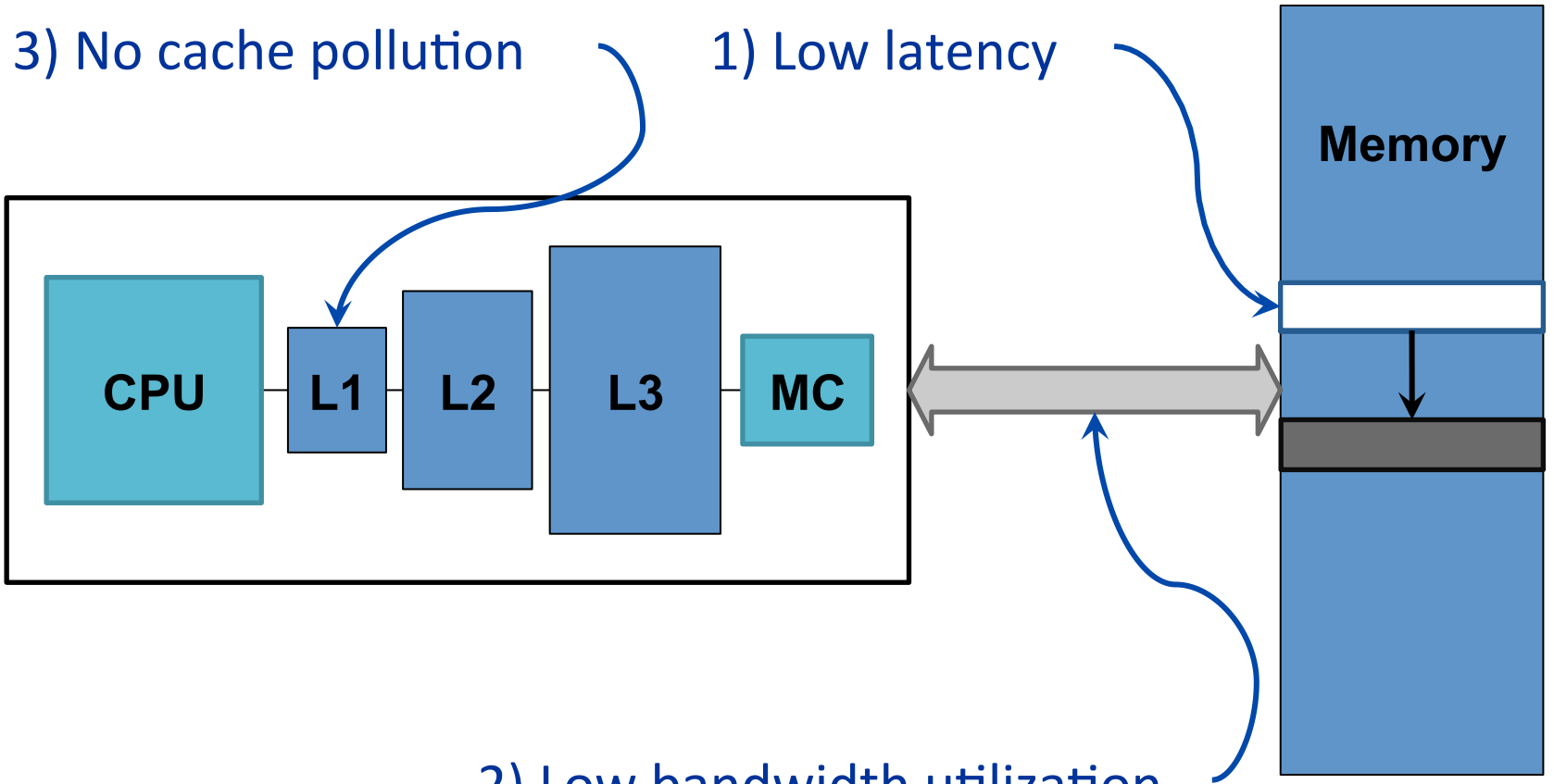
• • •
Many more

Today's Systems: Bulk Data Copy



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

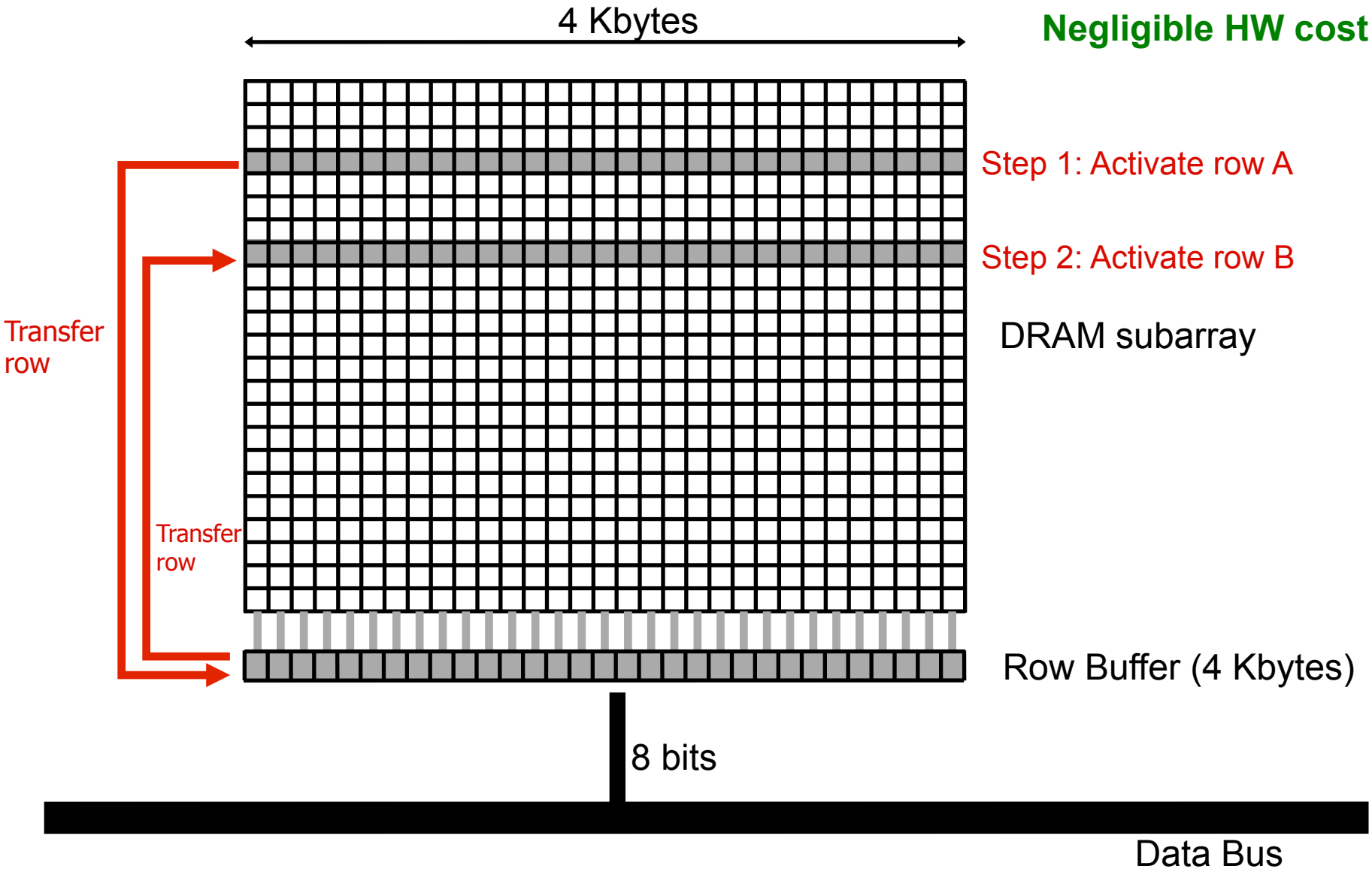
Future Systems: In-Memory Copy



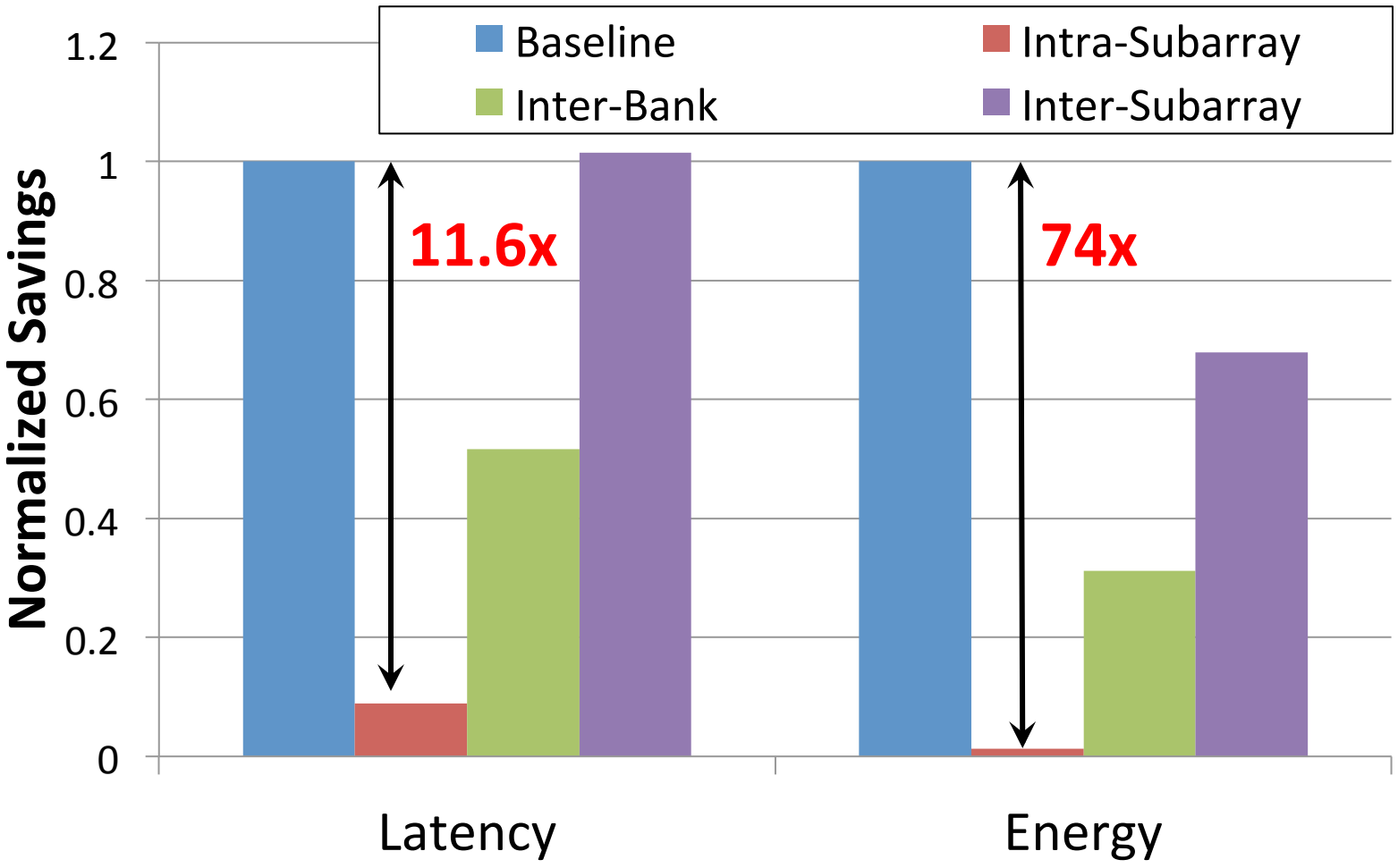
1046ns, 3.6uJ → 90ns, 0.04uJ

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

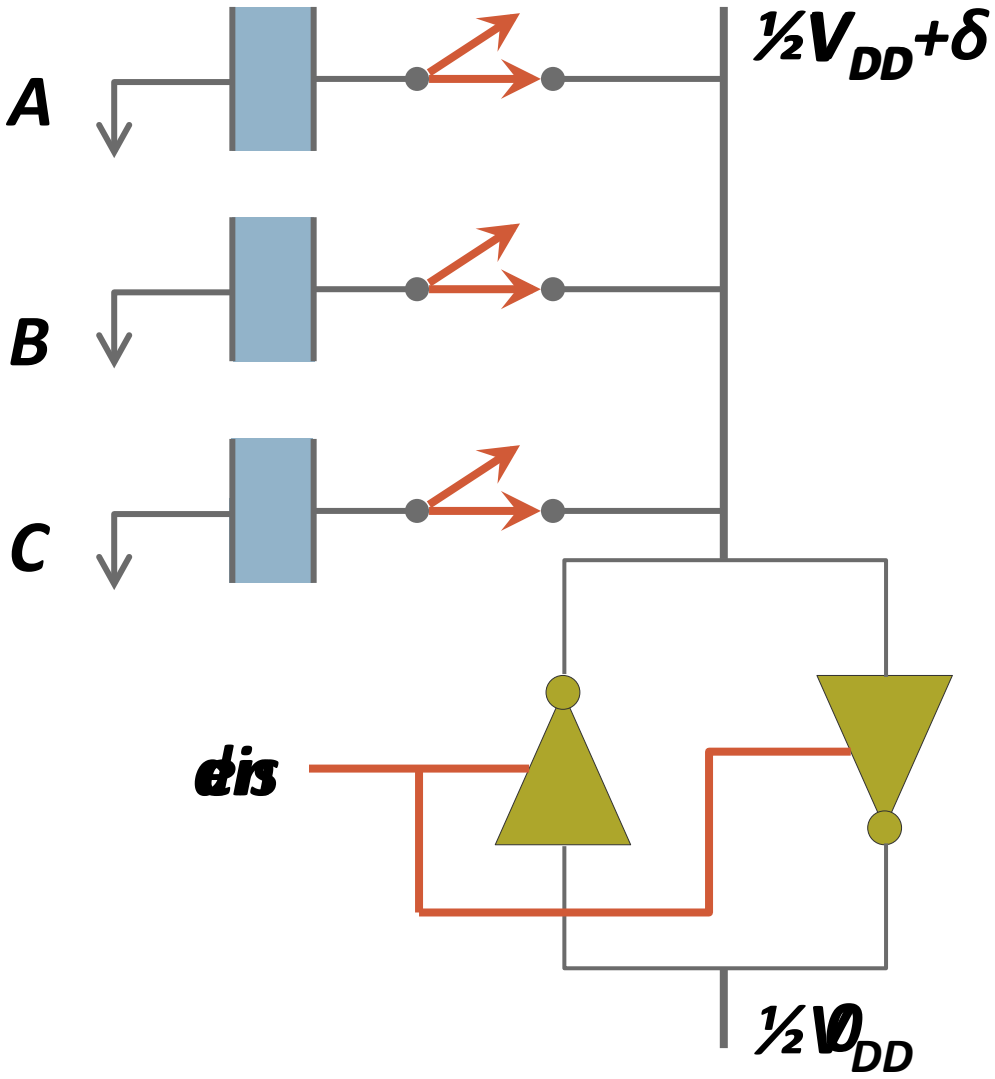
Vivek Seshadri vseshadr@cs.cmu.edu
Yoongu Kim yoongukim@cmu.edu
Chris Fallin* cfallin@c1f.net
Donghyuk Lee donghyuk1@cmu.edu
Rachata Ausavarungnirun rachata@cmu.edu
Gennady Pekhimenko gpekhime@cs.cmu.edu
Yixin Luo yixinluo@andrew.cmu.edu
Onur Mutlu onur@cmu.edu
Phillip B. Gibbons† phillip.b.gibbons@intel.com
Michael A. Kozuch† michael.a.kozuch@intel.com
Todd C. Mowry tcm@cs.cmu.edu
Carnegie Mellon University †Intel Pittsburgh

(Truly) In-Memory Computation

- Similarly, we can support in-DRAM AND, OR, NOT, MAJ
- At low cost
- Using analog behavior of memory
- 30-60X performance and energy improvement
  - Seshadri+, "In-DRAM Bulk Bitwise AND and OR," CAL 2016.
  - Seshadri+, "Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM," arxiv 2016.

- 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) + ~C(AB)

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

In-DRAM Bulk Bitwise AND/OR Operation

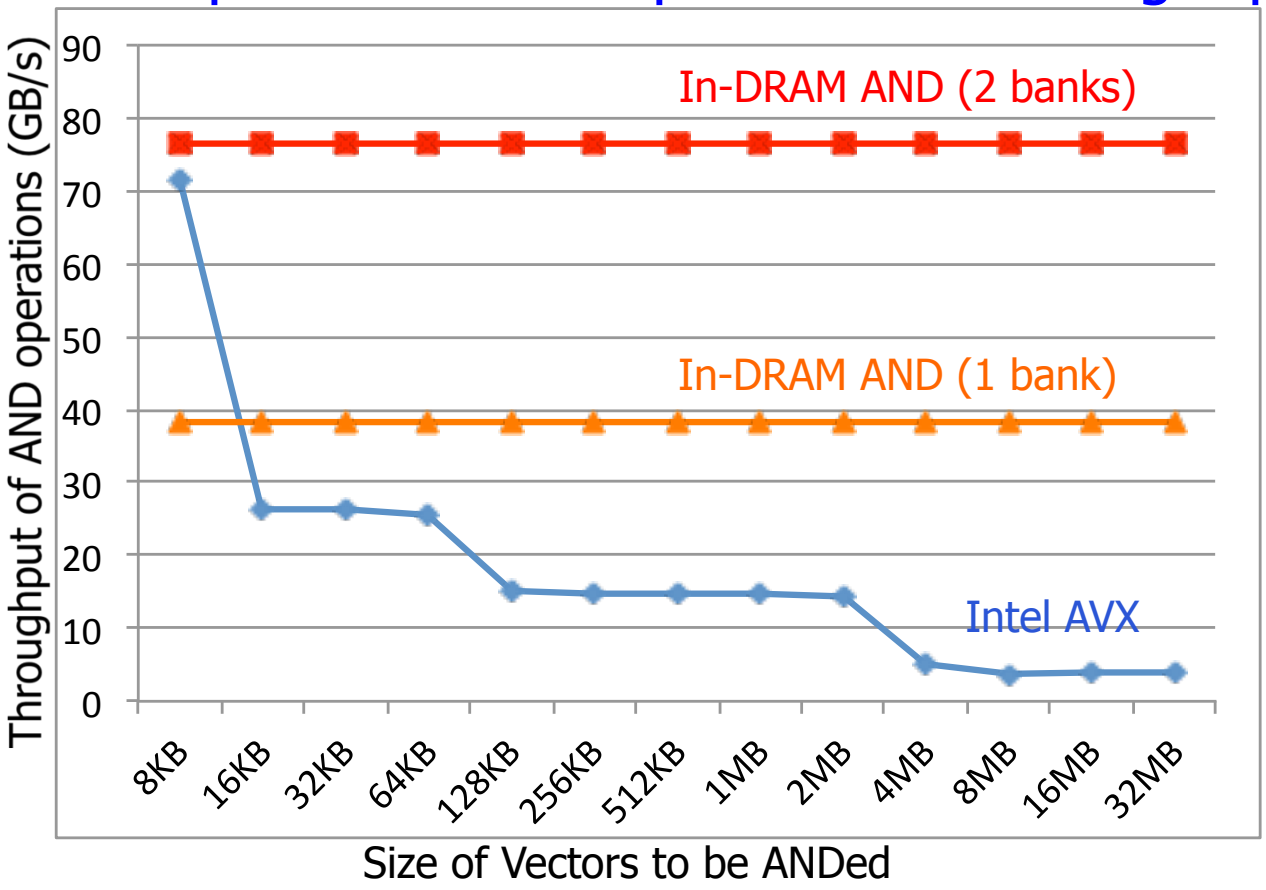
- BULKAND A, B → C
- Semantics: Perform a bitwise AND of two rows A and B and store the result in row C

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

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

In-DRAM AND/OR Results

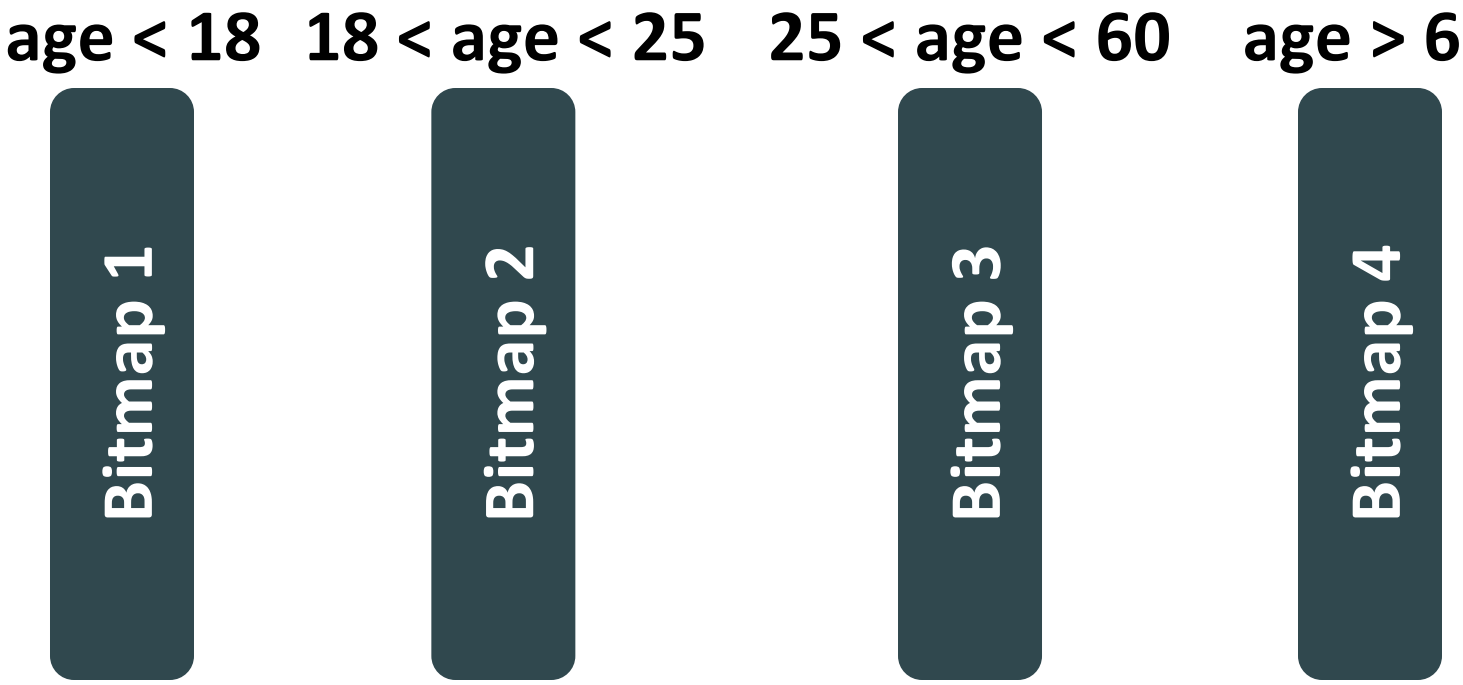
- 20X improvement in AND/OR throughput vs. Intel AVX
- 50.5X reduction in memory energy consumption
- At least 30% performance improvement in range queries



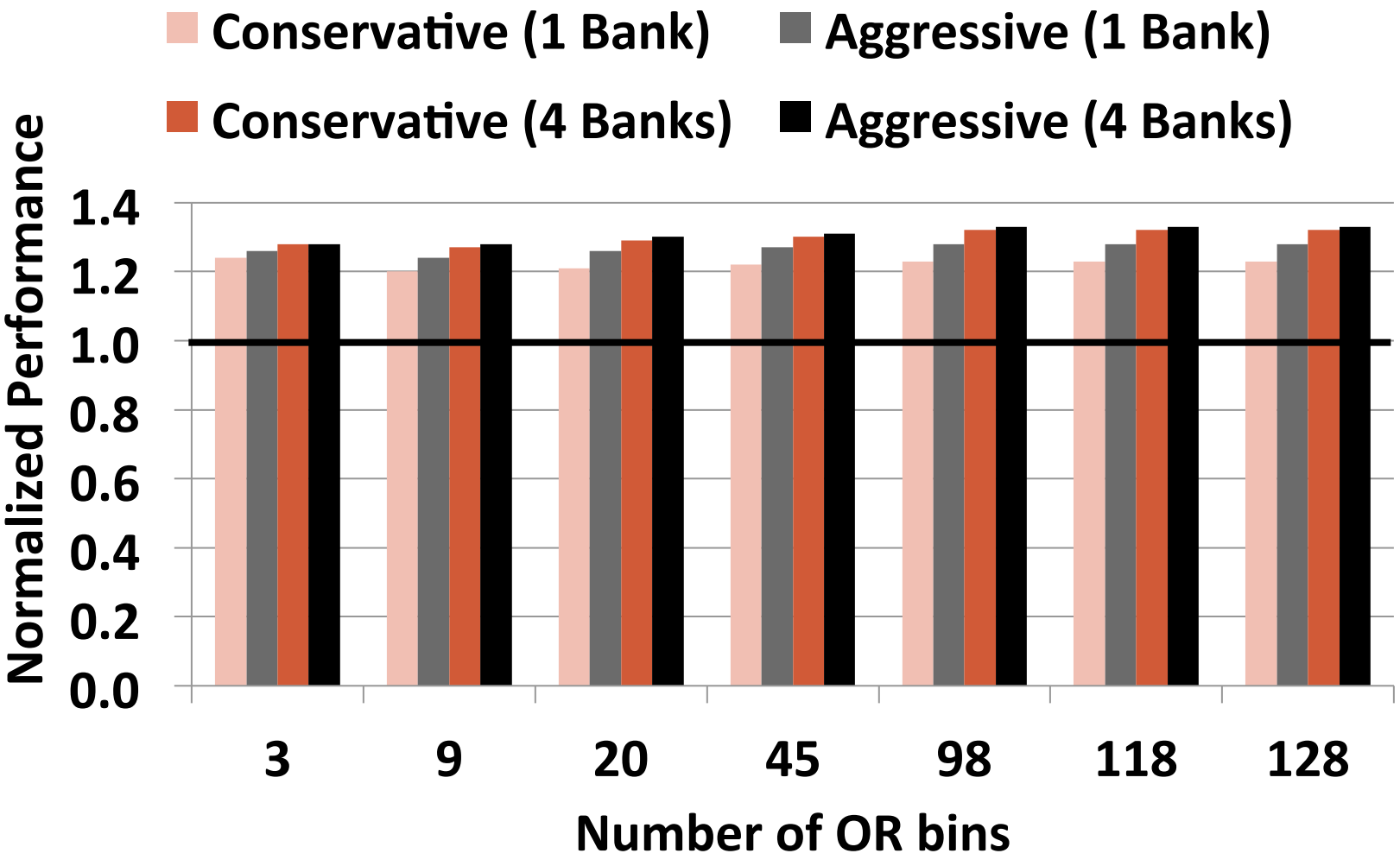
Seshadri+, "Fast Bulk Bitwise AND and OR in DRAM", IEEE CAL 2015.

Bitmap Index

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



Performance Evaluation



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

In-DRAM NOT: Dual Contact Cell

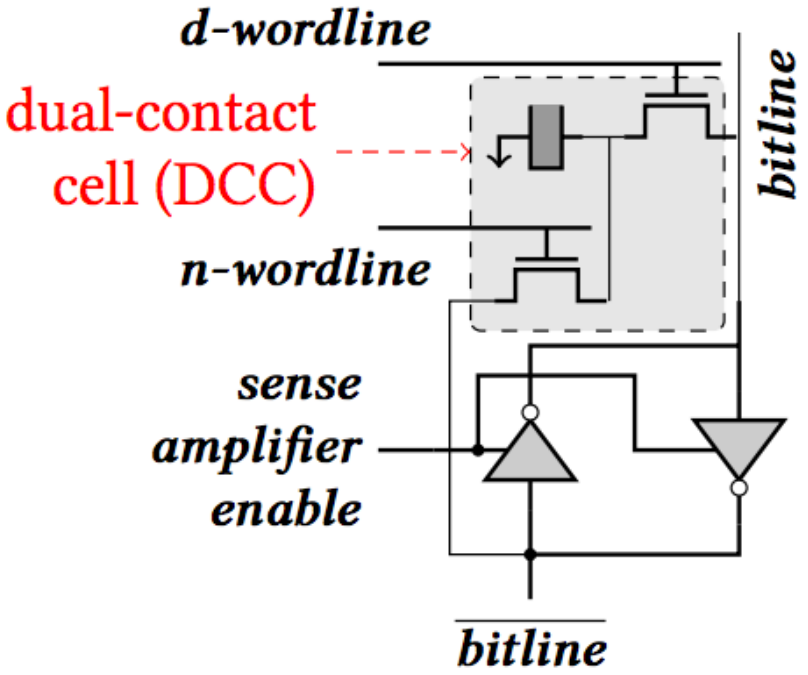


Figure 5: A dual-contact cell connected to both ends of a sense amplifier

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

Seshadri+, "Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM," arxiv 2016.

In-DRAM NOT Operation

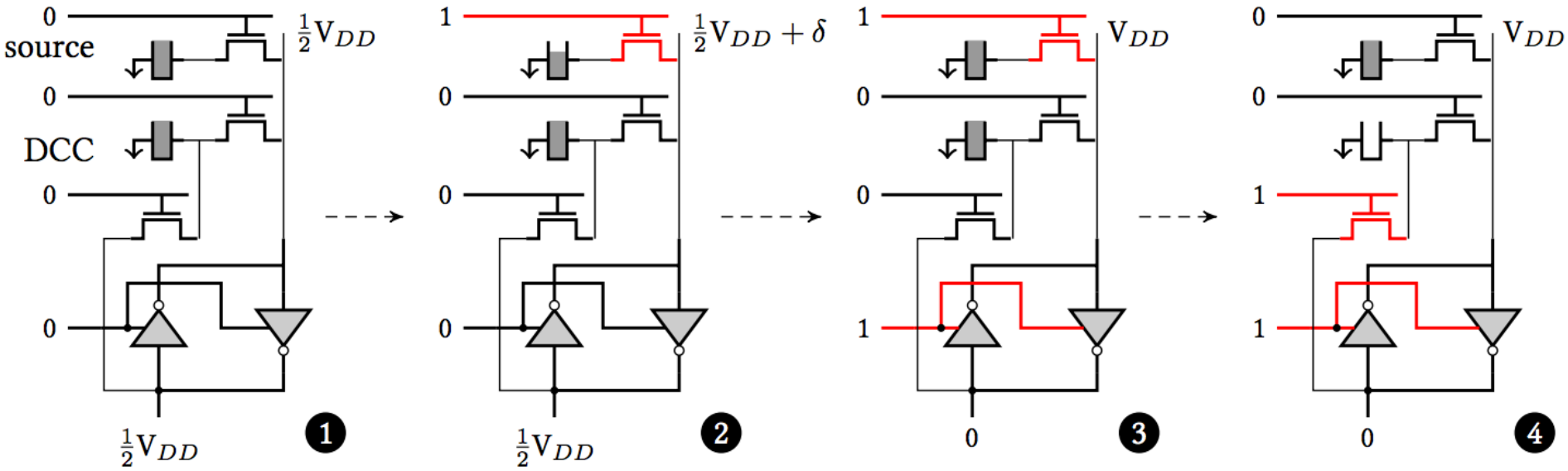


Figure 6: Bitwise NOT using a dual contact capacitor

Seshadri+, "Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM," arxiv 2016.

Energy of In-DRAM Bitwise Operations

	Interface	not	and/or	nand/nor	xor/xnor
Energy (nJ/KB)	DDR3	93.7	137.9	137.9	137.9
	Buddy	1.6	3.2	4.0	5.5
	(↓)	59.5X	43.9X	35.1X	25.1X

Table 3: Comparison of energy for various groups of bitwise operations. (↓) indicates reduction in energy of Buddy over the traditional DDR3 interface.

Seshadri+, "Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM," arxiv 2016.

An Example Result

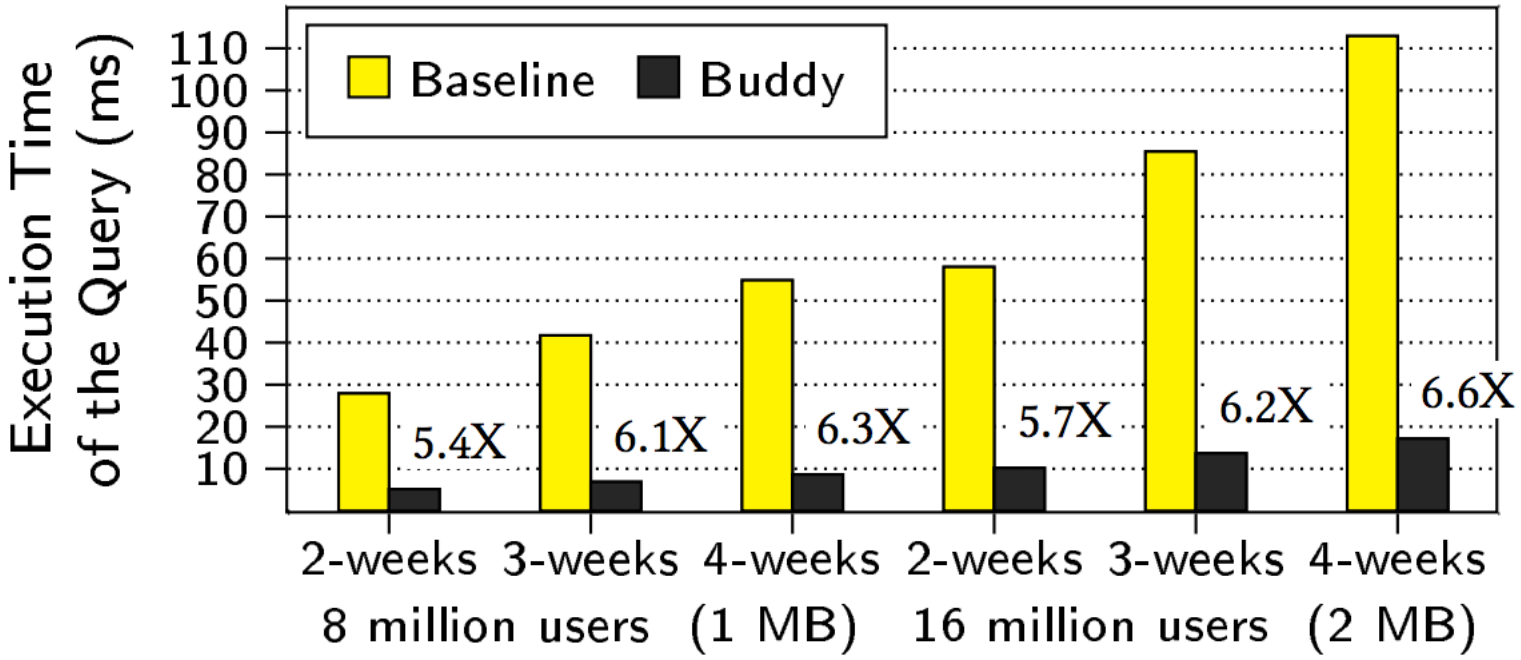


Figure 10: Performance of Buddy for bitmap indices The values on top of each bar indicates the factor reduction in execution time due to Buddy.

