

# Processing Data Where It Makes Sense in Modern Computing Systems: Enabling In-Memory Computation

Onur Mutlu

[omutlu@gmail.com](mailto:omutlu@gmail.com)

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

4 October 2018

HNEP 2018 Keynote



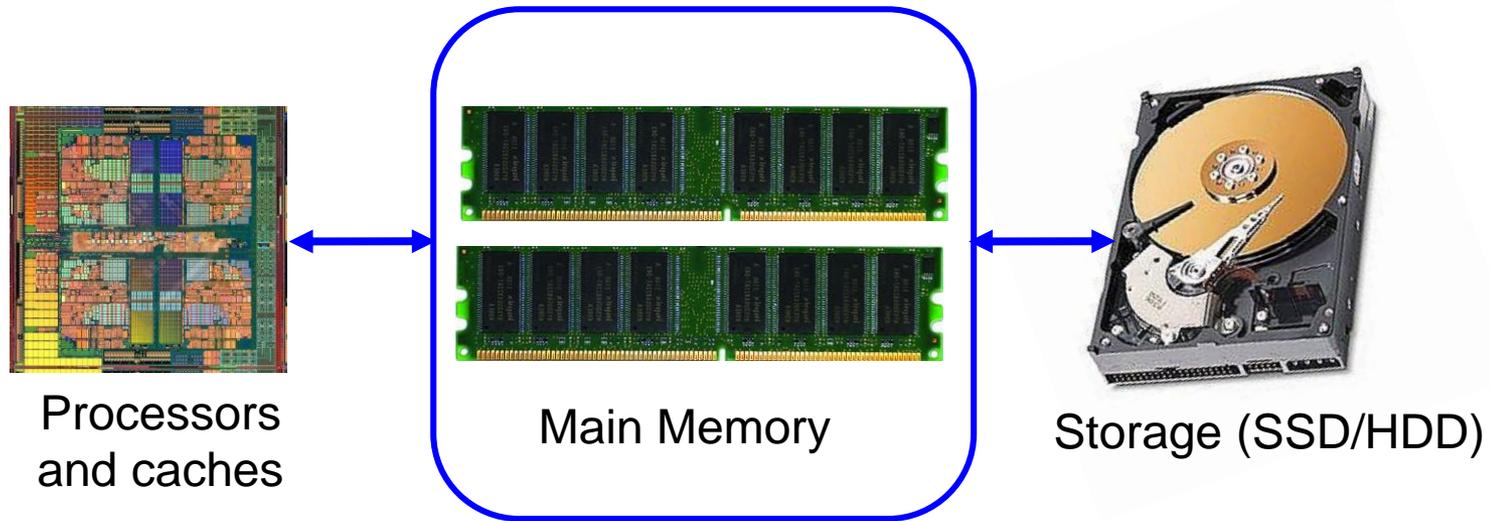
**SAFARI**

**ETH** zürich

**Carnegie Mellon**

# The Main Memory System

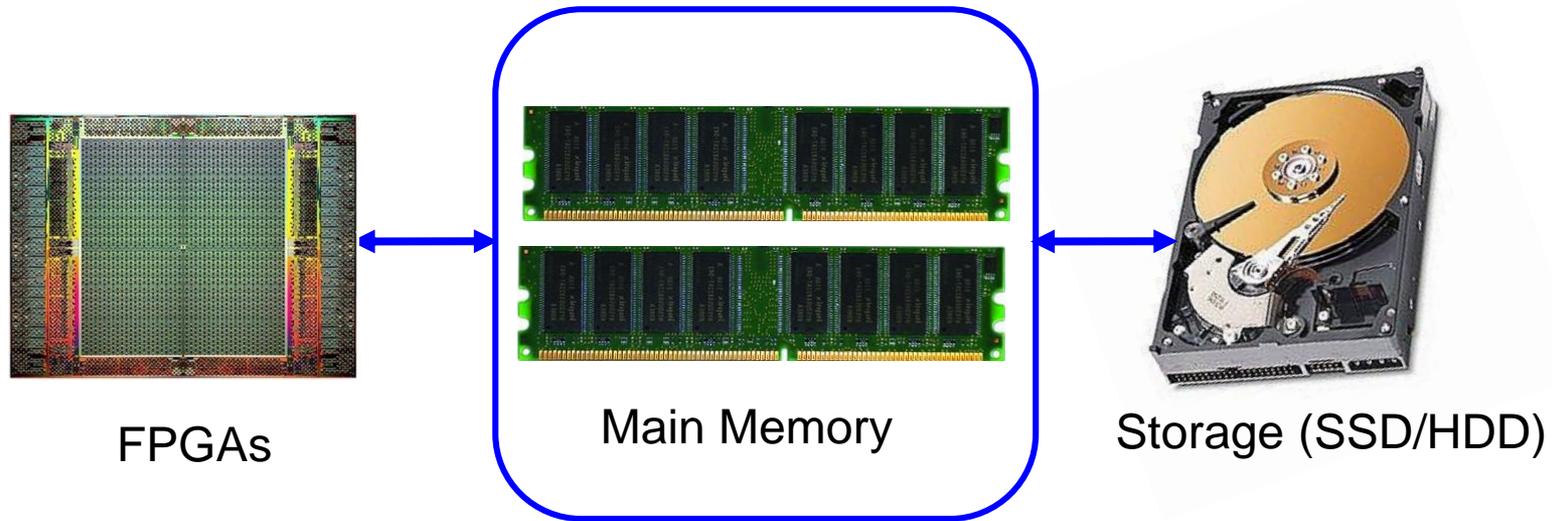
---



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

# The Main Memory System

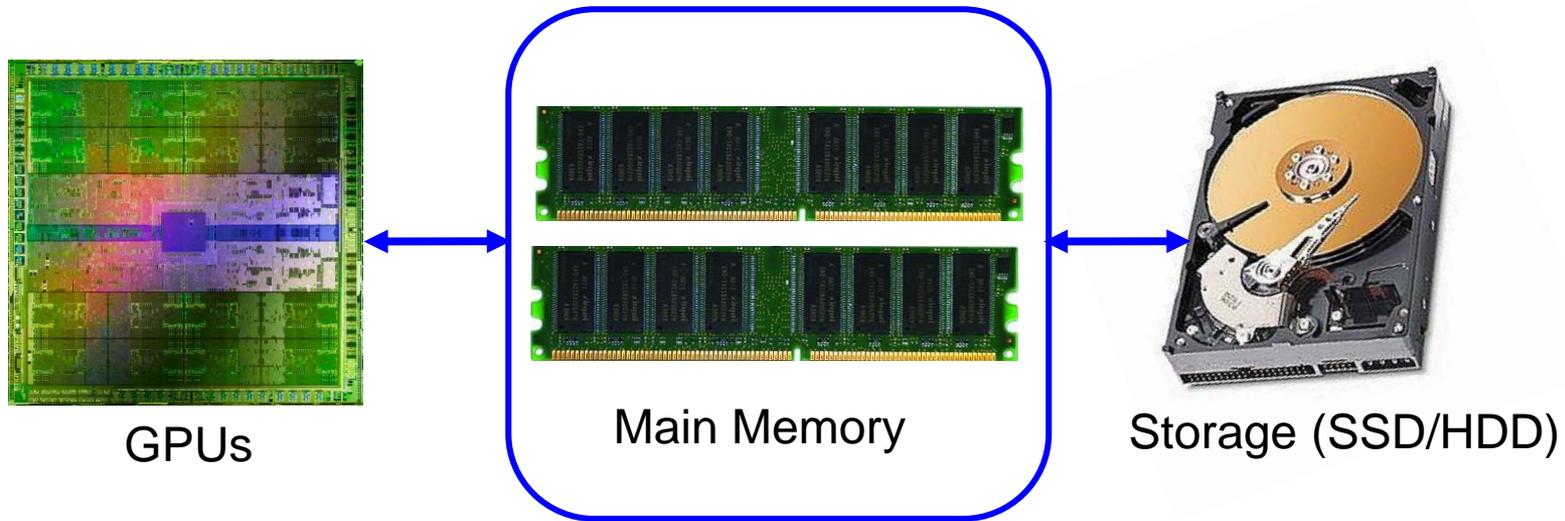
---



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

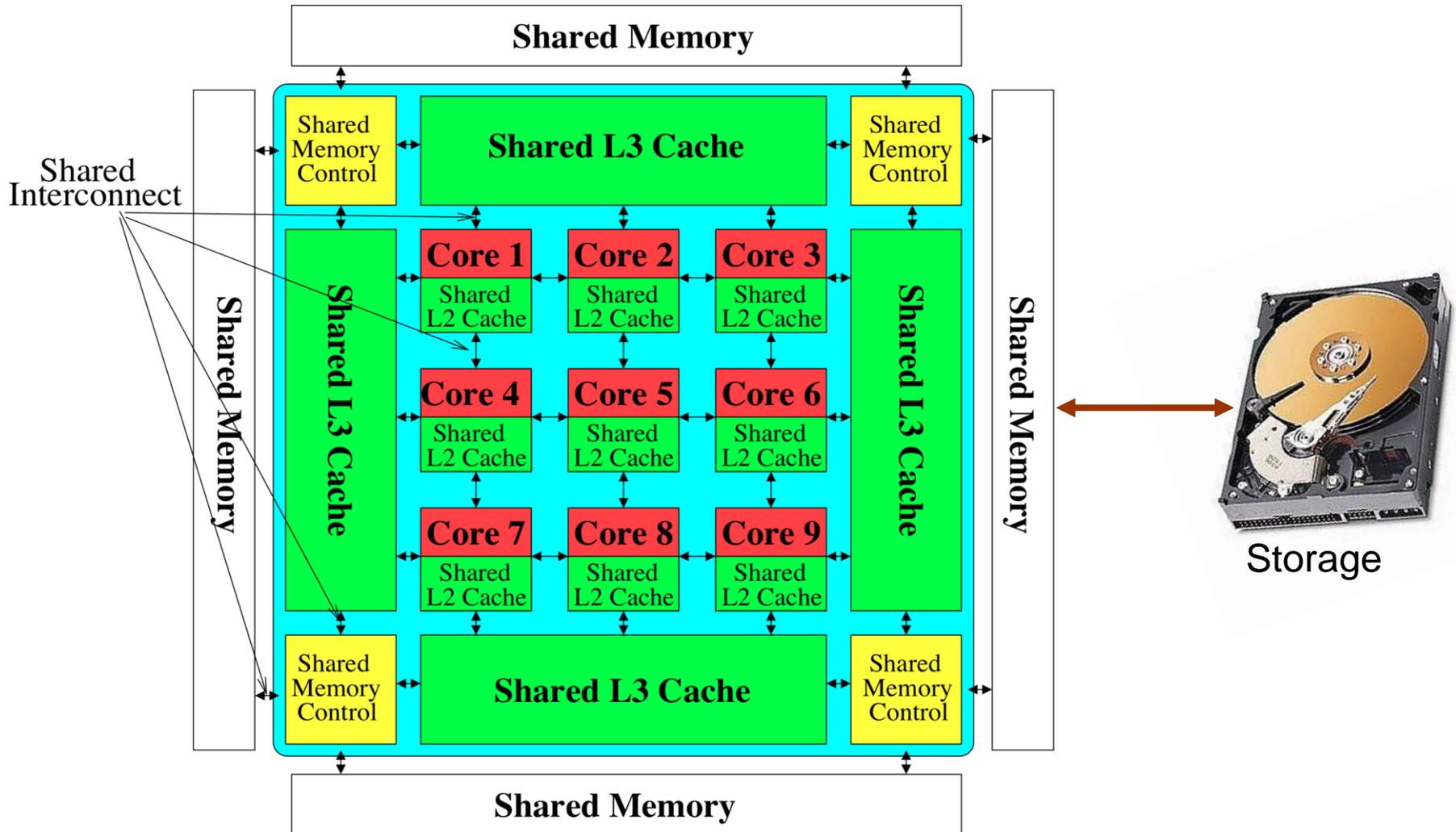
# The Main Memory System

---



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

# Memory System: A *Shared Resource* View



**Most of the system is dedicated to storing and moving data**

# State of the Main Memory System

---

- Recent technology, architecture, and application trends
  - lead to new requirements
  - exacerbate old requirements
- DRAM and memory controllers, as we know them today, are (will be) unlikely to satisfy all requirements
- Some emerging non-volatile memory technologies (e.g., PCM) enable new opportunities: memory+storage merging
- We need to rethink the main memory system
  - to fix DRAM issues and enable emerging technologies
  - to satisfy all requirements

# Major Trends Affecting Main Memory (I)

---

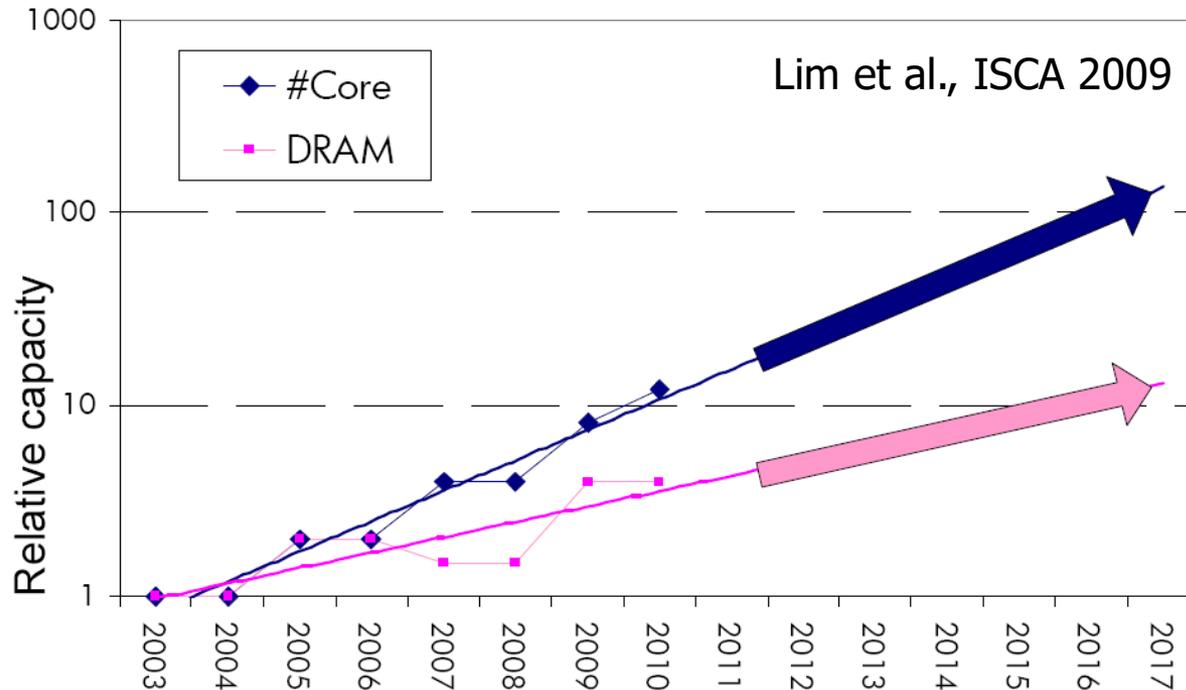
- Need for main memory capacity, bandwidth, QoS increasing
- Main memory energy/power is a key system design concern
- DRAM technology scaling is ending



# Example: The Memory Capacity Gap

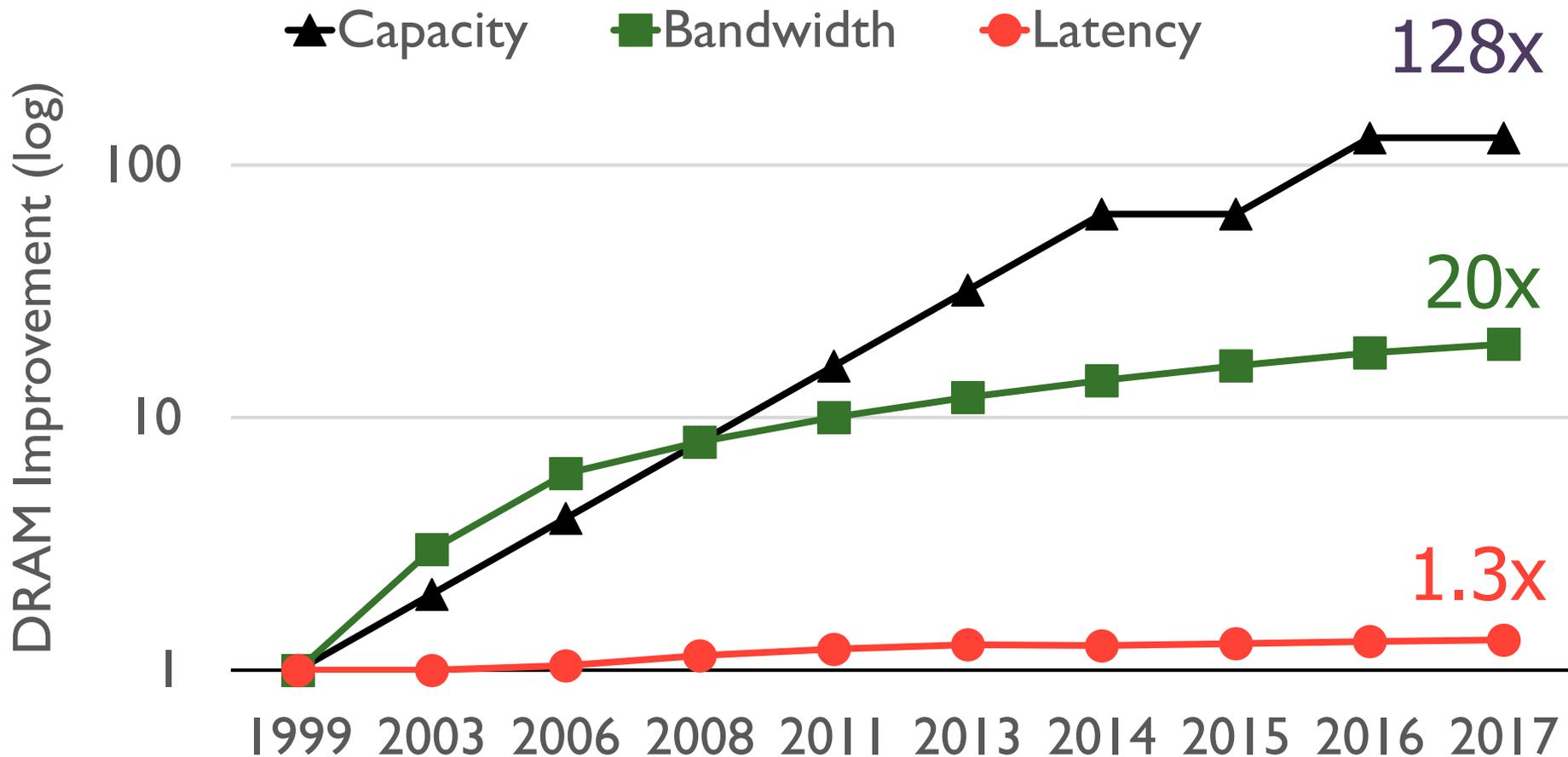
Core count doubling ~ every 2 years

DRAM DIMM capacity doubling ~ every 3 years



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

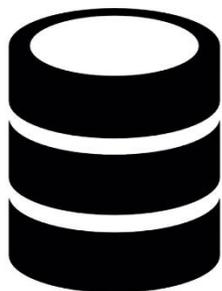
# Example: Capacity, Bandwidth & Latency



Memory latency remains almost constant

# DRAM Latency Is Critical for Performance

---



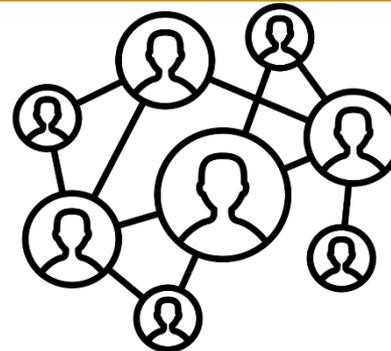
## In-memory Databases

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



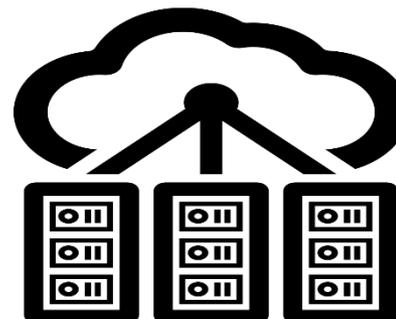
## In-Memory Data Analytics

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



## Graph/Tree Processing

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

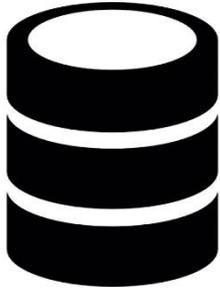


## Datacenter Workloads

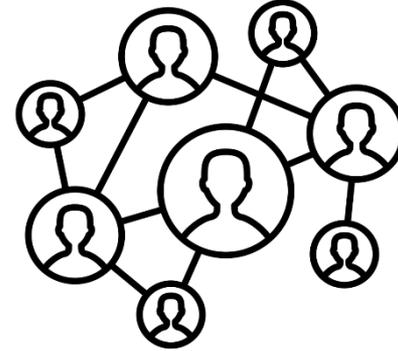
[Kanev+ (Google), ISCA'15]

# DRAM Latency Is Critical for Performance

---



## In-memory Databases



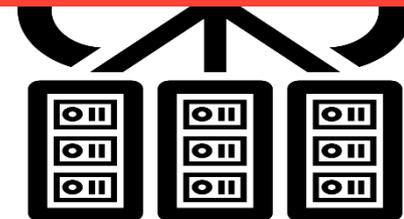
## Graph/Tree Processing

Long memory latency → performance bottleneck



## In-Memory Data Analytics

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



## Datacenter Workloads

[Kanev+ (Google), ISCA'15]

# Major Trends Affecting Main Memory (III)

---

- Need for main memory capacity, bandwidth, QoS increasing
- Main memory energy/power is a key system design concern
  - ~40-50% energy spent in off-chip memory hierarchy [Lefurgy, IEEE Computer'03] >40% power in DRAM [Ware, HPCA'10][Paul,ISCA'15]
  - DRAM consumes power even when not used (periodic refresh)
- DRAM technology scaling is ending

# Major Trends Affecting Main Memory (IV)

---

- Need for main memory capacity, bandwidth, QoS increasing
- Main memory energy/power is a key system design concern
- DRAM technology scaling is ending
  - ITRS projects DRAM will not scale easily below X nm
  - Scaling has provided many benefits:
    - higher capacity (density), lower cost, lower energy

# Major Trends Affecting Main Memory (V)

---

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

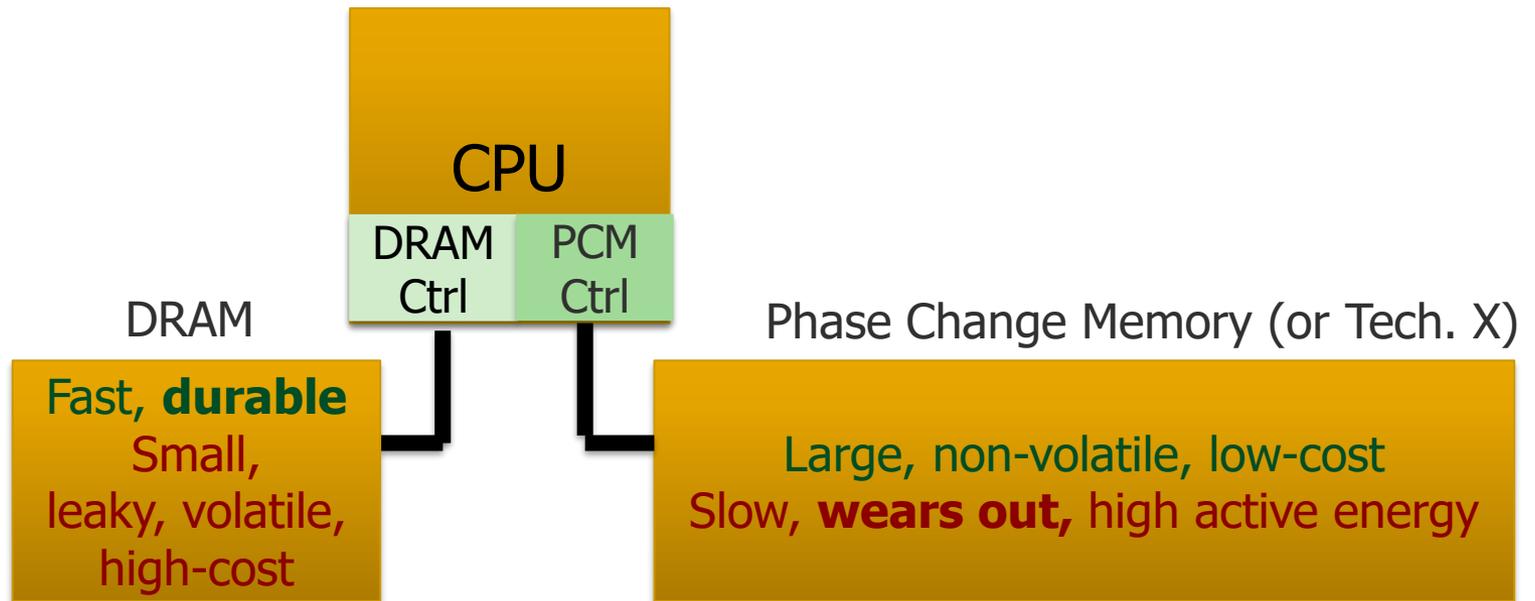

# Major Trends Affecting Main Memory (V)

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

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

# Major Trend: Hybrid Main Memory

---



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

Meza+, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters, 2012.  
Yoon+, "Row Buffer Locality Aware Caching Policies for Hybrid Memories," ICCD 2012 Best Paper Award.

## Main Memory Needs Intelligent Controllers

# Industry Is Writing Papers About It, Too

## DRAM Process Scaling Challenges

### ❖ Refresh

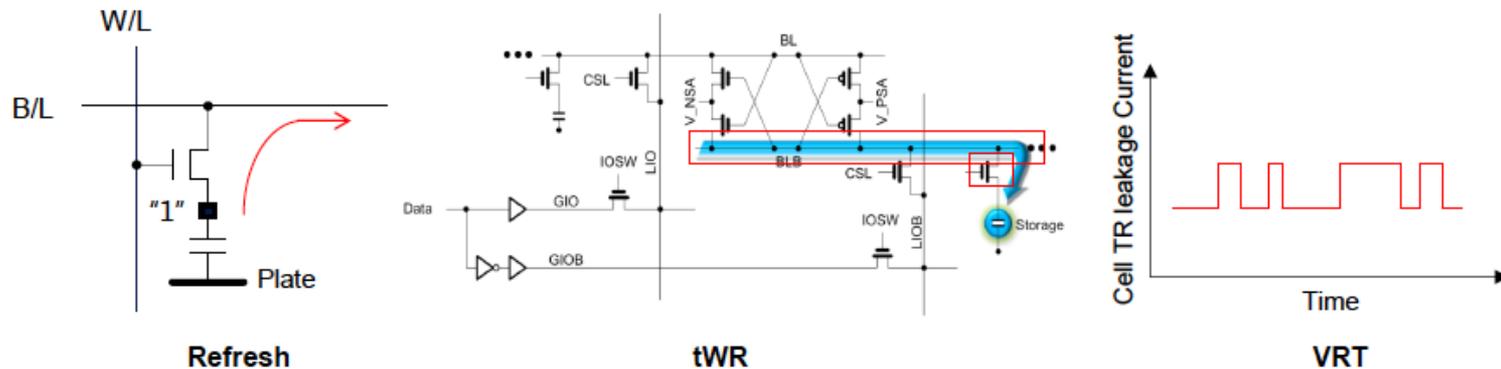
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance
- Leakage current of cell access transistors increasing

### ❖ 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



# Call for Intelligent Memory Controllers

## DRAM Process Scaling Challenges

### ❖ Refresh

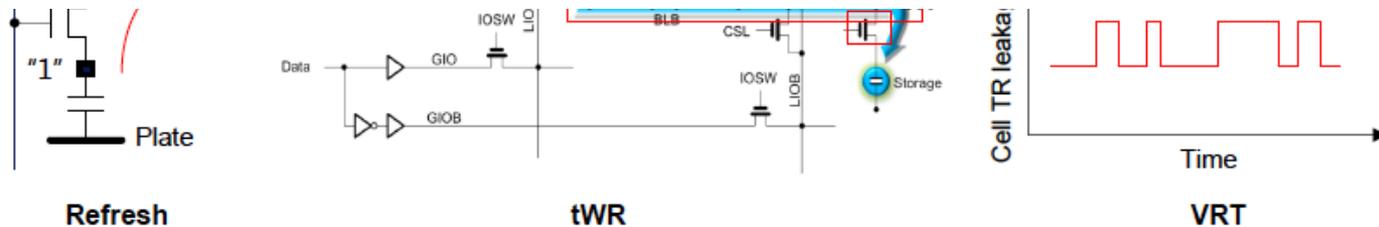
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

THE MEMORY FORUM 2014

# Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling

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

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



# Agenda

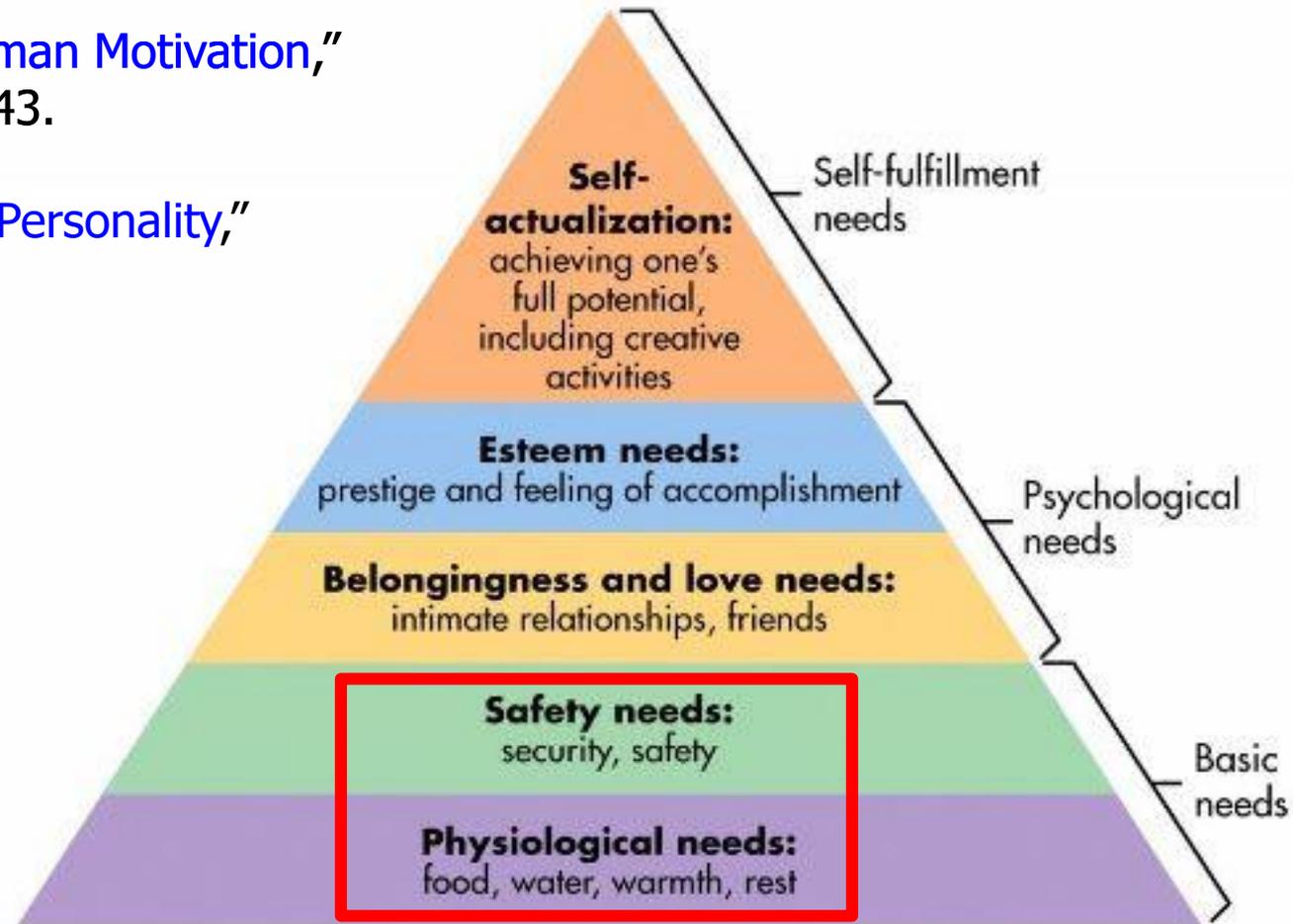
---

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
  - Bottom Up: Push from Circuits and Devices
  - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips
  - Exploiting 3D-Stacked Memory
- How to Enable Adoption of Processing in Memory
- Conclusion

# Maslow's (Human) Hierarchy of Needs

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

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



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

# How Reliable/Secure/Safe is This Bridge?

---



# Collapse of the “Galloping Gertie”

---



# How Secure Are These People?

---

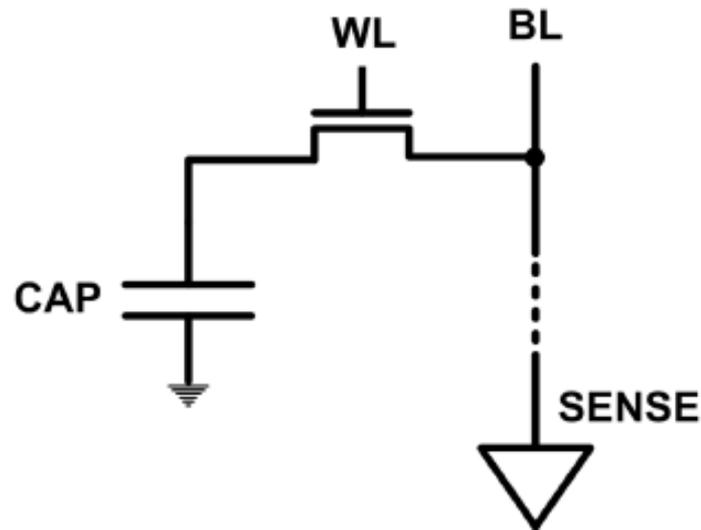


**Security is about preventing unforeseen consequences**

# The DRAM Scaling Problem

---

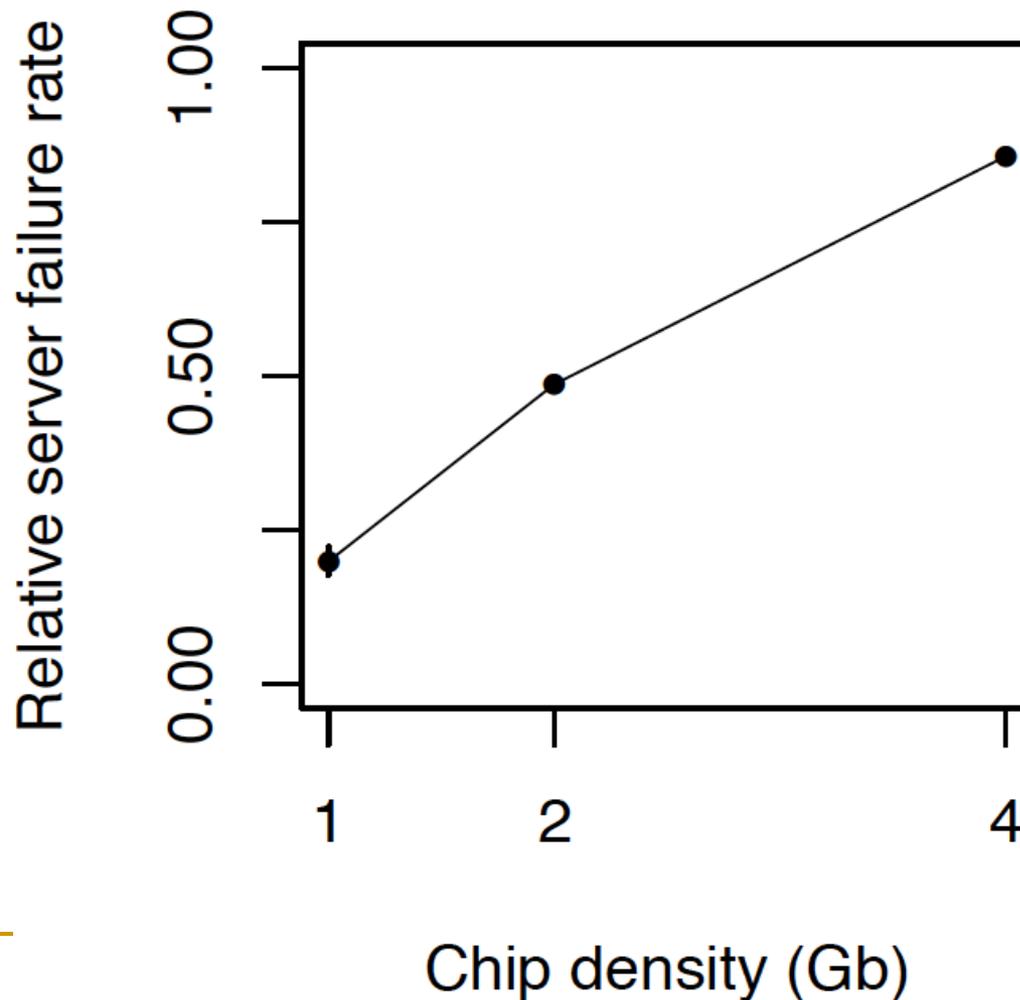
- DRAM stores charge in a capacitor (charge-based memory)
  - Capacitor must be large enough for reliable sensing
  - Access transistor should be large enough for low leakage and high retention time
  - Scaling beyond 40-35nm (2013) is challenging [ITRS, 2009]



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

# As Memory Scales, It Becomes Unreliable

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



*Intuition:  
quadratic  
increase  
in  
capacity*

# Large-Scale Failure Analysis of DRAM Chips

---

- Analysis and modeling of memory errors found in all of Facebook's server fleet
- Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu, **"Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field"** *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Rio de Janeiro, Brazil, June 2015.  
[[Slides \(pptx\)](#)] [[pdf](#)] [[DRAM Error Model](#)]

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

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

# Infrastructures to Understand Such Issues



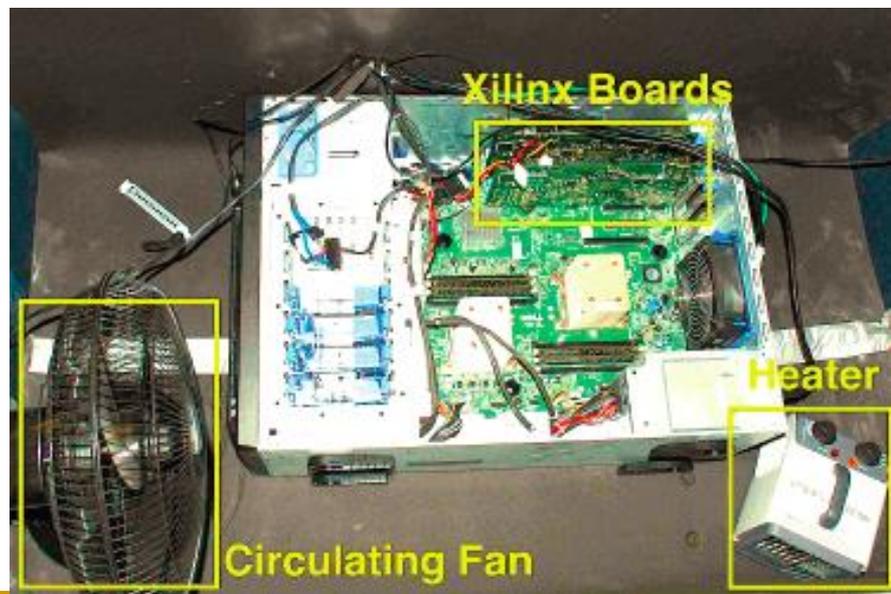
An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

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

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

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

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



# Infrastructures to Understand Such Issues

Temperature  
Controller

FPGAs

Heater

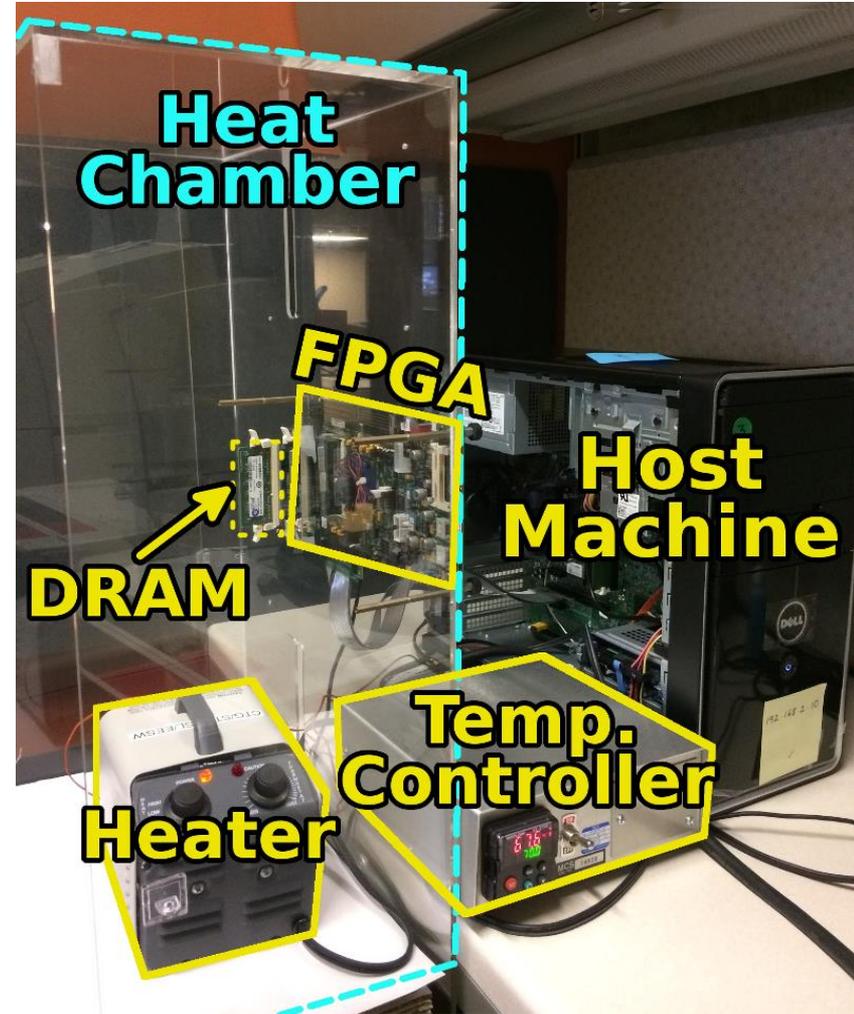
FPGAs

PC



# SoftMC: Open Source DRAM Infrastructure

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



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

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

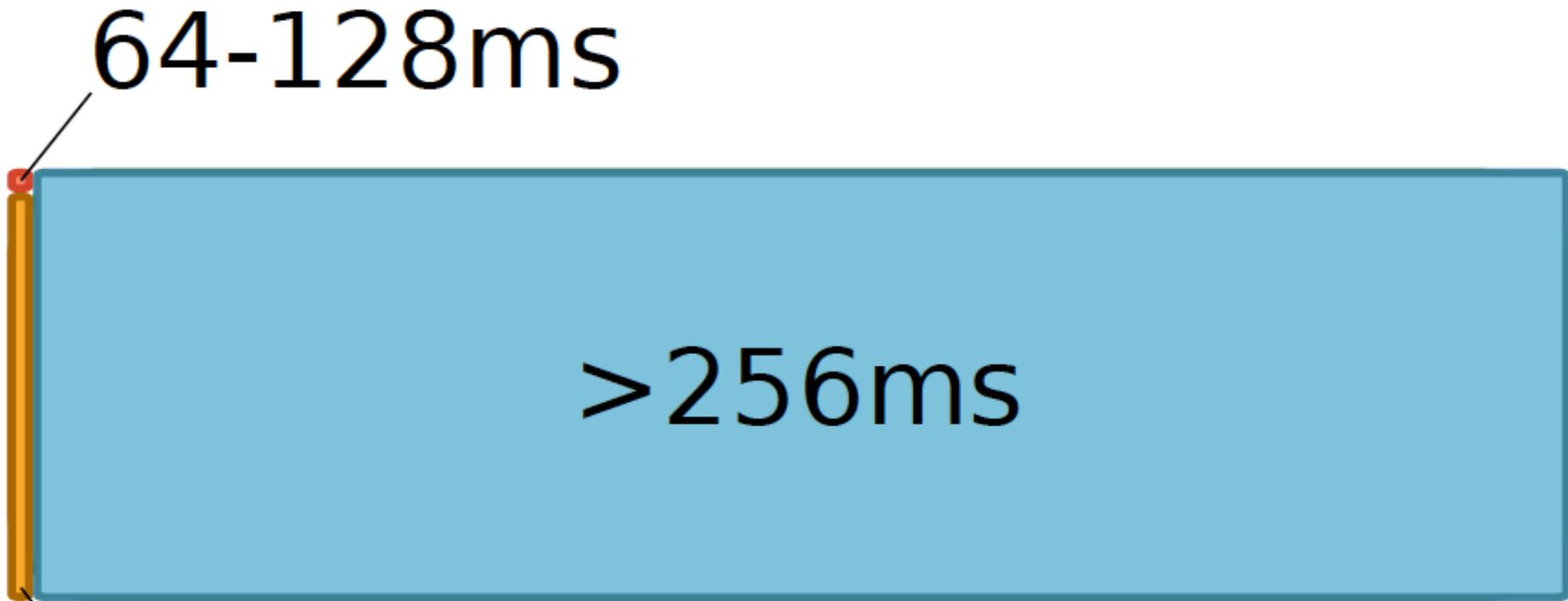
Hasan Hassan<sup>1,2,3</sup> Nandita Vijaykumar<sup>3</sup> Samira Khan<sup>4,3</sup> Saugata Ghose<sup>3</sup> Kevin Chang<sup>3</sup>  
Gennady Pekhimenko<sup>5,3</sup> Donghyuk Lee<sup>6,3</sup> Oguz Ergin<sup>2</sup> Onur Mutlu<sup>1,3</sup>

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

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

---

- Retention Time Profile of DRAM looks like this:



**Location** dependent  
**Stored value pattern** dependent  
**Time** dependent

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

---

One can  
predictably induce errors  
in most DRAM memory chips

# DRAM RowHammer

---

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

**WIRED**

Forget Software—Now Hackers Are Exploiting Physics

BUSINESS

CULTURE

DESIGN

GEAR

SCIENCE

ANDY GREENBERG SECURITY 08.31.16 7:00 AM

SHARE



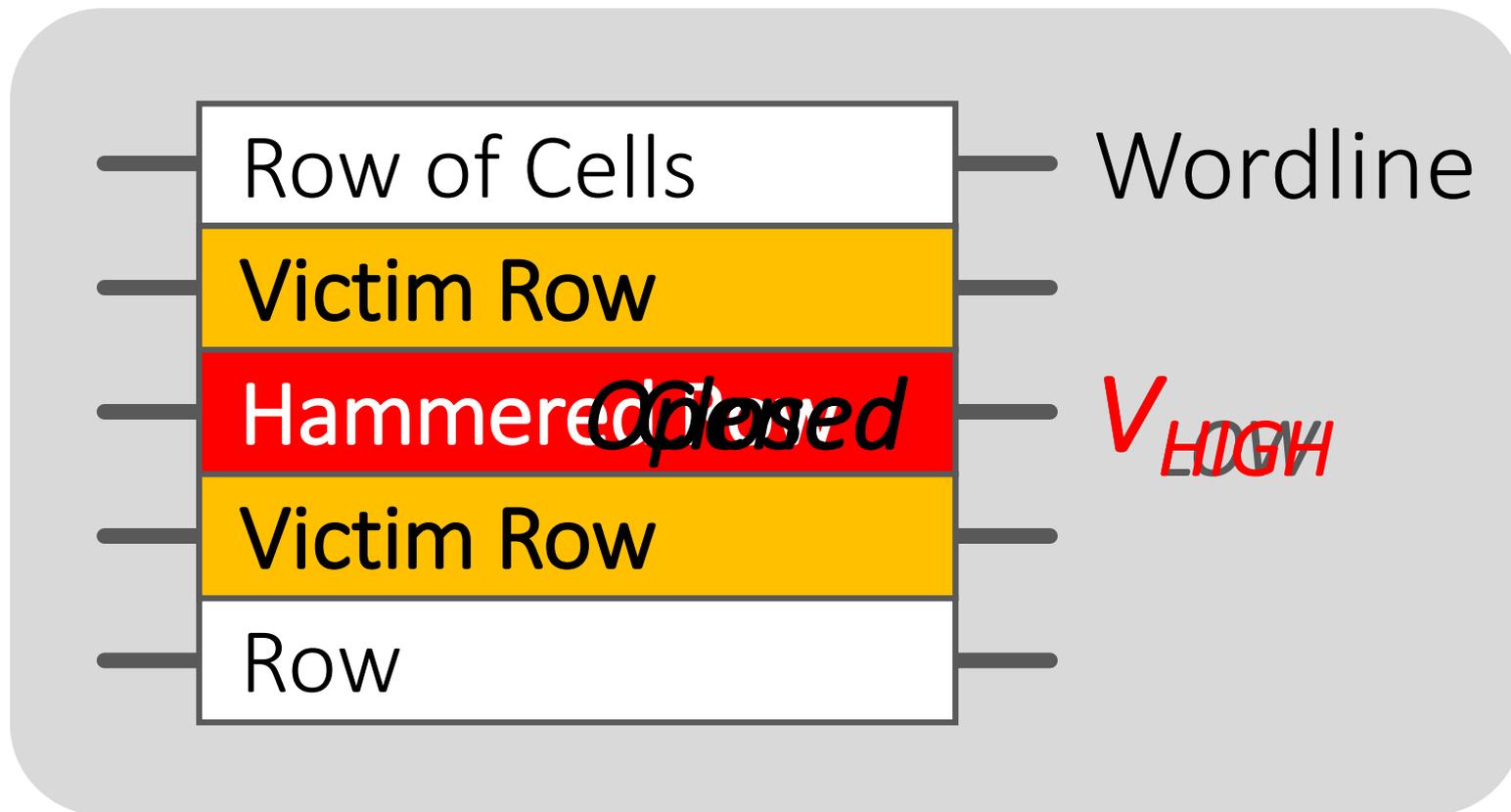
SHARE  
18276



TWEET

# FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS

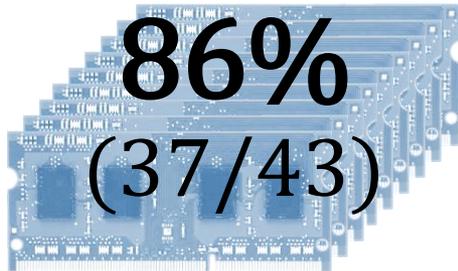
# Modern DRAM is Prone to Disturbance Errors



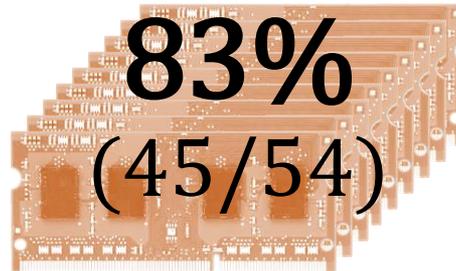
Repeatedly reading a row enough times (before memory gets refreshed) induces **disturbance errors** in **adjacent rows** in **most real DRAM chips you can buy today**

# Most DRAM Modules Are Vulnerable

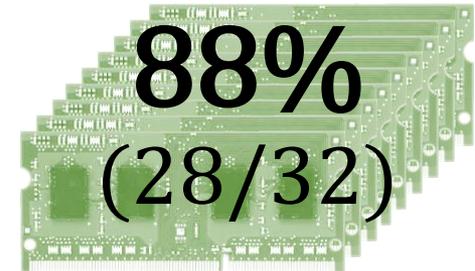
A company



B company



C company

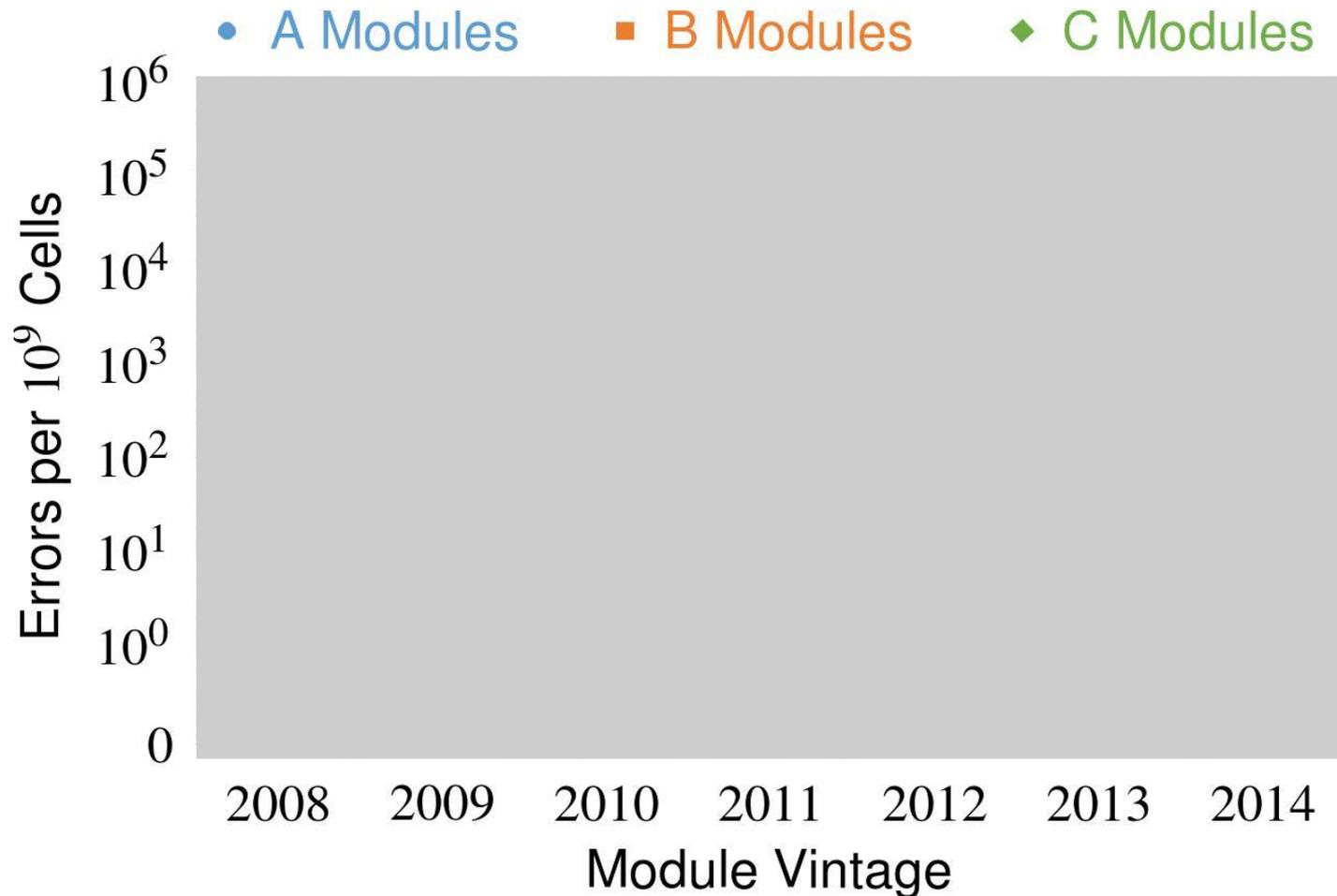


Up to  
 $1.0 \times 10^7$   
errors

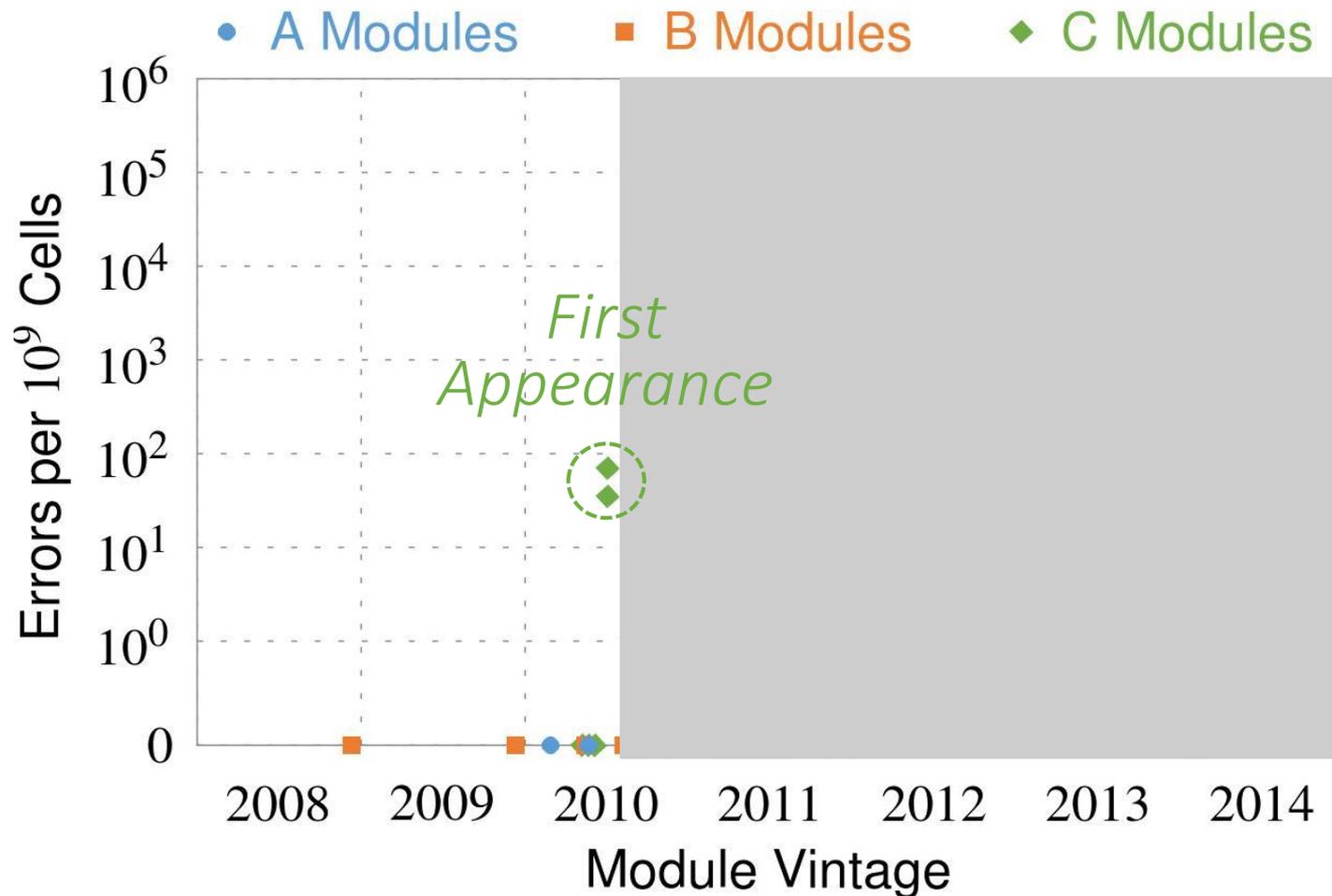
Up to  
 $2.7 \times 10^6$   
errors

Up to  
 $3.3 \times 10^5$   
errors

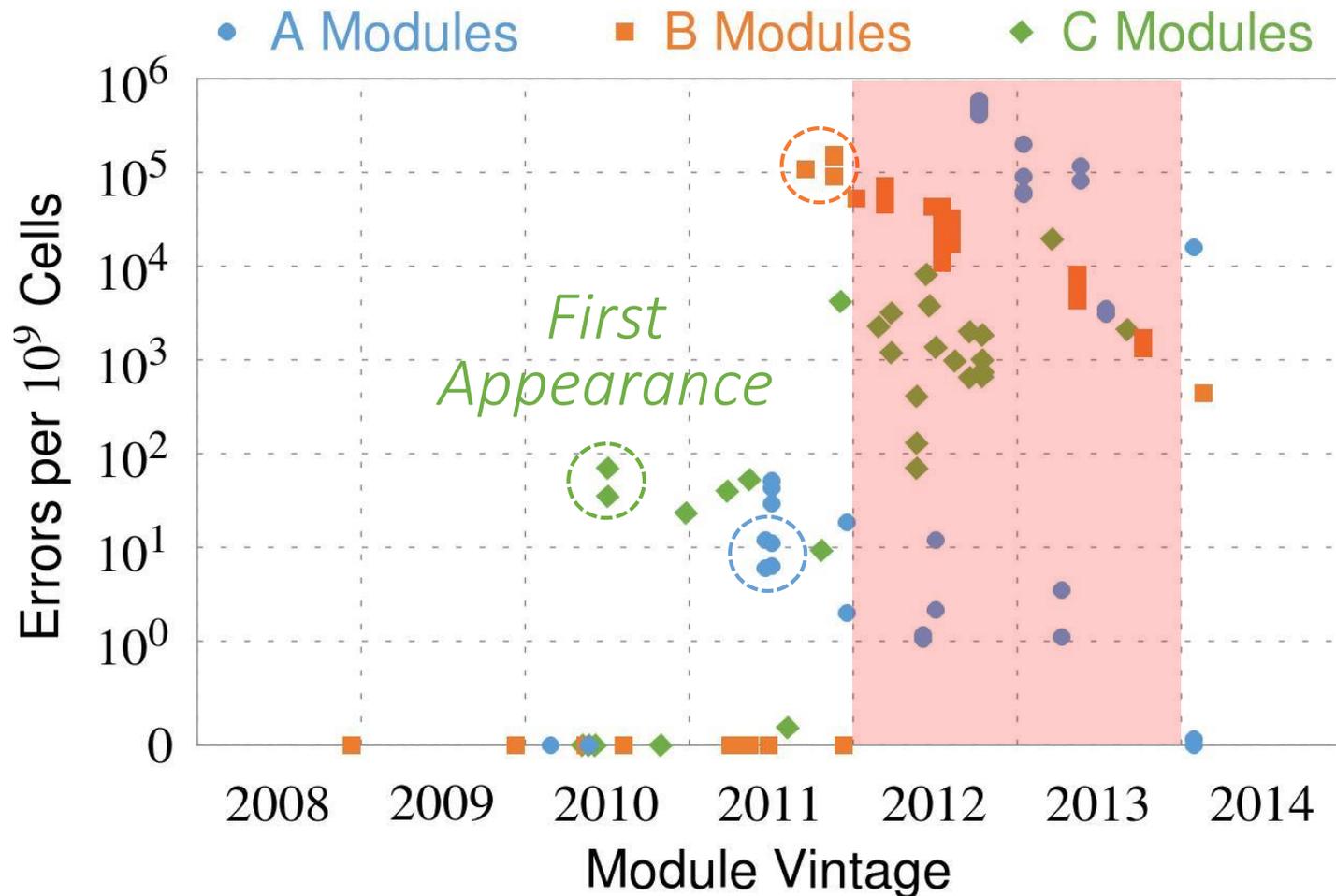
# Recent DRAM Is More Vulnerable



# Recent DRAM Is More Vulnerable

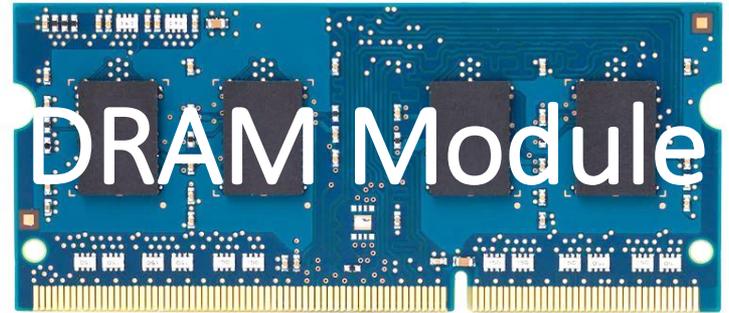
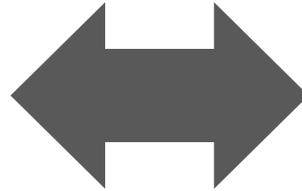
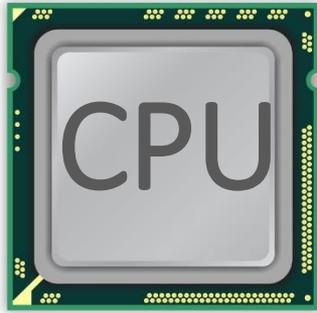


# Recent DRAM Is More Vulnerable

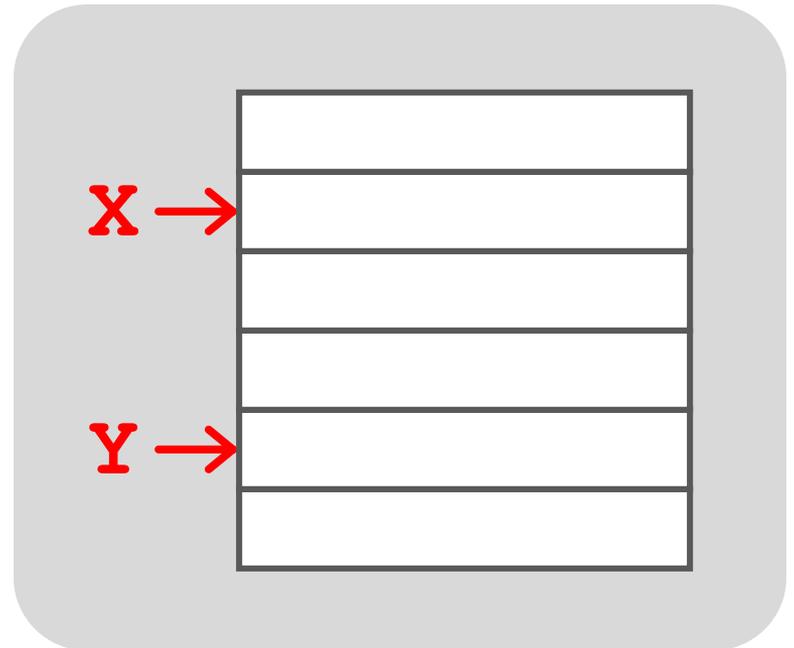


*All modules from 2012-2013 are vulnerable*

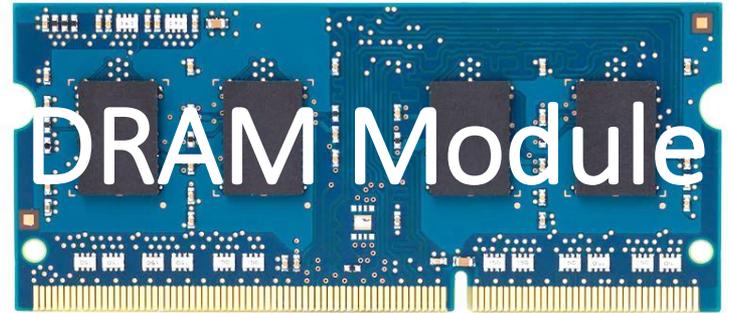
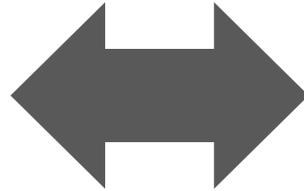
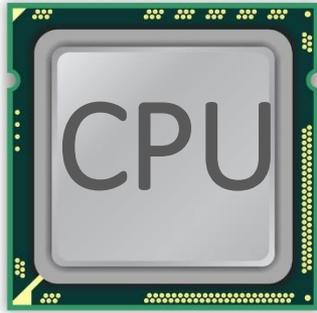
# A Simple Program Can Induce Many Errors



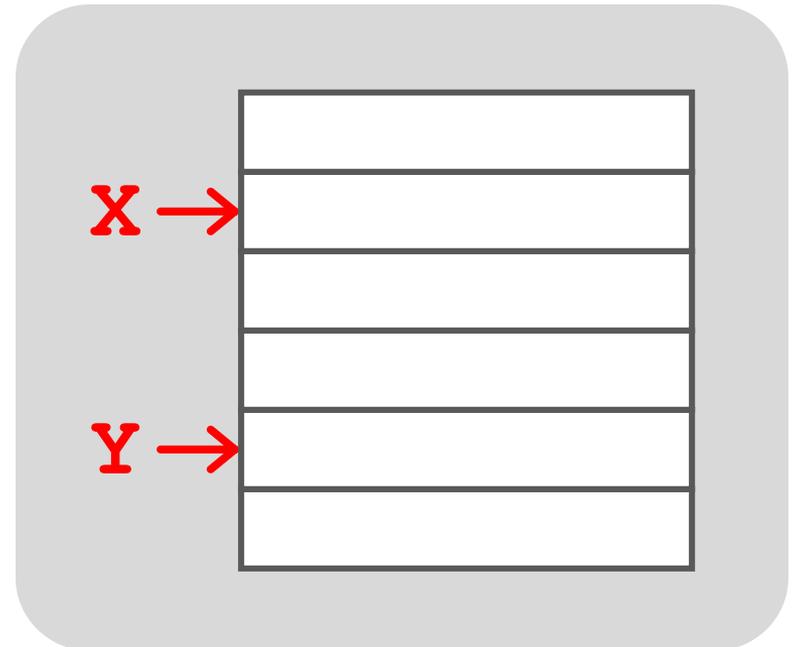
```
loop:  
  mov  (X), %eax  
  mov  (Y), %ebx  
  clflush (X)  
  clflush (Y)  
  mfence  
  jmp  loop
```



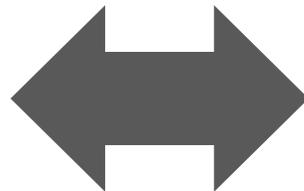
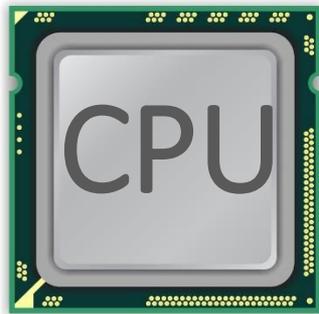
# A Simple Program Can Induce Many Errors



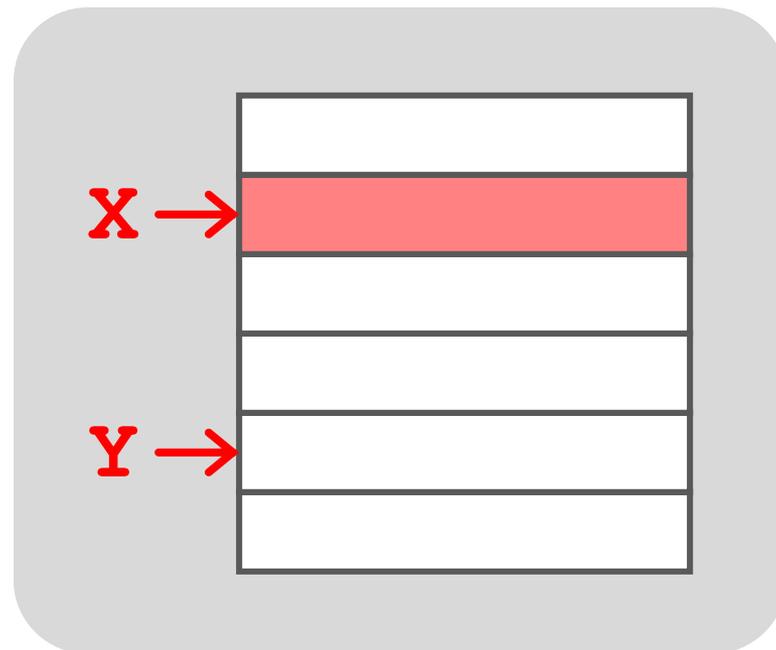
1. Avoid *cache hits*
  - Flush **X** from cache
2. Avoid *row hits* to **X**
  - Read **Y** in another row



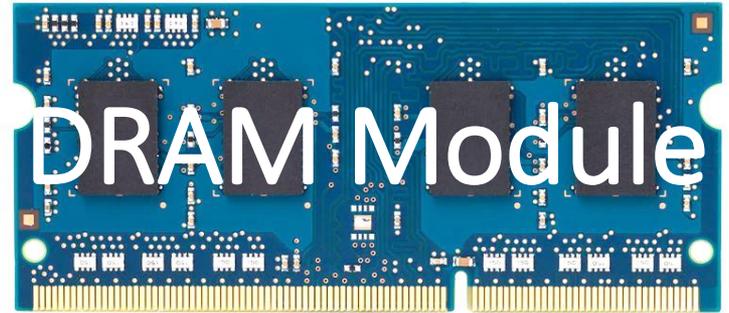
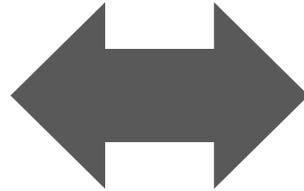
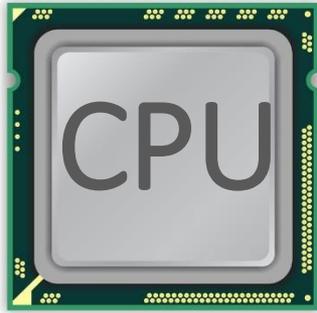
# A Simple Program Can Induce Many Errors



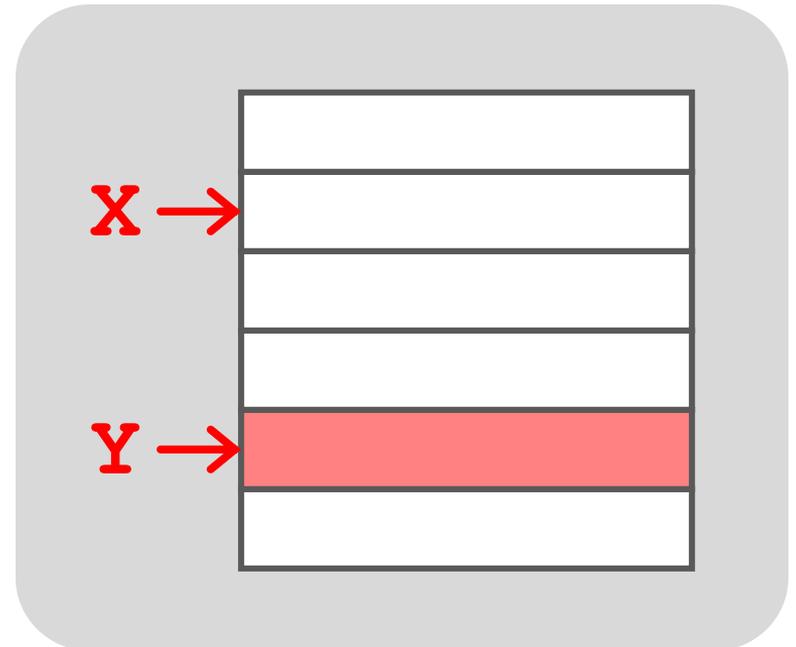
```
loop:  
  mov  (X), %eax  
  mov  (Y), %ebx  
  clflush (X)  
  clflush (Y)  
  mfence  
  jmp  loop
```



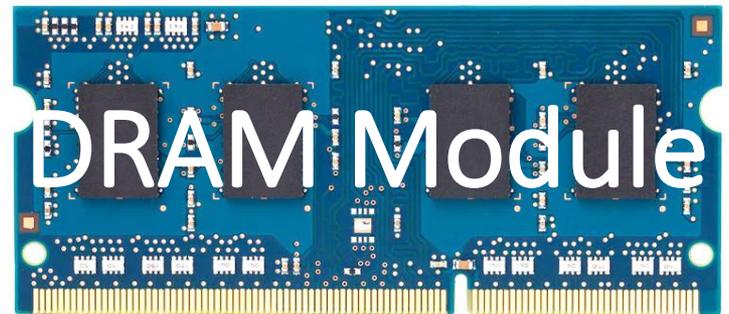
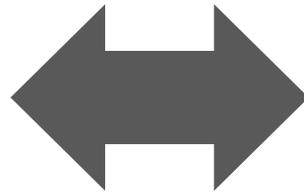
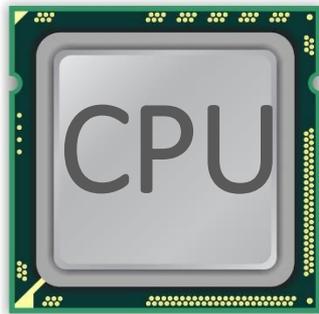
# A Simple Program Can Induce Many Errors



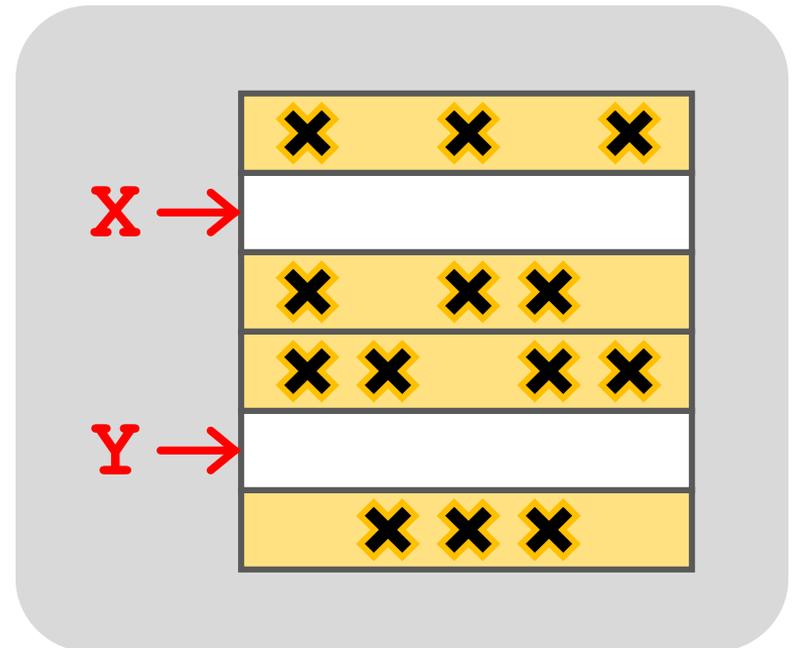
```
loop:  
  mov  (X), %eax  
  mov  (Y), %ebx  
  clflush (X)  
  clflush (Y)  
  mfence  
  jmp  loop
```



# A Simple Program Can Induce Many Errors



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



# Observed Errors in Real Systems

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

**A real reliability & security issue**

# One Can Take Over an Otherwise-Secure System

---

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

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

## Project Zero

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

News and updates from the Project Zero team at Google

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

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

# RowHammer Security Attack Example

---

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

# Security Implications



# 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 (I)

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

www.iaik.tugraz.at

Not there yet, but ...



ROOT privileges for web apps!

29

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



GATED  
COMMUNITIES

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

# More Security Implications (II)

**"Can gain control of a smart phone deterministically"**



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

# More Security Implications (III)

- Using an integrated GPU in a mobile system to remotely escalate privilege via the WebGL interface



BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

"GRAND PWNING UNIT" —

## Drive-by Rowhammer attack uses GPU to compromise an Android phone

JavaScript based GLitch pwns browsers by flipping bits inside memory chips.

DAN GOODIN - 5/3/2018, 12:00 PM

## Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU

Pietro Frigo  
Vrije Universiteit  
Amsterdam  
p.frigo@vu.nl

Cristiano Giuffrida  
Vrije Universiteit  
Amsterdam  
giuffrida@cs.vu.nl

Herbert Bos  
Vrije Universiteit  
Amsterdam  
herbertb@cs.vu.nl

Kaveh Razavi  
Vrije Universiteit  
Amsterdam  
kaveh@cs.vu.nl

# More Security Implications (IV)

---

- Rowhammer over RDMA (I)



BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

THROWHAMMER —

## Packets over a LAN are all it takes to trigger serious Rowhammer bit flips

The bar for exploiting potentially serious DDR weakness keeps getting lower.

DAN GOODIN - 5/10/2018, 5:26 PM

### Throwhammer: Rowhammer Attacks over the Network and Defenses

Andrei Tatar  
*VU Amsterdam*

Radhesh Krishnan  
*VU Amsterdam*

Elias Athanasopoulos  
*University of Cyprus*

Cristiano Giuffrida  
*VU Amsterdam*

Herbert Bos  
*VU Amsterdam*

Kaveh Razavi  
*VU Amsterdam*

# More Security Implications (V)

- Rowhammer over RDMA (II)



## Nethammer—Exploiting DRAM Rowhammer Bug Through Network Requests



### **Nethammer: Inducing Rowhammer Faults through Network Requests**

Moritz Lipp  
Graz University of Technology

Daniel Gruss  
Graz University of Technology

Misiker Tadesse Aga  
University of Michigan

Clémentine Maurice  
Univ Rennes, CNRS, IRISA

Michael Schwarz  
Graz University of Technology

Lukas Raab  
Graz University of Technology

Lukas Lamster  
Graz University of Technology

# More Security Implications?

---



# Apple's Patch for RowHammer

---

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

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

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

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

CVE-ID

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

HP, Lenovo, and other vendors released similar patches

---

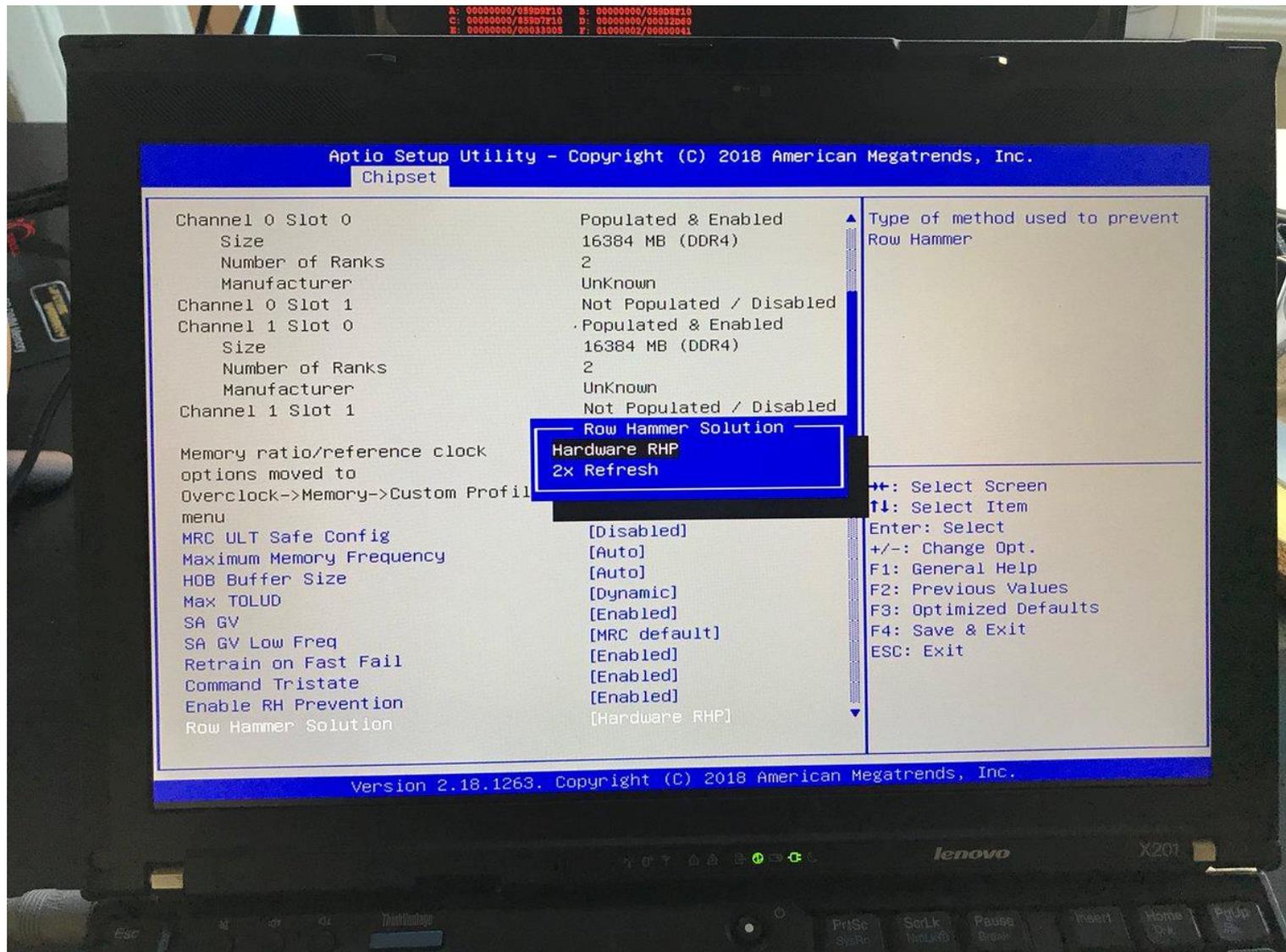
# Our Solution to RowHammer

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

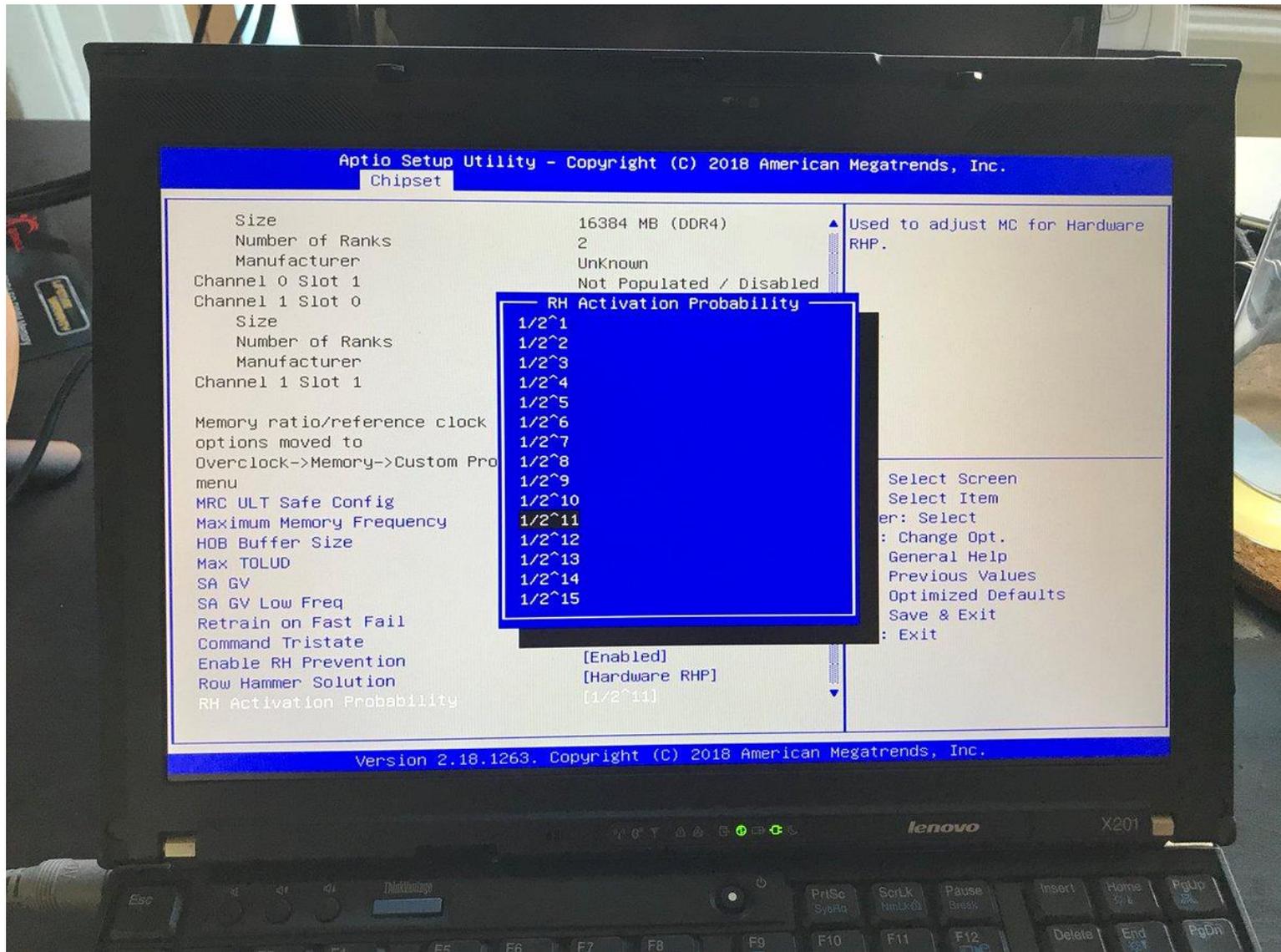
# Advantages of PARA

- *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*

# Probabilistic Activation in Real Life (I)



# Probabilistic Activation in Real Life (II)



# 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<sup>1</sup> Ross Daly\* Jeremie Kim<sup>1</sup> Chris Fallin\* Ji Hye Lee<sup>1</sup>  
Donghyuk Lee<sup>1</sup> Chris Wilkerson<sup>2</sup> Konrad Lai Onur Mutlu<sup>1</sup>

<sup>1</sup>Carnegie Mellon University      <sup>2</sup>Intel Labs

# Future of Memory Reliability

---

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

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

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

# Industry Is Writing Papers About It, Too

## DRAM Process Scaling Challenges

### ❖ Refresh

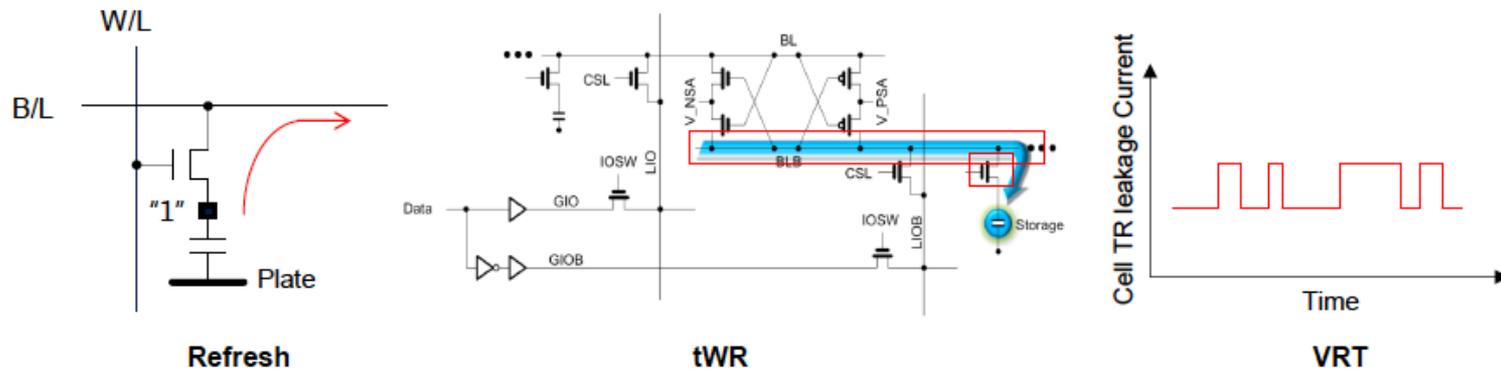
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance
- Leakage current of cell access transistors increasing

### ❖ $t_{WR}$

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

### ❖ VRT

- Occurring more frequently with cell capacitance decreasing



# Call for Intelligent Memory Controllers

## DRAM Process Scaling Challenges

### ❖ Refresh

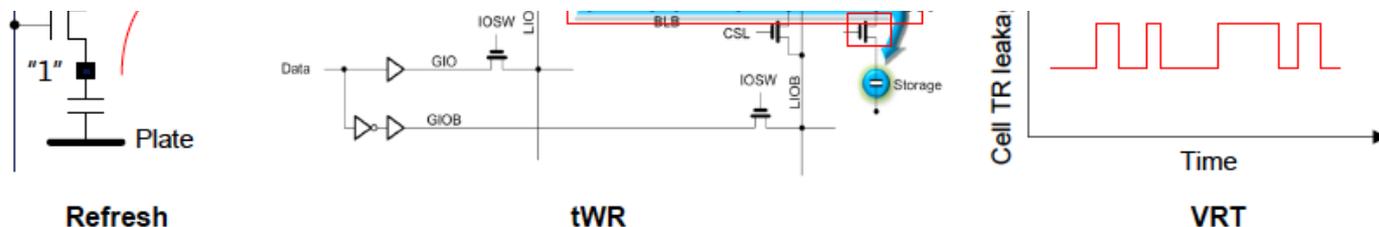
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

THE MEMORY FORUM 2014

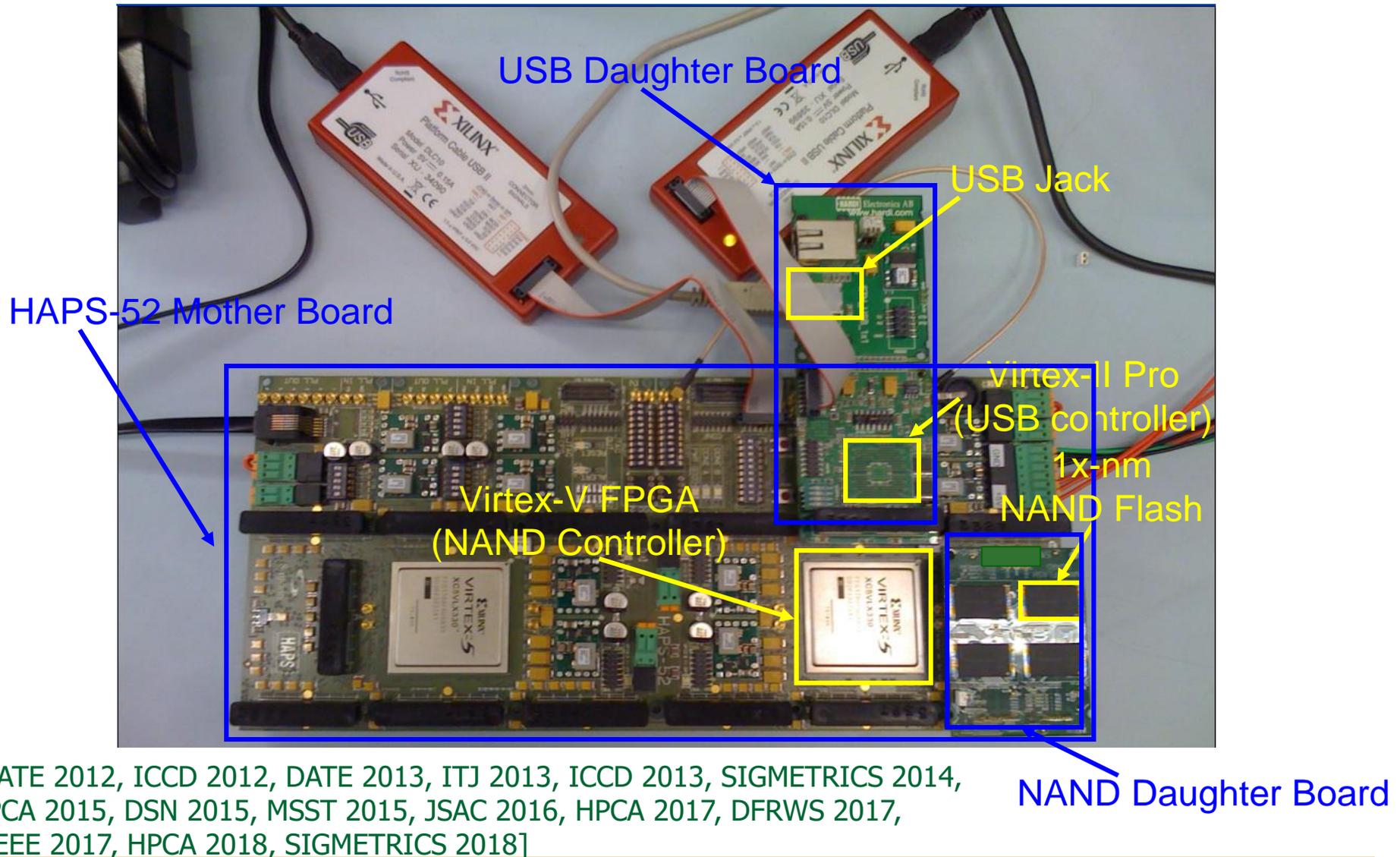
# Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling

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

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



# Aside: Intelligent Controller for NAND Flash



[DATE 2012, ICCD 2012, DATE 2013, ITJ 2013, ICCD 2013, SIGMETRICS 2014, HPCA 2015, DSN 2015, MSST 2015, JSAC 2016, HPCA 2017, DFRWS 2017, PIEEE 2017, HPCA 2018, SIGMETRICS 2018]

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



*Proceedings of the IEEE, Sept. 2017*



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

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

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

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

## Main Memory Needs Intelligent Controllers

# Agenda

---

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
  - Bottom Up: Push from Circuits and Devices
  - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips
  - Exploiting 3D-Stacked Memory
- How to Enable Adoption of Processing in Memory
- Conclusion

# Three Key Systems Trends

---

## 1. Data access is a major bottleneck

- Applications are increasingly data hungry

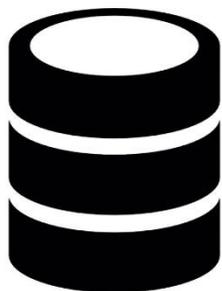
## 2. Energy consumption is a key limiter

## 3. Data movement energy dominates compute

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

# The Need for More Memory Performance

---



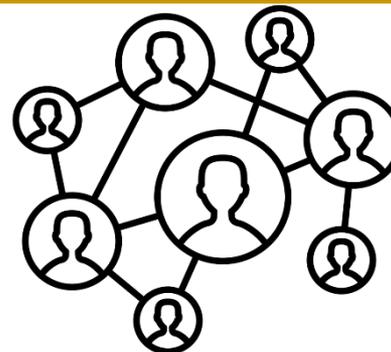
## In-memory Databases

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



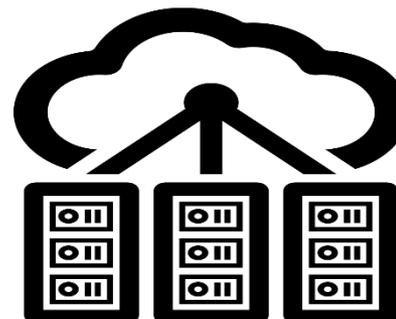
## In-Memory Data Analytics

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



## Graph/Tree Processing

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



## Datacenter Workloads

[Kanev+ (Google), ISCA'15]

# Do We Want This?

---



# Or This?

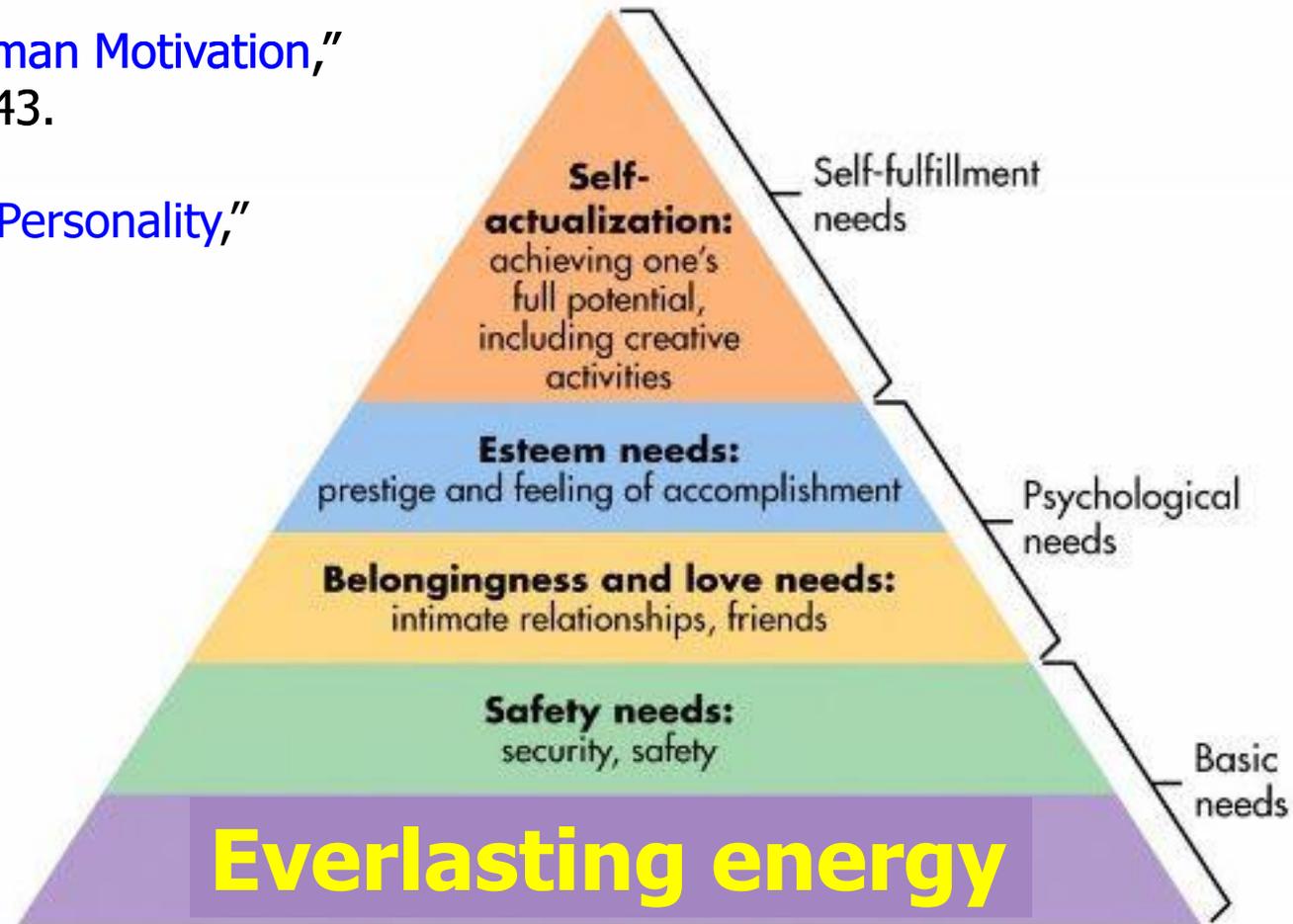
---



# Maslow's (Human) Hierarchy of Needs, Revisited

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

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



# Challenge and Opportunity for Future

---

High Performance,  
Energy Efficient,  
Sustainable

# The Problem

---

Data access is the major performance and energy bottleneck

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

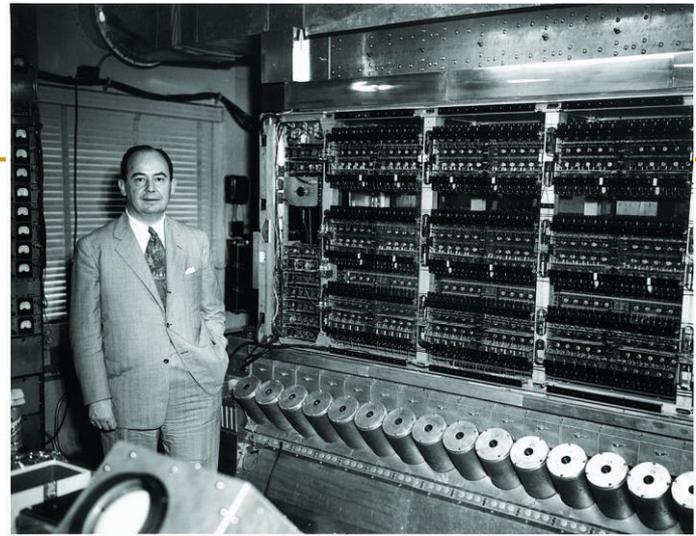
# The Problem

---

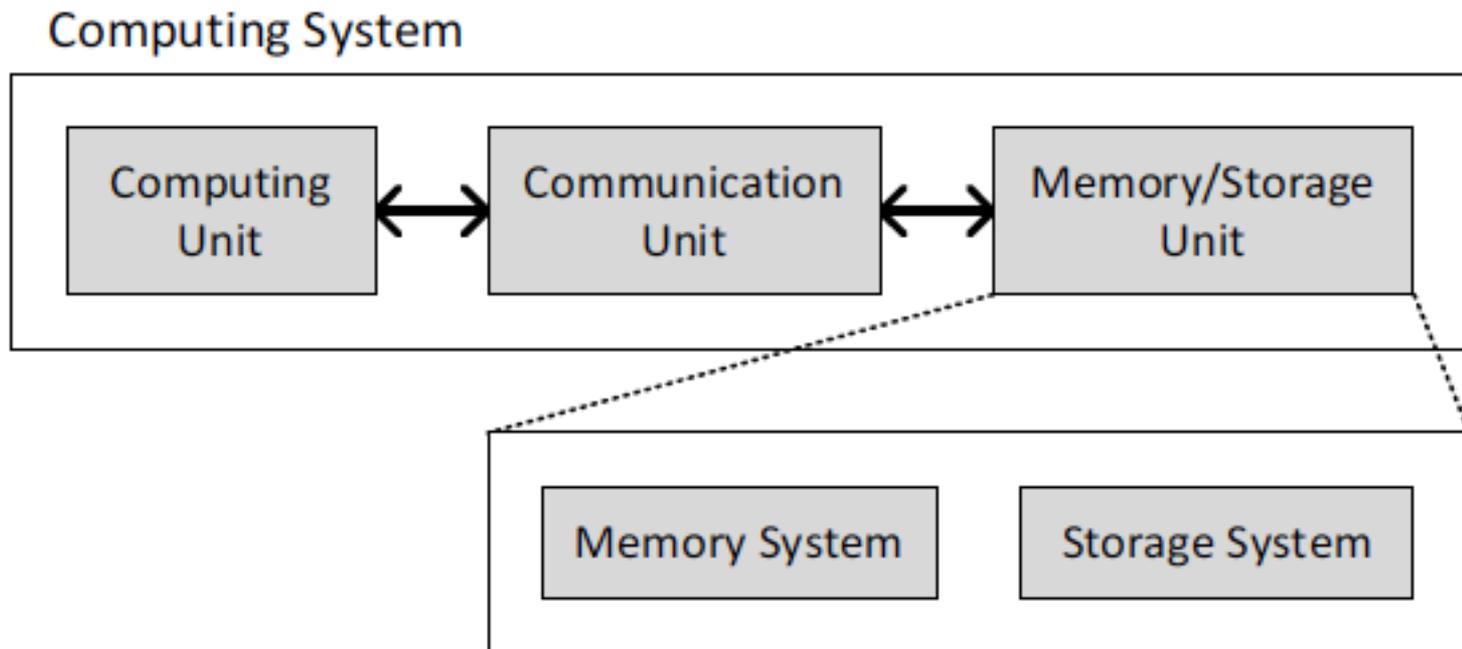
Processing of data  
is performed  
far away from the data

# A Computing System

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

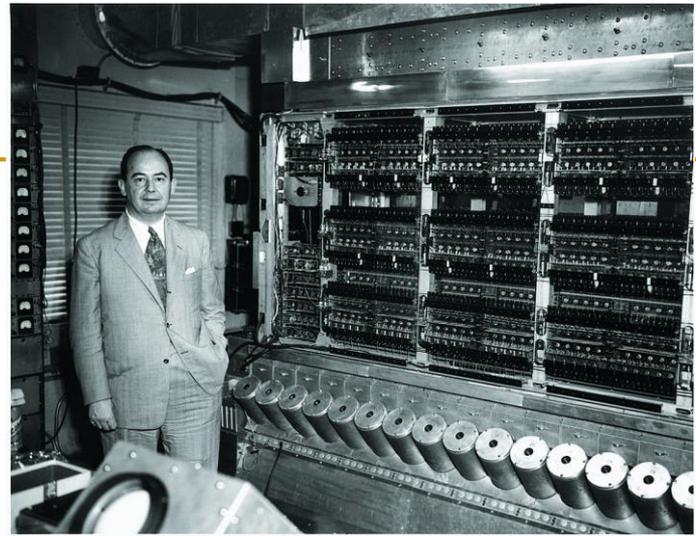


Burks, Goldstein, von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument," 1946.



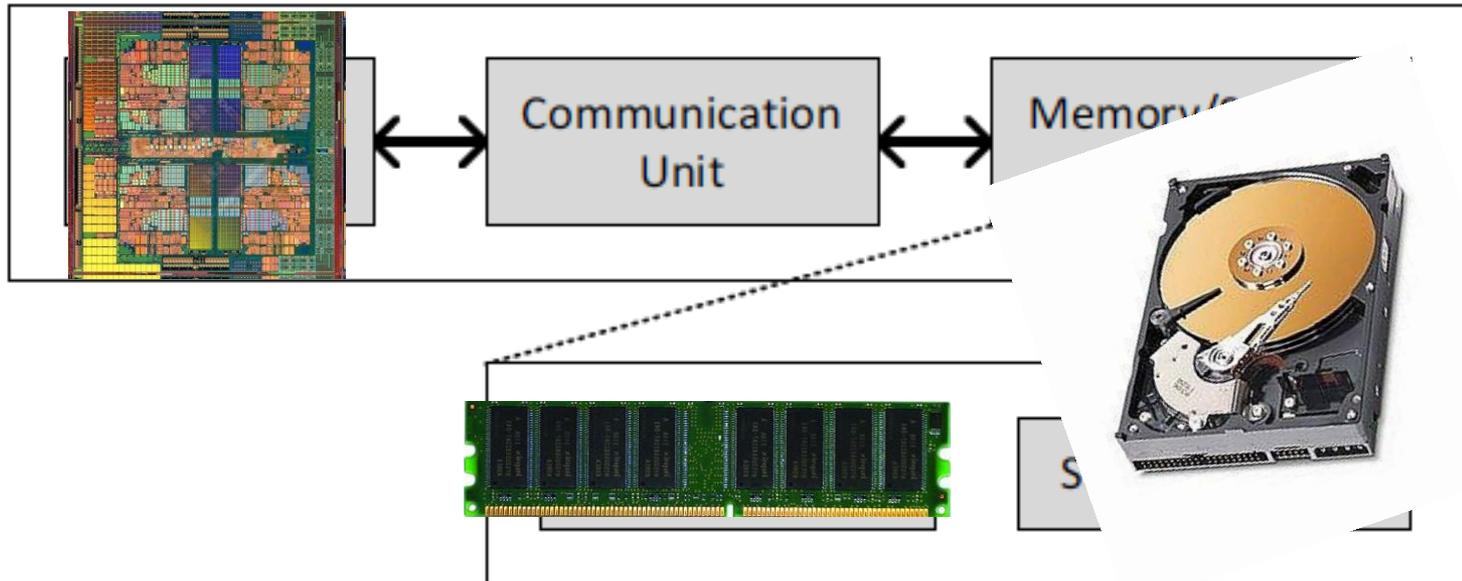
# A Computing System

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



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

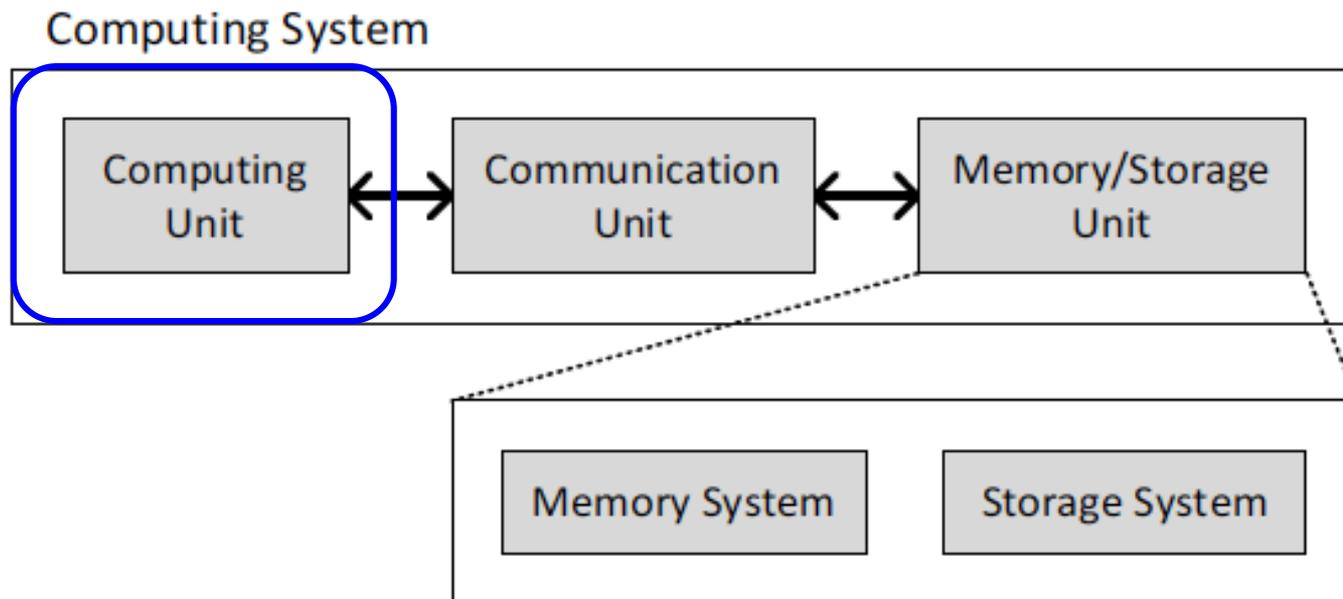
## Computing System



# Today's Computing Systems

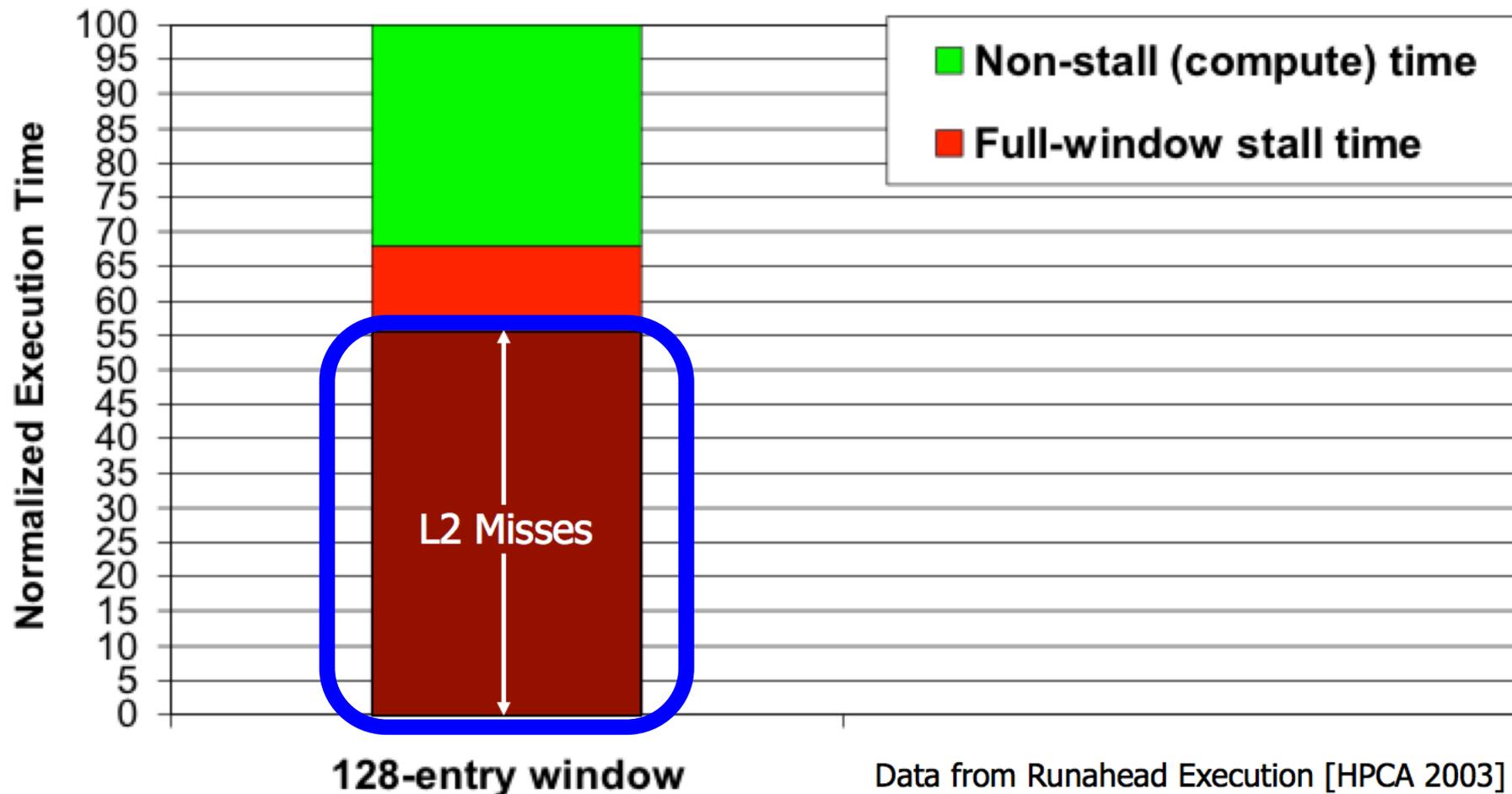
---

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



# Yet ...

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



# The Performance Perspective

---

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

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

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

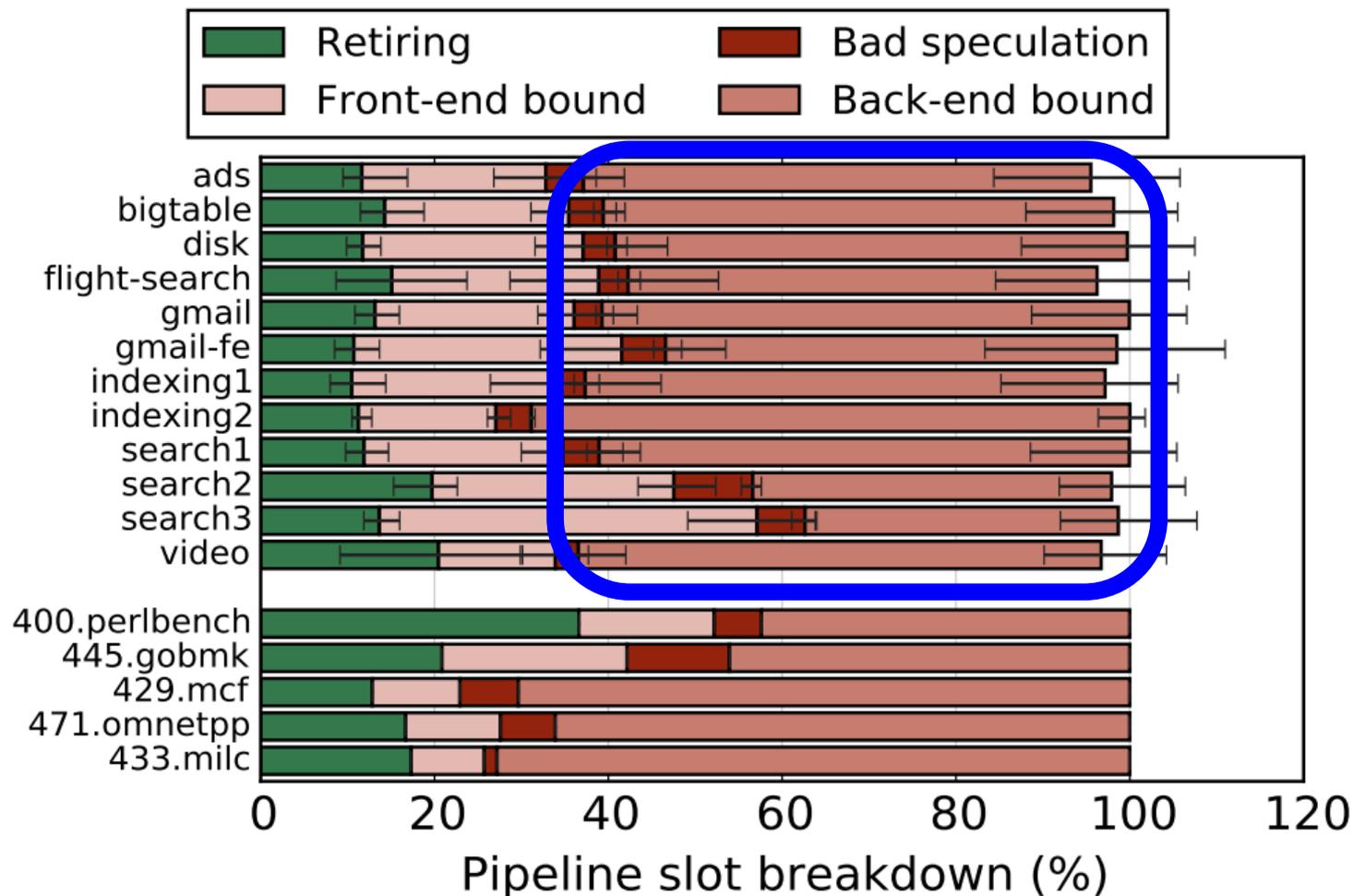
§ECE Department  
The University of Texas at Austin  
{onur,patt}@ece.utexas.edu

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

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

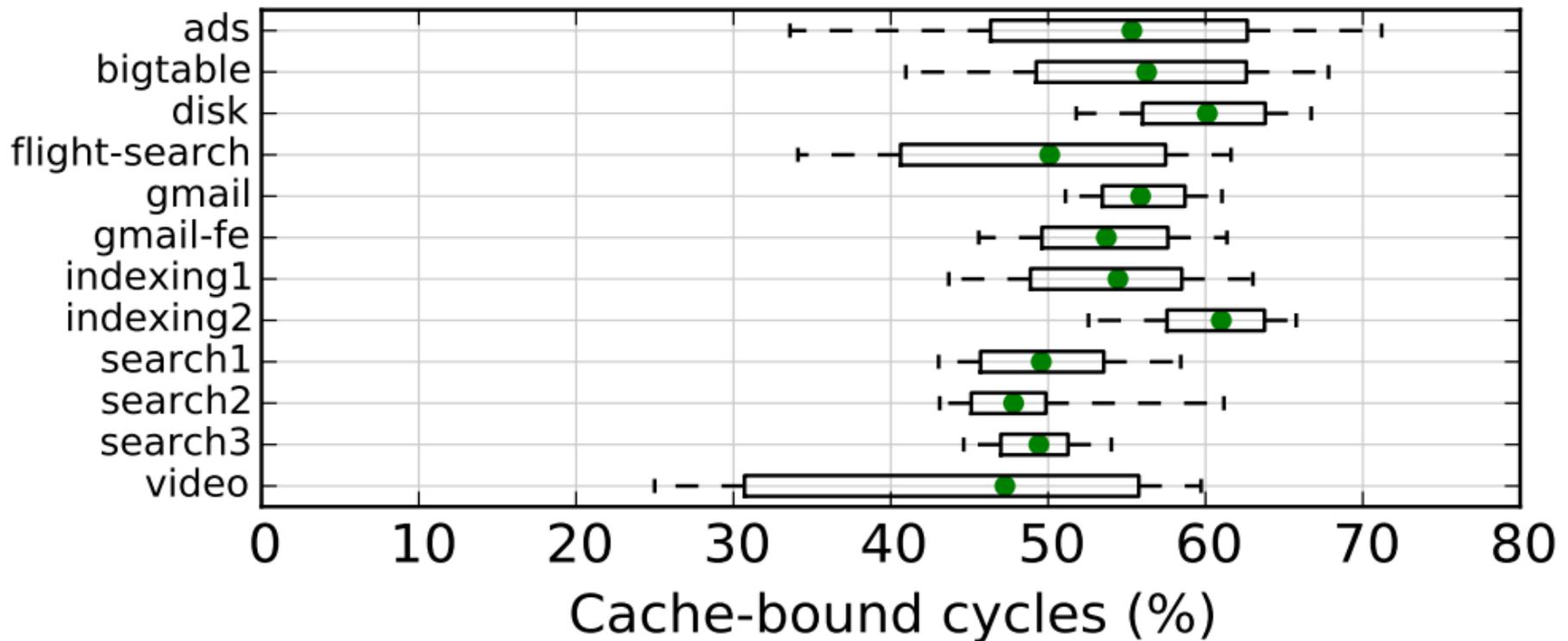
# The Performance Perspective (Today)

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



# The Performance Perspective (Today)

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



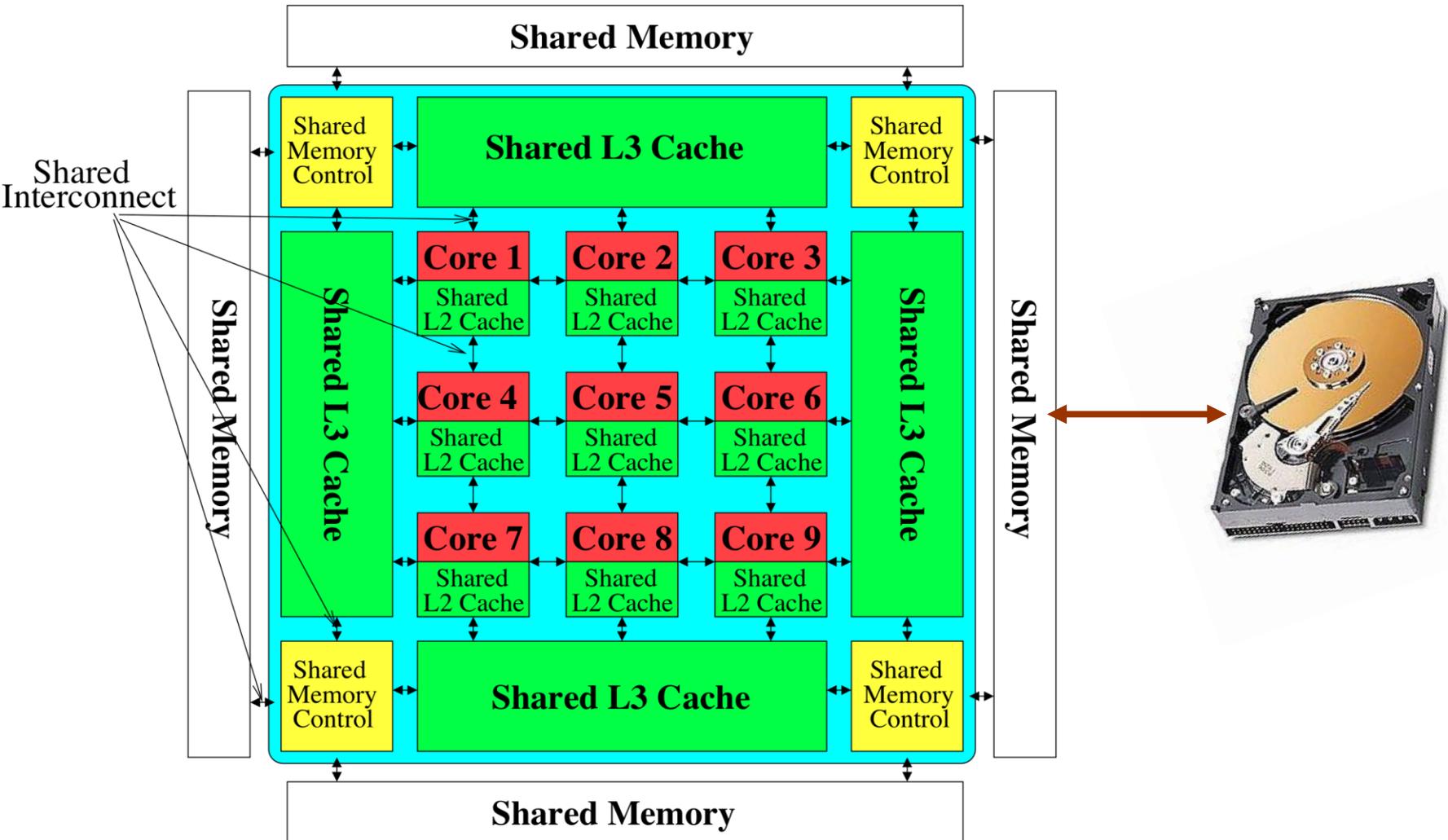
**Figure 11: Half of cycles are spent stalled on caches.**

# 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

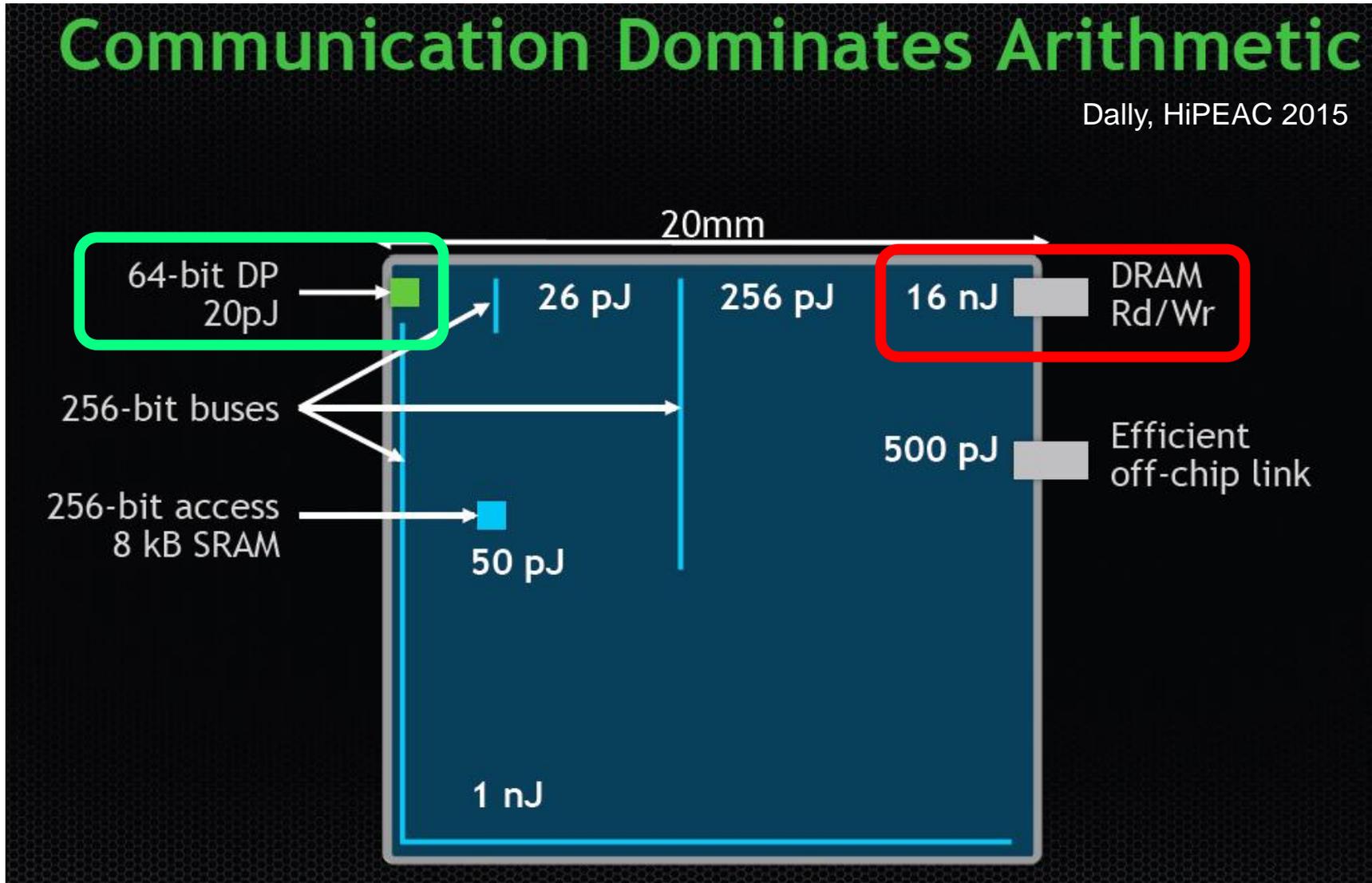


**Most of the system is dedicated to storing and moving data**

# The Energy Perspective

## Communication Dominates Arithmetic

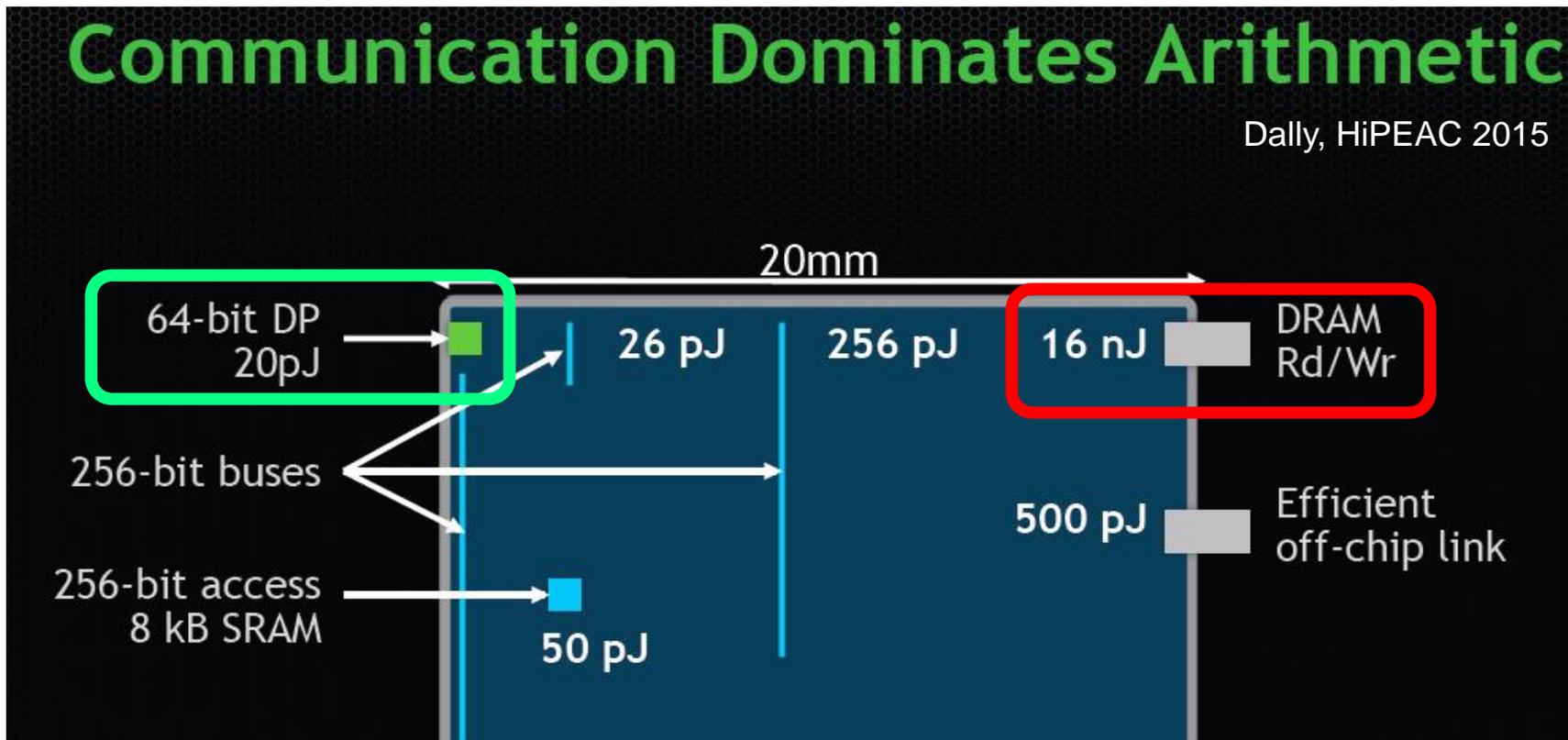
Dally, HiPEAC 2015



# Data Movement vs. Computation Energy

## Communication Dominates Arithmetic

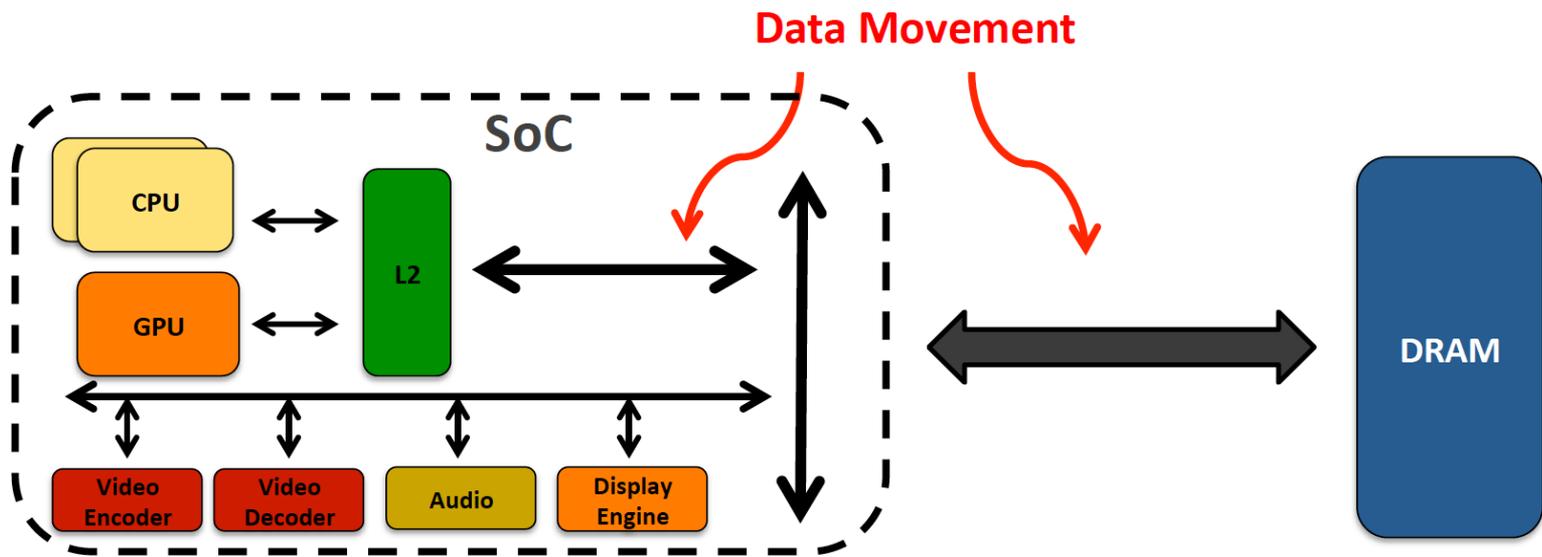
Dally, HiPEAC 2015



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

# Data Movement vs. Computation Energy

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



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

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

# Energy Waste in Mobile Devices

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu, **"Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"** *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Williamsburg, VA, USA, March 2018.

**62.7%** of the total system energy  
is spent on **data movement**

## Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand<sup>1</sup>

Saugata Ghose<sup>1</sup>

Youngsok Kim<sup>2</sup>

Rachata Ausavarungnirun<sup>1</sup>

Eric Shiu<sup>3</sup>

Rahul Thakur<sup>3</sup>

Daehyun Kim<sup>4,3</sup>

Aki Kuusela<sup>3</sup>

Allan Knies<sup>3</sup>

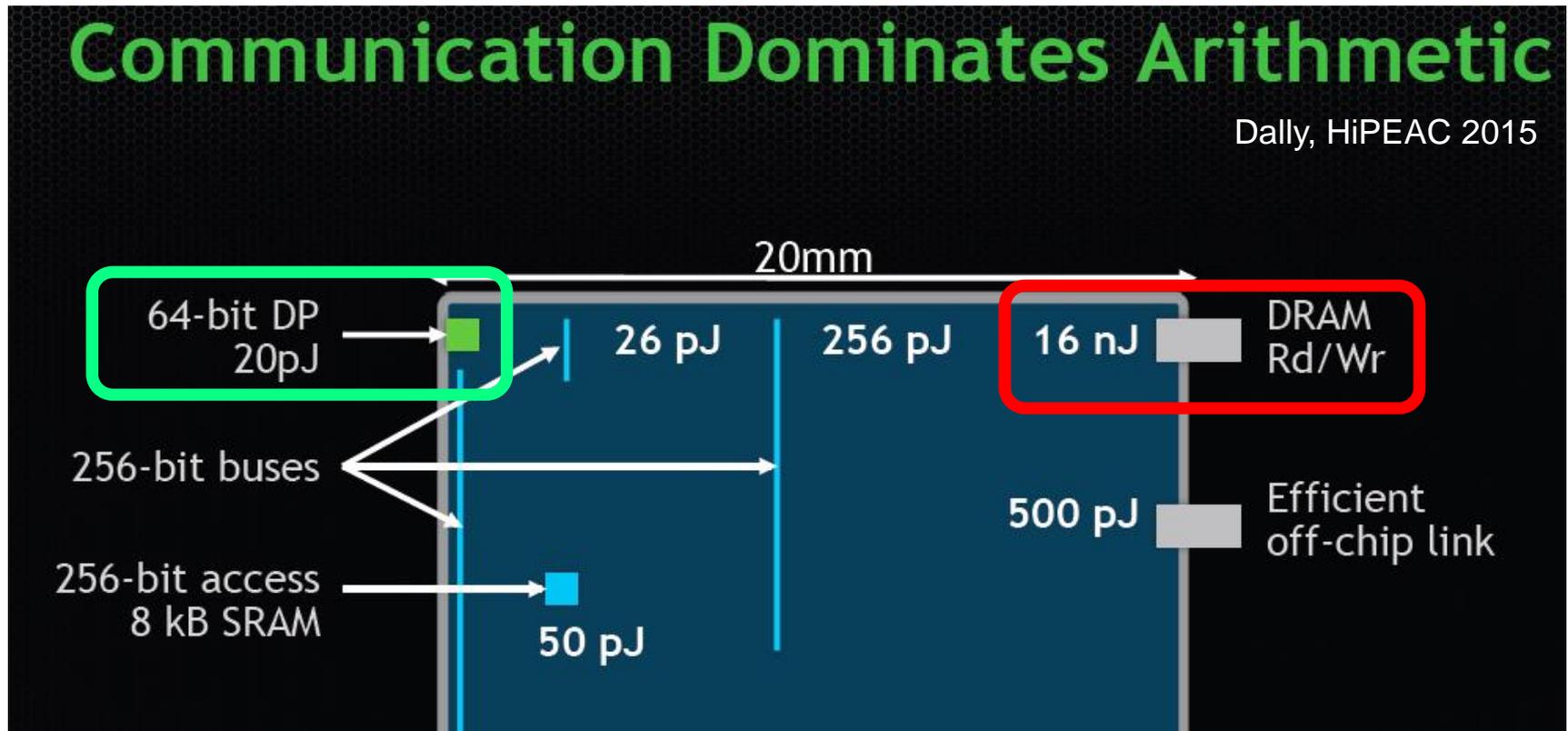
Parthasarathy Ranganathan<sup>3</sup>

Onur Mutlu<sup>5,1</sup>

# We Do Not Want to Move Data!

## Communication Dominates Arithmetic

Dally, HiPEAC 2015



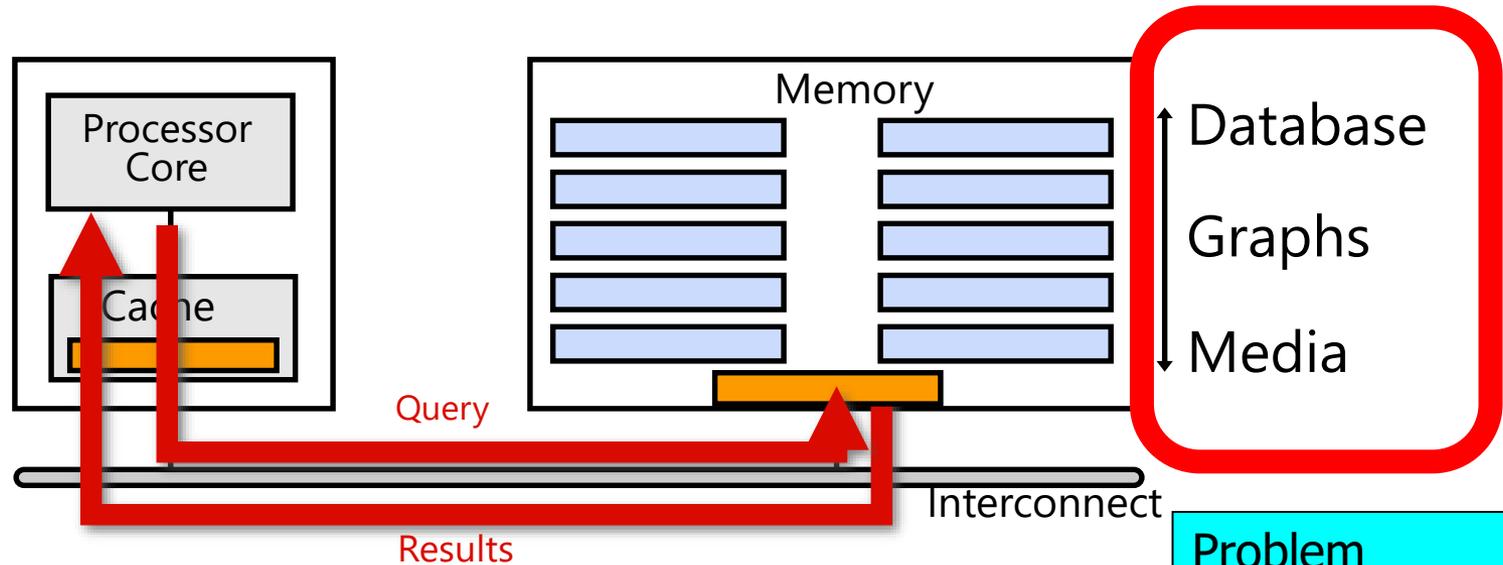
A memory access consumes  $\sim 1000X$  the energy of a complex addition

# We Need A Paradigm Shift To ...

---

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

# Goal: Processing Inside Memory



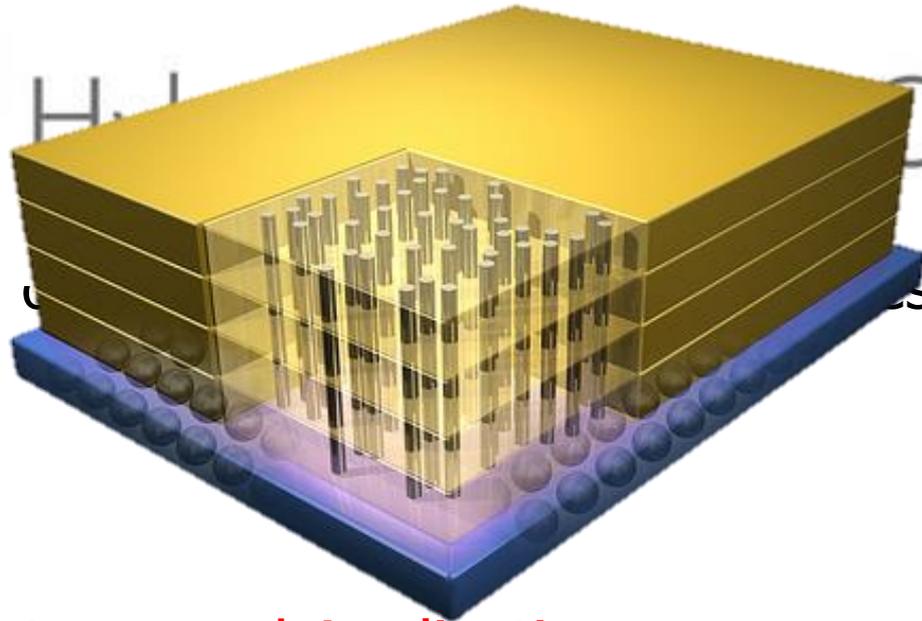
- Many questions ... How do we design the:
  - ❑ compute-capable memory & controllers?
  - ❑ processor chip and in-memory units?
  - ❑ software and hardware interfaces?
  - ❑ system software and languages?
  - ❑ algorithms?