Seshadri+, "Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM," arxiv 2016.

Buddy-RAM: In-DRAM Bitwise Operations

- Vivek Seshadri et al., "Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM," arxiv.org, Nov 2016.
  - https://arxiv.org/pdf/1611.09988.pdf

Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM

Vivek Seshadri[1], Donghyuk Lee[2], Thomas Mullins[3], Hasan Hassan[4], Amirali Boroumand[5], Jeremie Kim[5], Michael A. Kozuch[6], Onur Mutlu[7], Phillip B. Gibbons[5], Todd C. Mowry[5]

[1]Microsoft Research [2]NVIDIA Research [3]Intel [4]TOBB University of Economics and Technology

[5]Carnegie Mellon University [6]Intel Pittsburgh [7]ETH Zurich

Another Example: In-Memory Graph Processing

- Large graphs are everywhere (circa 2015)

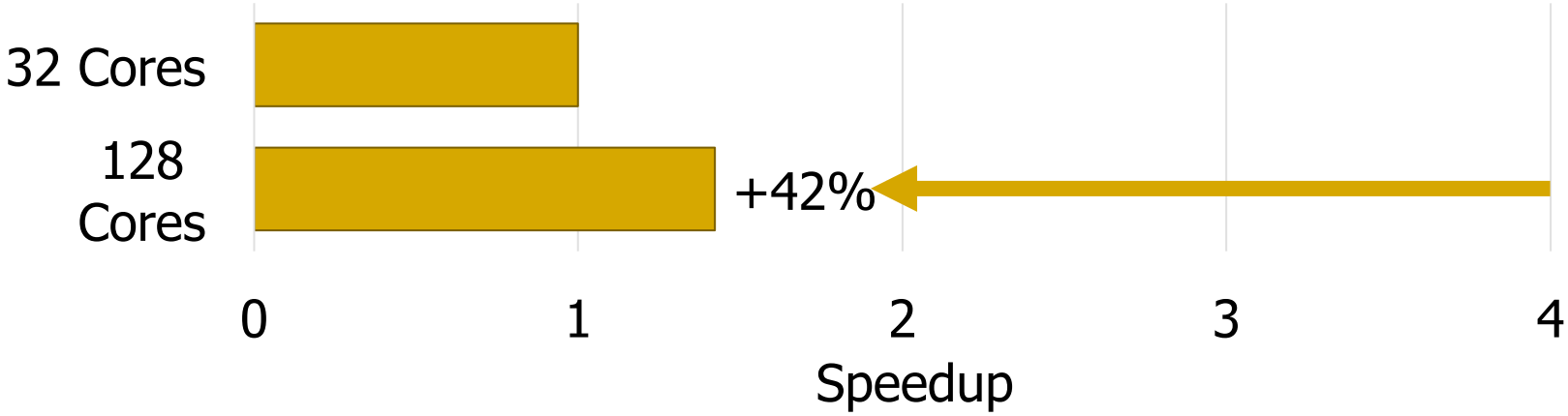
36 Million
Wikipedia Pages

1.4 Billion
Facebook Users

300 Million
Twitter Users

30 Billion
Instagram Photos

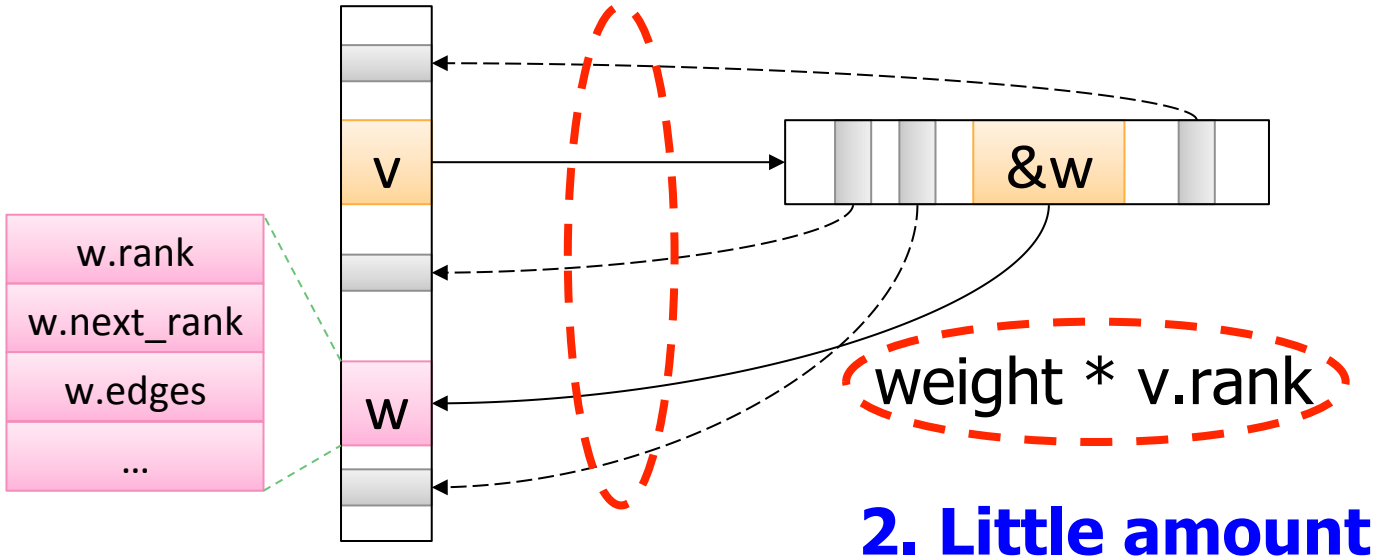
- Scalable large-scale graph processing is challenging



Key Bottlenecks in Graph Processing

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

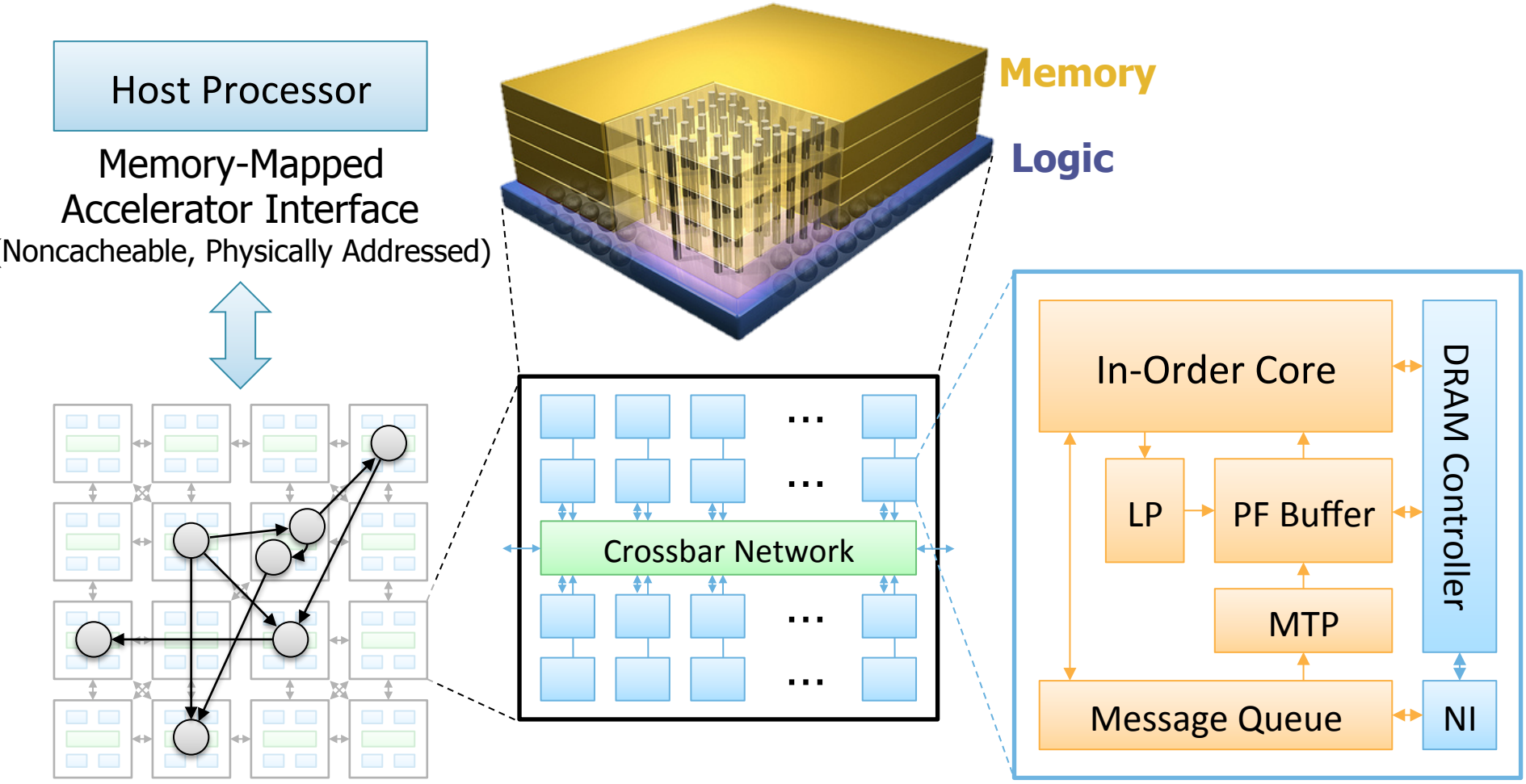
1. Frequent random memory accesses



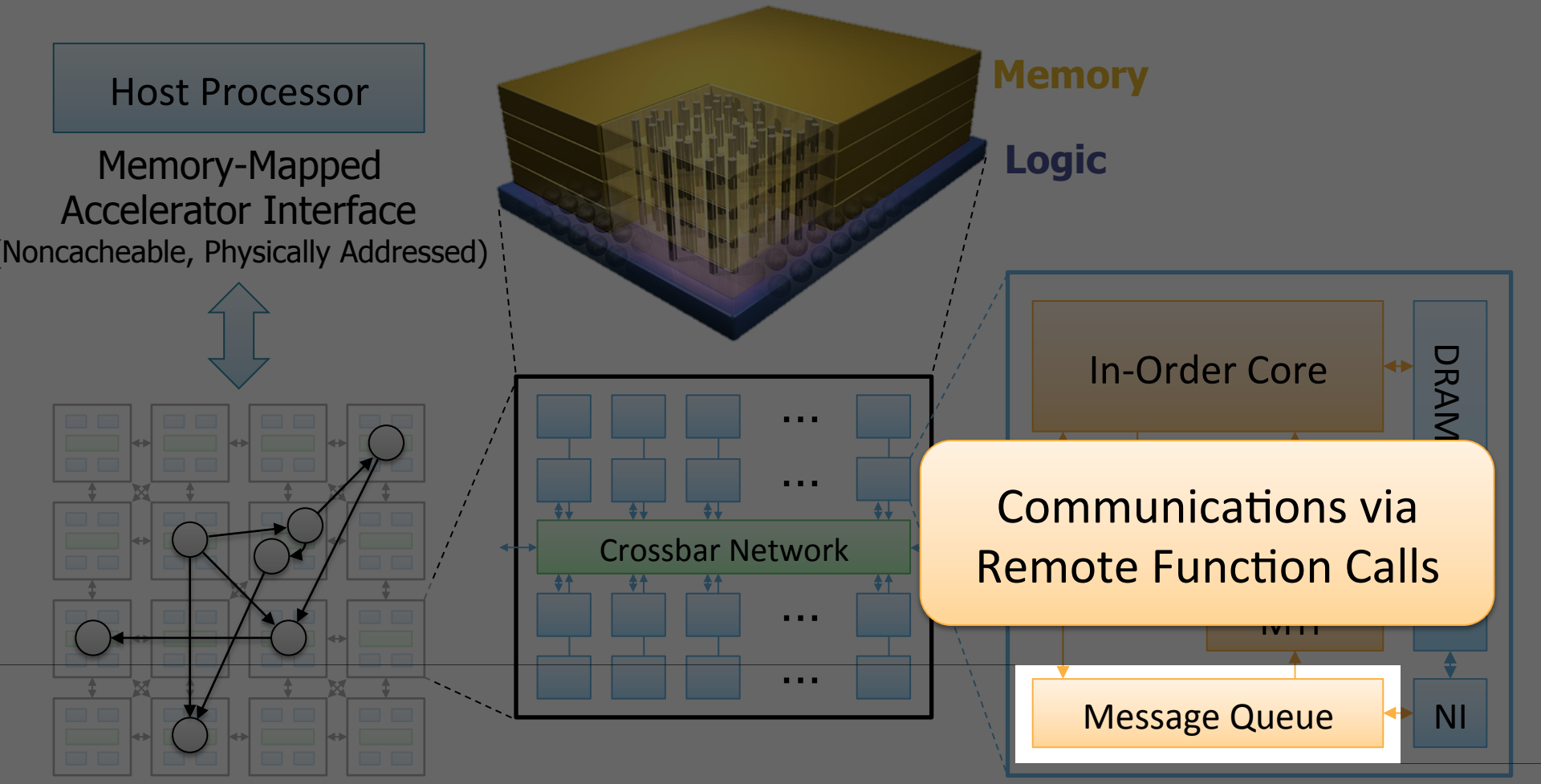
2. Little amount of computation

Tesseract System for Graph Processing

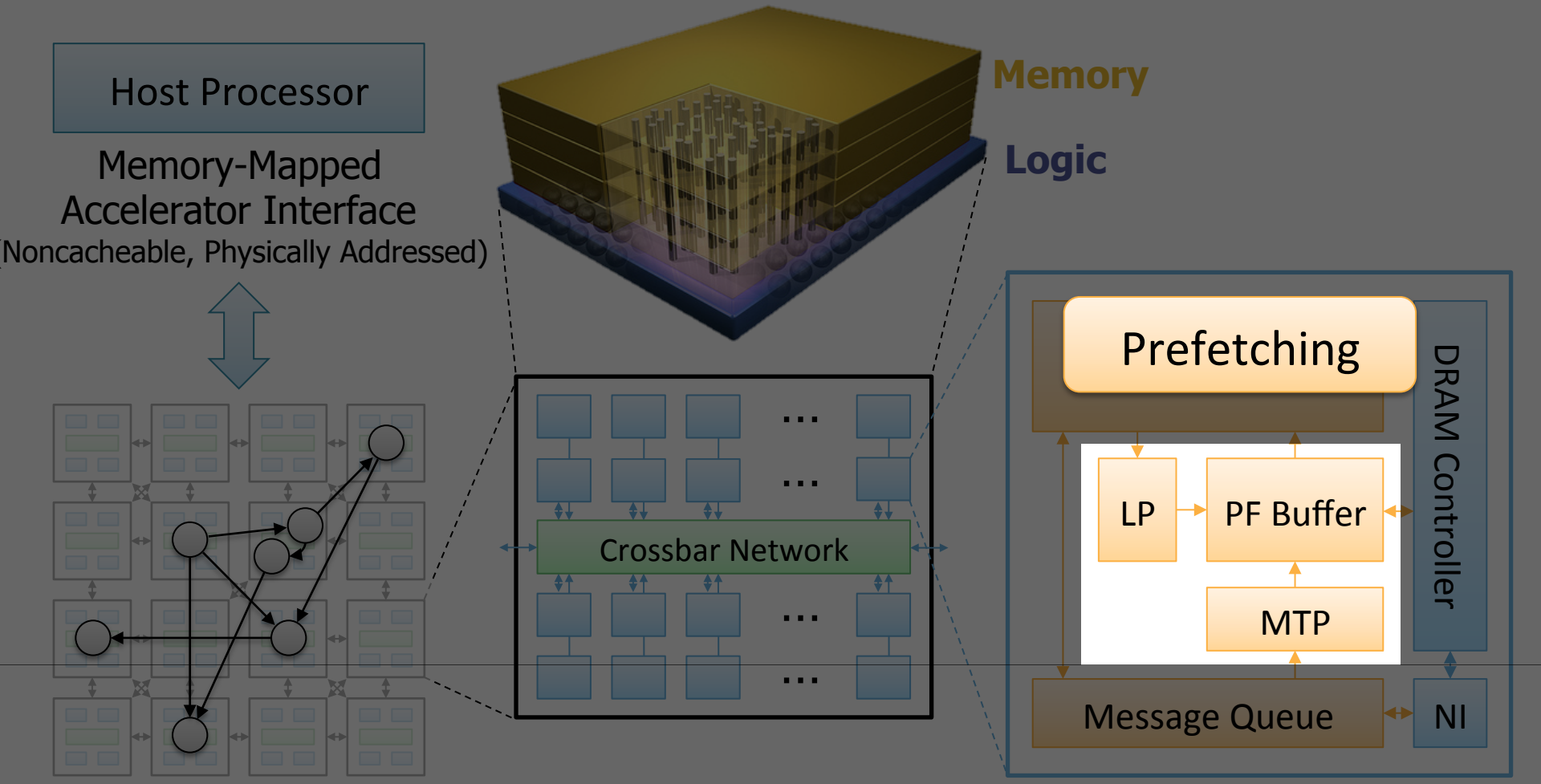
Interconnected set of 3D-stacked memory+logic chips with simple cores



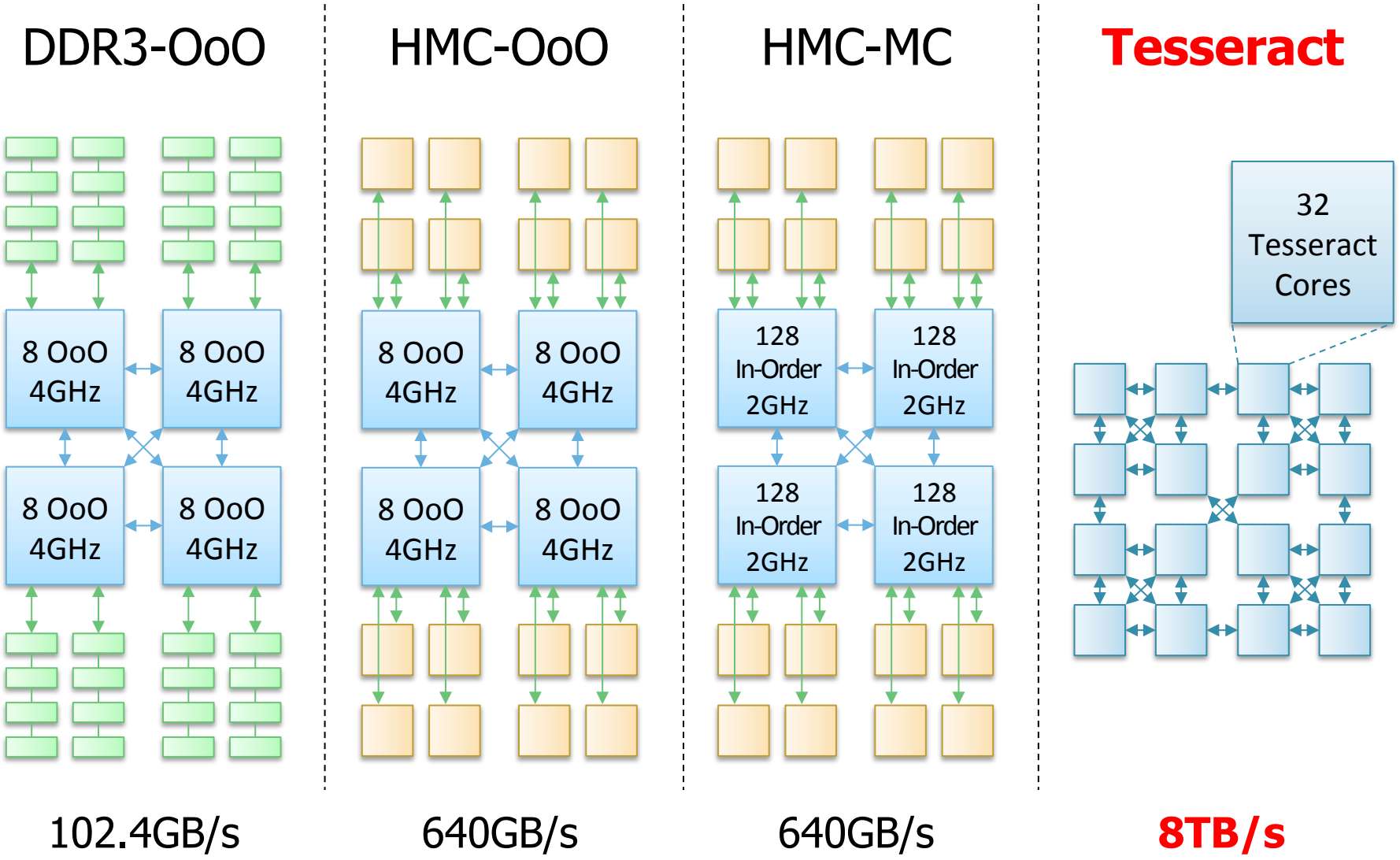
Tesseract System for Graph Processing



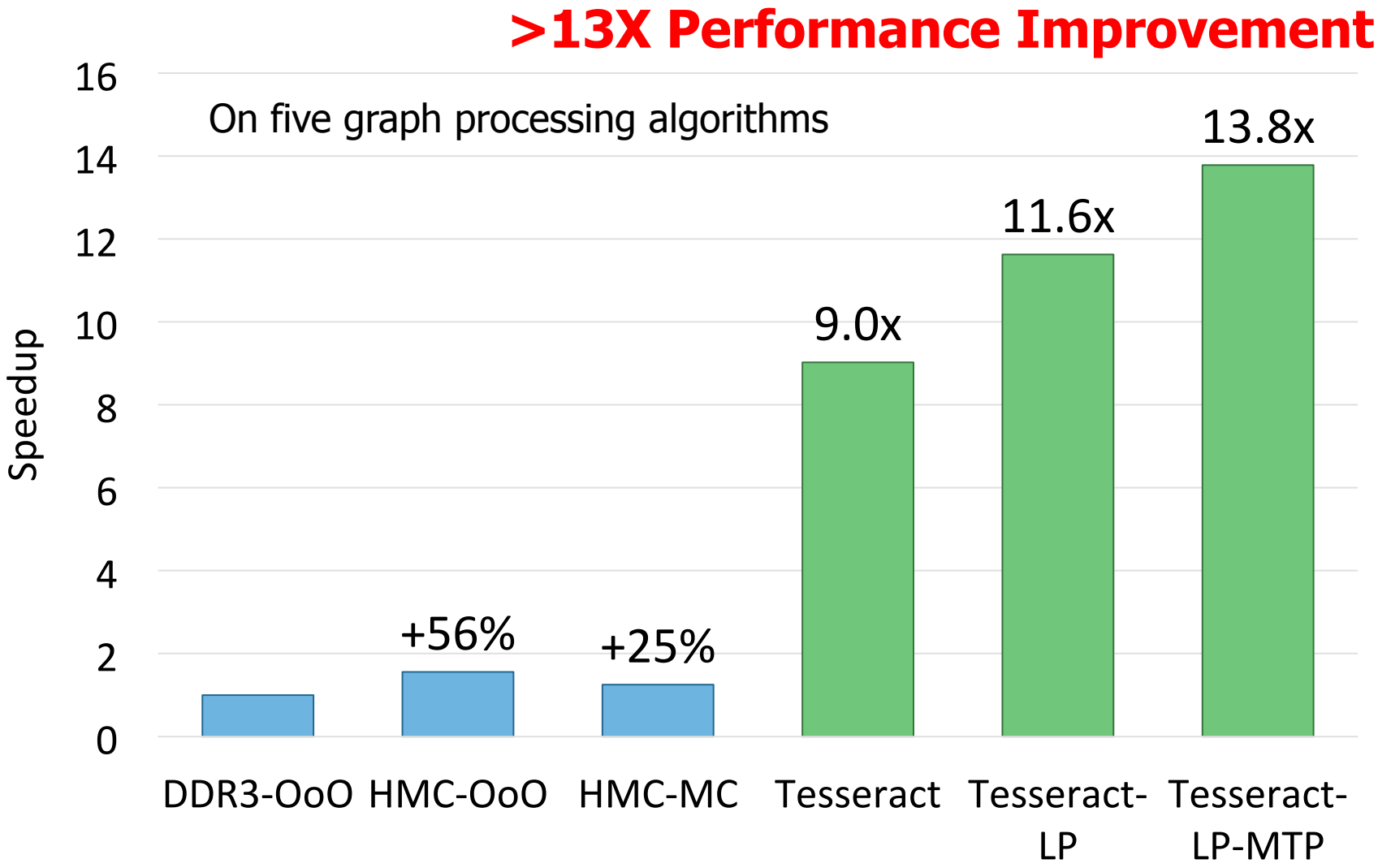
Tesseract System for Graph Processing



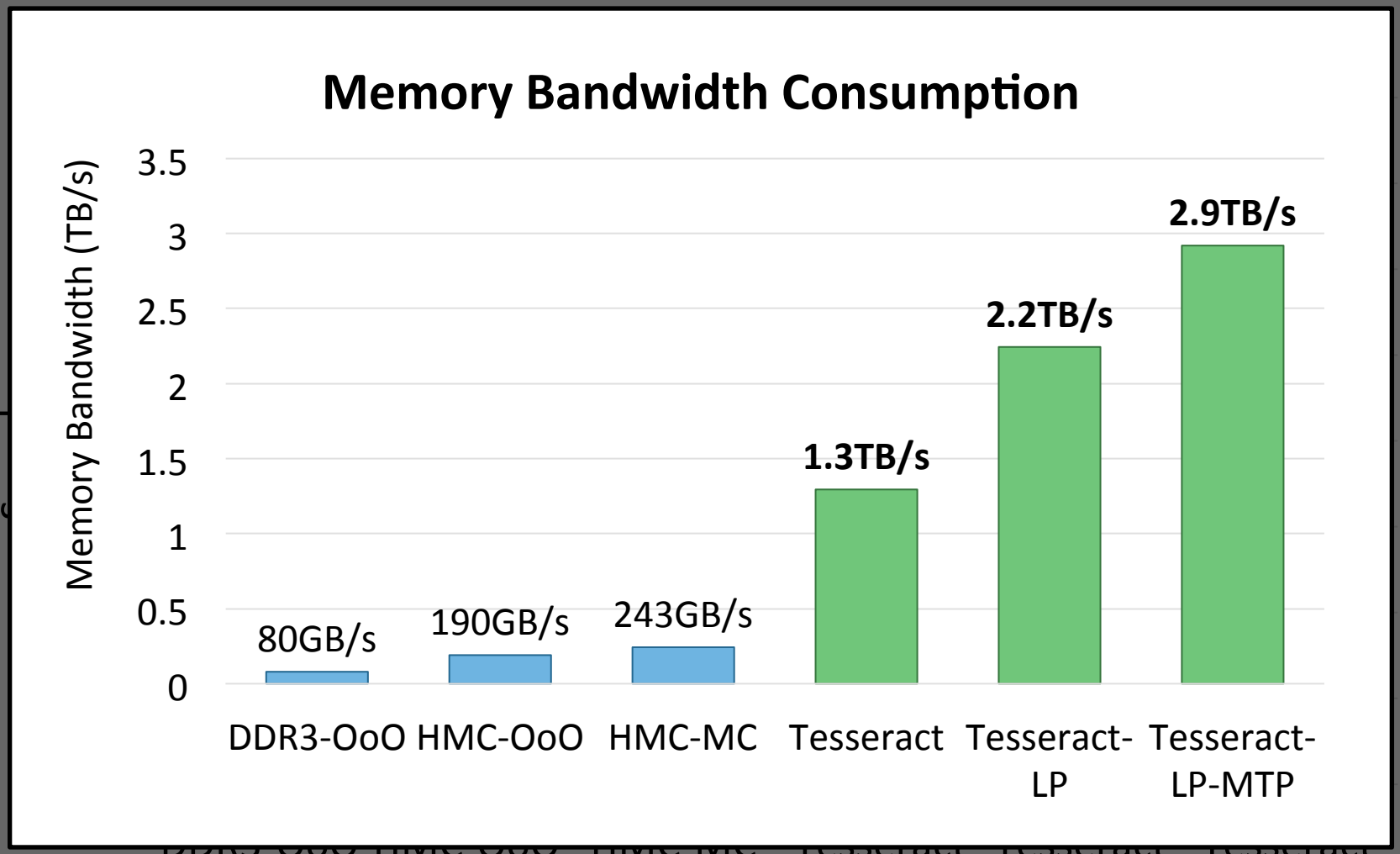
Evaluated Systems



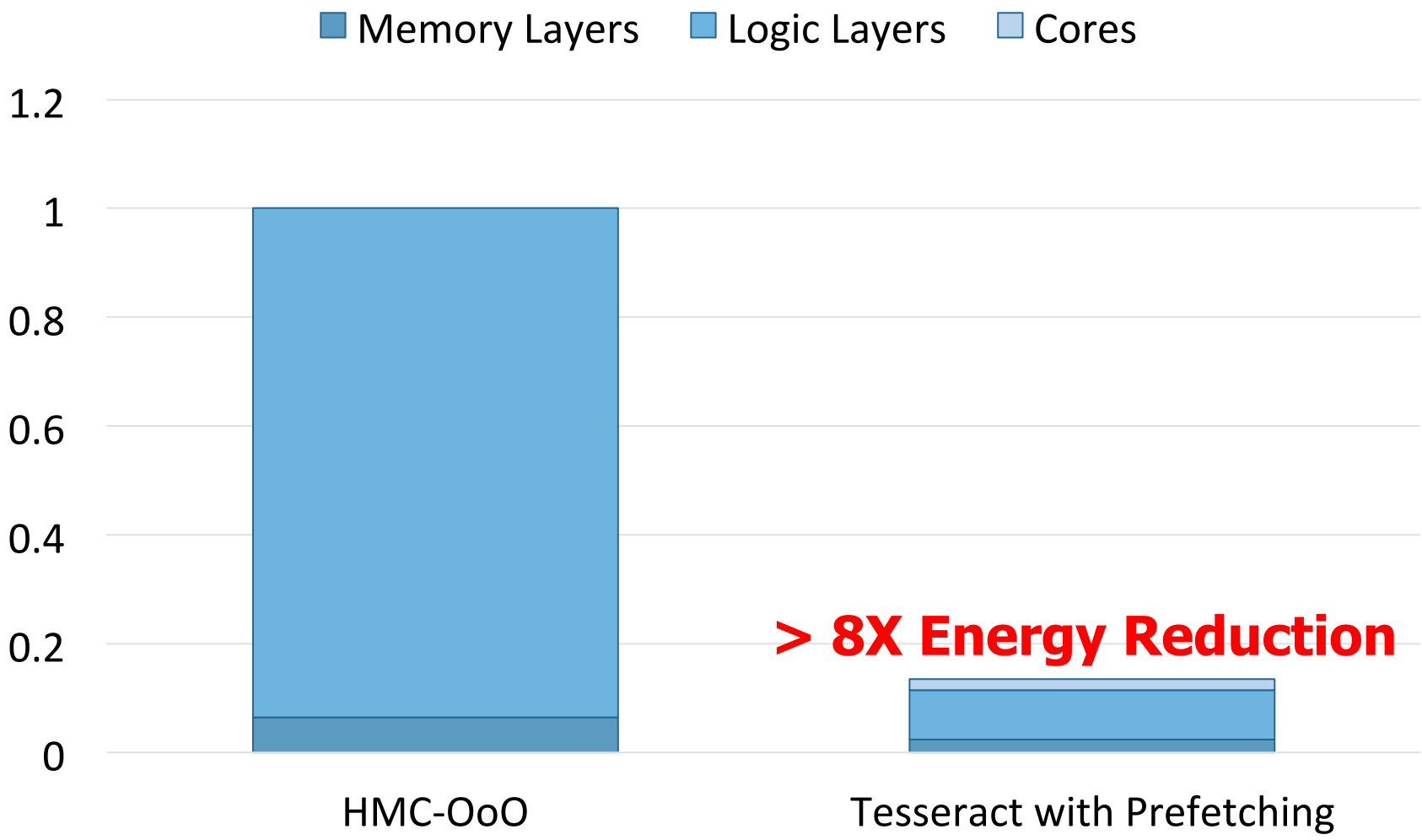
Tesseract Graph Processing Performance



Tesseract Graph Processing Performance



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

Challenge and Opportunity for Future

Fundamentally Energy-Efficient (Data-Centric) Computing Architectures

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

PEI: PIM-Enabled Instructions (Example)

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

pim.add r1, (r2)