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

# Why In-Memory Computation Today?



→ Industry C



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

# Agenda

---

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
  - Bottom Up: Push from Circuits and Devices
  - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips
  - Exploiting 3D-Stacked Memory
- How to Enable Adoption of Processing in Memory
- Conclusion

# Processing in Memory: Two Approaches

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

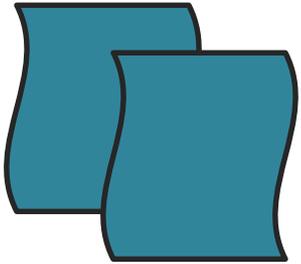
# Approach 1: Minimally Changing DRAM

---

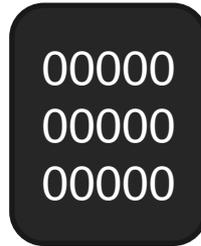
- DRAM has great capability to perform **bulk data movement and computation** internally with small changes
  - Can **exploit internal connectivity** to move data
  - Can **exploit analog computation capability**
  - ...
- Examples: RowClone, In-DRAM AND/OR, Gather/Scatter DRAM
  - RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data (Seshadri et al., MICRO 2013)
  - Fast Bulk Bitwise AND and OR in DRAM (Seshadri et al., IEEE CAL 2015)
  - Gather-Scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses (Seshadri et al., MICRO 2015)
  - "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology" (Seshadri et al., MICRO 2017)

# Starting Simple: Data Copy and Initialization

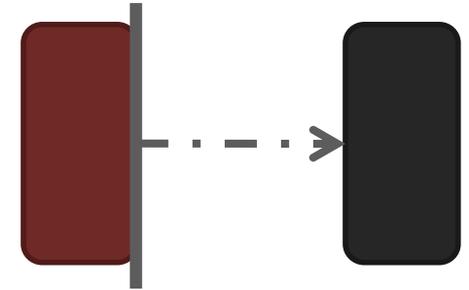
*memcpy & memmove: 5% cycles in Google's datacenter [Kanev+ ISCA'15]*



**Forking**



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



**Checkpointing**



**VM Cloning  
Deduplication**



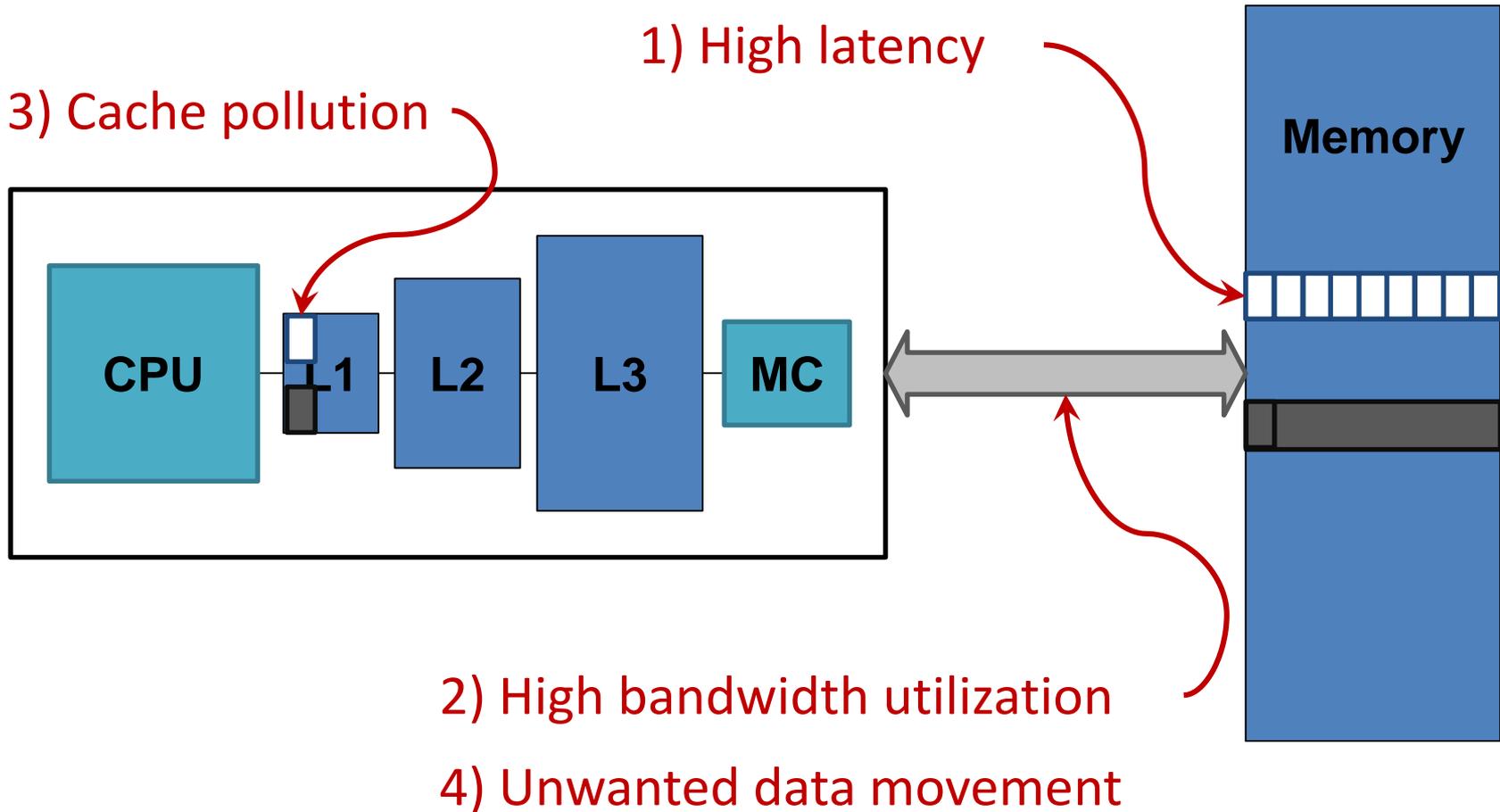
**Page Migration**

...  
Many more

# Today's Systems: Bulk Data Copy

1) High latency

3) Cache pollution



2) High bandwidth utilization

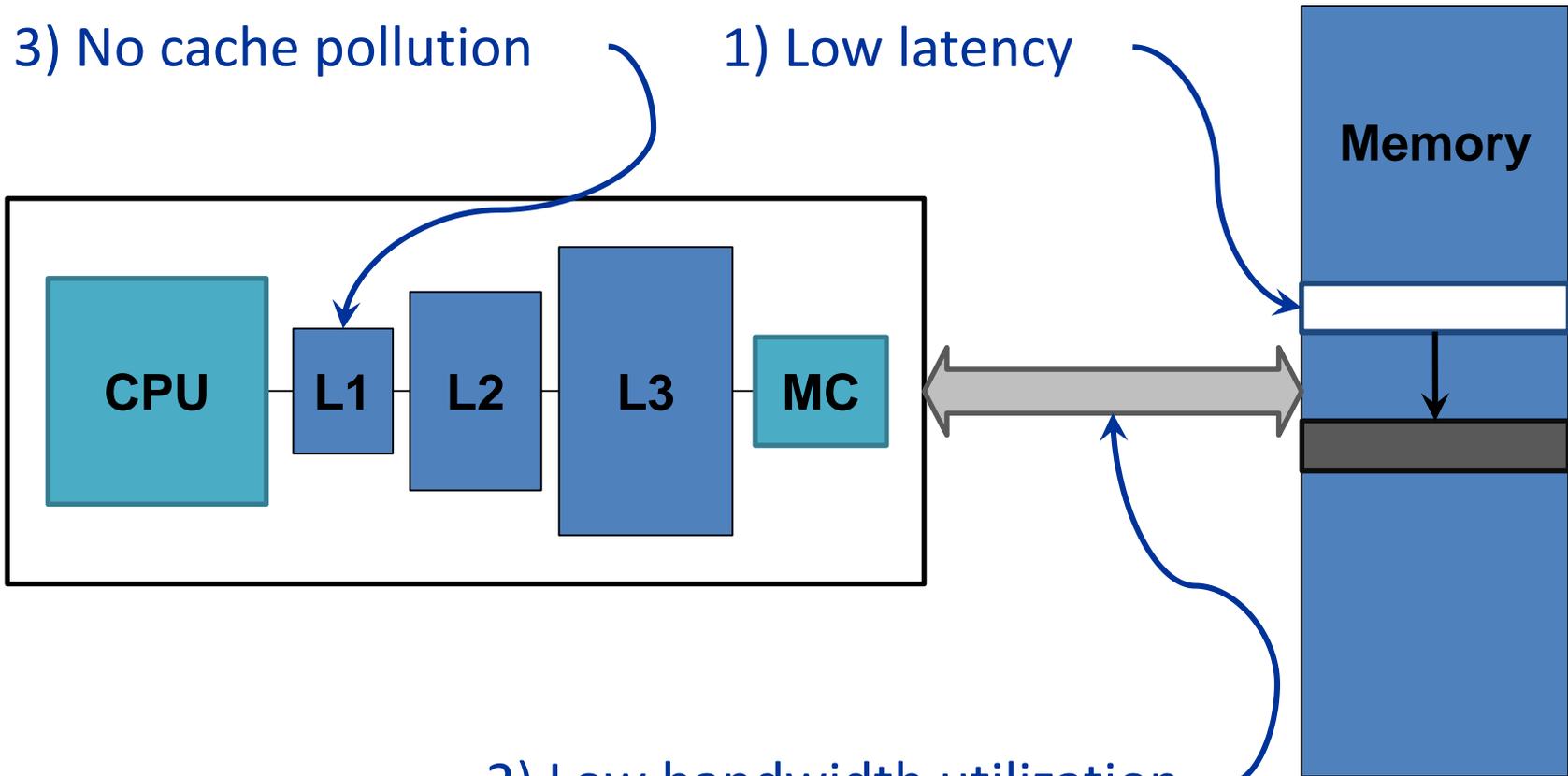
4) Unwanted data movement

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

# Future Systems: In-Memory Copy

3) No cache pollution

1) Low latency



2) Low bandwidth utilization

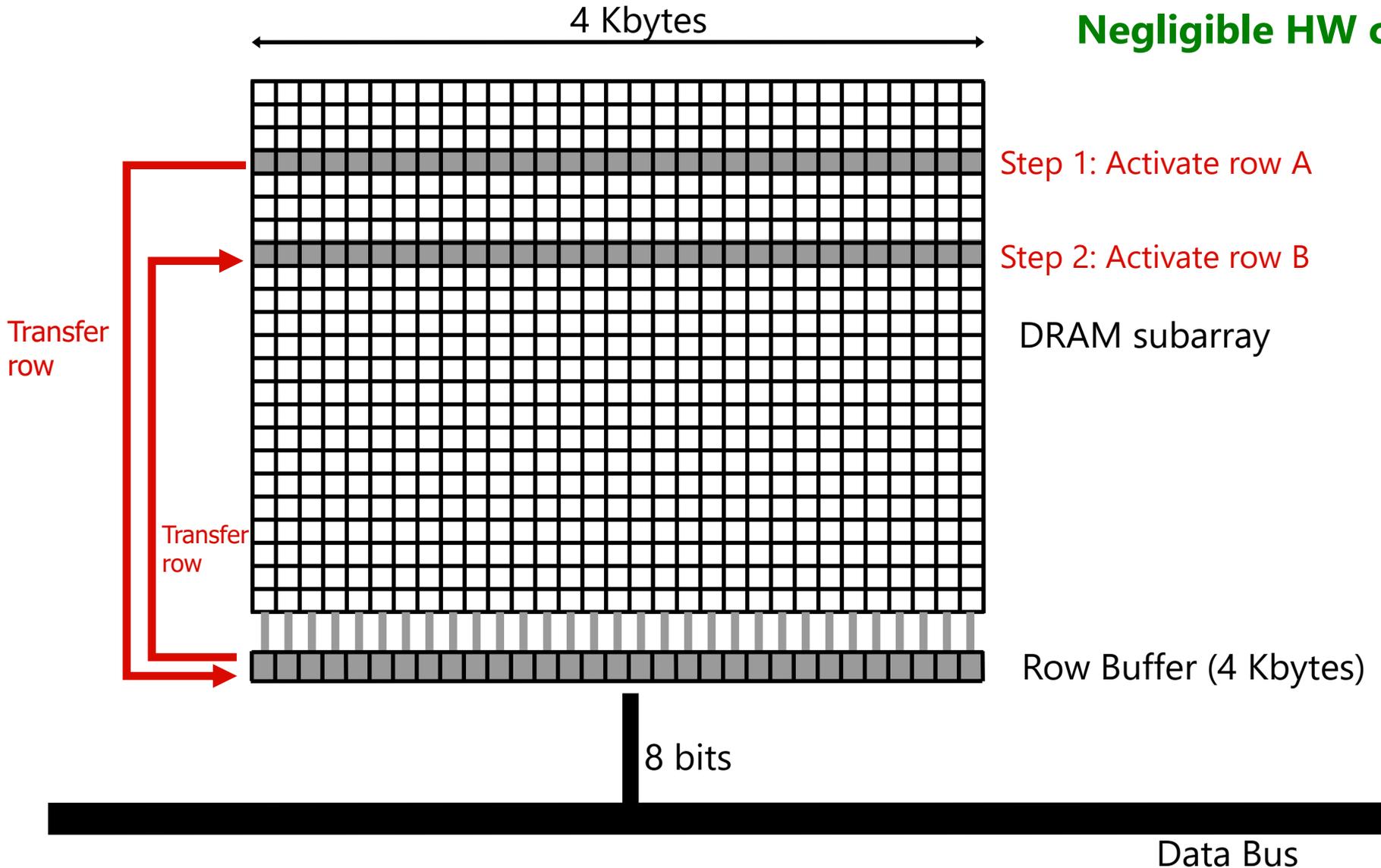
4) No unwanted data movement

1046ns, 3.6uJ

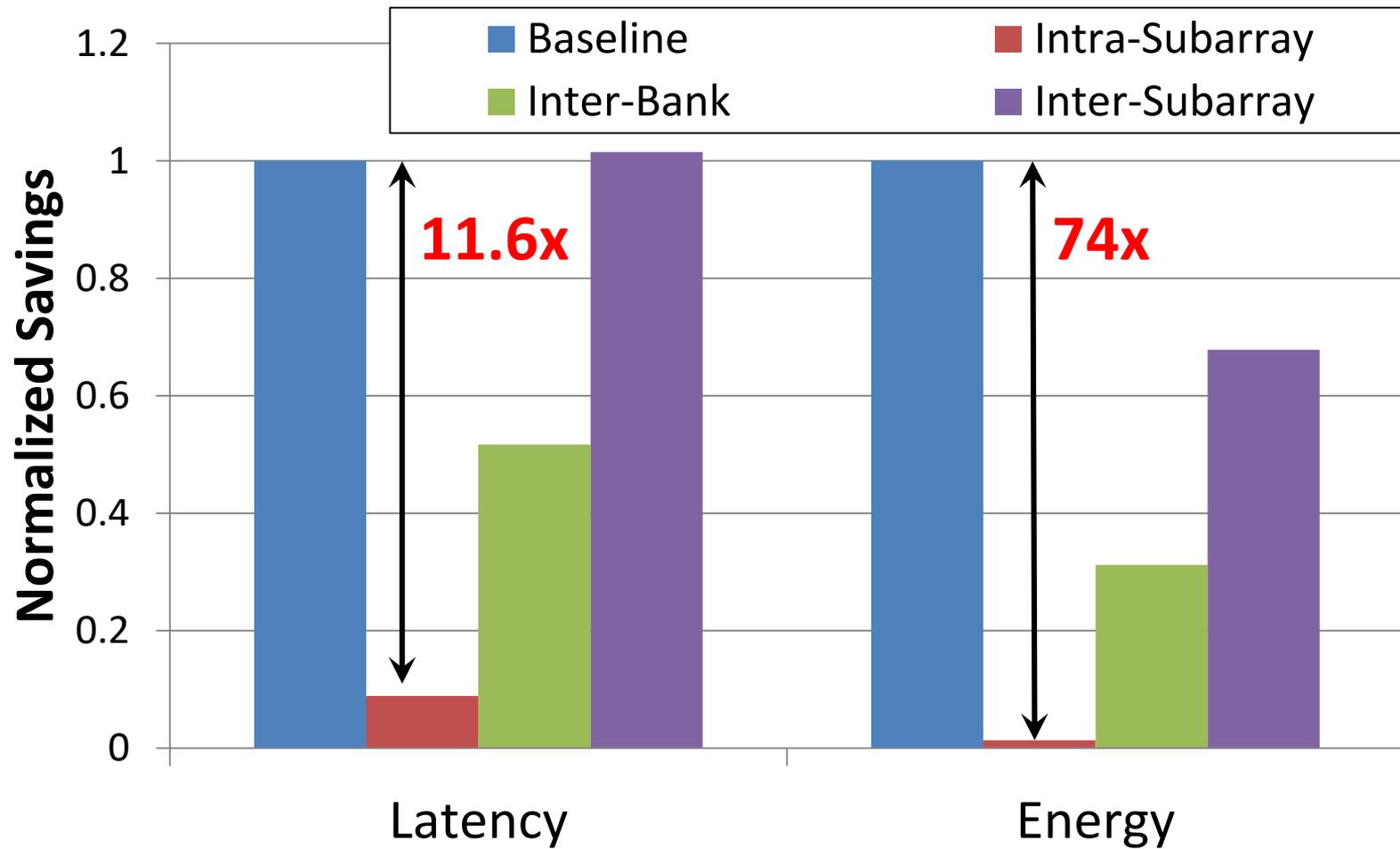
→ 90ns, 0.04uJ

# RowClone: In-DRAM Row Copy

**Idea: Two consecutive ACTivates**  
**Negligible HW cost**



# RowClone: Latency and Energy Savings



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

# More on RowClone

---

- Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry,  
**"RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization"**  
*Proceedings of the 46th International Symposium on Microarchitecture (MICRO)*, Davis, CA, December 2013. [[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Session Slides \(pptx\)](#)] [[pdf](#)] [[Poster \(pptx\)](#)] [[pdf](#)]

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

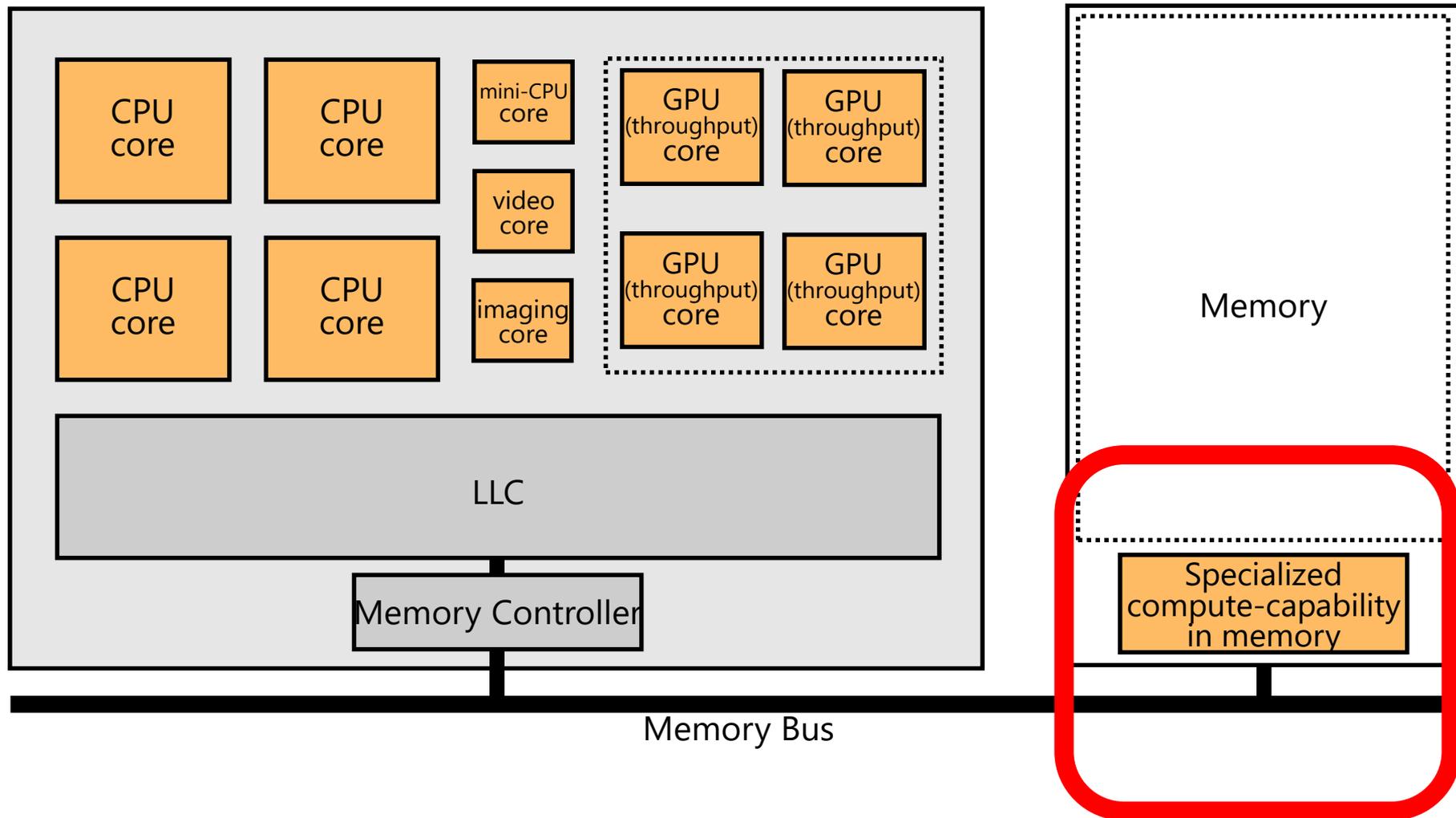
Vivek Seshadri      Yoongu Kim      Chris Fallin\*      Donghyuk Lee  
vseshadr@cs.cmu.edu    yoongukim@cmu.edu    cfallin@c1f.net    donghyuk1@cmu.edu

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

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

Carnegie Mellon University    †Intel Pittsburgh

# Memory as an Accelerator



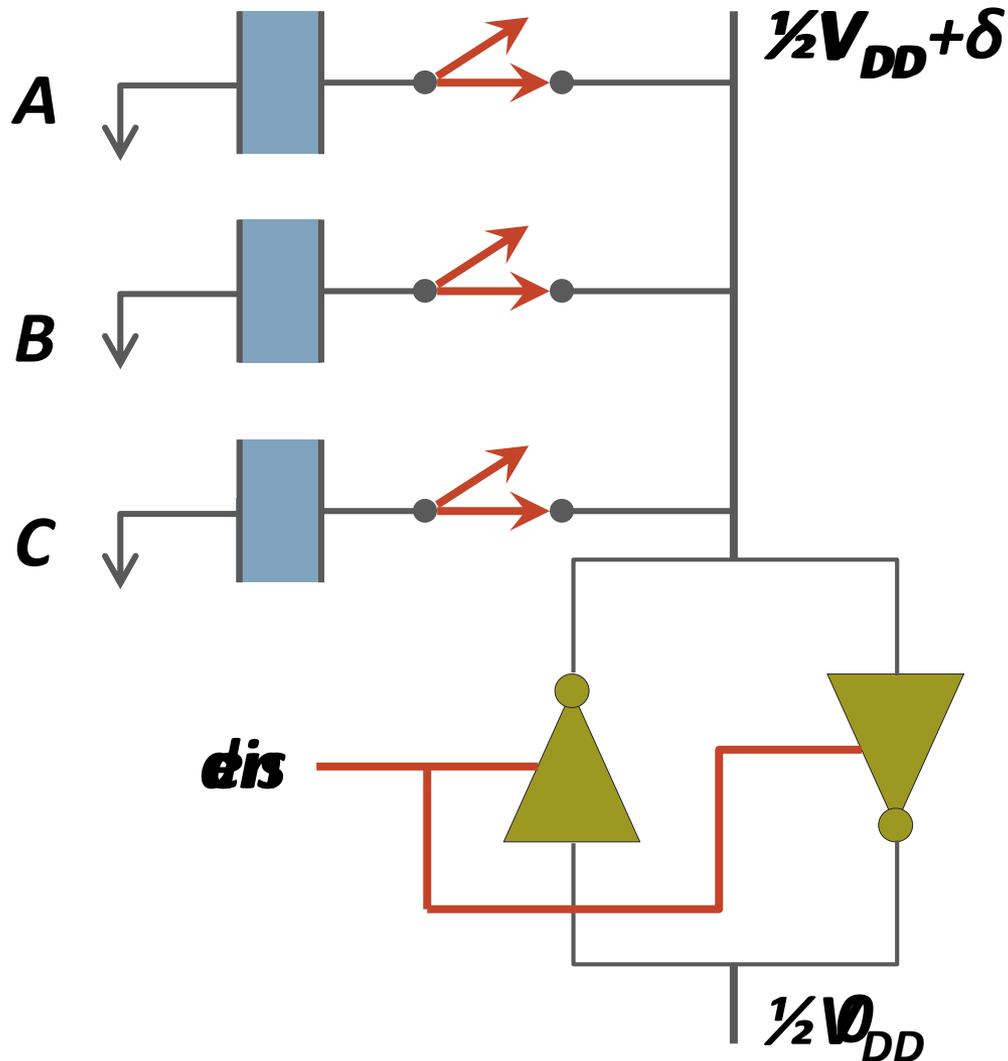
**Memory similar to a "conventional" accelerator**

# In-Memory Bulk Bitwise Operations

---

- We can support **in-DRAM COPY, ZERO, AND, OR, NOT, MAJ**
- At low cost
- Using analog computation capability of DRAM
  - Idea: activating multiple rows performs computation
- **30-60X performance and energy improvement**
  - Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," MICRO 2017.
  
- **New memory technologies** enable even more opportunities
  - Memristors, resistive RAM, phase change mem, STT-MRAM, ...
  - Can operate on data **with minimal movement**

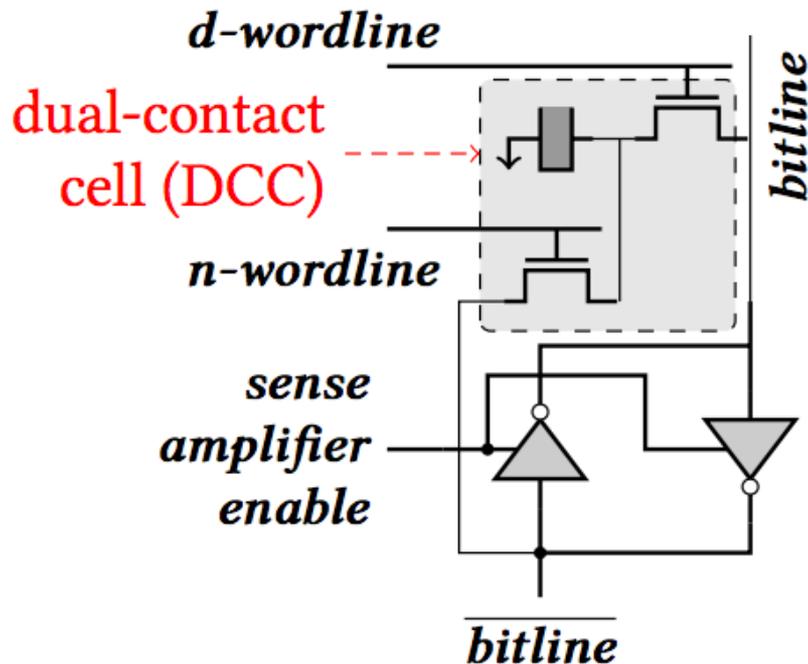
# In-DRAM AND/OR: Triple Row Activation



**Final State**  
 $AB + BC + AC$

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

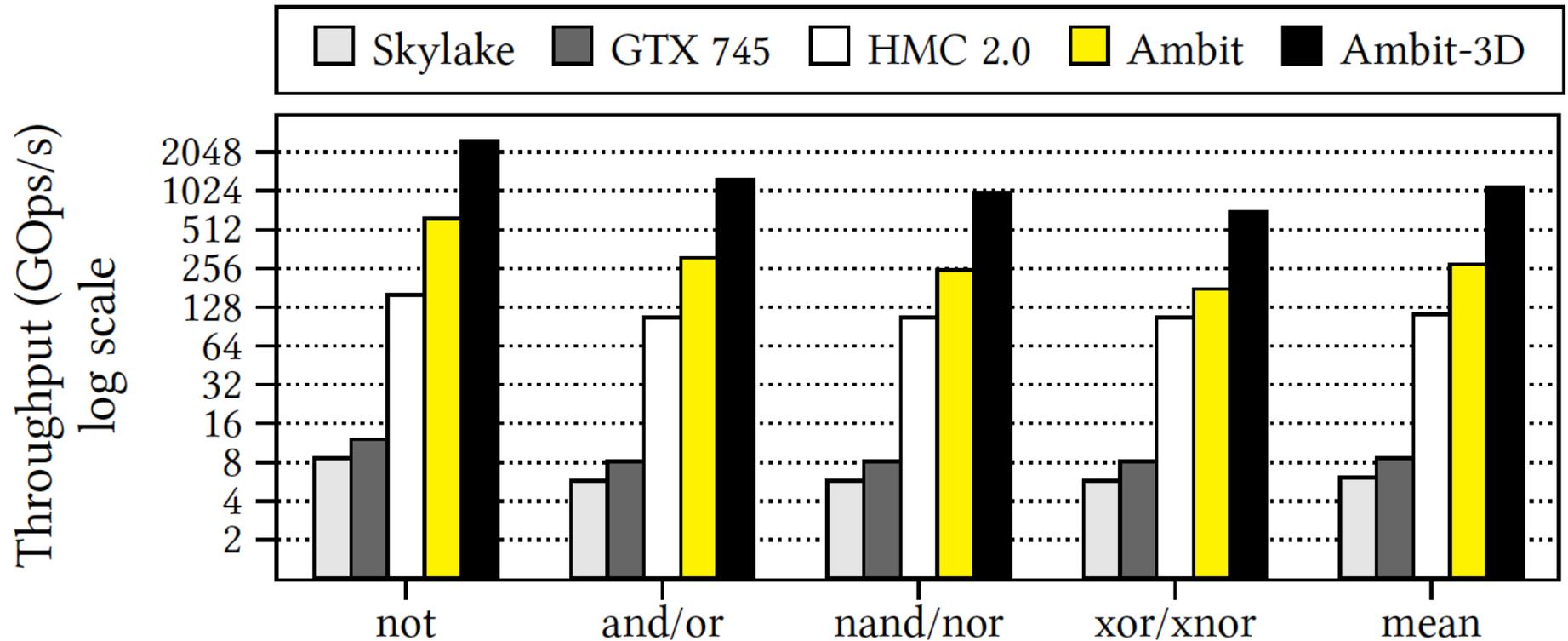
# In-DRAM NOT: Dual Contact Cell



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

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

# Performance: In-DRAM Bitwise Operations



**Figure 9: Throughput of bitwise operations on various systems.**

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

# Energy of In-DRAM Bitwise Operations

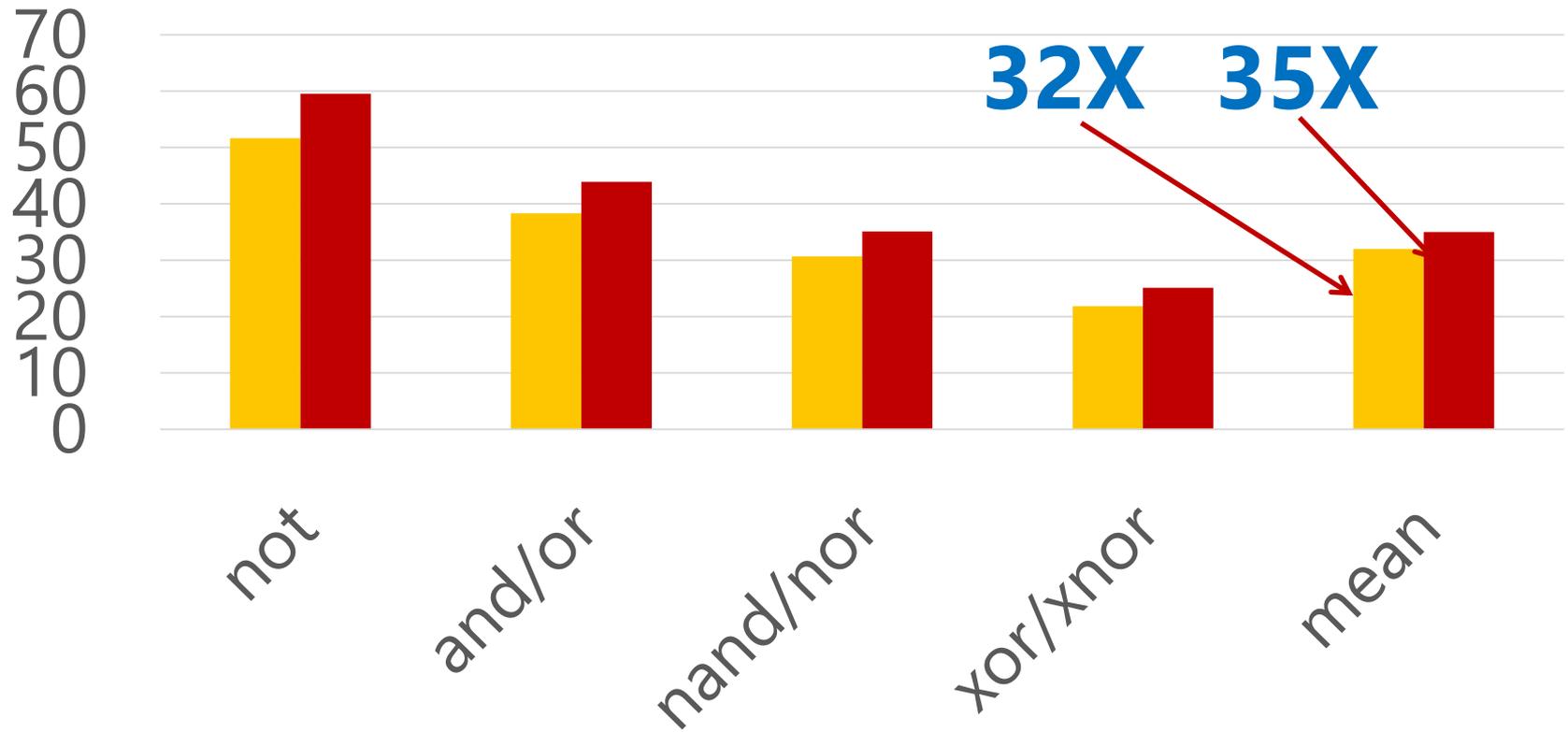
---

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

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

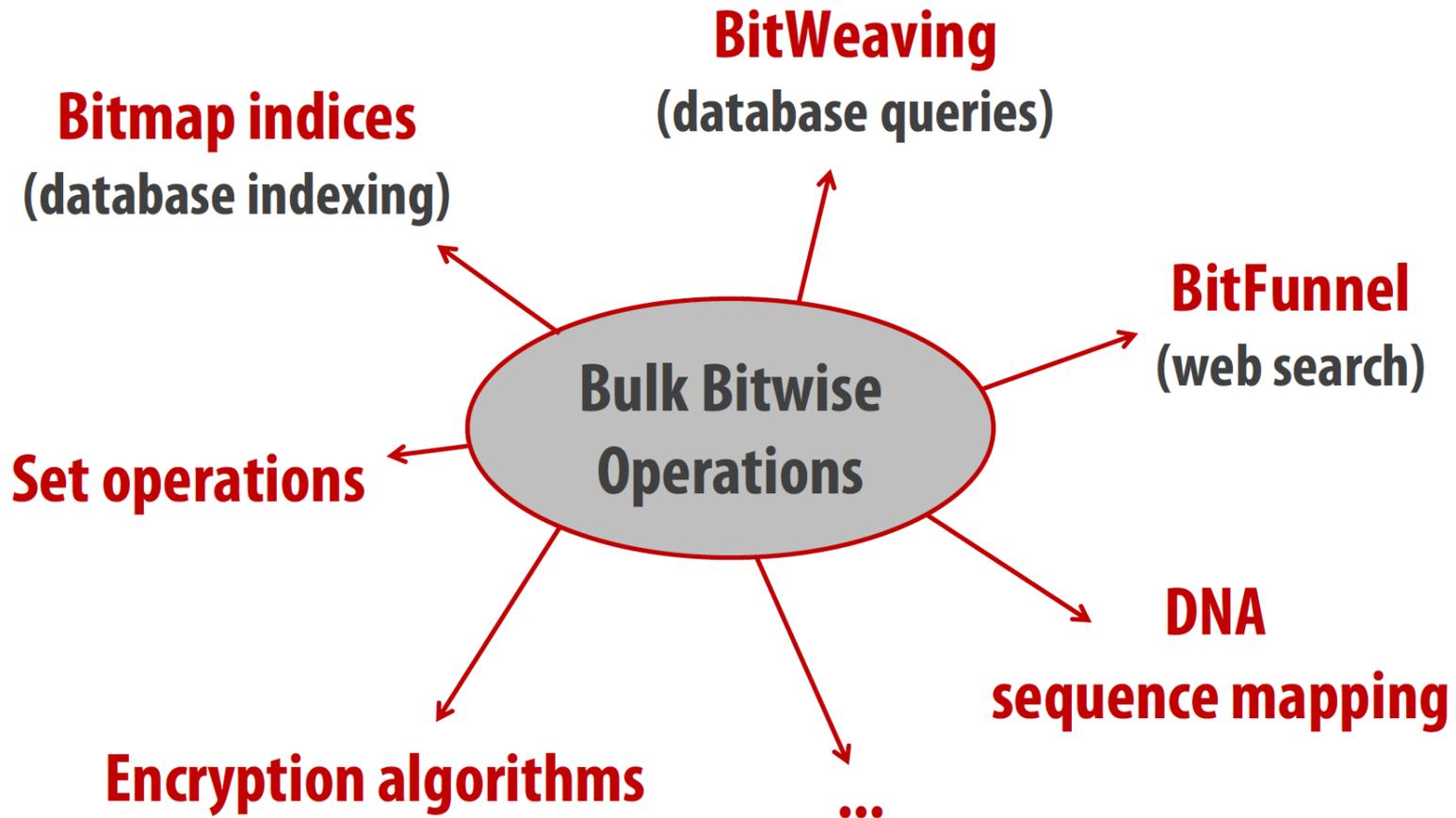
# Ambit vs. DDR3: Performance and Energy

■ Performance Improvement  
■ Energy Reduction



# Bulk Bitwise Operations in Workloads

---

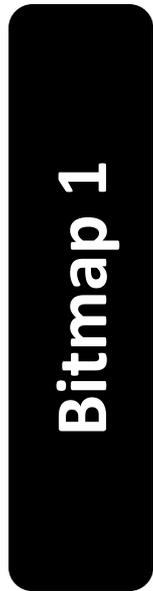


# Example Data Structure: Bitmap Index

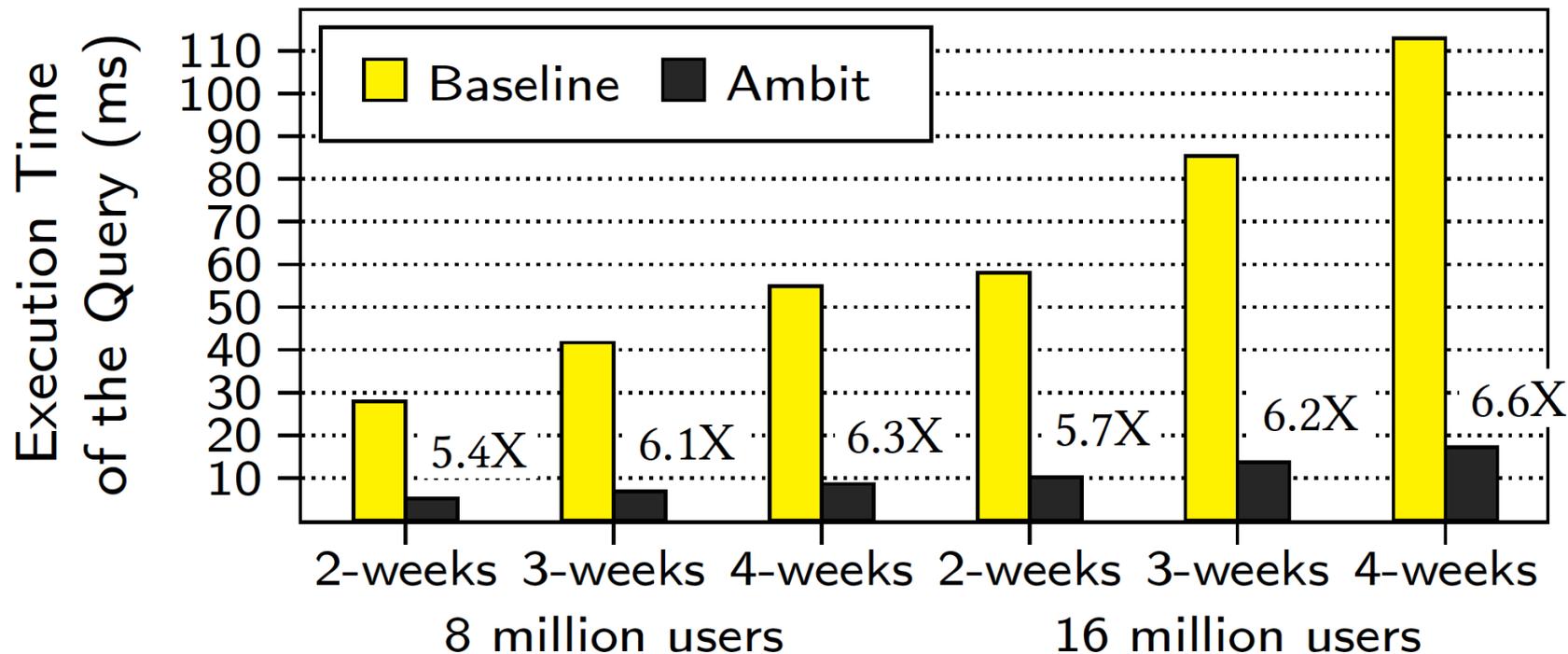
---

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

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



# Performance: Bitmap Index on Ambit

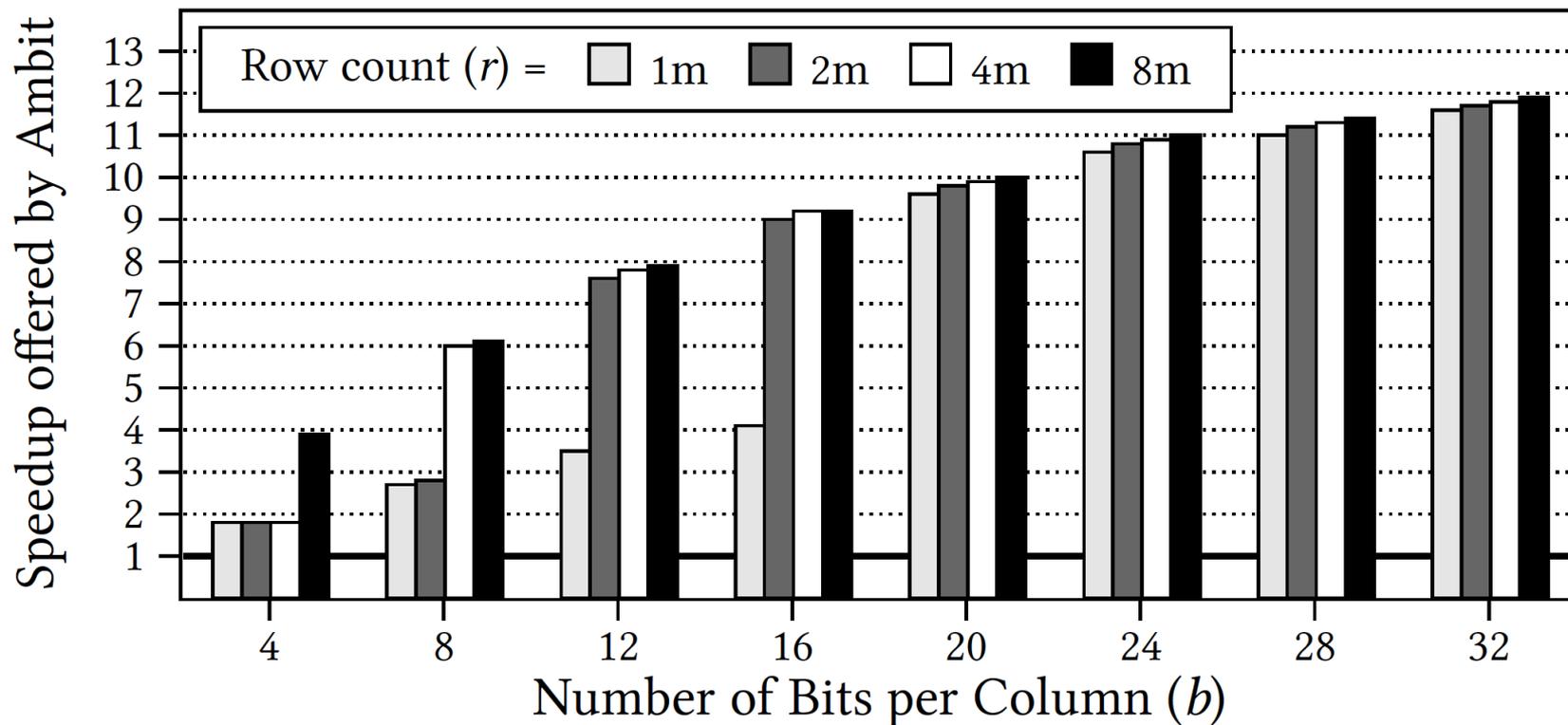


**Figure 10: Bitmap index performance. The value above each bar indicates the reduction in execution time due to Ambit.**

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

# Performance: BitWeaving on Ambit

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



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

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

# More on In-DRAM Bulk AND/OR

---

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

## Fast Bulk Bitwise AND and OR in DRAM

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

\*Carnegie Mellon University

†Intel Pittsburgh

# More on Ambit

---

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

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

Vivek Seshadri<sup>1,5</sup> Donghyuk Lee<sup>2,5</sup> Thomas Mullins<sup>3,5</sup> Hasan Hassan<sup>4</sup> Amirali Boroumand<sup>5</sup>  
Jeremie Kim<sup>4,5</sup> Michael A. Kozuch<sup>3</sup> Onur Mutlu<sup>4,5</sup> Phillip B. Gibbons<sup>5</sup> Todd C. Mowry<sup>5</sup>

<sup>1</sup>Microsoft Research India   <sup>2</sup>NVIDIA Research   <sup>3</sup>Intel   <sup>4</sup>ETH Zürich   <sup>5</sup>Carnegie Mellon University

# Computing Architectures with Minimal Data Movement

# Challenge: Intelligent Memory Device

---

Does **memory**  
have to be  
**dumb?**

# Agenda

---

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
  - Bottom Up: Push from Circuits and Devices
  - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips
  - Exploiting 3D-Stacked Memory
- How to Enable Adoption of Processing in Memory
- Conclusion

# Processing in Memory: Two Approaches

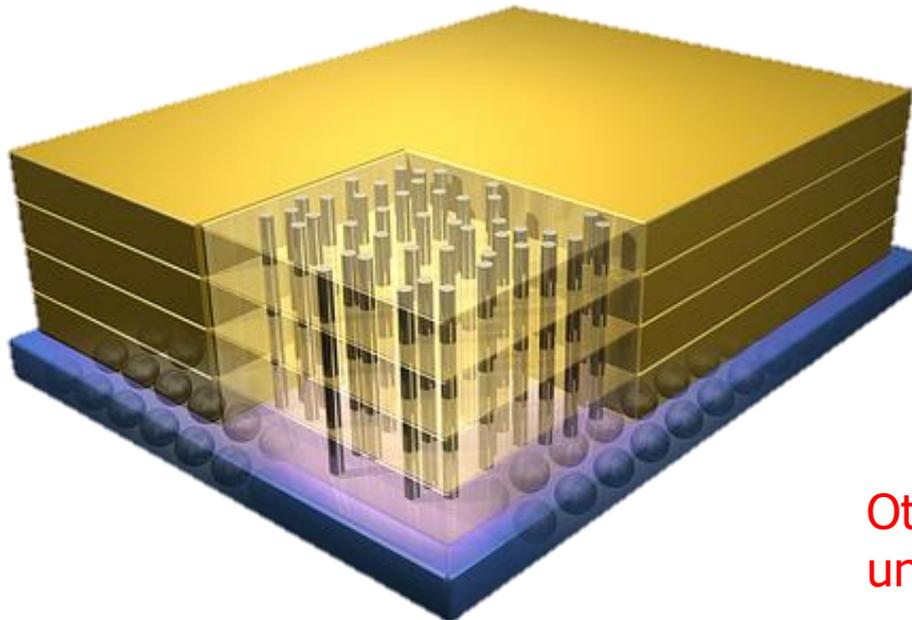
1. Minimally changing memory chips
2. Exploiting 3D-stacked memory

# Opportunity: 3D-Stacked Logic+Memory

---



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



**Memory**

**Logic**

Other "True 3D" technologies  
under development

# DRAM Landscape (circa 2015)

<i>Segment</i>	<i>DRAM Standards &amp; Architectures</i>
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLD RAM3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

Table 1. Landscape of DRAM-based memory

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

# Two Key Questions in 3D-Stacked PIM

---

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

# Graph Processing

- Large graphs are everywhere (circa 2015)



36 Million  
Wikipedia Pages



1.4 Billion  
Facebook Users

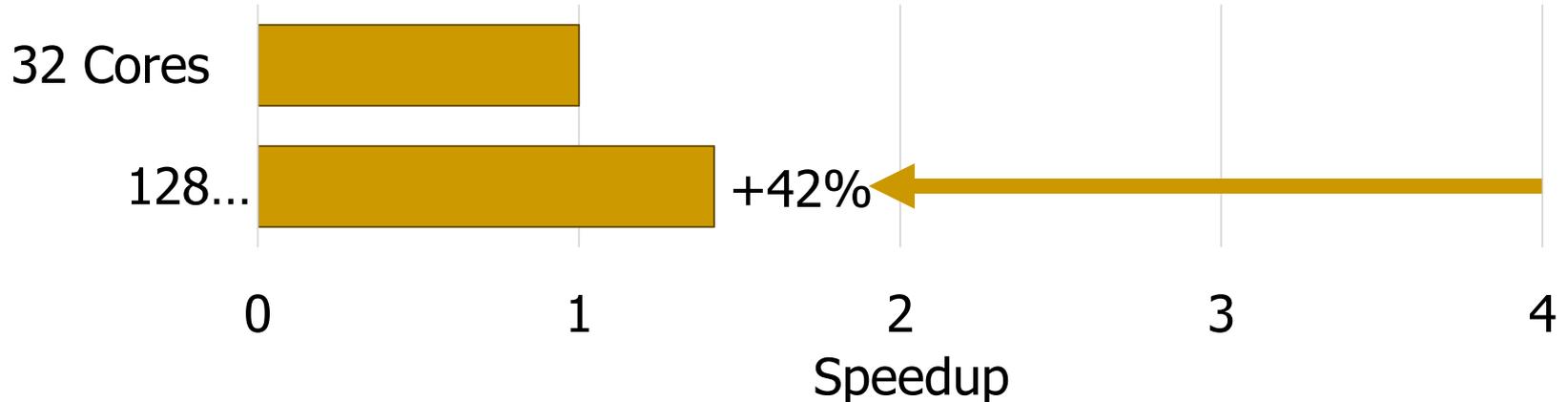


300 Million  
Twitter Users



30 Billion  
Instagram Photos

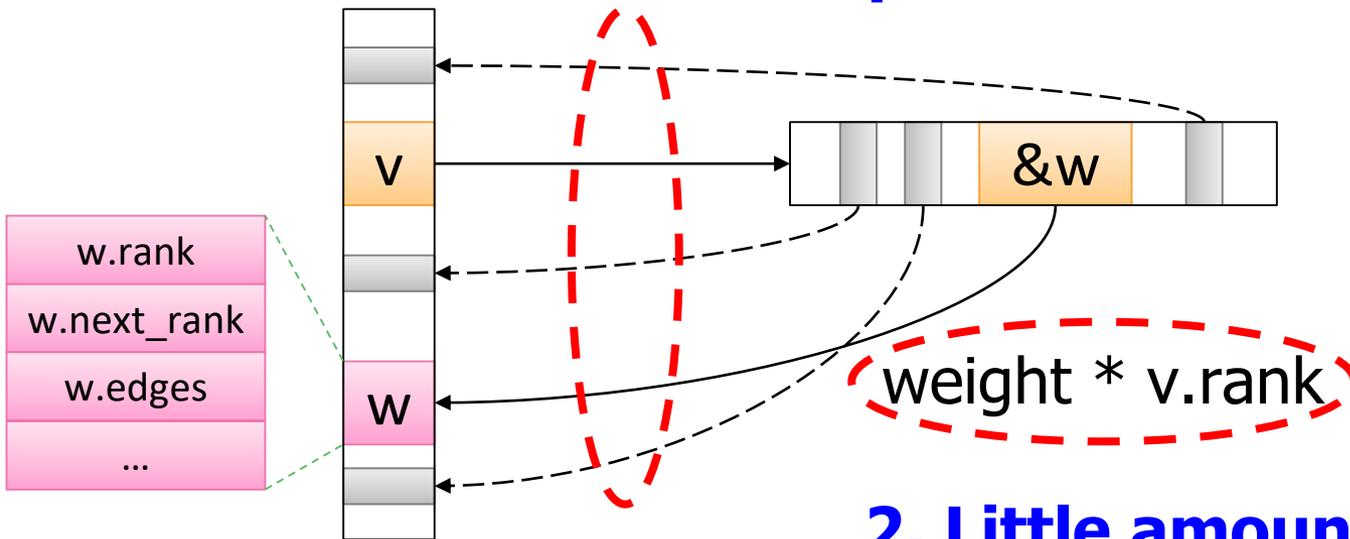
- Scalable large-scale graph processing is challenging



# Key Bottlenecks in Graph Processing

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

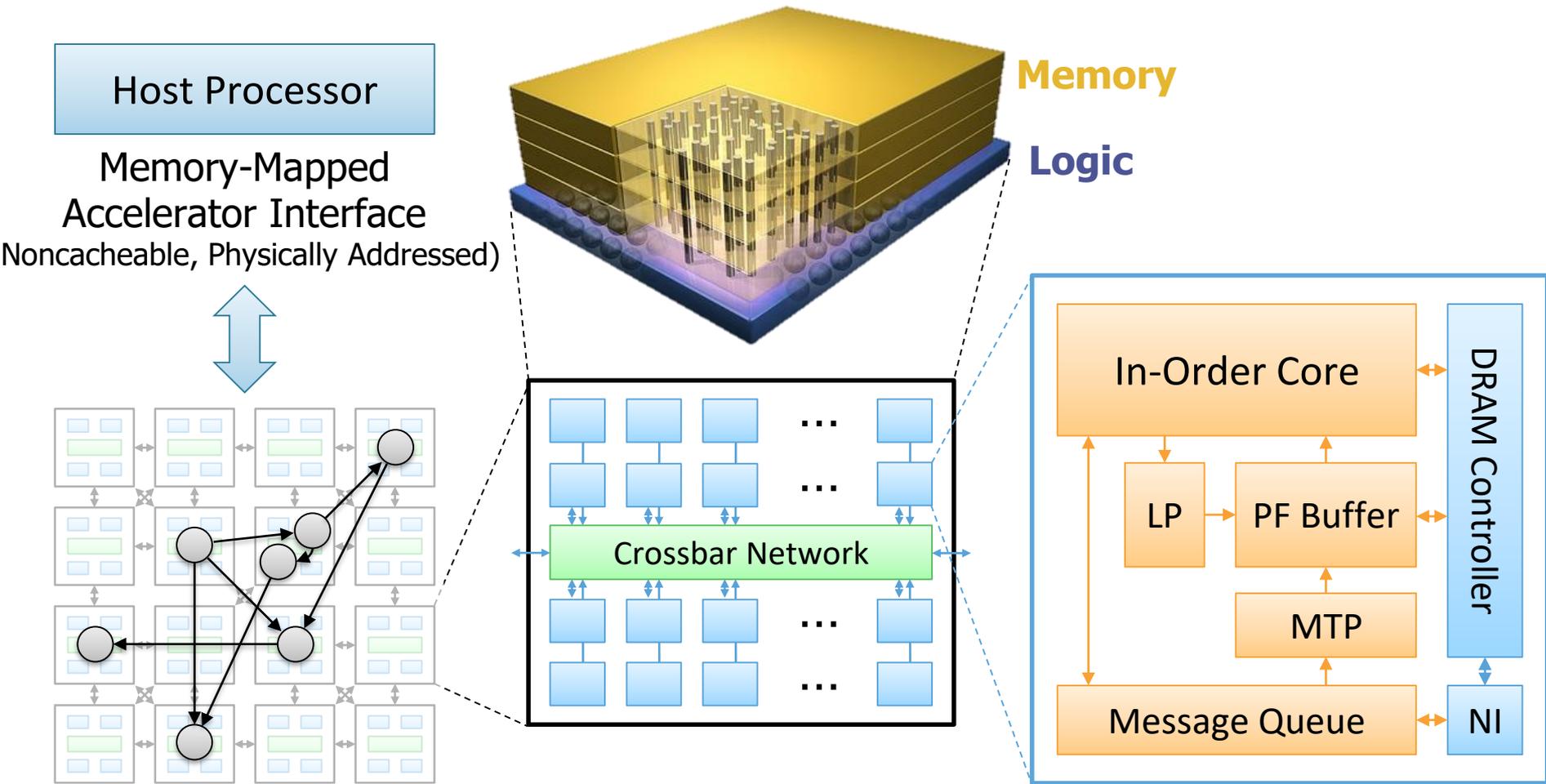
**1. Frequent random memory accesses**



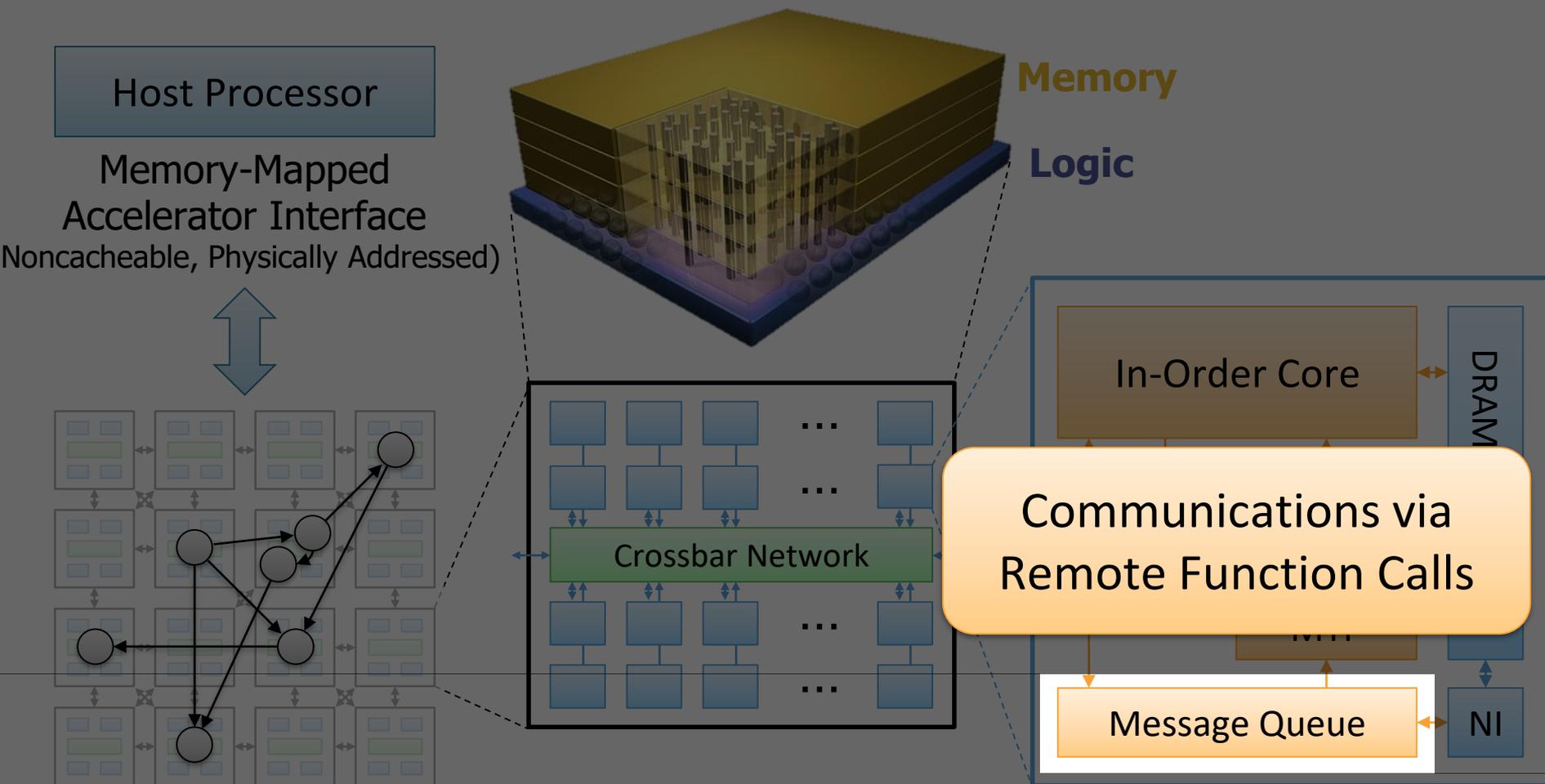
**2. Little amount of computation**

# Tesseract System for Graph Processing

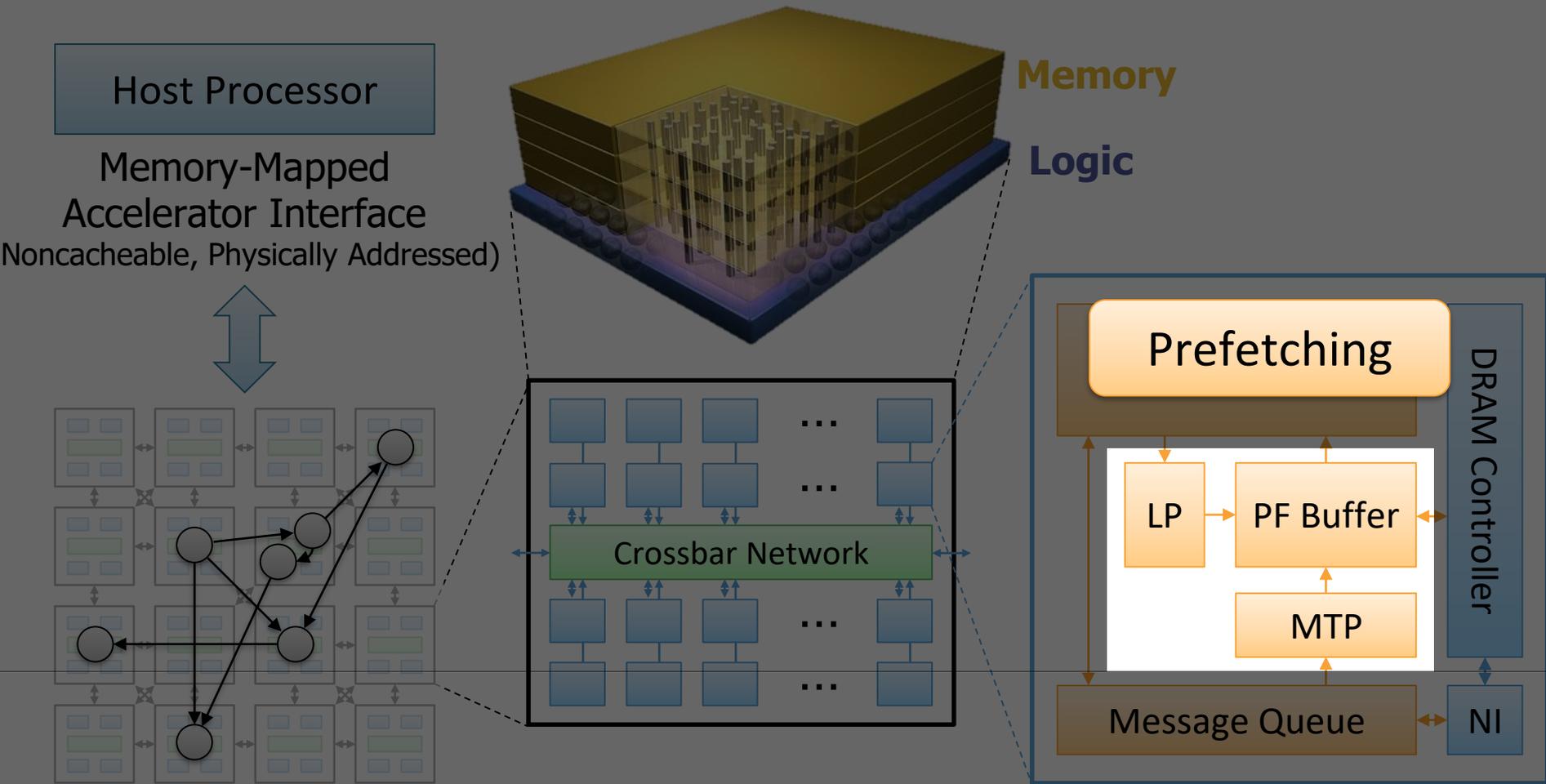
Interconnected set of 3D-stacked memory+logic chips with simple cores



# Tesseract System for Graph Processing



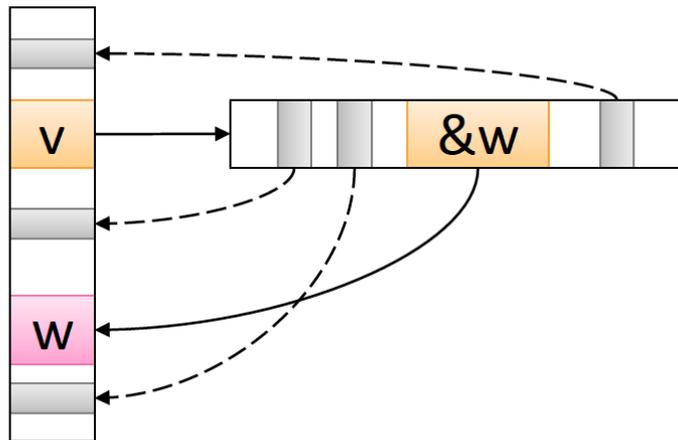
# Tesseract System for Graph Processing



# Communications In Tesseract (I)

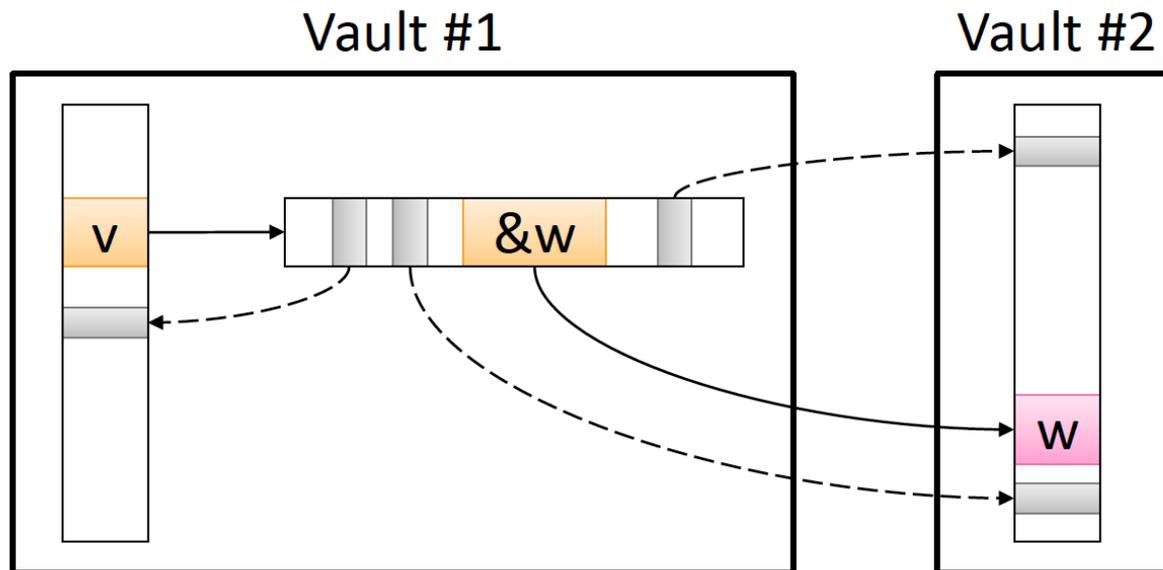
---

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



# Communications In Tesseract (II)

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

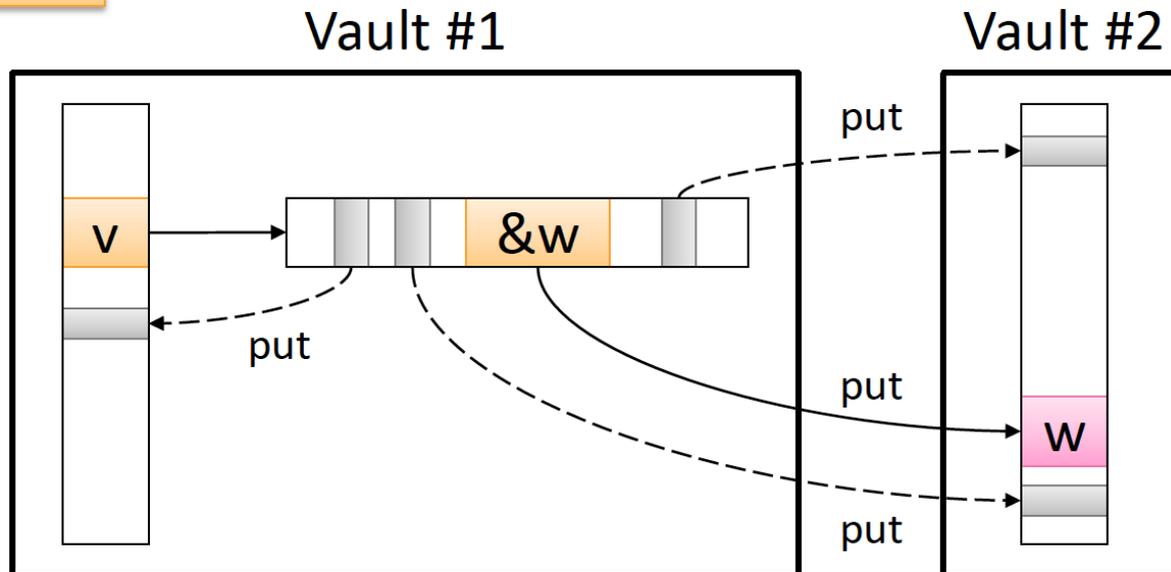


# Communications In Tesseract (III)

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

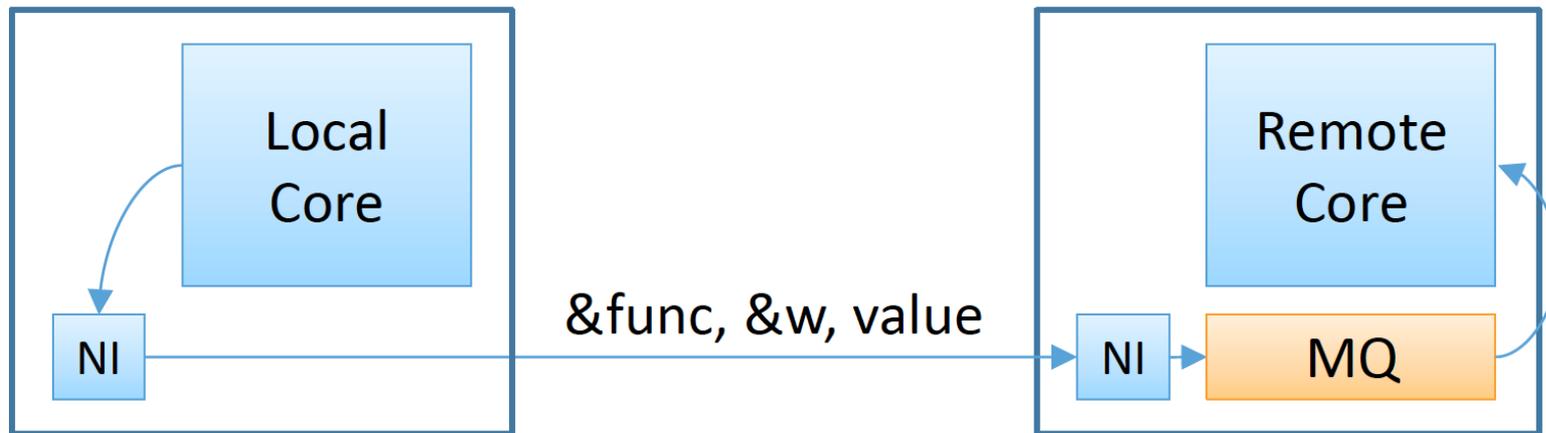
**Non-blocking Remote Function Call**

Can be **delayed**  
until the nearest barrier



# Remote Function Call (Non-Blocking)

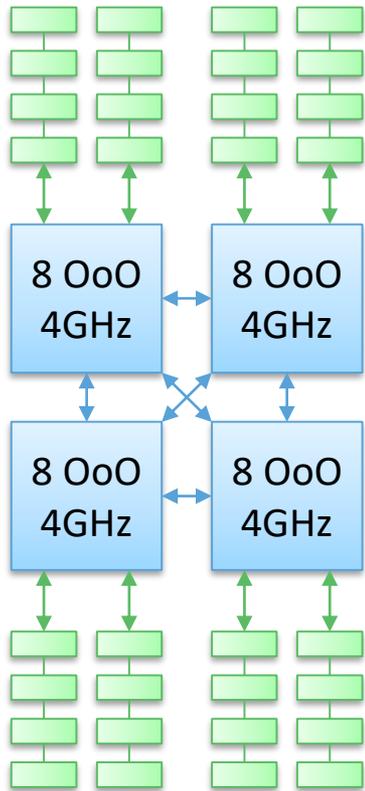
1. Send function address & args to the remote core
2. Store the incoming message to the message queue
3. Flush the message queue when it is full or a synchronization barrier is reached



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

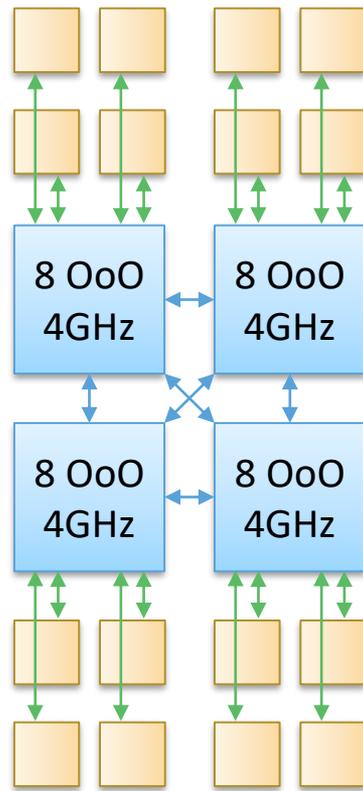
# Evaluated Systems

## DDR3-OoO



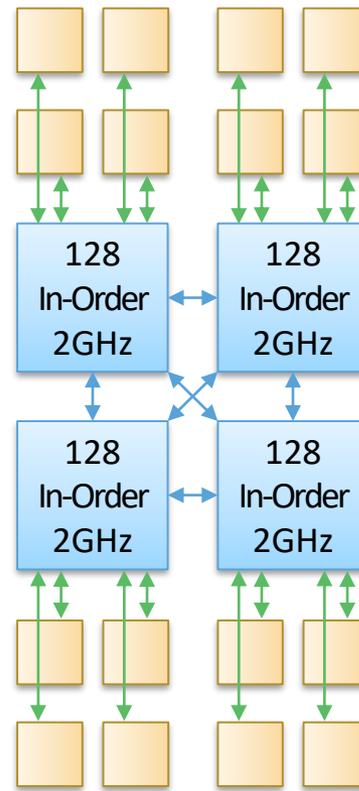
102.4GB/s

## HMC-OoO



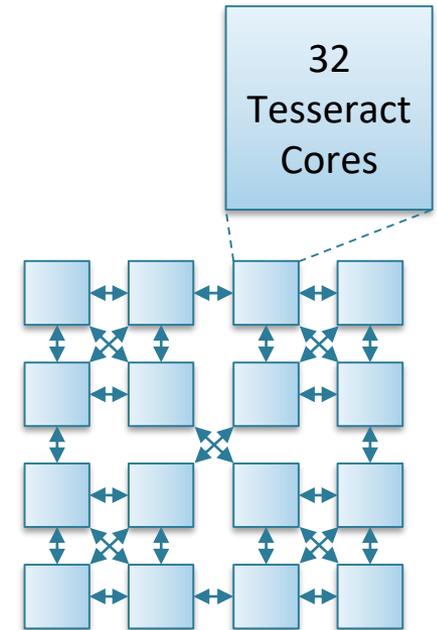
640GB/s

## HMC-MC



640GB/s

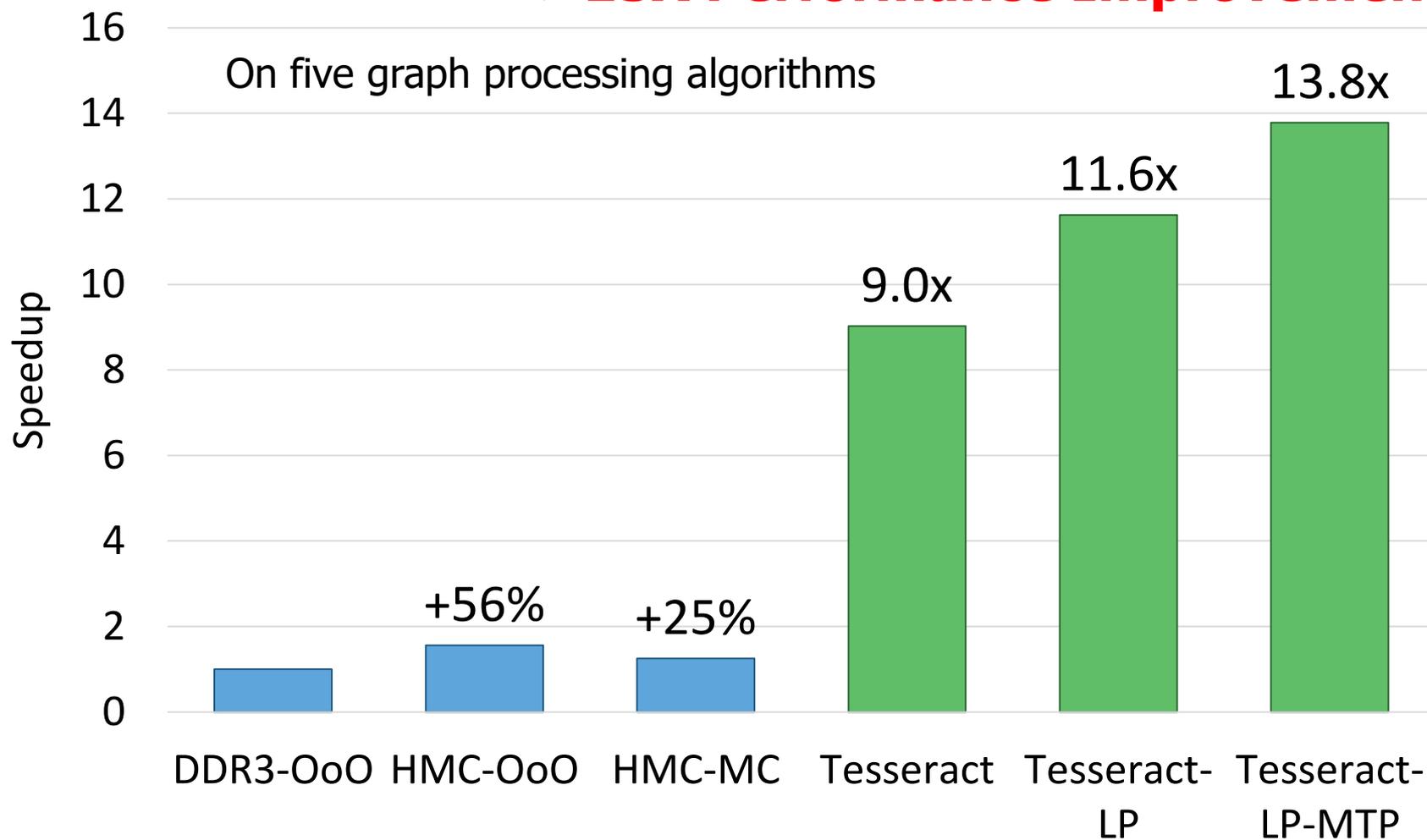
## Tesseract



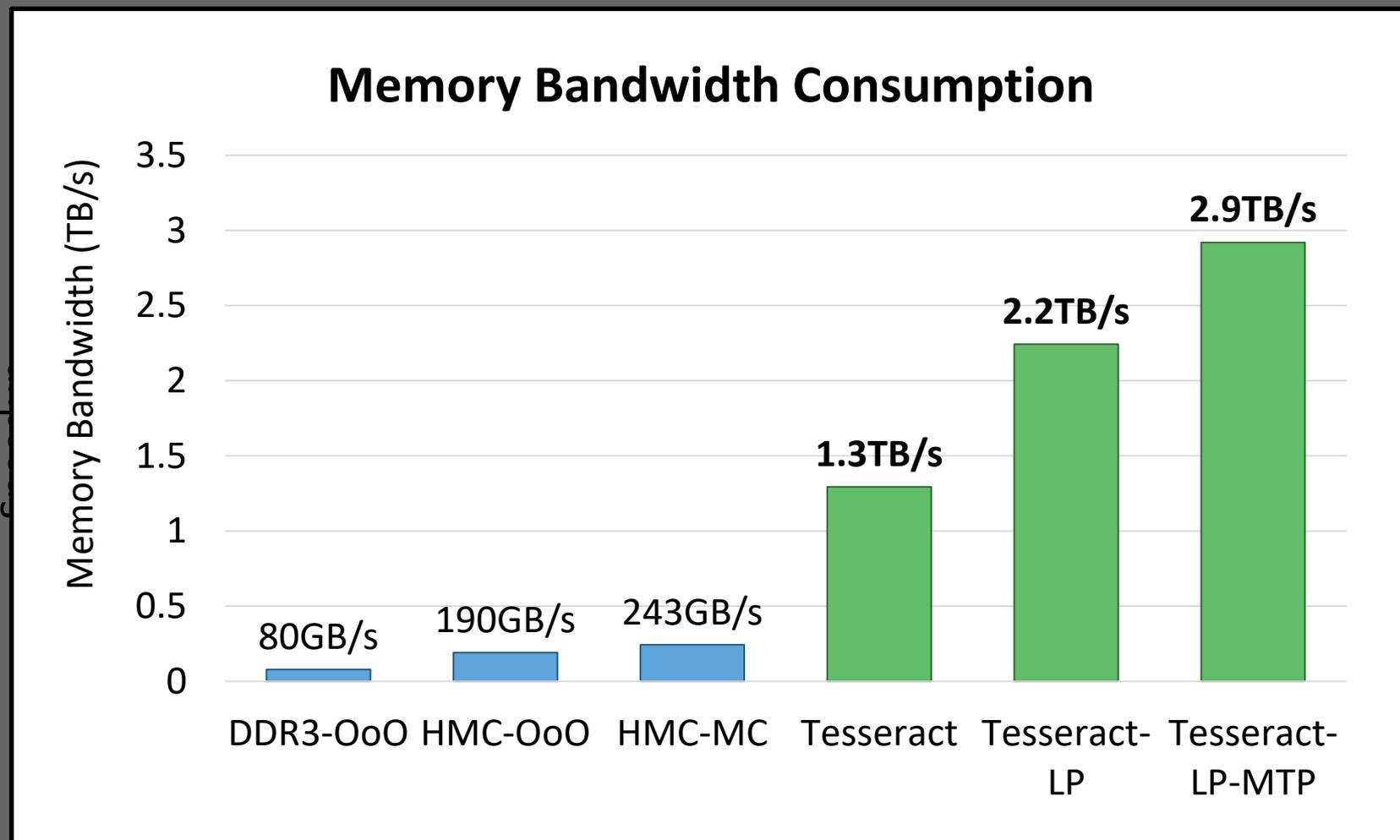
**8TB/s**

# Tesseract Graph Processing Performance

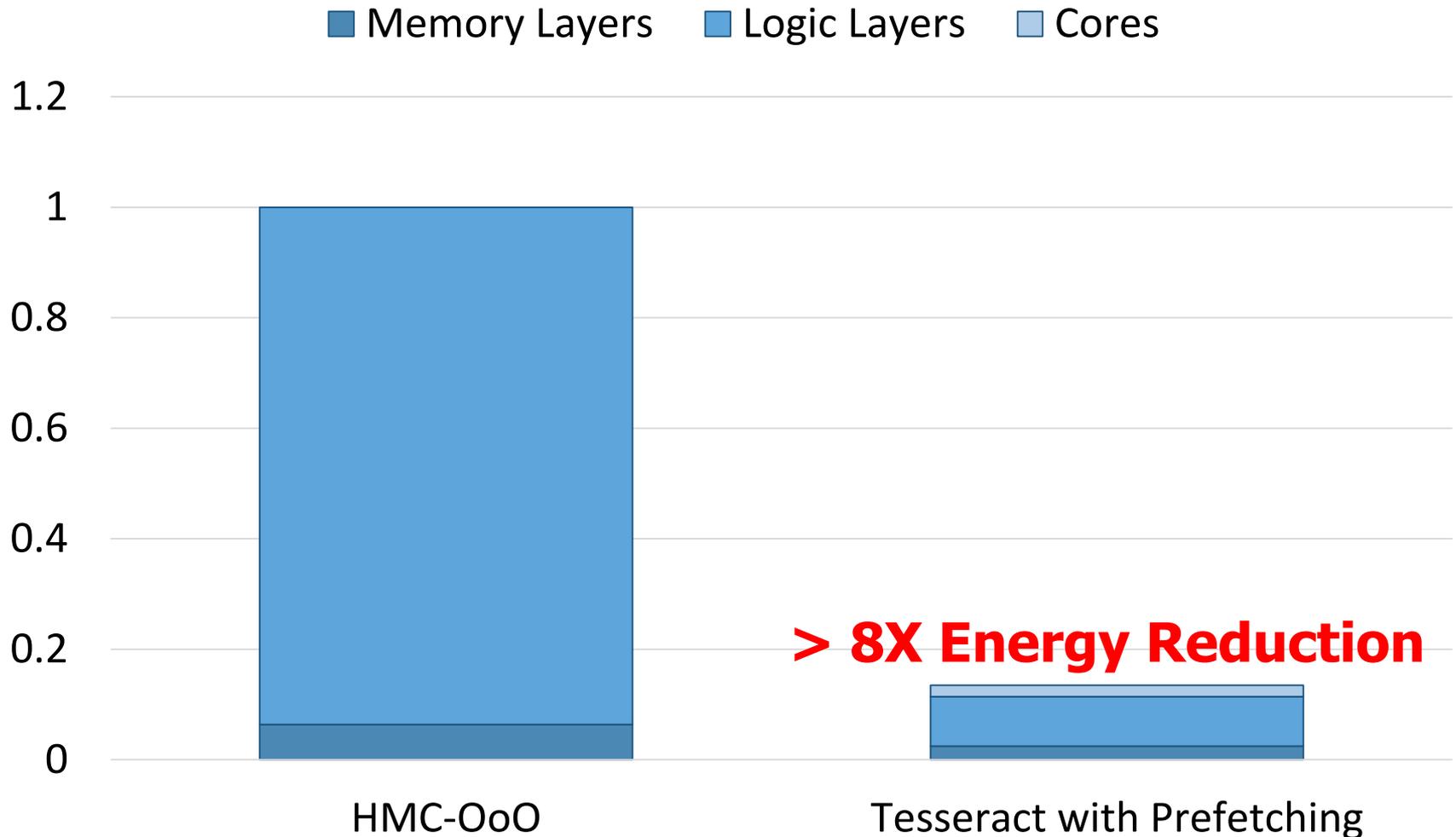
**>13X Performance Improvement**



# Tesseract Graph Processing Performance



# 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<sup>§</sup>   Sungjoo Yoo   Onur Mutlu<sup>†</sup>   Kiyoung Choi  
junwhan@snu.ac.kr, sungpack.hong@oracle.com, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