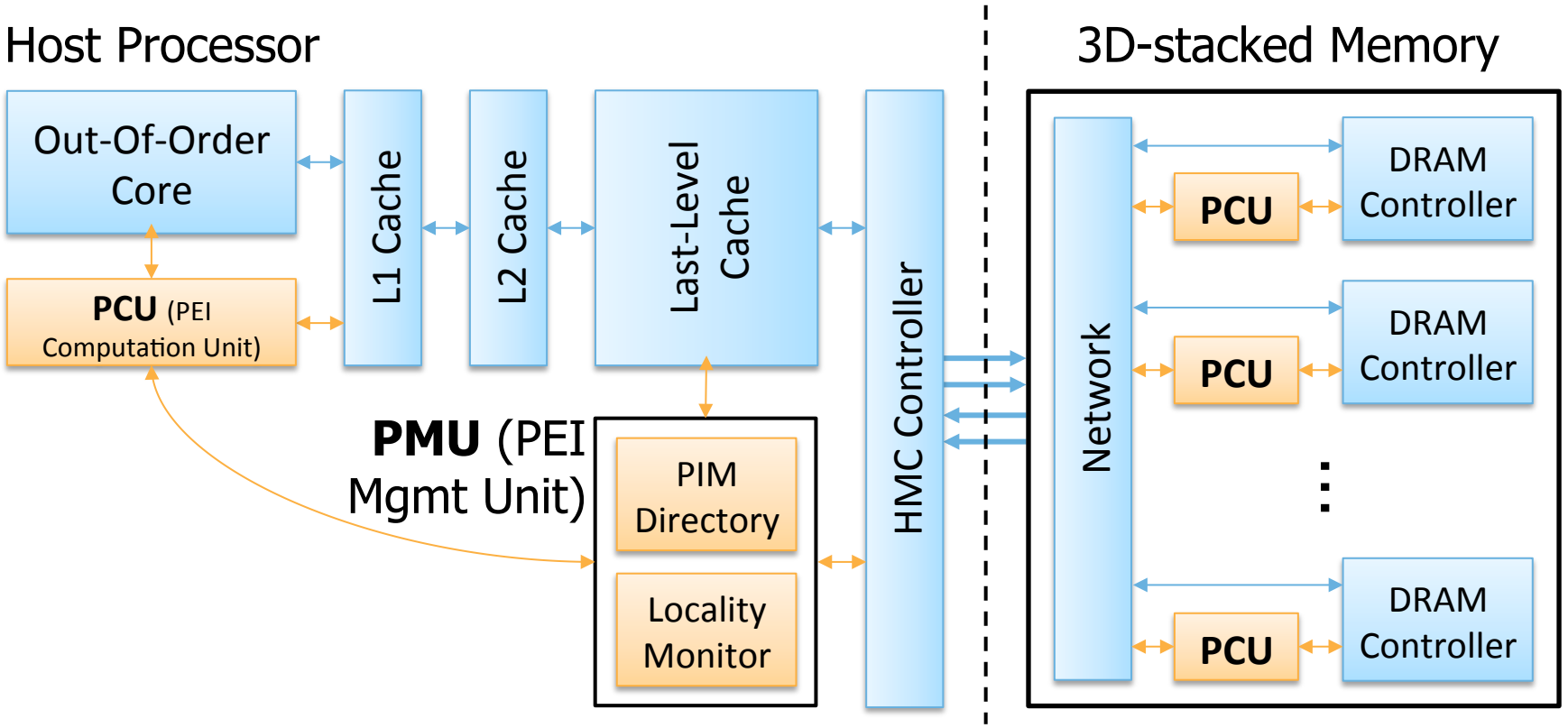
pfence

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)

Example (Abstract) PEI uArchitecture



Example PEI uArchitecture

PEI: Initial Evaluation Results

- Initial evaluations with 10 emerging data-intensive workloads
  - Large-scale graph processing
  - In-memory data analytics
  - Machine learning and data mining
  - Three input sets (small, medium, large) for each workload to analyze the impact of data locality

Table 2: Baseline Simulation Configuration

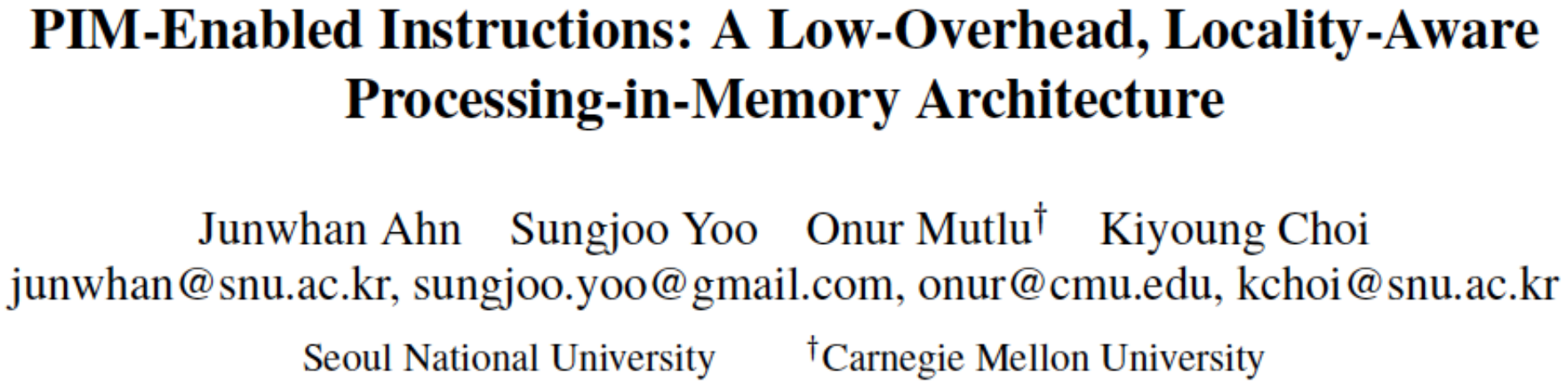
Component	Configuration
Core	16 out-of-order cores, 4 GHz, 4-issue
L1 I/D-Cache	Private, 32 KB, 4/8-way, 64 B blocks, 16 MSHRs
L2 Cache	Private, 256 KB, 8-way, 64 B blocks, 16 MSHRs
L3 Cache	Shared, 16 MB, 16-way, 64 B blocks, 64 MSHRs
On-Chip Network	Crossbar, 2 GHz, 144-bit links
Main Memory	32 GB, 8 HMCs, daisy-chain (80 GB/s full-duplex)
HMC	4 GB, 16 vaults, 256 DRAM banks [20]
– DRAM	FR-FCFS, tCL = tRCD = tRP = 13.75 ns [27]
– Vertical Links	64 TSVs per vault with 2 Gb/s signaling rate [23]

- Pin-based cycle-level x86-64 simulation

- **Performance Improvement and Energy Reduction:**
  - 47% average speedup with large input data sets
  - 32% speedup with small input data sets
  - 25% avg. energy reduction in a single node with large input data sets

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)]

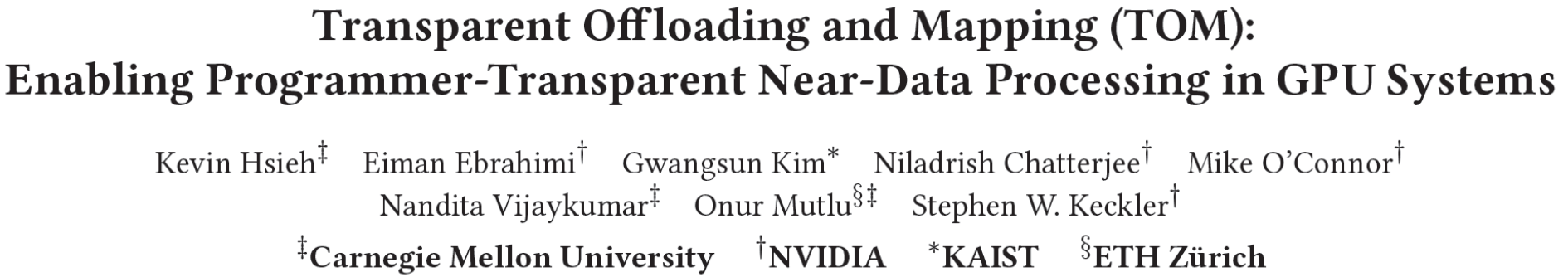


PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture

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

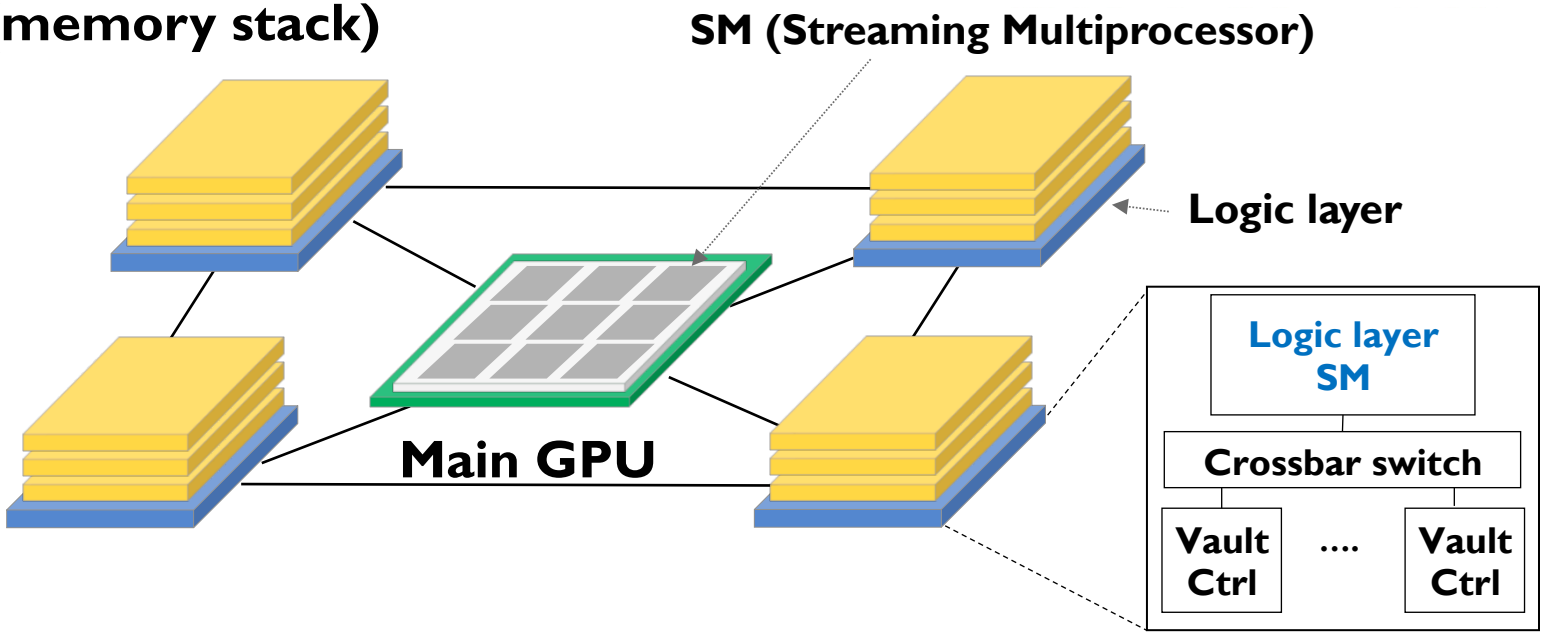
More on PIM Design: 3D-Stacked GPU I

- 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)]



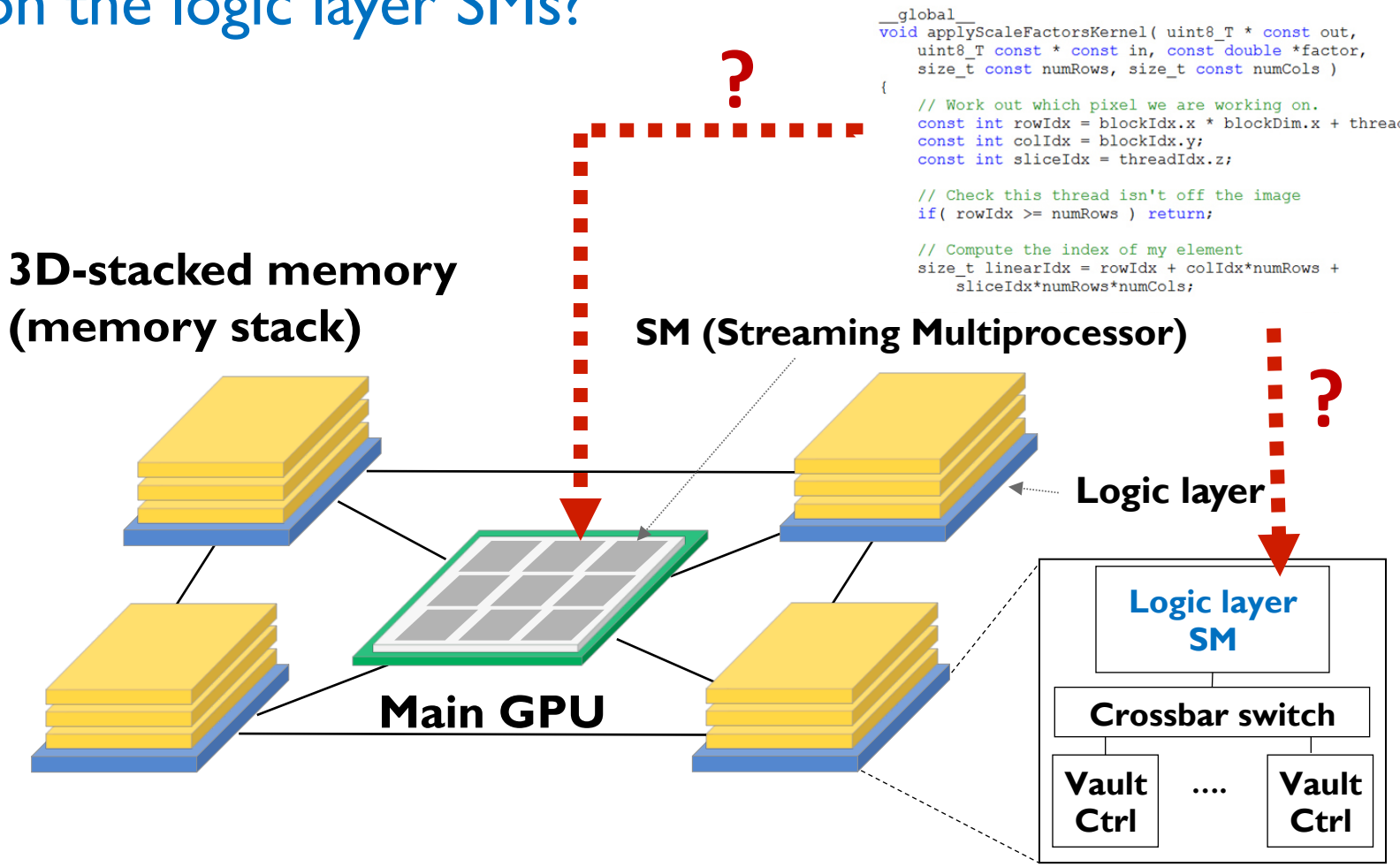
Key Challenge 1

```
__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;
```



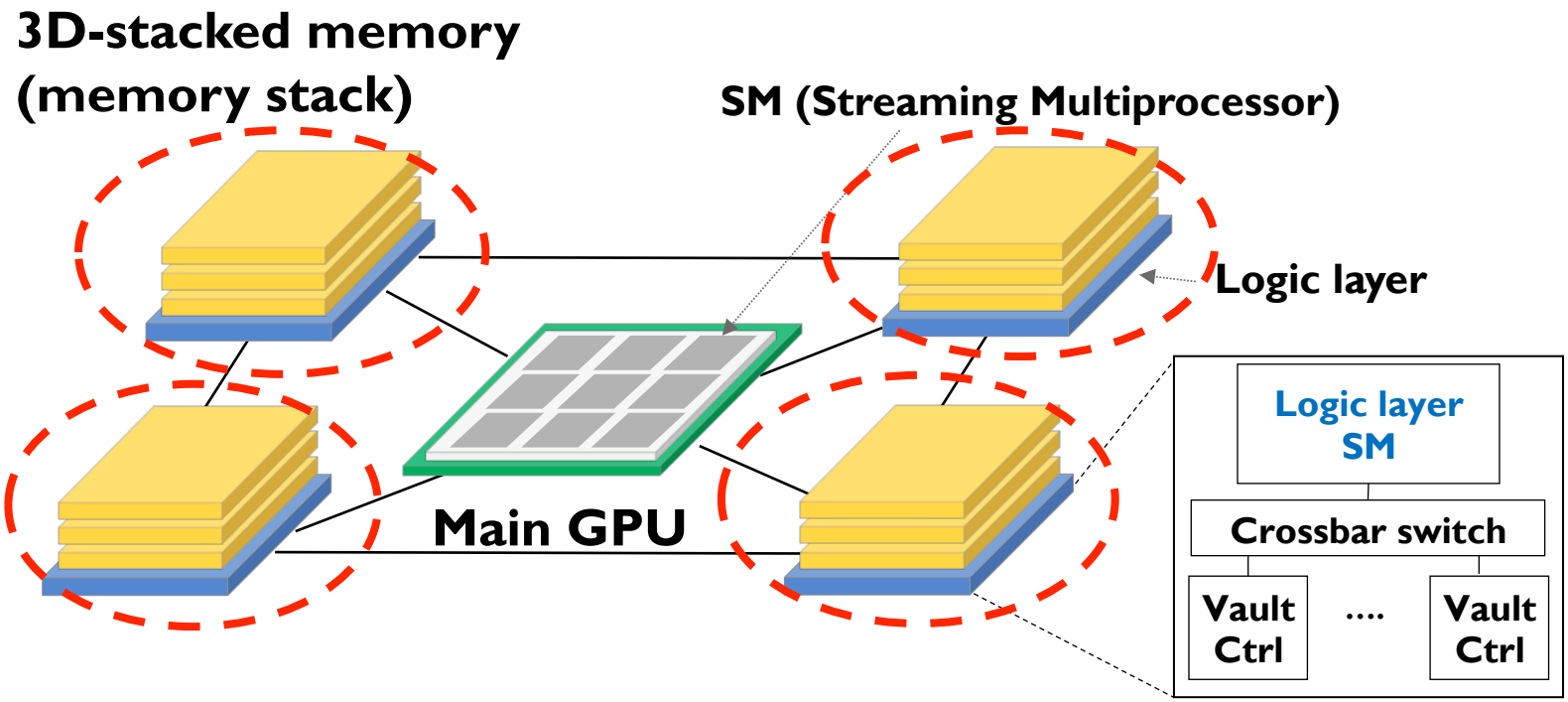
Key Challenge 1

- **Challenge 1:** Which operations should be executed on the logic layer SMs?



Key Challenge 2

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



More on PIM Design: 3D-Stacked GPU II

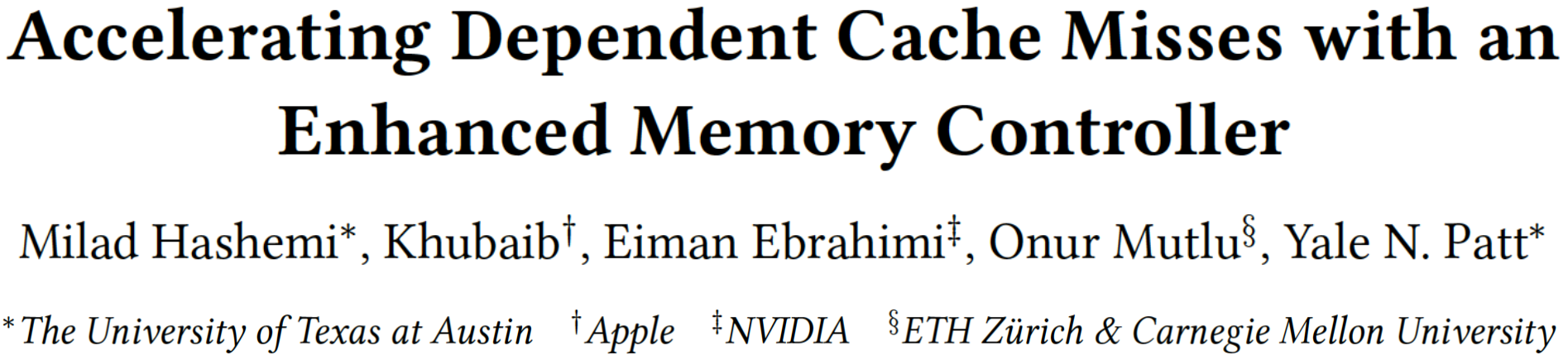
- Ashutosh Pattnaik, Xulong Tang, Adwait Jog, Onur Kayiran, Asit K. Mishra, Mahmut T. Kandemir, Onur Mutlu, and Chita R. Das, **"Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities"**
  Proceedings of the
  25th International Conference on Parallel Architectures and Compilation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

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

Ashutosh Pattnaik[1] Xulong Tang[1] Adwait Jog[2] Onur Kayıran[3]
Asit K. Mishra[4] Mahmut T. Kandemir[1] Onur Mutlu[5,6] Chita R. Das[1]
[1]Pennsylvania State University [2]College of William and Mary
[3]Advanced Micro Devices, Inc. [4]Intel Labs [5]ETH Zürich [6]Carnegie Mellon University

More on PIM Design: Dependent Misses

- Milad Hashemi, Khubaib, Eiman Ebrahimi, Onur Mutlu, and Yale N. Patt, **"Accelerating Dependent Cache Misses with an Enhanced Memory Controller"**
  Proceedings of the
  43rd International Symposium on Computer Architecture (***ISCA***), Seoul, South Korea, June 2016.
  [Slides (pptx) (pdf)]
  [Lightning Session Slides (pptx) (pdf)]



Accelerating Dependent Cache Misses with an Enhanced Memory Controller

Milad Hashemi*, Khubaib†, Eiman Ebrahimi‡, Onur Mutlu§, Yale N. Patt*

*The University of Texas at Austin †Apple ‡NVIDIA §ETH Zürich & Carnegie Mellon University

More on PIM: 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*

More on PIM Design: Coherence

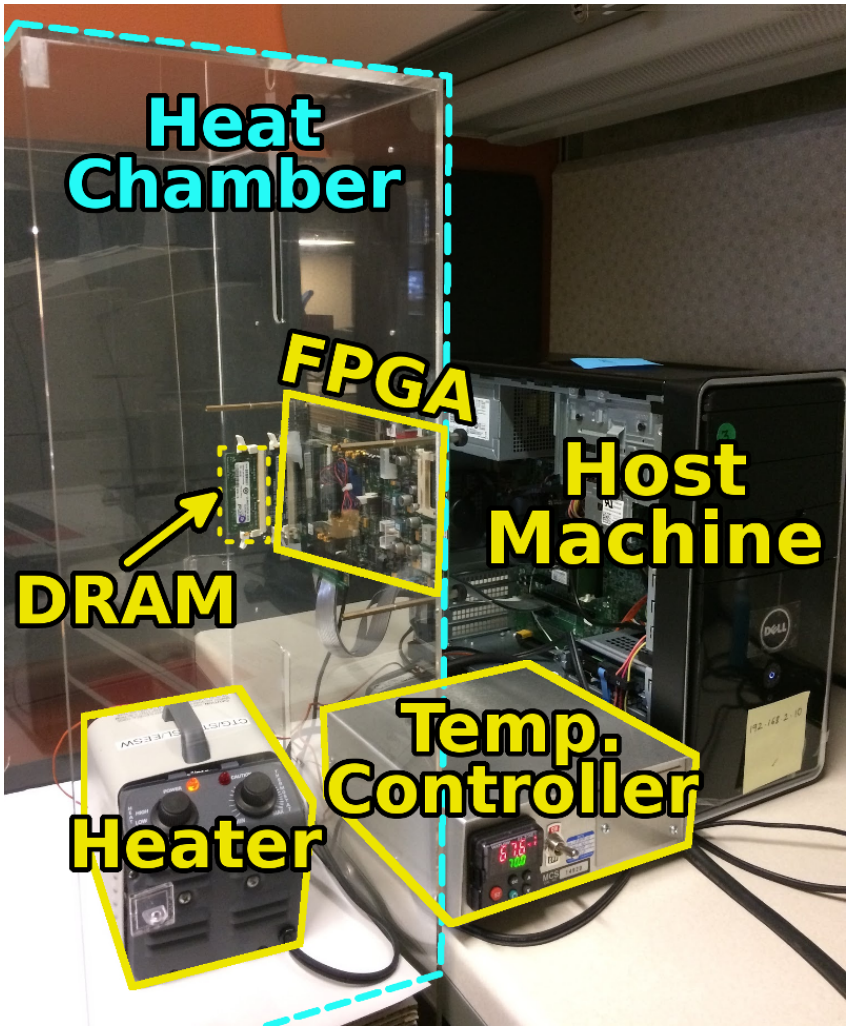
- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,
  "LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory"
  IEEE Computer Architecture Letters (***CAL***), June 2016.

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*

SoftMC: An FPGA-based Testbed 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

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 Vijaykumar[3] Samira Khan[4,3] Saugata Ghose[3] Kevin Chang[3]
Gennady Pekhimenko[5,3] Donghyuk Lee[6,3] Oguz Ergin[2] 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*

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

Ramulator: A Fast and Extensible DRAM Simulator

[IEEE Comp Arch Letters'15]

Ramulator Motivation

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

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

Table 1. Landscape of DRAM-based memory

Ramulator

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

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

Table 3. Comparison of five simulators using two traces

Case Study: Comparison of DRAM Standards

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

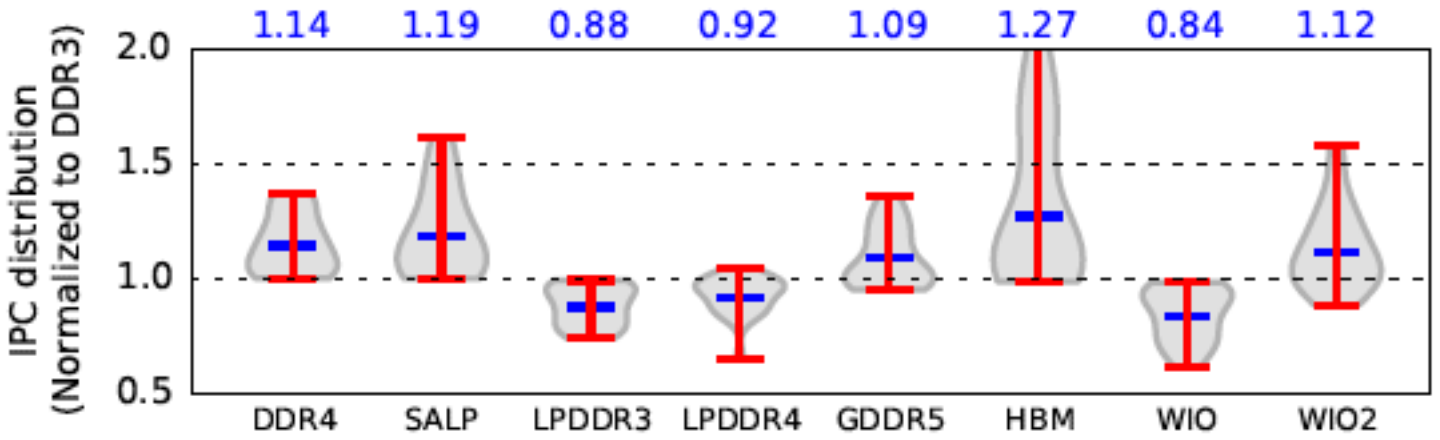


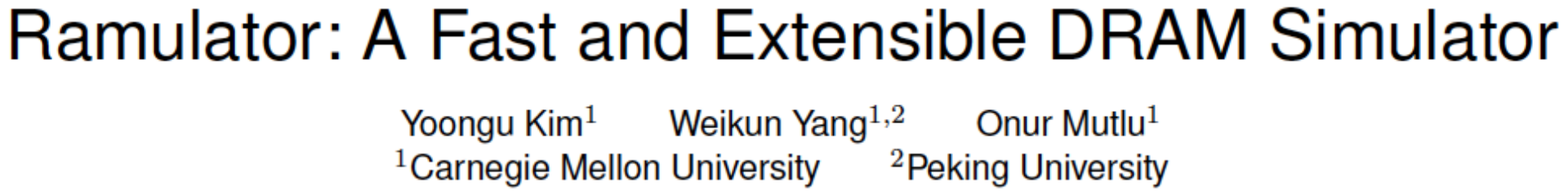
Figure 2. Performance comparison of DRAM standards

Across 22 workloads, simple CPU model

Ramulator Paper and Source Code

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

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



Challenge and Opportunity for Future

Fundamentally Energy-Efficient (Data-Centric) Computing Architectures

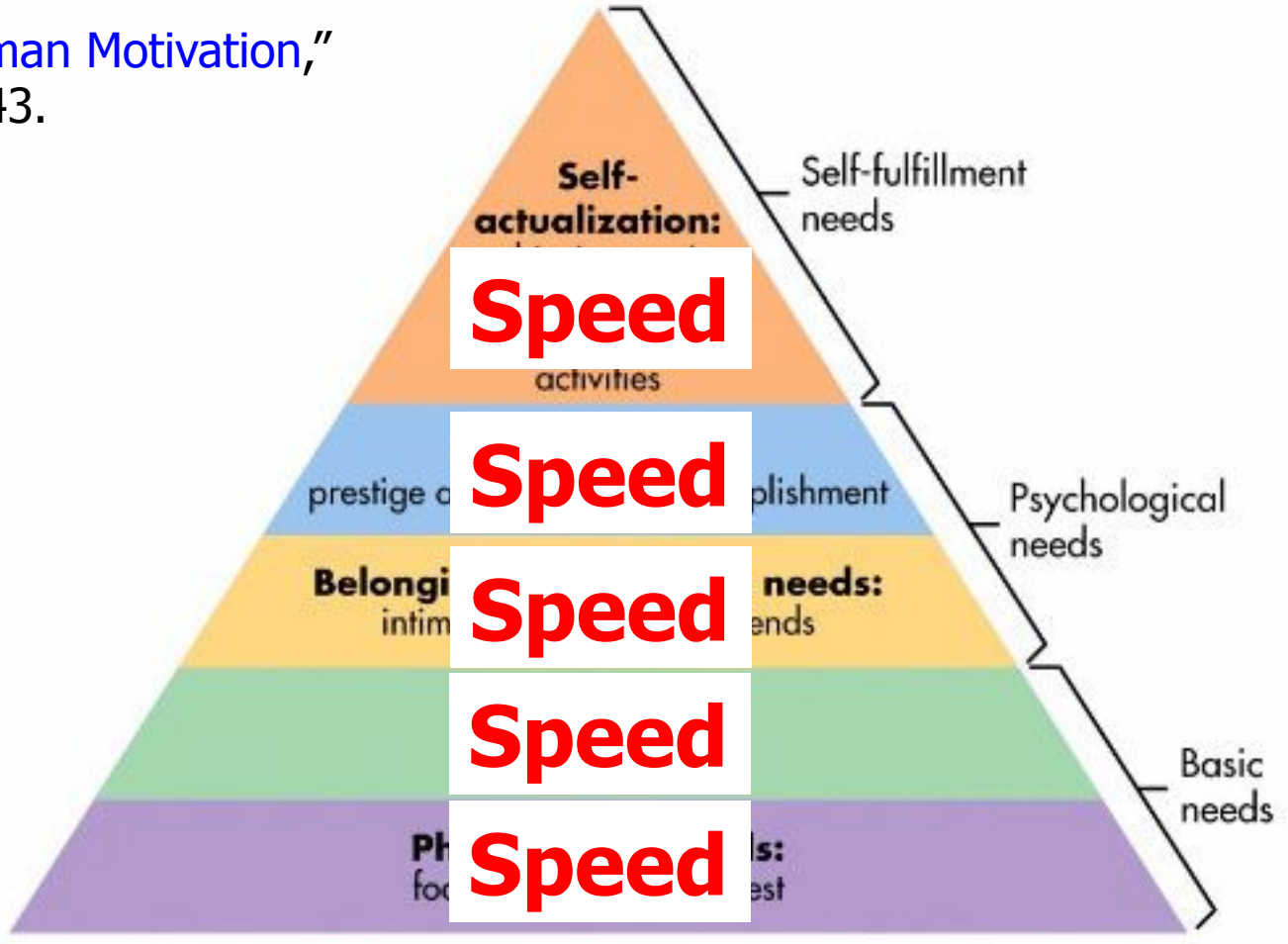
Three Key Issues in Future Platforms

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

Source: http://spectrum.ieee.org/image/MjYzMzAyMg.jpeg

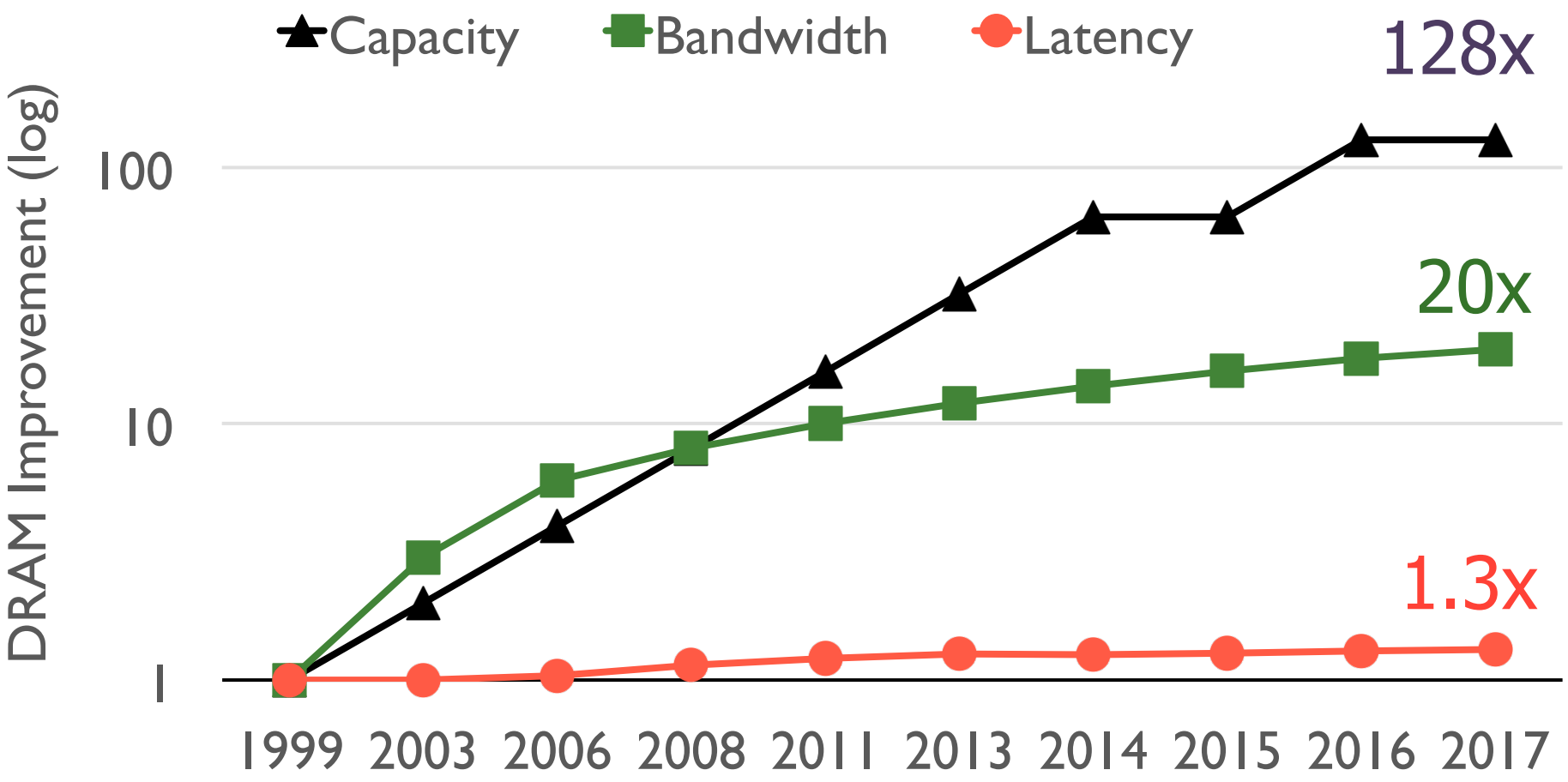
Maslow's Hierarchy of Needs, A Third Time

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



Source: https://www.simplypsychology.org/maslow.html

Main Memory Latency Lags Behind



Memory latency remains almost constant

A Closer Look …

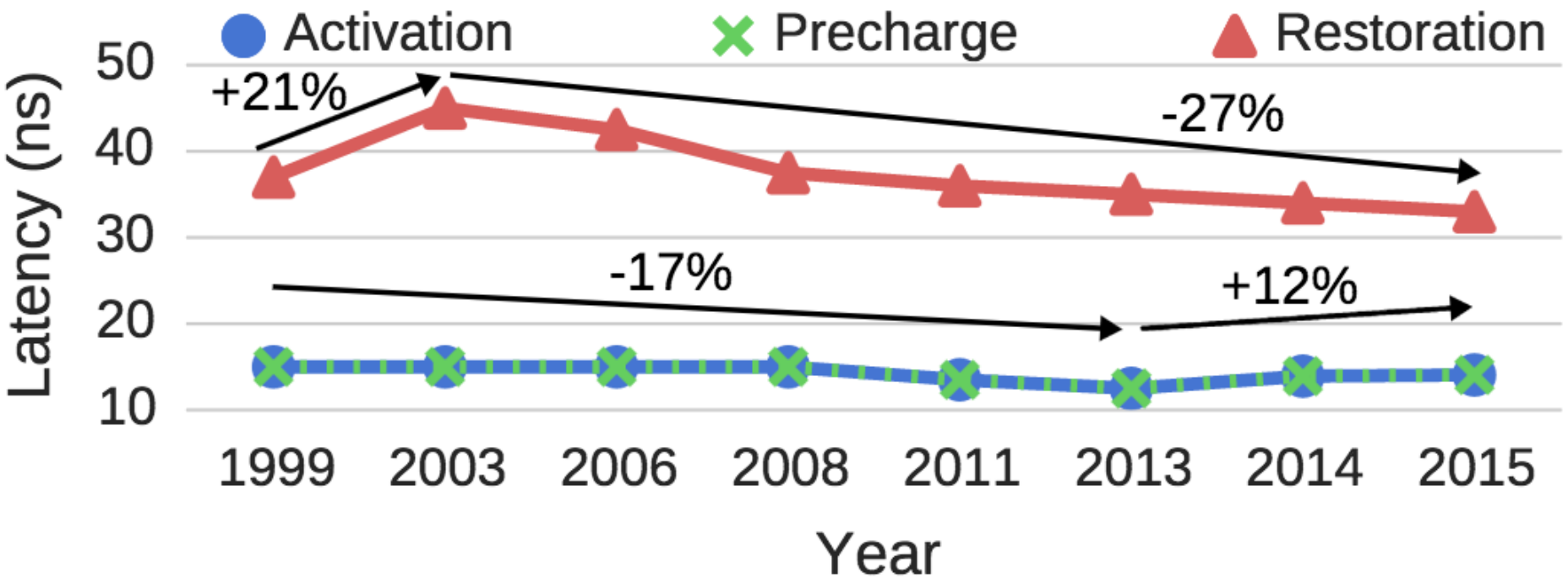
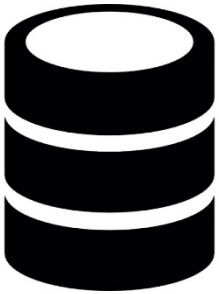


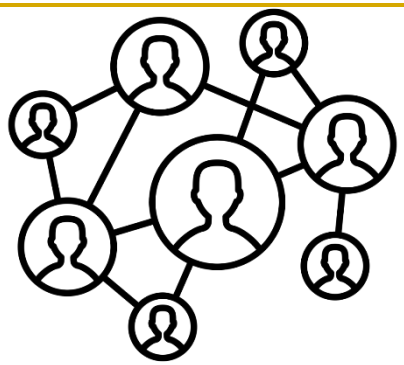
Figure 1: DRAM latency trends over time [20, 21, 23, 51].

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

DRAM Latency Is Critical for Performance

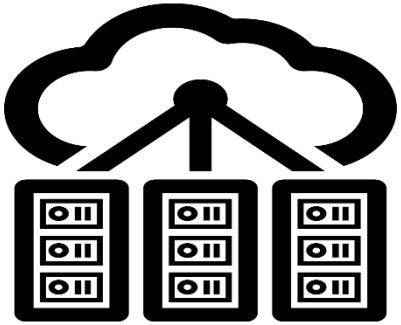


In-memory Databases
[Mao+, EuroSys'12;
Clapp+ (**Intel**), IISWC'15]



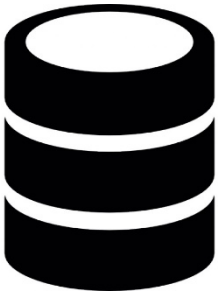
Graph/Tree Processing
[Xu+, IISWC'12; Umuroglu+, FPL'15]

In-Memory Data Analytics
[Clapp+ (**Intel**), IISWC'15;
Awan+, BDCloud'15]

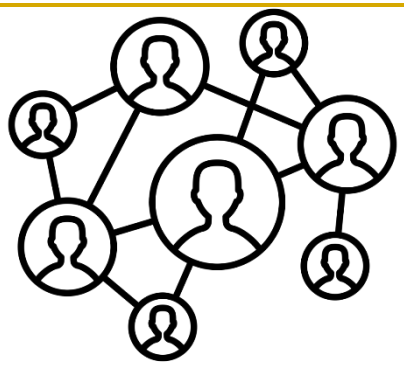


Datacenter Workloads
[Kanev+ (**Google**), ISCA'15]

DRAM Latency Is Critical for Performance



In-memory Databases

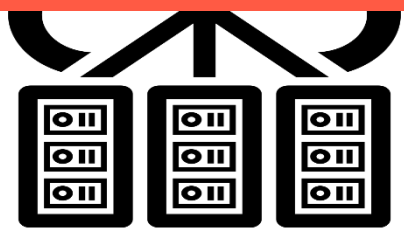


Graph/Tree Processing

Long memory latency → performance bottleneck

In-Memory Data Analytics

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



Datacenter Workloads

[Kanev+ (**Google**), ISCA'15]

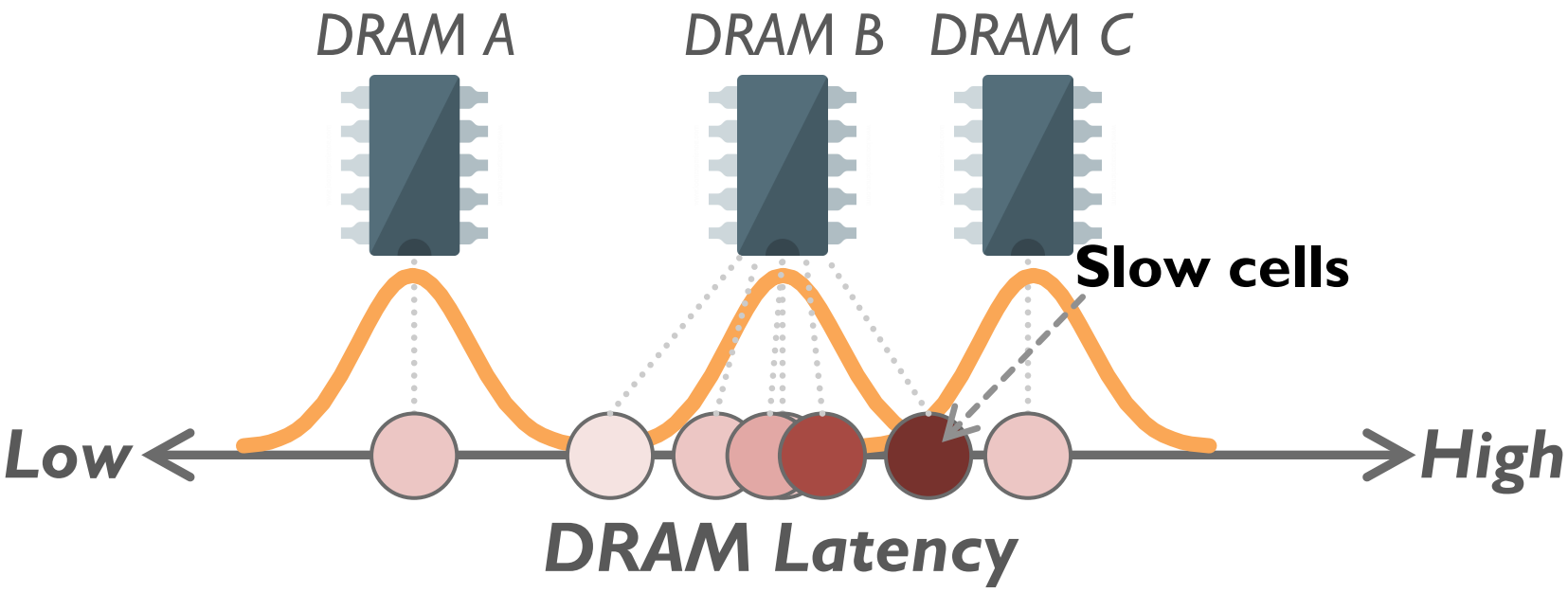
Why the Long Latency?

- 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



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]
  - …
- 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

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

Latency Reduction Summary of 115 DIMMs

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

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

AL-DRAM: Real System Evaluation

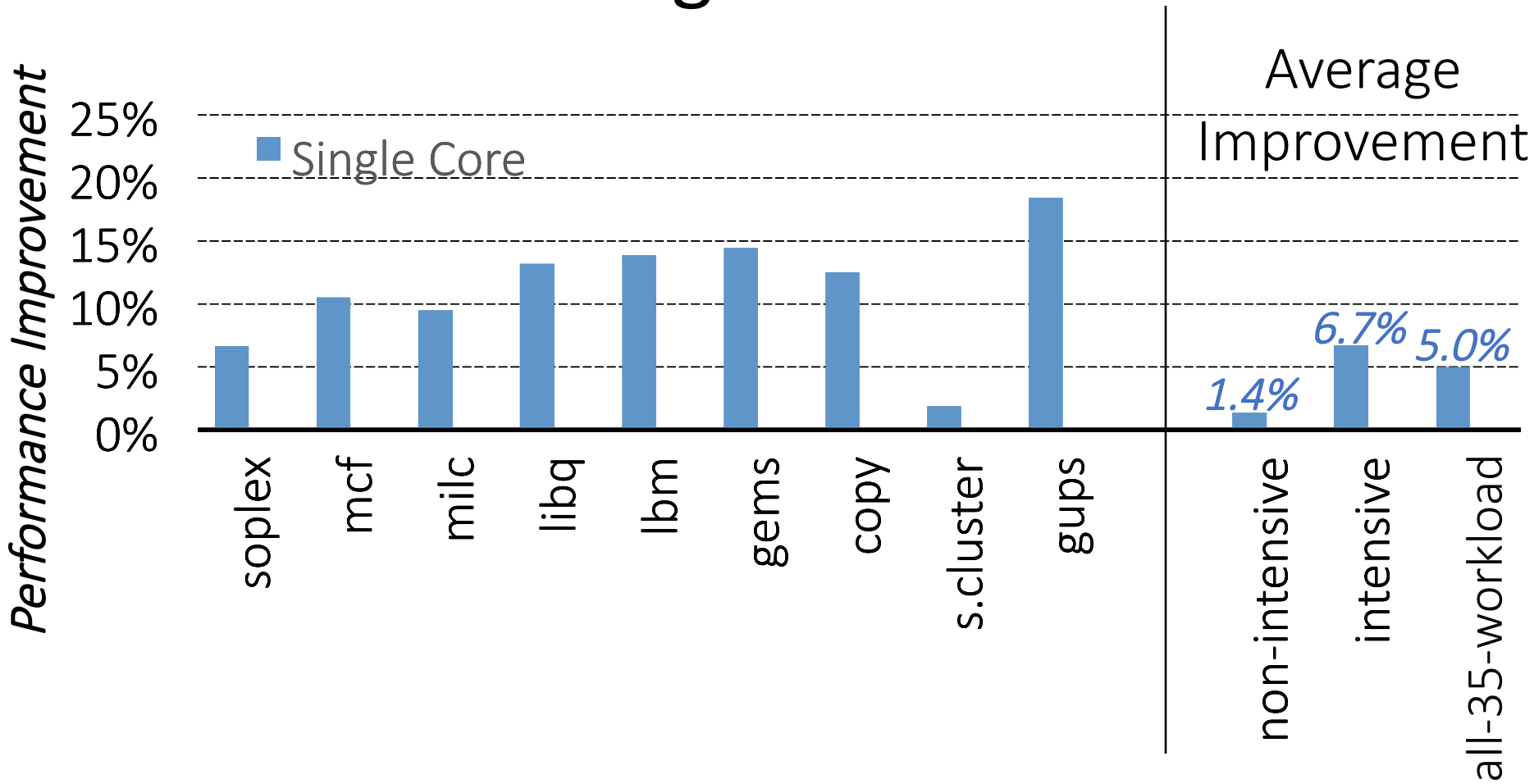
- *System*
  - *CPU: AMD 4386 (8 Cores, 3.1GHz, 8MB LLC)*

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

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

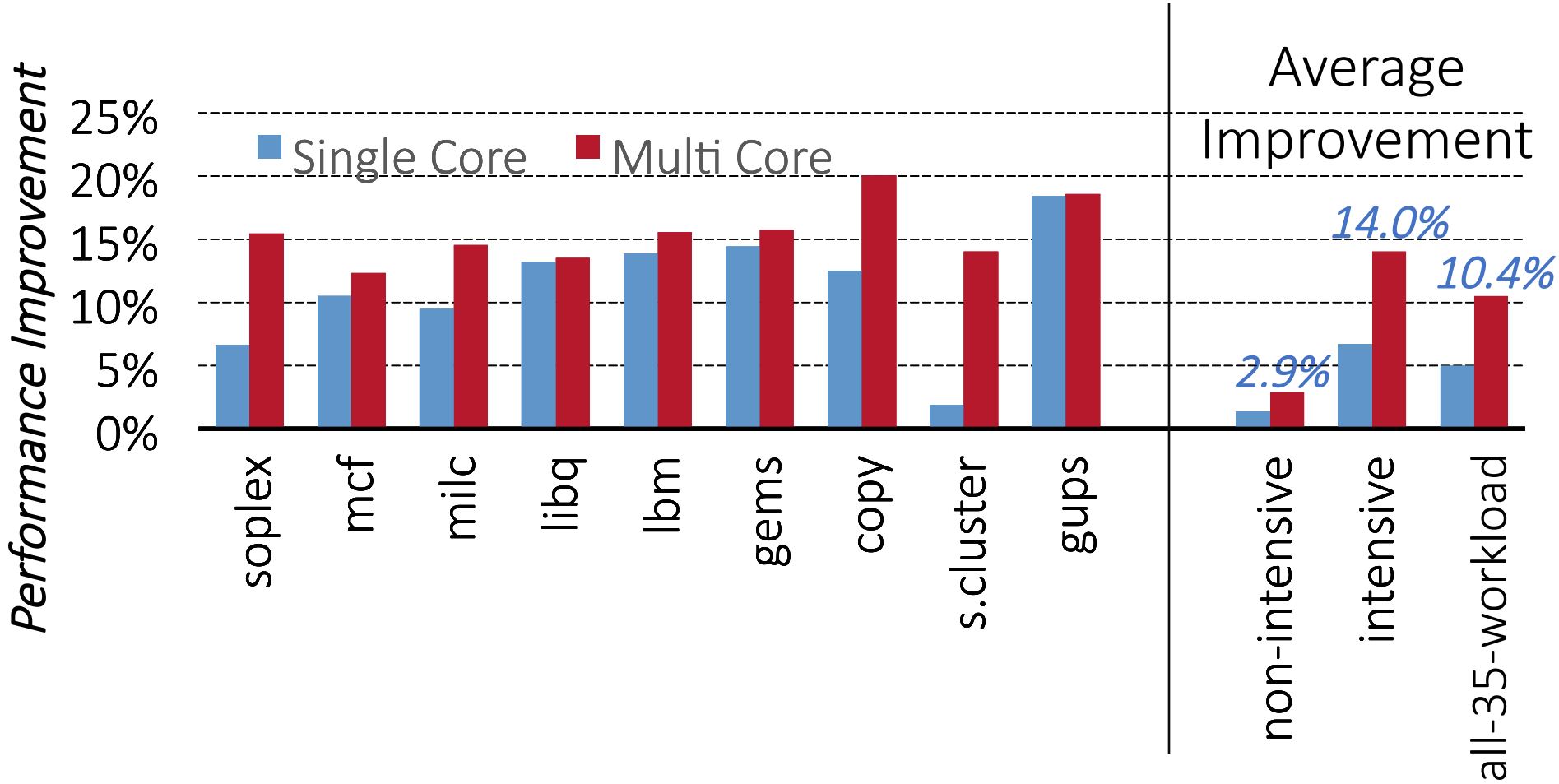
Bits	Description
31:30	Reserved.
29:24	**Tras: row active strobe**. Read-write. BIOS: See 2.9.7.5 [SPD ROM-Based Configuration]. Specifies the minimum time in memory clock cycles from an activate command to a precharge command, both to the same chip select bank. Bits — Description 07h-00h — Reserved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

AL-DRAM: Single-Core Evaluation



AL-DRAM ***improves single-core performance*** *on a real system*

AL-DRAM: Multi-Core Evaluation

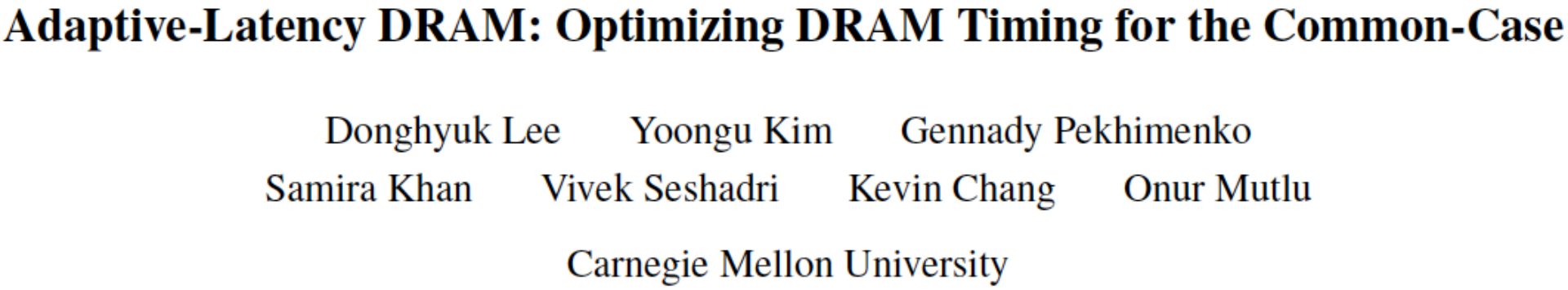


AL-DRAM provides higher performance on

multi-programmed & multi-threaded workloads

More on AL-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]

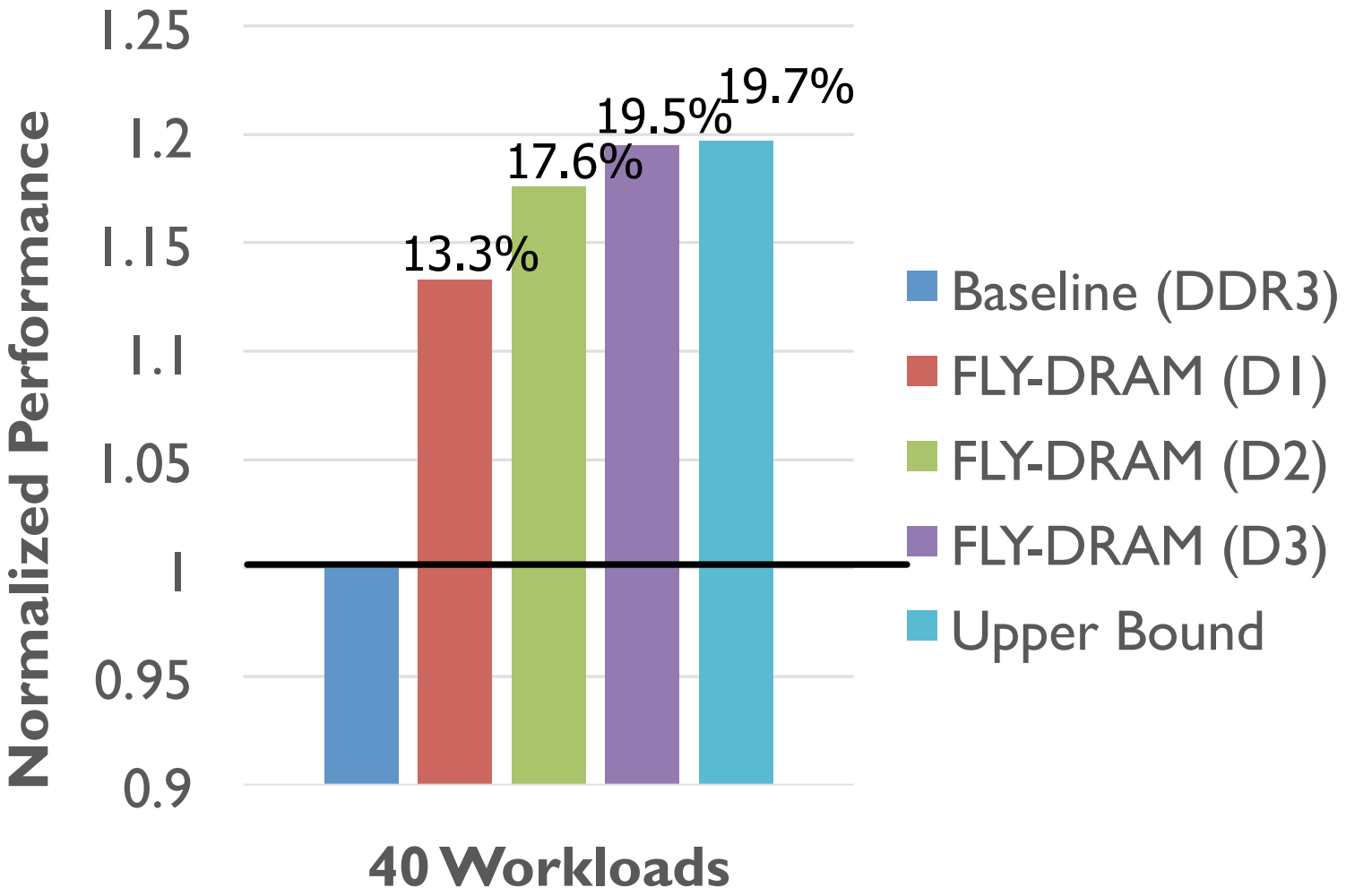


Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case

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 (I)

- 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. Chang[1] Abhijith Kashyap[1] Hasan Hassan[1,2]
Saugata Ghose[1] Kevin Hsieh[1] Donghyuk Lee[1] Tianshi Li[1,3]
Gennady Pekhimenko[1] Samira Khan[4] Onur Mutlu[5,1]

[1]Carnegie Mellon University [2]TOBB ETÜ [3]Peking University [4]University of Virginia [5]ETH Zürich

Analysis of Latency Variation in DRAM Chips (II)

- 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

Analysis of Latency-Voltage in DRAM Chips (I)

- 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?

Challenge and Opportunity for Future

Fundamentally

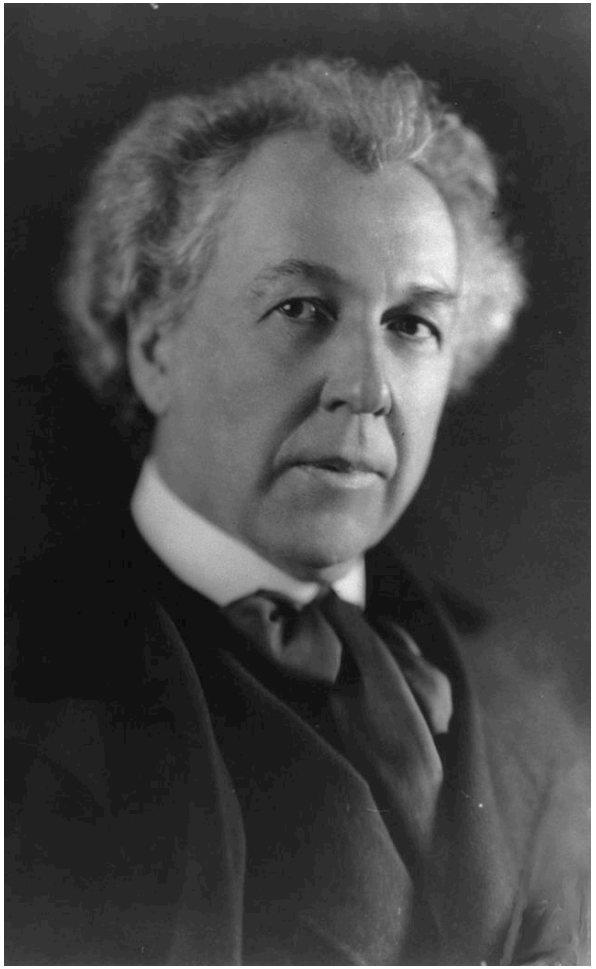
Low Latency

Computing Architectures

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"

Another Example: Precedent-Based Design

Source: http://cookiemagik.deviantart.com/art/Train-station-207266944

Principled Design

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, https://commons.wikimedia.org/w/index.php?curid=[illegible], https://en.wikipedia.org/wiki/Santiago_Calatrava

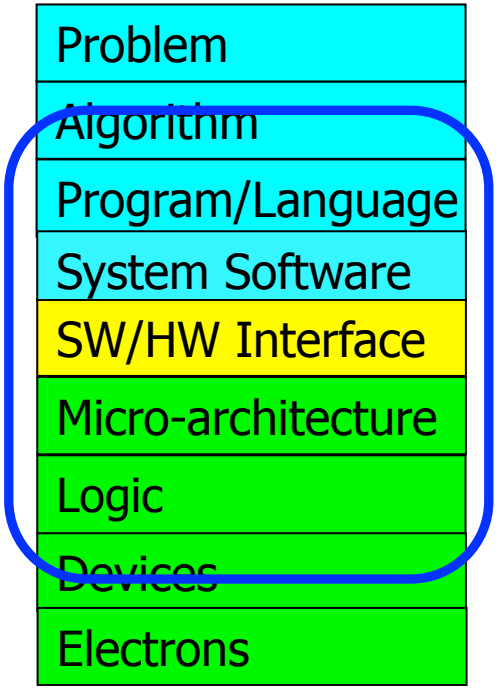
Source: https://www.dezeen.com/2016/08/29/santiago-calatrava-oculus-world-trade-center-transportation-hub-new-york-photographs-hufton-crow/

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**
  - **…**

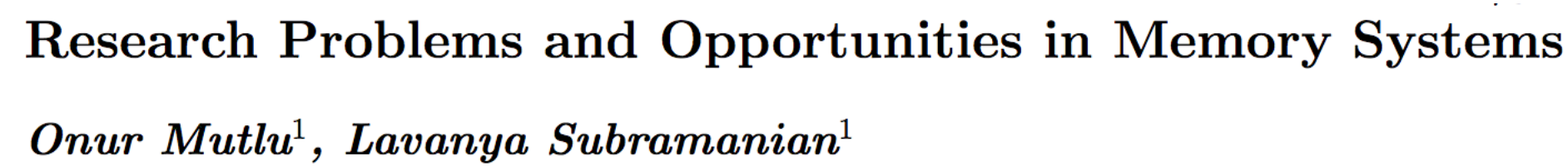
The Future is Very Bright

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



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.

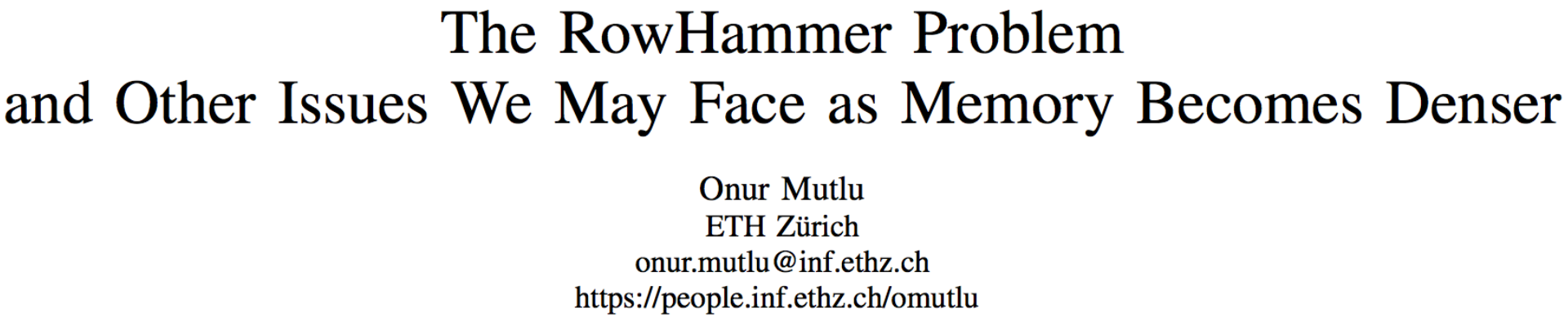


Research Problems and Opportunities in Memory Systems

Onur Mutlu[1], *Lavanya Subramanian*[1]

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)]

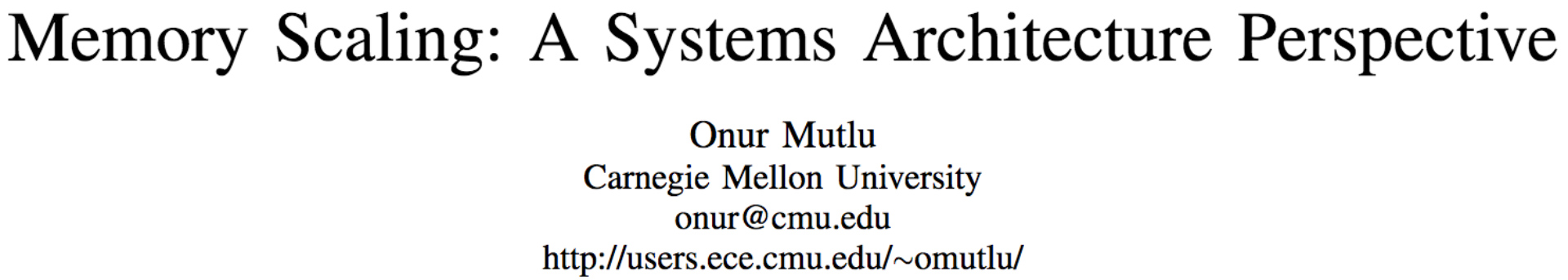

The RowHammer Problem
and Other Issues We May Face as Memory Becomes Denser

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

https://people.inf.ethz.ch/omutlu/pub/rowhammer-and-other-memory-issues_date17.pdf

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.