<sup>§</sup>Oracle Labs

<sup>†</sup>Carnegie Mellon University

# PIM on Mobile Devices

---

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu, **"Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"**  
*Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Williamsburg, VA, USA, March 2018.*

## Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand<sup>1</sup>

Saugata Ghose<sup>1</sup>

Youngsok Kim<sup>2</sup>

Rachata Ausavarungnirun<sup>1</sup>

Eric Shiu<sup>3</sup>

Rahul Thakur<sup>3</sup>

Daehyun Kim<sup>4,3</sup>

Aki Kuusela<sup>3</sup>

Allan Knies<sup>3</sup>

Parthasarathy Ranganathan<sup>3</sup>

Onur Mutlu<sup>5,1</sup>

# Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

**Amirali Boroumand**

Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun,  
Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela,  
Allan Knies, Parthasarathy Ranganathan, Onur Mutlu

**SAFARI**

**Carnegie Mellon**

**Google**



SEOUL  
NATIONAL  
UNIVERSITY

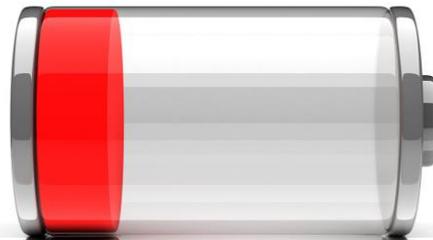
**ETH** zürich

# Consumer Devices



**Consumer devices are everywhere!**

**Energy consumption is  
a first-class concern in consumer devices**



# Popular Google Consumer Workloads



## Chrome

Google's web browser



## TensorFlow Mobile

Google's machine learning framework

# VP9



## Video Playback

Google's **video codec**

# VP9

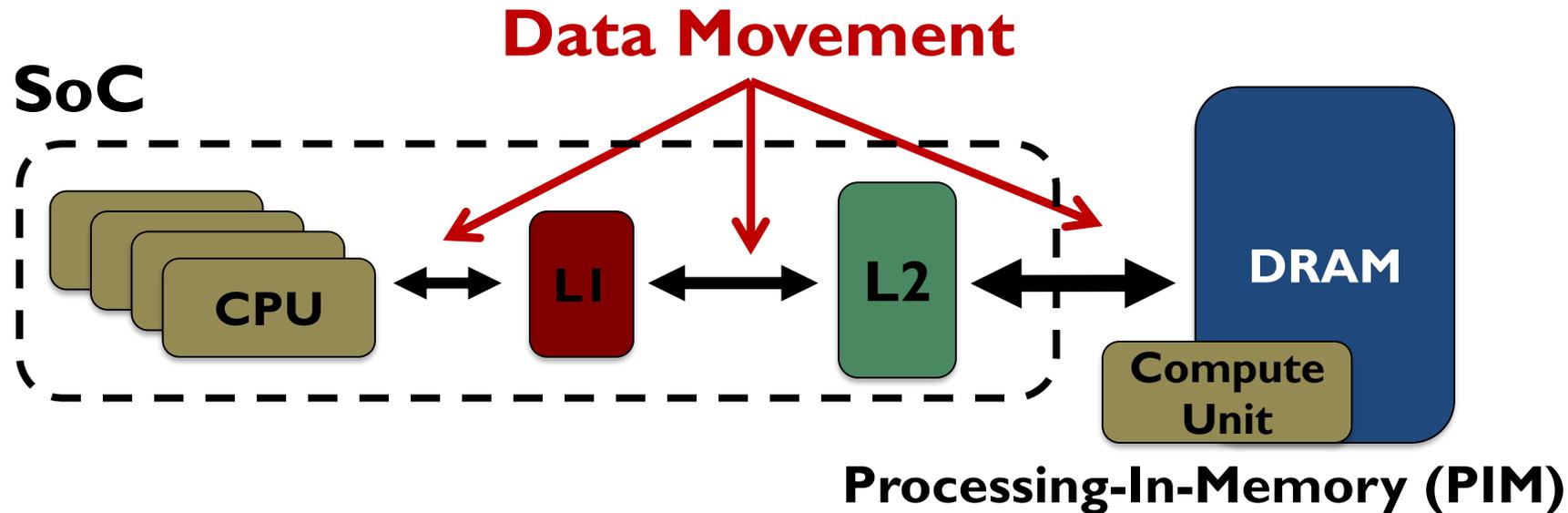


## Video Capture

Google's **video codec**

# Energy Cost of Data Movement

1<sup>st</sup> key observation: **62.7%** of the total system energy is spent on **data movement**



Potential solution: move computation **close to data**

Challenge: limited area and energy budget

# Using PIM to Reduce Data Movement

**2<sup>nd</sup> key observation:** a significant fraction of the **data movement** often comes from **simple functions**

We can design lightweight logic to implement these simple functions in **memory**

Small embedded  
low-power core



Small fixed-function  
accelerators



Offloading to PIM logic reduces energy and improves performance, on average, by 55.4% and 54.2%

# Workload Analysis



## Chrome

Google's web browser



## TensorFlow Mobile

Google's machine learning framework

# VP9



## Video Playback

Google's **video codec**

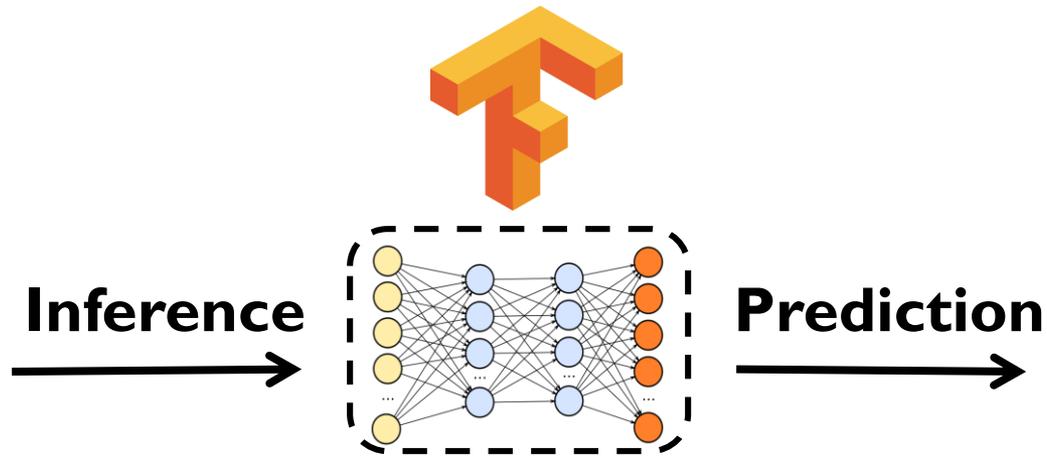
# VP9



## Video Capture

Google's **video codec**

# TensorFlow Mobile



**57.3%** of the inference energy is spent on data movement



**54.4%** of the **data movement** energy comes from packing/unpacking and quantization

# Packing



**Reorders** elements of matrices to minimize **cache misses** during **matrix multiplication**



Up to **40%** of the inference **energy** and **31%** of inference **execution time**



Packing's **data movement** accounts for up to **35.3%** of the inference **energy**

A simple **data reorganization** process that requires **simple arithmetic**

# Quantization



Converts 32-bit floating point to 8-bit integers to improve inference execution time and energy consumption



Up to **16.8%** of the inference **energy** and **16.1%** of inference **execution time**



Majority of **quantization** energy comes from **data movement**

A simple **data conversion** operation that requires **shift, addition, and multiplication** operations

# Quantization



Converts 32-bit floating point to 8-bit integers to improve inference execution time and energy consumption

**Based on our analysis, we conclude that:**

- Both functions are good candidates for **PIM execution**
- It is **feasible** to implement them in **PIM logic**

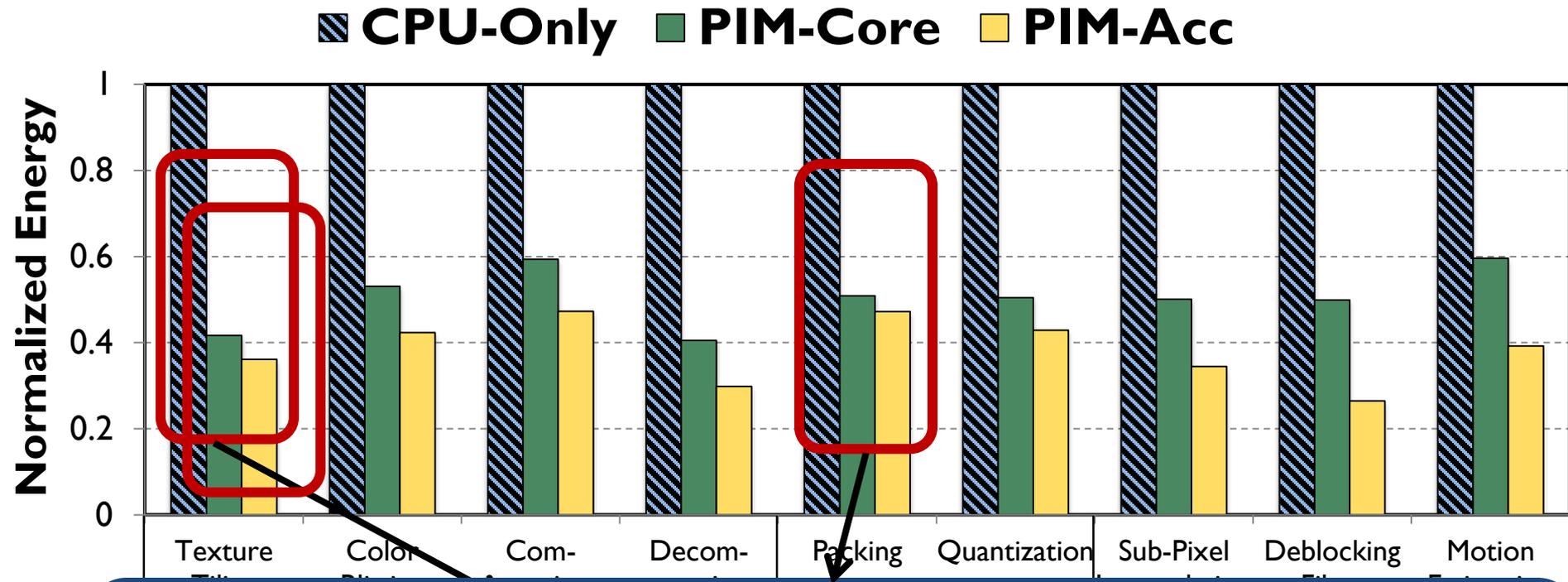
inference execution time

A simple **data conversion** operation that requires **shift, addition, and multiplication** operations

# Evaluation Methodology

- **System Configuration (gem5 Simulator)**
  - **SoC:** 4 OoO cores, 8-wide issue, 64 kB L1 cache, 2MB L2 cache
  - **PIM Core:** 1 core per vault, 1-wide issue, 4-wide SIMD, 32kB L1 cache
  - **3D-Stacked Memory:** 2GB cube, 16 vaults per cube
    - Internal Bandwidth: 256GB/S
    - Off-Chip Channel Bandwidth: 32 GB/s
  - **Baseline Memory:** LPDDR3, 2GB, FR-FCFS scheduler
- **We study each target in isolation and emulate each separately and run them in our simulator**

# Normalized Energy

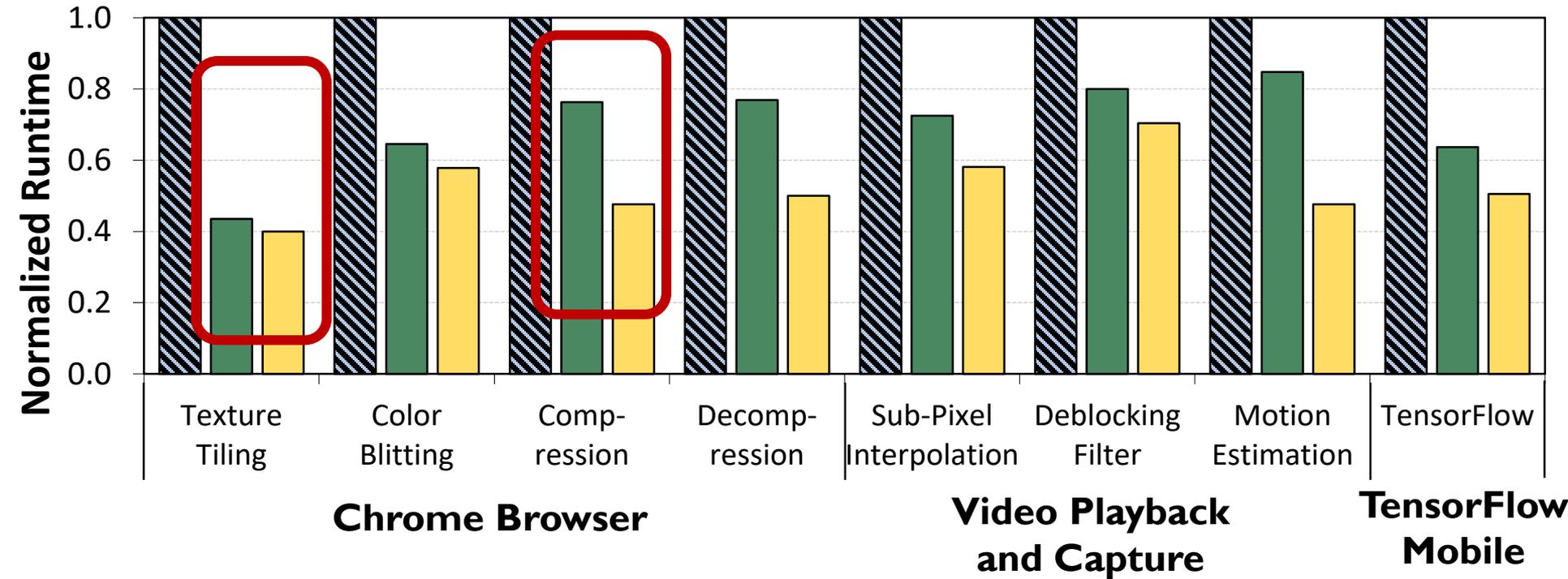


**77.7%** and **82.6%** of energy reduction for **texture tiling** and **packing** comes from eliminating **data movement**

**PIM core** and **PIM accelerator** reduces **energy consumption** on average by **49.1%** and **55.4%**

# Normalized Runtime

▨ CPU-Only █ PIM-Core █ PIM-Acc



Offloading these kernels to **PIM core** and **PIM accelerator** improves **performance** on average by **44.6%** and **54.2%**

# Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

**Amirali Boroumand**

Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun,  
Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela,  
Allan Knies, Parthasarathy Ranganathan, Onur Mutlu

**ASPLOS 2018**

**SAFARI**

**Carnegie Mellon**

**Google**



SEOUL  
NATIONAL  
UNIVERSITY

**ETH** zürich

# More on PIM for Mobile Devices

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu, **"Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"** *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Williamsburg, VA, USA, March 2018.

**62.7%** of the total system energy  
is spent on **data movement**

## Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand<sup>1</sup>

Saugata Ghose<sup>1</sup>

Youngsok Kim<sup>2</sup>

Rachata Ausavarungnirun<sup>1</sup>

Eric Shiu<sup>3</sup>

Rahul Thakur<sup>3</sup>

Daehyun Kim<sup>4,3</sup>

Aki Kuusela<sup>3</sup>

Allan Knies<sup>3</sup>

Parthasarathy Ranganathan<sup>3</sup>

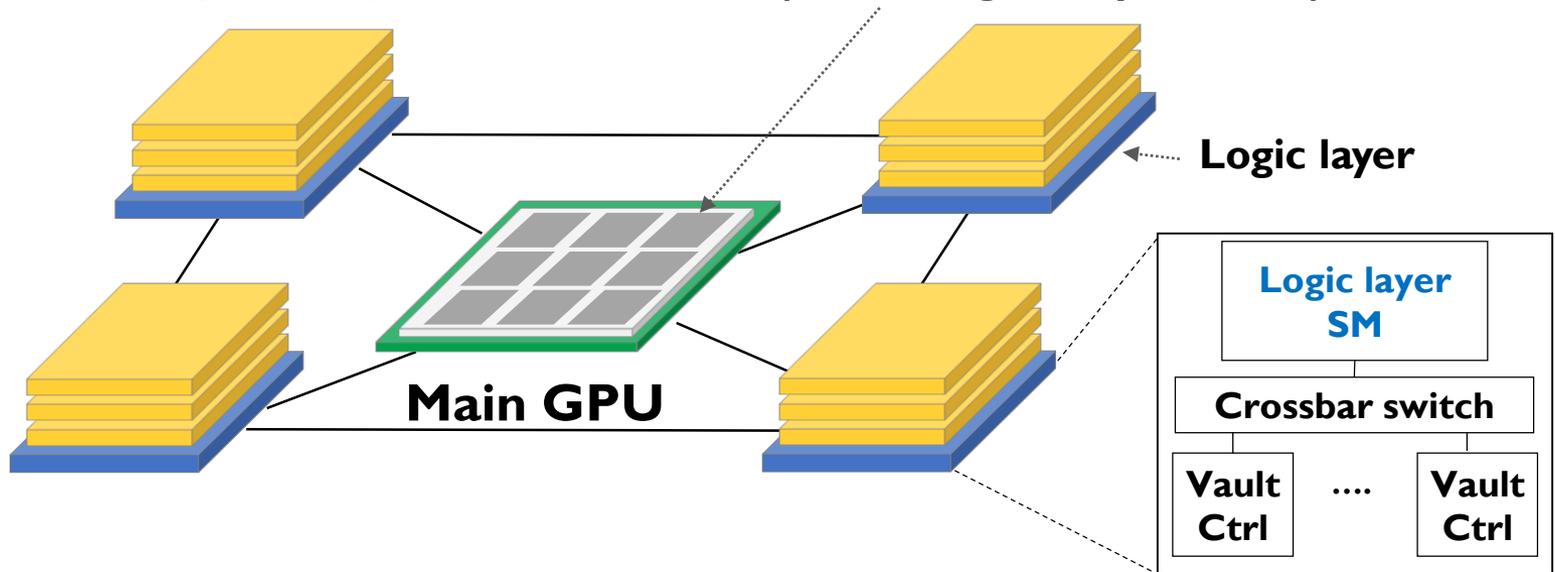
Onur Mutlu<sup>5,1</sup>

# Truly Distributed GPU Processing with PIM?

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

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

**SM (Streaming Multiprocessor)**



# Accelerating GPU Execution with PIM (I)

---

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

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

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

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

## Transparent Offloading and Mapping (TOM):

## Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh<sup>‡</sup> Eiman Ebrahimi<sup>†</sup> Gwangsun Kim\* Niladrish Chatterjee<sup>†</sup> Mike O'Connor<sup>†</sup>  
Nandita Vijaykumar<sup>‡</sup> Onur Mutlu<sup>§‡</sup> Stephen W. Keckler<sup>†</sup>

<sup>‡</sup>Carnegie Mellon University <sup>†</sup>NVIDIA <sup>\*</sup>KAIST <sup>§</sup>ETH Zürich

# Accelerating GPU Execution with PIM (II)

---

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

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

Ashutosh Pattnaik<sup>1</sup>    Xulong Tang<sup>1</sup>    Adwait Jog<sup>2</sup>    Onur Kayiran<sup>3</sup>  
Asit K. Mishra<sup>4</sup>    Mahmut T. Kandemir<sup>1</sup>    Onur Mutlu<sup>5,6</sup>    Chita R. Das<sup>1</sup>

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

# Accelerating Linked Data Structures

---

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

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

Kevin Hsieh<sup>†</sup> Samira Khan<sup>‡</sup> Nandita Vijaykumar<sup>†</sup>  
Kevin K. Chang<sup>†</sup> Amirali Boroumand<sup>†</sup> Saugata Ghose<sup>†</sup> Onur Mutlu<sup>§†</sup>  
<sup>†</sup>*Carnegie Mellon University*   <sup>‡</sup>*University of Virginia*   <sup>§</sup>*ETH Zürich*

# Accelerating Dependent Cache 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

# Two Key Questions in 3D-Stacked PIM

---

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

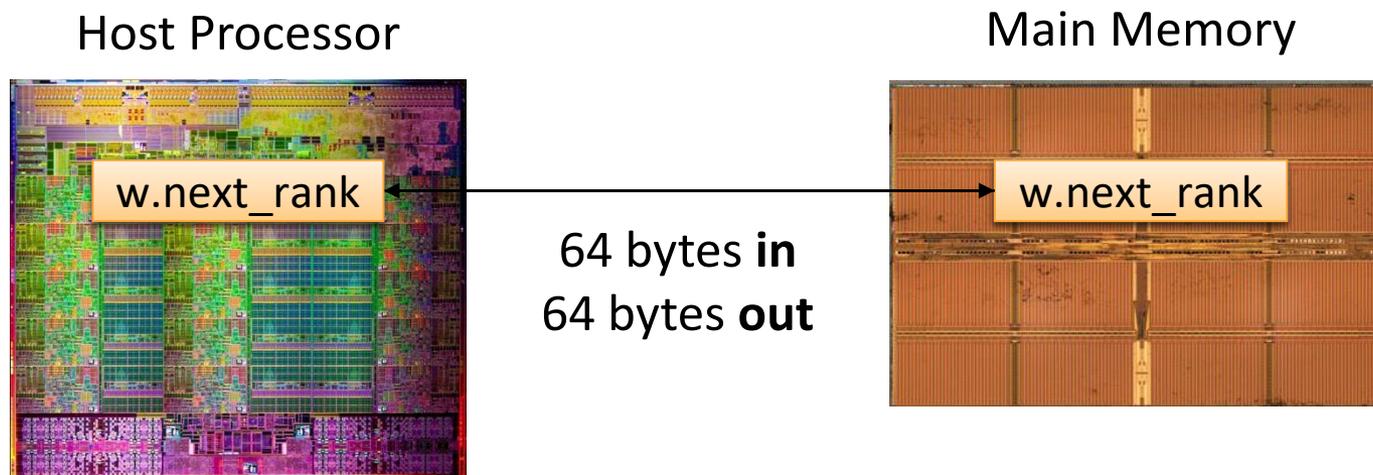
# PEI: PIM-Enabled Instructions (Ideas)

---

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

# Simple PIM Operations as ISA Extensions (II)

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



Conventional Architecture

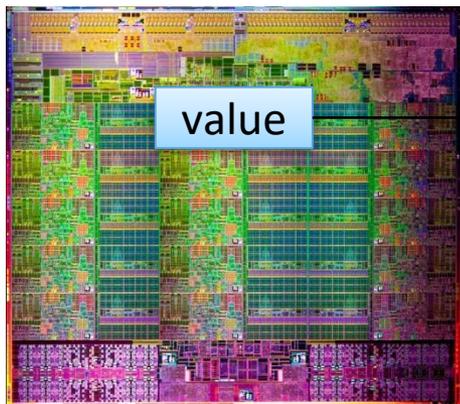
# Simple PIM Operations as ISA Extensions (III)

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

pim.add r1, (r2)

\_\_pim\_add(&w.next\_rank, value);

Host Processor



Main Memory



8 bytes in  
0 bytes out

In-Memory Addition

# PEI: PIM-Enabled Instructions (Example)

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

pim.add r1, (r2)

\_\_pim\_add(&w.next\_rank, value);

pfence

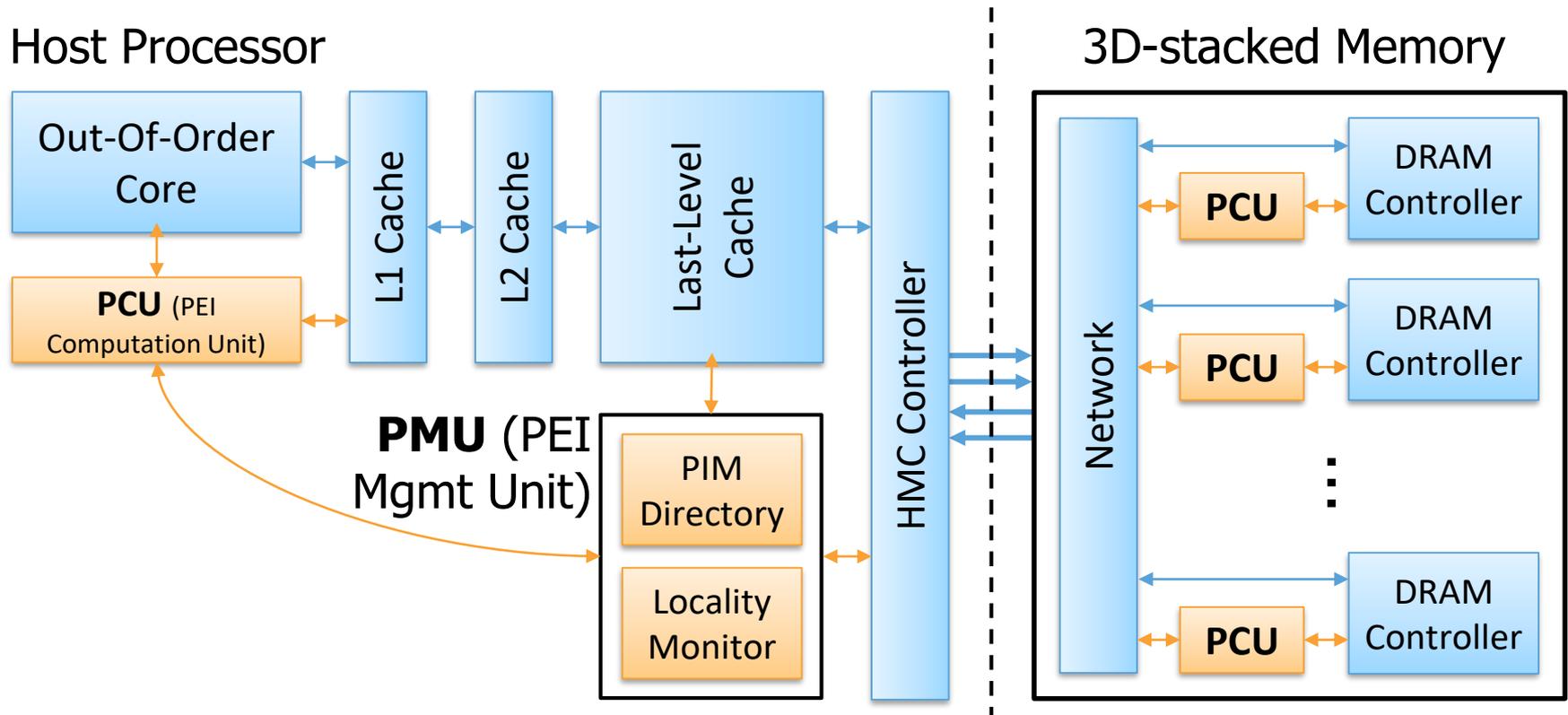
pfence();

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
- 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

**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]

# Simpler PIM: 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<sup>†</sup>   Kiyoung Choi

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

Seoul National University

<sup>†</sup>Carnegie Mellon University

# Automatic Code and Data Mapping

---

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

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

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

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

## Transparent Offloading and Mapping (TOM):

## Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh<sup>‡</sup> Eiman Ebrahimi<sup>†</sup> Gwangsun Kim\* Niladrish Chatterjee<sup>†</sup> Mike O'Connor<sup>†</sup>  
Nandita Vijaykumar<sup>‡</sup> Onur Mutlu<sup>§‡</sup> Stephen W. Keckler<sup>†</sup>

<sup>‡</sup>Carnegie Mellon University   <sup>†</sup>NVIDIA   \*KAIST   <sup>§</sup>ETH Zürich

# Automatic Offloading of Critical Code

---

- 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<sup>\*</sup>, Khubaib<sup>†</sup>, Eiman Ebrahimi<sup>‡</sup>, Onur Mutlu<sup>§</sup>, Yale N. Patt<sup>\*</sup>

<sup>\*</sup>*The University of Texas at Austin*   <sup>†</sup>*Apple*   <sup>‡</sup>*NVIDIA*   <sup>§</sup>*ETH Zürich & Carnegie Mellon University*

# Automatic Offloading of Prefetch Mechanisms

---

- Milad Hashemi, Onur Mutlu, and Yale N. Patt,  
**"Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads"**  
*Proceedings of the 49th International Symposium on Microarchitecture (MICRO)*, Taipei, Taiwan, October 2016.  
[\[Slides \(pptx\) \(pdf\)\]](#) [\[Lightning Session Slides \(pdf\)\]](#) [\[Poster \(pptx\) \(pdf\)\]](#)

## Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads

Milad Hashemi\*, Onur Mutlu<sup>§</sup>, Yale N. Patt\*

\**The University of Texas at Austin*    <sup>§</sup>*ETH Zürich*

# Efficient Automatic Data Coherence Support

---

- 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<sup>†</sup>, Saugata Ghose<sup>†</sup>, Minesh Patel<sup>†</sup>, Hasan Hassan<sup>†§</sup>, Brandon Lucia<sup>†</sup>,  
Kevin Hsieh<sup>†</sup>, Krishna T. Malladi<sup>\*</sup>, Hongzhong Zheng<sup>\*</sup>, and Onur Mutlu<sup>‡†</sup>

<sup>†</sup> *Carnegie Mellon University*   <sup>\*</sup> *Samsung Semiconductor, Inc.*   <sup>§</sup> *TOBB ETÜ*   <sup>‡</sup> *ETH Zürich*

## Fundamentally Energy-Efficient (Data-Centric)

## Computing Architectures

# Fundamentally High-Performance (Data-Centric) Computing Architectures

# Computing Architectures with Minimal Data Movement

# Agenda

---

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
  - Bottom Up: Push from Circuits and Devices
  - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips
  - Exploiting 3D-Stacked Memory
- **How to Enable Adoption of Processing in Memory**
- Conclusion

## How to Enable Adoption of Processing in Memory

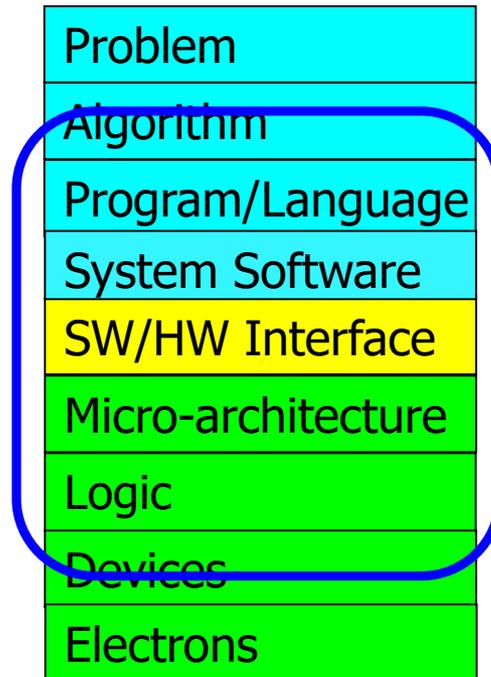
# Barriers to Adoption of PIM

---

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

# We Need to Revisit the Entire Stack

---



# Open Problems: PIM Adoption

---

## Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions

SAUGATA GHOSE, KEVIN HSIEH, AMIRALI BOROUMAND,  
RACHATA AUSAVARUNGNIRUN

Carnegie Mellon University

ONUR MUTLU

ETH Zürich and Carnegie Mellon University

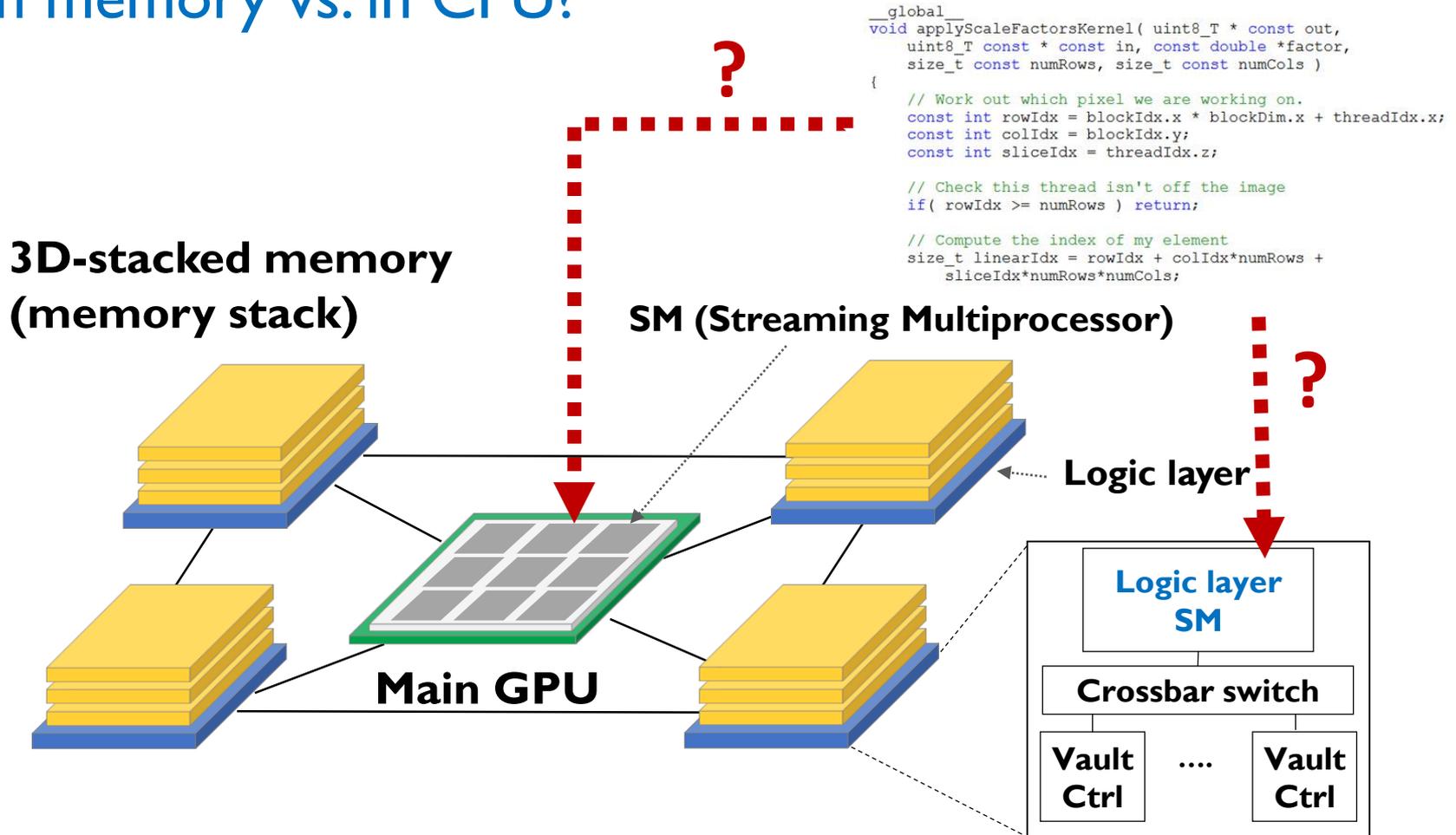
Saugata Ghose, Kevin Hsieh, Amirali Boroumand, Rachata Ausavarungnirun, Onur Mutlu,  
**"Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms,  
Future Research Directions"**

*Invited Book Chapter, to appear in 2018.*

[\[Preliminary arxiv.org version\]](https://arxiv.org/abs/1802.00320)

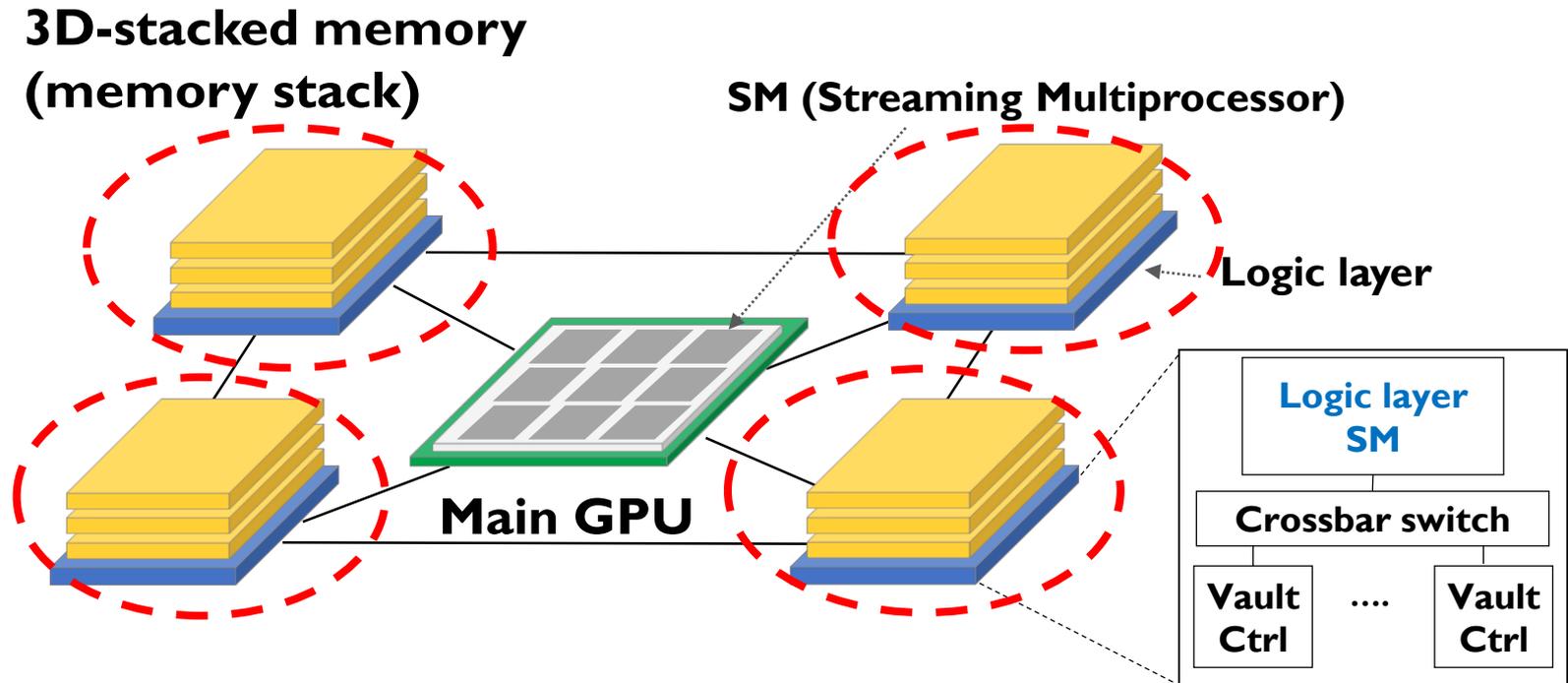
# Key Challenge 1: Code Mapping

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



# Key Challenge 2: Data Mapping

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



# How to Do the Code and Data Mapping?

---

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

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

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

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

## Transparent Offloading and Mapping (TOM):

## Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh<sup>‡</sup> Eiman Ebrahimi<sup>†</sup> Gwangsun Kim\* Niladrish Chatterjee<sup>†</sup> Mike O'Connor<sup>†</sup>  
Nandita Vijaykumar<sup>‡</sup> Onur Mutlu<sup>§‡</sup> Stephen W. Keckler<sup>†</sup>

<sup>‡</sup>Carnegie Mellon University <sup>†</sup>NVIDIA <sup>\*</sup>KAIST <sup>§</sup>ETH Zürich

# How to Schedule Code?

---

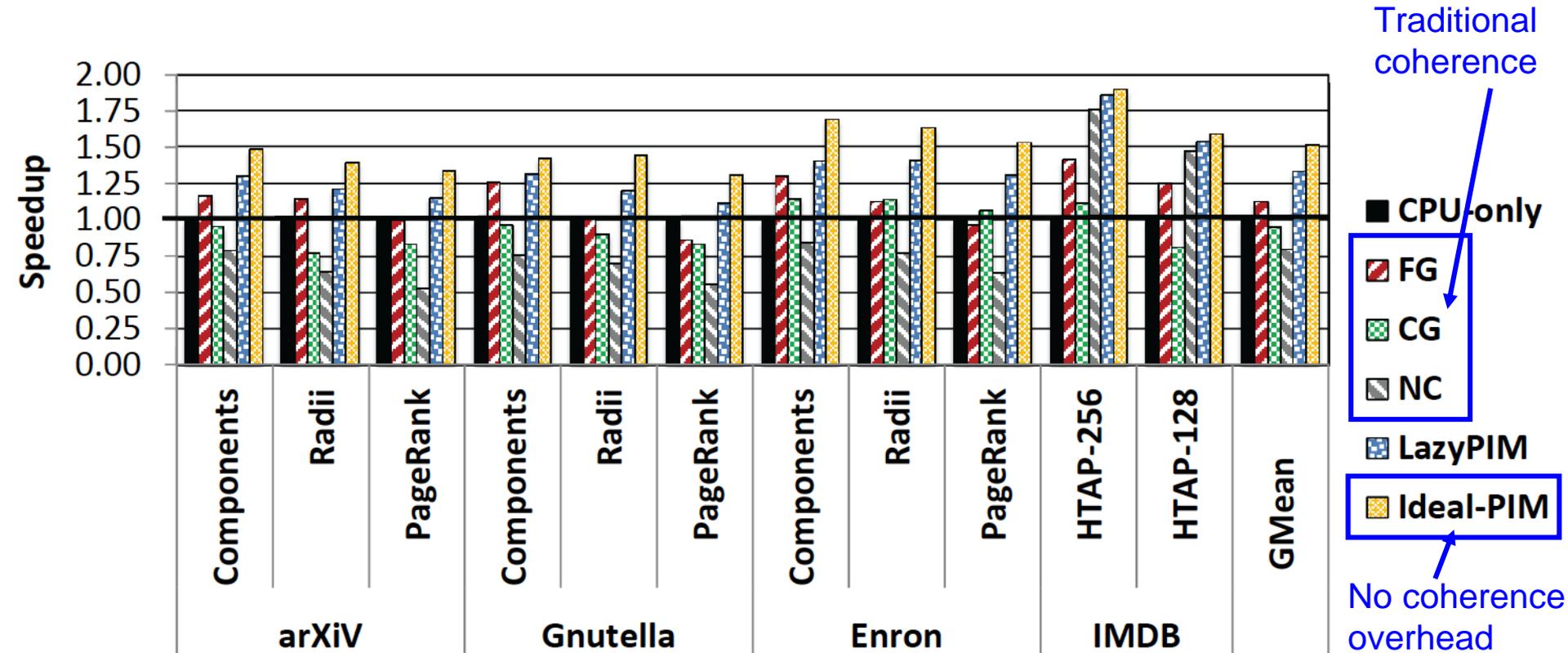
- Ashutosh Pattnaik, Xulong Tang, Adwait Jog, Onur Kayiran, Asit K. Mishra, Mahmut T. Kandemir, Onur Mutlu, and Chita R. Das, **"Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities"**  
*Proceedings of the 25th International Conference on Parallel Architectures and Compilation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

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

Ashutosh Pattnaik<sup>1</sup>    Xulong Tang<sup>1</sup>    Adwait Jog<sup>2</sup>    Onur Kayiran<sup>3</sup>  
Asit K. Mishra<sup>4</sup>    Mahmut T. Kandemir<sup>1</sup>    Onur Mutlu<sup>5,6</sup>    Chita R. Das<sup>1</sup>

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

# Challenge: Coherence for Hybrid CPU-PIM Apps



# How to Maintain Coherence?

---

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

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

Amirali Boroumand<sup>†</sup>, Saugata Ghose<sup>†</sup>, Minesh Patel<sup>†</sup>, Hasan Hassan<sup>†§</sup>, Brandon Lucia<sup>†</sup>,  
Kevin Hsieh<sup>†</sup>, Krishna T. Malladi<sup>\*</sup>, Hongzhong Zheng<sup>\*</sup>, and Onur Mutlu<sup>‡†</sup>

<sup>†</sup> *Carnegie Mellon University*   <sup>\*</sup> *Samsung Semiconductor, Inc.*   <sup>§</sup> *TOBB ETÜ*   <sup>‡</sup> *ETH Zürich*

# How to Support Virtual Memory?

---

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

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

Kevin Hsieh<sup>†</sup> Samira Khan<sup>‡</sup> Nandita Vijaykumar<sup>†</sup>

Kevin K. Chang<sup>†</sup> Amirali Boroumand<sup>†</sup> Saugata Ghose<sup>†</sup> Onur Mutlu<sup>§†</sup>

<sup>†</sup>*Carnegie Mellon University*   <sup>‡</sup>*University of Virginia*   <sup>§</sup>*ETH Zürich*

# How to Design Data Structures for PIM?

---

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

## Concurrent Data Structures for Near-Memory Computing

Zhiyu Liu

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

Irina Calciu

VMware Research Group  
icalciu@vmware.com

Maurice Herlihy

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

Onur Mutlu

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

# Simulation Infrastructures for PIM

---

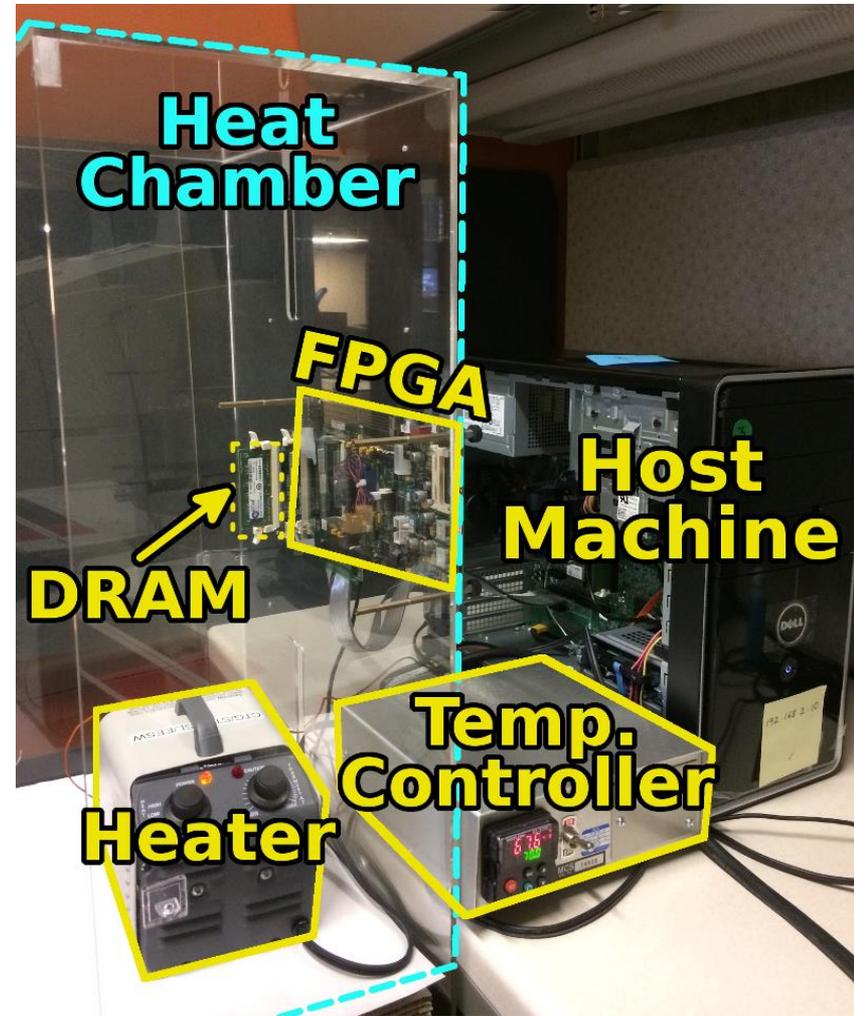
- **Ramulator** extended for PIM
  - Flexible and extensible DRAM simulator
  - Can model many different memory standards and proposals
  - Kim+, “**Ramulator: A Flexible and Extensible DRAM Simulator**”, IEEE CAL 2015.
  - <https://github.com/CMU-SAFARI/ramulator>

## Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim<sup>1</sup>    Weikun Yang<sup>1,2</sup>    Onur Mutlu<sup>1</sup>  
<sup>1</sup>Carnegie Mellon University    <sup>2</sup>Peking University

# An FPGA-based Test-bed for PIM?

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



# Simulation Infrastructures for PIM (in SSDs)

---

- Arash Tavakkol, Juan Gomez-Luna, Mohammad Sadrosadati, Saugata Ghose, and Onur Mutlu,  
**"MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices"**  
*Proceedings of the 16th USENIX Conference on File and Storage Technologies (FAST), Oakland, CA, USA, February 2018.*  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Source Code](#)]

## **MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices**

Arash Tavakkol<sup>†</sup>, Juan Gómez-Luna<sup>†</sup>, Mohammad Sadrosadati<sup>†</sup>, Saugata Ghose<sup>‡</sup>, Onur Mutlu<sup>†‡</sup>  
<sup>†</sup>*ETH Zürich*                      <sup>‡</sup>*Carnegie Mellon University*

# New Applications and Use Cases for PIM

---

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

## GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies

Jeremie S. Kim<sup>1,6\*</sup>, Damla Senol Cali<sup>1</sup>, Hongyi Xin<sup>2</sup>, Donghyuk Lee<sup>3</sup>, Saugata Ghose<sup>1</sup>, Mohammed Alser<sup>4</sup>, Hasan Hassan<sup>6</sup>, Oguz Ergin<sup>5</sup>, Can Alkan<sup>4\*</sup> and Onur Mutlu<sup>6,1\*</sup>

From The Sixteenth Asia Pacific Bioinformatics Conference 2018  
Yokohama, Japan. 15-17 January 2018

## Genome Read In-Memory (GRIM) Filter: Fast Seed Location Filtering in DNA Read Mapping using Processing-in-Memory Technologies

**Jeremie Kim,**

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

**Carnegie Mellon**



**ETH zürich**

# Executive Summary

---

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

# Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

**Amirali Boroumand**

Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun,  
Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela,  
Allan Knies, Parthasarathy Ranganathan, Onur Mutlu

**SAFARI**

**Carnegie Mellon**

**Google**



SEOUL  
NATIONAL  
UNIVERSITY

**ETH** zürich

# Open Problems: PIM Adoption

---

## Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions

SAUGATA GHOSE, KEVIN HSIEH, AMIRALI BOROUMAND,  
RACHATA AUSAVARUNGNIRUN

Carnegie Mellon University

ONUR MUTLU

ETH Zürich and Carnegie Mellon University

Saugata Ghose, Kevin Hsieh, Amirali Boroumand, Rachata Ausavarungnirun, Onur Mutlu,  
**"Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms,  
Future Research Directions"**

*Invited Book Chapter, to appear in 2018.*

[[Preliminary arxiv.org version](https://arxiv.org/abs/1802.00320)]

# Enabling the Paradigm Shift

# Computer Architecture Today

---

- You can revolutionize the way computers are built, if you understand both the hardware and the software (and change each accordingly)
- You can invent new paradigms for computation, communication, and storage
- Recommended book: Thomas Kuhn, “[The Structure of Scientific Revolutions](#)” (1962)
  - Pre-paradigm science: no clear consensus in the field
  - Normal science: dominant theory used to explain/improve things (business as usual); exceptions considered anomalies
  - Revolutionary science: underlying assumptions re-examined

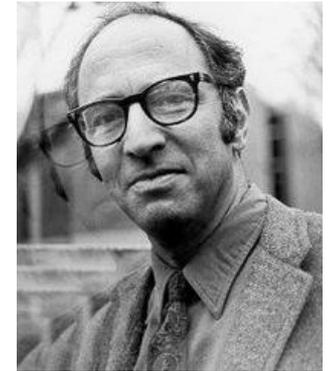
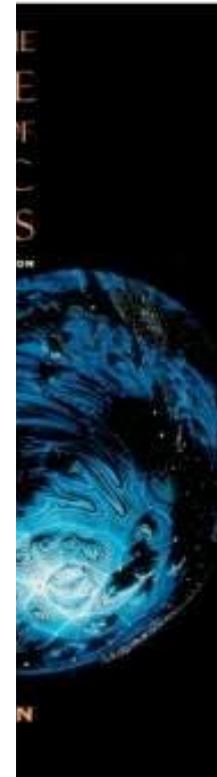
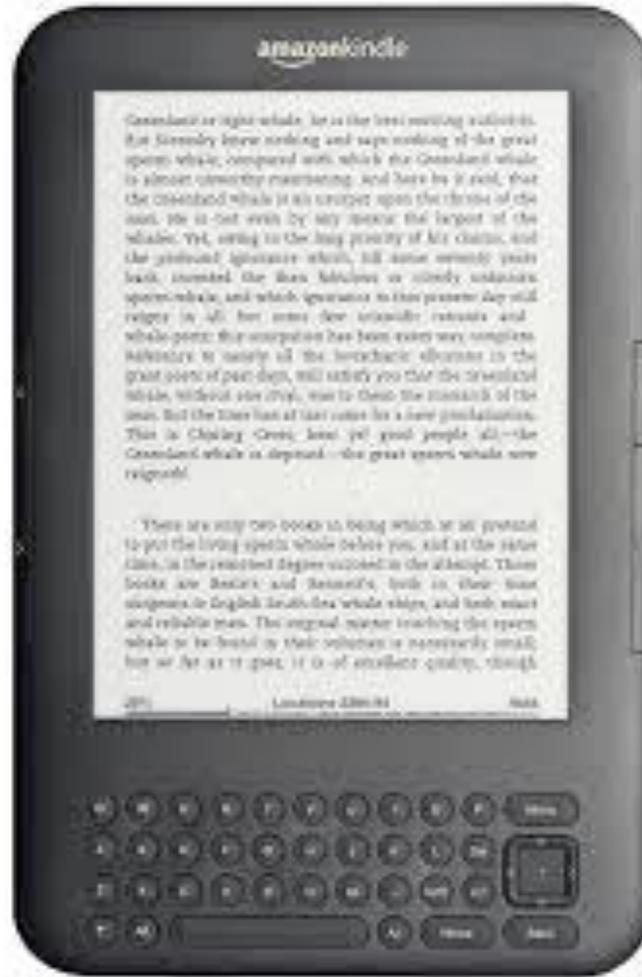
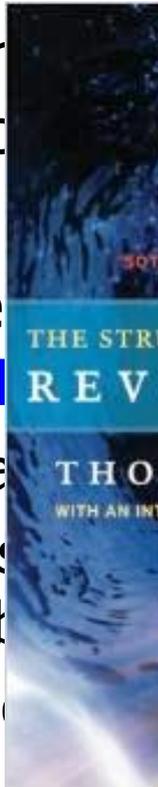
# Computer Architecture Today

- You can revolutionize the way computers are built, if you understand both hardware and software (and change each accordingly)

- You can improve communication

- Recommendation: Scientific Illustration

- Pre-prepare things (books)
- Normal science
- Revolutionary science



ure of  
field  
improve  
anomalies  
examined

# Agenda

---

- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
  - Bottom Up: Push from Circuits and Devices
  - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips
  - Exploiting 3D-Stacked Memory
- How to Enable Adoption of Processing in Memory
- Conclusion

# Four Key Directions

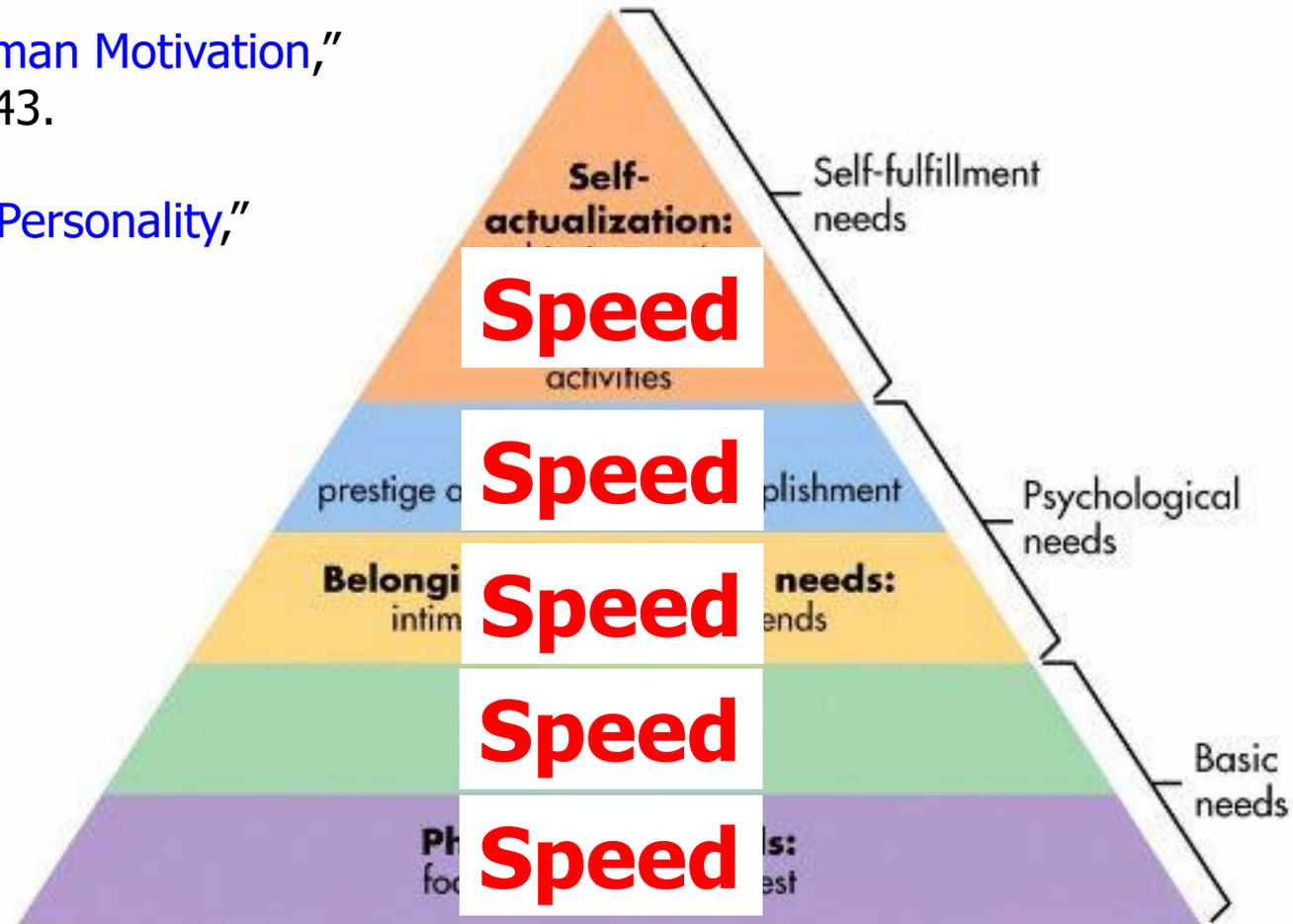
---

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

# Maslow's Hierarchy of Needs, A Third Time

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

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



## Fundamentally Energy-Efficient (Data-Centric)

## Computing Architectures

Fundamentally

Low-Latency

(Data-Centric)

Computing Architectures

# Computing Architectures with Minimal Data Movement

# One Important Takeaway

---

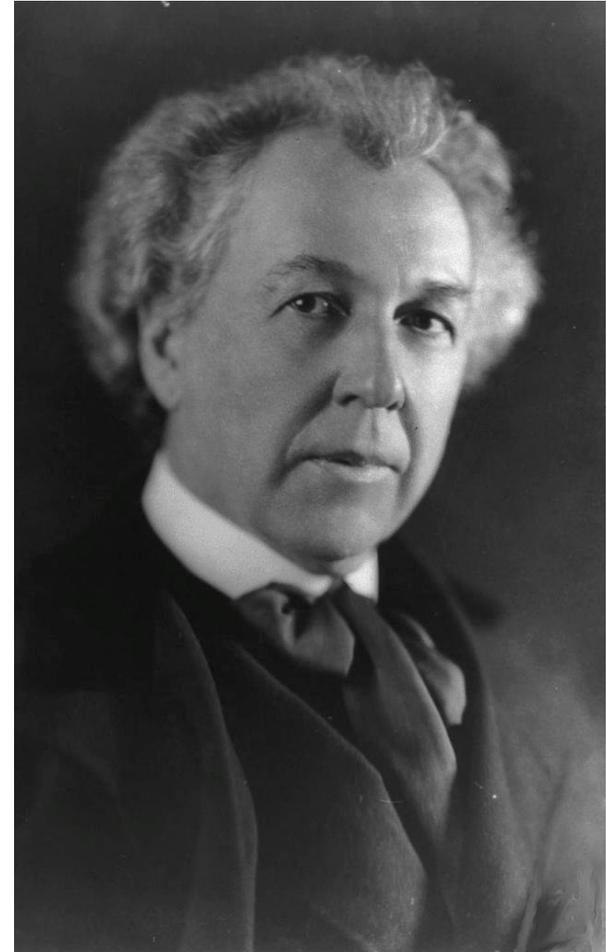
**Main Memory Needs  
Intelligent Controllers**

# Concluding Remarks

# A Quote from A Famous Architect

---

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



# Precedent-Based Design?

---

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



# Principled Design

---

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





# The Overarching Principle

---

## Organic architecture

---

From Wikipedia, the free encyclopedia

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

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

# Another Example: Precedent-Based Design

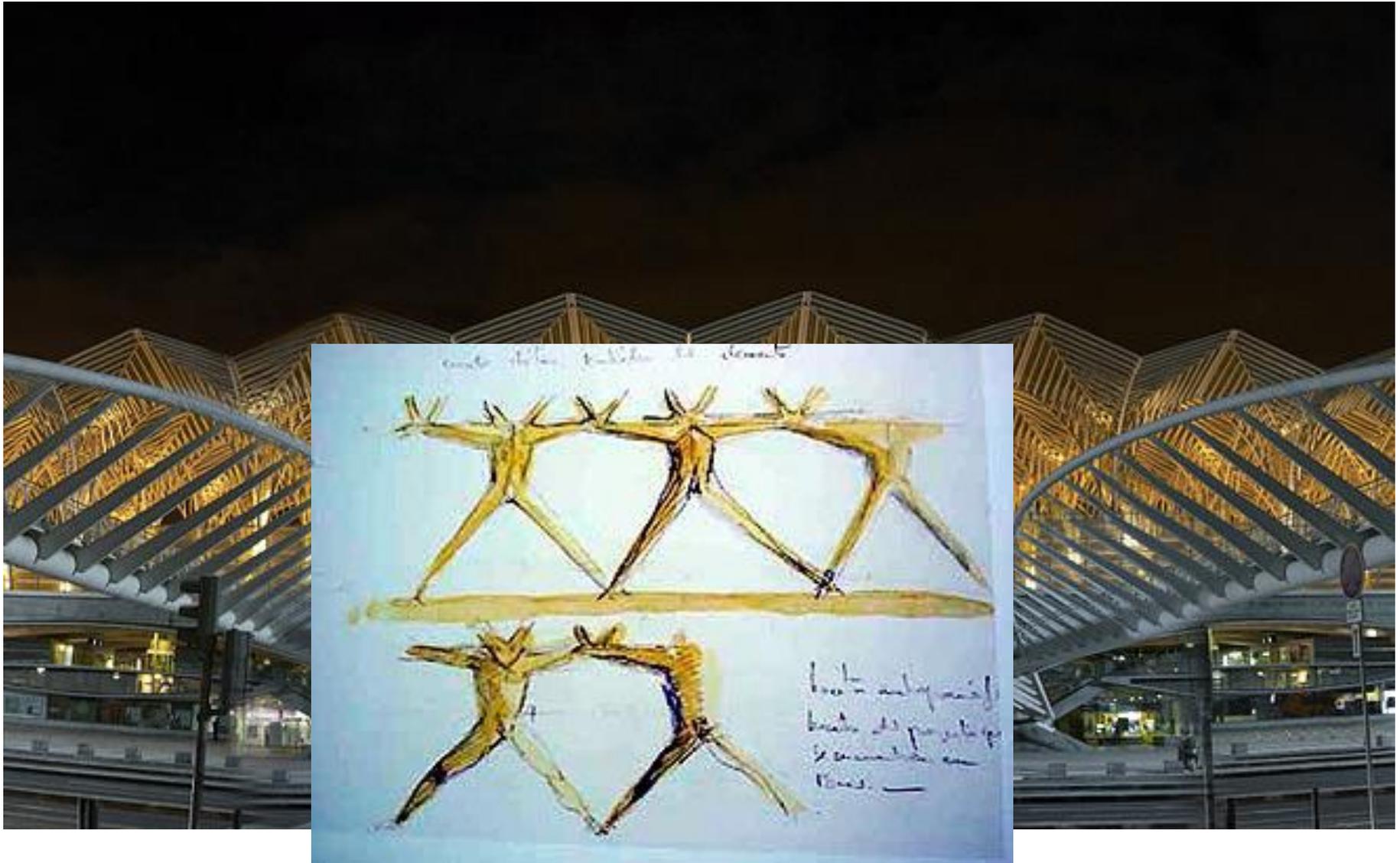
---



# Principled Design



# Another Principled Design



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

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

# Another 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,

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

# The Overarching Principle

---

## Zoomorphic architecture

---

From Wikipedia, the free encyclopedia

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

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

# Overarching Principle for Computing?

---



# Concluding Remarks

---

- It is time to design **principled system architectures** to solve the **memory problem**
- Design complete systems to be balanced, high-performance, and energy-efficient, i.e., **data-centric (or memory-centric)**
- Enable computation capability inside and close to memory
- **This** can
  - Lead to **orders-of-magnitude** improvements
  - **Enable new applications & computing platforms**
  - **Enable better understanding of nature**
  - ...

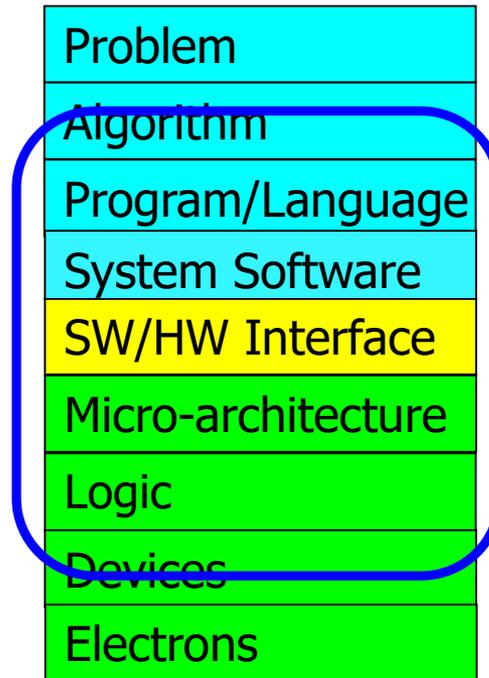
# The Future of Processing in Memory is Bright

---

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

Can enable:

- Orders of magnitude improvements
- New applications and computing systems



Yet, we have to

- Think across the stack
- Design enabling systems

# If In Doubt, See Other Doubtful Technologies

---

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



*Proceedings of the IEEE, Sept. 2017*

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

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

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

# For Some Open Problems, See

---

## **Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions**

SAUGATA GHOSE, KEVIN HSIEH, AMIRALI BOROUMAND,  
RACHATA AUSAVARUNGNIRUN

Carnegie Mellon University

ONUR MUTLU

ETH Zürich and Carnegie Mellon University

Saugata Ghose, Kevin Hsieh, Amirali Boroumand, Rachata Ausavarungnirun, Onur Mutlu,  
**"Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms,  
Future Research Directions"**

*Invited Book Chapter, to appear in 2018.*

[[Preliminary arxiv.org version](https://arxiv.org/abs/1802.00320)]

# Accelerated Memory Course (~6.5 hours)

---

## ■ ACACES 2018

- ❑ Memory Systems and Memory-Centric Computing Systems
- ❑ Taught by Onur Mutlu July 9-13, 2018
- ❑ ~6.5 hours of lectures

## ■ Website for the Course including Videos, Slides, Papers

- ❑ <https://people.inf.ethz.ch/omutlu/acaces2018.html>
- ❑ <https://www.youtube.com/playlist?list=PL5Q2soXY2Zi-HXxomthrpDpMJm05P6J9x>

## ■ All Papers are at:

- ❑ <https://people.inf.ethz.ch/omutlu/projects.htm>
- ❑ Final lecture notes and readings (for all topics)

# Processing Data Where It Makes Sense in Modern Computing Systems: Enabling In-Memory Computation

Onur Mutlu

[omutlu@gmail.com](mailto:omutlu@gmail.com)

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

4 October 2018

HNEP 2018 Keynote



**SAFARI**

**ETH** zürich

**Carnegie Mellon**

Slides Not Covered  
But Could Be Useful

# Readings, Videos, Reference Materials

# Accelerated Memory Course (~6.5 hours)

---

## ■ ACACES 2018

- ❑ Memory Systems and Memory-Centric Computing Systems
- ❑ Taught by Onur Mutlu July 9-13, 2018
- ❑ ~6.5 hours of lectures

## ■ Website for the Course including Videos, Slides, Papers

- ❑ <https://people.inf.ethz.ch/omutlu/acaces2018.html>
- ❑ <https://www.youtube.com/playlist?list=PL5Q2soXY2Zi-HXxomthrpDpMJm05P6J9x>

## ■ All Papers are at:

- ❑ <https://people.inf.ethz.ch/omutlu/projects.htm>
- ❑ Final lecture notes and readings (for all topics)

# Reference Overview Paper I

---

## **Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions**

SAUGATA GHOSE, KEVIN HSIEH, AMIRALI BOROUMAND,  
RACHATA AUSAVARUNGNIRUN

Carnegie Mellon University

ONUR MUTLU

ETH Zürich and Carnegie Mellon University

Saugata Ghose, Kevin Hsieh, Amirali Boroumand, Rachata Ausavarungnirun, Onur Mutlu,  
**"Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms,  
Future Research Directions"**

*Invited Book Chapter, to appear in 2018.*

[[Preliminary arxiv.org version](https://arxiv.org/abs/1802.00320)]

# Reference Overview Paper II

---

- 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<sup>1</sup>, Lavanya Subramanian<sup>1</sup>*

# Reference Overview Paper III

---

- 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](mailto:onur.mutlu@inf.ethz.ch)  
<https://people.inf.ethz.ch/omutlu>

# Reference Overview Paper IV

---

- 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/>



*Proceedings of the IEEE, Sept. 2017*

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

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

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

# Related Videos and Course Materials (I)

---

- **Undergraduate Computer Architecture Course Lecture Videos (2015, 2014, 2013)**
- **Undergraduate Computer Architecture Course Materials (2015, 2014, 2013)**
  
- **Graduate Computer Architecture Course Lecture Videos (2017, 2015, 2013)**
- **Graduate Computer Architecture Course Materials (2017, 2015, 2013)**
  
- **Parallel Computer Architecture Course Materials (Lecture Videos)**

# Related Videos and Course Materials (II)

---

- **Freshman Digital Circuits and Computer Architecture Course Lecture Videos (2018, 2017)**
- **Freshman Digital Circuits and Computer Architecture Course Materials (2018)**
  
- **Memory Systems Short Course Materials (Lecture Video on Main Memory and DRAM Basics)**

# Some Open Source Tools (I)

---

- Rowhammer – Program to Induce RowHammer Errors
  - <https://github.com/CMU-SAFARI/rowhammer>
- Ramulator – Fast and Extensible DRAM Simulator
  - <https://github.com/CMU-SAFARI/ramulator>
- MemSim – Simple Memory Simulator
  - <https://github.com/CMU-SAFARI/memsim>
- NOCulator – Flexible Network-on-Chip Simulator
  - <https://github.com/CMU-SAFARI/NOCulator>
- SoftMC – FPGA-Based DRAM Testing Infrastructure
  - <https://github.com/CMU-SAFARI/SoftMC>
- Other open-source software from my group
  - <https://github.com/CMU-SAFARI/>
  - <http://www.ece.cmu.edu/~safari/tools.html>

# Some Open Source Tools (II)

---

- MQSim – A Fast Modern SSD Simulator
  - <https://github.com/CMU-SAFARI/MQSim>
- Mosaic – GPU Simulator Supporting Concurrent Applications
  - <https://github.com/CMU-SAFARI/Mosaic>
- IMPICA – Processing in 3D-Stacked Memory Simulator
  - <https://github.com/CMU-SAFARI/IMPICA>
- SMLA – Detailed 3D-Stacked Memory Simulator
  - <https://github.com/CMU-SAFARI/SMLA>
- HWASim – Simulator for Heterogeneous CPU-HWA Systems
  - <https://github.com/CMU-SAFARI/HWASim>
- Other open-source software from my group
  - <https://github.com/CMU-SAFARI/>
  - <http://www.ece.cmu.edu/~safari/tools.html>

# More Open Source Tools (III)

- A lot more open-source software from my group
  - <https://github.com/CMU-SAFARI/>
  - <http://www.ece.cmu.edu/~safari/tools.html>

**SAFARI** SAFARI Research Group at ETH Zurich and Carnegie Mellon University

Site for source code and tools distribution from SAFARI Research Group at ETH Zurich and Carnegie Mellon University.