Memory Reliability, Security & Beyond

Three Key Issues in Modern Systems

Onur Mutlu
onur.mutlu@inf.ethz.ch
https://people.inf.ethz.ch/omutlu

June 18, 2017

Design Automation Summer School @ DAC 2017

SAFARI

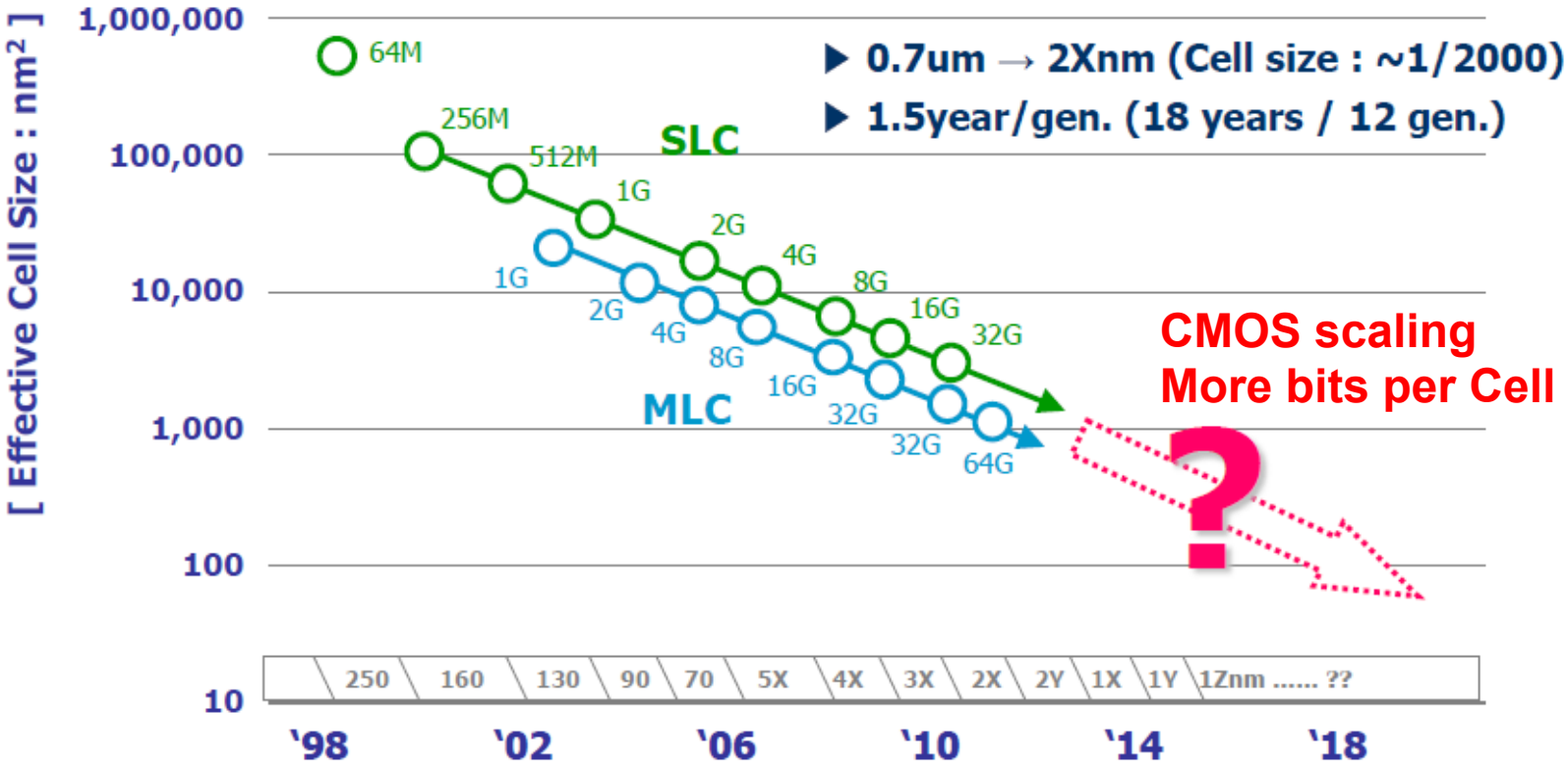
ETH zürich

NAND Flash Memory Reliability and Security

Upcoming Overview Paper

- 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.

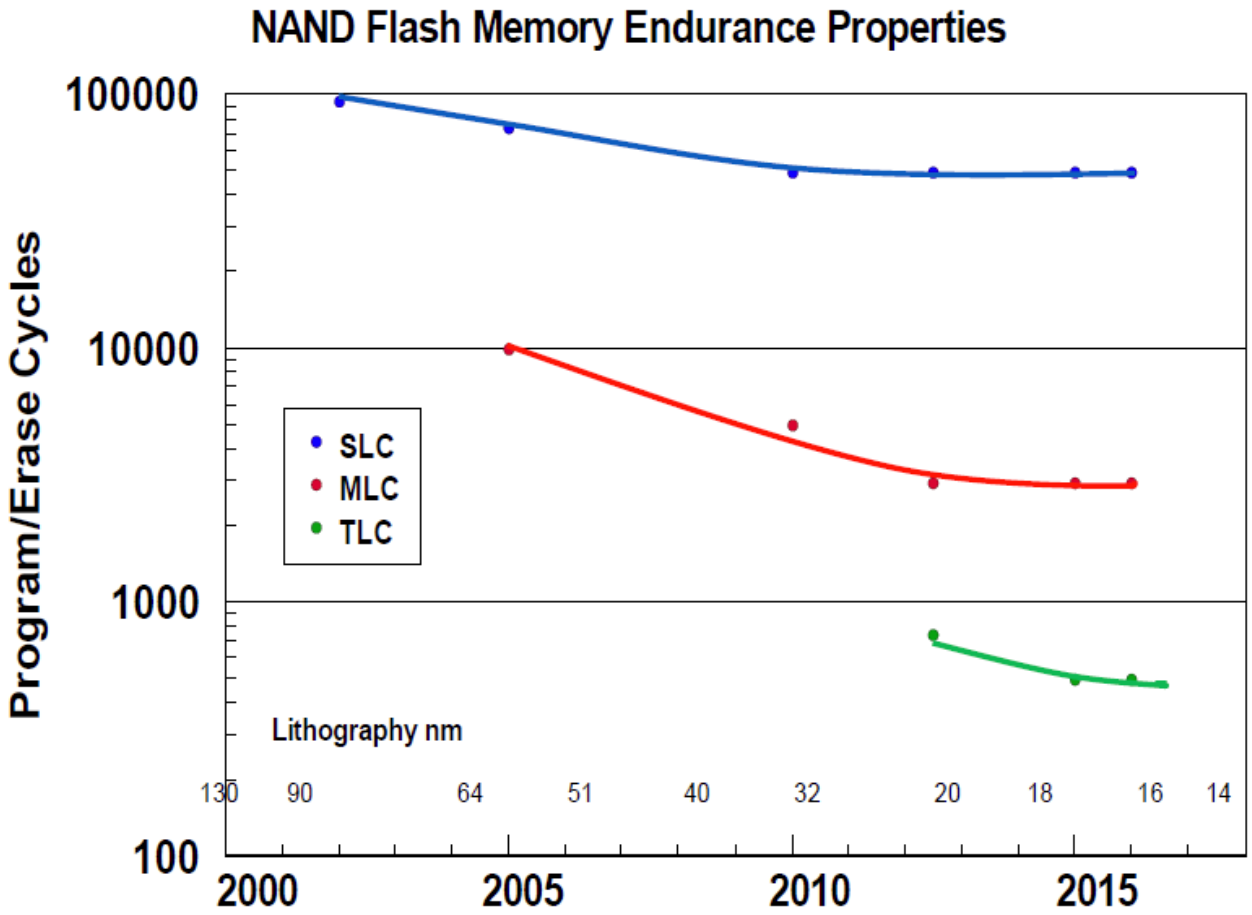
Evolution of NAND Flash Memory



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



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

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

A few thousand

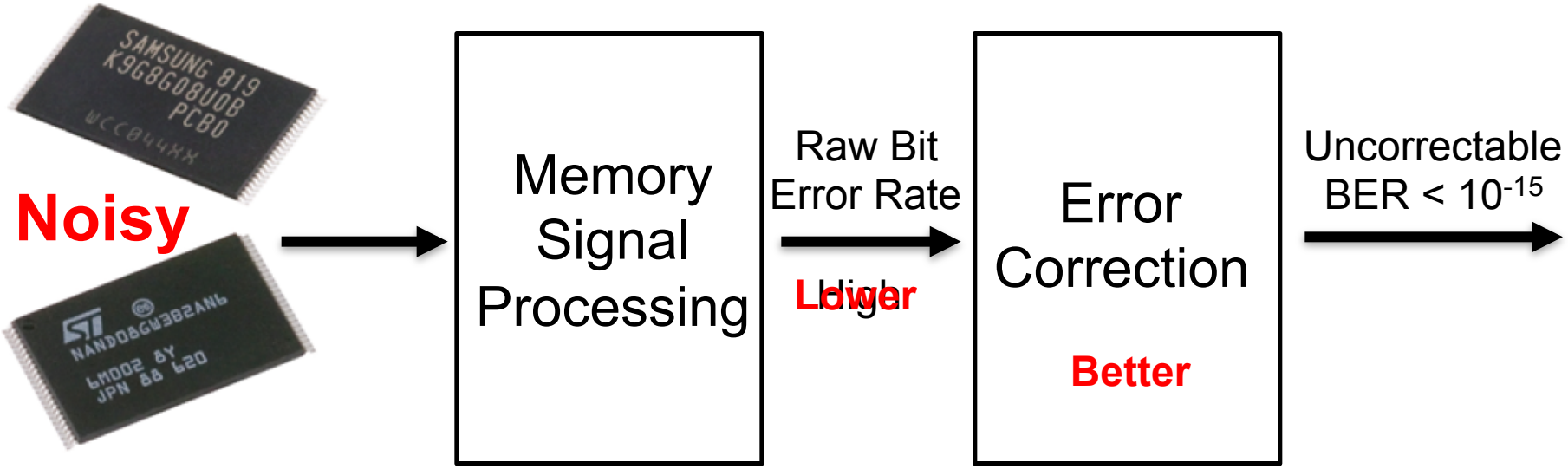
- **P/E cycles (required)**

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

> 50k P/E cycles

NAND Flash Memory is Increasingly Noisy

Future NAND Flash-based Storage Architecture



Our Goals:

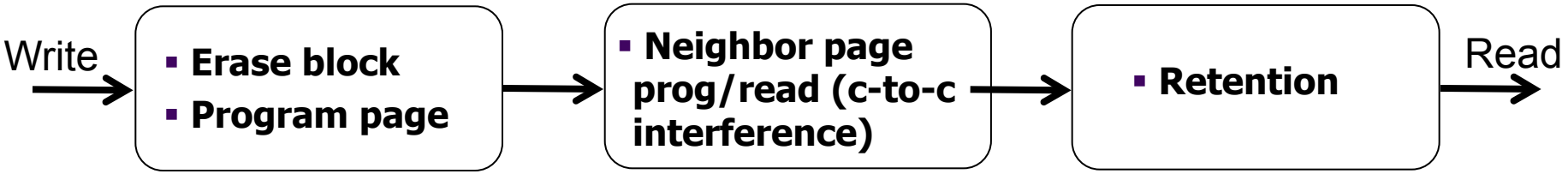
Build reliable error models for NAND flash memory

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**

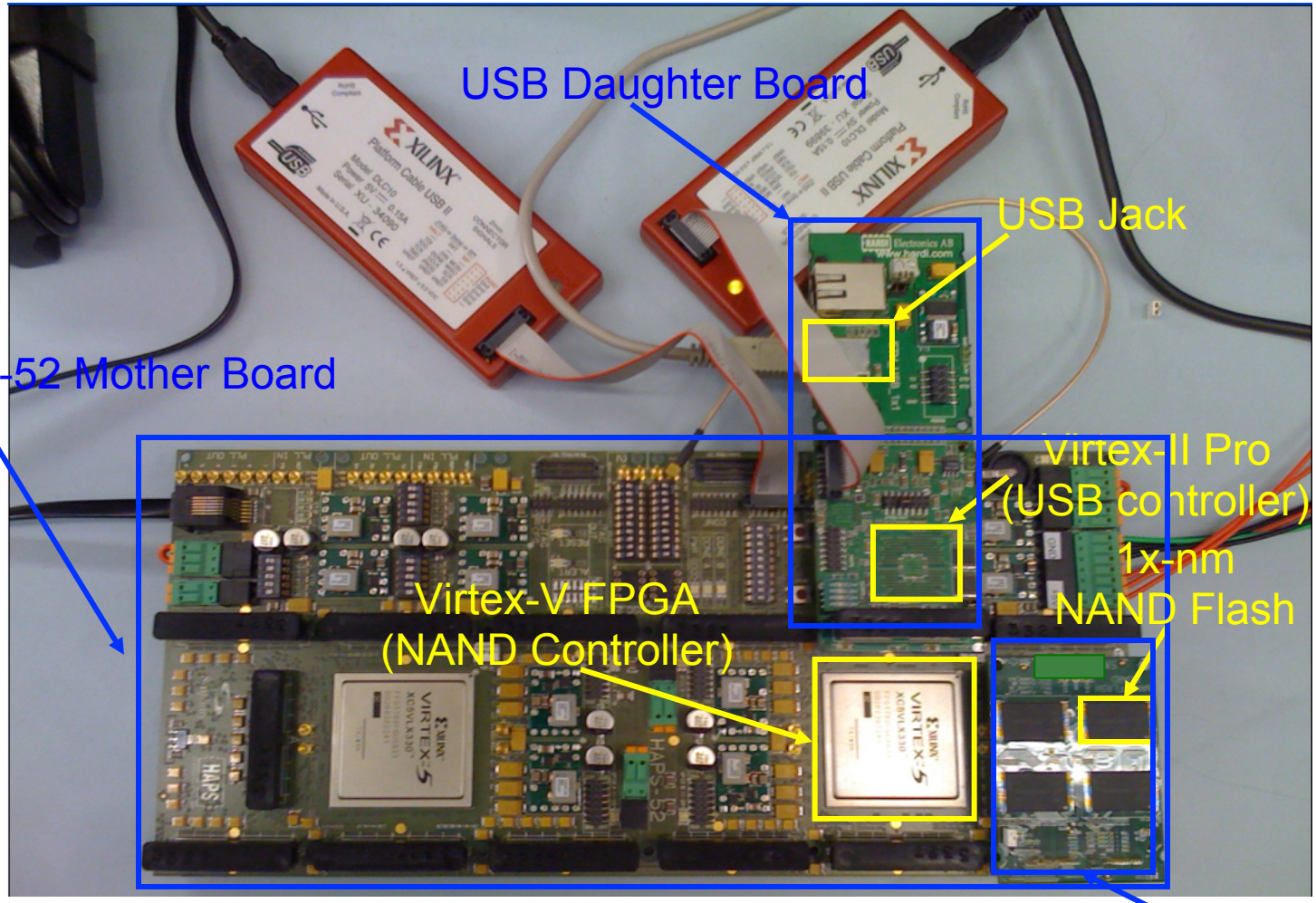
Cai et al., "Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery" **, HPCA 2015**

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]

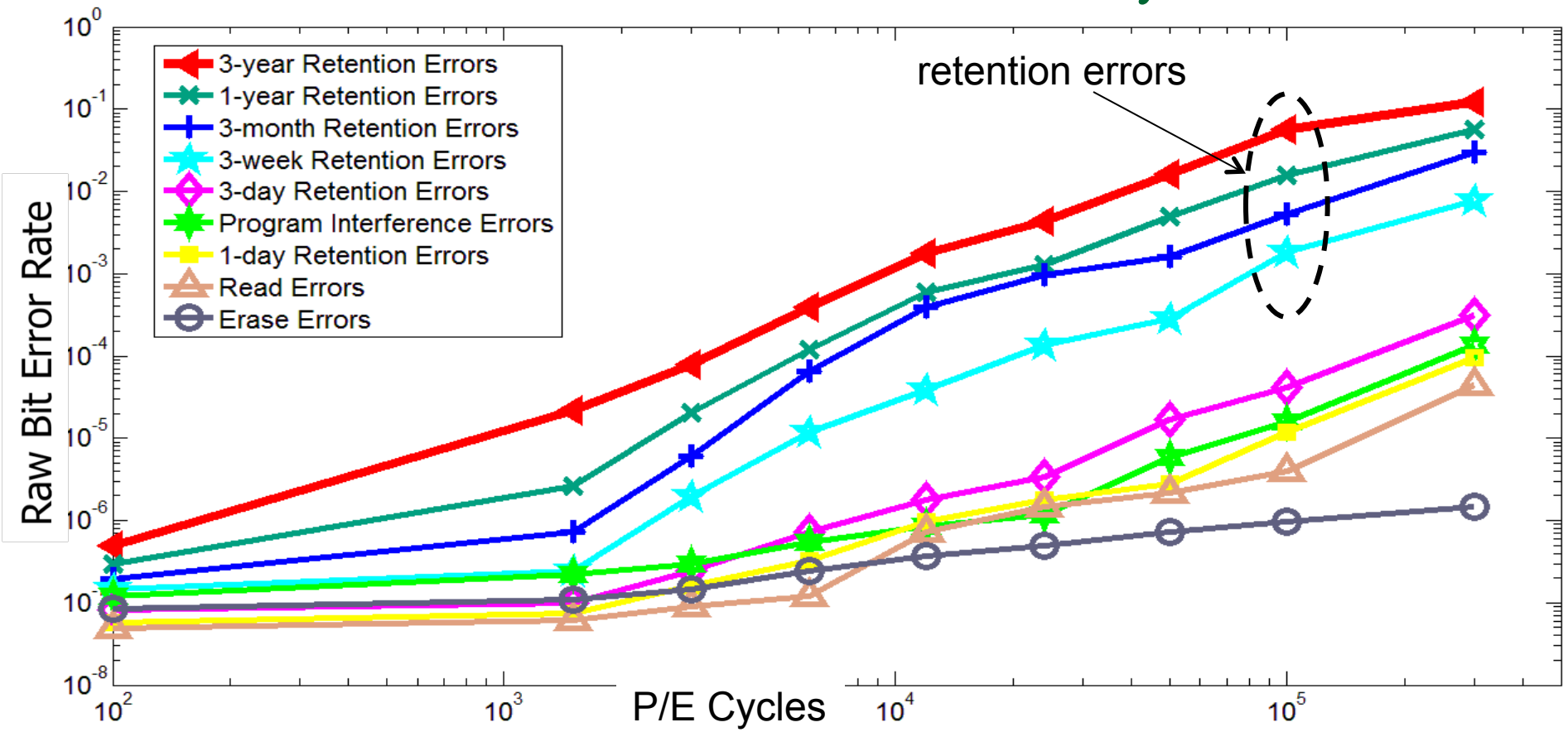
NAND Flash Error Types

- 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

Cai et al., Error Patterns in MLC NAND Flash Memory, DATE 2012.

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 Cai[1], Erich F. Haratsch[2], Onur Mutlu[1] and Ken Mai[1]
[1]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
[2]LSI Corporation, 1110 American Parkway NE, Allentown, PA
[1]{yucai, onur, kenmai}@andrew.cmu.edu, [2]erich.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

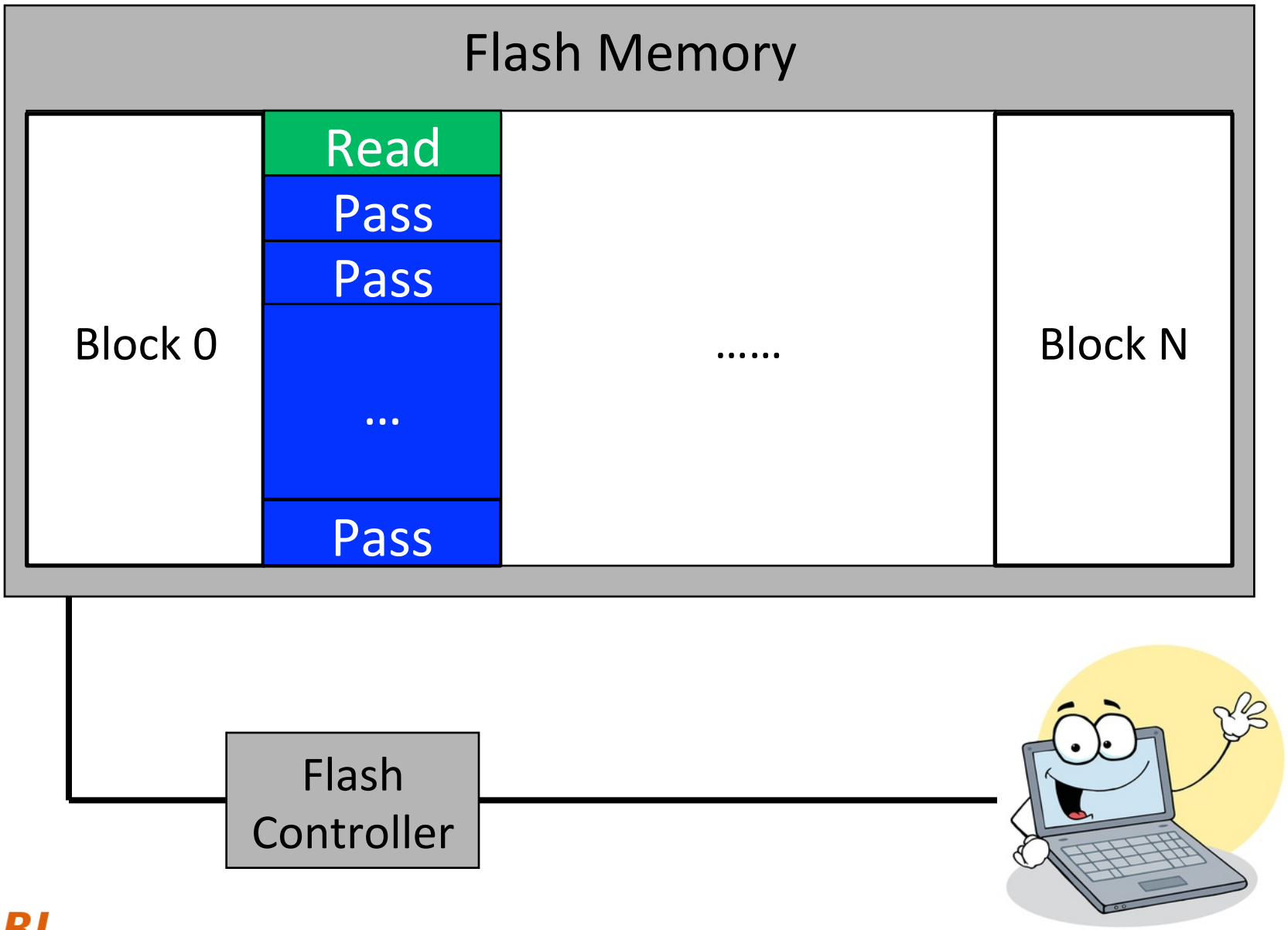
Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime

Yu Cai[1], Gulay Yalcin[2], Onur Mutlu[1], Erich F. Haratsch[3], Adrian Cristal[2], Osman S. Unsal[2] and Ken Mai[1]
[1]DSSC, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
[2]Barcelona Supercomputing Center, C/Jordi Girona 29, Barcelona, Spain
[3]LSI Corporation, 1110 American Parkway NE, Allentown, PA

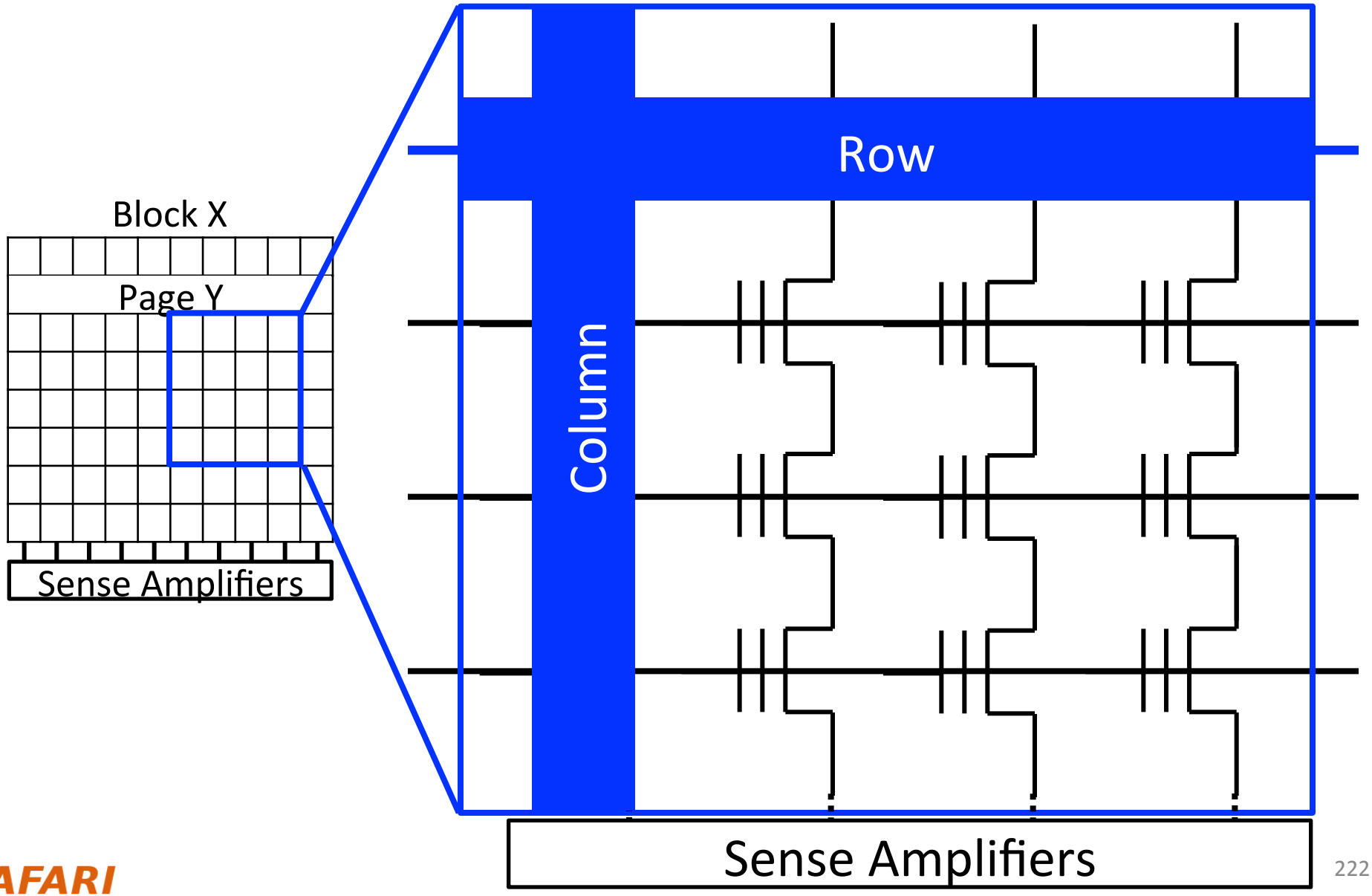
One Issue: Read Disturb in Flash Memory

- All scaled memories are prone to read disturb errors

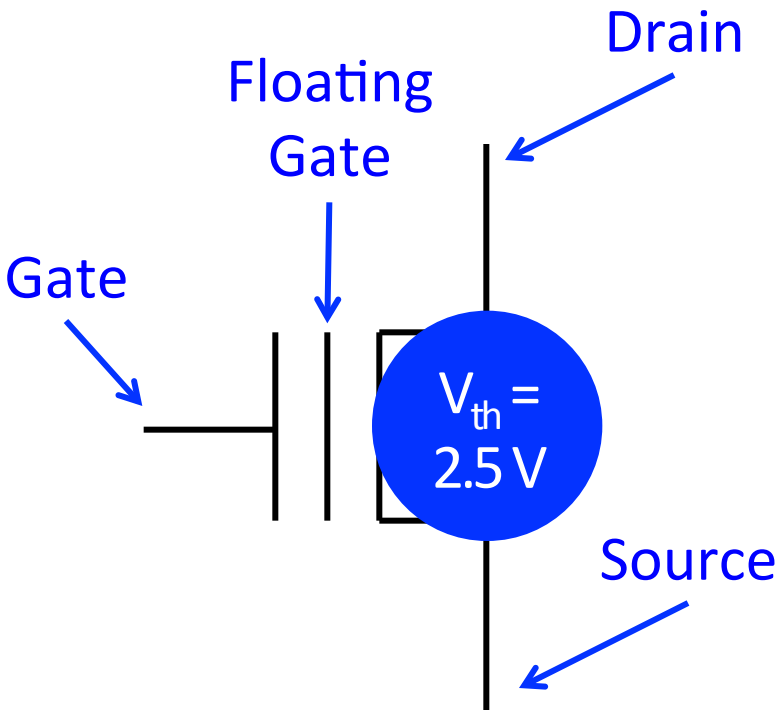
NAND Flash Memory Background



Flash Cell Array

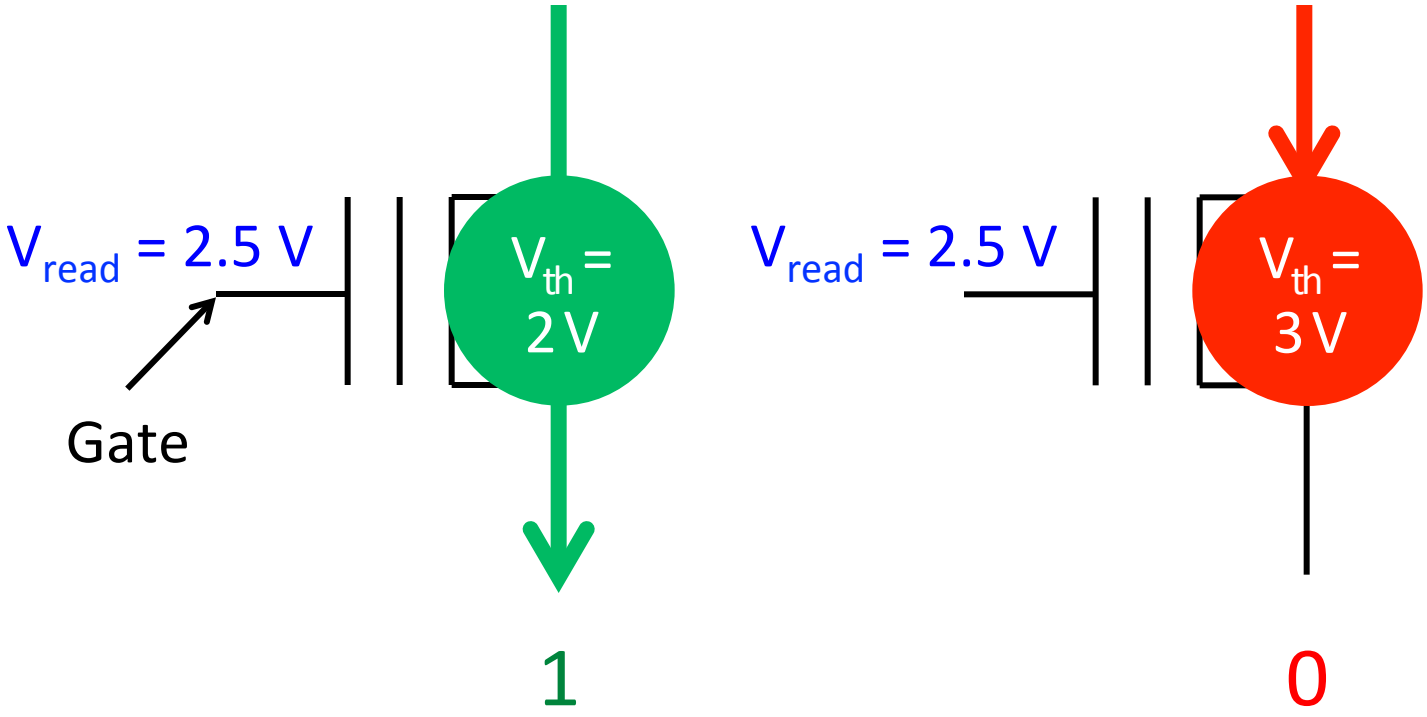


Flash Cell

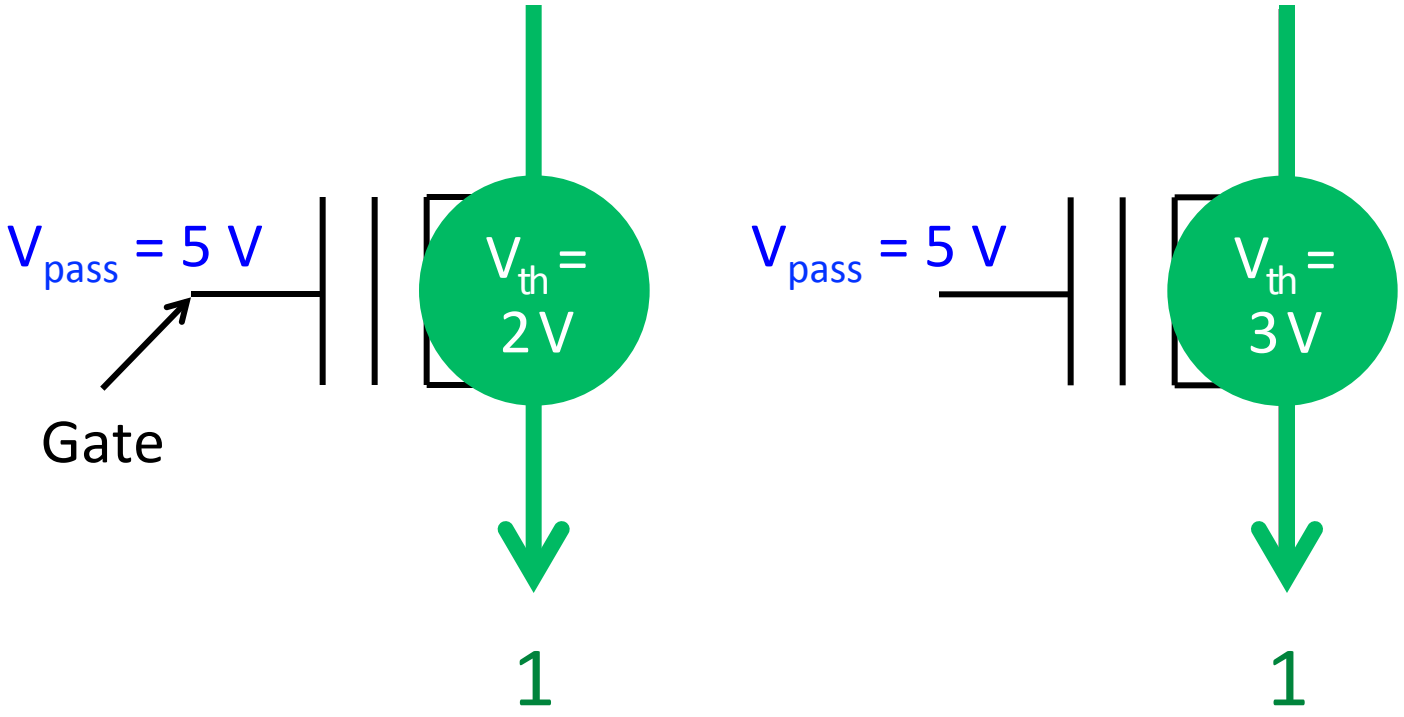


Floating Gate Transistor
(Flash Cell)

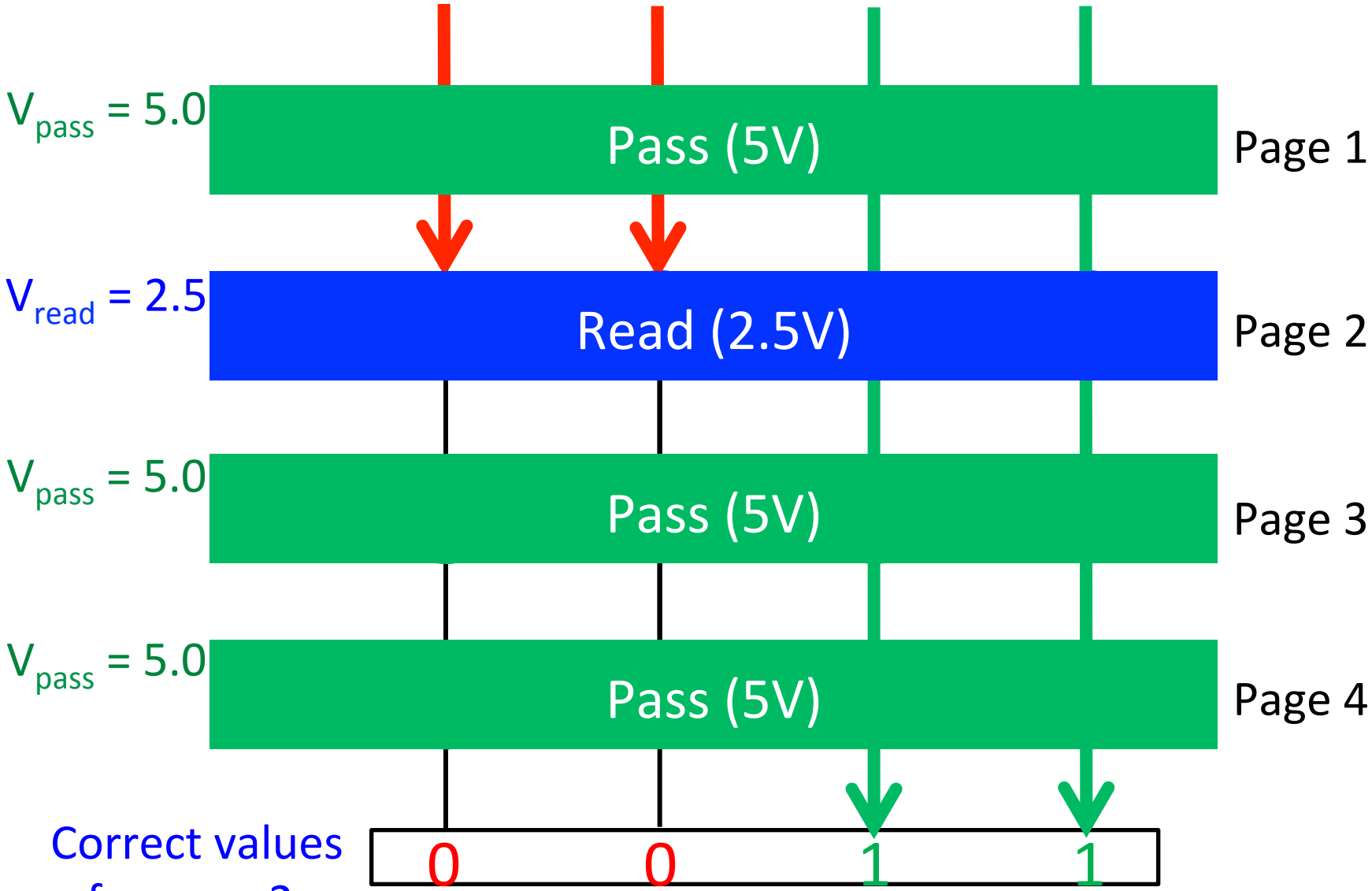
Flash Read



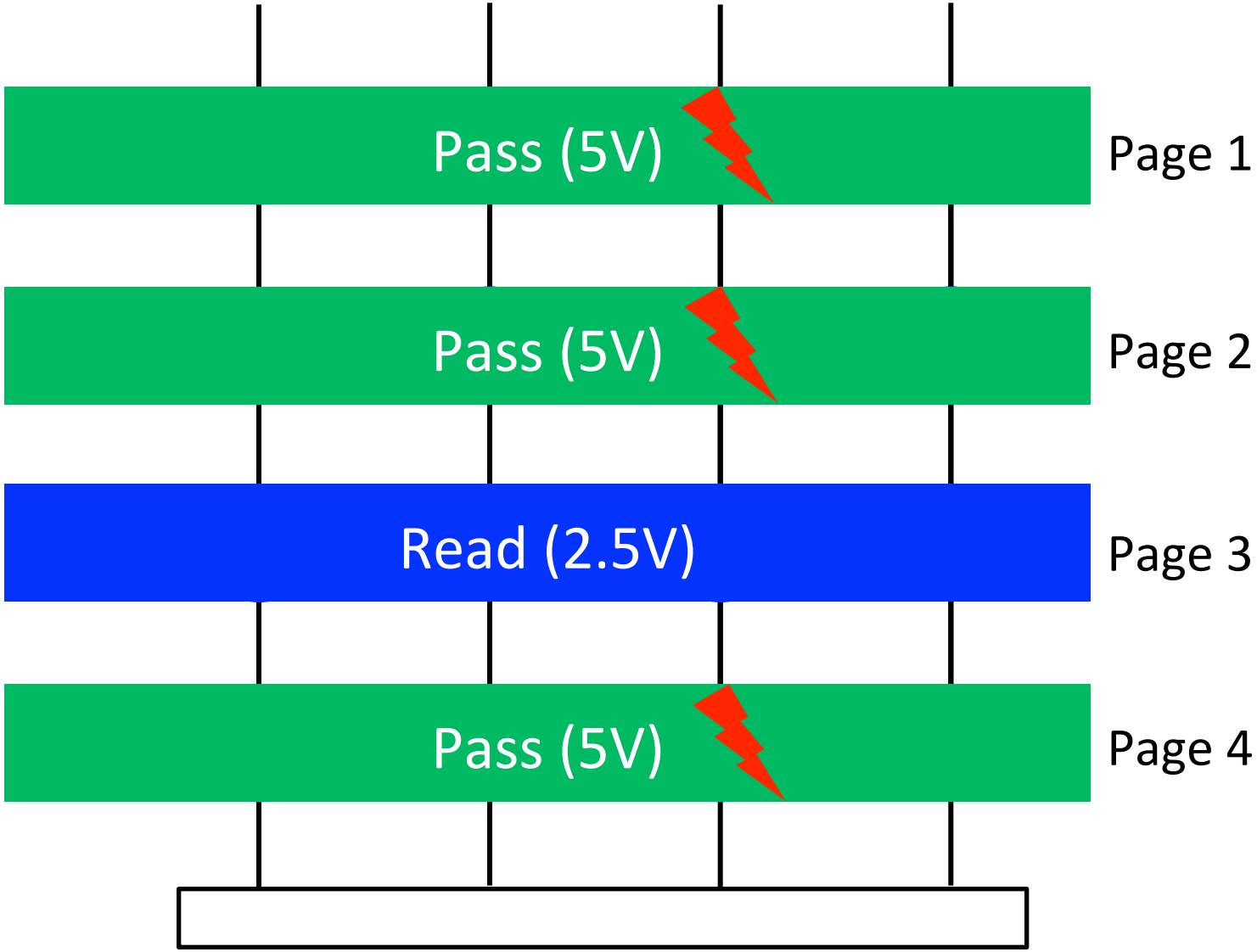
Flash Pass-Through



Read from Flash Cell Array

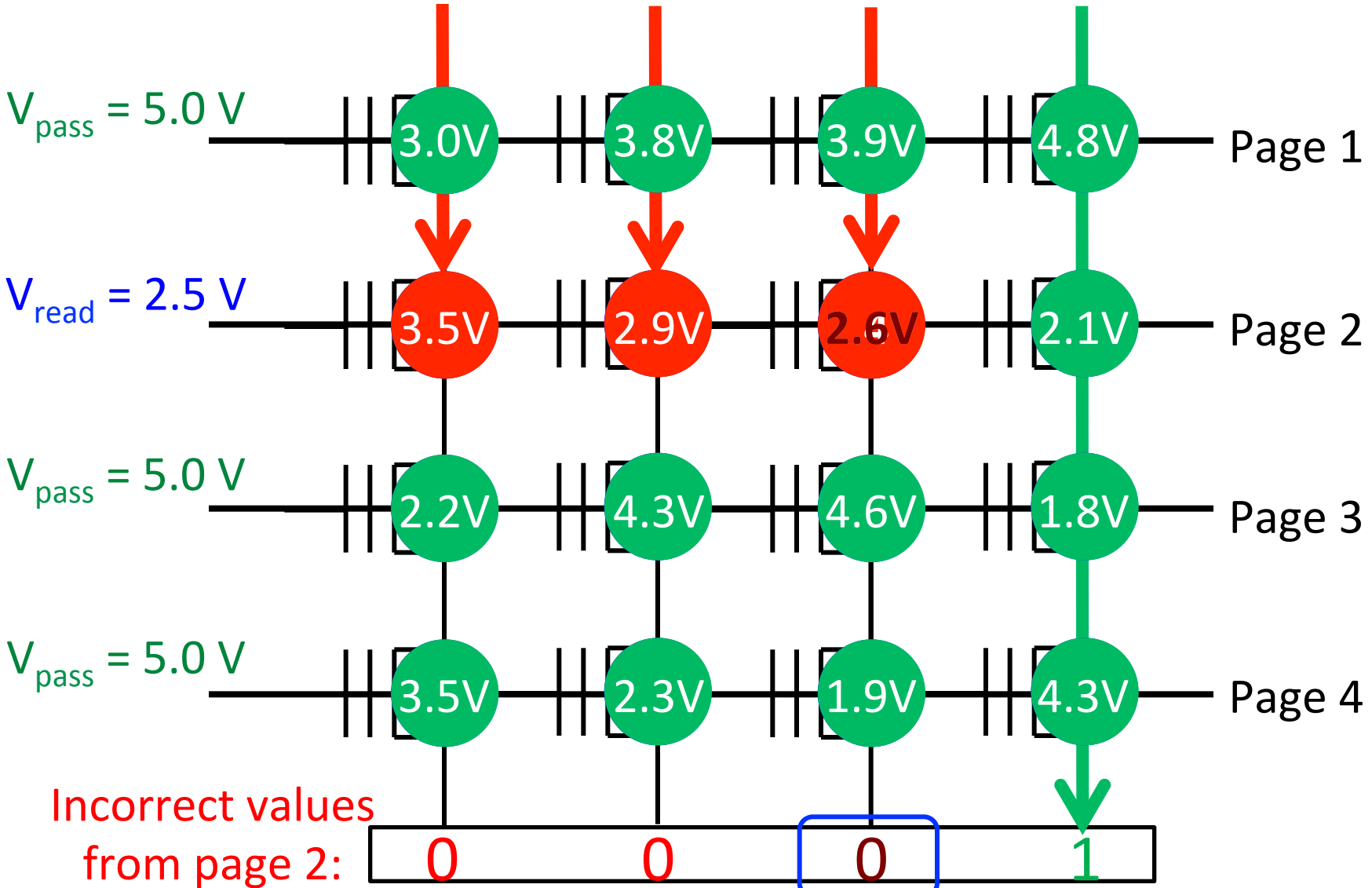


Read Disturb Problem: "Weak Programming" Effect



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

Read Disturb Problem: "Weak Programming" Effect



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[‡]

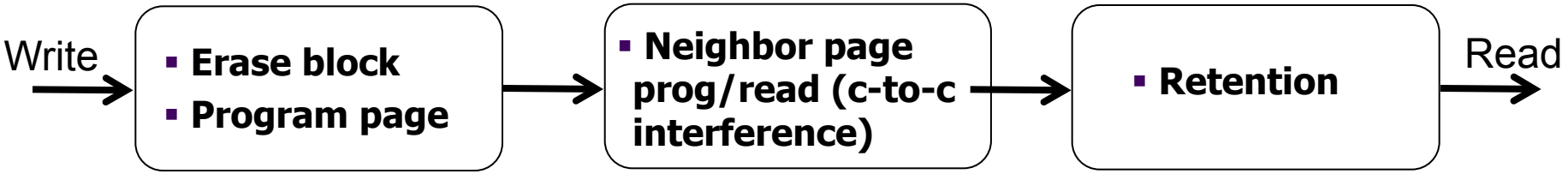
[†]*Carnegie Mellon University* [‡]*Seagate Technology* [§]*ETH Zürich*

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

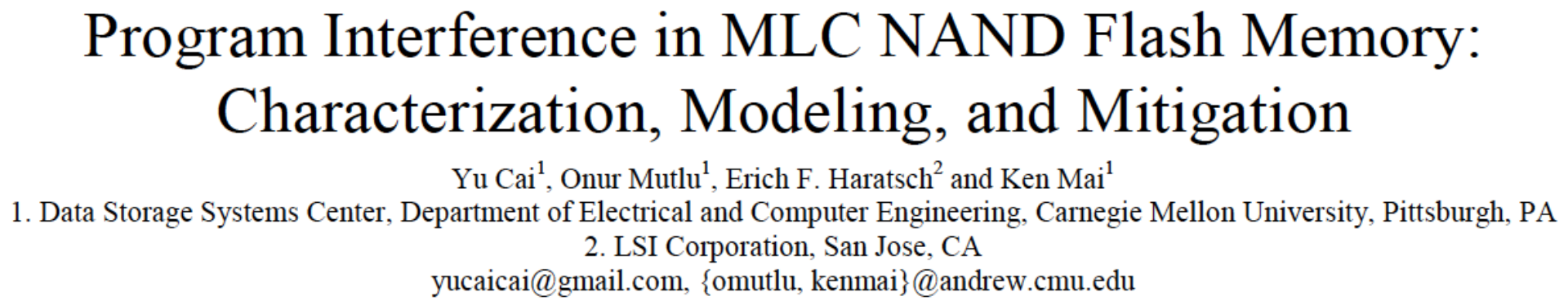
- Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai, **"Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis and Modeling"** *Proceedings of the Design, Automation, and Test in Europe Conference* (***DATE***), Grenoble, France, March 2013. Slides (ppt)

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

Yu Cai[1], Erich F. Haratsch[2], Onur Mutlu[1] and Ken Mai[1]
[1]DSSC, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
[2]LSI Corporation, 1110 American Parkway NE, Allentown, PA
[1]{yucai, onur, kenmai}@andrew.cmu.edu, [2]erich.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)

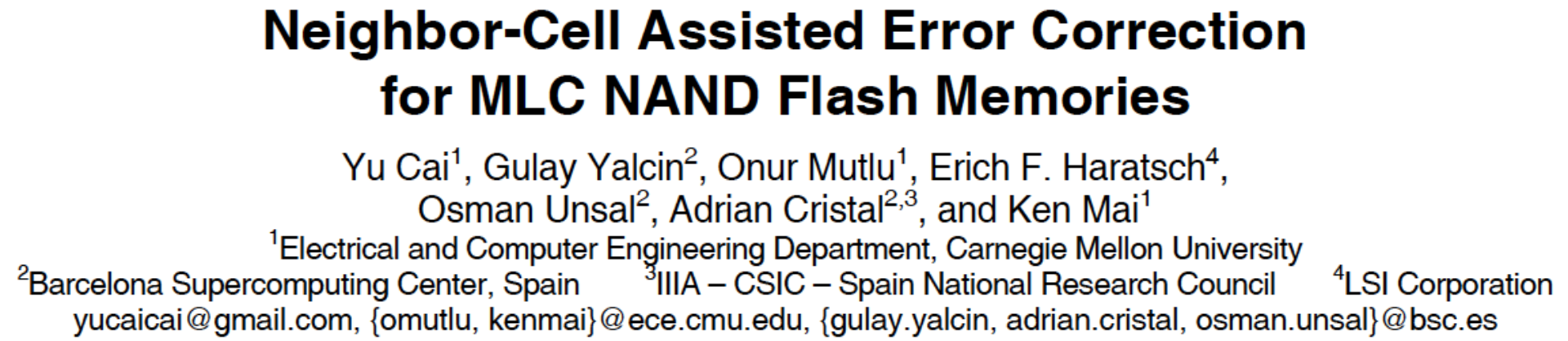


Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation

Yu Cai[1], Onur Mutlu[1], Erich F. Haratsch[2] and Ken Mai[1]
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)

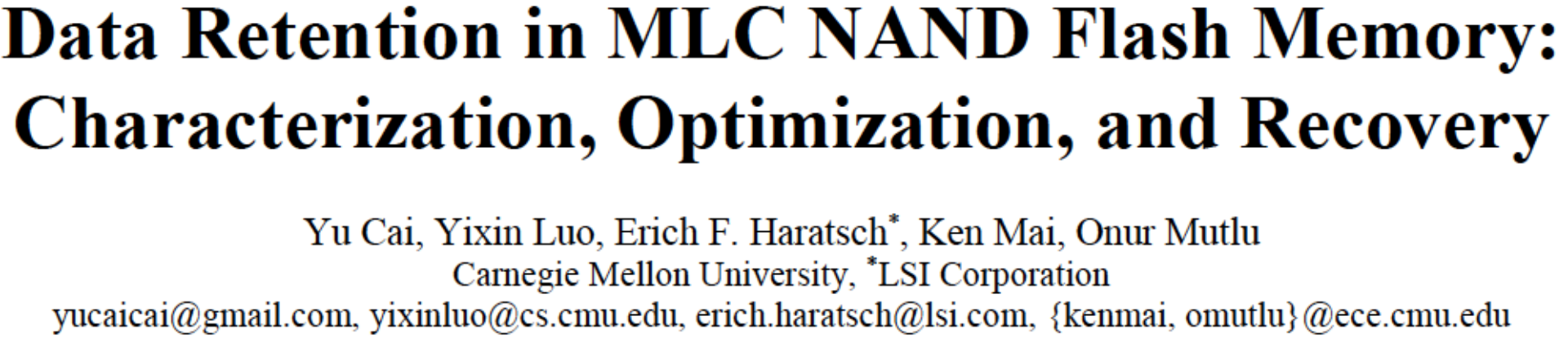


Neighbor-Cell Assisted Error Correction for MLC NAND Flash Memories

Yu Cai[1], Gulay Yalcin[2], Onur Mutlu[1], Erich F. Haratsch[4], Osman Unsal[2], Adrian Cristal[2,3], and Ken Mai[1]
[1]Electrical and Computer Engineering Department, Carnegie Mellon University
[2]Barcelona Supercomputing Center, Spain [3]IIIA – CSIC – Spain National Research Council [4]LSI 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, omutlu}@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)]

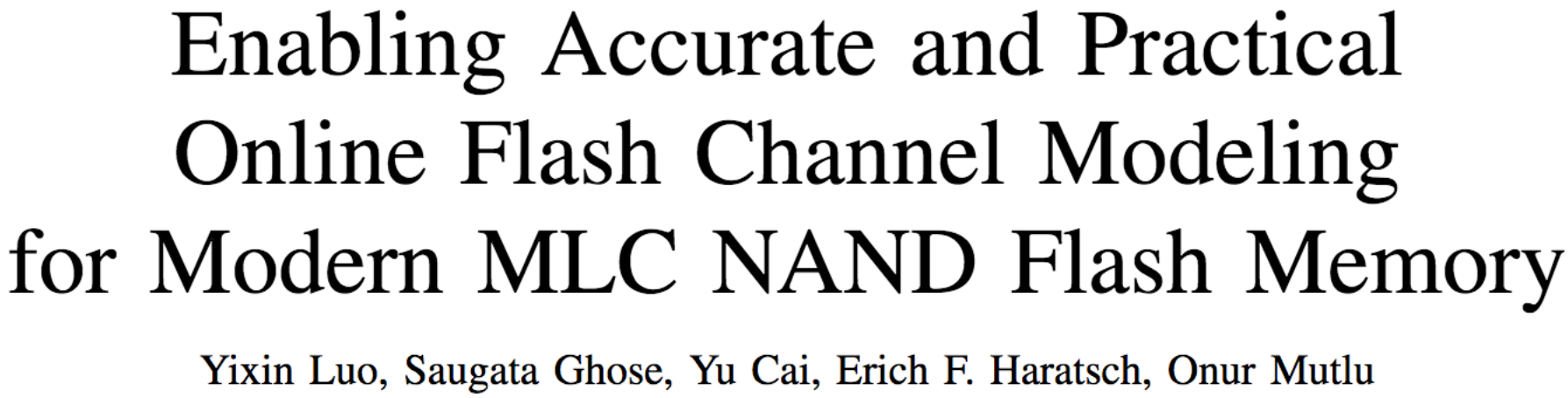
Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques

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.



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]

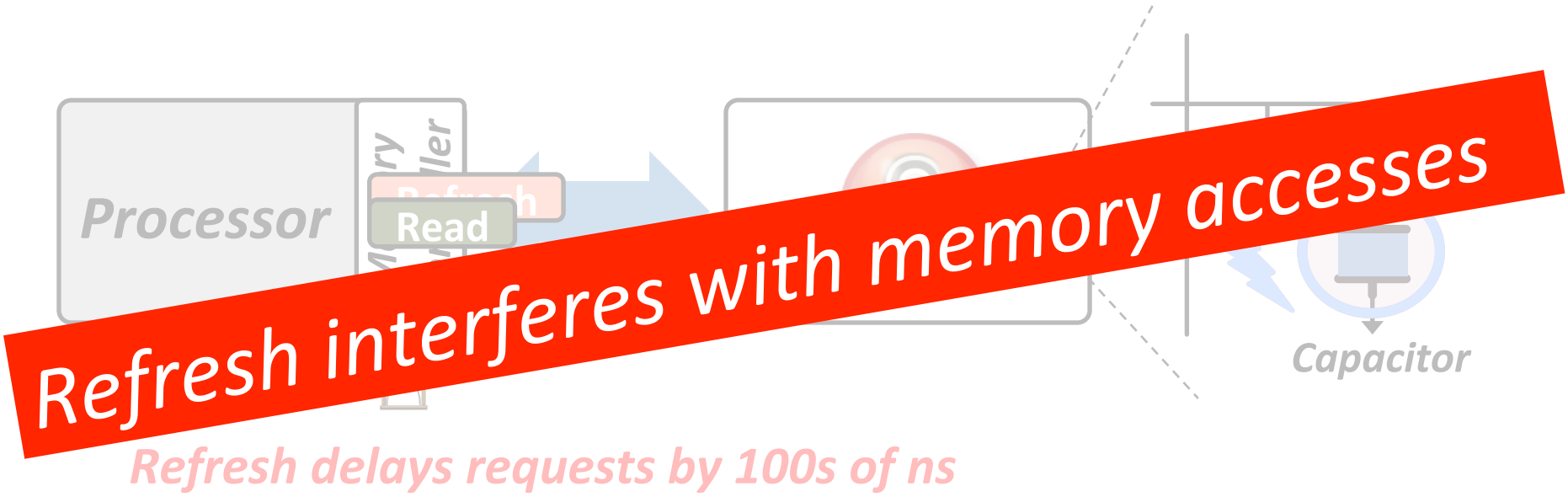
- Understand retention time behavior in DRAM [Liu+ ISCA'13]

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

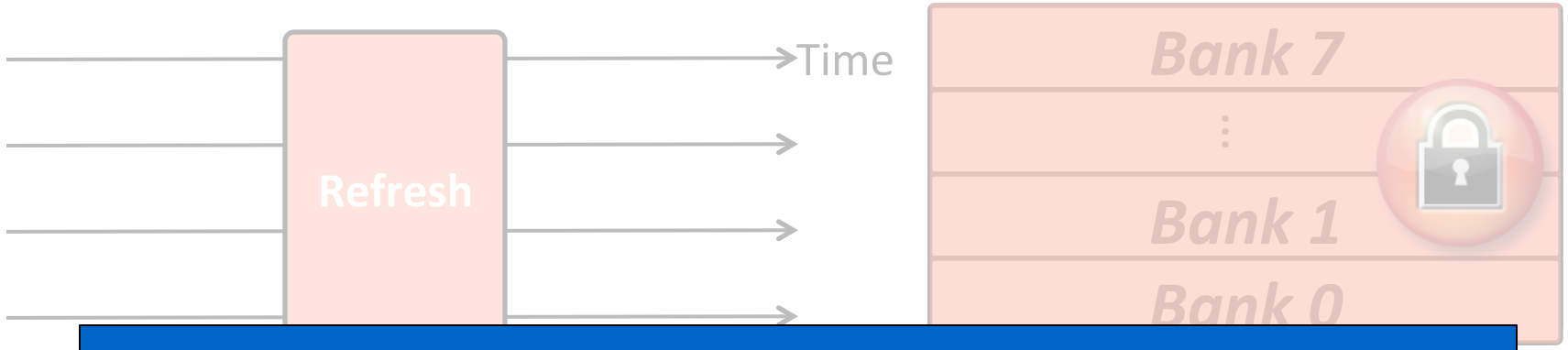
Chang+, "Improving DRAM Performance by Parallelizing Refreshes with Accesses," HPCA 2014.

Refresh Penalty



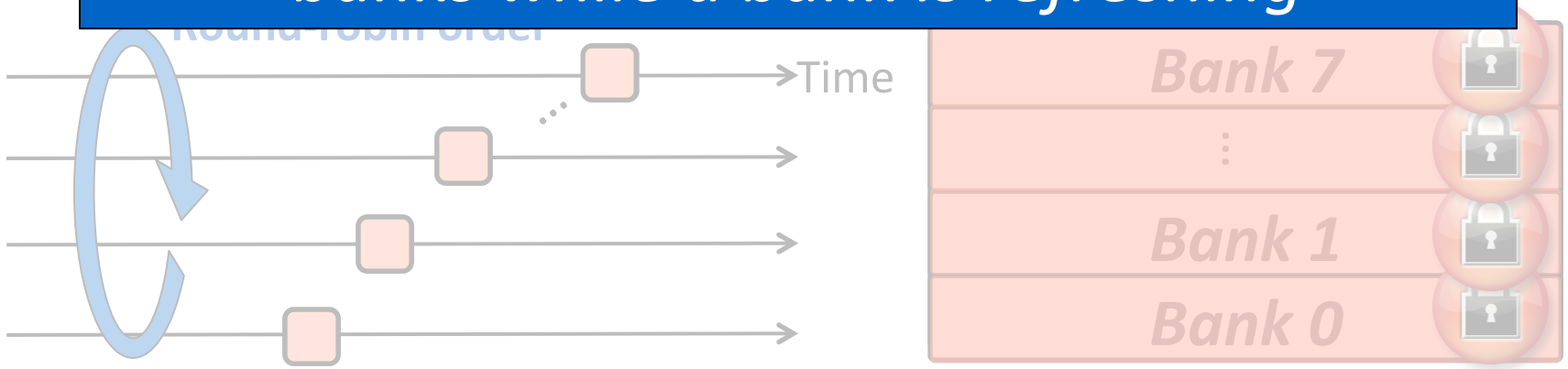
Existing Refresh Modes

All-bank refresh in commodity DRAM (DDRx)



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

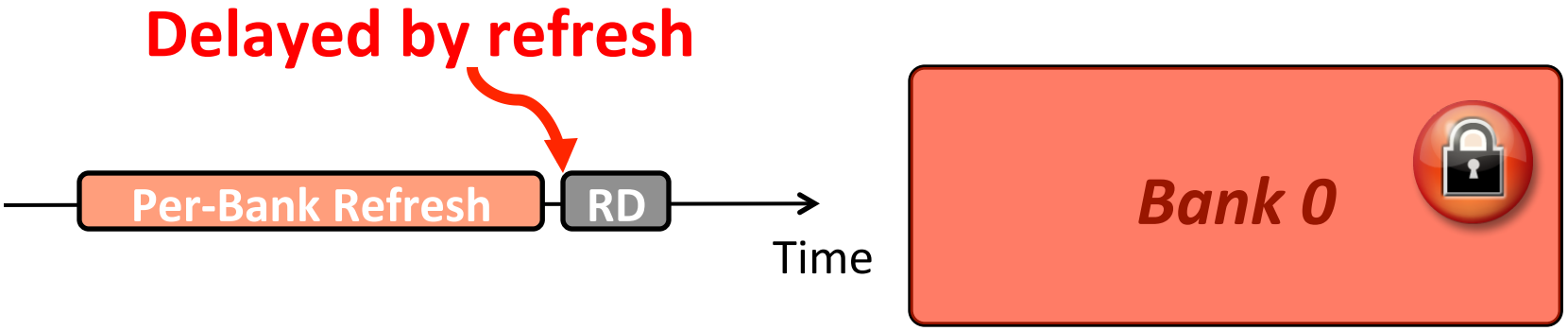
- **Dynamic Access-Refresh Parallelization (DARP)**
  - An improved scheduling policy for per-bank refreshes
  - Exploits refresh scheduling flexibility in DDR DRAM