ETH Zurich and Carnegi... <http://www.ece.cmu.ed...> [omutlu@gmail.com](mailto:omutlu@gmail.com)

Repositories 30 People 27 Teams 1 Projects 0 Settings

Search repositories... Type: All Language: All Customize pinned repositories [New](#)

**MQSim**

MQSim is a fast and accurate simulator modeling the performance of modern multi-queue (MQ) SSDs as well as traditional SATA based SSDs. MQSim faithfully models new high-bandwidth protocol implementations, steady-state SSD conditions, and the full end-to-end latency of requests in modern SSDs. It is described in detail in the FAST 2018 paper by A...

● C++ ★ 14 🍴 14 🏢 MIT Updated 8 days ago

Top languages

- C++ ● C ● C# ● AGS Script
- Verilog

Most used topics [Manage](#)

- dram reliability

# Referenced Papers

---

- All are available at

**<https://people.inf.ethz.ch/omutlu/projects.htm>**

**<http://scholar.google.com/citations?user=7XyGUGkAAAAJ&hl=en>**

**<https://people.inf.ethz.ch/omutlu/acaces2018.html>**

# Ramulator: A Fast and Extensible DRAM Simulator

**[IEEE Comp Arch Letters'15]**

# Ramulator Motivation

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

<i>Segment</i>	<i>DRAM Standards &amp; Architectures</i>
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLD RAM3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

Table 1. Landscape of DRAM-based memory

# Ramulator

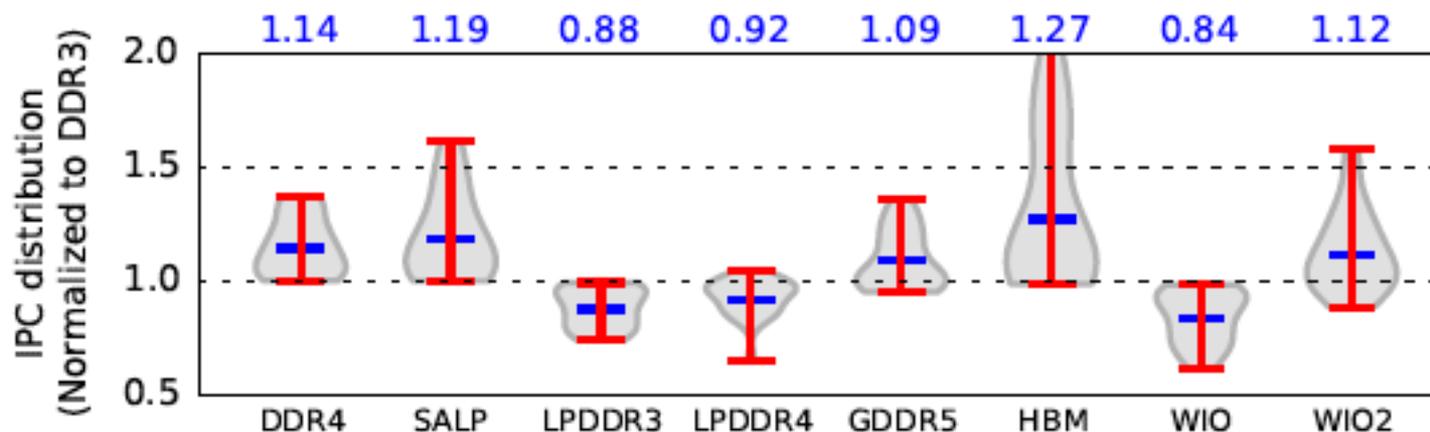
- Provides out-of-the box support for many DRAM standards:
  - DDR3/4, LPDDR3/4, GDDR5, WIO1/2, HBM, plus new proposals (SALP, AL-DRAM, TLDRAM, RowClone, and SARP)
- ~2.5X faster than fastest open-source simulator
- Modular and extensible to different standards

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

Table 3. Comparison of five simulators using two traces

# Case Study: Comparison of DRAM Standards

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



Across 22 workloads, simple CPU model

Figure 2. Performance comparison of DRAM standards

# Ramulator Paper and Source Code

---

- Yoongu Kim, Weikun Yang, and Onur Mutlu,  
**"Ramulator: A Fast and Extensible DRAM Simulator"**  
*IEEE Computer Architecture Letters (CAL)*, March 2015.  
[\[Source Code\]](#)
- Source code is released under the liberal MIT License
  - <https://github.com/CMU-SAFARI/ramulator>

## Ramulator: A Fast and Extensible DRAM Simulator

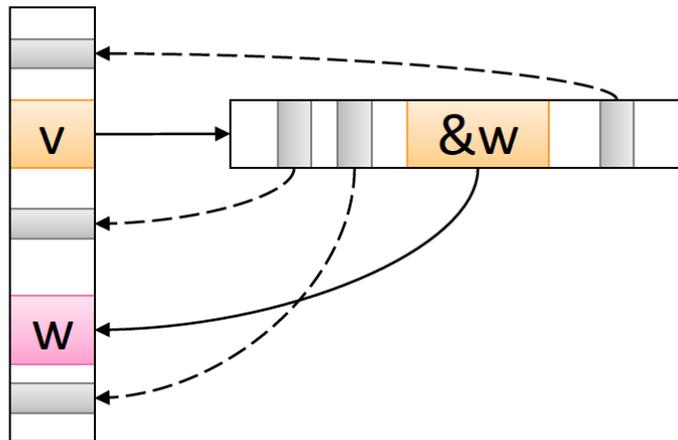
Yoongu Kim<sup>1</sup>      Weikun Yang<sup>1,2</sup>      Onur Mutlu<sup>1</sup>  
<sup>1</sup>Carnegie Mellon University      <sup>2</sup>Peking University

# Tesseract: Extra Slides

# Communications In Tesseract (I)

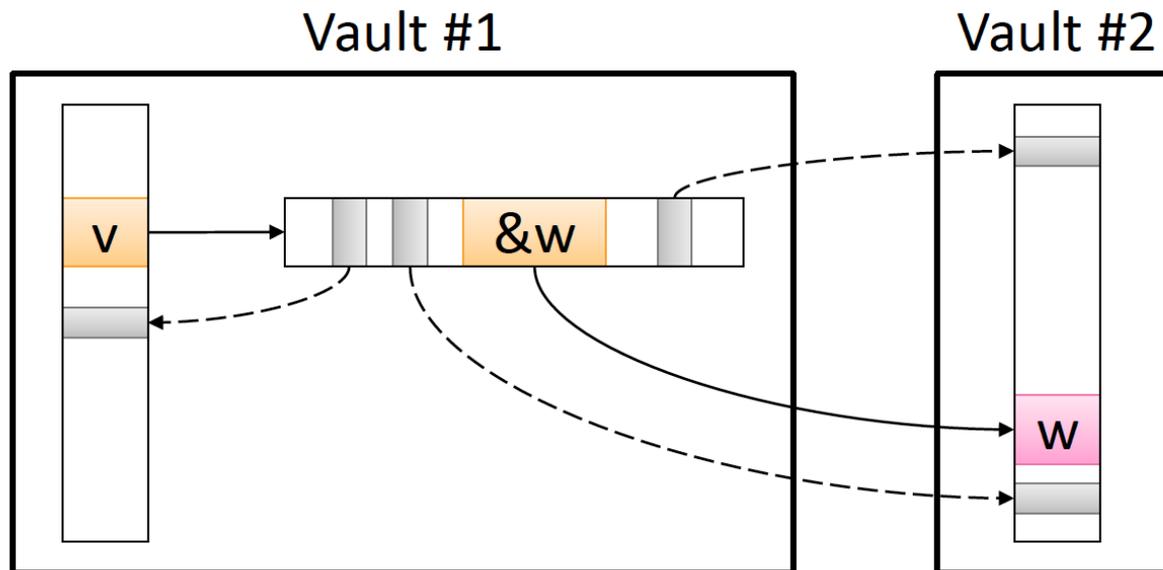
---

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



# Communications In Tesseract (II)

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

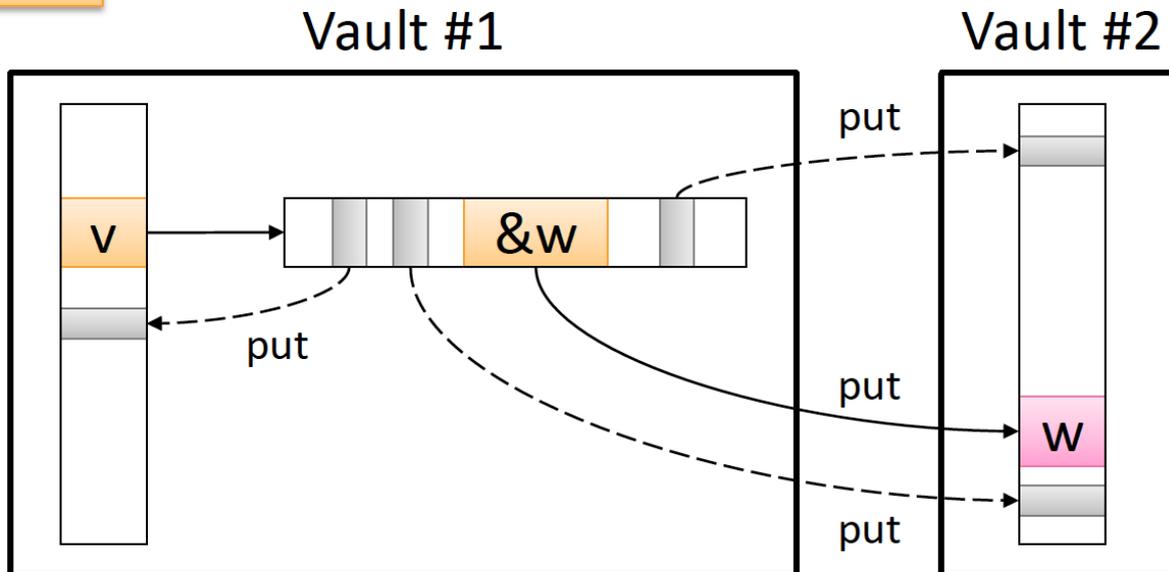


# Communications In Tesseract (III)

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

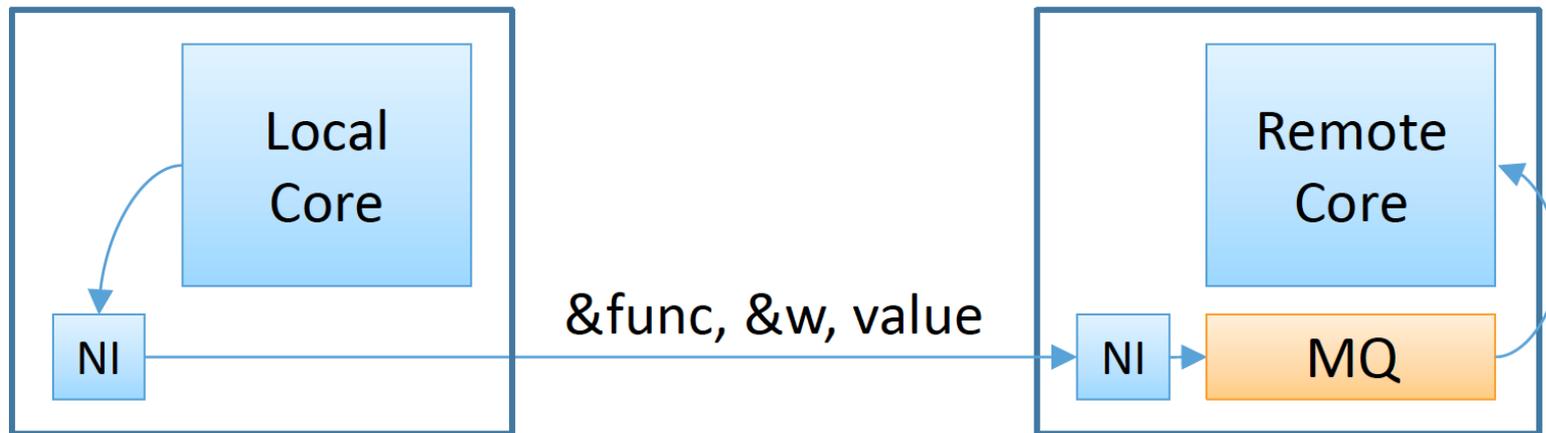
**Non-blocking Remote Function Call**

Can be **delayed** until the nearest barrier



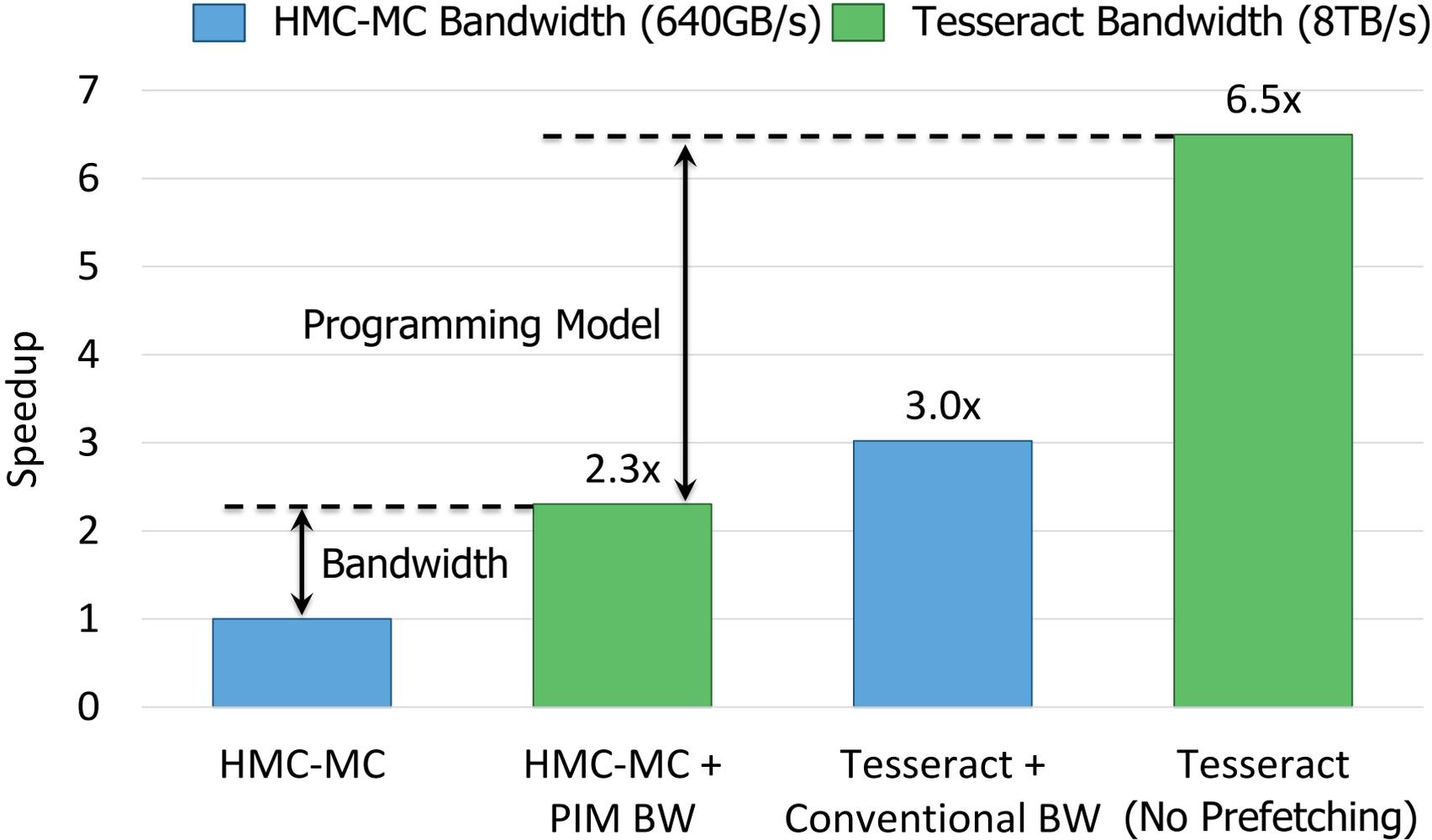
# Remote Function Call (Non-Blocking)

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



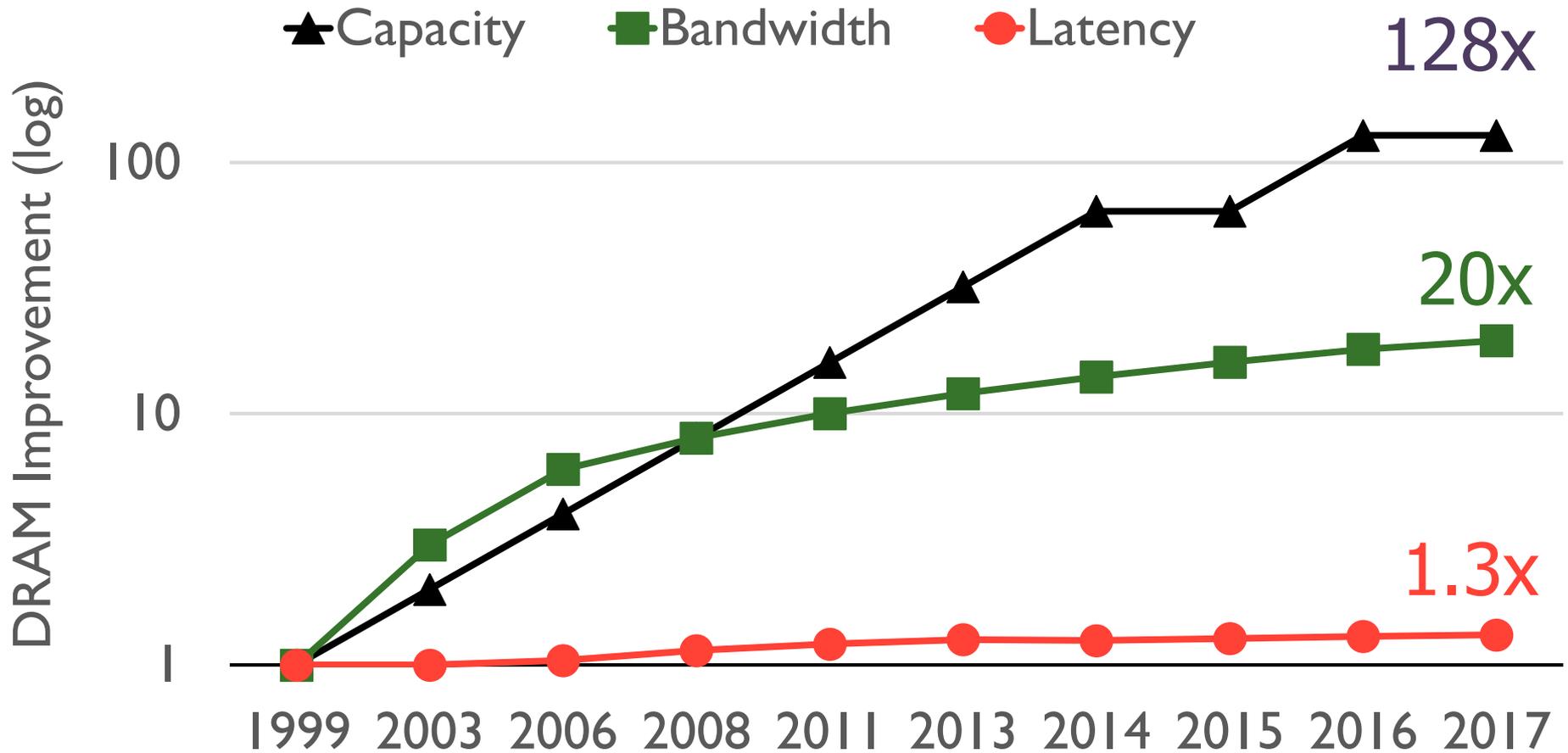
```
put(w.id, function() { w.next_rank += value; })
```

# Effect of Bandwidth & Programming Model



# Reducing Memory Latency

# Main Memory Latency Lags Behind



Memory latency remains almost constant

# A Closer Look ...

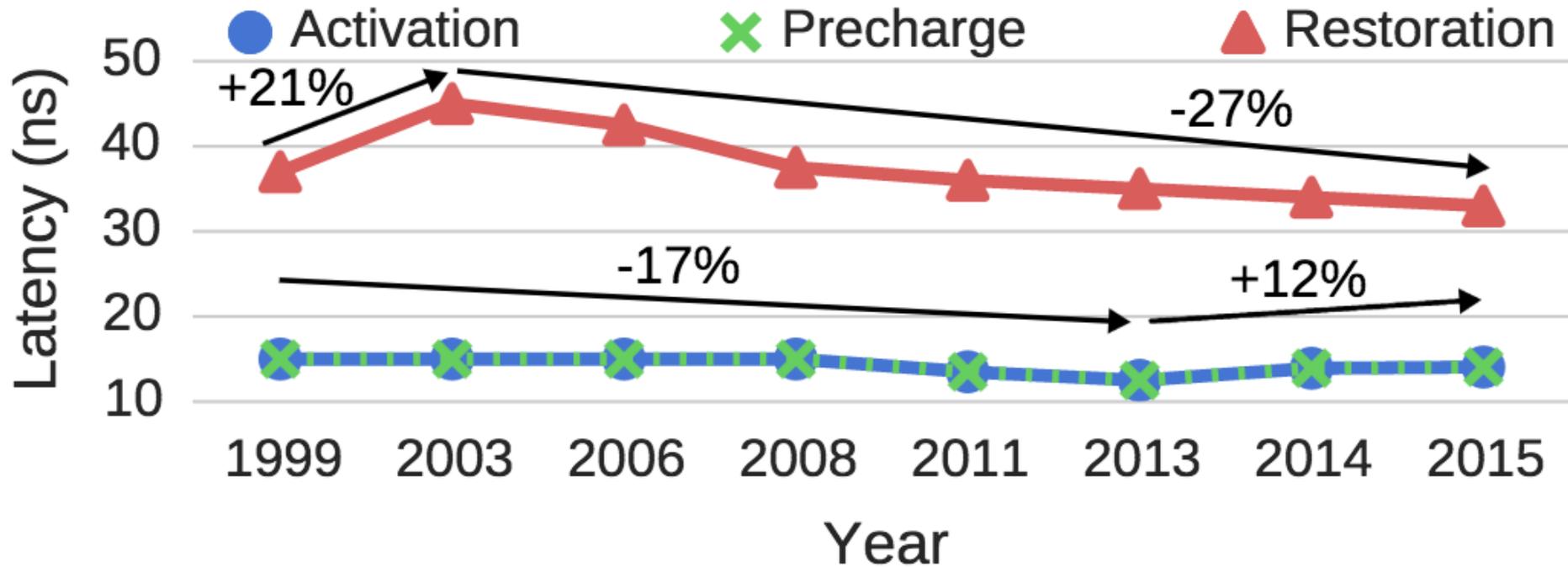
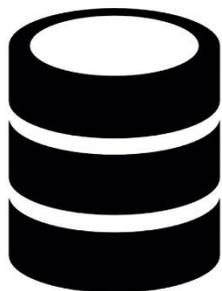


Figure 1: DRAM latency trends over time [20, 21, 23, 51].

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

# DRAM Latency Is Critical for Performance

---



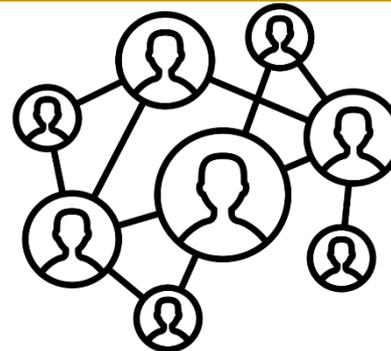
## In-memory Databases

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



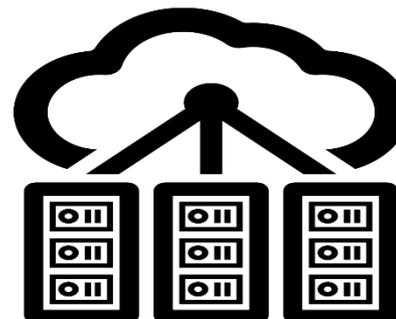
## In-Memory Data Analytics

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



## Graph/Tree Processing

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

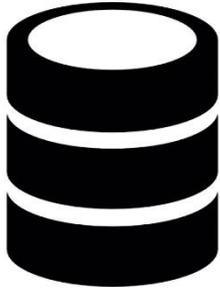


## Datacenter Workloads

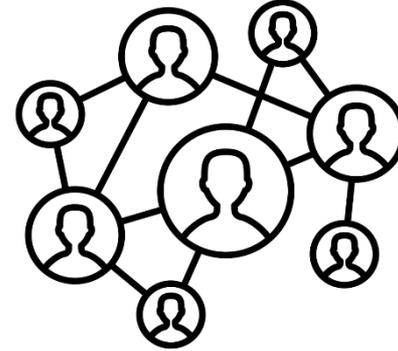
[Kanev+ (Google), ISCA'15]

# DRAM Latency Is Critical for Performance

---



## In-memory Databases



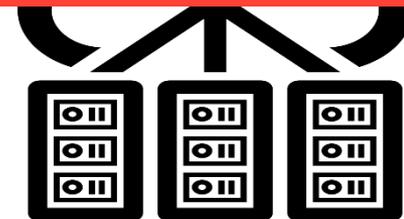
## Graph/Tree Processing

Long memory latency → performance bottleneck



## In-Memory Data Analytics

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



## Datacenter Workloads

[Kanev+ (Google), ISCA'15]

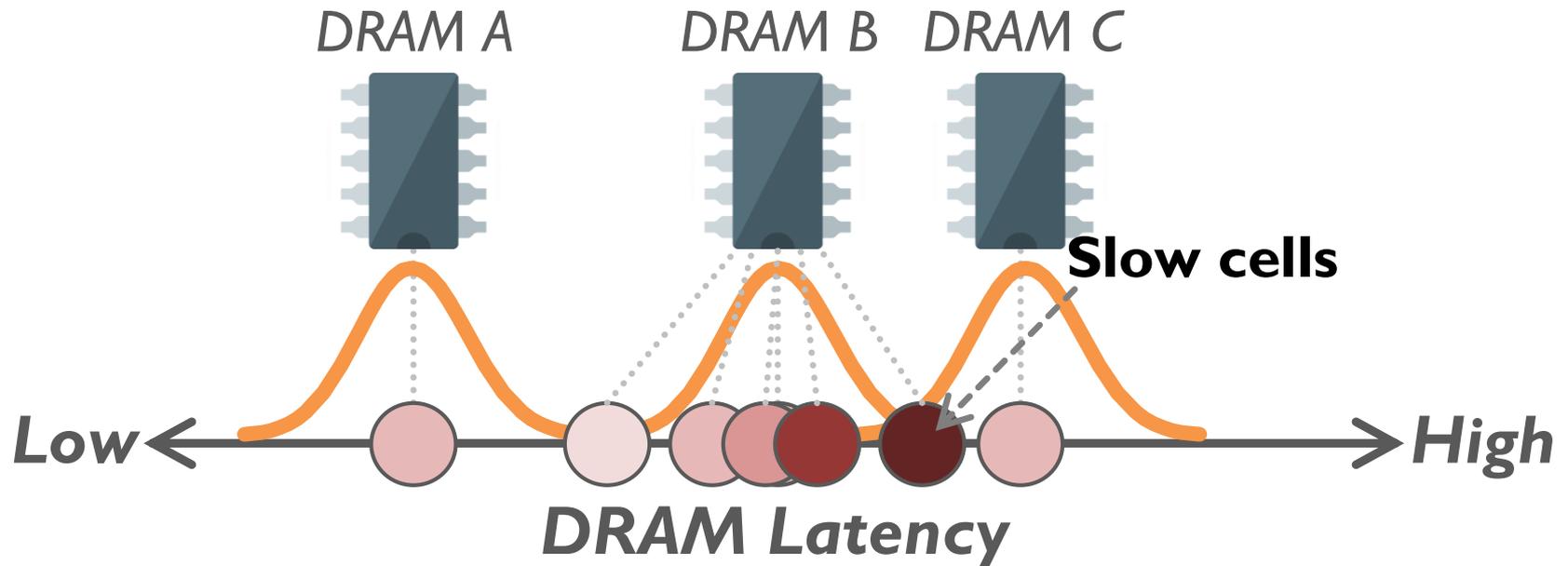
# Why the Long Latency?

---

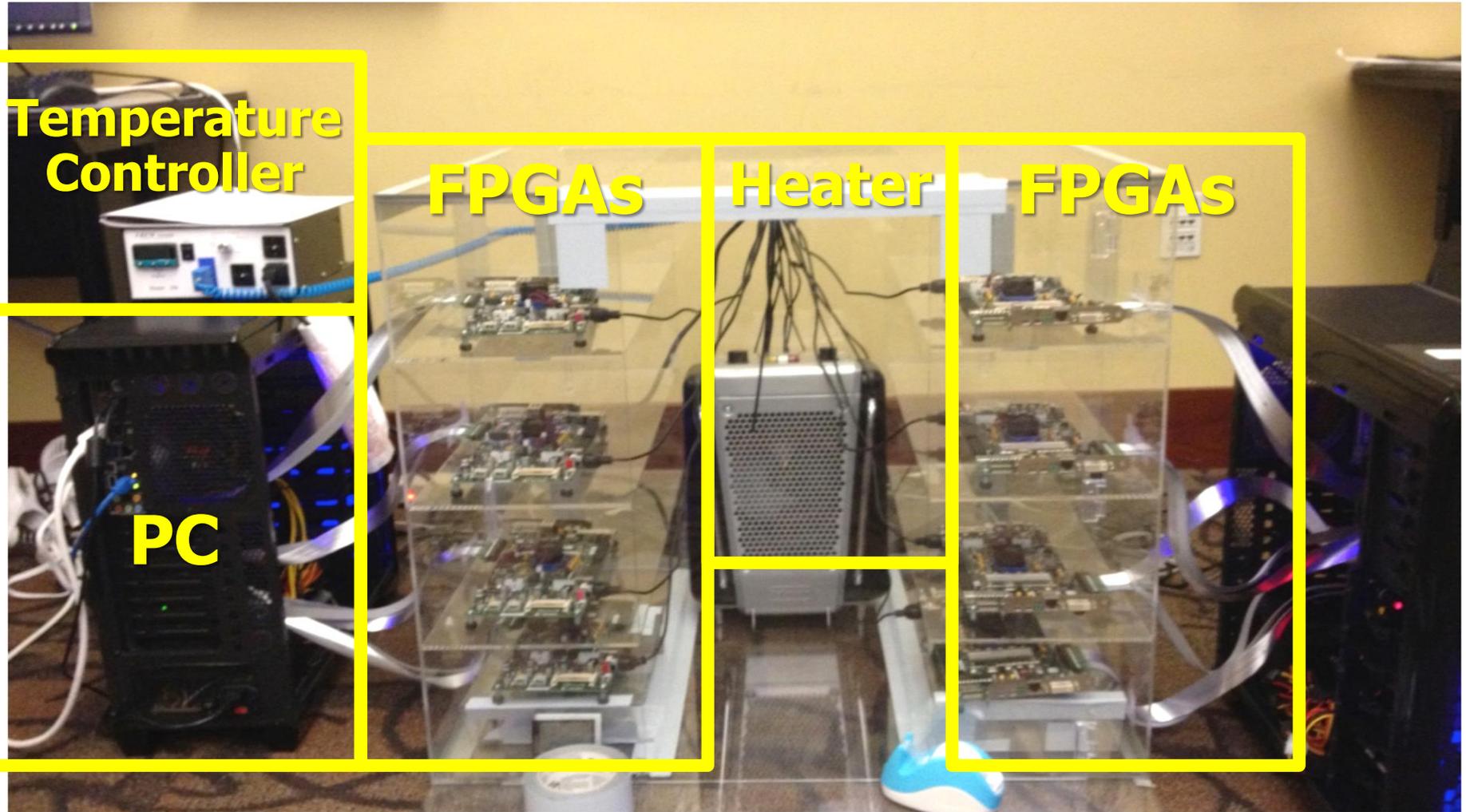
- Design of DRAM uArchitecture
  - Goal: Maximize capacity/area, not minimize latency
- “One size fits all” approach to latency specification
  - Same latency parameters for all temperatures
  - Same latency parameters for all DRAM chips (e.g., rows)
  - Same latency parameters for all parts of a DRAM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

# Latency Variation in Memory Chips

Heterogeneous manufacturing & operating conditions →  
latency variation in timing parameters

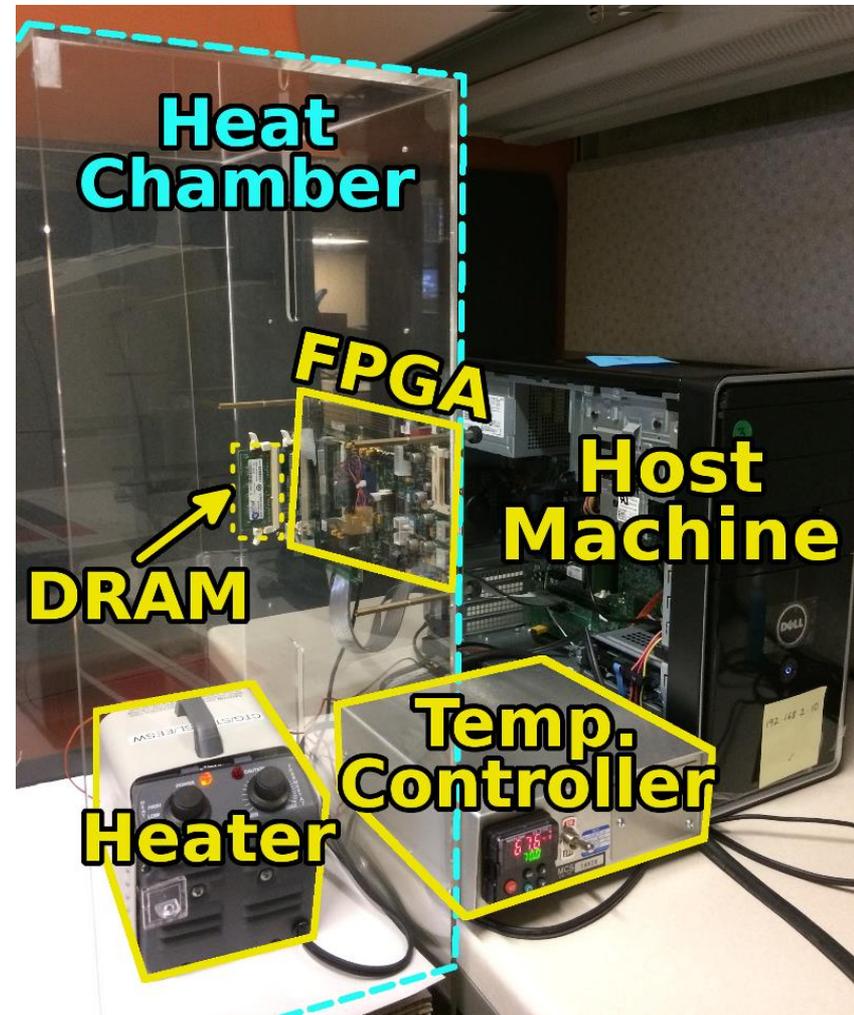


# DRAM Characterization Infrastructure



# DRAM Characterization Infrastructure

- Hasan Hassan et al., [SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies](#), HPCA 2017.
- Flexible
- Easy to Use (C++ API)
- Open-source  
[github.com/CMU-SAFARI/SoftMC](https://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<sup>1,2,3</sup> Nandita Vijaykumar<sup>3</sup> Samira Khan<sup>4,3</sup> Saugata Ghose<sup>3</sup> Kevin Chang<sup>3</sup>  
Gennady Pekhimenko<sup>5,3</sup> Donghyuk Lee<sup>6,3</sup> Oguz Ergin<sup>2</sup> Onur Mutlu<sup>1,3</sup>

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

# Tackling the Fixed Latency Mindset

---

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

# Adaptive-Latency DRAM

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

# Latency Reduction Summary of 115 DIMMs

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

# AL-DRAM: Real System Evaluation

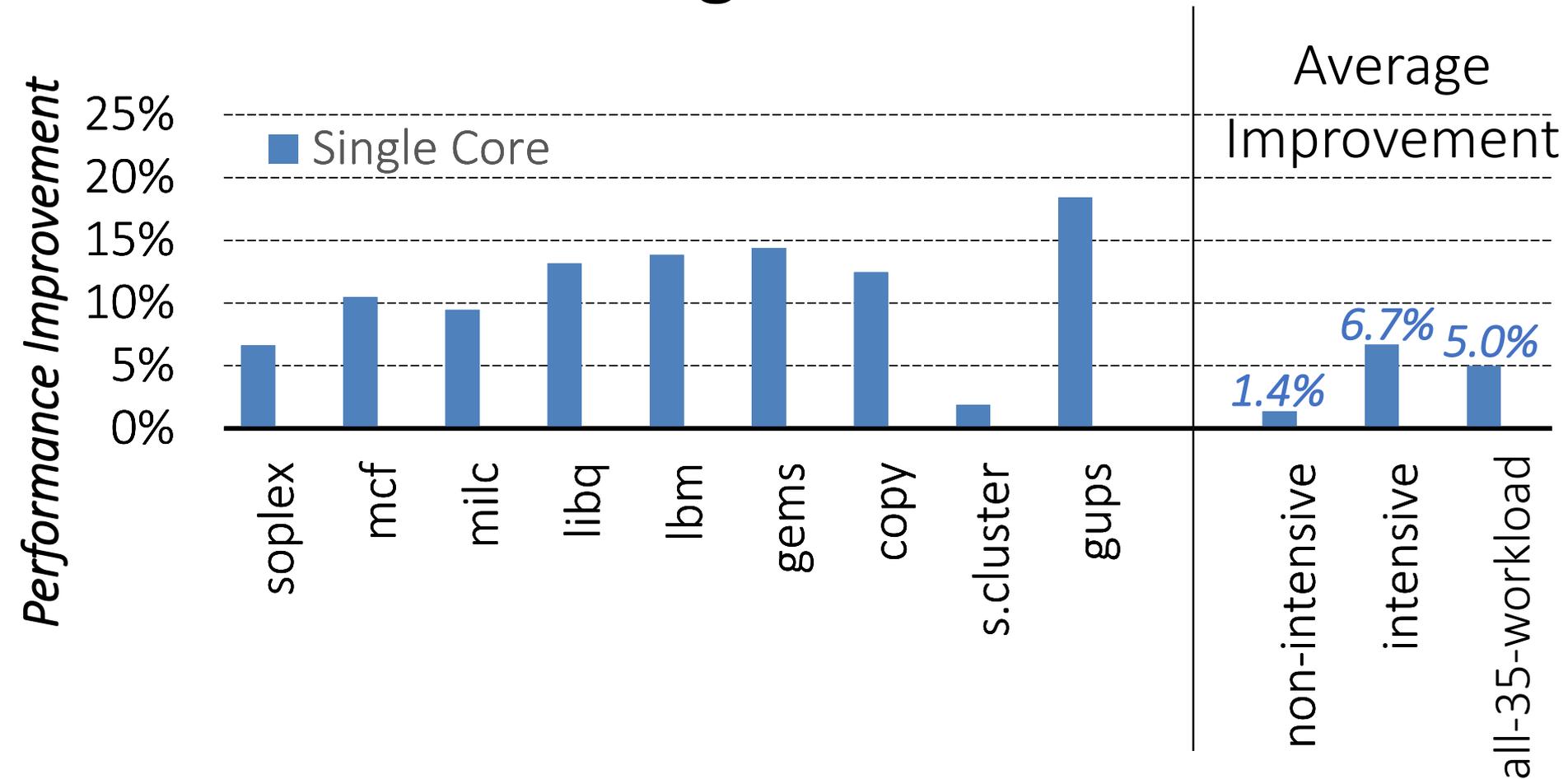
- *System*
  - CPU: AMD 4386 ( 8 Cores, 3.1GHz, 8MB LLC)

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

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

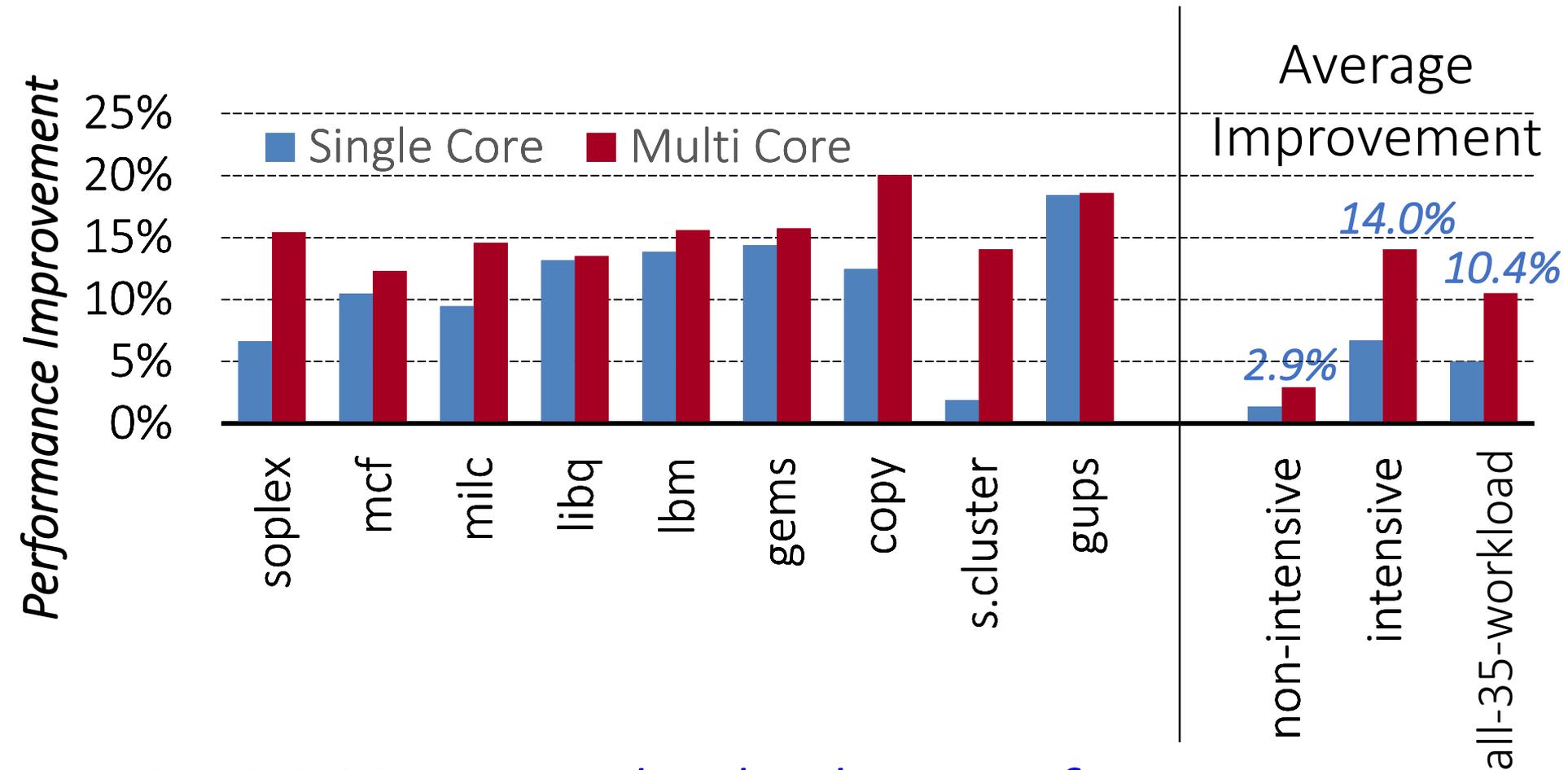
Bits	Description								
31:30	Reserved.								
29:24	<b>Tras: row active strobe.</b> Read-write. BIOS: See 2.9.7.5 [SPD ROM-Based Configuration]. Specifies the minimum time in memory clock cycles from an activate command to a precharge command, both to the same chip select bank. <table border="1">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

# AL-DRAM: Single-Core Evaluation



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

# AL-DRAM: Multi-Core Evaluation



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

# Reducing Latency Also Reduces Energy

---

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

# More on Adaptive-Latency DRAM

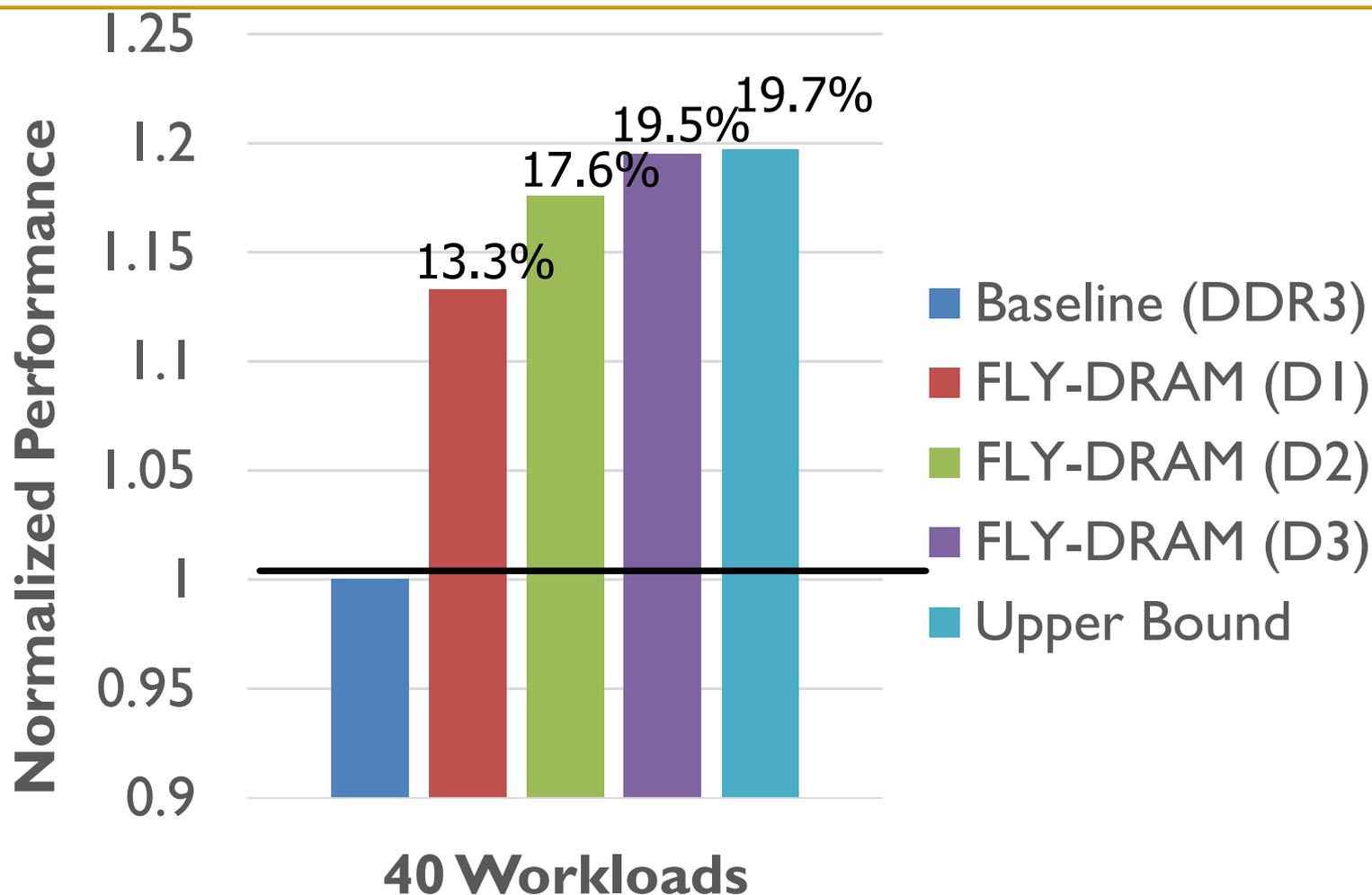
---

- Donghyuk Lee, Yoongu Kim, Gennady Pekhimenko, Samira Khan, Vivek Seshadri, Kevin Chang, and Onur Mutlu,  
**"Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case"**  
*Proceedings of the 21st International Symposium on High-Performance Computer Architecture (HPCA)*, Bay Area, CA, February 2015.  
[\[Slides \(pptx\) \(pdf\)\]](#) [\[Full data sets\]](#)

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

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

# Heterogeneous Latency within A Chip



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

# Analysis of Latency Variation in DRAM Chips

---

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

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

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

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

[[Source Code](#)]

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

Kevin K. Chang<sup>1</sup>

Abhijith Kashyap<sup>1</sup>

Hasan Hassan<sup>1,2</sup>

Saugata Ghose<sup>1</sup>

Kevin Hsieh<sup>1</sup>

Donghyuk Lee<sup>1</sup>

Tianshi Li<sup>1,3</sup>

Gennady Pekhimenko<sup>1</sup>

Samira Khan<sup>4</sup>

Onur Mutlu<sup>5,1</sup>

<sup>1</sup>Carnegie Mellon University

<sup>2</sup>TOBB ETÜ

<sup>3</sup>Peking University

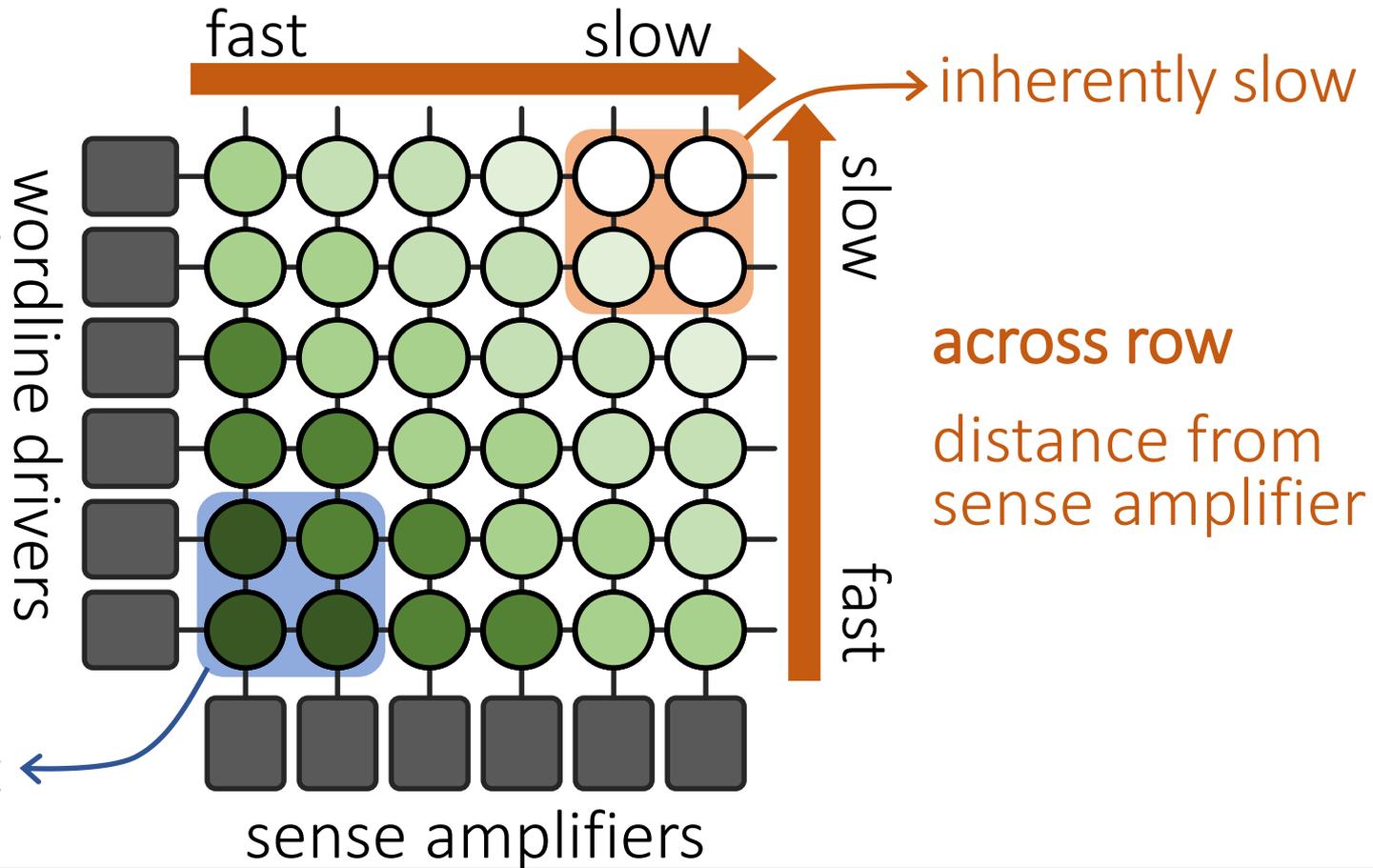
<sup>4</sup>University of Virginia

<sup>5</sup>ETH Zürich

# What Is Design-Induced Variation?

across column

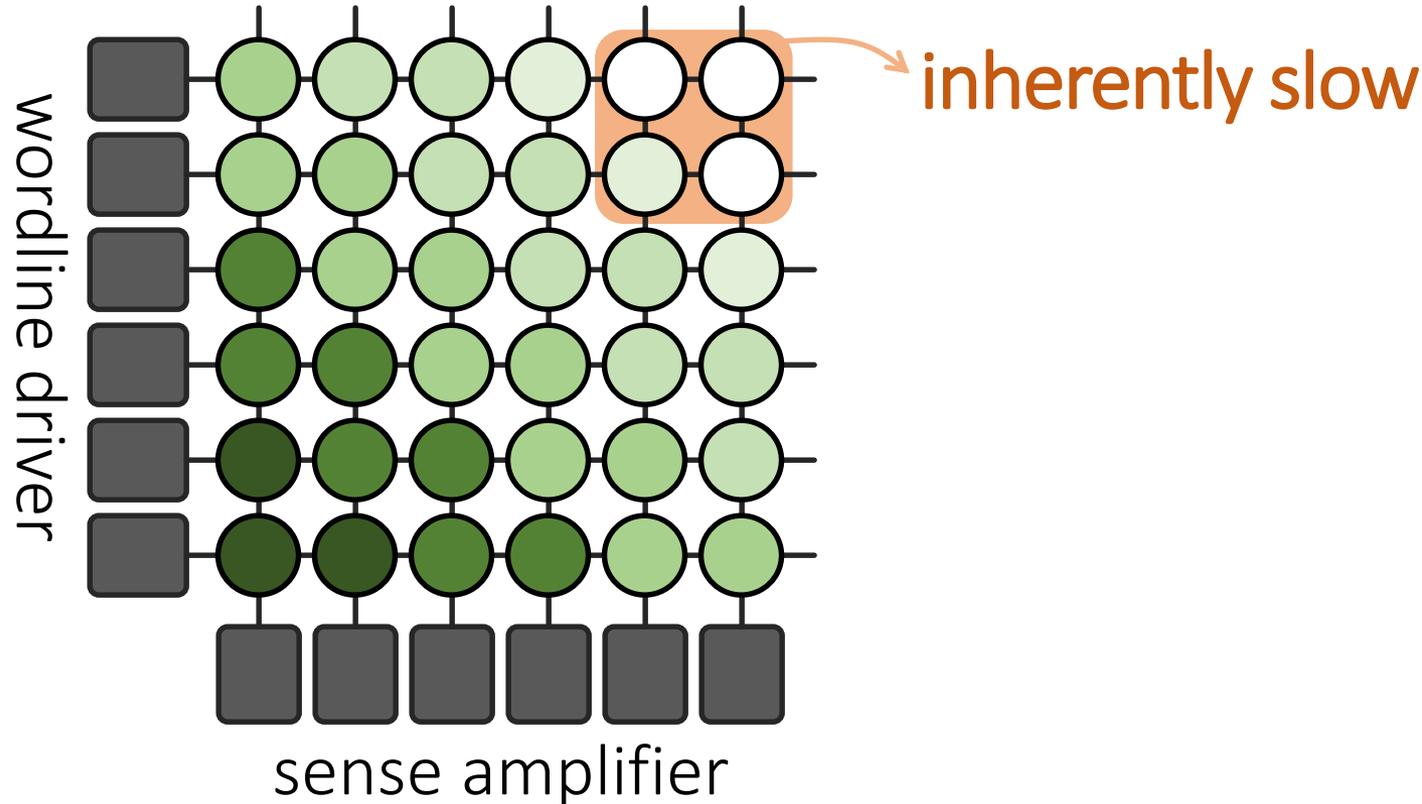
distance from  
wordline driver



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

# DIVA Online Profiling

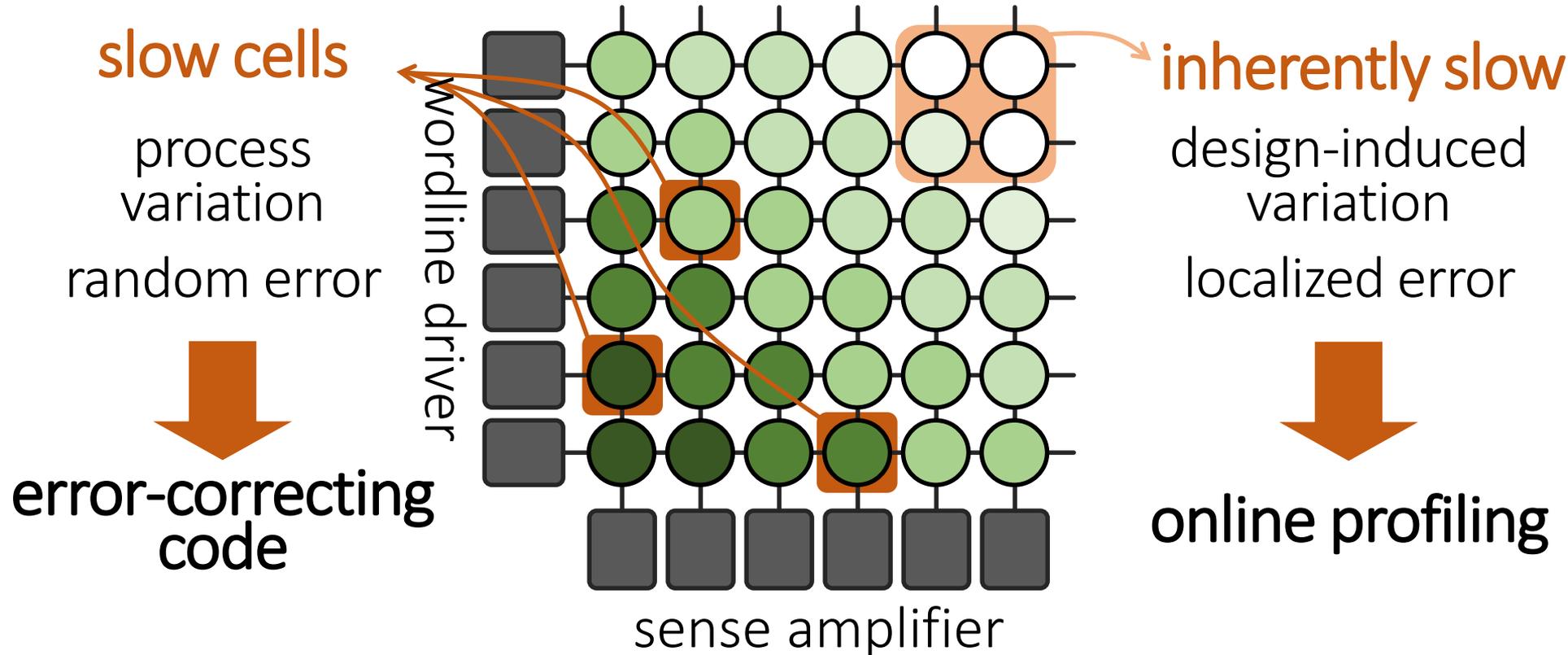
Design-Induced-Variation-Aware



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

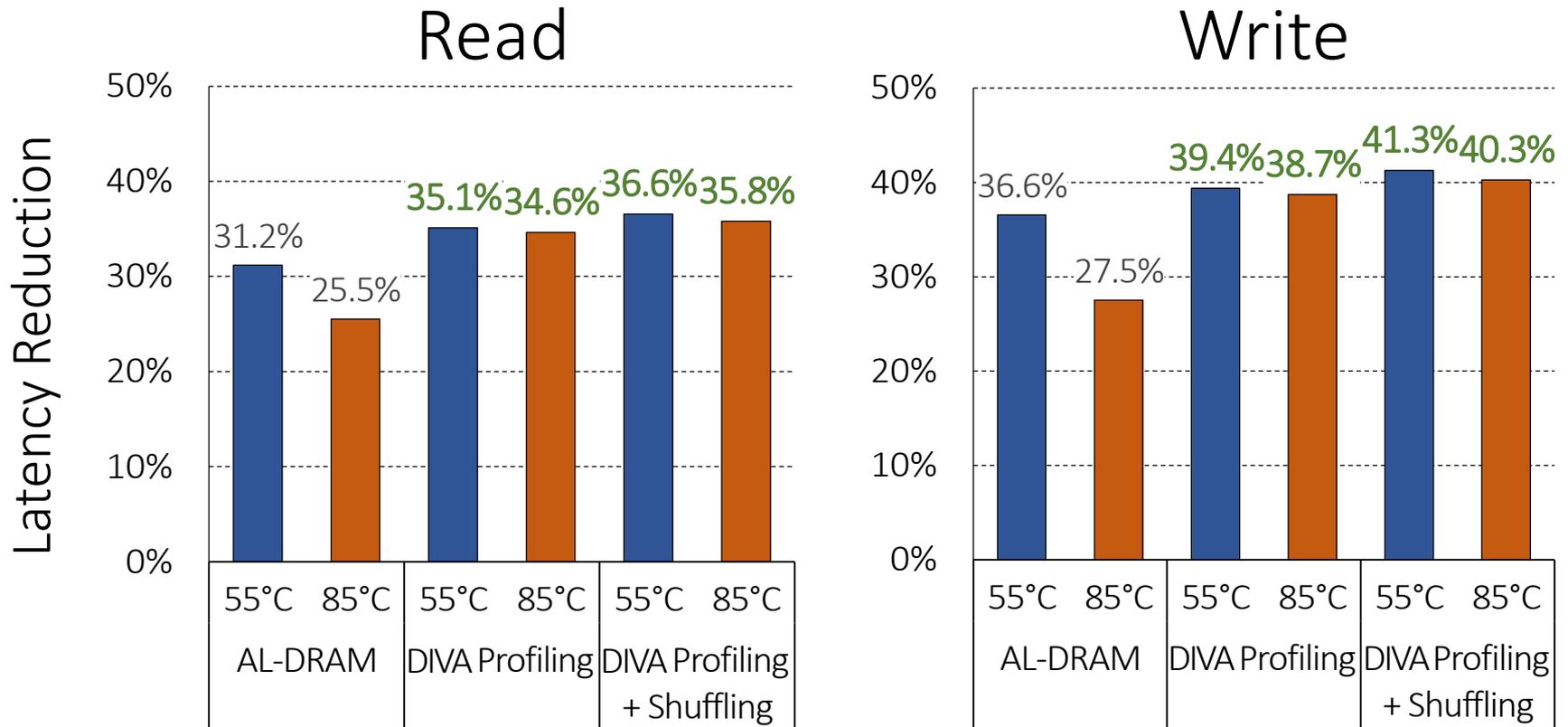
# DIVA Online Profiling

Design-Induced-Variation-Aware



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

# DIVA-DRAM Reduces Latency



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

# Design-Induced Latency Variation in DRAM

---

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

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

Donghyuk Lee, NVIDIA and Carnegie Mellon University

Samira Khan, University of Virginia

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

Gennady Pekhimenko, Vivek Seshadri, Microsoft Research

Onur Mutlu, ETH Zürich and Carnegie Mellon University

# Voltron: Exploiting the Voltage-Latency-Reliability Relationship

# Executive Summary

---

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

# Analysis of Latency-Voltage in DRAM Chips

---

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

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

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

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

Kevin K. Chang<sup>†</sup>    Abdullah Giray Yağlıkçı<sup>†</sup>    Saugata Ghose<sup>†</sup>    Aditya Agrawal<sup>¶</sup>    Niladrish Chatterjee<sup>¶</sup>  
Abhijith Kashyap<sup>†</sup>    Donghyuk Lee<sup>¶</sup>    Mike O'Connor<sup>¶,‡</sup>    Hasan Hassan<sup>§</sup>    Onur Mutlu<sup>§,†</sup>

<sup>†</sup>Carnegie Mellon University

<sup>¶</sup>NVIDIA

<sup>‡</sup>The University of Texas at Austin

<sup>§</sup>ETH Zürich

# And, What If ...

---

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

# *The DRAM Latency PUF:*

Quickly Evaluating Physical Unclonable Functions  
by Exploiting the Latency-Reliability Tradeoff  
in Modern Commodity DRAM Devices

Jeremie S. Kim    Minesh Patel

Hasan Hassan    Onur Mutlu



**SAFARI**

**ETH** zürich

**Carnegie Mellon**

# Motivation

- A **PUF** is function that generates a signature **unique** to a given device
- Used in a **Challenge-Response Protocol**
  - Each device generates a unique **PUF response** depending the inputs
  - A trusted server **authenticates** a device if it generates the expected PUF response

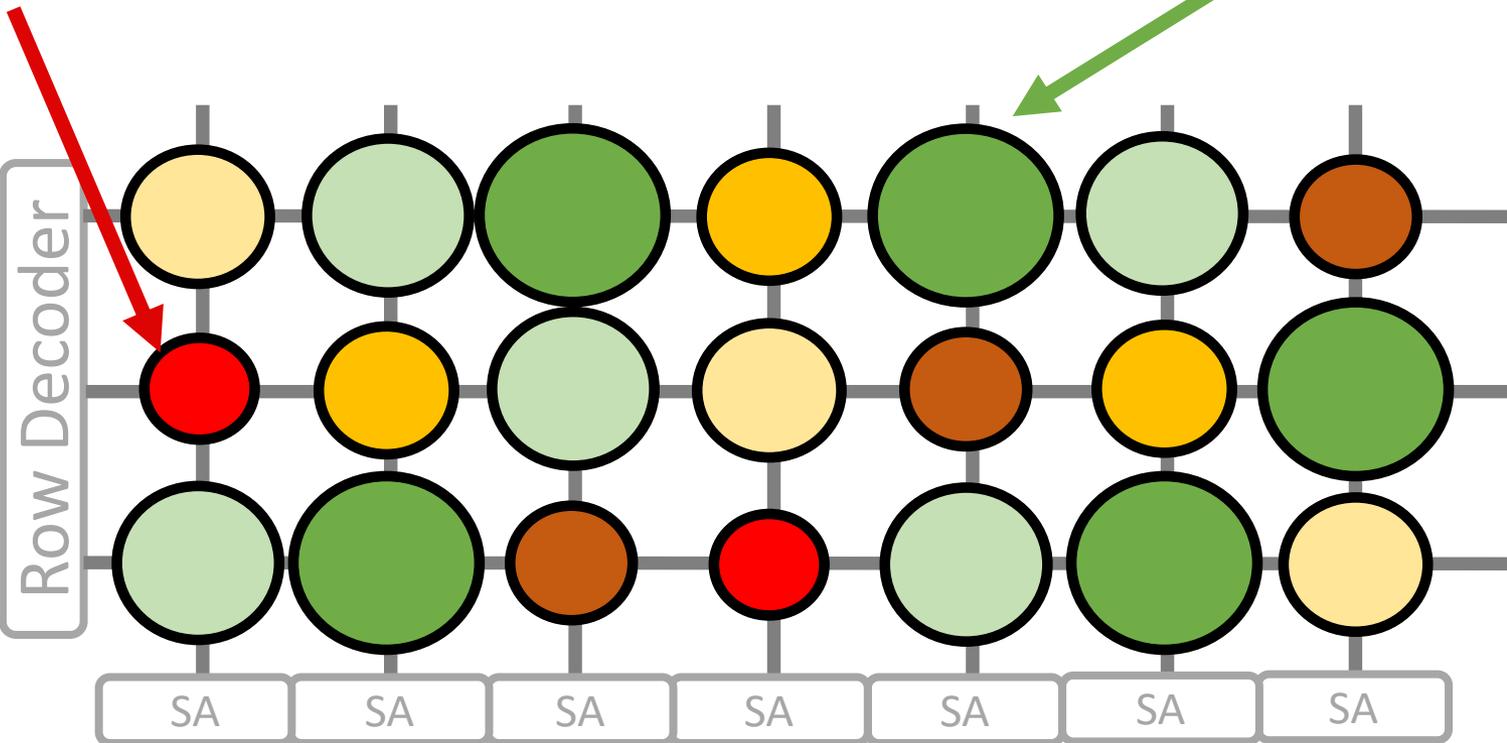
# DRAM Latency Characterization of 223 LPDDR4 DRAM Devices

- Latency failures come from accessing DRAM with **reduced** timing parameters.
- **Key Observations:**
  1. A cell's **latency failure** probability is determined by **random process variation**
  2. Latency failure patterns are **repeatable and unique to a device**

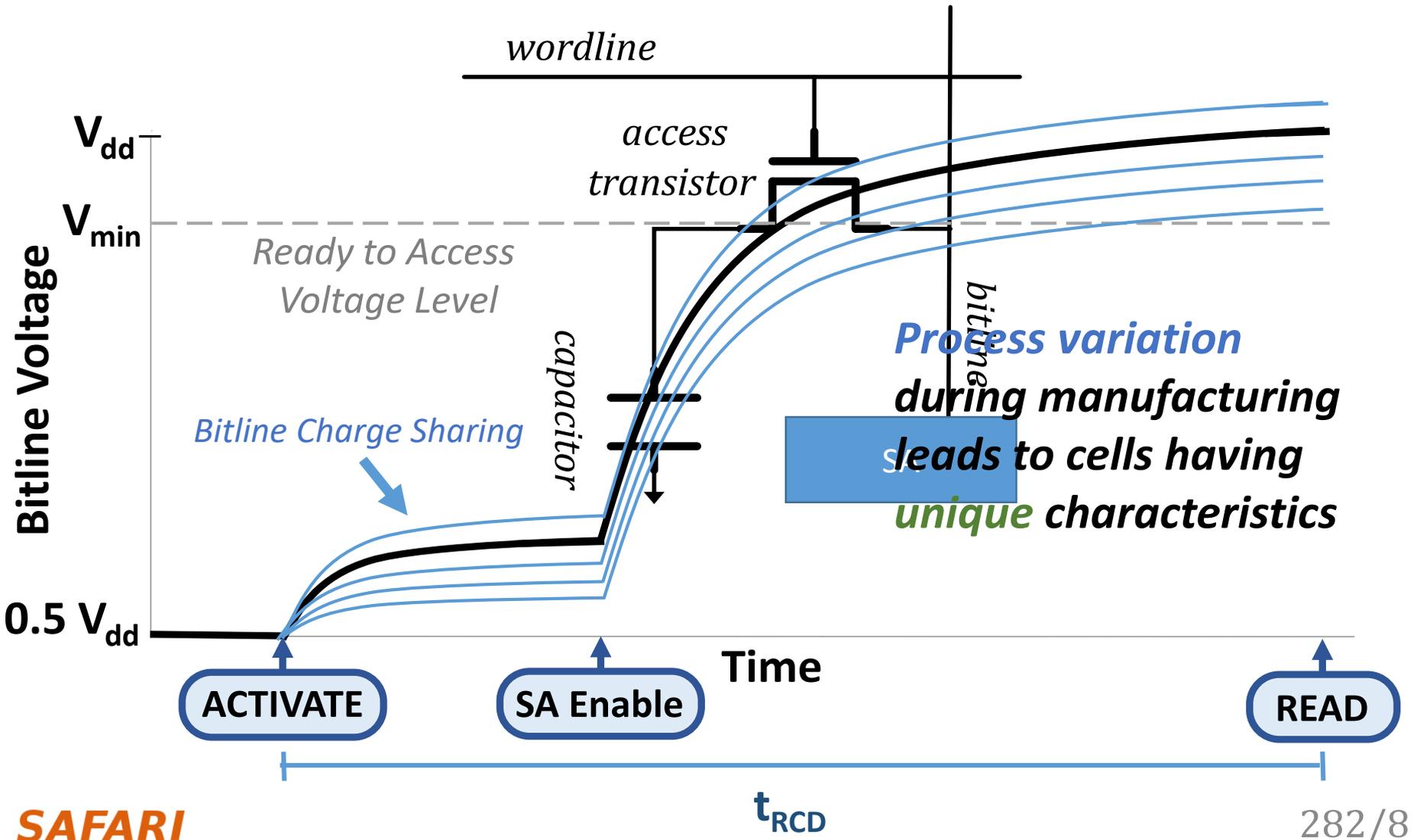
# DRAM Latency PUF Key Idea

High % chance to fail with reduced  $t_{RCD}$

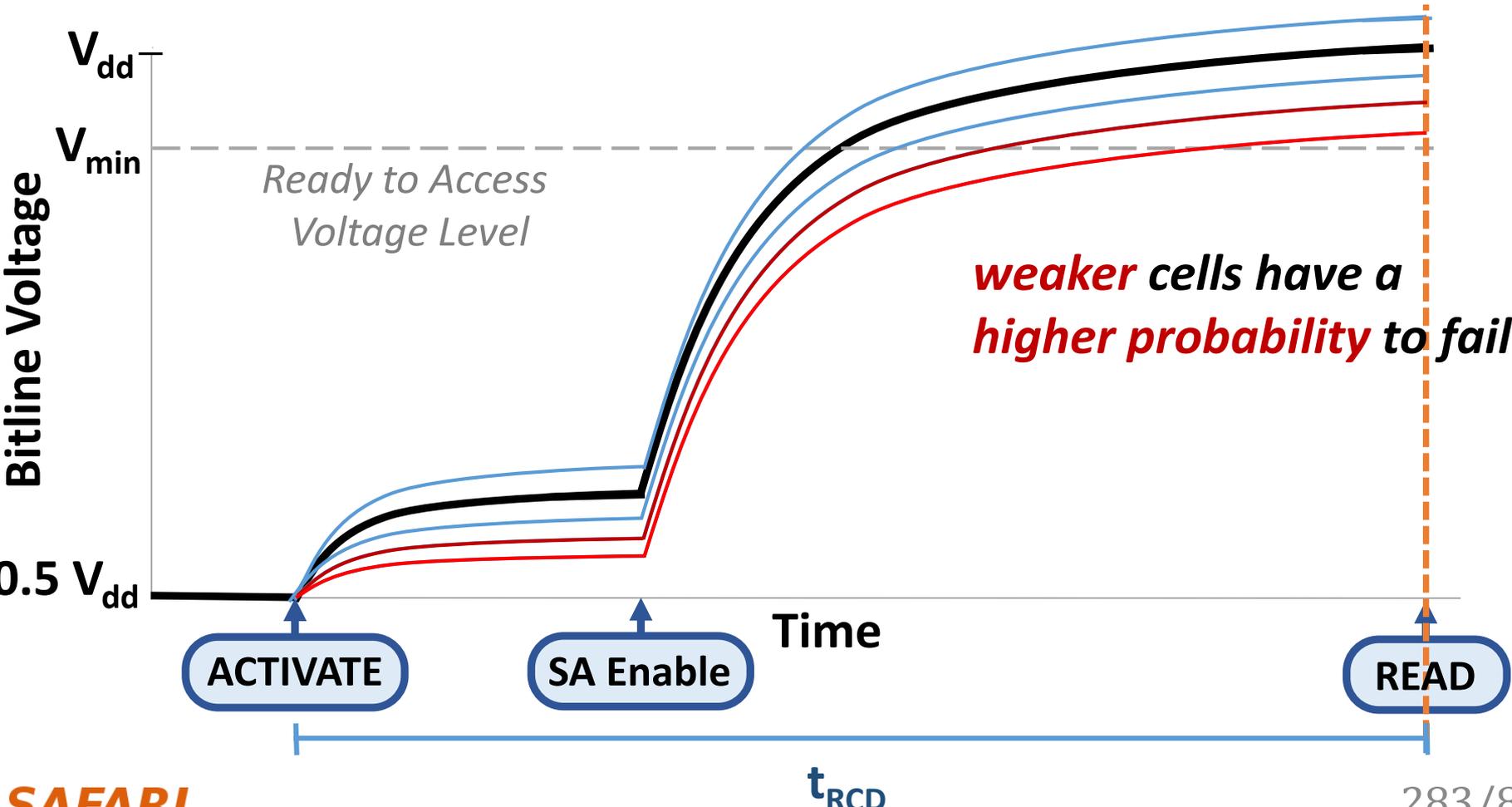
Low % chance to fail with reduced  $t_{RCD}$



# DRAM Accesses and Failures



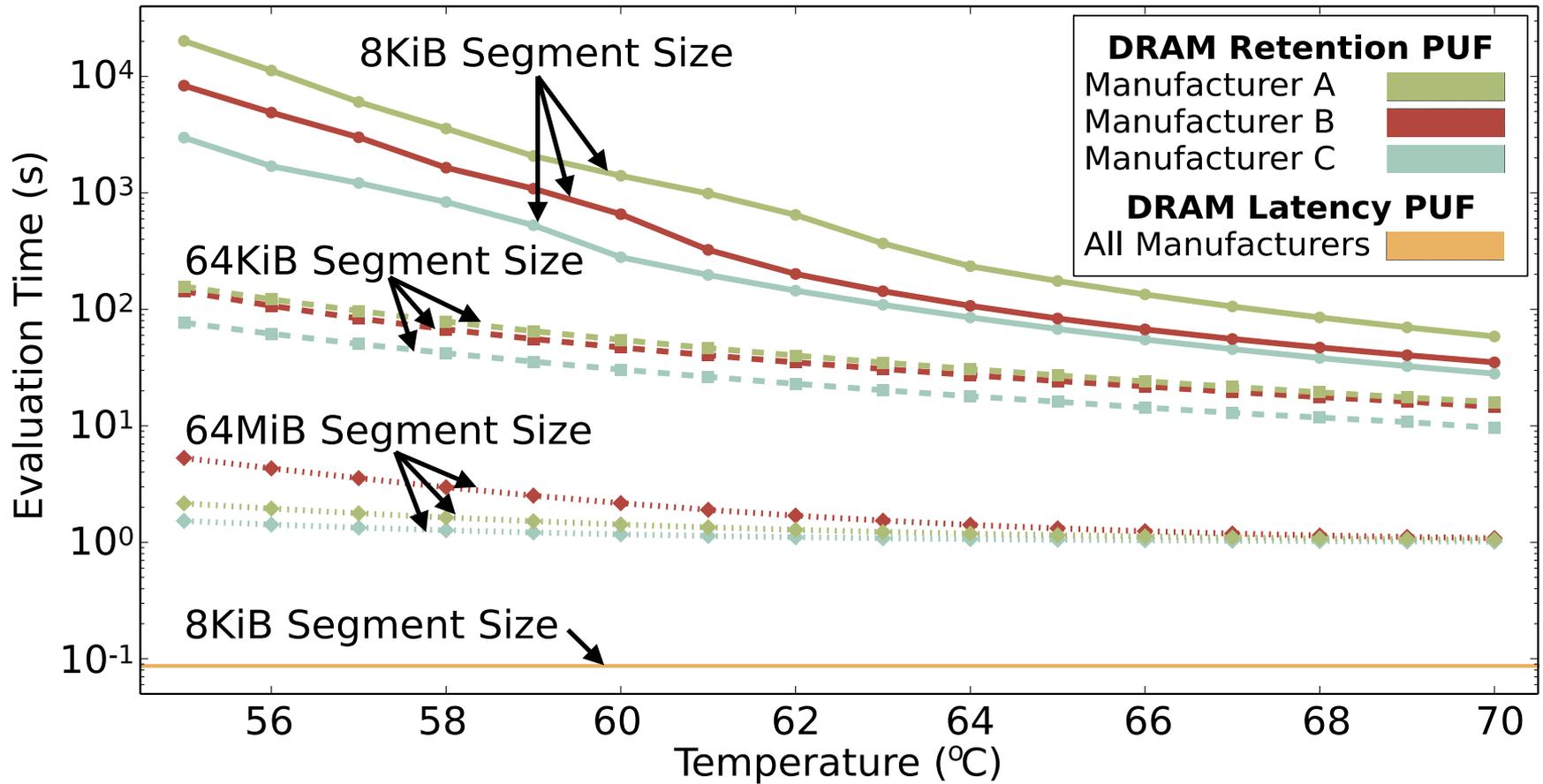
# DRAM Accesses and Failures



# The DRAM Latency PUF Evaluation

- We generate PUF responses using **latency errors** in a region of DRAM
- The latency error patterns **satisfy PUF requirements**
- The DRAM Latency PUF **generates PUF responses in 88.2ms**

# Results



- We are **orders of magnitude faster** than prior DRAM PUFs!