- 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**

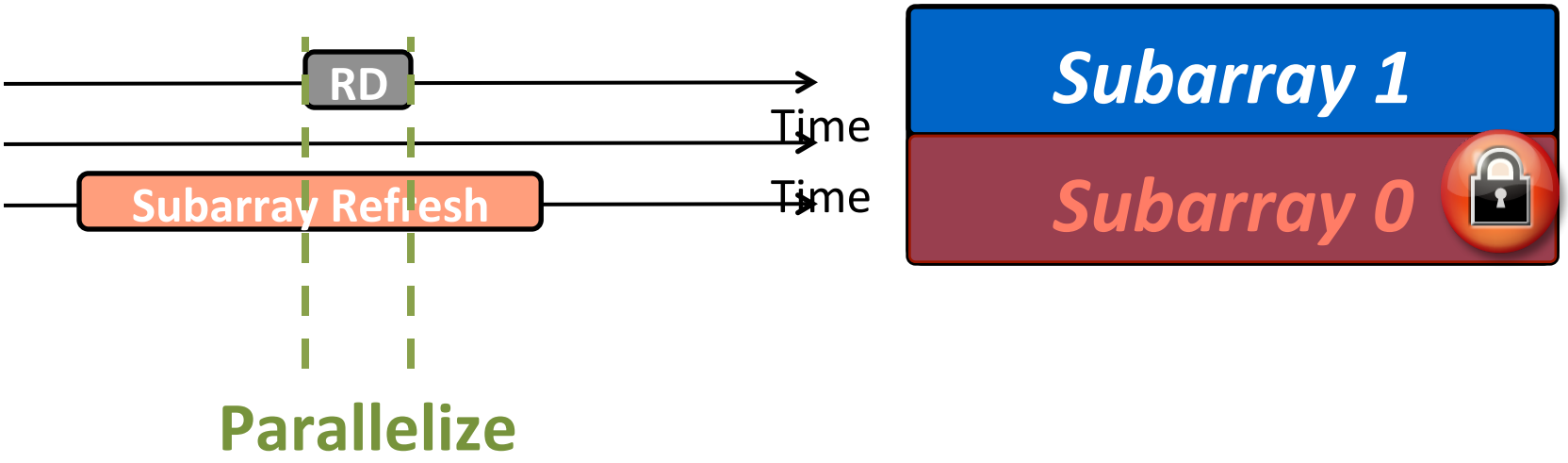
Shortcomings of Per-Bank Refresh

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



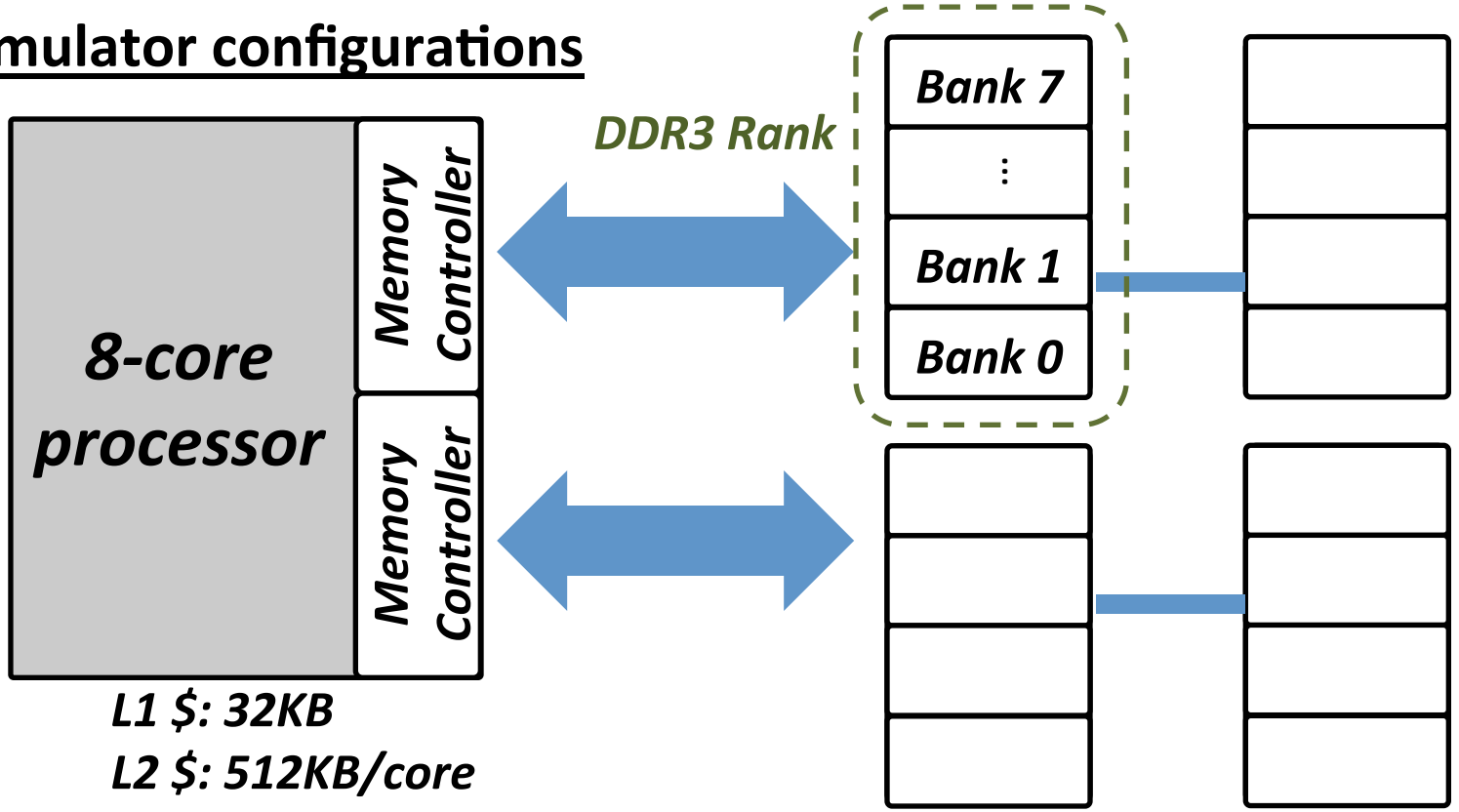
Shortcomings of Per-Bank Refresh

- 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

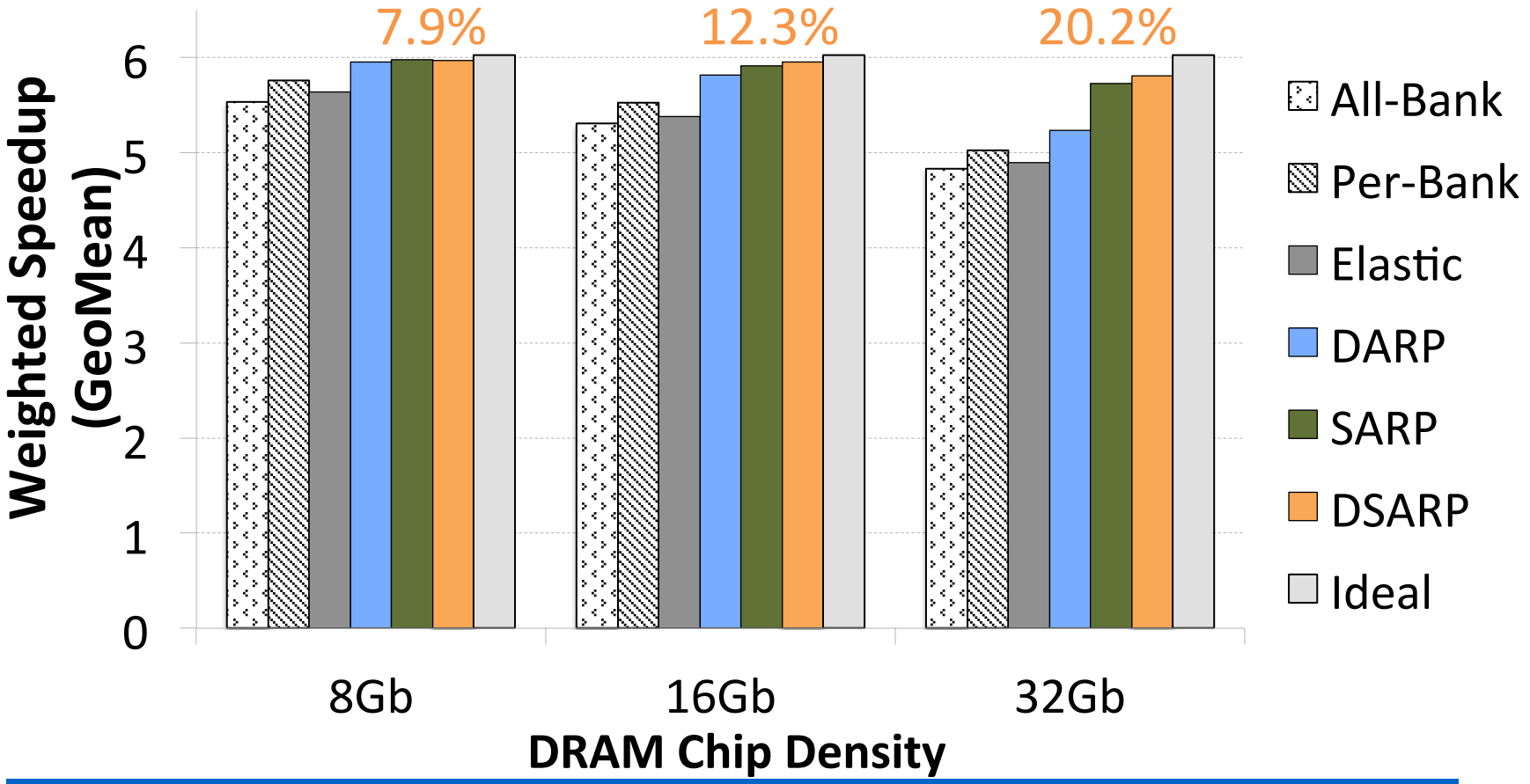


- **100 workloads**: SPEC CPU2006, STREAM, TPC-C/H, random access
- **System performance metric**: *Weighted speedup*

Comparison Points

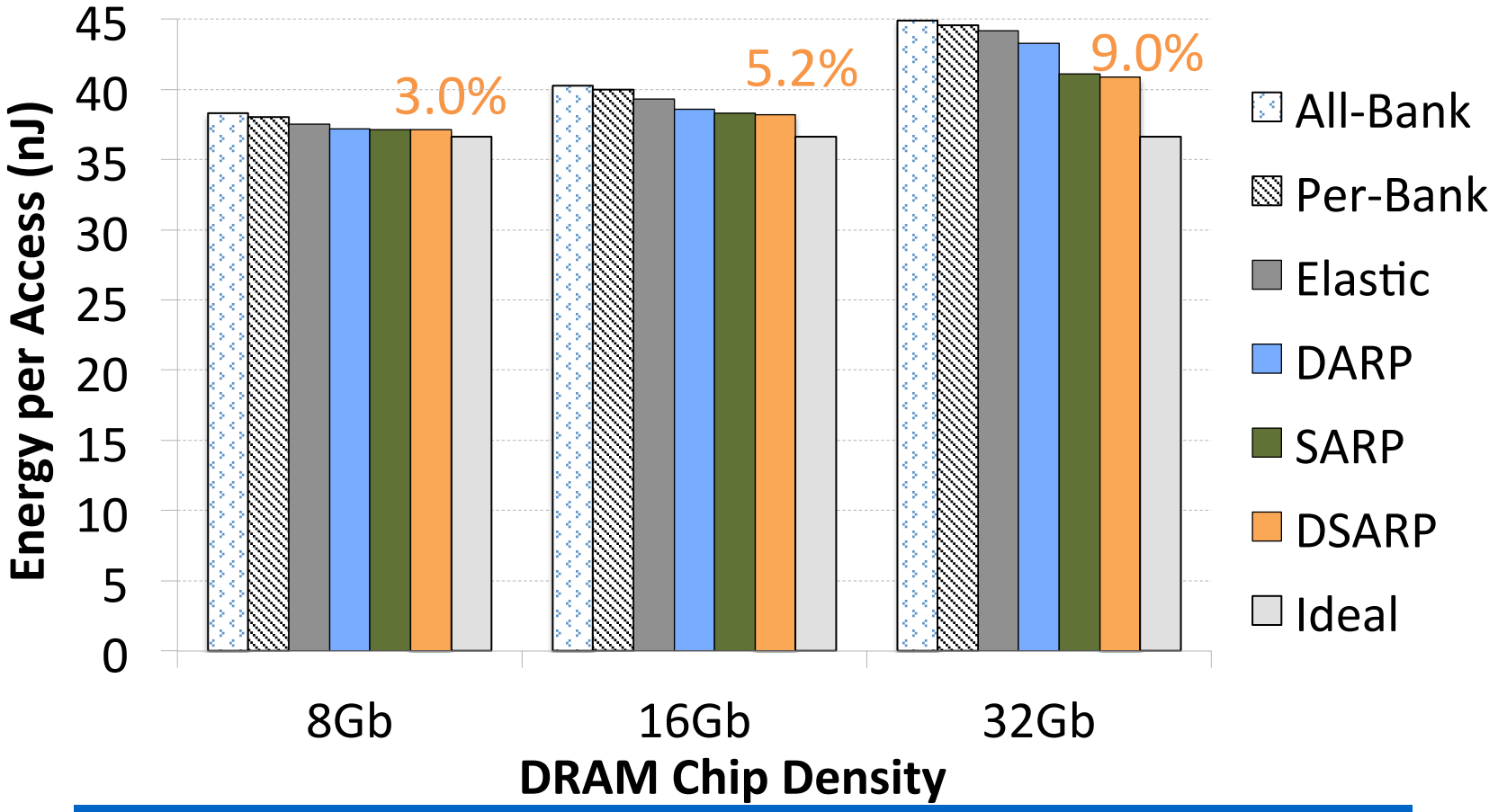
- **All-bank refresh** [DDR3, LPDDR3, …]
- **Per-bank refresh** [LPDDR3]
- **Elastic refresh** [Stuecheli et al., MICRO '10]:
  - Postpones refreshes by a time delay based on the predicted rank idle time to avoid interference on memory requests
  - Proposed to schedule all-bank refreshes without exploiting 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

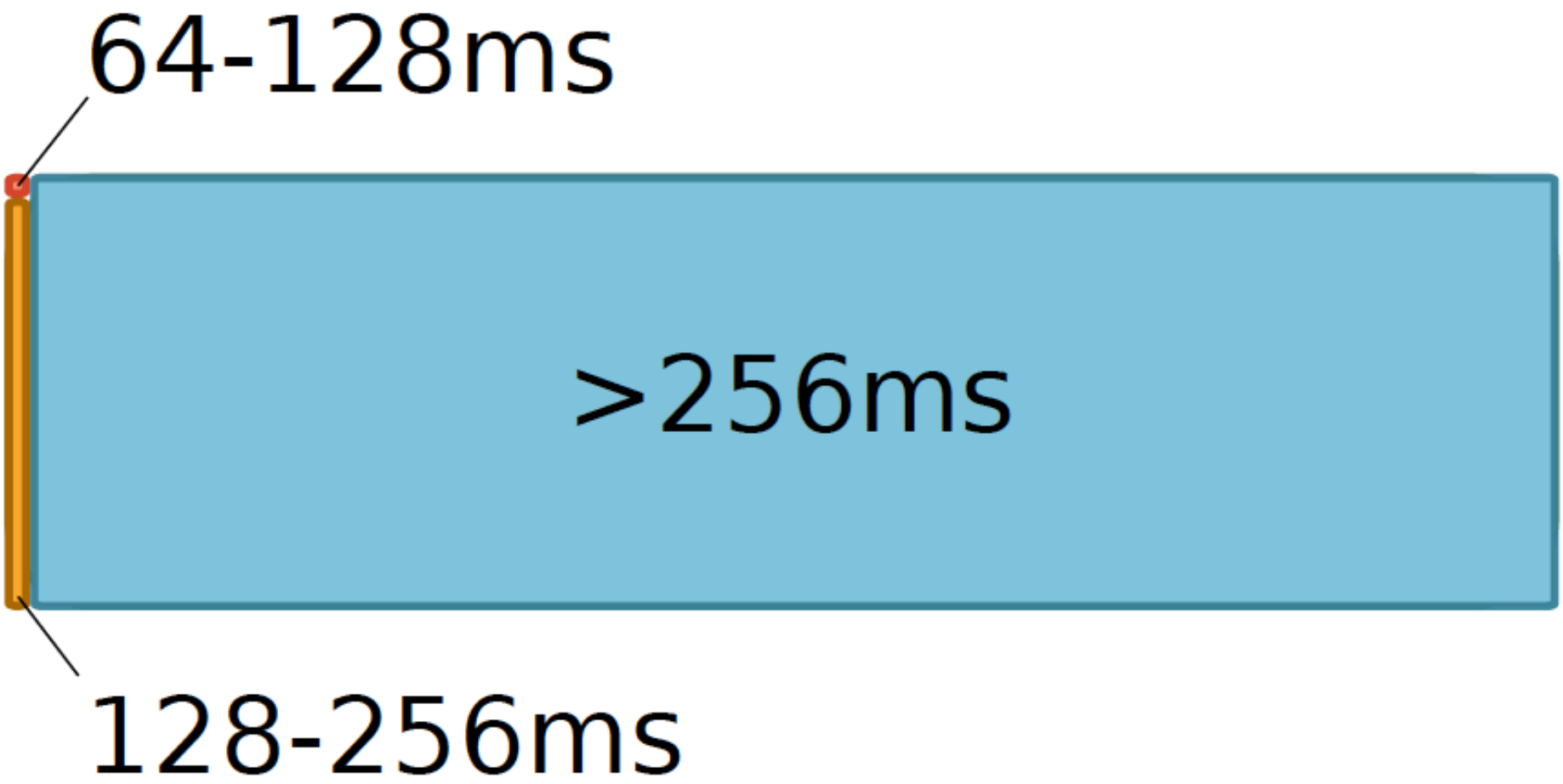
Kevin Kai-Wei Chang Donghyuk Lee Zeshan Chishti†
Alaa R. Alameldeen† Chris Wilkerson† Yoongu Kim Onur Mutlu
Carnegie Mellon University †Intel Labs

Tackling Refresh: Solutions

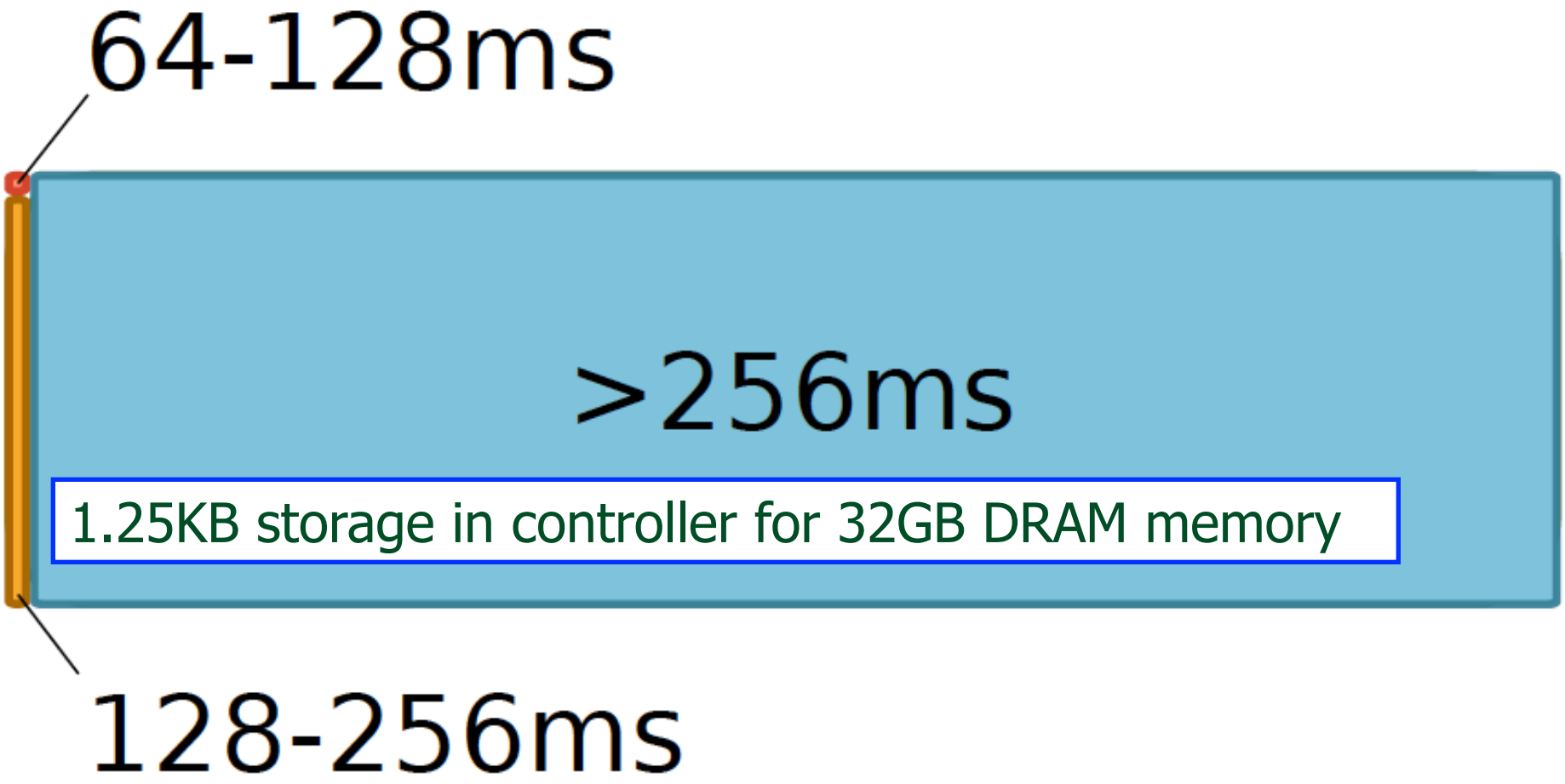
- Parallelize refreshes with accesses [Chang+ HPCA'14]
- Eliminate unnecessary refreshes [Liu+ ISCA'12]
  - Exploit device characteristics
  - Exploit data and application characteristics
- Reduce refresh rate and detect+correct errors that occur [Khan+ SIGMETRICS'14]
- Understand retention time behavior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

Most Refreshes Are Unnecessary

- Retention Time Profile of DRAM looks like this:

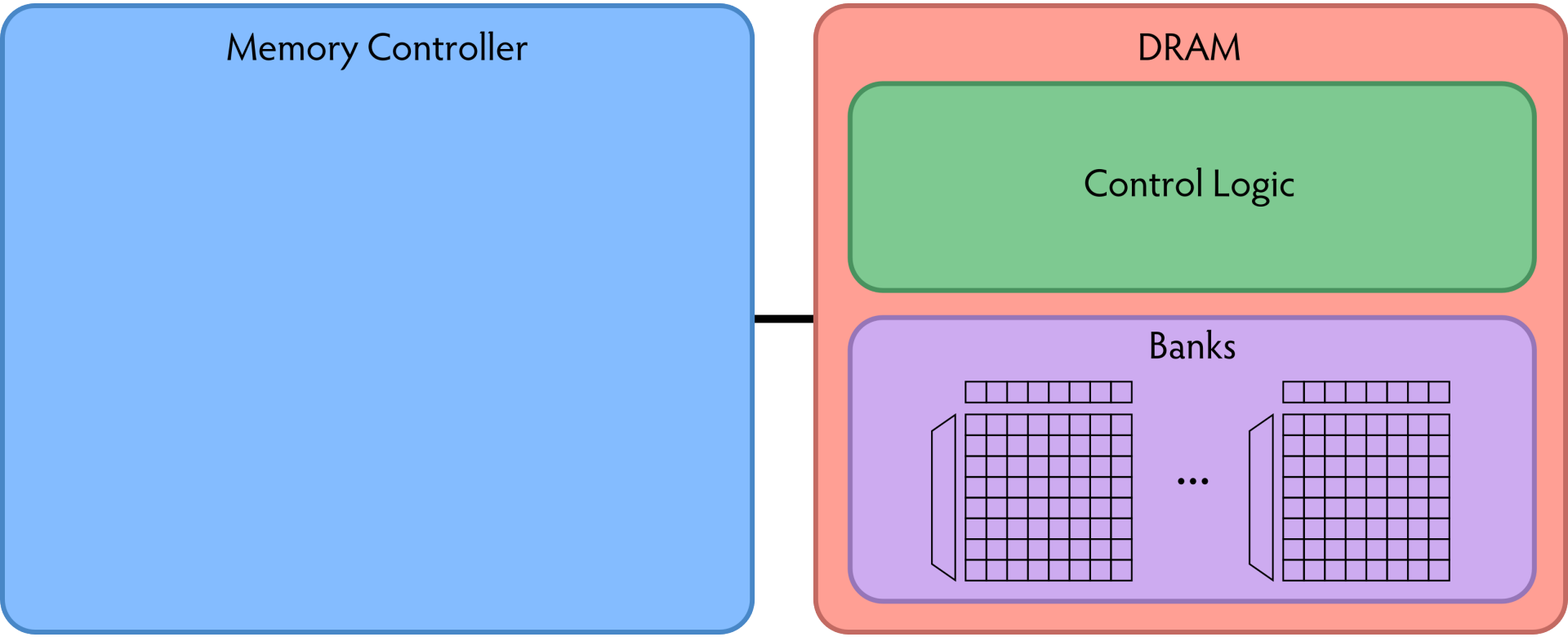


RAIDR: Eliminating Unnecessary Refreshes



Can reduce refreshes by ~75%
→ reduces energy consumption and improves performance

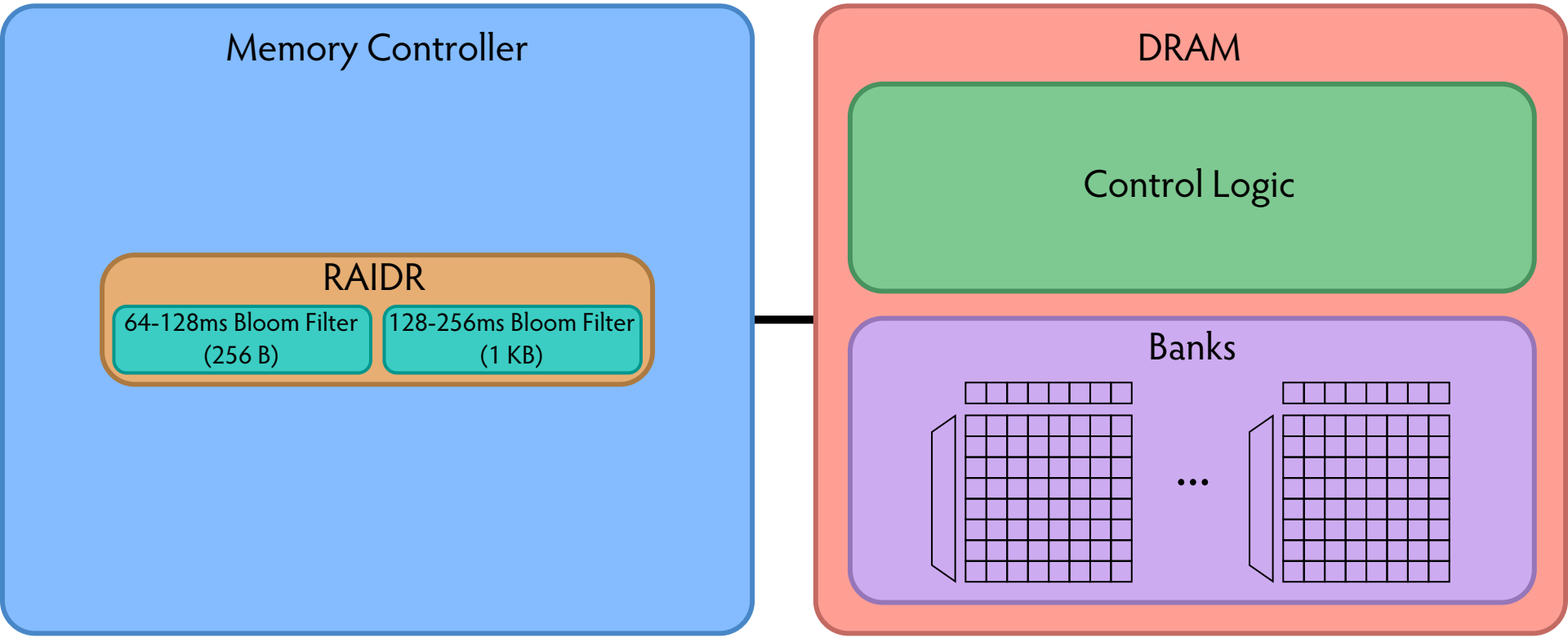
RAIDR: Baseline Design



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

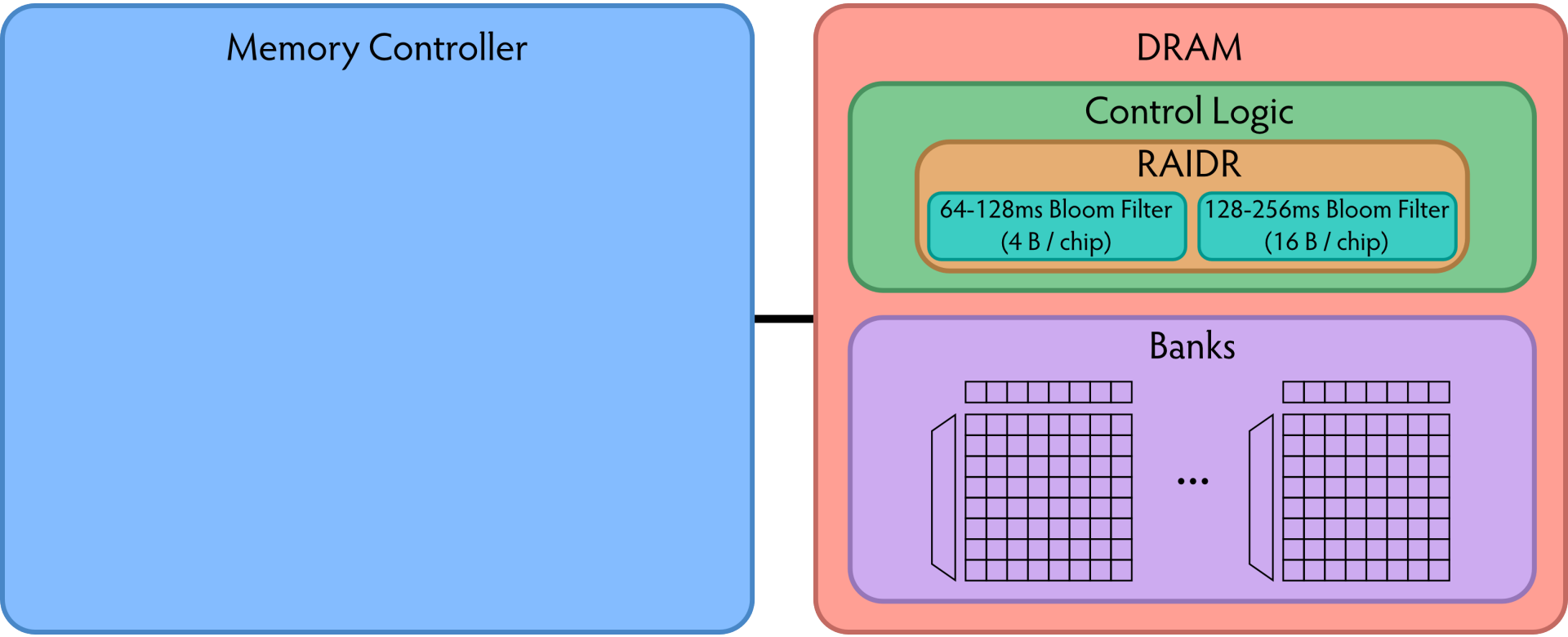
RAIDR can be implemented in either the controller or DRAM

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



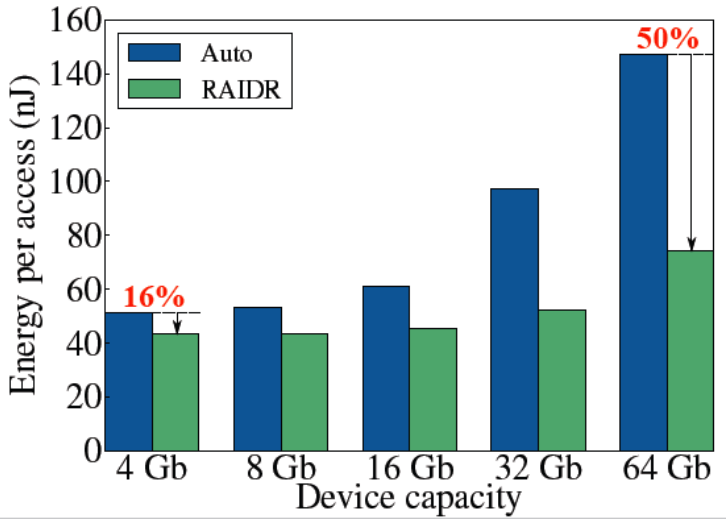
Overhead of RAIDR in DRAM chip:

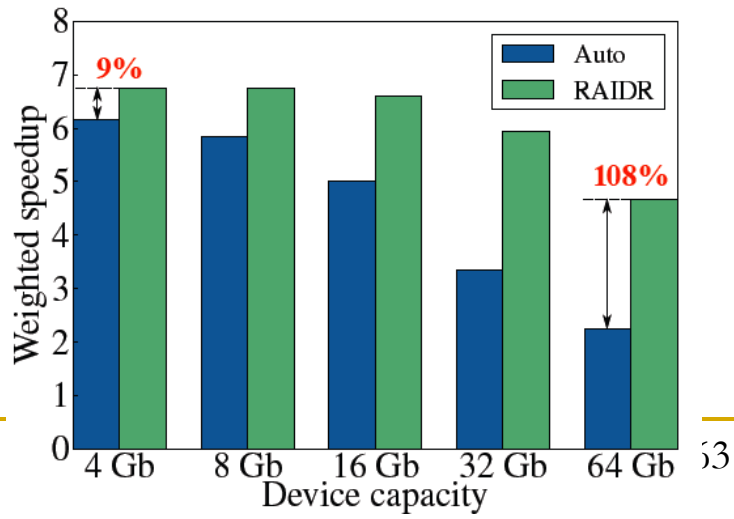
Per-chip overhead: 20B Bloom Filters, 1 counter (4 Gbit chip)

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

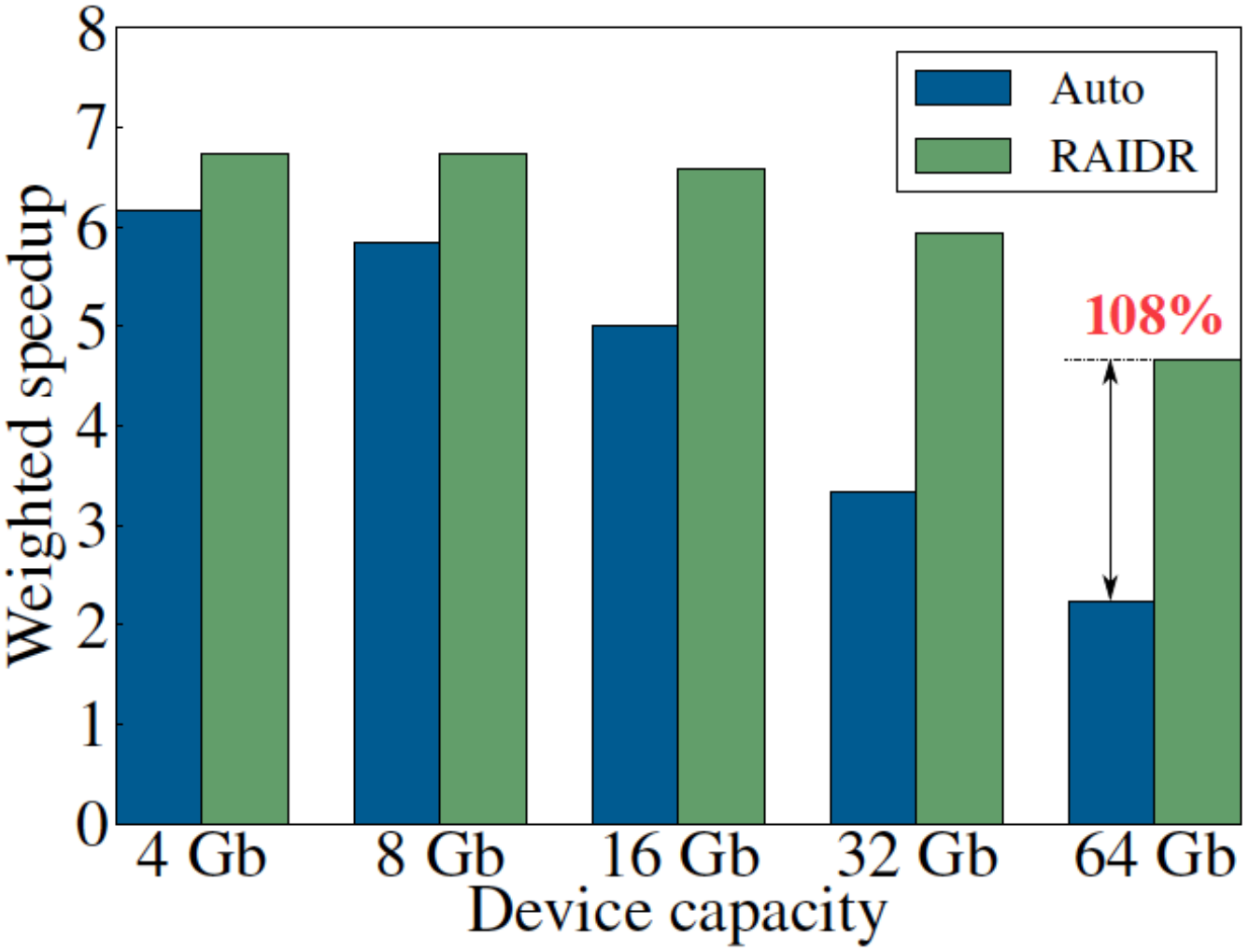
RAIDR: Results and Takeaways

- 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



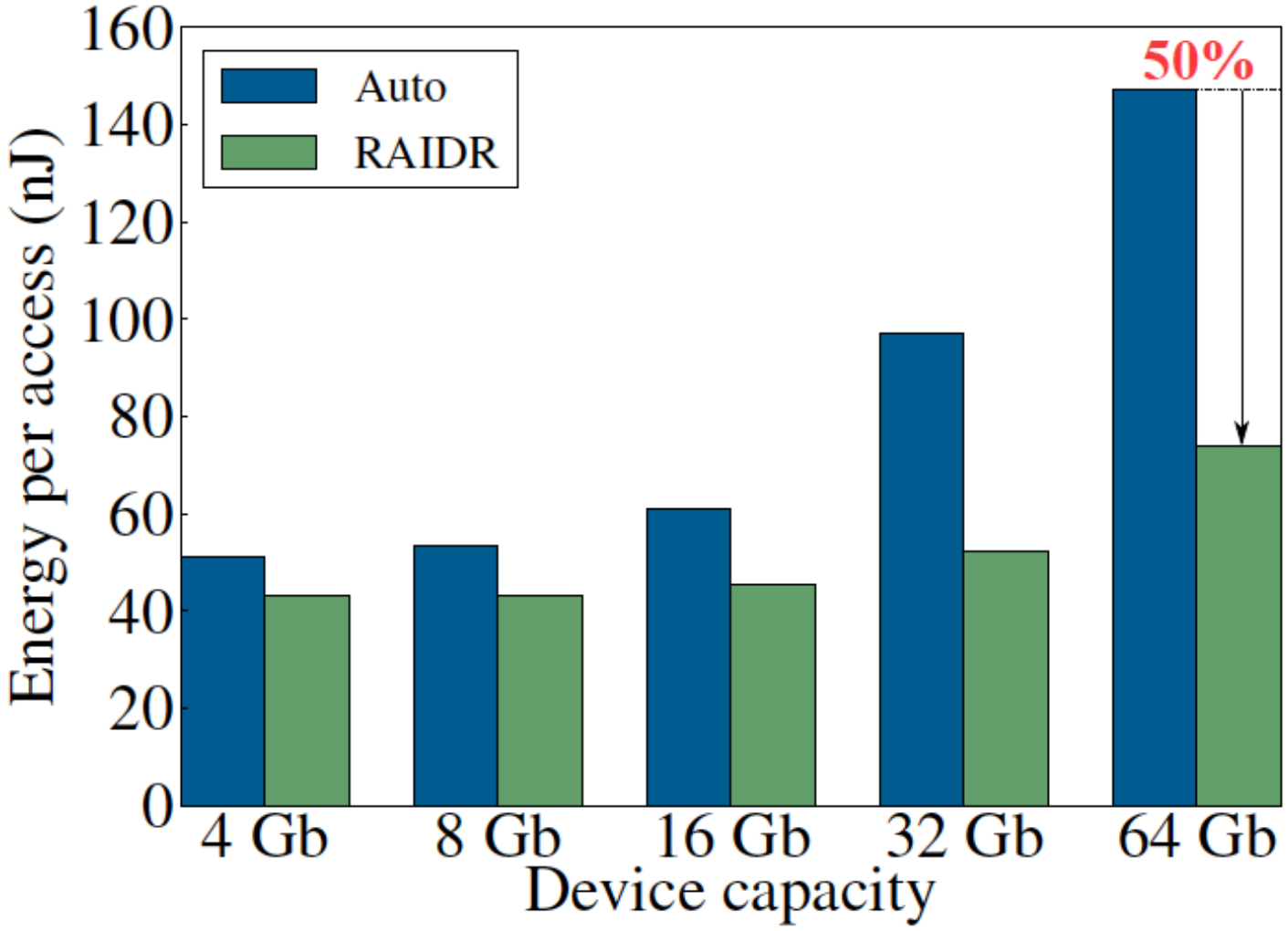


DRAM Device Capacity Scaling: Performance



RAIDR performance benefits increase with DRAM chip capacity

DRAM Device Capacity Scaling: Energy

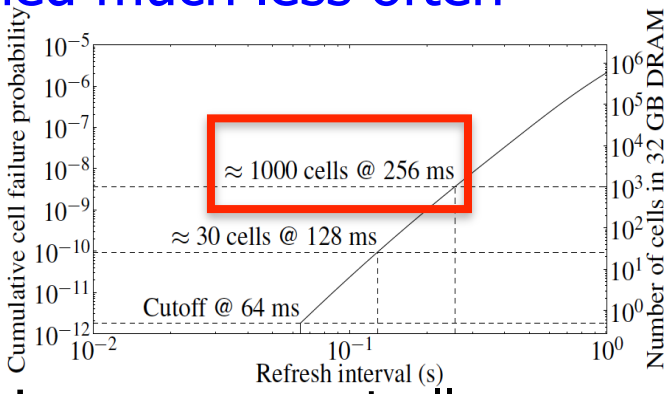


RAIDR energy benefits increase with DRAM chip capacity

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

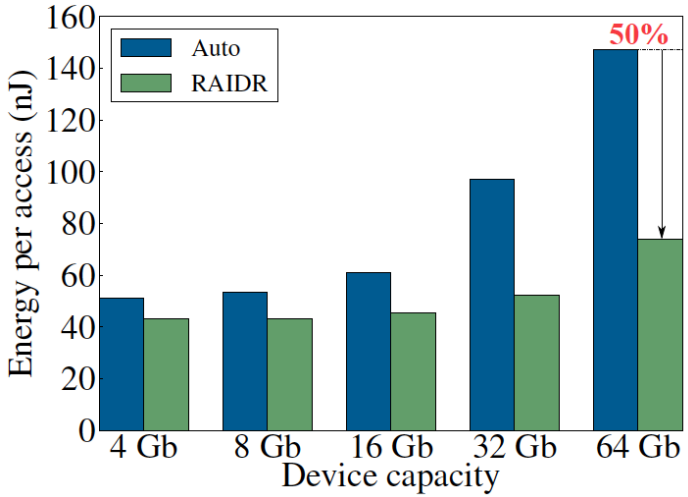
RAIDR: Eliminating Unnecessary Refreshes

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



- Key idea: Refresh rows containing weak cells more frequently, other rows less frequently
  1. Profiling: Profile retention time of all rows
  2. Binning: Store rows into bins by retention time in memory controller
     Efficient storage with Bloom Filters (only 1.25KB for 32GB memory)
  3. Refreshing: Memory controller refreshes rows in different bins at different rates

- Results: 8-core, 32GB, SPEC, TPC-C, TPC-H
  - 74.6% refresh reduction @ 1.25KB storage
  - ~16%/20% DRAM dynamic/idle power reduction
  - ~9% performance improvement
  - Benefits increase with DRAM capacity



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

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

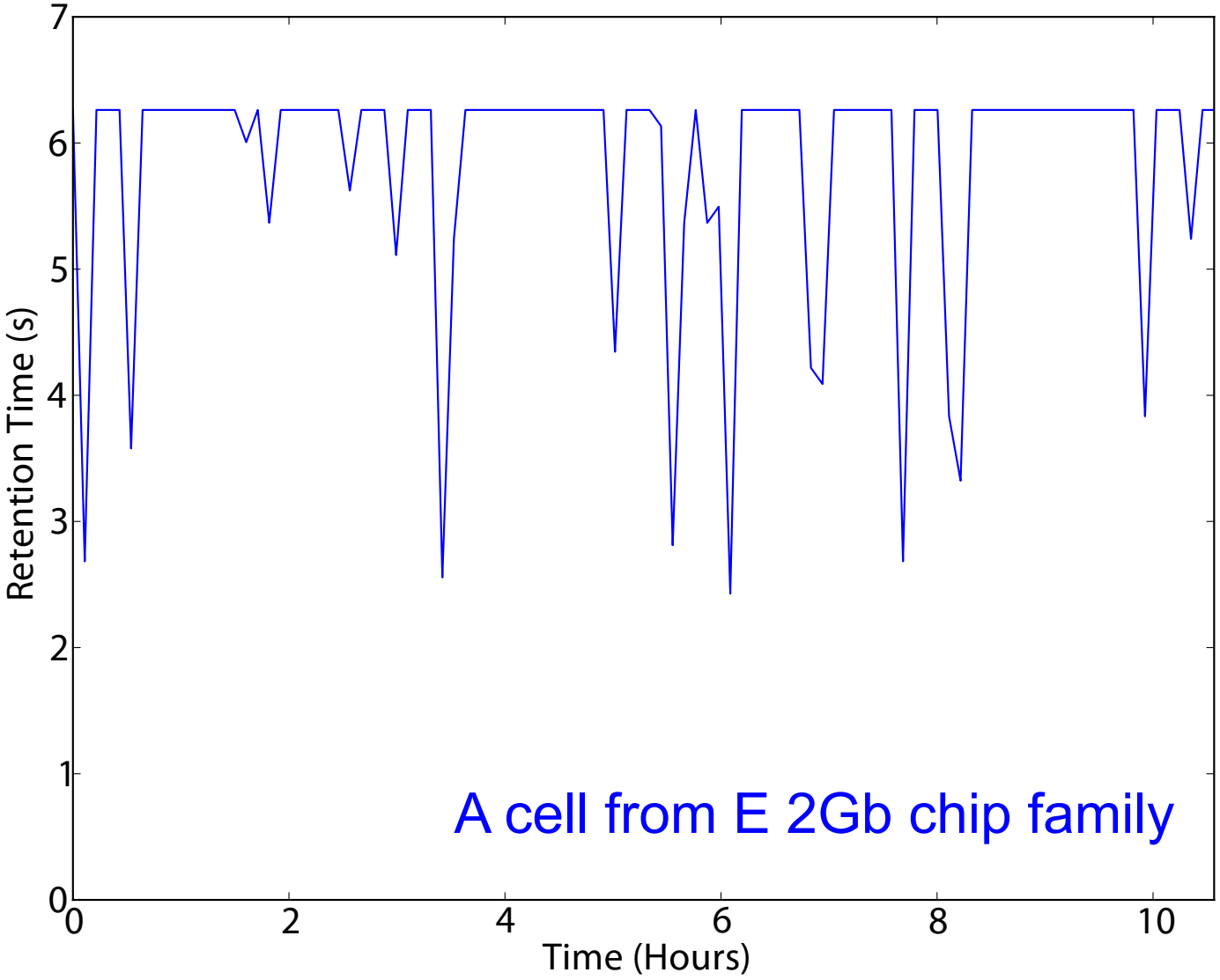
- Question:
  - Can we reliably and accurately determine retention times of all DRAM rows?

Two Challenges to Retention Time Profiling

- Data Pattern Dependence (DPD) of retention time

- Variable Retention Time (VRT) phenomenon

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

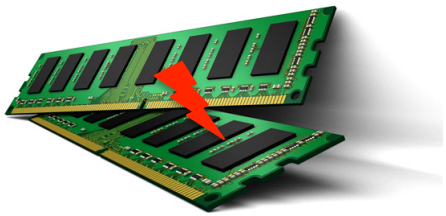
Key Observations:

- **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**

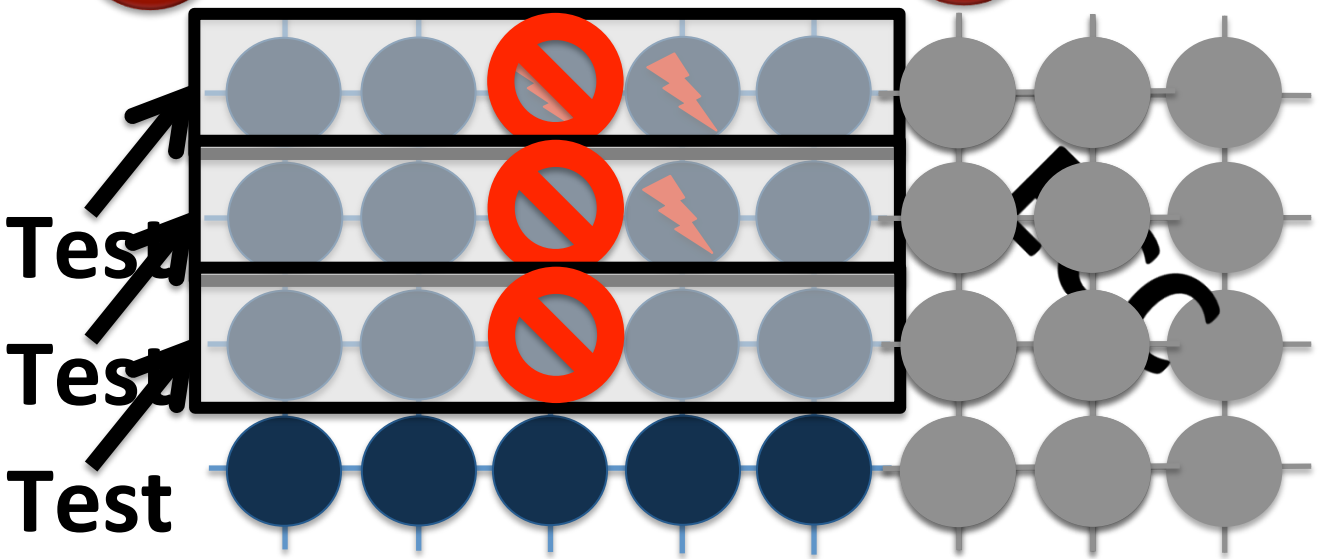
Khan+, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," SIGMETRICS 2014.

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)]

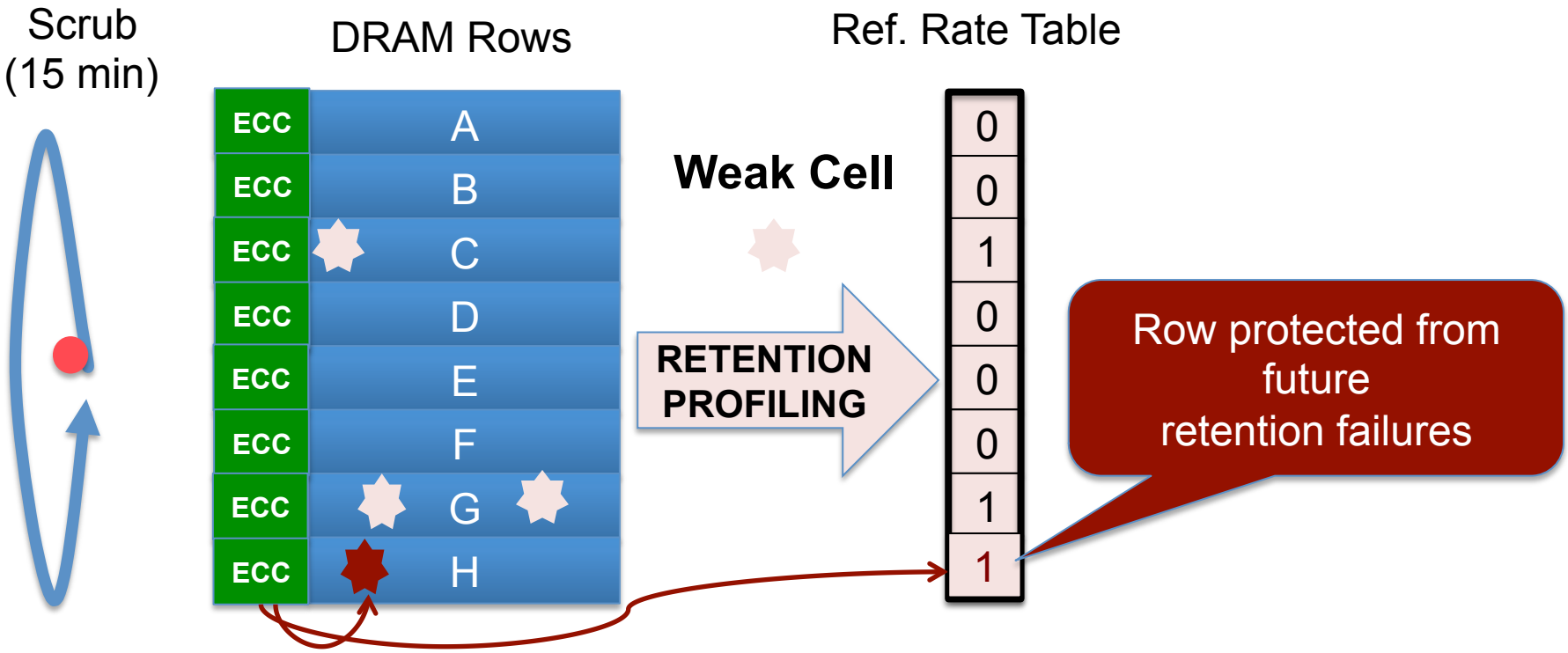
AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems

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

AVATAR

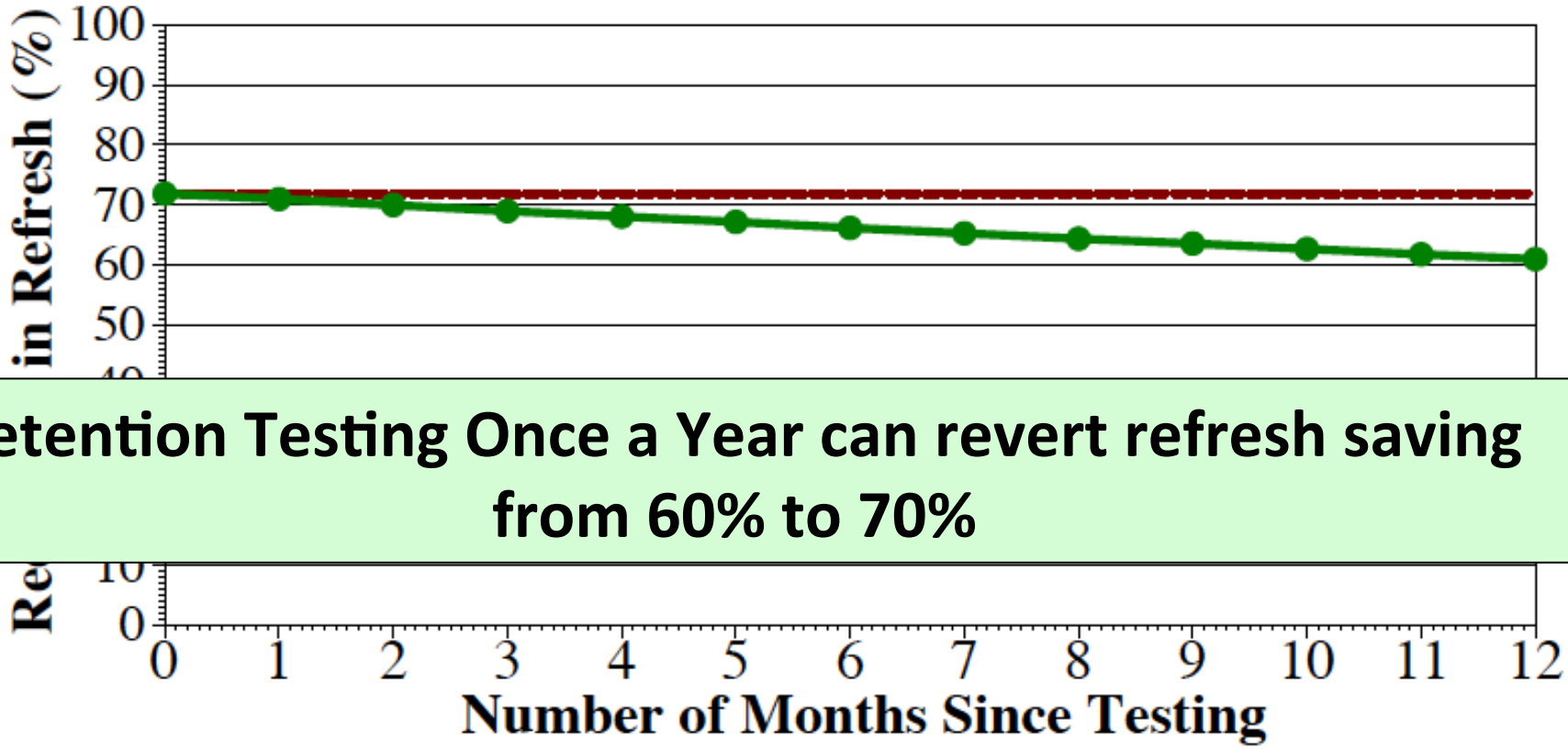
Insight: Avoid retention failures ➔ Upgrade row on ECC error

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



AVATAR mitigates VRT by increasing refresh rate on error

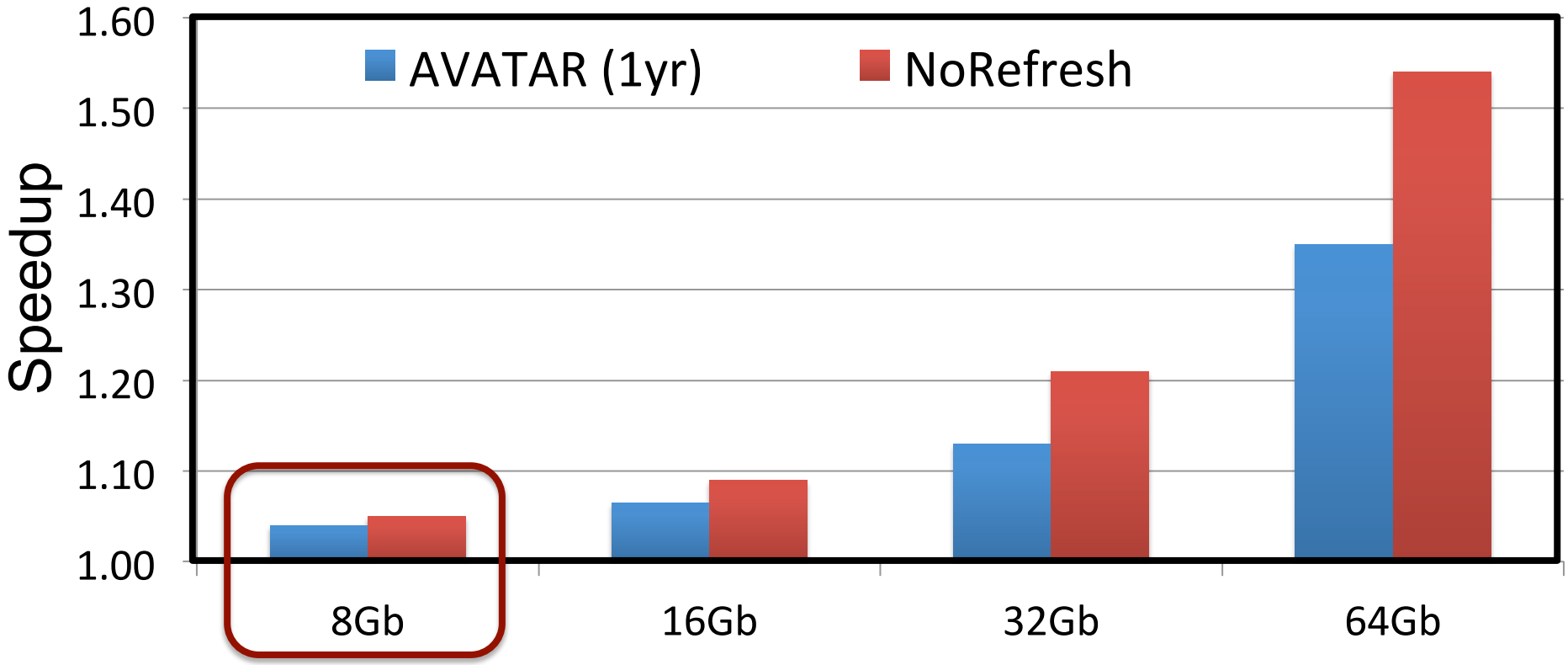
RESULTS: REFRESH SAVINGS



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

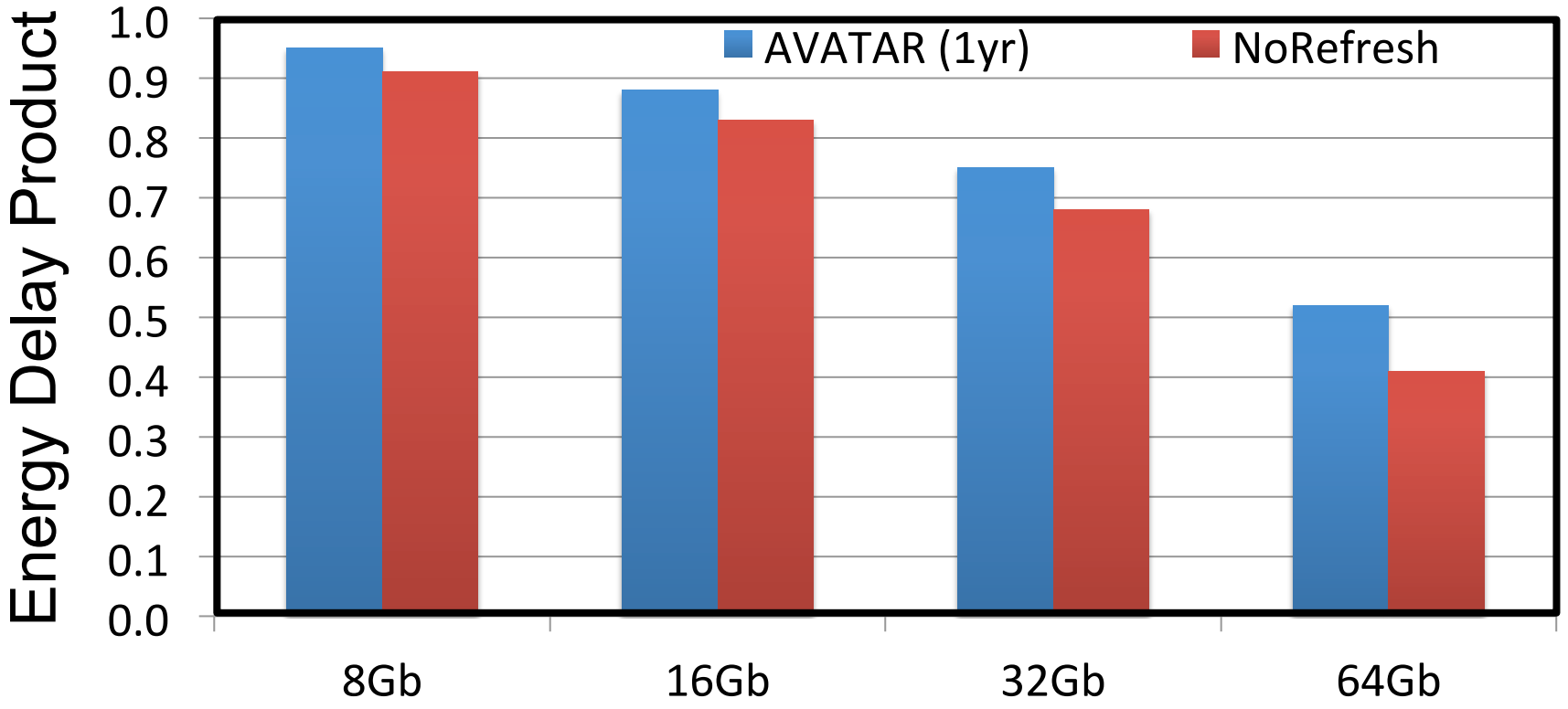
AVATAR reduces refresh by 60%-70%, similar to multi rate refresh but with VRT tolerance

SPEEDUP



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

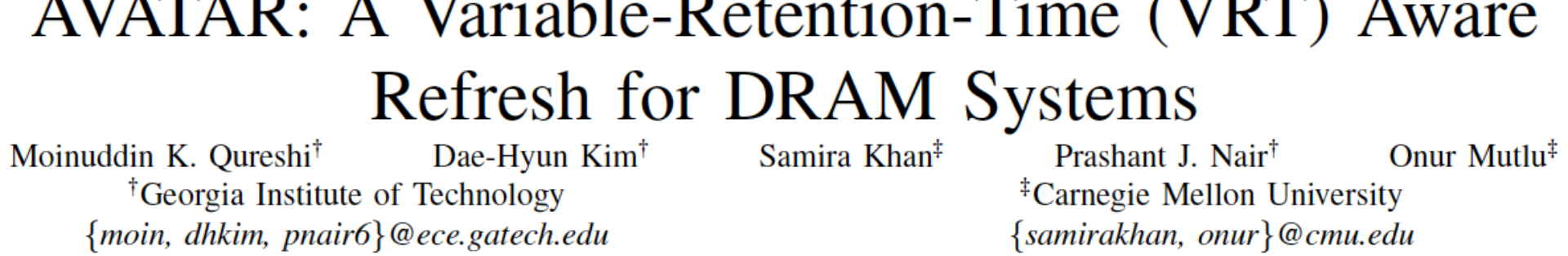
ENERGY DELAY PRODUCT



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
{moin, dhkim, pnair6}@ece.gatech.edu
‡Carnegie Mellon University
{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

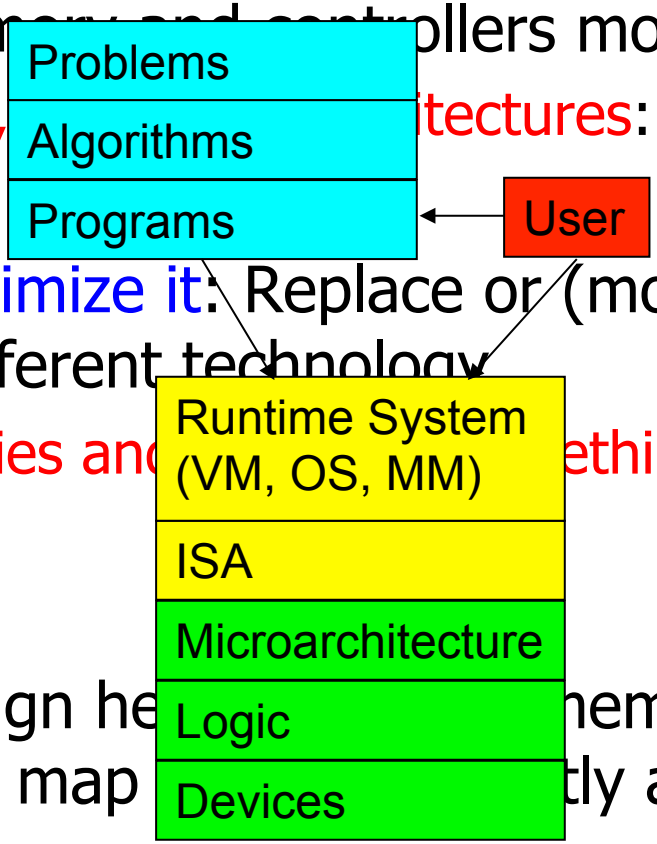
- My current and past students and postdocs
  - Rachata Ausavarungnirun, Abhishek Bhowmick, Amirali Boroumand, Rui Cai, Yu Cai, Kevin Chang, Saugata Ghose, Kevin Hsieh, Tyler Huberty, Ben Jaiyen, Samira Khan, Jeremie Kim, Yoongu Kim, Yang Li, Jamie Liu, Lavanya Subramanian, Donghyuk Lee, Yixin Luo, Justin 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

Solving the Memory Scaling Problem

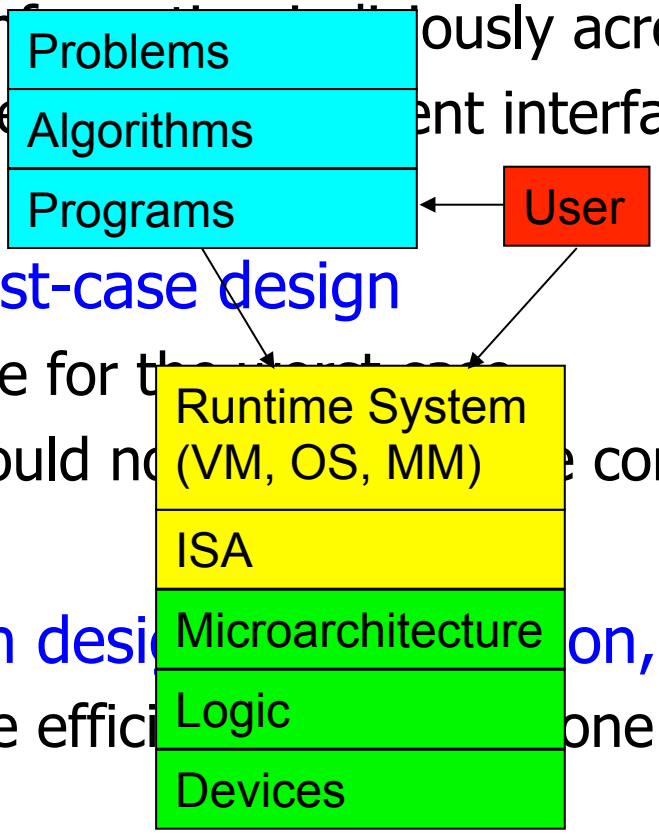
- Fix it: Make mem[illegible]llers more intelligent
  - New interfaces, [illegible]itectures: system-mem codesign
- Eliminate or minimize it: Replace or (more likely) augment DRAM with a different technology
  - New technologies an[illegible]ethinking of memory & storage
- Embrace it: Design he[illegible]nemories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



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

Principles (So Far)

- Better interfaces between layers of the system stack
  - Expose more in ously across the system stack
  - Design more fle ent interfaces
- Better-than-worst-case design
  - Do not optimize for t
  - Worst case should nc common case
- Heterogeneity in desi on, asymmetry)
  - Enables a more effici one size fits all)
- These principles are coupled (and require broad thinking)



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

Referenced Papers

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

- A detailed accompanying overview paper

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

Related Videos and Course Materials

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