# *The DRAM Latency PUF:*

Quickly Evaluating Physical Unclonable Functions  
by Exploiting the Latency-Reliability Tradeoff  
in Modern Commodity DRAM Devices

Jeremie S. Kim Minesh Patel

Hasan Hassan Onur Mutlu



QR Code for the paper

[https://people.inf.ethz.ch/omutlu/pub/dram-latency-puf\\_hpca18.pdf](https://people.inf.ethz.ch/omutlu/pub/dram-latency-puf_hpca18.pdf)

**HPCA 2018**

**SAFARI**



**ETH** zürich

**Carnegie Mellon**

# DRAM Latency PUFs

---

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu,  
**"The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern DRAM Devices"**  
*Proceedings of the 24th International Symposium on High-Performance Computer Architecture (HPCA)*, Vienna, Austria, February 2018.  
[[Lightning Talk Video](#)]  
[[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Session Slides \(pptx\)](#)] [[pdf](#)]

## The DRAM Latency PUF:

Quickly Evaluating Physical Unclonable Functions

by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices

Jeremie S. Kim<sup>†§</sup>

Minesh Patel<sup>§</sup>

Hasan Hassan<sup>§</sup>

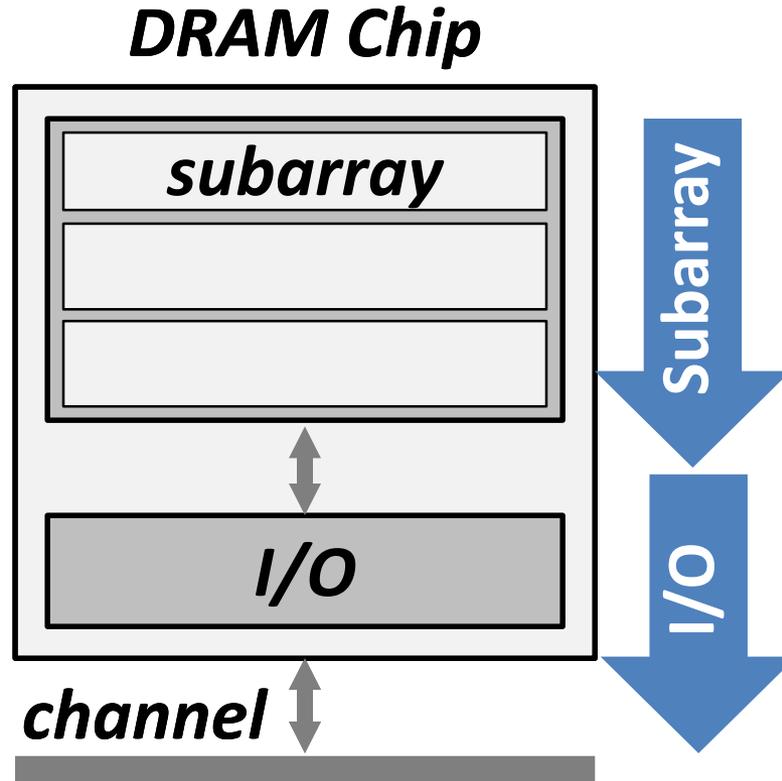
Onur Mutlu<sup>§†</sup>

<sup>†</sup>Carnegie Mellon University

<sup>§</sup>ETH Zürich

# Tiered Latency DRAM

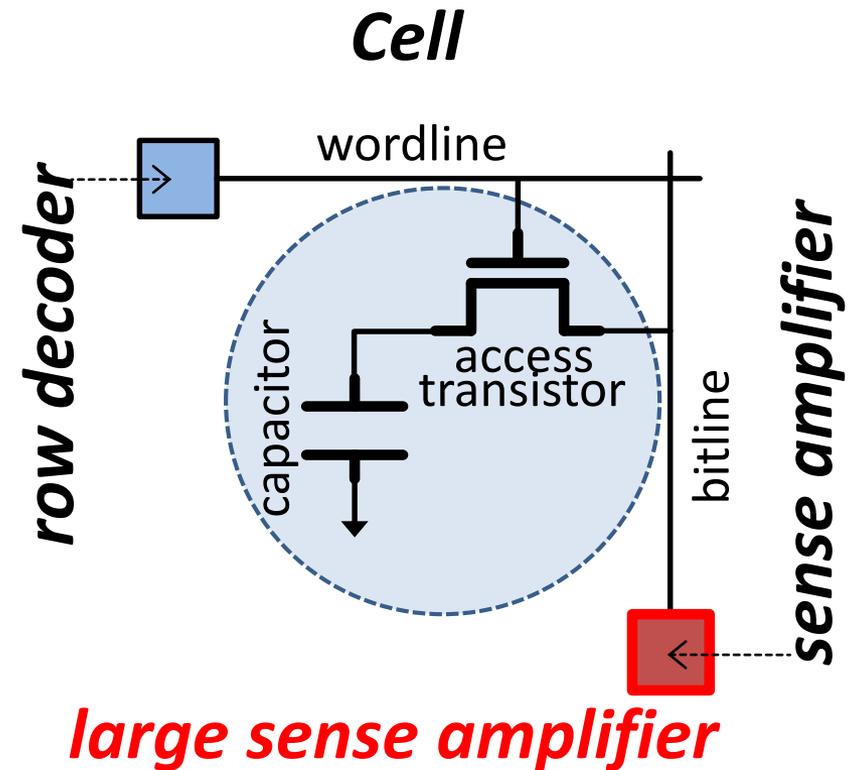
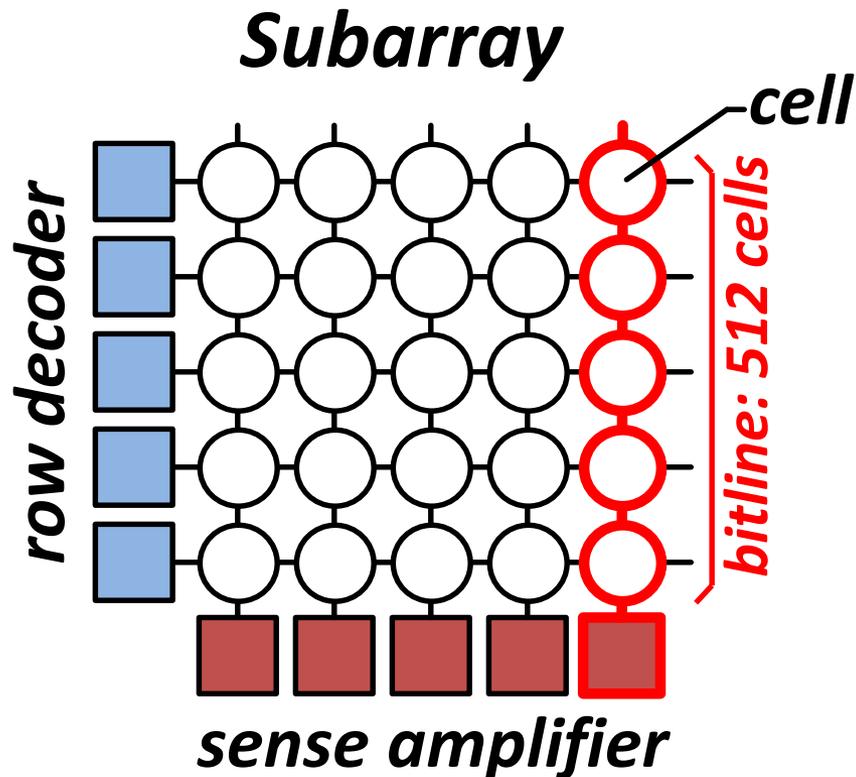
# What Causes the Long Latency?



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

**Dominant**

# Why is the Subarray So Slow?

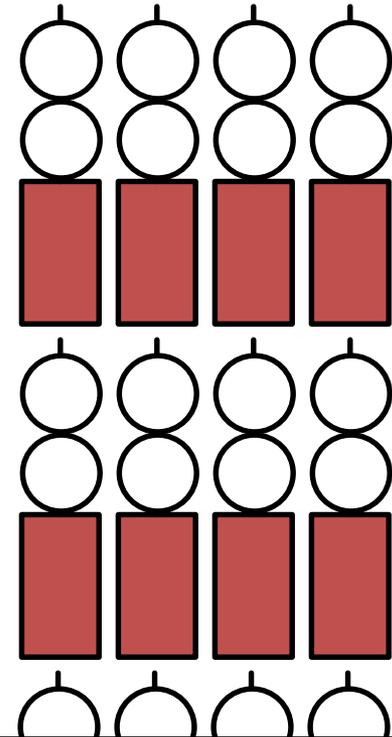
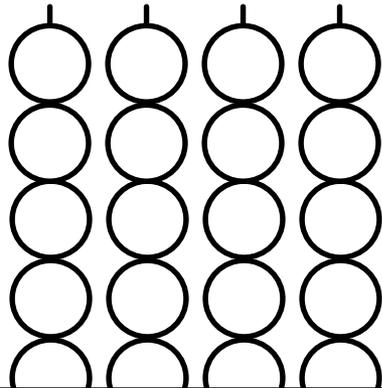


- Long bitline
  - Amortizes sense amplifier cost → Small area
  - Large bitline capacitance → High latency & power

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

Long Bitline

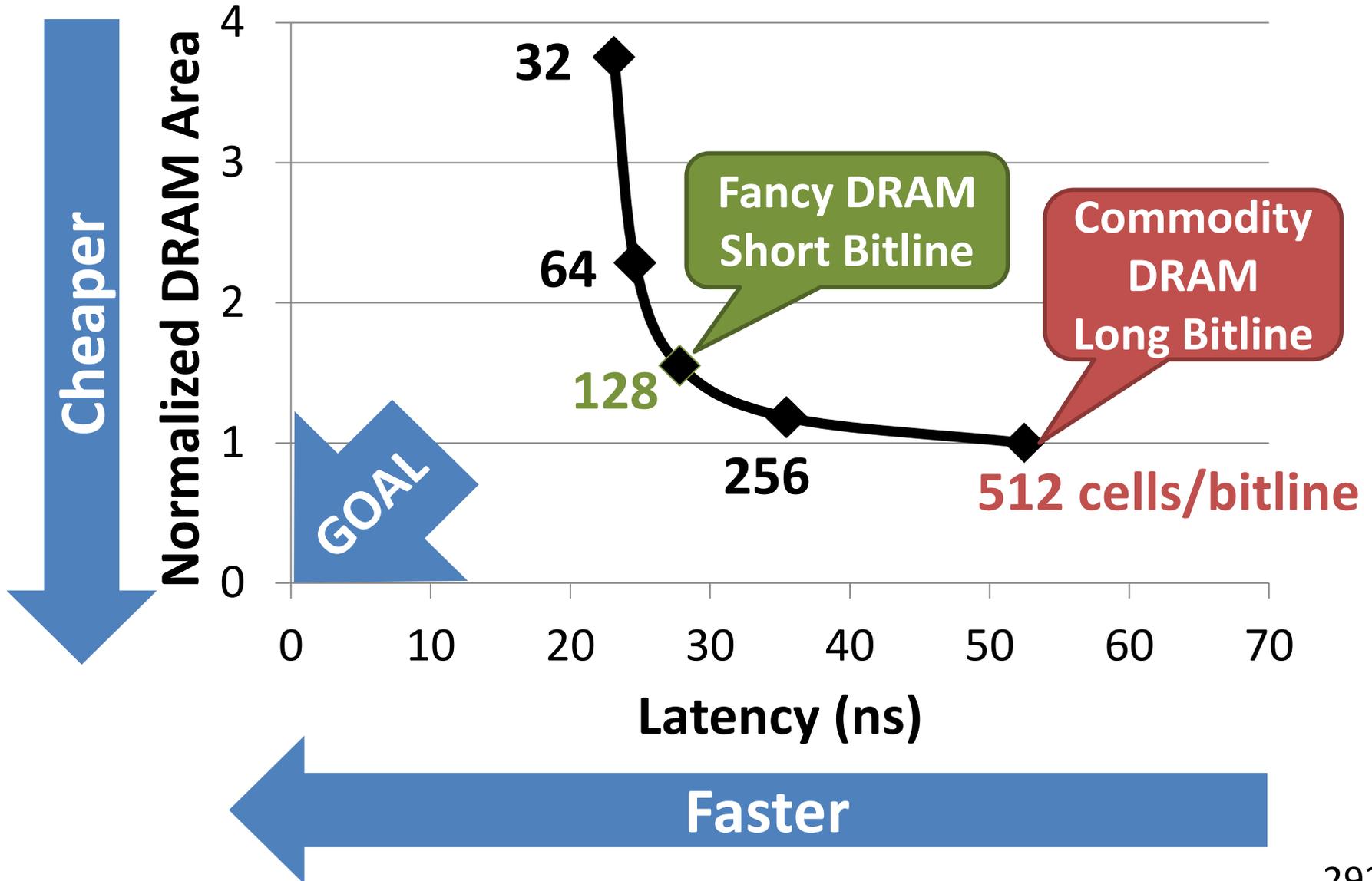
Short Bitline



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



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

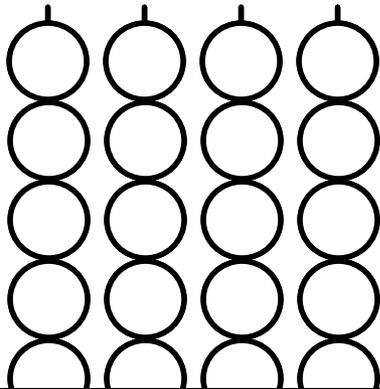


# Approximating the Best of Both Worlds

**Long Bitline**

*Small Area*

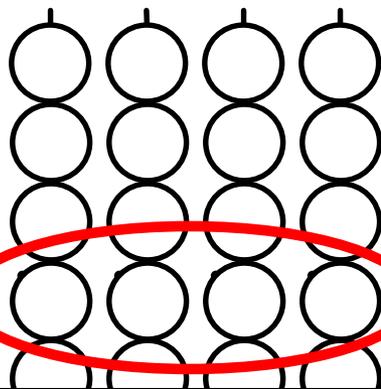
~~High Latency~~



**Need Isolation**

**Our Proposal**

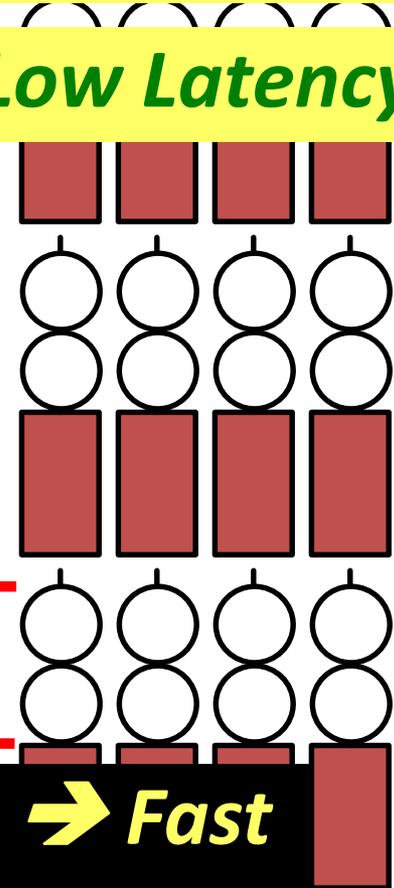
**Add Isolation Transistors**



**Short Bitline**

~~Large Area~~

*Low Latency*



**tline → Fast**

# Approximating the Best of Both Worlds

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

*Small Area*

*Small Area*

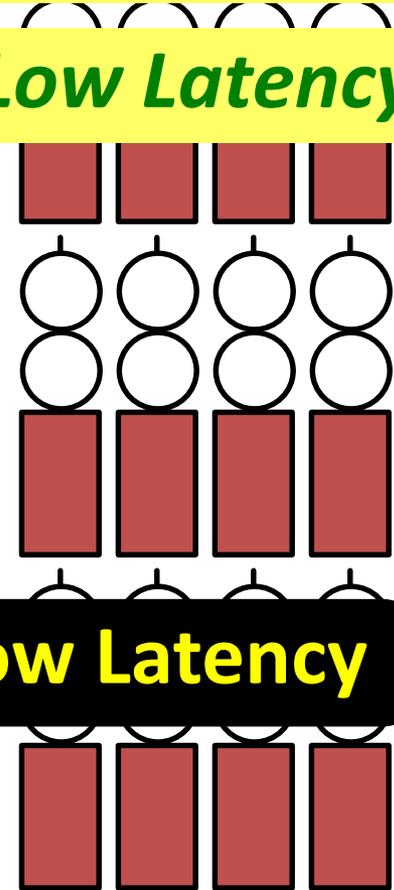
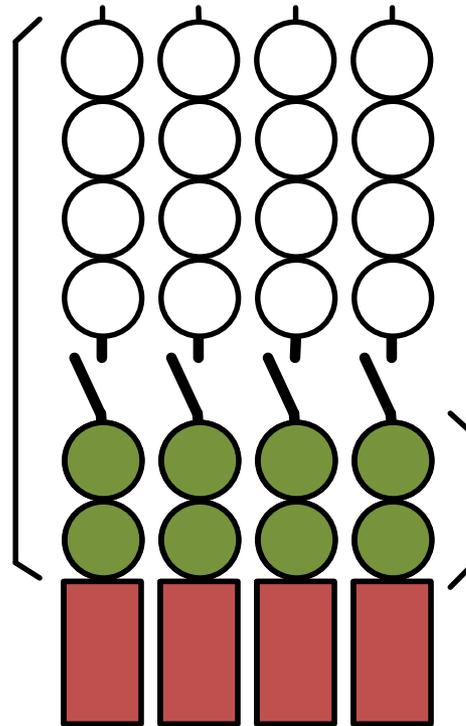
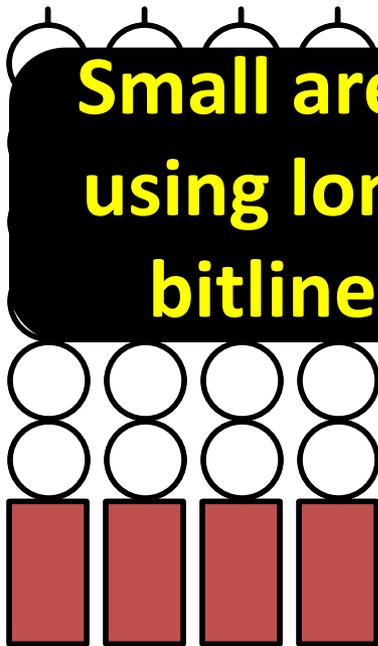
~~*Large Area*~~

~~*High Latency*~~

*Low Latency*

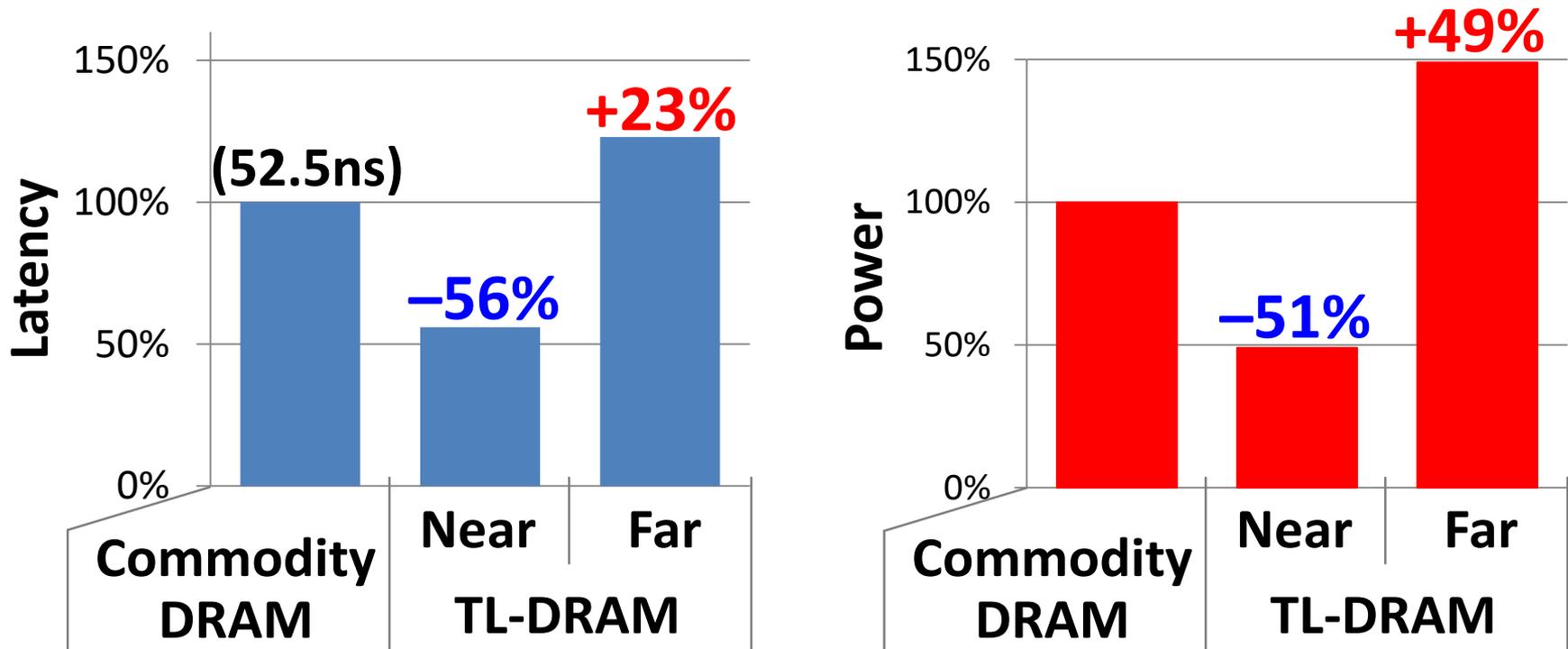
*Low Latency*

**Small area  
using long  
bitline**



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

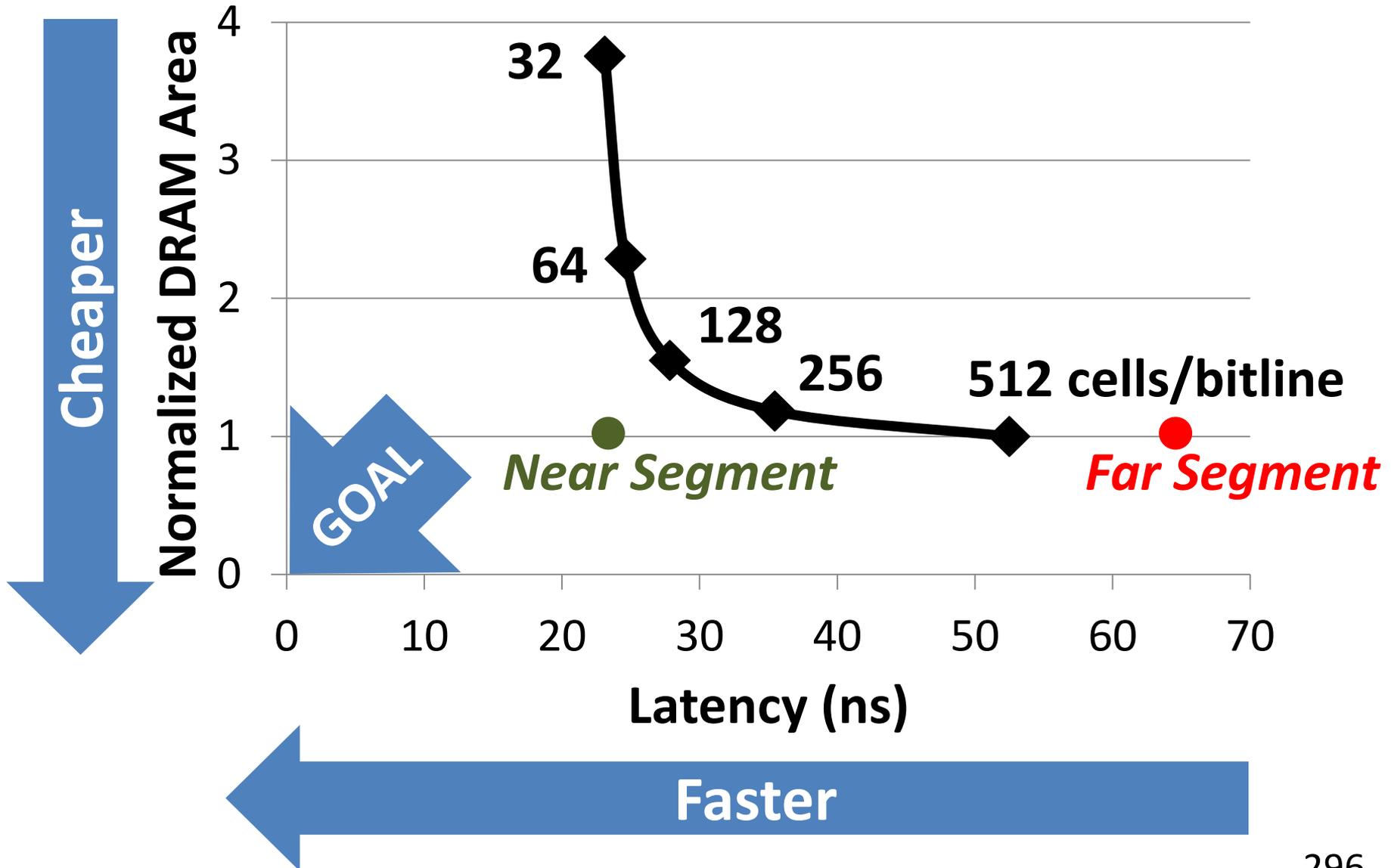
- DRAM Latency (tRC) • DRAM Power



- DRAM Area Overhead

~3%: mainly due to the isolation transistors

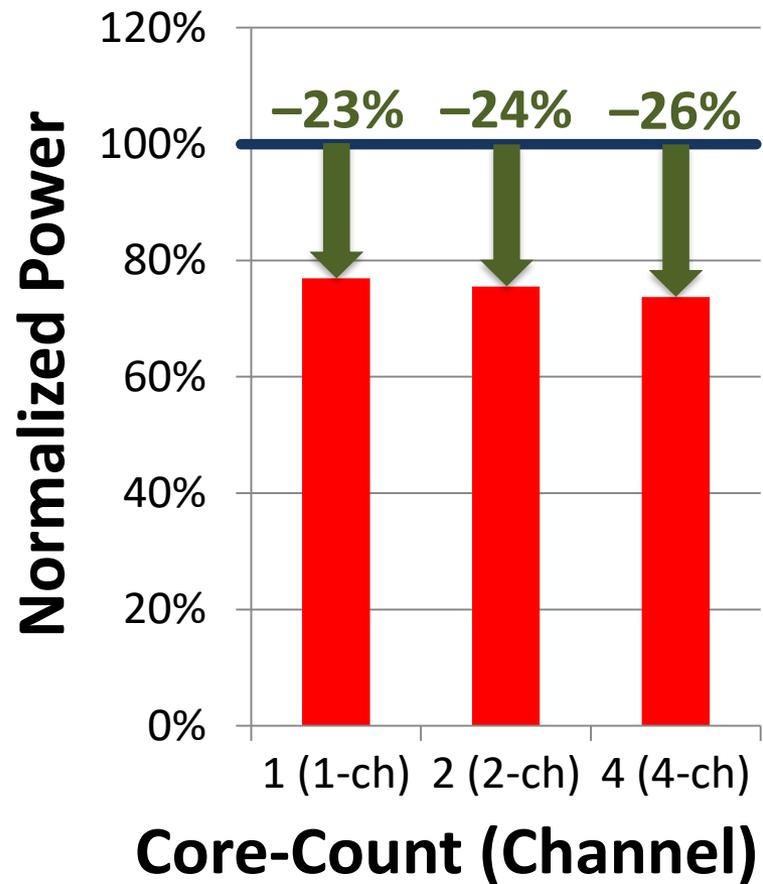
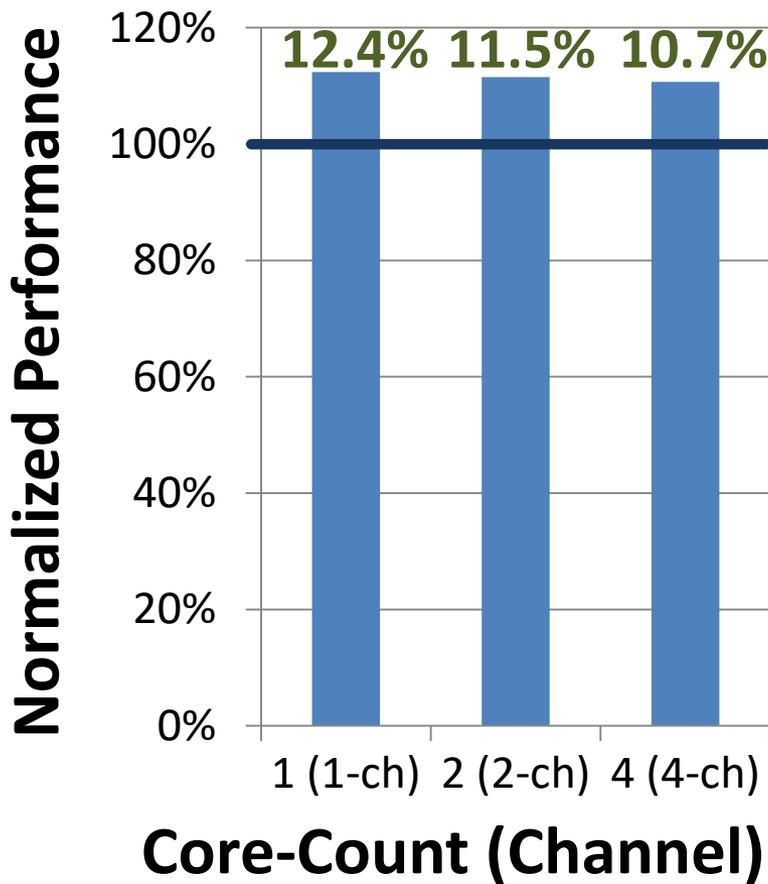
# Trade-Off: Area (Die-Area) vs. Latency



# Leveraging Tiered-Latency DRAM

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

# Performance & Power Consumption



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

# Fundamentally Low Latency Computing Architectures

# Ramulator: A Fast and Extensible DRAM Simulator

**[IEEE Comp Arch Letters'15]**

# Ramulator Motivation

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

<i>Segment</i>	<i>DRAM Standards &amp; Architectures</i>
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLD RAM3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

Table 1. Landscape of DRAM-based memory

# Ramulator

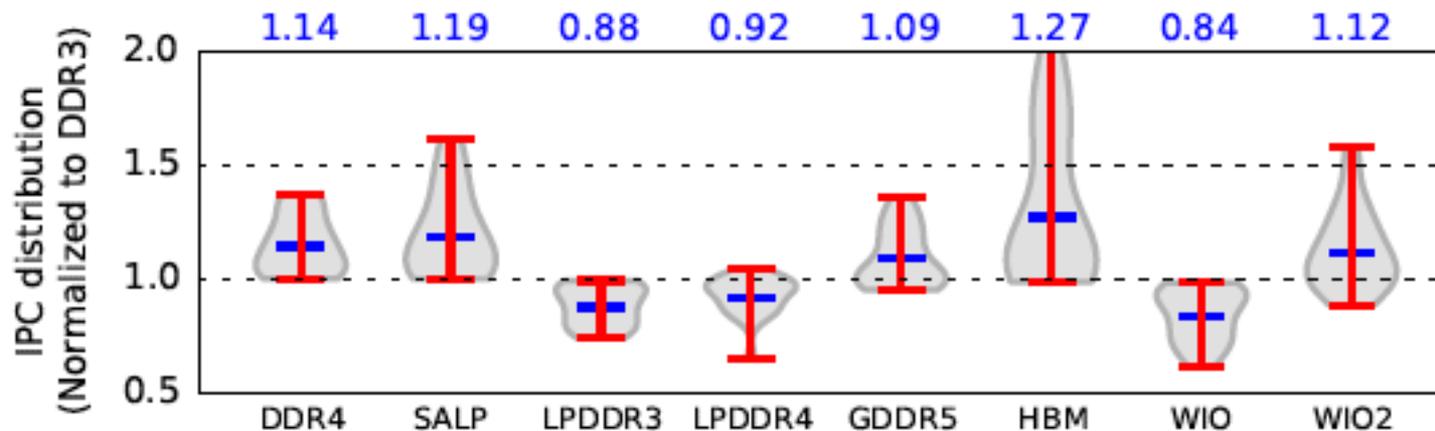
- Provides out-of-the box support for many DRAM standards:
  - DDR3/4, LPDDR3/4, GDDR5, WIO1/2, HBM, plus new proposals (SALP, AL-DRAM, TLDRAM, RowClone, and SARP)
- ~2.5X faster than fastest open-source simulator
- Modular and extensible to different standards

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

Table 3. Comparison of five simulators using two traces

# Case Study: Comparison of DRAM Standards

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



Across 22 workloads, simple CPU model

Figure 2. Performance comparison of DRAM standards

# Ramulator Paper and Source Code

---

- Yoongu Kim, Weikun Yang, and Onur Mutlu,  
**"Ramulator: A Fast and Extensible DRAM Simulator"**  
*IEEE Computer Architecture Letters (CAL)*, March 2015.  
[Source Code]
- Source code is released under the liberal MIT License
  - <https://github.com/CMU-SAFARI/ramulator>

# A Deeper Dive into DRAM Reliability Issues

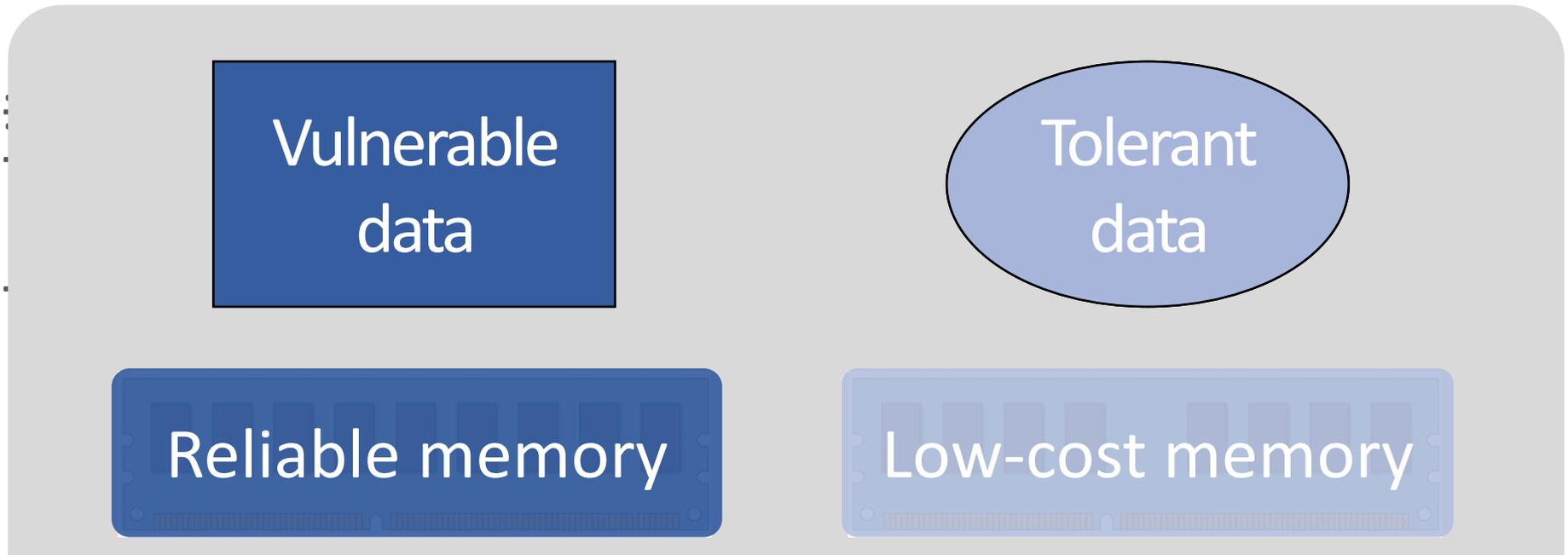
# There are Two Other Solution Directions

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



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

# Exploiting Memory Error Tolerance with Hybrid Memory Systems



On Microsoft's Web Search workload

Reduces server hardware **cost** by **4.7 %**

Achieves single server **availability** target of **99.90 %**

**Heterogeneous-Reliability Memory** [DSN 2014]

# More on Heterogeneous-Reliability Memory

---

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

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

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

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

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

# Root Causes of Disturbance Errors

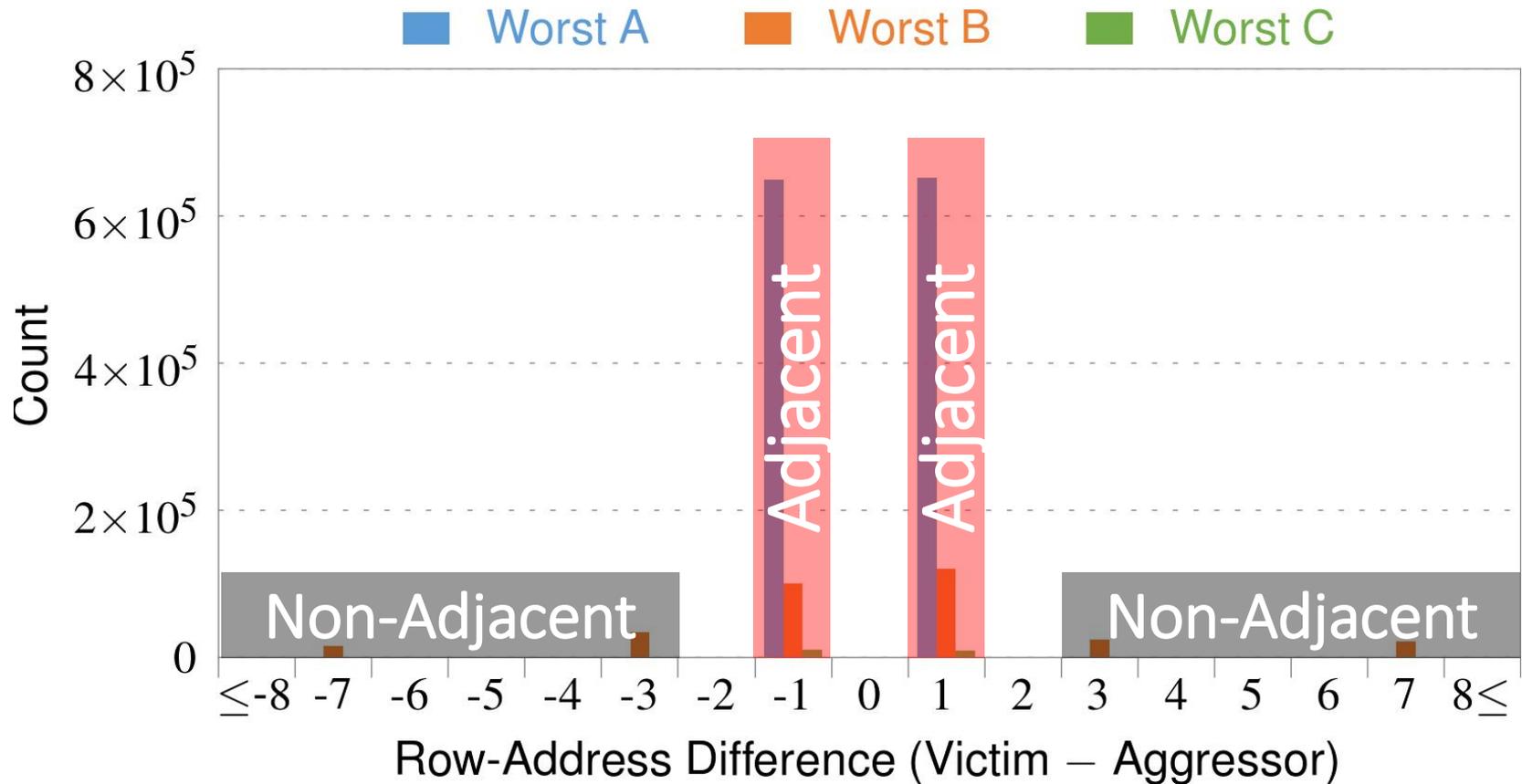
- *Cause 1: Electromagnetic coupling*
  - Toggling the wordline voltage briefly increases the voltage of adjacent wordlines
  - Slightly opens adjacent rows → Charge leakage
- *Cause 2: Conductive bridges*
- *Cause 3: Hot-carrier injection*

*Confirmed by at least one manufacturer*

# RowHammer Characterization Results

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

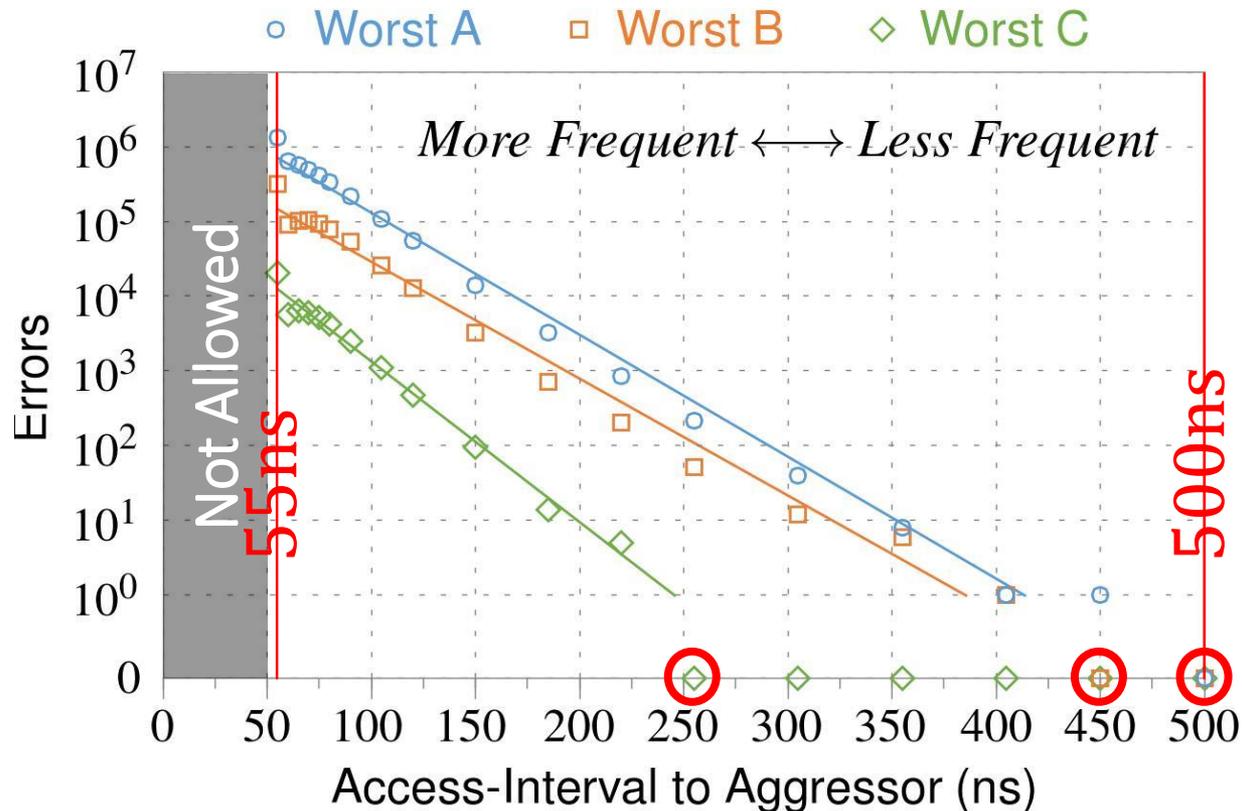
# 4. Adjacency: Aggressor & Victim



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

*Most aggressors & victims are adjacent*

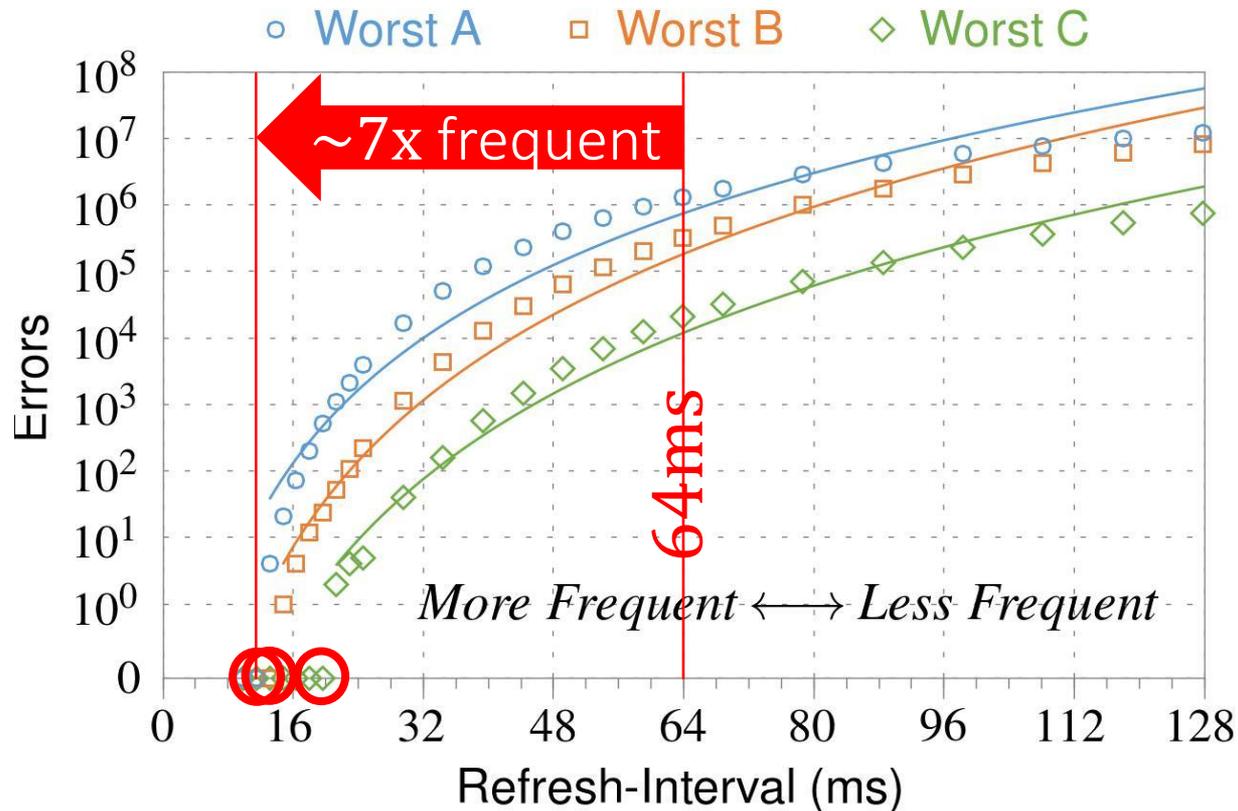
# 1 Access Interval (Aggressor)



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

*Less frequent accesses → Fewer errors*

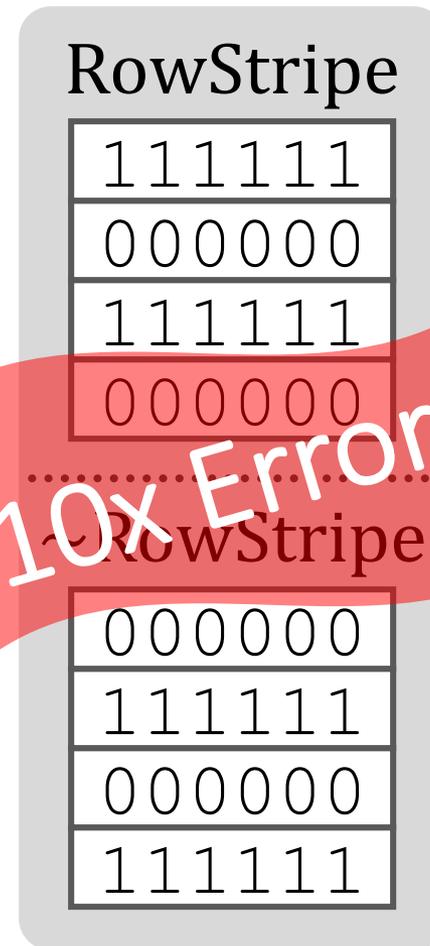
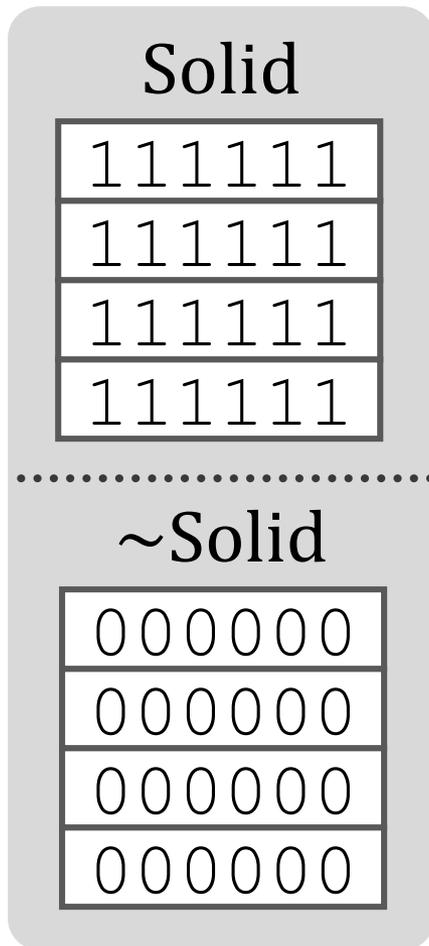
## 2 Refresh Interval



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

*More frequent refreshes → Fewer errors*

# 3 Data Pattern



10x Errors

*Errors affected by data stored in other cells*

# 6. Other Results (in Paper)

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

# 6. Other Results (in Paper) cont'd

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

# Some Potential Solutions

---

- Make better DRAM chips

Cost

- Refresh frequently

Power, Performance

- Sophisticated ECC

Cost, Power

- Access counters

Cost, Power, Complexity

# Naive Solutions

## 1 *Throttle accesses to same row*

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

## 2 *Refresh more frequently*

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

*Both naive solutions introduce significant overhead in performance and power*

# Apple's Patch for RowHammer

---

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

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

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

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

CVE-ID

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

HP and Lenovo released similar patches

---

# Our Solution to RowHammer

- **PARA:** *Probabilistic Adjacent Row Activation*
- **Key Idea**
  - After closing a row, we activate (i.e., refresh) one of its neighbors with a low probability:  $p = 0.005$
- **Reliability Guarantee**
  - When  $p=0.005$ , errors in one year:  $9.4 \times 10^{-14}$
  - 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<sup>1</sup> Ross Daly\* Jeremie Kim<sup>1</sup> Chris Fallin\* Ji Hye Lee<sup>1</sup>  
Donghyuk Lee<sup>1</sup> Chris Wilkerson<sup>2</sup> Konrad Lai Onur Mutlu<sup>1</sup>

<sup>1</sup>Carnegie Mellon University      <sup>2</sup>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>

## Fundamentally Secure, Reliable, Safe Computing Architectures

# Future of Main Memory

---

- DRAM is becoming less reliable → more vulnerable

# Large-Scale Failure Analysis of DRAM Chips

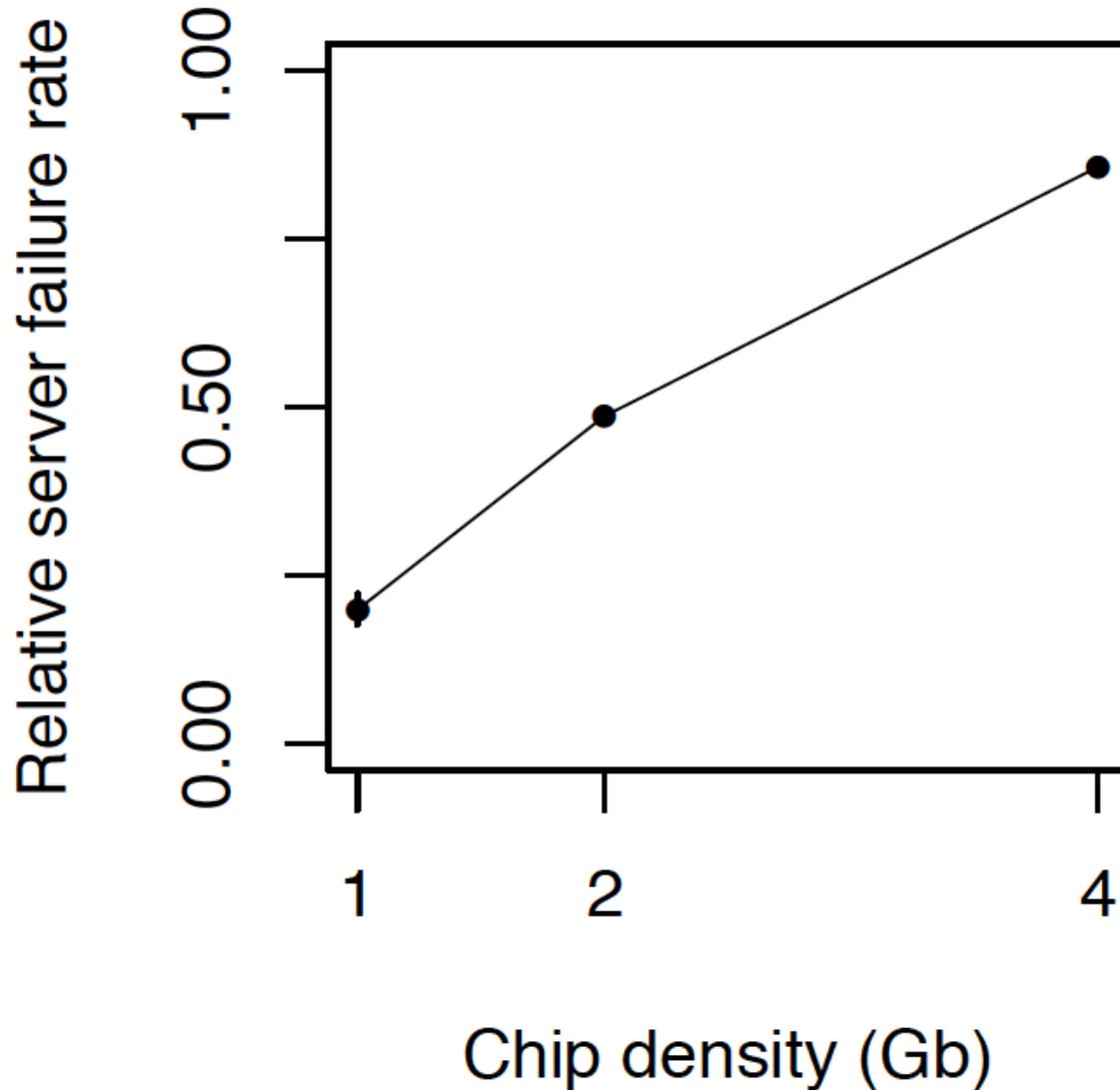
---

- Analysis and modeling of memory errors found in all of Facebook's server fleet
- Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu, **"Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field"** *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Rio de Janeiro, Brazil, June 2015.  
[[Slides \(pptx\)](#)] [[pdf](#)] [[DRAM Error Model](#)]

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

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

# DRAM Reliability Reducing



*Intuition:  
quadratic  
increase in  
capacity*

# Aside: SSD Error Analysis in the Field

---

- First large-scale field study of flash memory errors
- Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu, **"A Large-Scale Study of Flash Memory Errors in the Field"** *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, Portland, OR, June 2015. [[Slides \(pptx\)](#)] [[pdf](#)] [[Coverage at ZDNet](#)]

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

Justin Meza  
Carnegie Mellon University  
meza@cmu.edu

Qiang Wu  
Facebook, Inc.  
qwu@fb.com

Sanjeev Kumar  
Facebook, Inc.  
skumar@fb.com

Onur Mutlu  
Carnegie Mellon University  
onur@cmu.edu

# Future of Main Memory

---

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

# DRAM Data Retention Time Failures

---

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

# Analysis of Retention Failures [ISCA'13]

---

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

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

Jamie Liu\*

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

Ben Jaiyen\*

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

Yoongu Kim

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

Chris Wilkerson

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

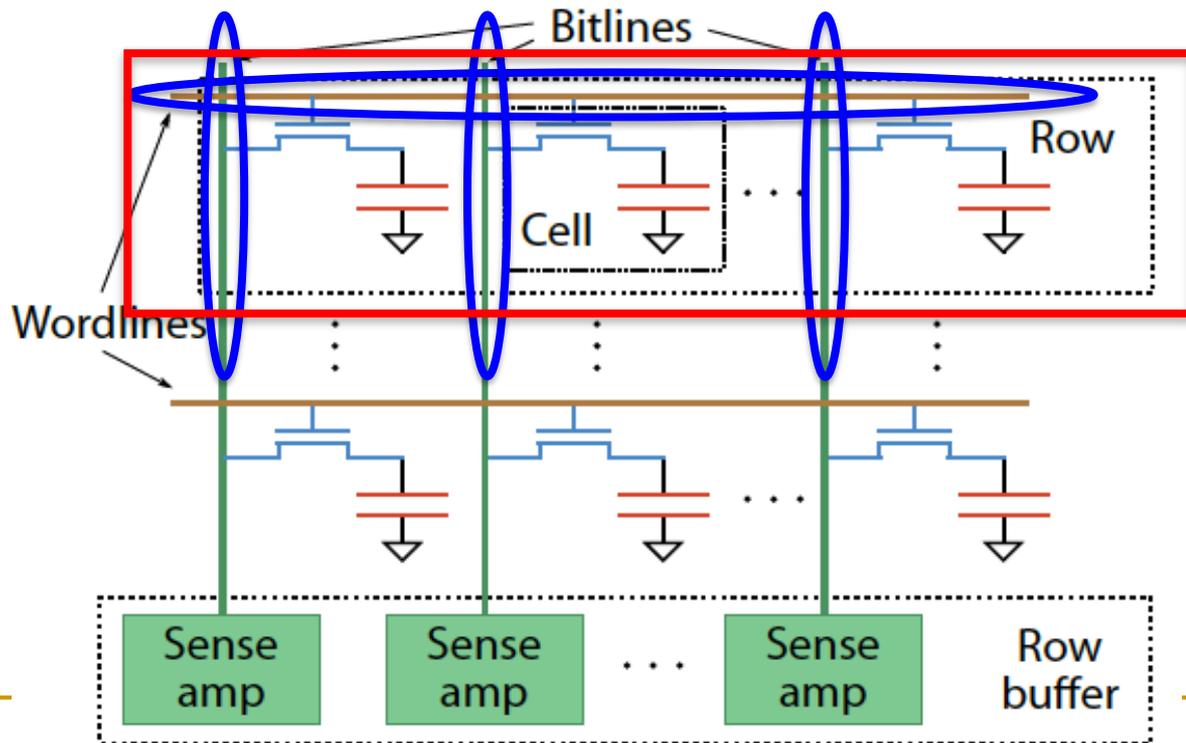
Onur Mutlu

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



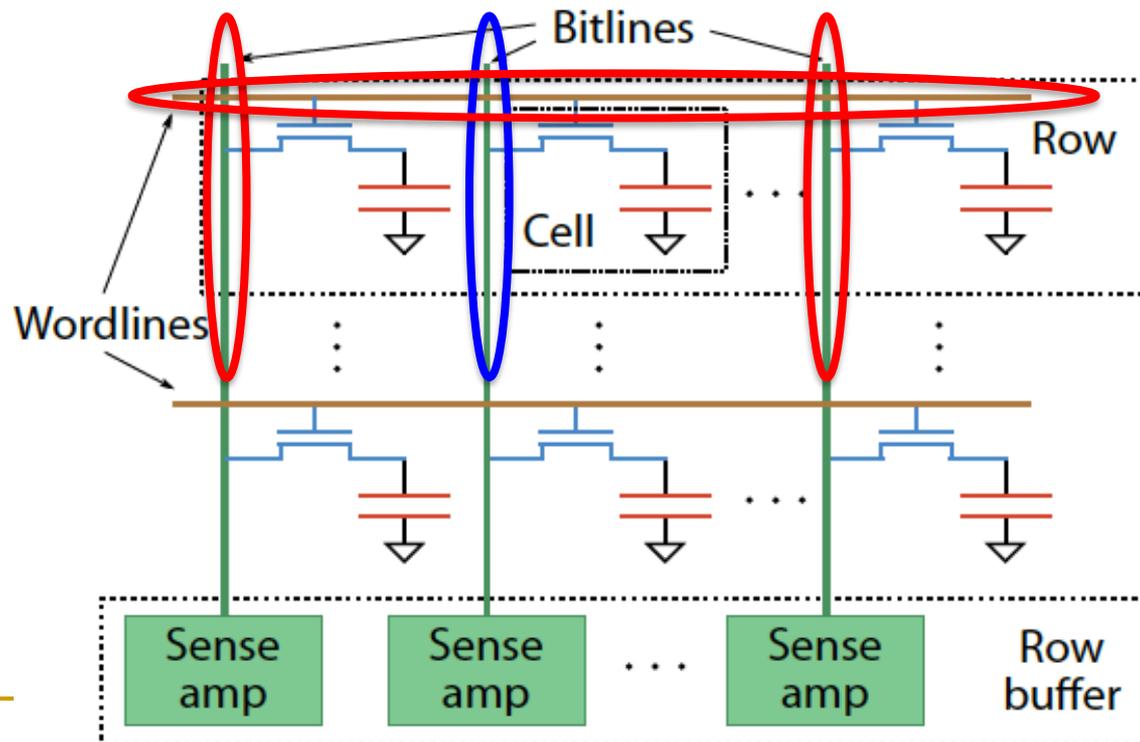
# Two Challenges to Retention Time Profiling

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



# Data Pattern Dependence

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



# Data Pattern Dependence

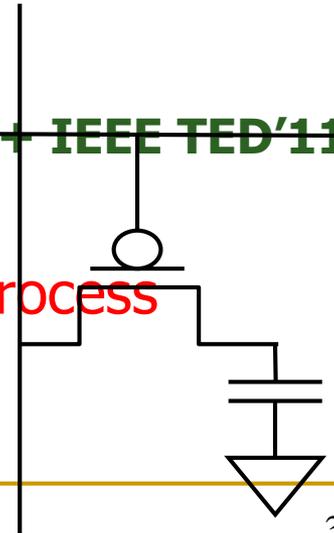
---

- Electrical noise on the bitline affects reliable sensing of a DRAM cell
- The magnitude of this noise is affected by values of nearby cells via
  - Bitline-bitline coupling → electrical coupling between adjacent bitlines
  - Bitline-wordline coupling → electrical coupling between each bitline and the activated wordline
  
- Retention time of a cell depends on data patterns stored in nearby cells
  - need to find the worst data pattern to find worst-case retention time
  - this pattern is location dependent

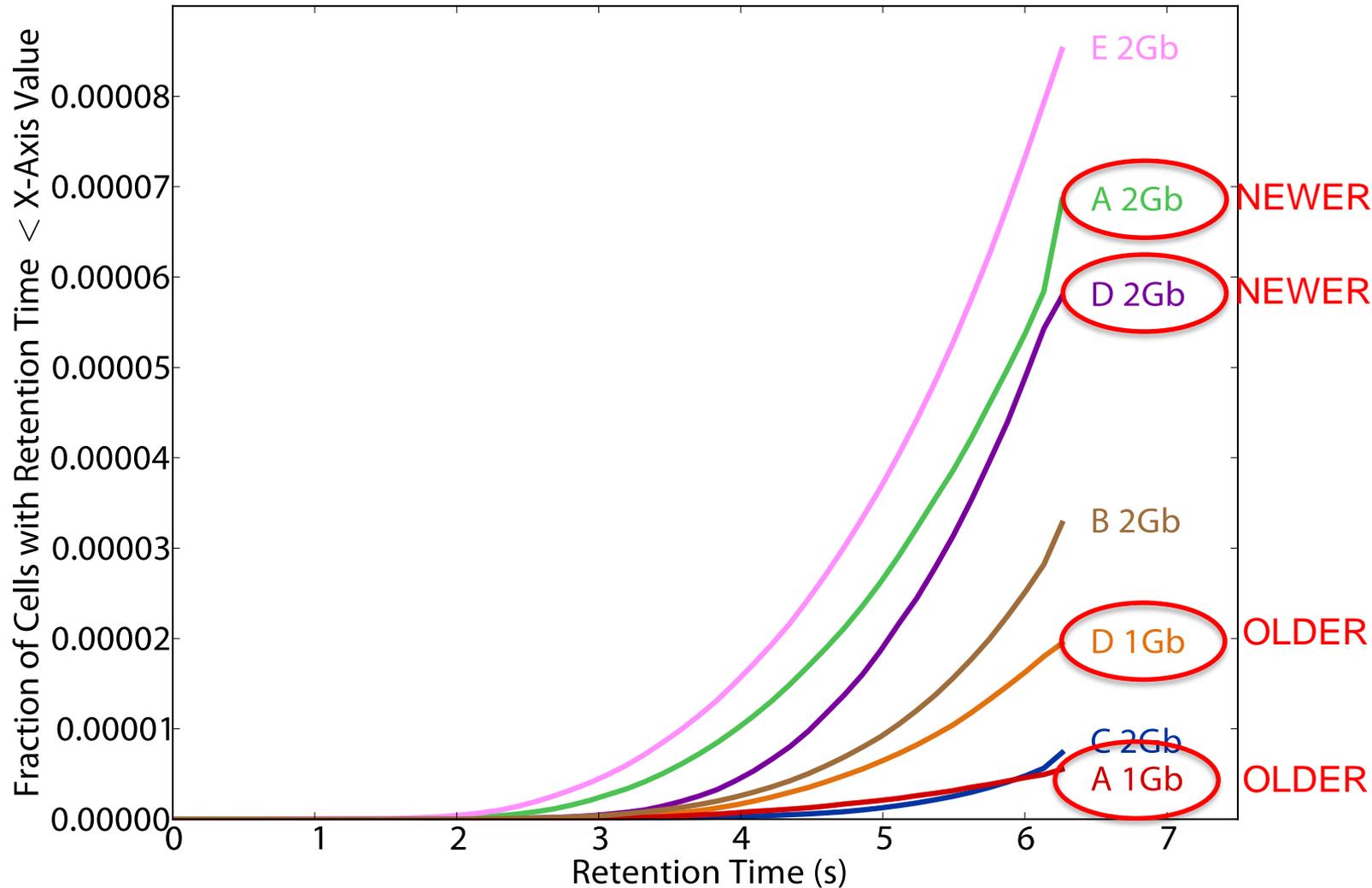
# Two Challenges to Retention Time Profiling

## ■ Challenge 2: Variable Retention Time (VRT)

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

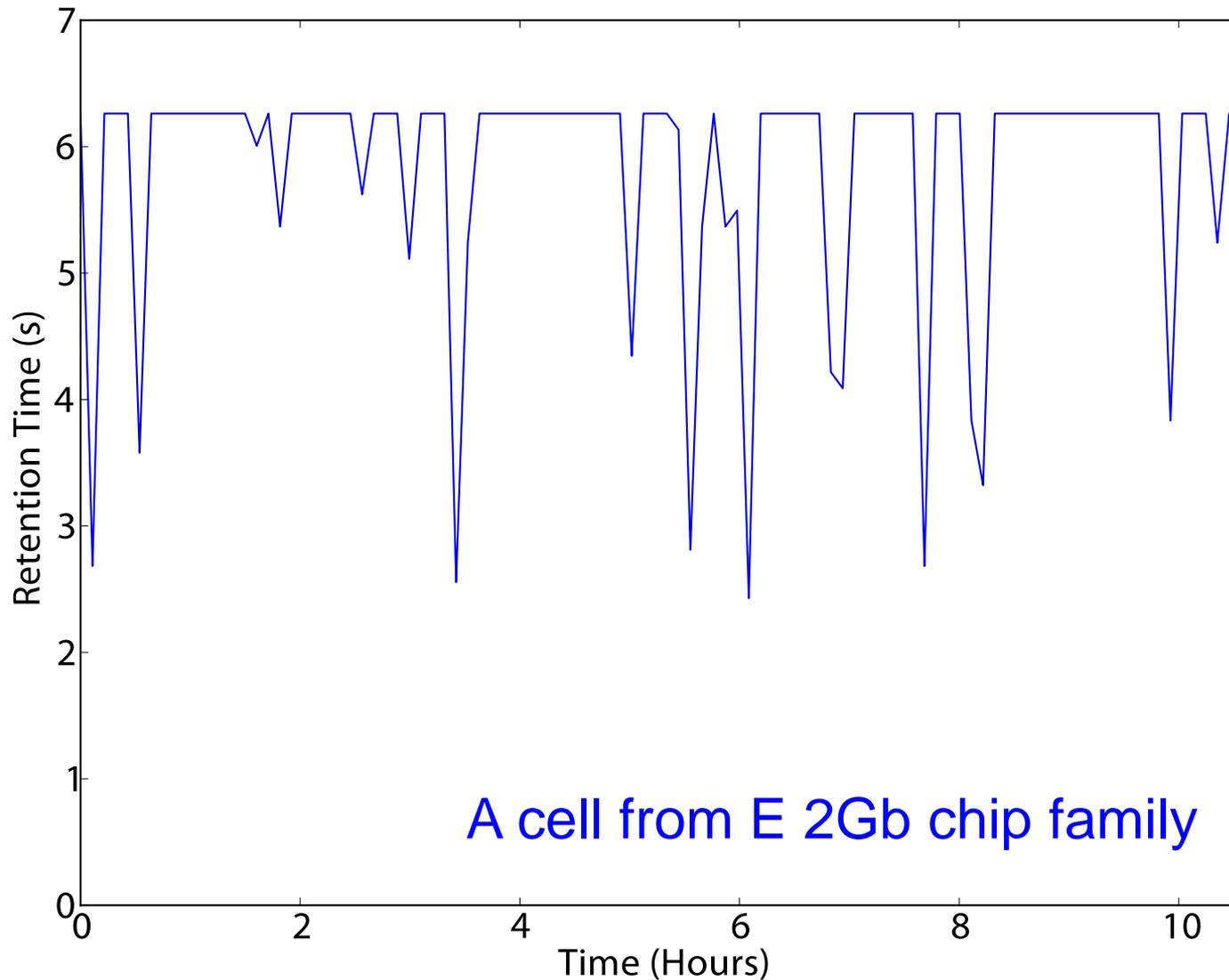


# Modern DRAM Retention Time Distribution

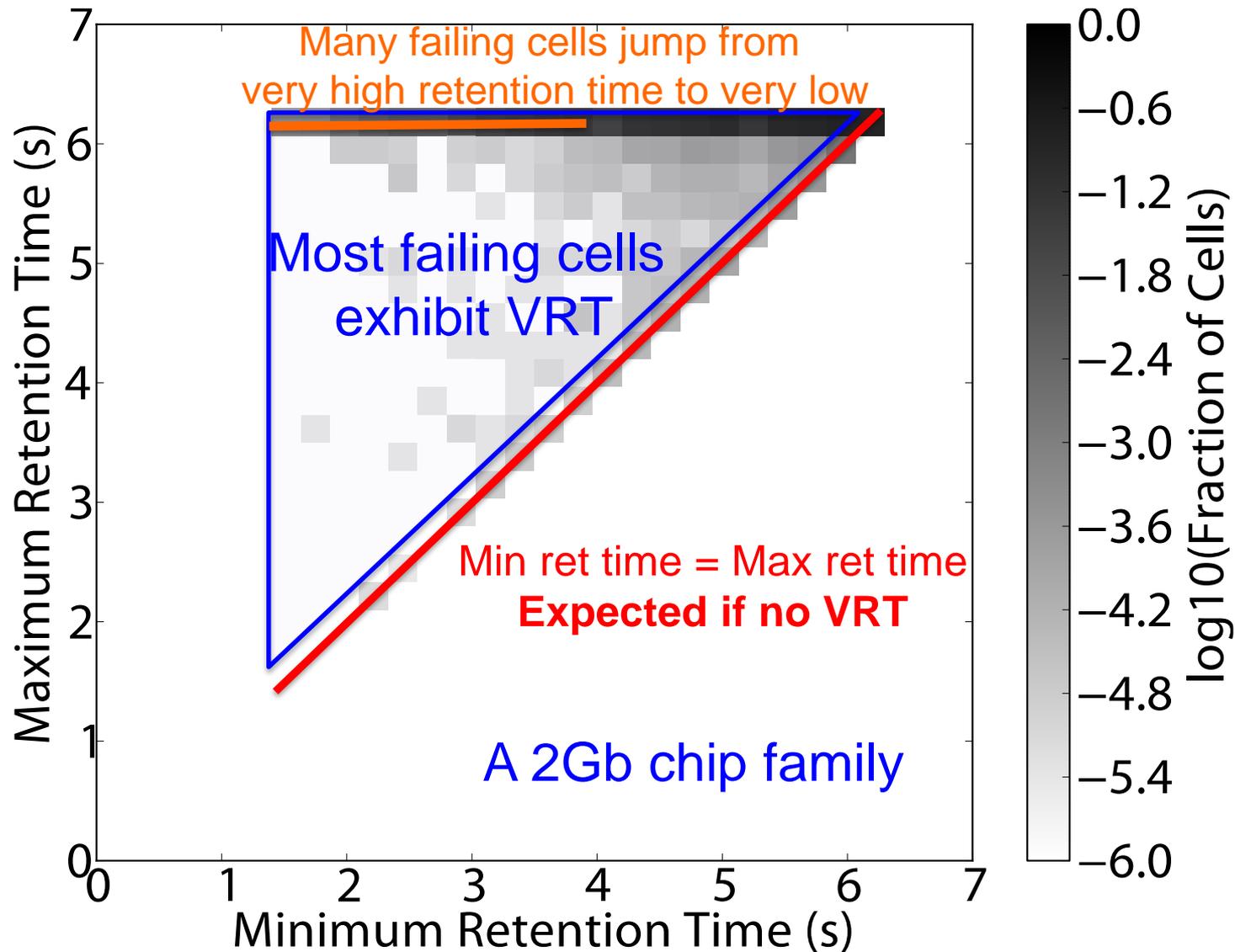


**Newer device families have more weak cells than older ones**  
**Likely a result of technology scaling**

# An Example VRT Cell



# 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

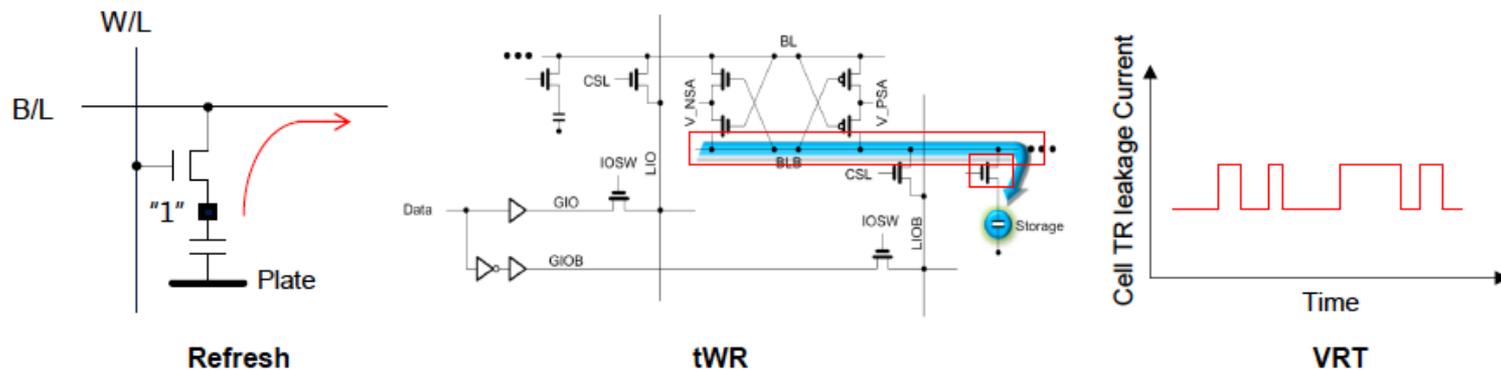
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance
- Leakage current of cell access transistors increasing

### ❖ 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



# Industry Is Writing Papers About It, Too

## DRAM Process Scaling Challenges

### ❖ Refresh

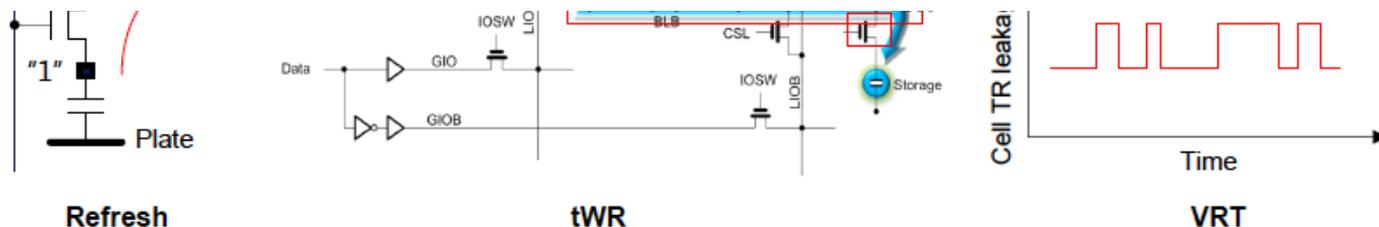
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

THE MEMORY FORUM 2014

# Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling

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

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



# Mitigation of Retention Issues [SIGMETRICS'14]

---

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

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

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

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

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

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

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

Onur Mutlu<sup>†</sup>  
onur@cmu.edu

<sup>†</sup>Carnegie Mellon University

<sup>\*</sup>Intel Labs

# 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<sup>\*</sup>

<sup>\*</sup>University of Virginia

Donghyuk Lee<sup>†‡</sup>

<sup>†</sup>Carnegie Mellon University

Onur Mutlu<sup>\*†</sup>

<sup>‡</sup>Nvidia

<sup>\*</sup>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<sup>\*</sup>, Chris Wilkerson<sup>†</sup>, Donghyuk Lee<sup>‡</sup>, Alaa R. Alameldeen<sup>†</sup>, Onur Mutlu<sup>\*‡</sup>

<sup>\*</sup>University of Virginia    <sup>†</sup>Intel Labs    <sup>‡</sup>Carnegie Mellon University    <sup>\*</sup>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<sup>†</sup>      Dae-Hyun Kim<sup>†</sup>      Samira Khan<sup>‡</sup>      Prashant J. Nair<sup>†</sup>      Onur Mutlu<sup>‡</sup>  
<sup>†</sup>Georgia Institute of Technology      <sup>‡</sup>Carnegie Mellon University  
{*moin, dhkim, pnair6*}@ece.gatech.edu      {*samirakhan, onur*}@cmu.edu

# Handling Both DPD and VRT [ISCA'17]

---

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

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

Minesh Patel<sup>§‡</sup>   Jeremie S. Kim<sup>‡§</sup>   Onur Mutlu<sup>§‡</sup>  
§ETH Zürich   ‡Carnegie Mellon University

# 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.

---

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

# Overview Paper on Flash Reliability

---

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

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

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

# Fundamentally Secure, Reliable, Safe Computing Architectures

# NAND Flash Memory

## Reliability and Security

# Upcoming Overview Paper

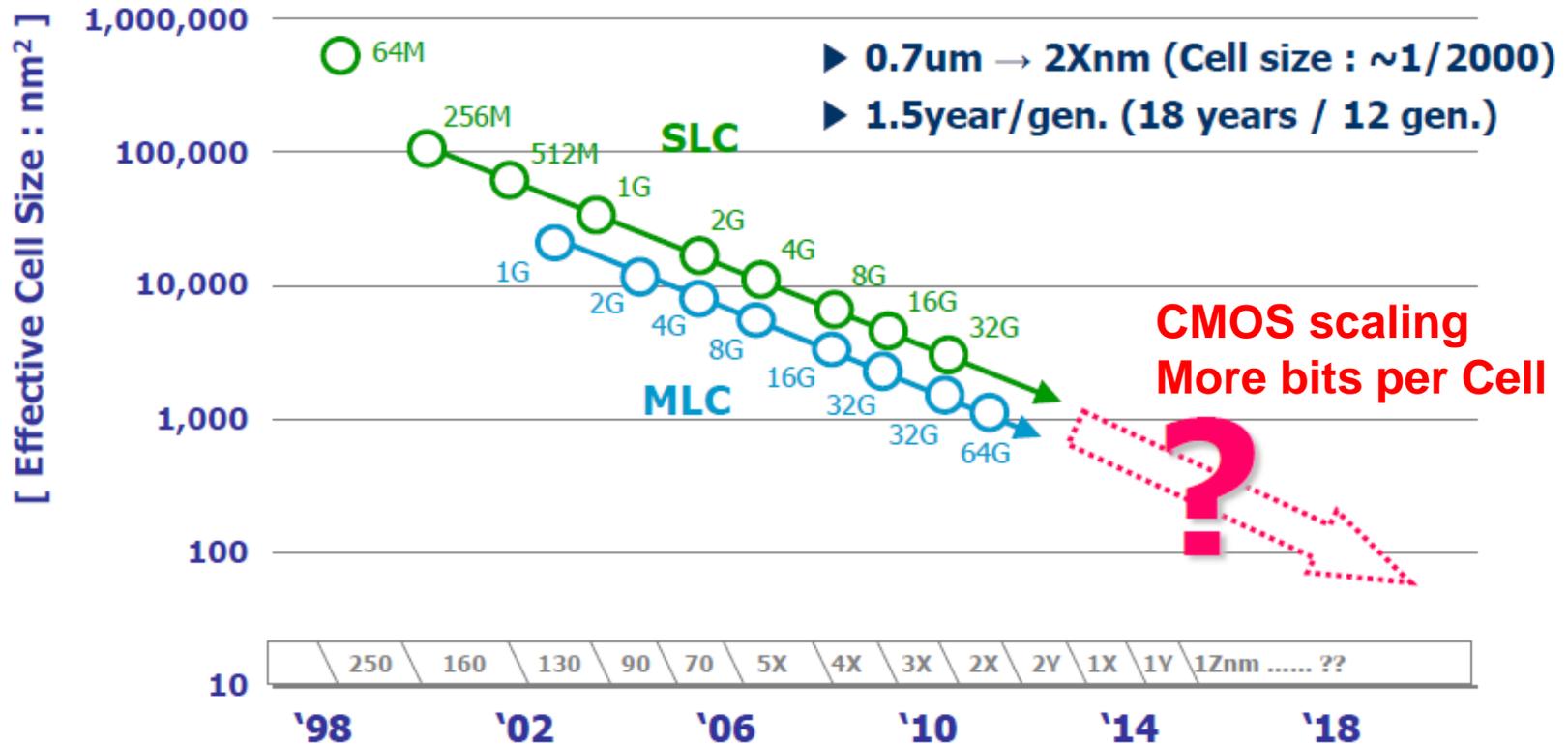
---

- 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.*

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

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

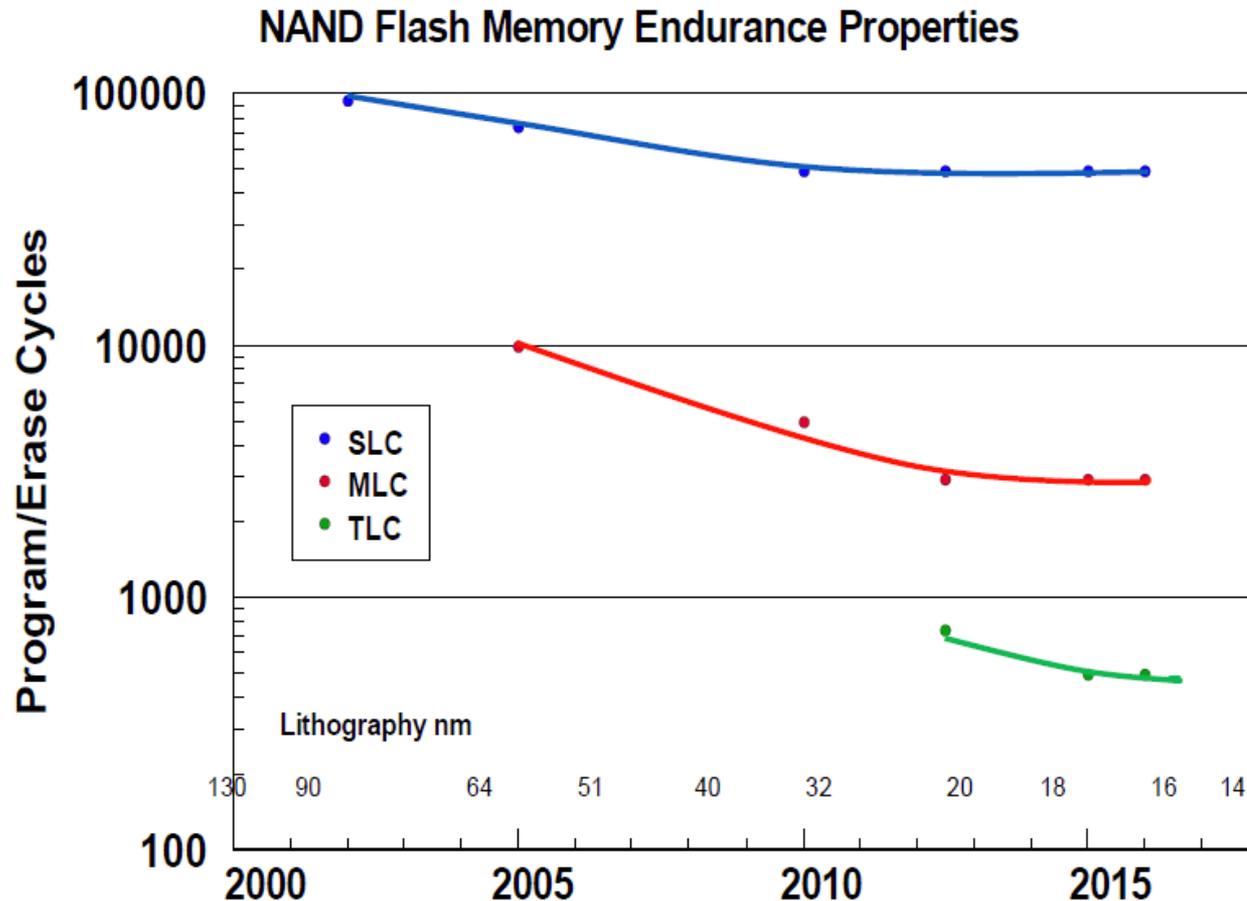
# Evolution of NAND Flash Memory



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

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

# Flash Challenges: Reliability and Endurance



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

A few thousand

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

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

> 50k P/E cycles

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

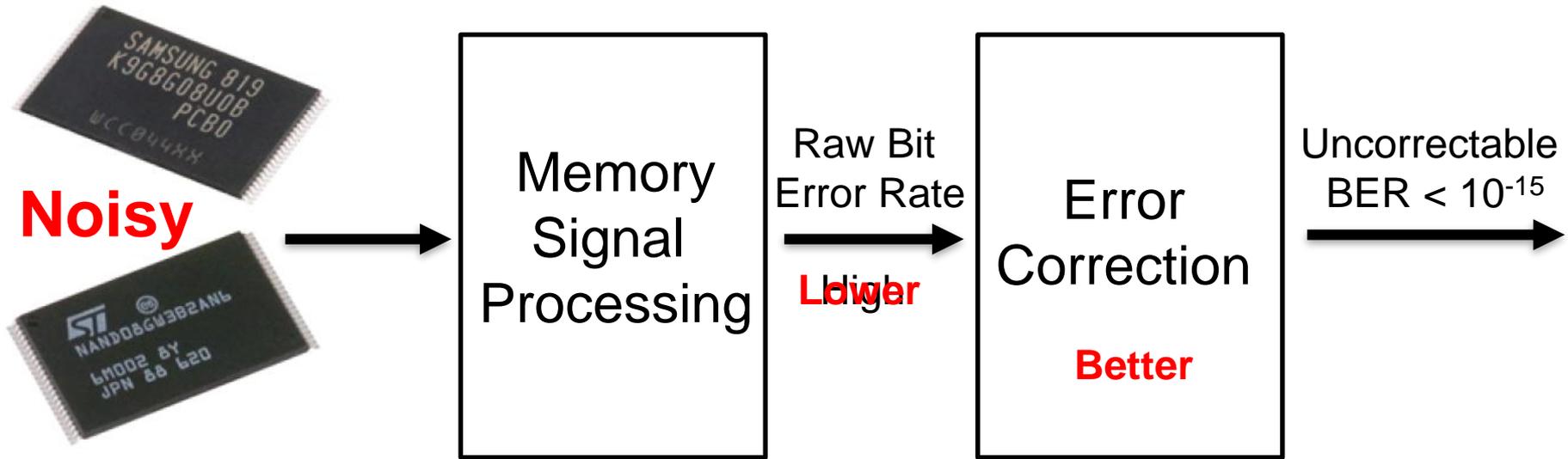
# NAND Flash Memory is Increasingly Noisy

---



# Future NAND Flash-based Storage Architecture

---



## Our Goals:

Build reliable error models for NAND flash memory

Design efficient reliability mechanisms based on the model

# NAND Flash Error Model



## Experimentally characterize and model dominant errors

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

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



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

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

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

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

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

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

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

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

# Our Goals and Approach

---

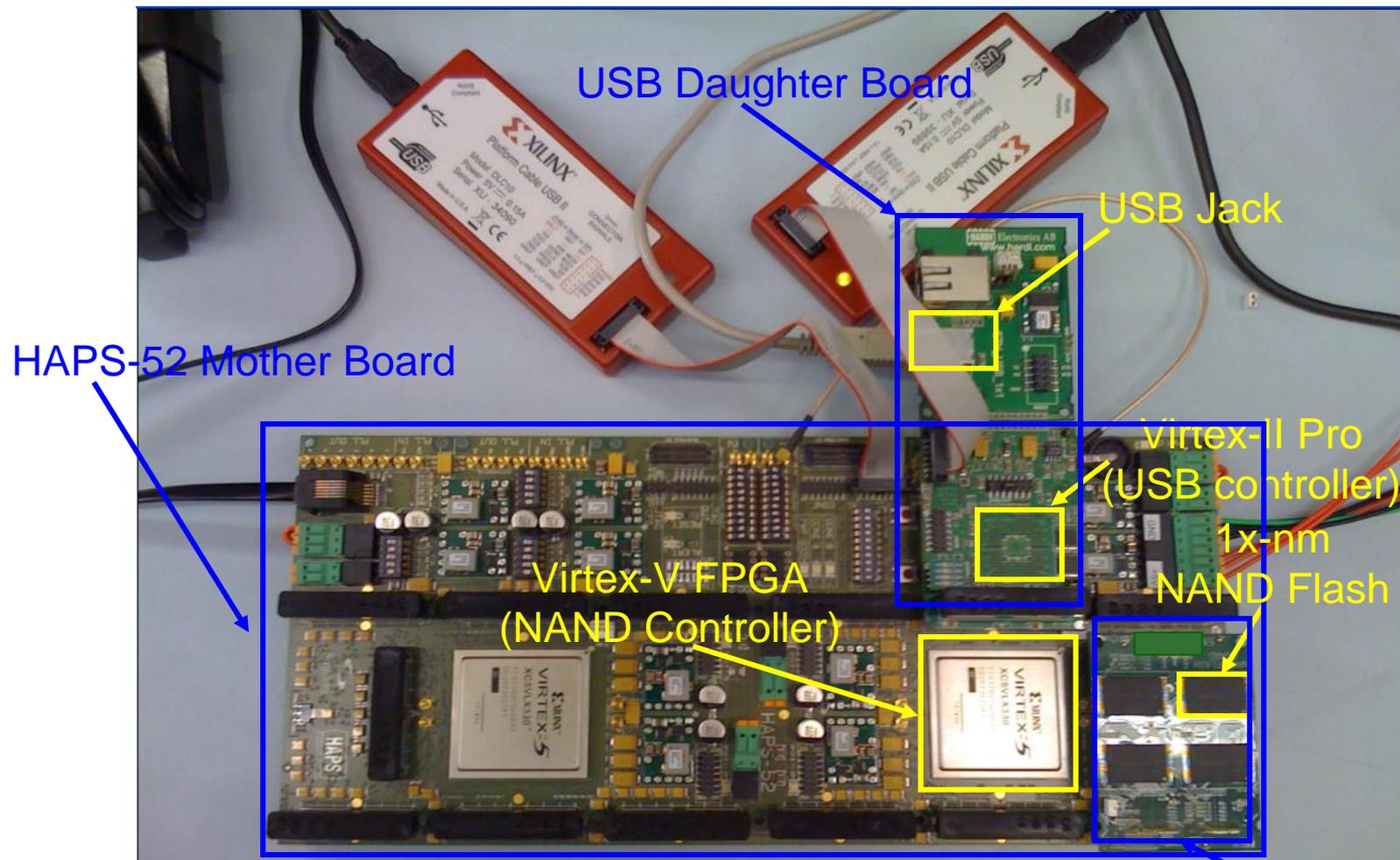
## ■ Goals:

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

## ■ Approach:

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

# Experimental Testing Platform



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

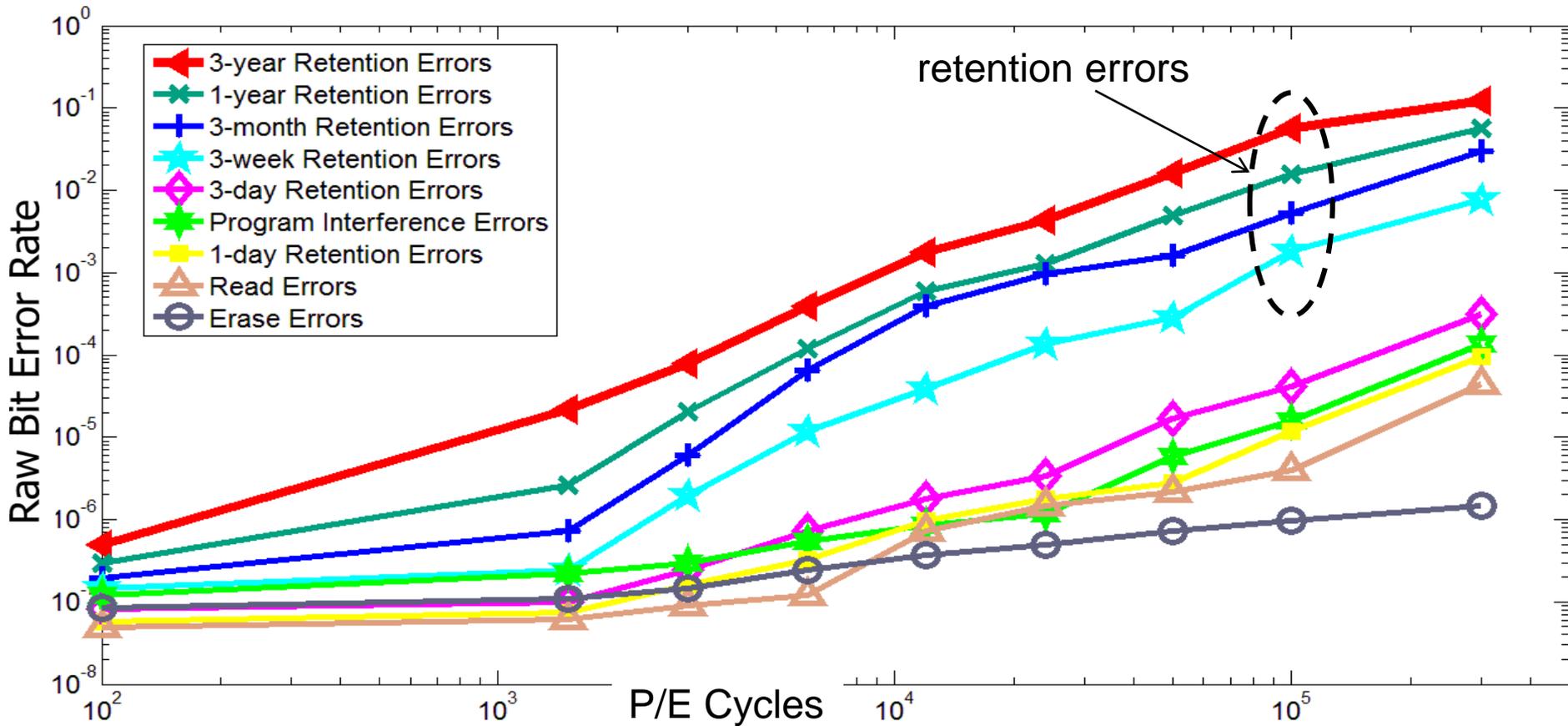
NAND Daughter Board

# NAND Flash Error Types

---

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

# Observations: Flash Error Analysis



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

# More on Flash Error Analysis

---

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

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

Yu Cai<sup>1</sup>, Erich F. Haratsch<sup>2</sup>, Onur Mutlu<sup>1</sup> and Ken Mai<sup>1</sup>

<sup>1</sup>Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA

<sup>2</sup>LSI Corporation, 1110 American Parkway NE, Allentown, PA

<sup>1</sup>{yucai, onur, kenmai}@andrew.cmu.edu, <sup>2</sup>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<sup>1</sup>, Gulay Yalcin<sup>2</sup>, Onur Mutlu<sup>1</sup>, Erich F. Haratsch<sup>3</sup>, Adrian Cristal<sup>2</sup>, Osman S. Unsal<sup>2</sup> and Ken Mai<sup>1</sup>

<sup>1</sup>DSSC, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA

<sup>2</sup>Barcelona Supercomputing Center, C/Jordi Girona 29, Barcelona, Spain

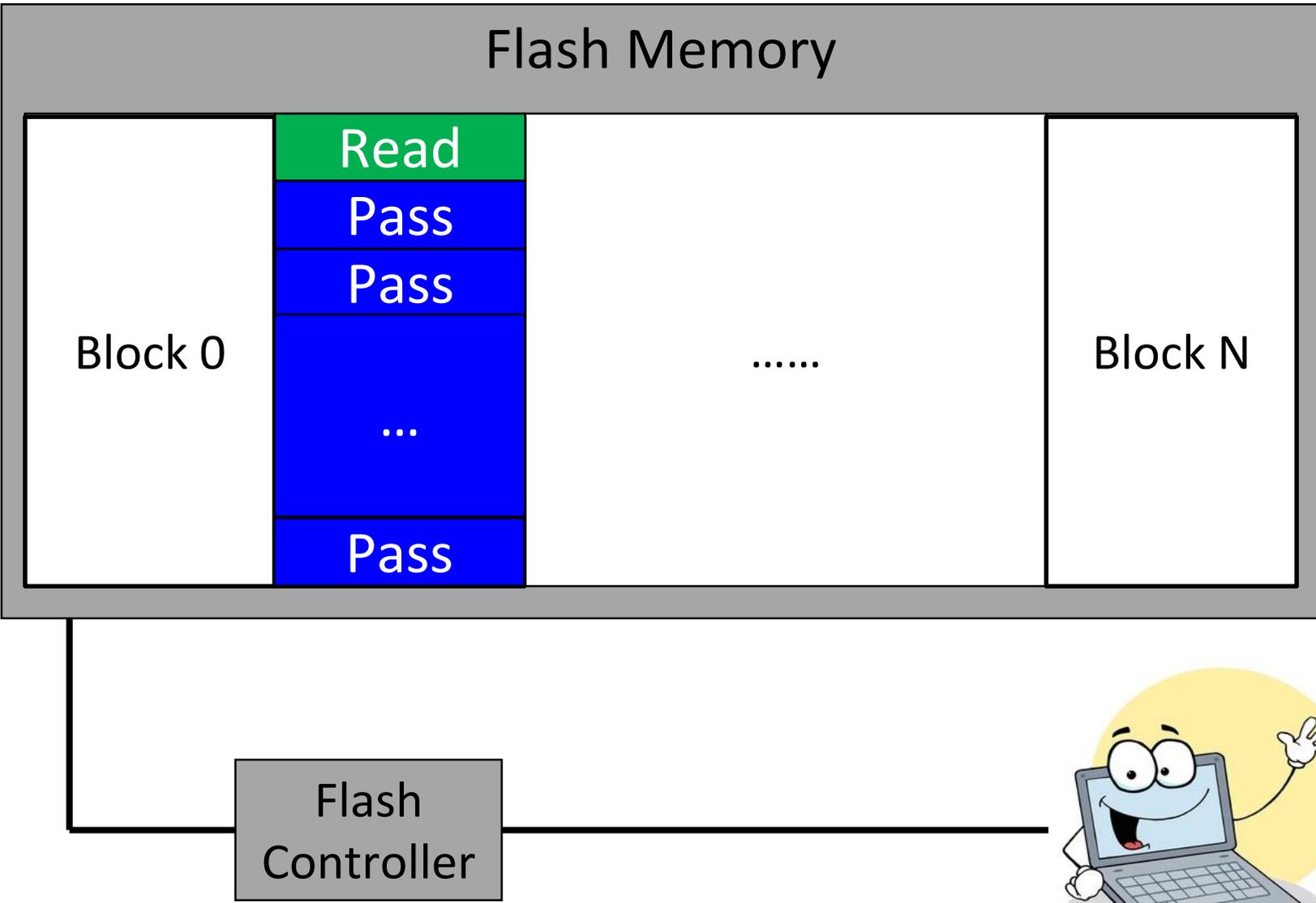
<sup>3</sup>LSI Corporation, 1110 American Parkway NE, Allentown, PA

# One Issue: Read Disturb in Flash Memory

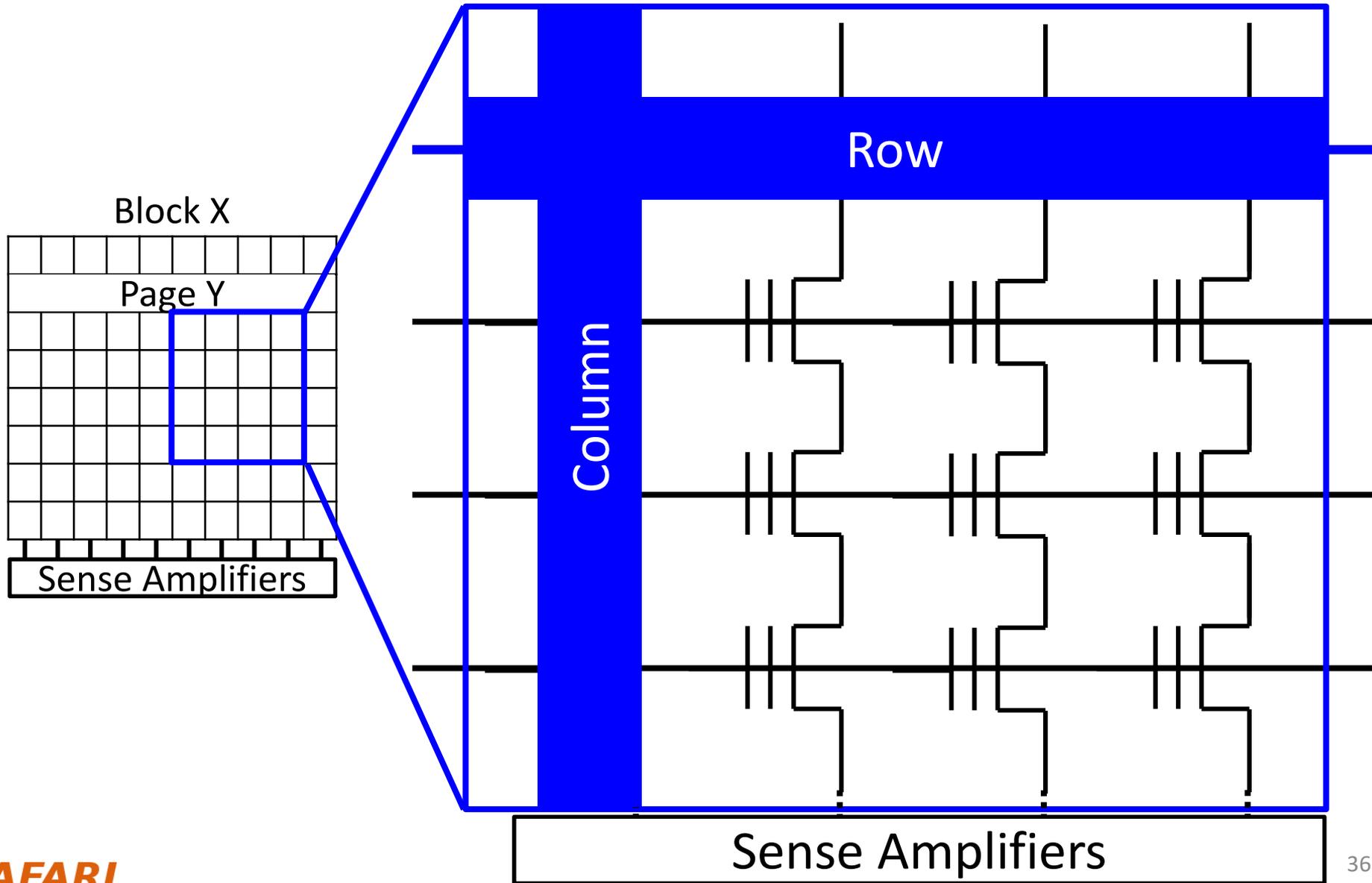
---

- All scaled memories are prone to read disturb errors

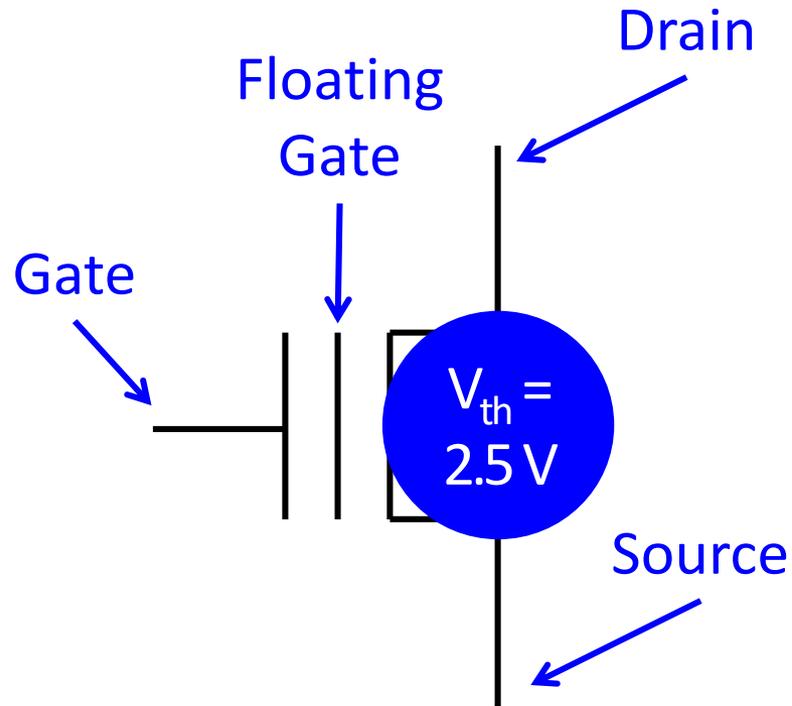
# NAND Flash Memory Background



# Flash Cell Array

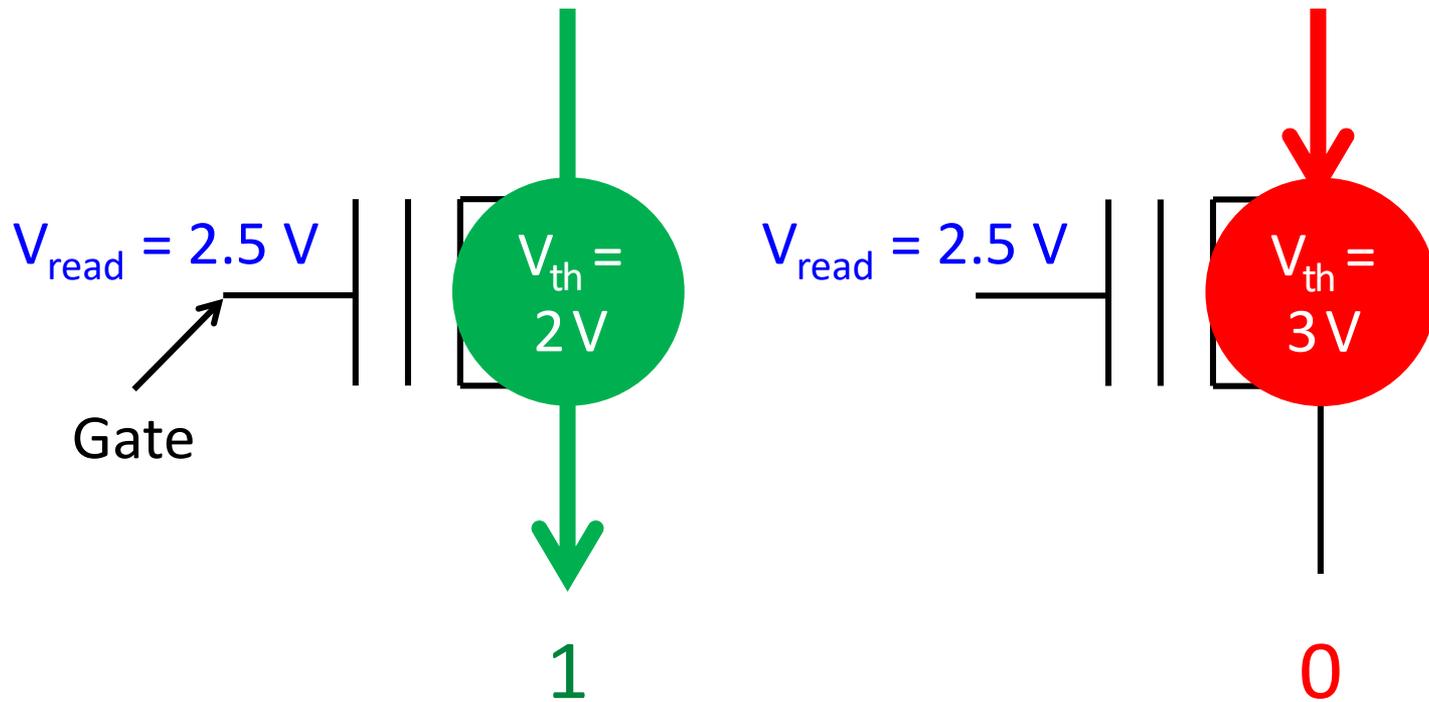


# Flash Cell

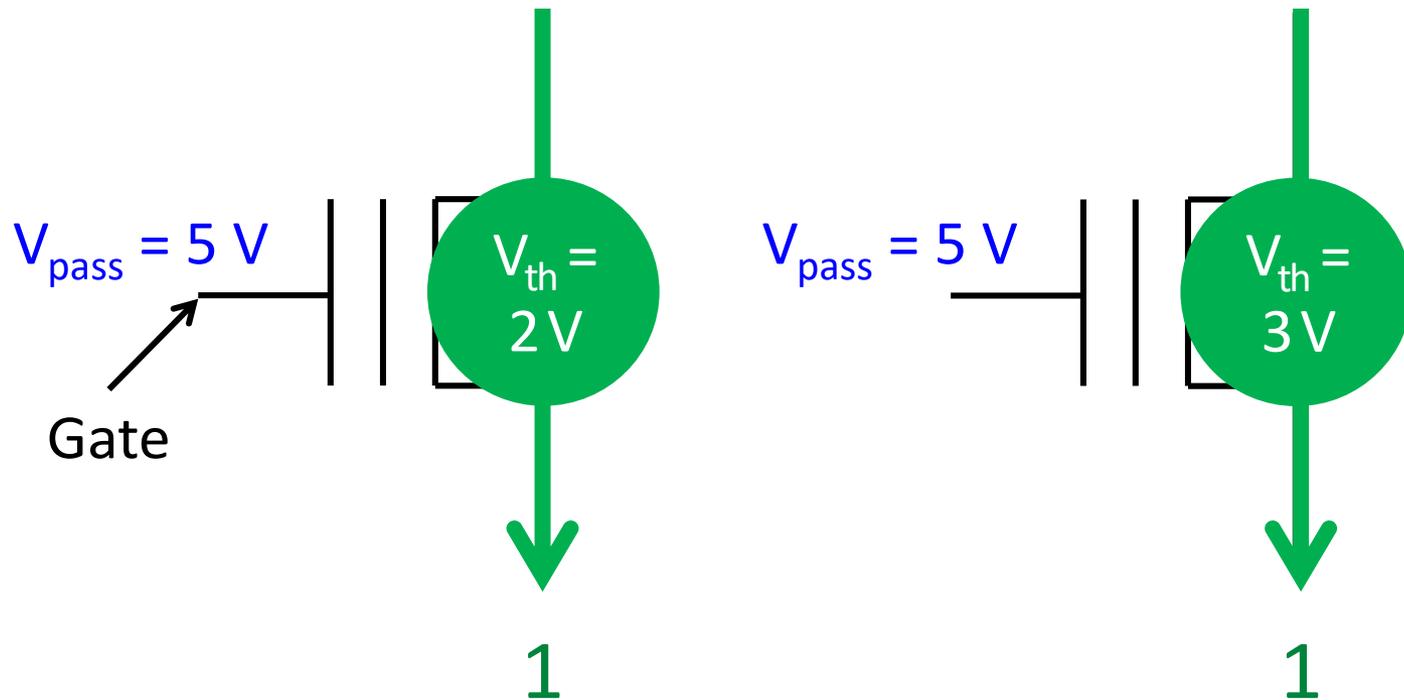


Floating Gate Transistor  
(Flash Cell)

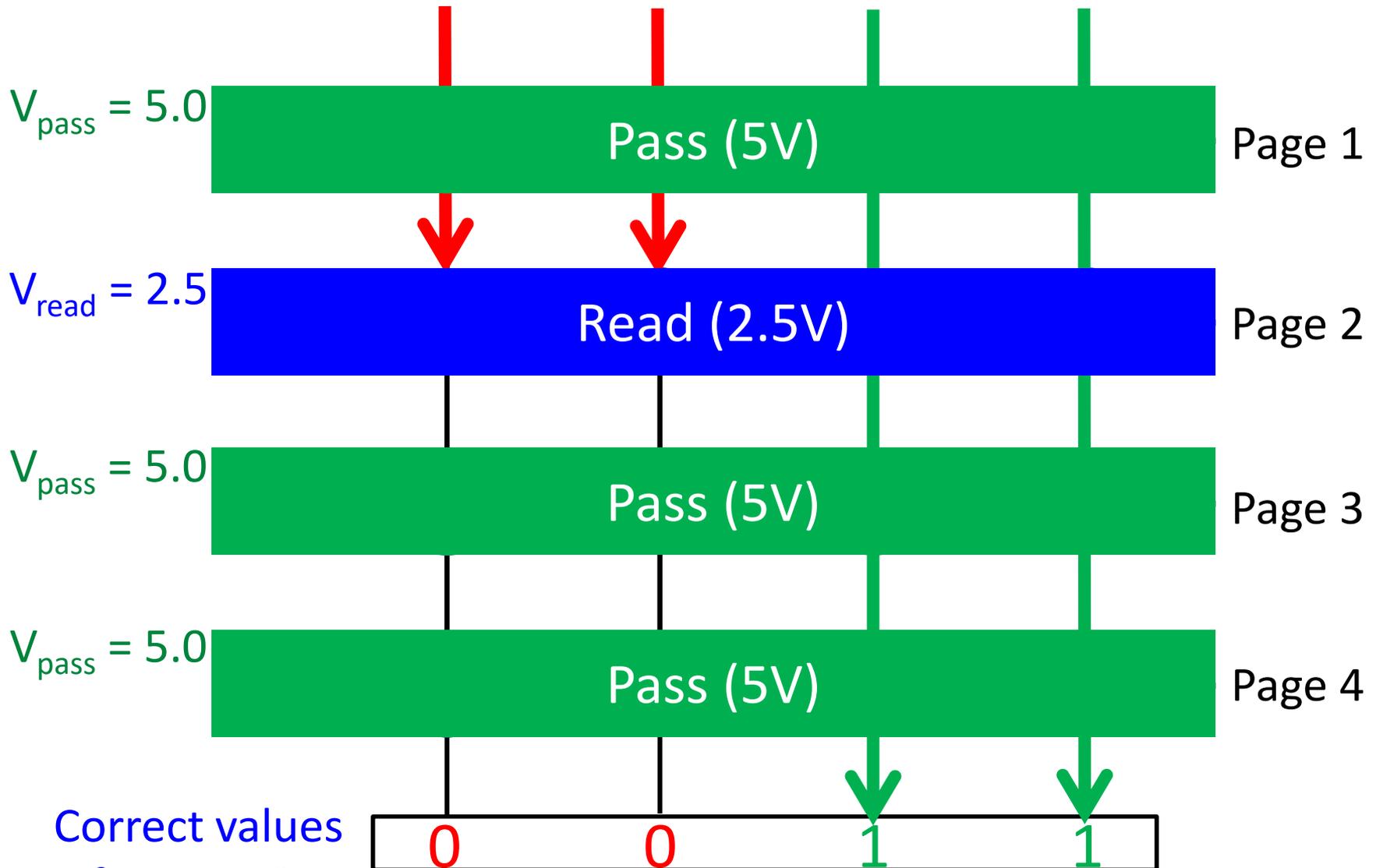
# Flash Read



# Flash Pass-Through

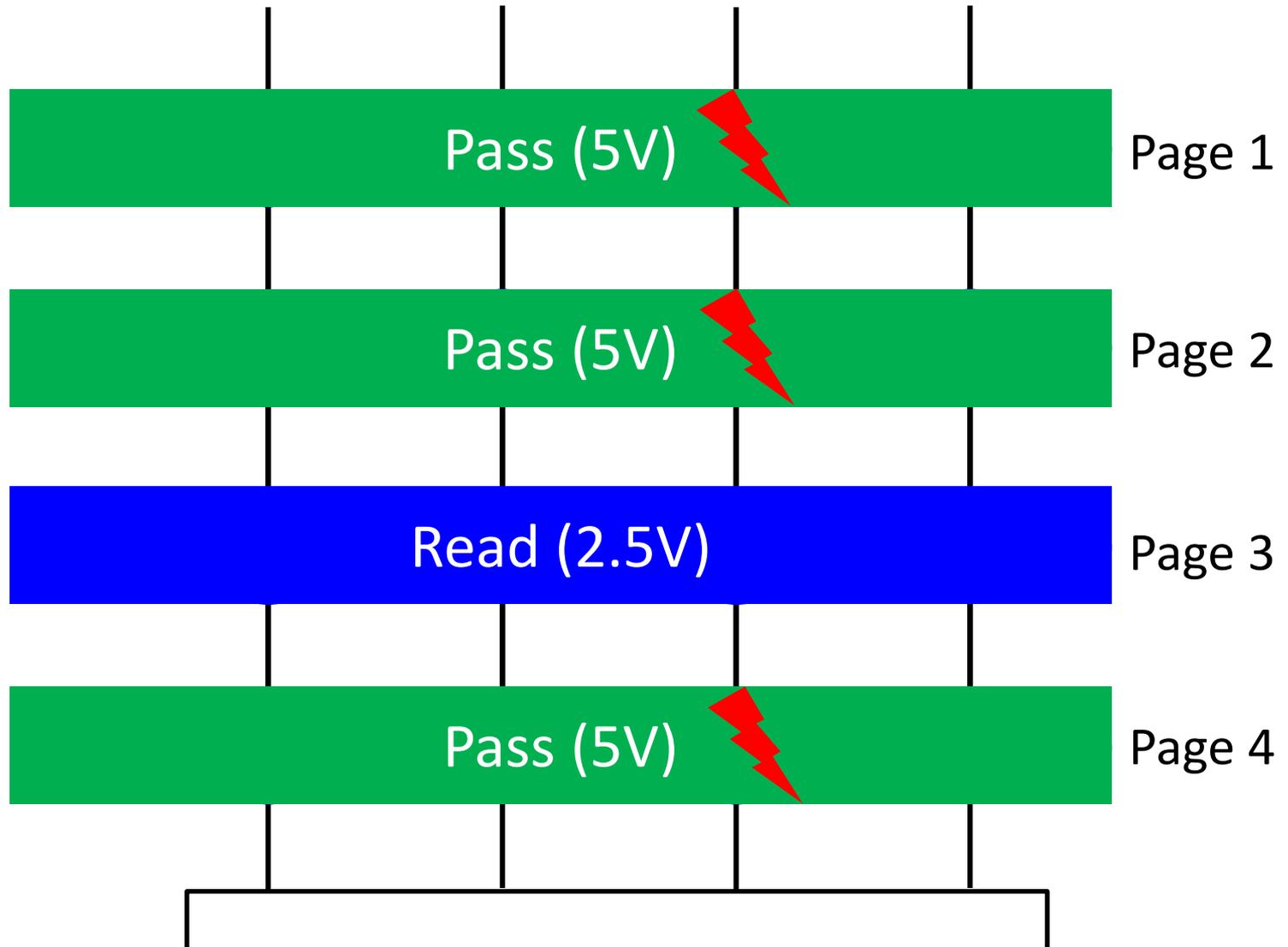


# Read from Flash Cell Array



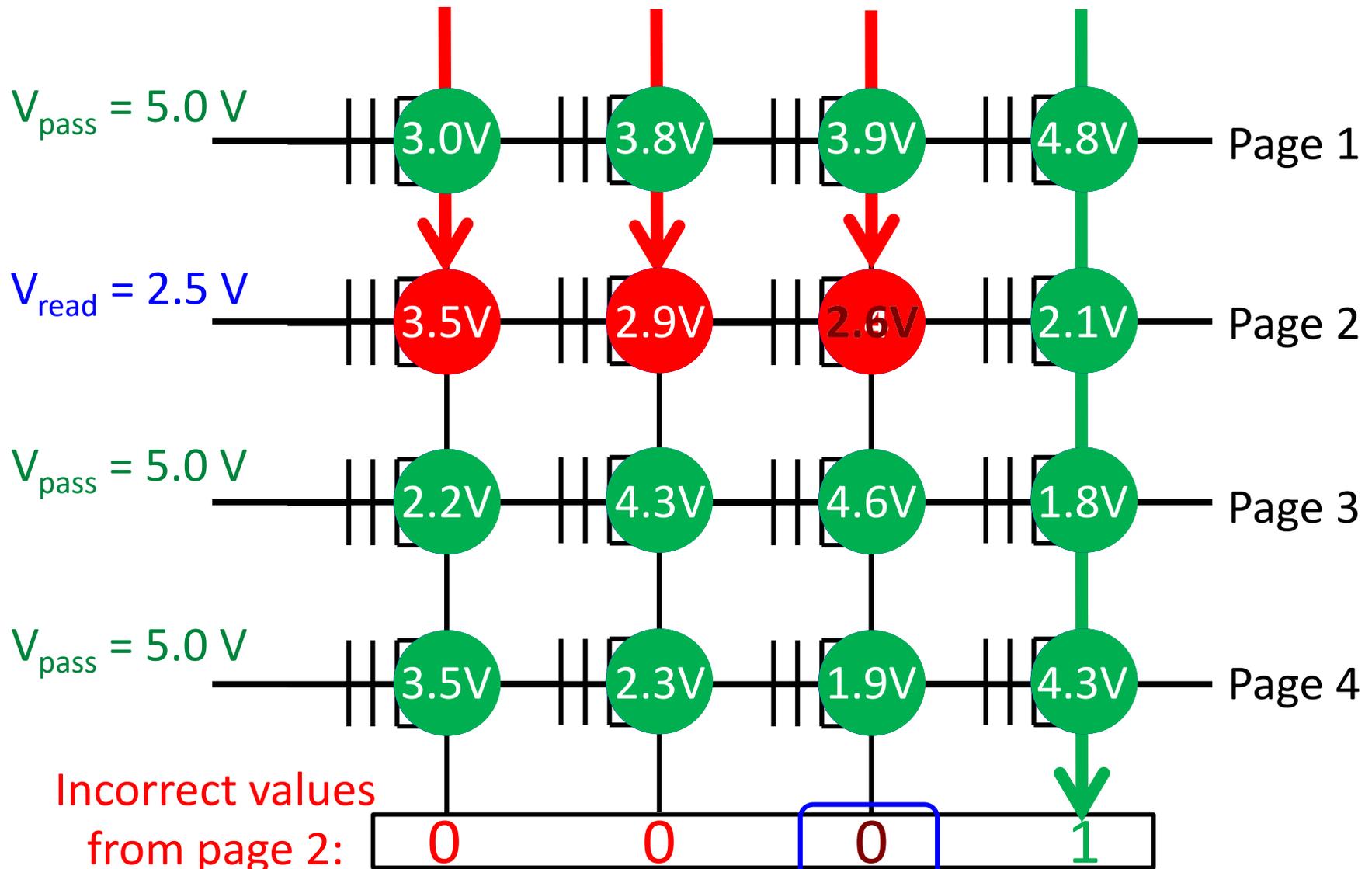
Correct values  
for page 2:

# Read Disturb Problem: “Weak Programming” Effect



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

# Read Disturb Problem: "Weak Programming" Effect



# 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](mailto:yucaicai@gmail.com), {[yixinluo](mailto:yixinluo@cmu.edu), [ghose](mailto:ghose@cmu.edu), [kenmai](mailto:kenmai@cmu.edu), [onur](mailto:onur@cmu.edu)}@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<sup>†</sup>      Saugata Ghose<sup>†</sup>      Yixin Luo<sup>‡†</sup>      Ken Mai<sup>†</sup>      Onur Mutlu<sup>§†</sup>      Erich F. Haratsch<sup>‡</sup>  
<sup>†</sup>*Carnegie Mellon University*      <sup>‡</sup>*Seagate Technology*      <sup>§</sup>*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<sup>1</sup>, Erich F. Haratsch<sup>2</sup>, Onur Mutlu<sup>1</sup> and Ken Mai<sup>1</sup>

<sup>1</sup>DSSC, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA

<sup>2</sup>LSI Corporation, 1110 American Parkway NE, Allentown, PA

<sup>1</sup>{yucai, onur, kenmai}@andrew.cmu.edu, <sup>2</sup>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<sup>1</sup>, Onur Mutlu<sup>1</sup>, Erich F. Haratsch<sup>2</sup> and Ken Mai<sup>1</sup>

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<sup>1</sup>, Gulay Yalcin<sup>2</sup>, Onur Mutlu<sup>1</sup>, Erich F. Haratsch<sup>4</sup>,  
Osman Unsal<sup>2</sup>, Adrian Cristal<sup>2,3</sup>, and Ken Mai<sup>1</sup>

<sup>1</sup>Electrical and Computer Engineering Department, Carnegie Mellon University

<sup>2</sup>Barcelona Supercomputing Center, Spain

<sup>3</sup>IIIA – CSIC – Spain National Research Council

<sup>4</sup>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](mailto:yucaicai@gmail.com), [yixinluo@cs.cmu.edu](mailto:yixinluo@cs.cmu.edu), [erich.haratsch@lsi.com](mailto:erich.haratsch@lsi.com), {[kenmai](mailto:kenmai@ece.cmu.edu), [omutlu](mailto:omutlu@ece.cmu.edu)}@ece.cmu.edu

# SSD Error Analysis in the Field

---

- First large-scale field study of flash memory errors
- Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu,  
**"A Large-Scale Study of Flash Memory Errors in the Field"**  
*Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, Portland, OR, June 2015.  
[\[Slides \(pptx\) \(pdf\)\]](#) [\[Coverage at ZDNet\]](#) [\[Coverage on The Register\]](#) [\[Coverage on TechSpot\]](#) [\[Coverage on The Tech Report\]](#)

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

Justin Meza  
Carnegie Mellon University  
meza@cmu.edu

Qiang Wu  
Facebook, Inc.  
qwu@fb.com

Sanjeev Kumar  
Facebook, Inc.  
skumar@fb.com

Onur Mutlu  
Carnegie Mellon University  
onur@cmu.edu

# Flash Memory Programming Vulnerabilities

---

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

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

Yu Cai<sup>†</sup>      Saugata Ghose<sup>†</sup>      Yixin Luo<sup>‡†</sup>      Ken Mai<sup>†</sup>      Onur Mutlu<sup>§†</sup>      Erich F. Haratsch<sup>‡</sup>  
                  <sup>†</sup>*Carnegie Mellon University*      <sup>‡</sup>*Seagate Technology*      <sup>§</sup>*ETH Zürich*

# Accurate and Online Channel Modeling

---

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

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

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

# More on DRAM Refresh

# Tackling Refresh: Solutions

---

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

# Summary: Refresh-Access Parallelization

---

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

# Refresh Penalty

---



*Refresh delays requests by 100s of ns*

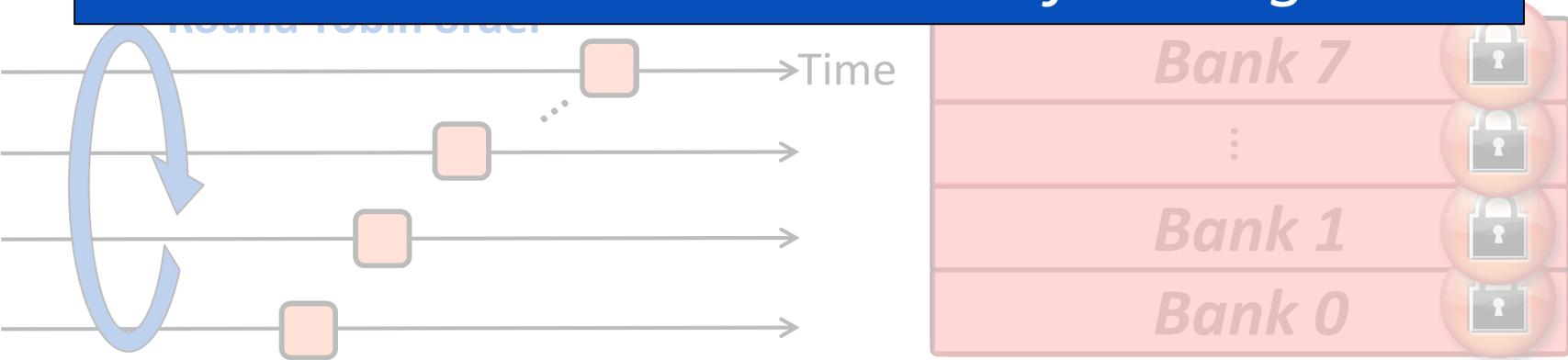
# Existing Refresh Modes

All-bank refresh in commodity DRAM (DDR<sub>x</sub>)



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

Per



# Shortcomings of Per-Bank Refresh

---

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

# Our First Approach: DARP

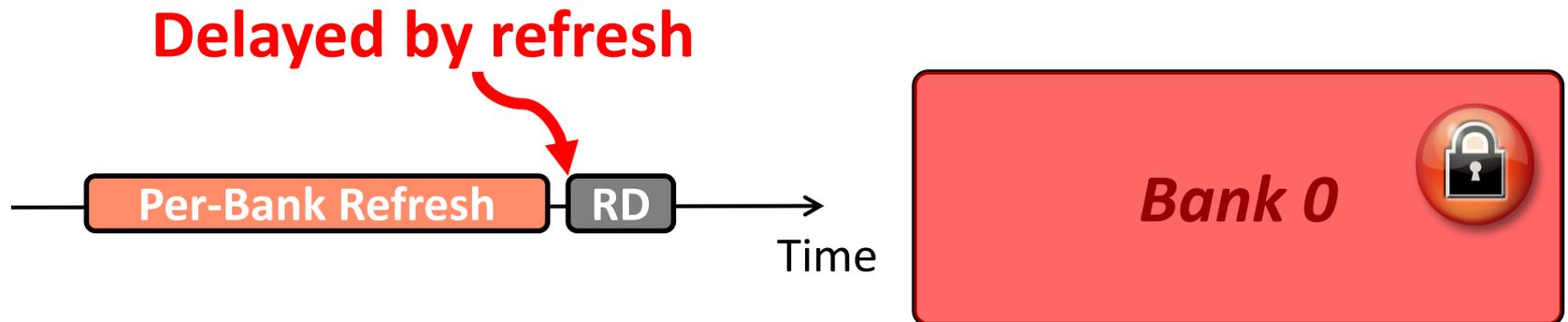
---

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

# Shortcomings of Per-Bank Refresh

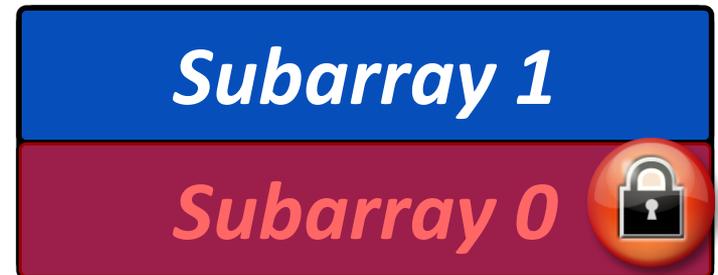
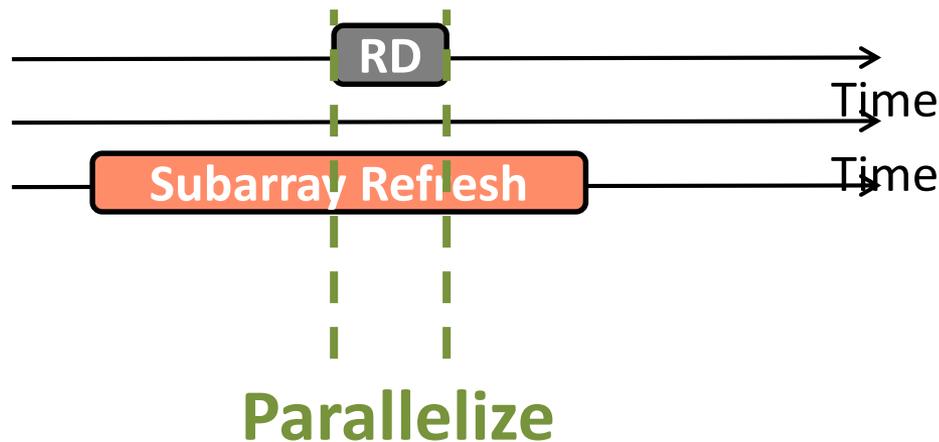
---

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



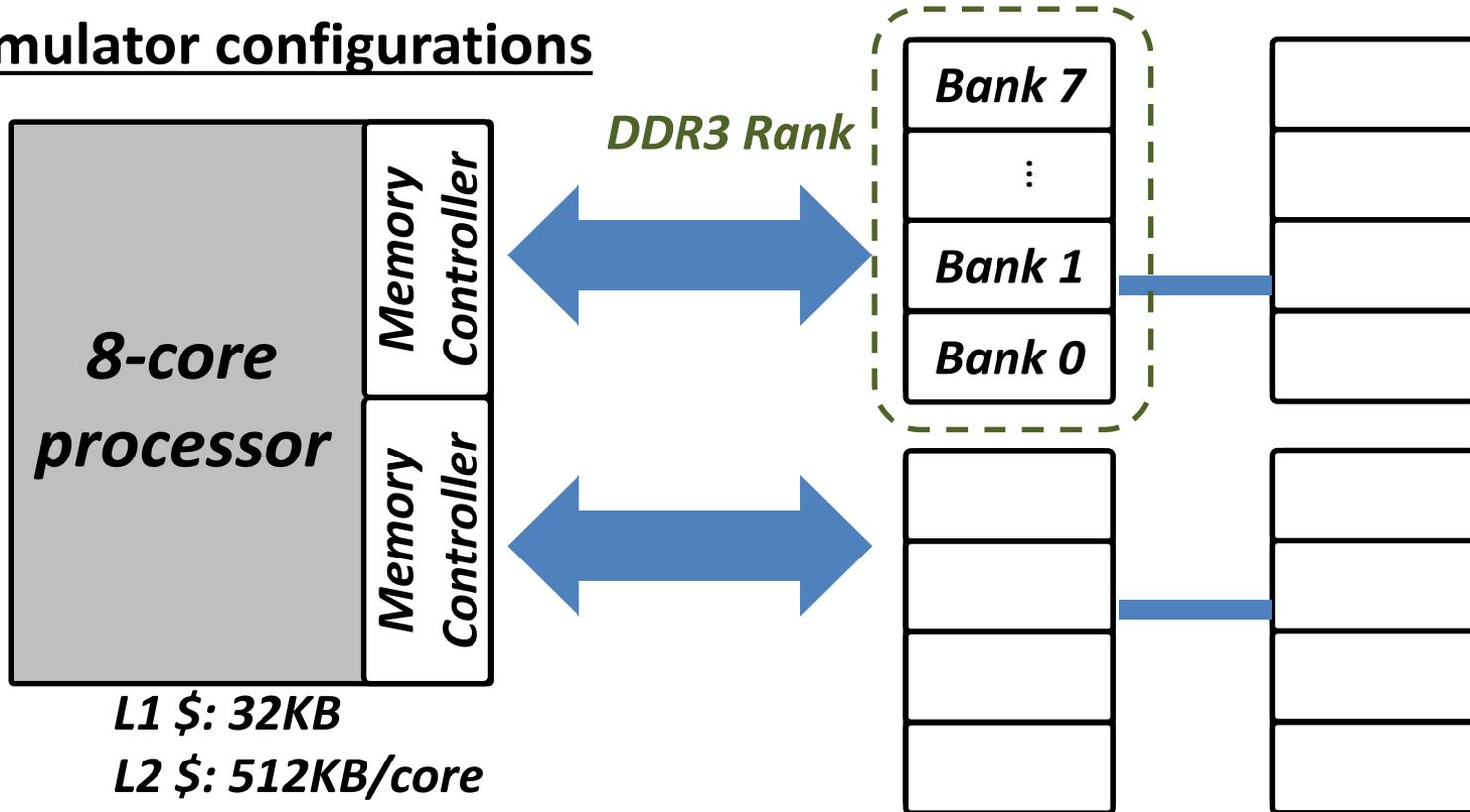
# Shortcomings of Per-Bank Refresh

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



# Methodology

## Simulator configurations



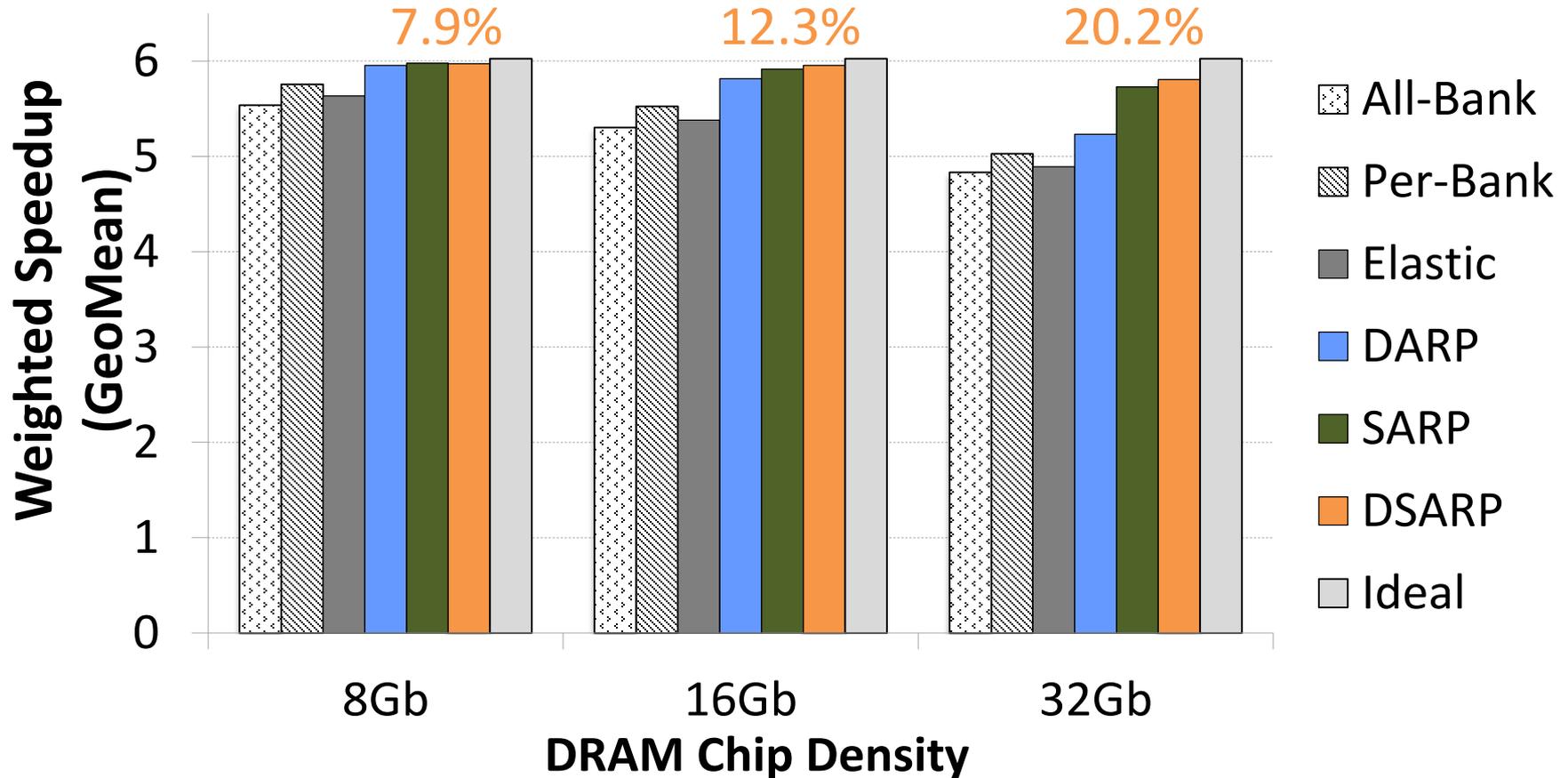
- **100 workloads**: SPEC CPU2006, STREAM, TPC-C/H, random access
- **System performance metric**: *Weighted speedup*

# Comparison Points

---

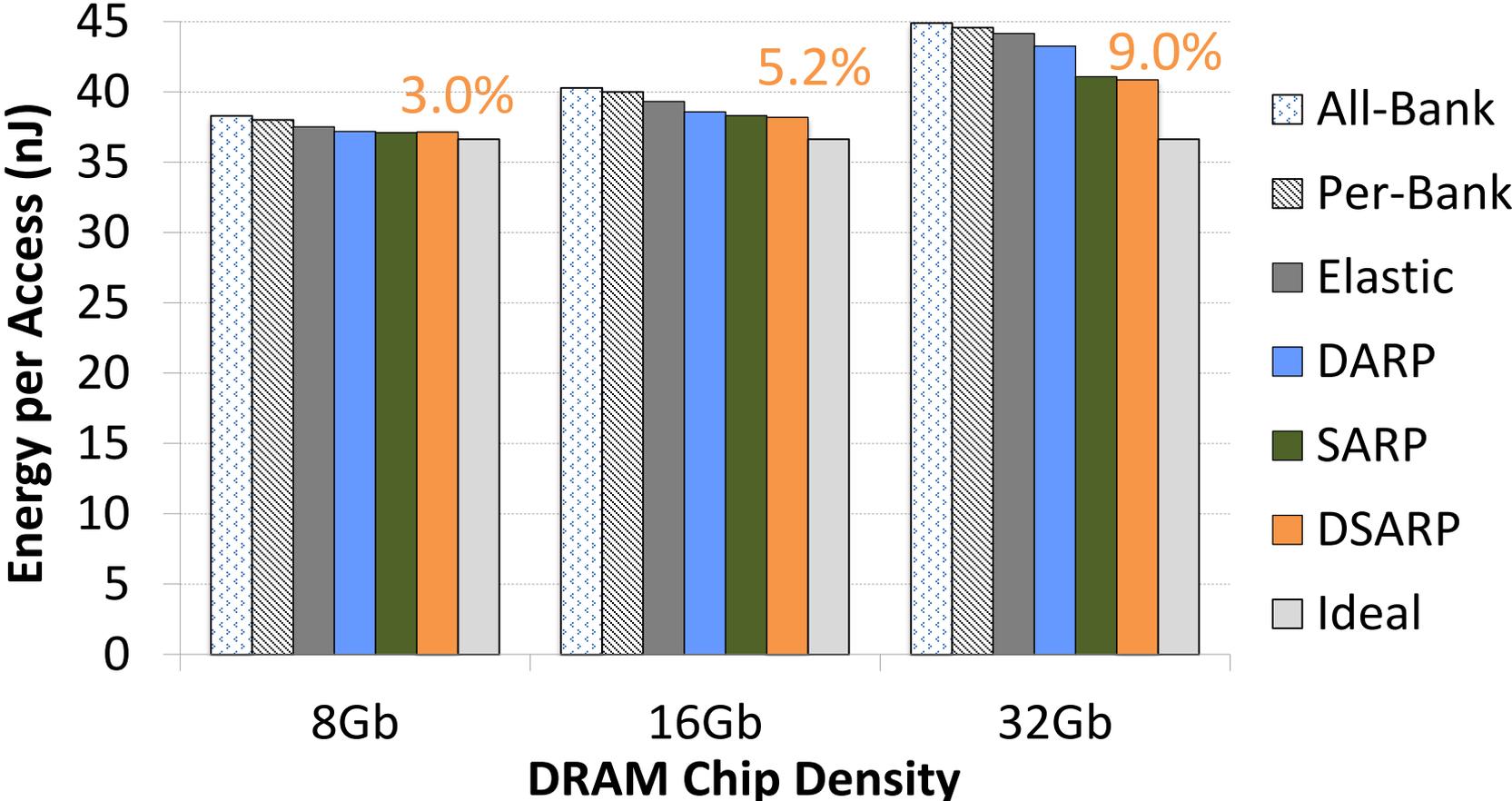
- **All-bank refresh** [DDR3, LPDDR3, ...]
- **Per-bank refresh** [LPDDR3]
- **Elastic refresh** [Stuecheli et al., MICRO '10]:
  - Postpones refreshes by a time delay based on the predicted rank idle time to avoid interference on memory requests
  - Proposed to schedule all-bank refreshes without exploiting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

# System Performance



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

# Energy Efficiency



*Consistent reduction on energy consumption*

# More Information on Refresh-Access Parallelization

---

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

## **Reducing Performance Impact of DRAM Refresh by Parallelizing Refreshes with Accesses**

Kevin Kai-Wei Chang   Donghyuk Lee   Zeshan Chishti†

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

Carnegie Mellon University   †Intel Labs

# Tackling Refresh: Solutions

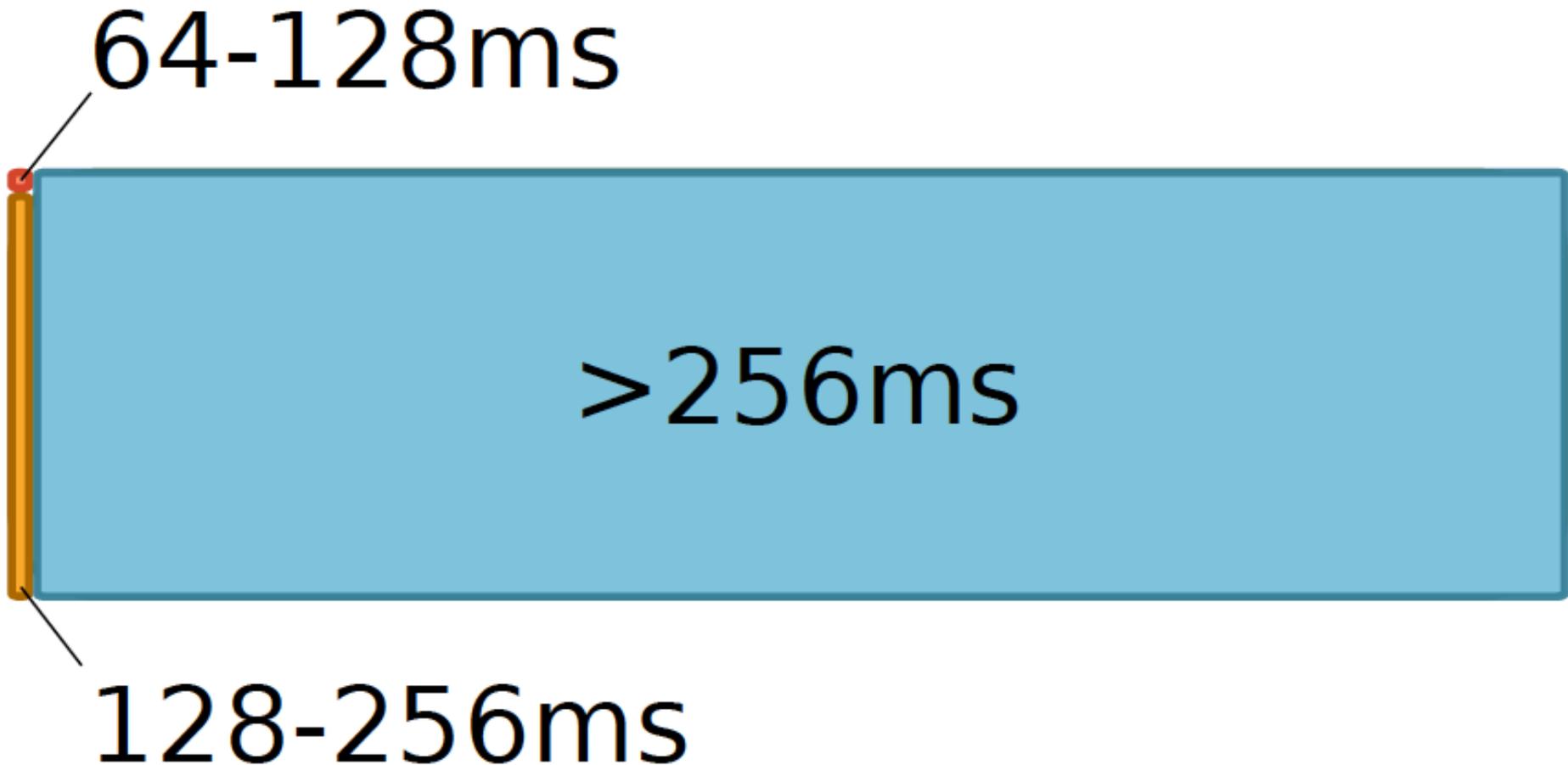
---

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

# Most Refreshes Are Unnecessary

---

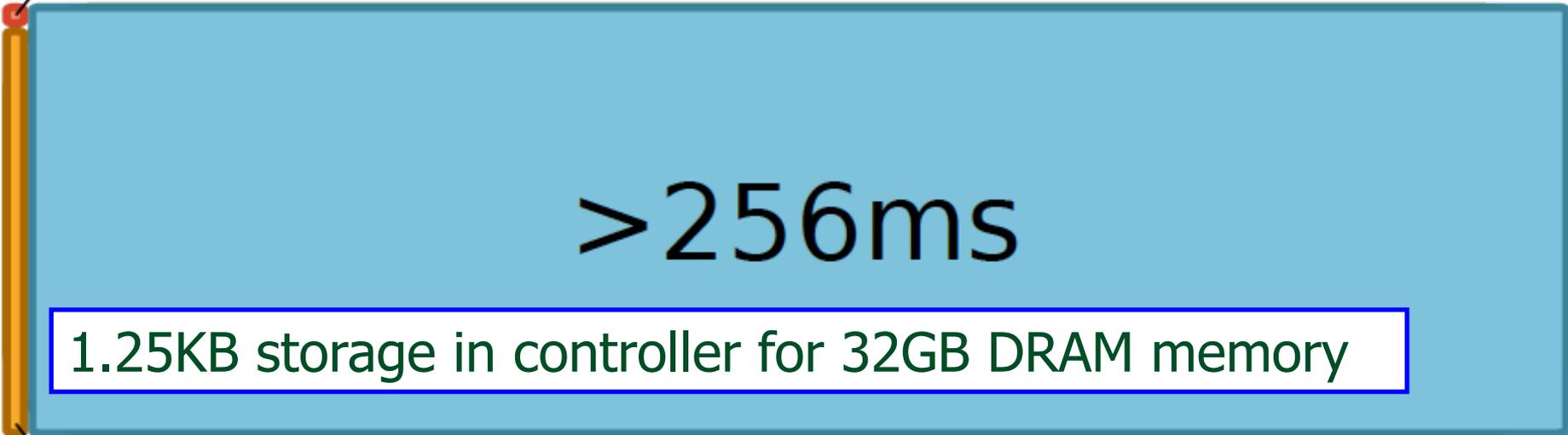
- Retention Time Profile of DRAM looks like this:



# RAIDR: Eliminating Unnecessary Refreshes

---

64-128ms



>256ms

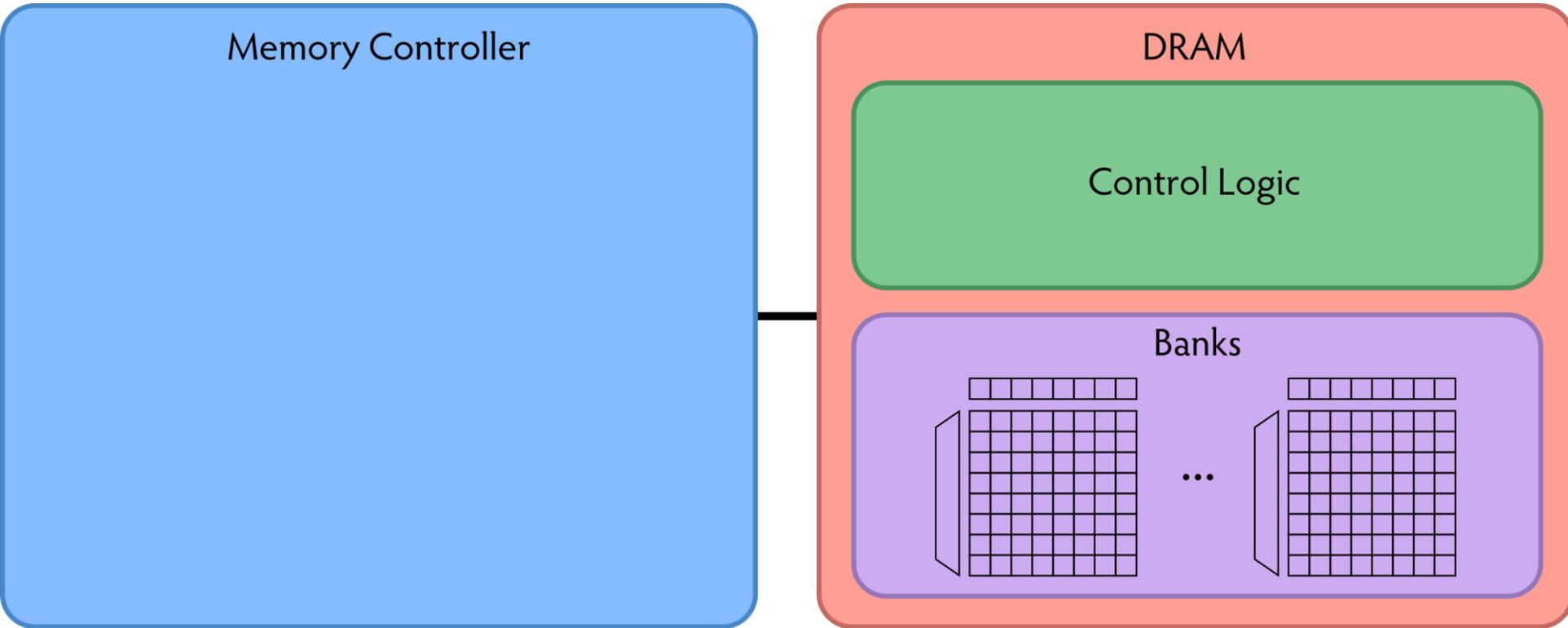
1.25KB storage in controller for 32GB DRAM memory

128-256ms

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

# RAIDR: Baseline Design

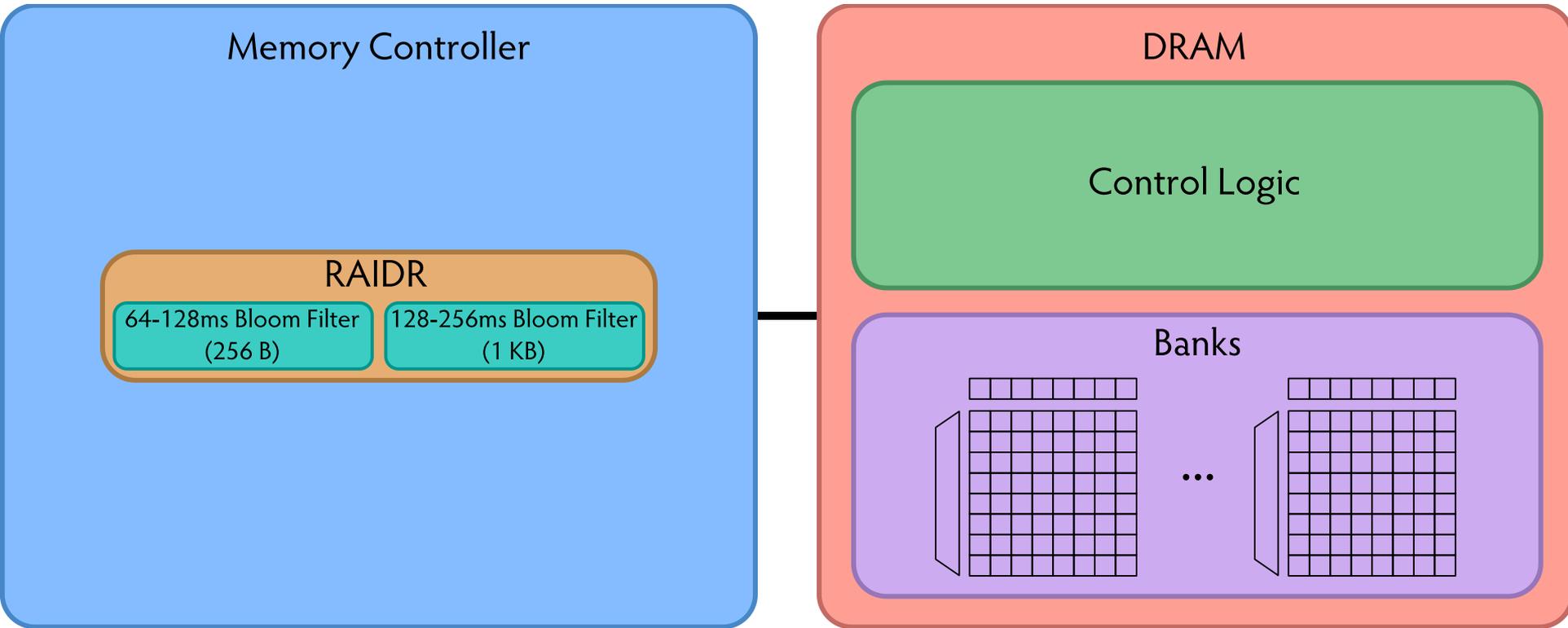
---



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

RAIDR can be implemented in either the controller or DRAM

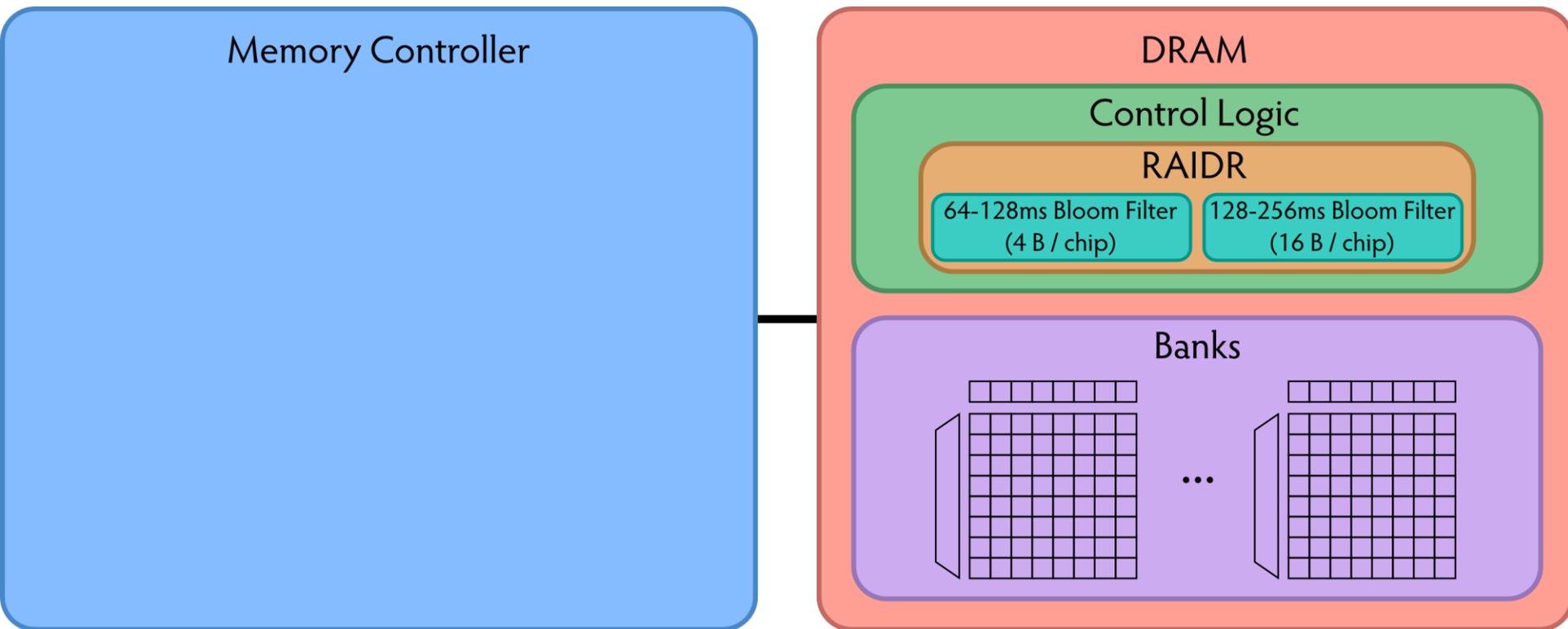
# RAIDR in Memory Controller: Option 1



Overhead of RAIDR in DRAM controller:

1.25 KB Bloom Filters, 3 counters, additional commands issued for per-row refresh (all accounted for in evaluations)

# RAIDR in DRAM Chip: Option 2



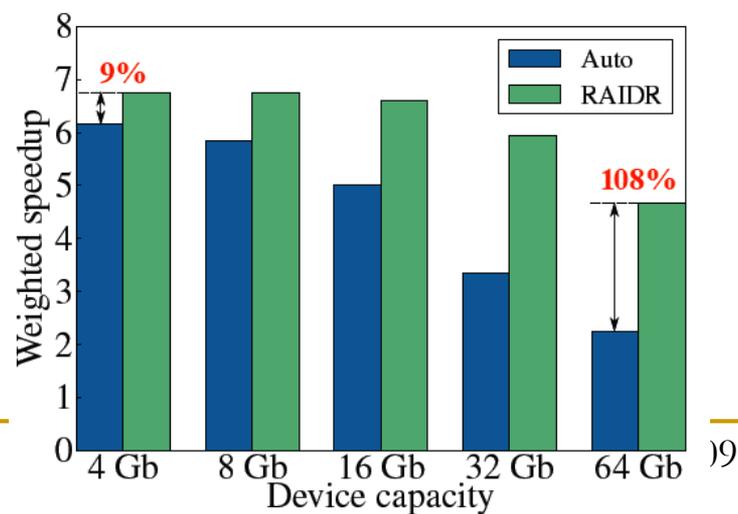
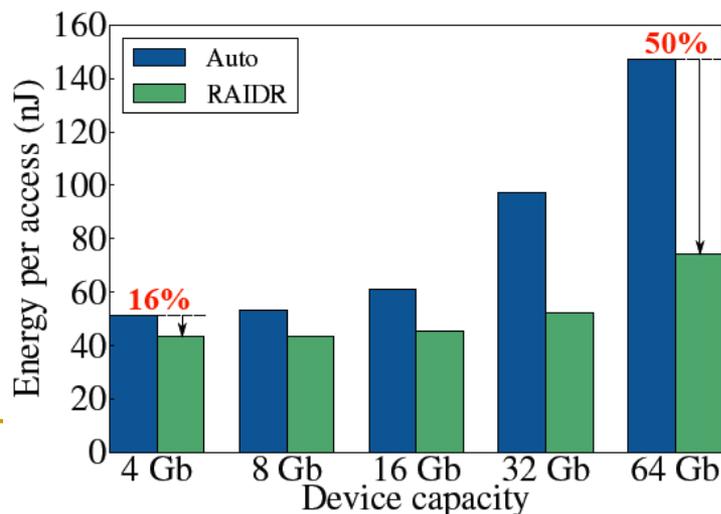
Overhead of RAIDR in DRAM chip:

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

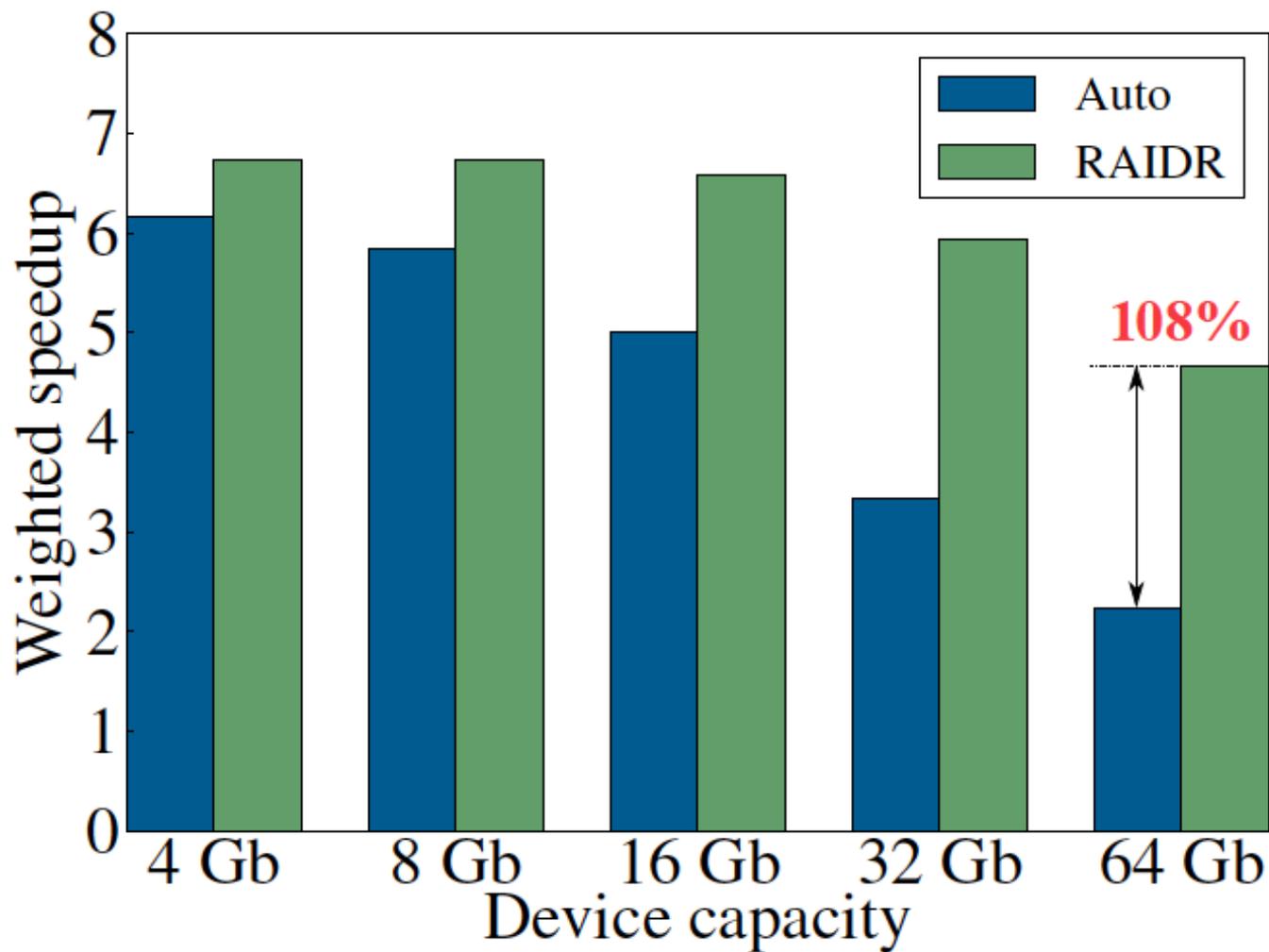
Total overhead: 1.25KB Bloom Filters, 64 counters (32 GB DRAM)

# RAIDR: Results and Takeaways

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

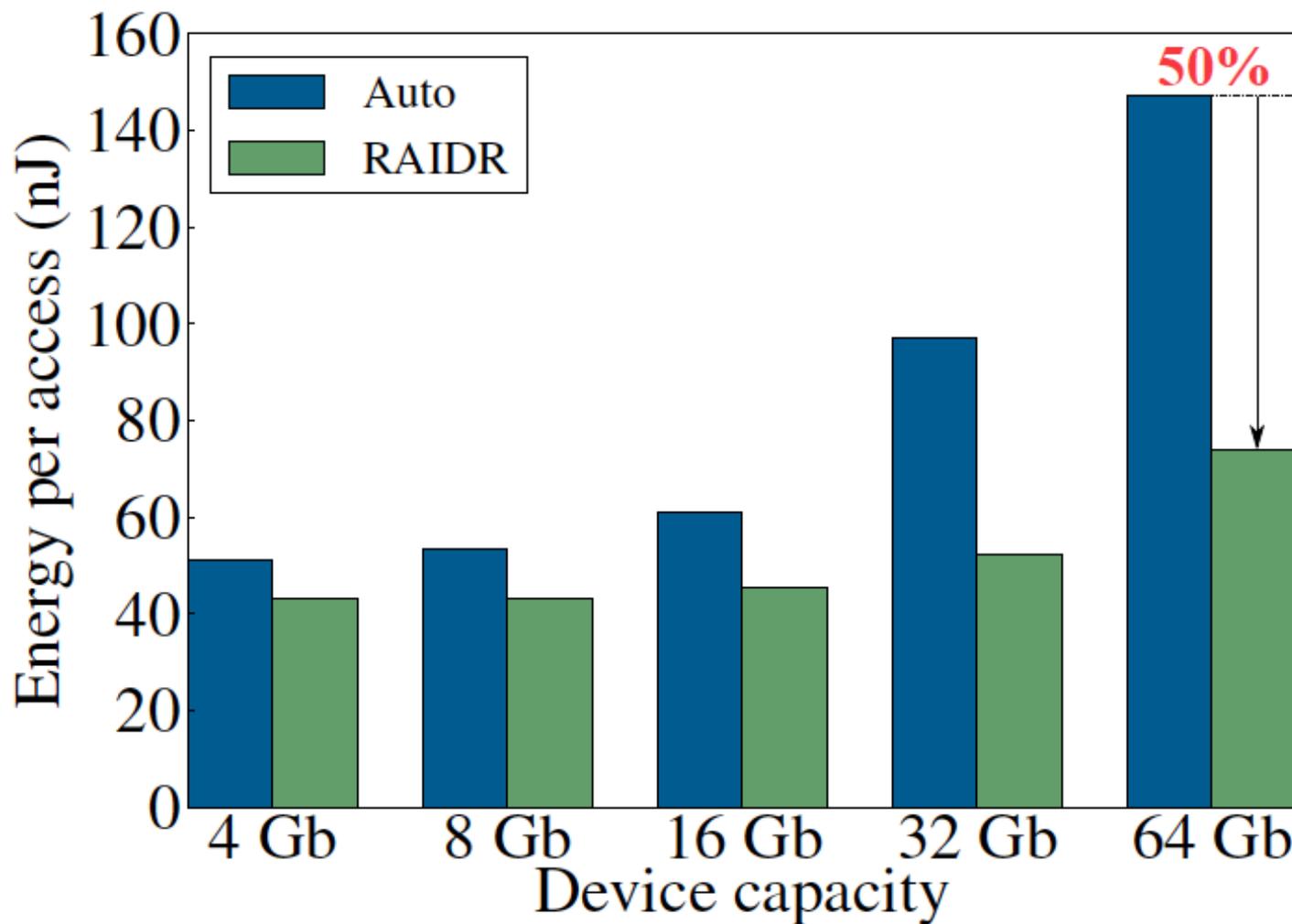


# DRAM Device Capacity Scaling: Performance



RAIDR performance benefits increase with DRAM chip capacity

# DRAM Device Capacity Scaling: Energy



RAIDR energy benefits increase with DRAM chip capacity

# RAIDR: Eliminating Unnecessary Refreshes

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

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

1. Profiling: Profile retention time of all rows

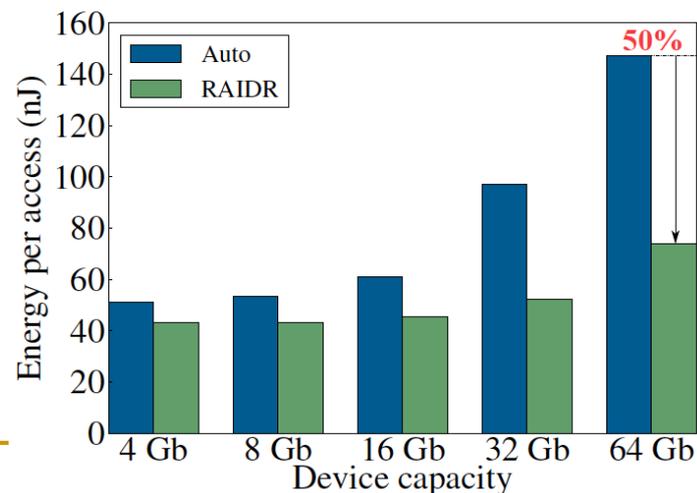
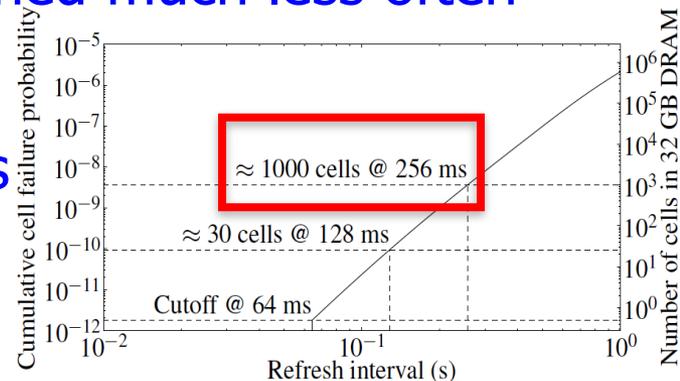
2. Binning: Store rows into bins by retention time in memory controller

*Efficient storage with Bloom Filters (only 1.25KB for 32GB memory)*

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

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

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



# More on RAIDR

---

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

## **RAIDR: Retention-Aware Intelligent DRAM Refresh**

Jamie Liu    Ben Jaiyen    Richard Veras    Onur Mutlu  
Carnegie Mellon University

# Tackling Refresh: Solutions

---

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

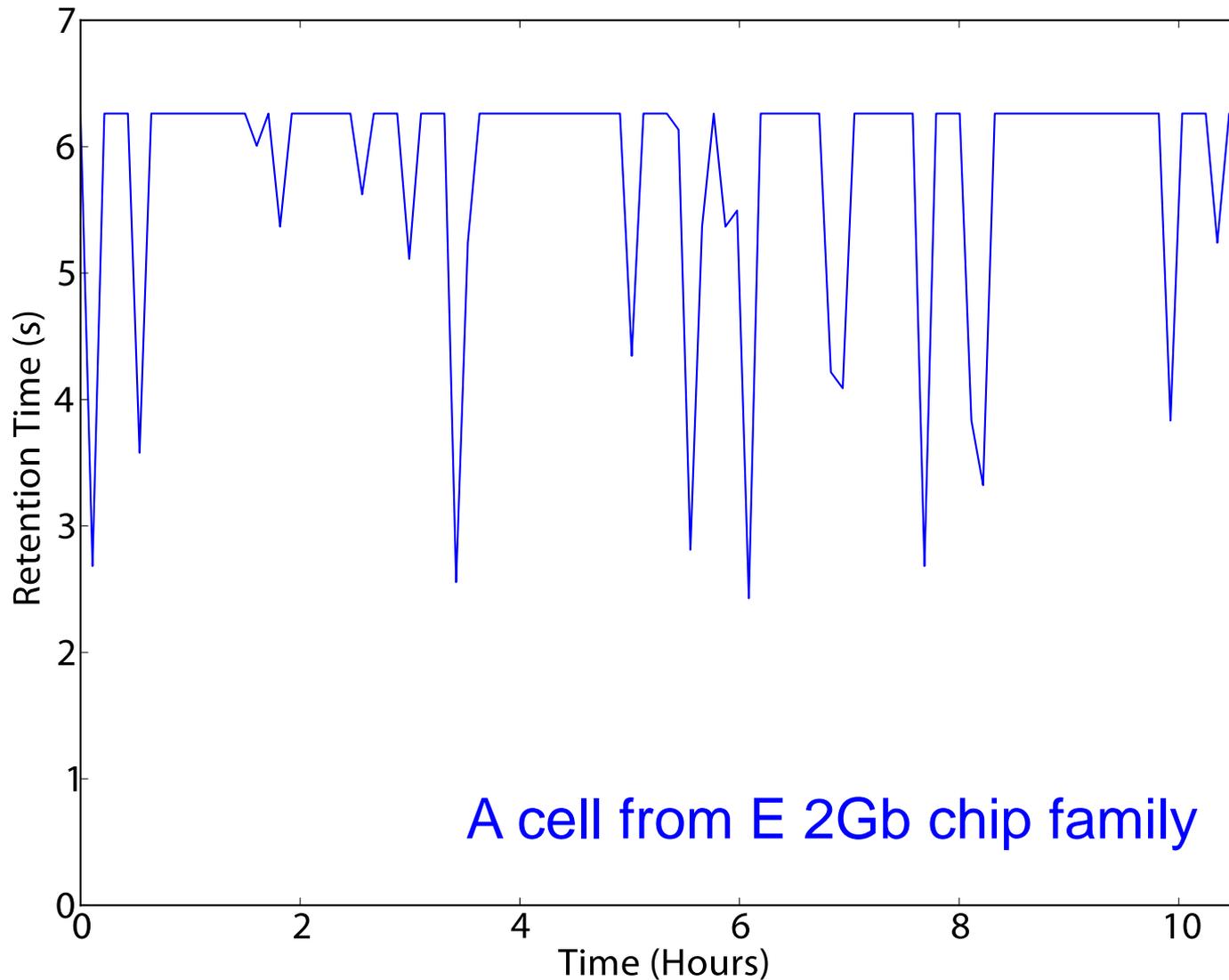
# Motivation: Understanding Retention

---

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



# An Example VRT Cell



# VRT: Implications on Profiling Mechanisms

---

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

# More on DRAM Retention Analysis

---

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

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

Jamie Liu\*

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

Ben Jaiyen\*

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

Yoongu Kim

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

Chris Wilkerson

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

Onur Mutlu

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

# Tackling Refresh: Solutions

---

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

# Towards an Online Profiling System

## *Key Observations:*

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

# Towards an Online Profiling System

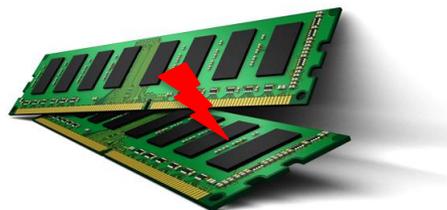
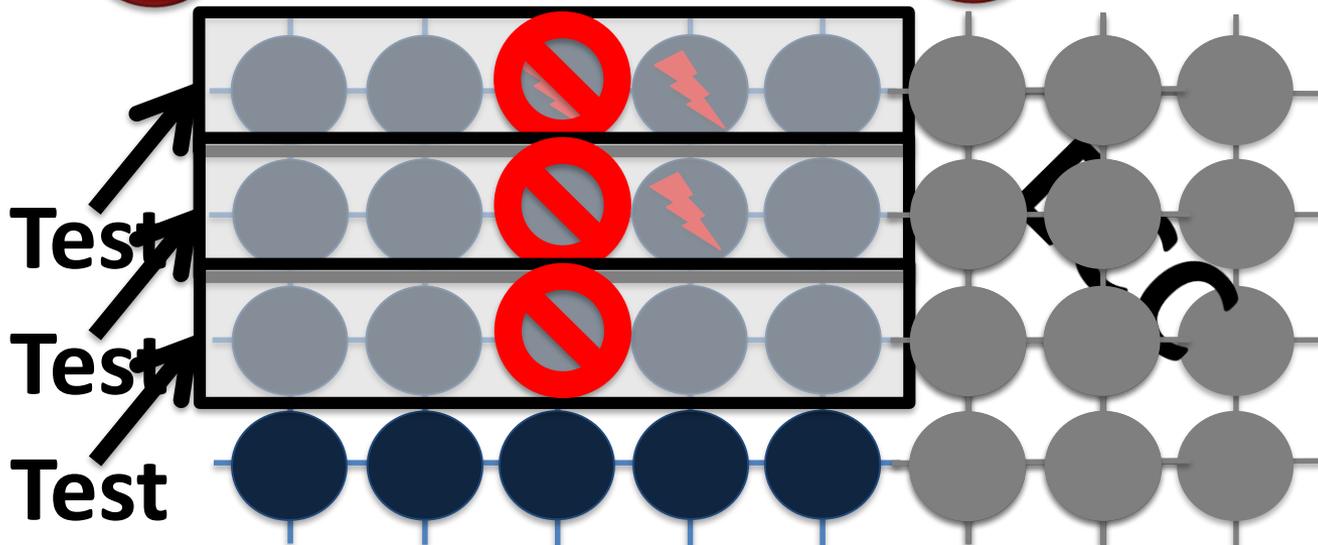
Initially Protect DRAM  
with Strong ECC

1



Periodically Test  
Parts of DRAM

2



Mitigate errors and  
reduce ECC

3

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

# More on Online Profiling of DRAM

---

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

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

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

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

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

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

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

Onur Mutlu<sup>†</sup>  
onur@cmu.edu

<sup>†</sup>Carnegie Mellon University

<sup>\*</sup>Intel Labs

# 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<sup>†</sup>      Dae-Hyun Kim<sup>†</sup>      Samira Khan<sup>‡</sup>      Prashant J. Nair<sup>†</sup>      Onur Mutlu<sup>‡</sup>  
<sup>†</sup>Georgia Institute of Technology      <sup>‡</sup>Carnegie Mellon University  
{*moin, dhkim, pnair6*}@ece.gatech.edu      {*samirakhan, onur*}@cmu.edu

# AVATAR

**Insight:** Avoid retention failures → Upgrade row on ECC error

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

Scrub  
(15 min)



DRAM Rows

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

Ref. Rate Table

0
0
1
0
0
0
0
1
1

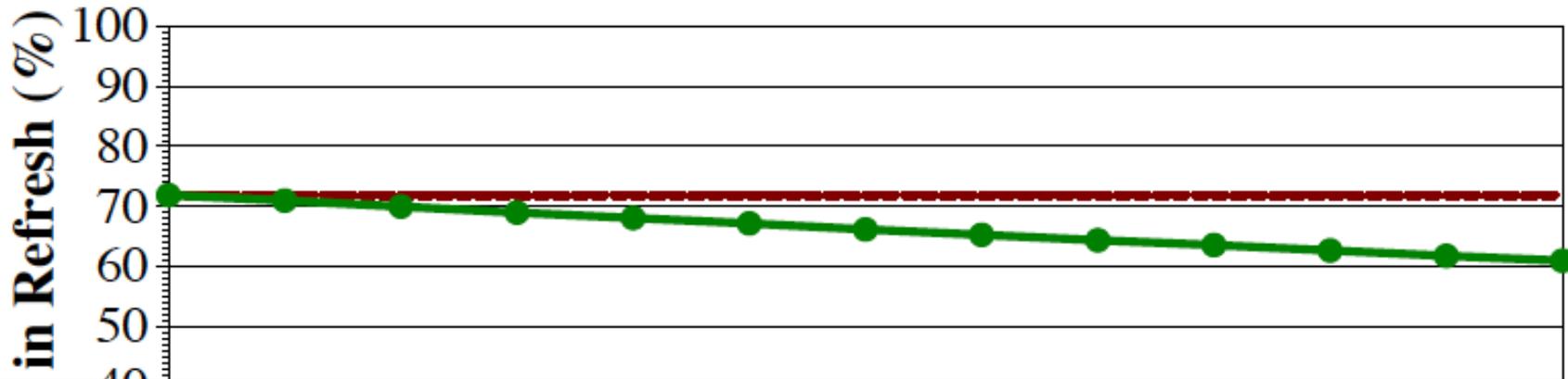
Weak Cell

RETENTION  
PROFILING

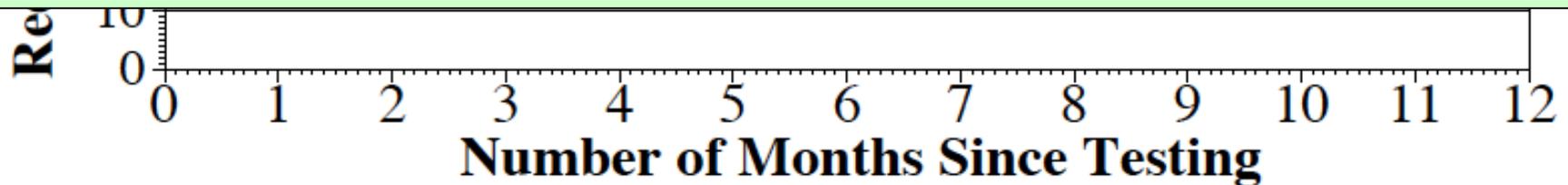
Row protected from  
future  
retention failures

**AVATAR mitigates VRT by increasing refresh rate on error**

# RESULTS: REFRESH SAVINGS

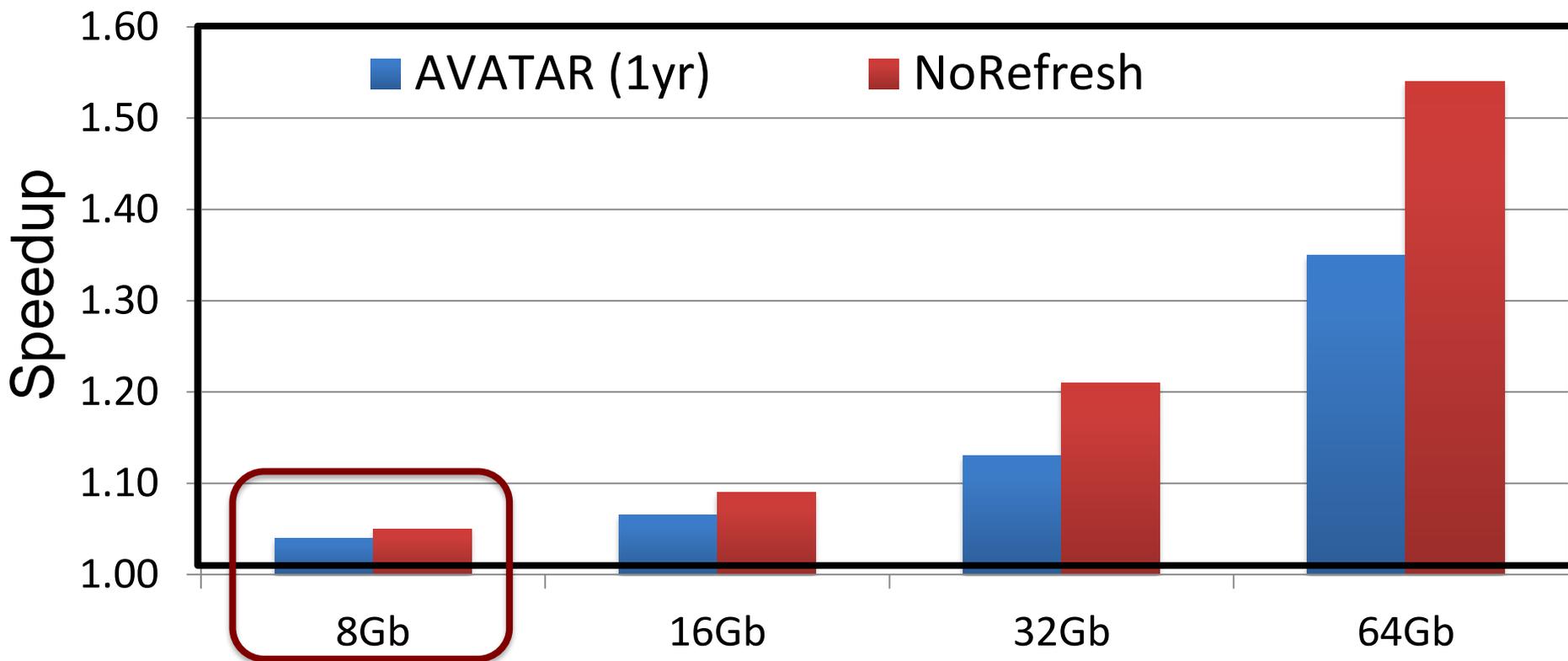


**Retention Testing Once a Year can revert refresh saving from 60% to 70%**



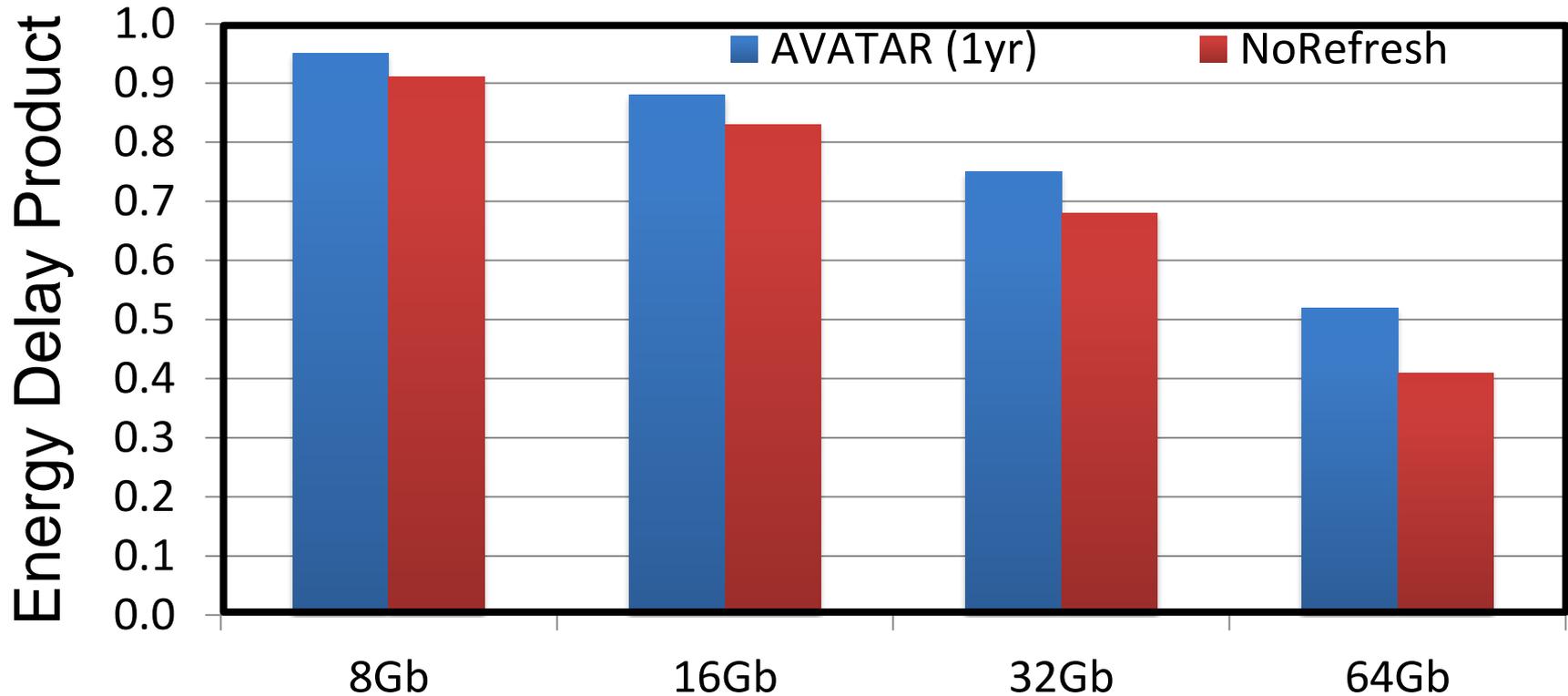
**AVATAR reduces refresh by 60%-70%, similar to multi rate refresh but with VRT tolerance**

# SPEEDUP



**AVATAR gets 2/3<sup>rd</sup> 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<sup>†</sup>      Dae-Hyun Kim<sup>†</sup>      Samira Khan<sup>‡</sup>      Prashant J. Nair<sup>†</sup>      Onur Mutlu<sup>‡</sup>  
<sup>†</sup>Georgia Institute of Technology      <sup>‡</sup>Carnegie Mellon University  
{*moin, dhkim, pnair6*}@ece.gatech.edu      {*samirakhan, onur*}@cmu.edu

# DRAM Refresh: Summary and Conclusions

---

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

# Other Backup Slides

# Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

**Amirali Boroumand**

Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun,  
Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela,  
Allan Knies, Parthasarathy Ranganathan, Onur Mutlu

**SAFARI**

**Carnegie Mellon**

**Google**



SEOUL  
NATIONAL  
UNIVERSITY

**ETH** zürich

# Consumer Devices



**Consumer devices are everywhere!**

**Energy consumption is  
a first-class concern in consumer devices**



# Popular Google Consumer Workloads



## Chrome

Google's web browser



## TensorFlow Mobile

Google's machine learning framework

# VP9



## Video Playback

Google's **video codec**

# VP9

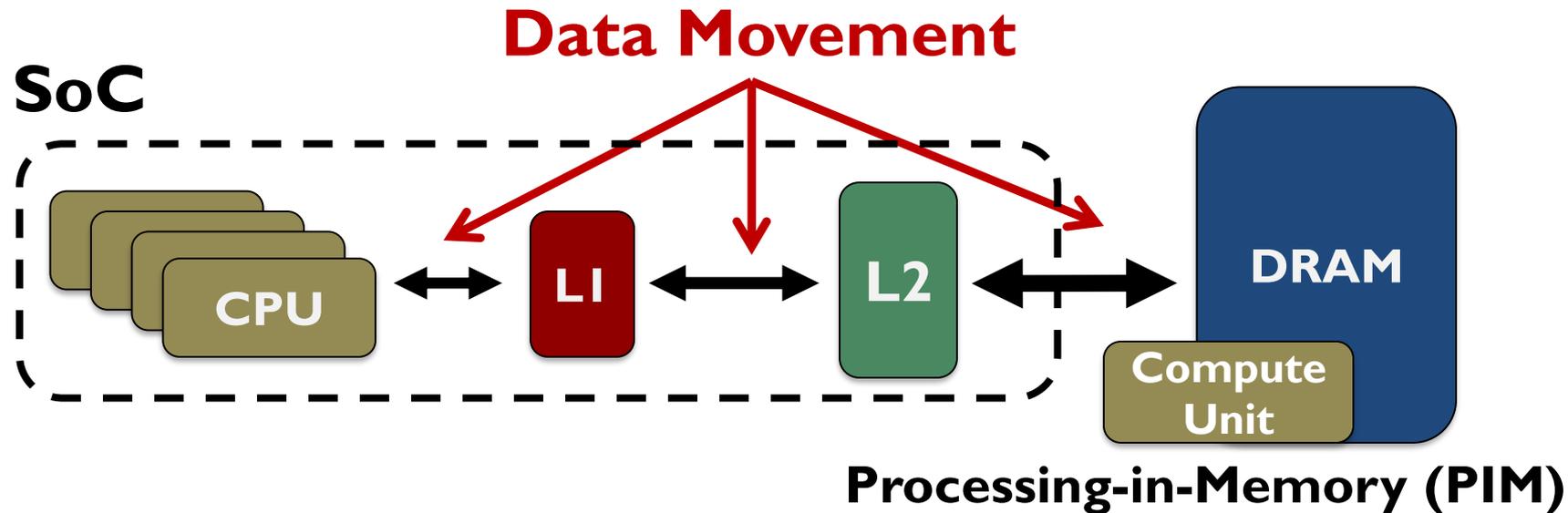


## Video Capture

Google's **video codec**

# Energy Cost of Data Movement

1<sup>st</sup> key observation: **62.7%** of the total system energy is spent on **data movement**



Potential solution: move computation **close to data**

Challenge: limited area and energy budget

# Using PIM to Reduce Data Movement

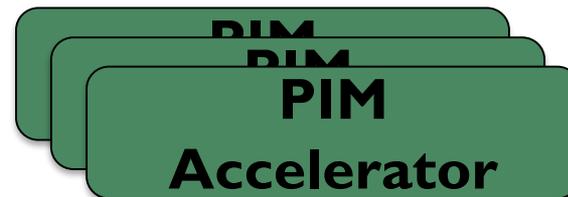
**2<sup>nd</sup> key observation:** a significant fraction of **data movement** often comes from **simple functions**

We can design lightweight logic to implement these simple functions in **memory**

Small embedded  
low-power core



Small fixed-function  
accelerators



Offloading to PIM logic reduces energy by 55.4% and improves performance by 54.2% on average

# Goals

- 1** Understand the **data movement** related bottlenecks in **modern consumer workloads**
- 2** Analyze opportunities to **mitigate data movement** by using **processing-in-memory (PIM)**
- 3** Design **PIM logic** that can **maximize energy efficiency** given **the limited area and energy budget** in consumer devices

# Outline

- Introduction
- **Background**
- Analysis Methodology
- Workload Analysis
- Evaluation
- Conclusion

# Potential Solution to Address Data Movement

- **Processing-in-Memory (PIM)**

- A potential solution to **reduce data movement**

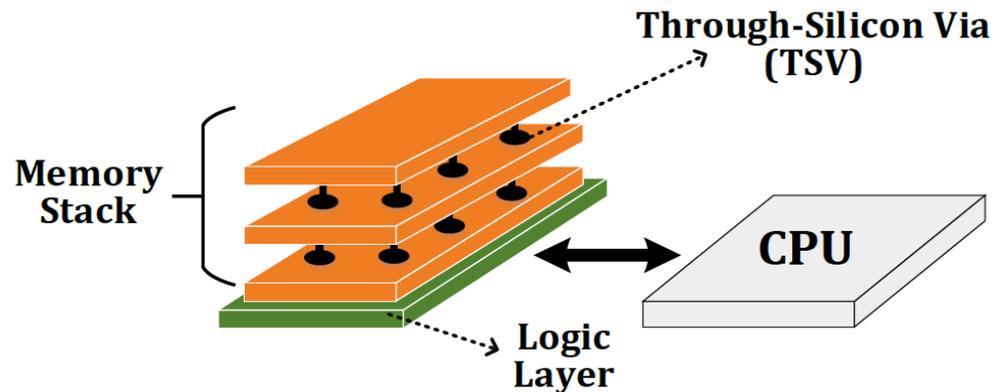
- **Idea:** move computation close to data

- ✓ Reduces data movement

- ✓ Exploits large in-memory bandwidth

- ✓ Exploits shorter access latency to memory

- **Enabled by recent advances in 3D-stacked memory**



# Outline

- Introduction
- Background
- **Analysis Methodology**
- Workload Analysis
- Evaluation
- Conclusion

# Workload Analysis Methodology

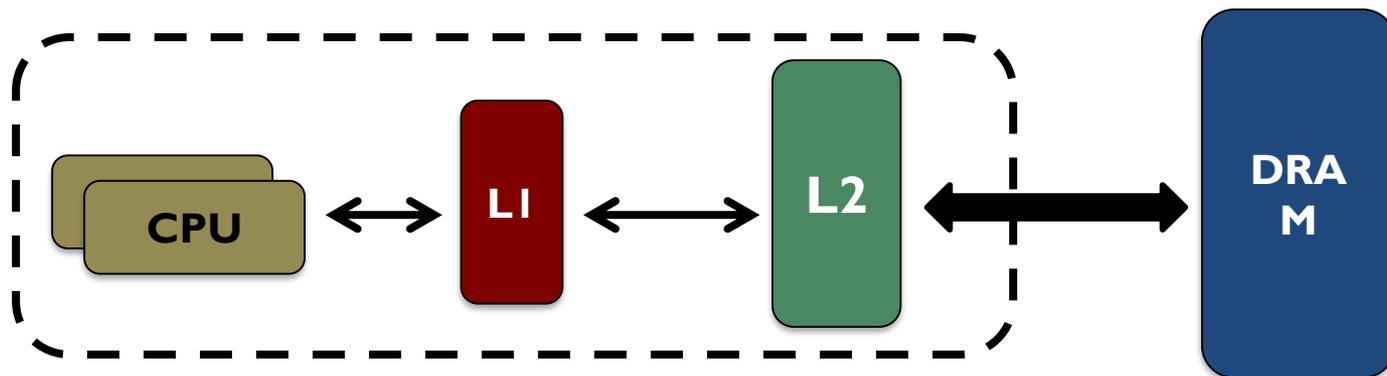
- **Workload Characterization**



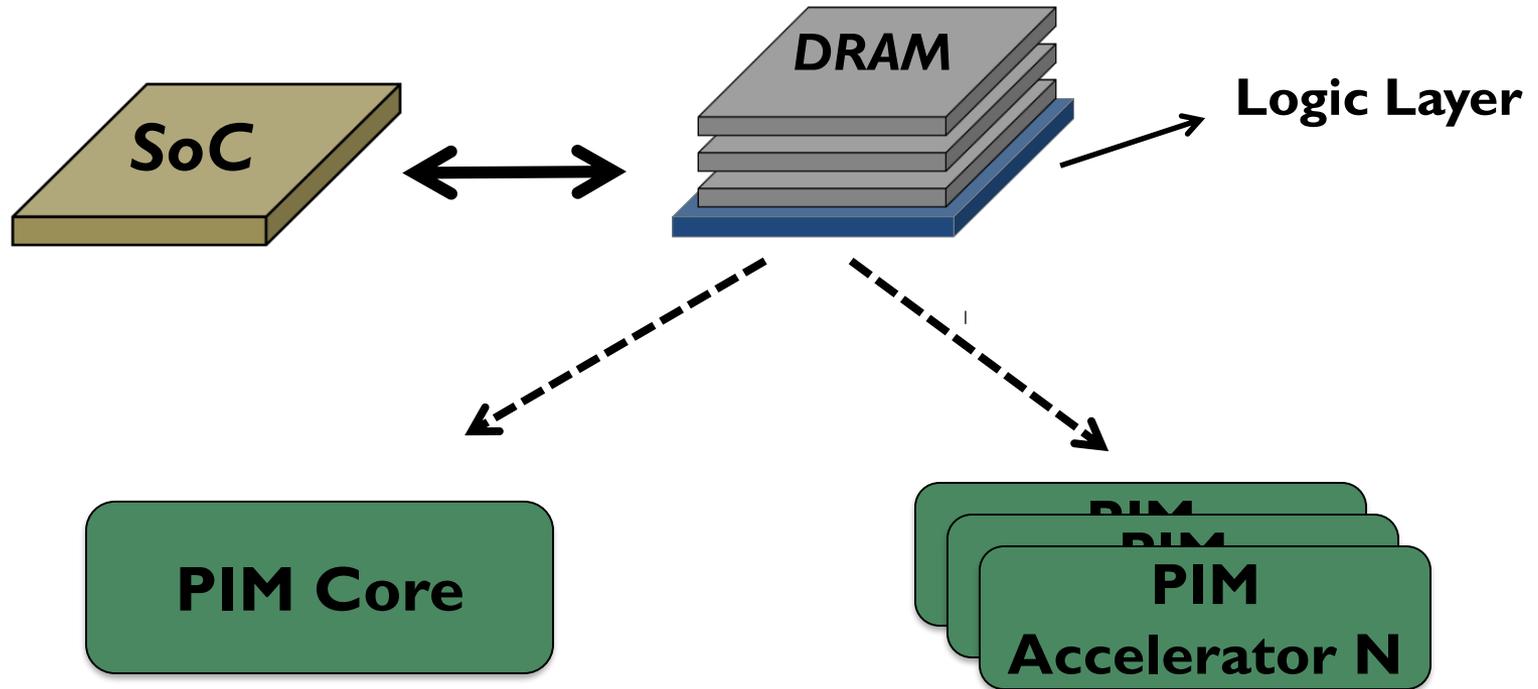
- Chromebook with an Intel Celeron SoC and 2GB of DRAM
- Extensively use performance counters within SoC

- **Energy Model**

- Sum of the energy consumption within the **CPU**, **all caches, off-chip interconnects**, and **DRAM**



# PIM Logic Implementation



**Customized embedded  
general-purpose core**

No aggressive ILP techniques  
256-bit SIMD unit

**Small fixed-function  
accelerators**

Multiple copies of customized  
in-memory logic unit

# Workload Analysis



**Chrome**

Google's web browser



**TensorFlow**

Google's machine learning framework

**VP9**



**Video Playback**

Google's **video codec**

**VP9**



**Video Capture**

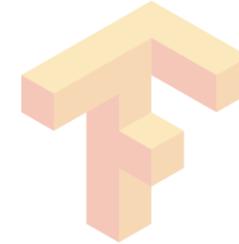
Google's **video Codec**

# Workload Analysis



**Chrome**

Google's web browser



**TensorFlow**

Google's machine learning framework

**VP9**



**Video Playback**

Google's video codec

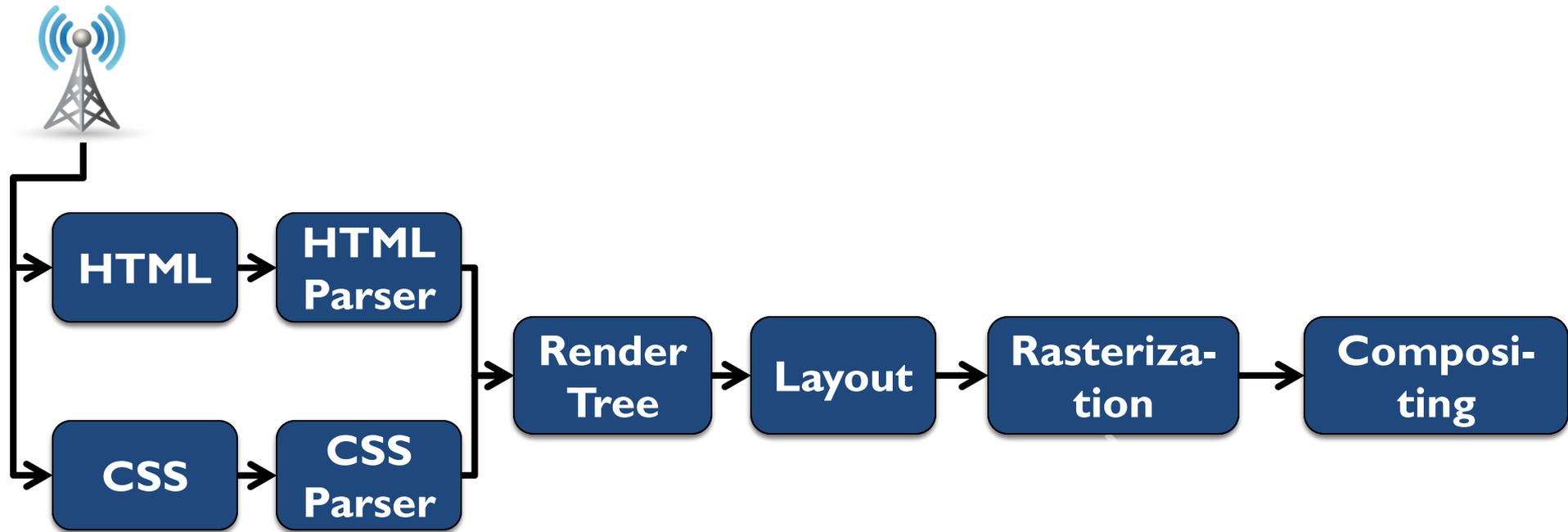
**VP9**



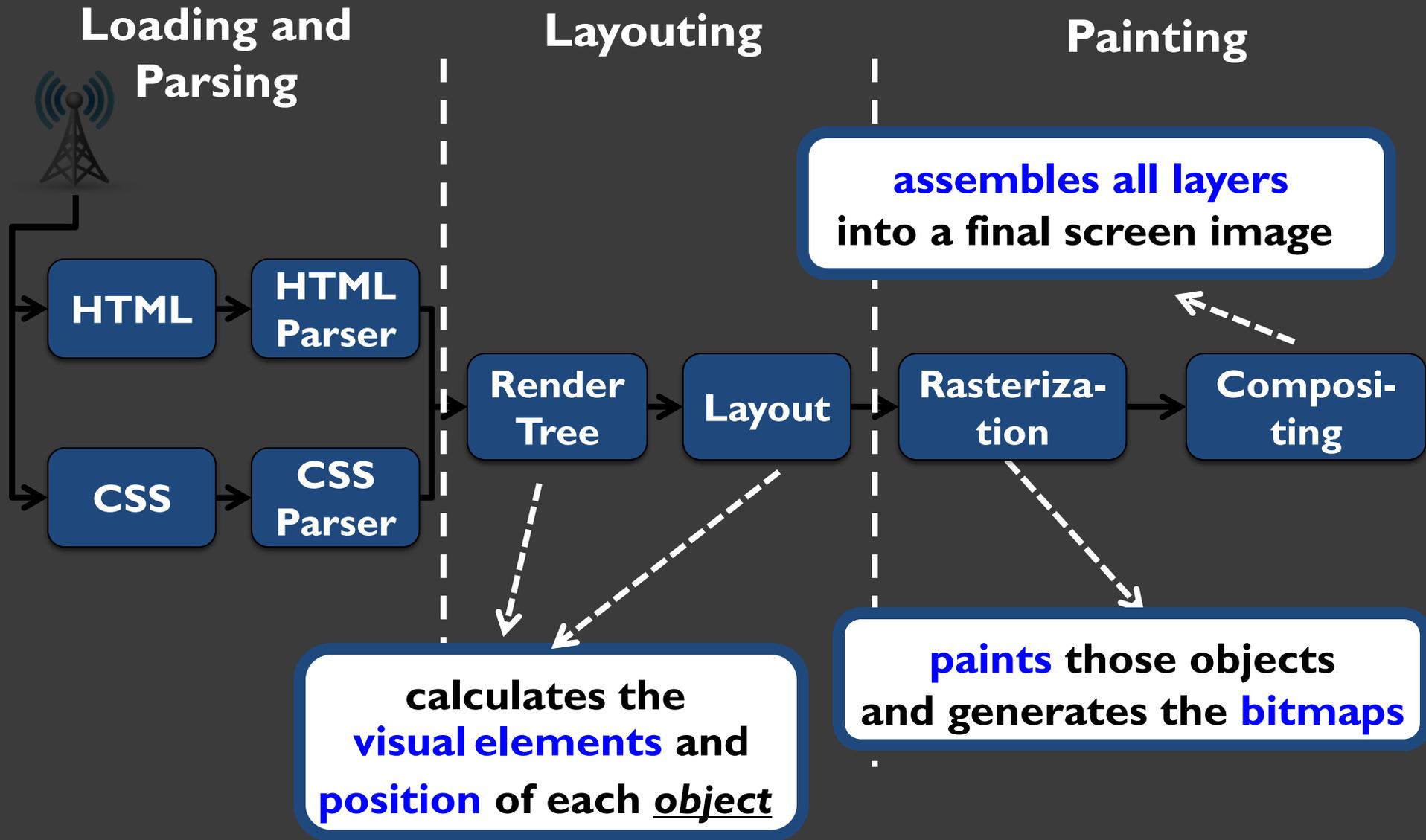
**Video Capture**

Google's video codec

# How Chrome Renders a Web Page



# How Chrome Renders a Web Page



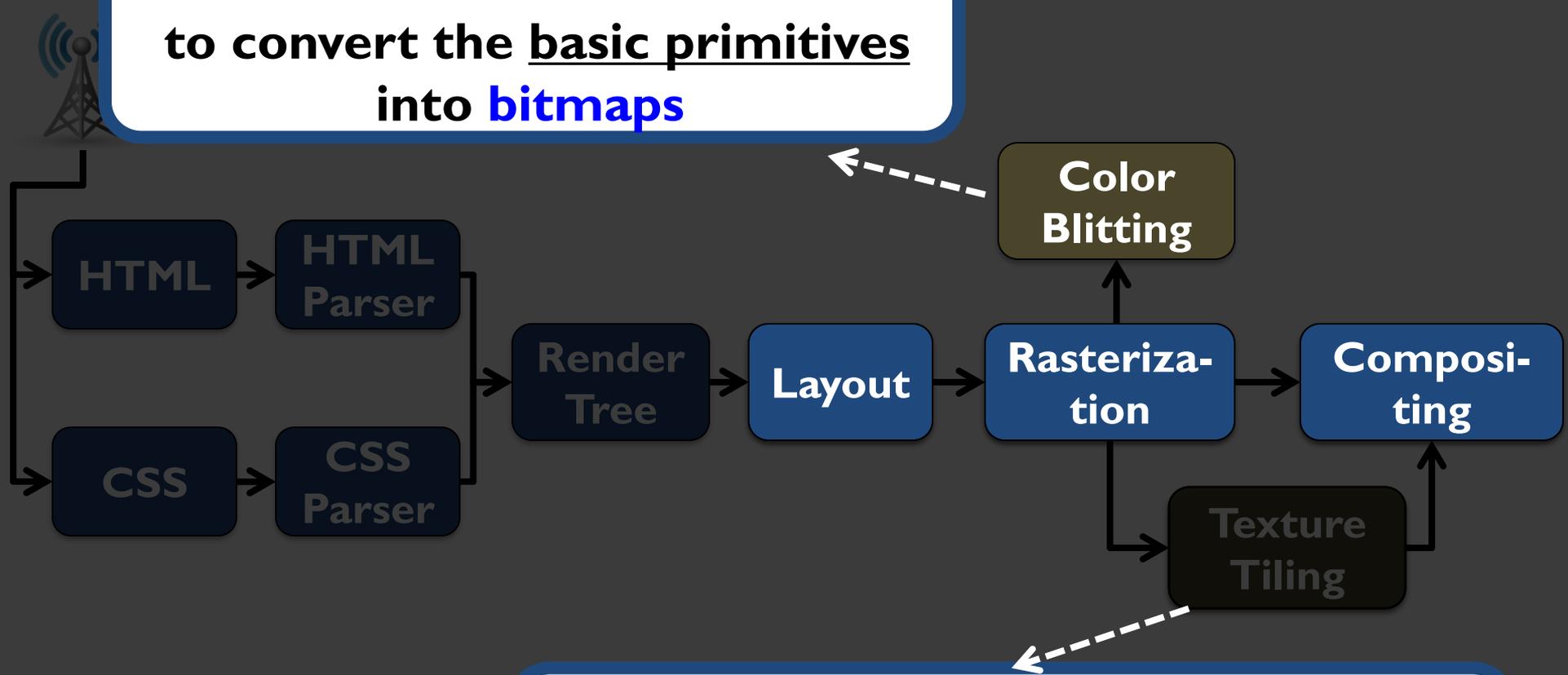
# Browser Analysis

- **To satisfy user experience, the browser must provide:**
  - Fast **loading** of webpages
  - Smooth **scrolling** of webpages
  - Quick **switching** between browser tabs
- **We focus on two important user interactions:**
  - 1) **Page Scrolling**
  - 2) **Tab Switching**
  - Both include page loading

# Scrolling

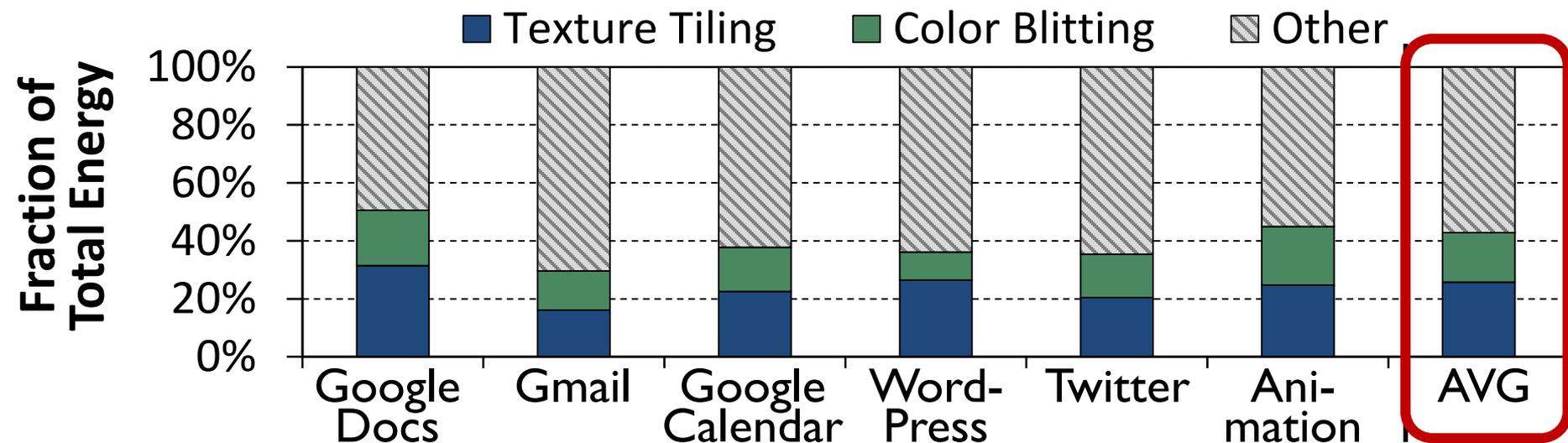
# What Does Happen During Scrolling?

rasterization uses **color blitters** to convert the basic primitives into **bitmaps**



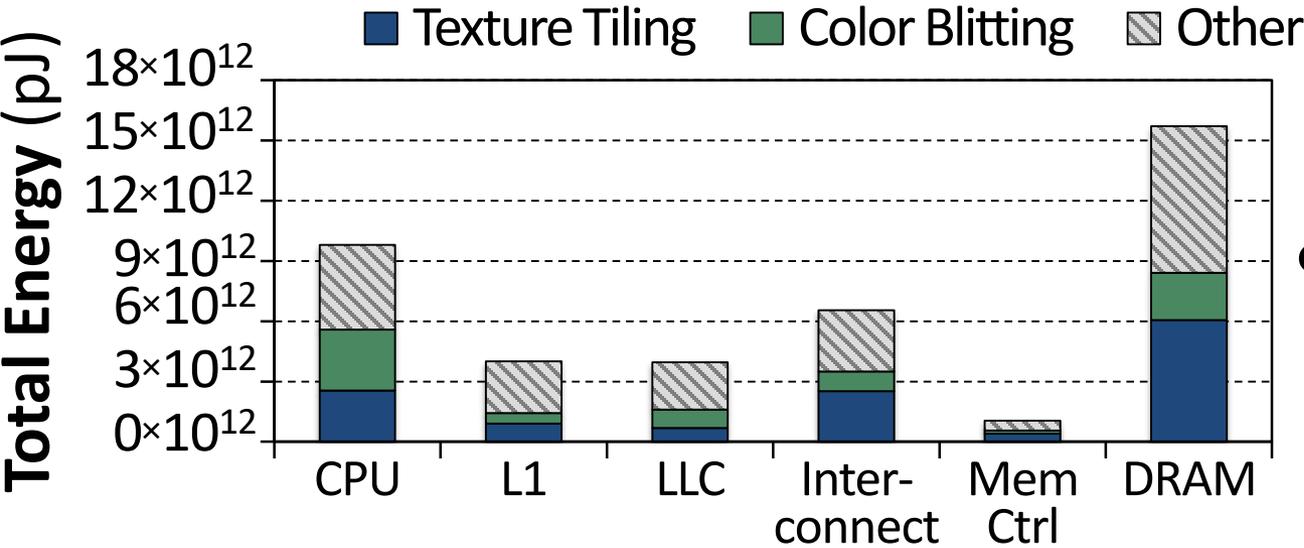
to minimize **cache misses** during **compositing**, the graphics driver reorganizes the bitmaps

# Scrolling Energy Analysis



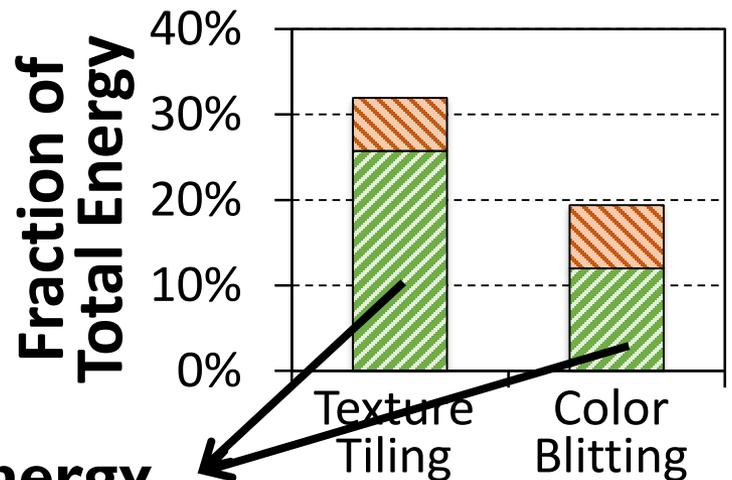
**41.9%** of page scrolling energy is spent on **texture tiling** and **color blitting**

# Scrolling a Google Docs Web Page



**77% of total energy consumption goes to data movement**

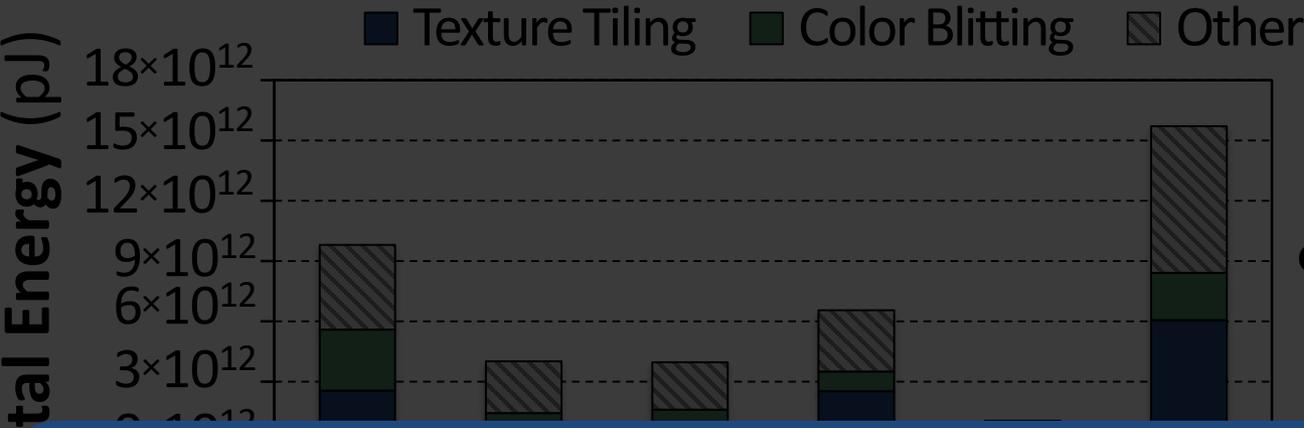
Legend for Fraction of Total Energy: Data Movement (green hatched), Compute (orange hatched)



**A significant portion of total data movement comes from texture tiling and color blitting**

**37.7% of total system energy**

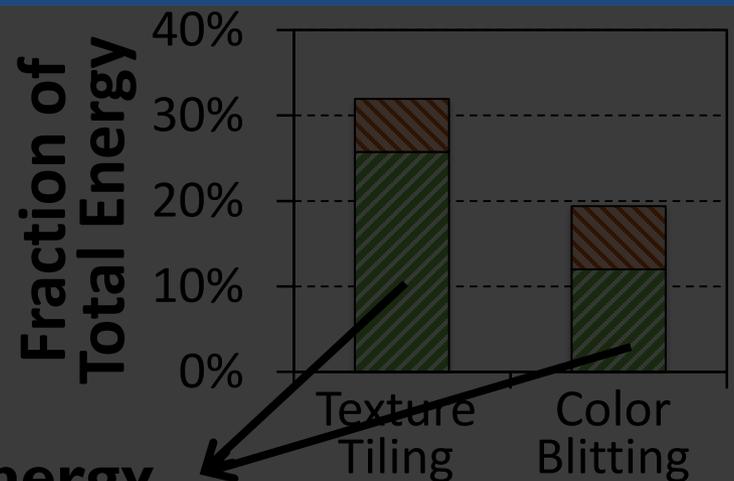
# Scrolling a Google Docs Web Page



**77%** of total energy consumption goes to data movement

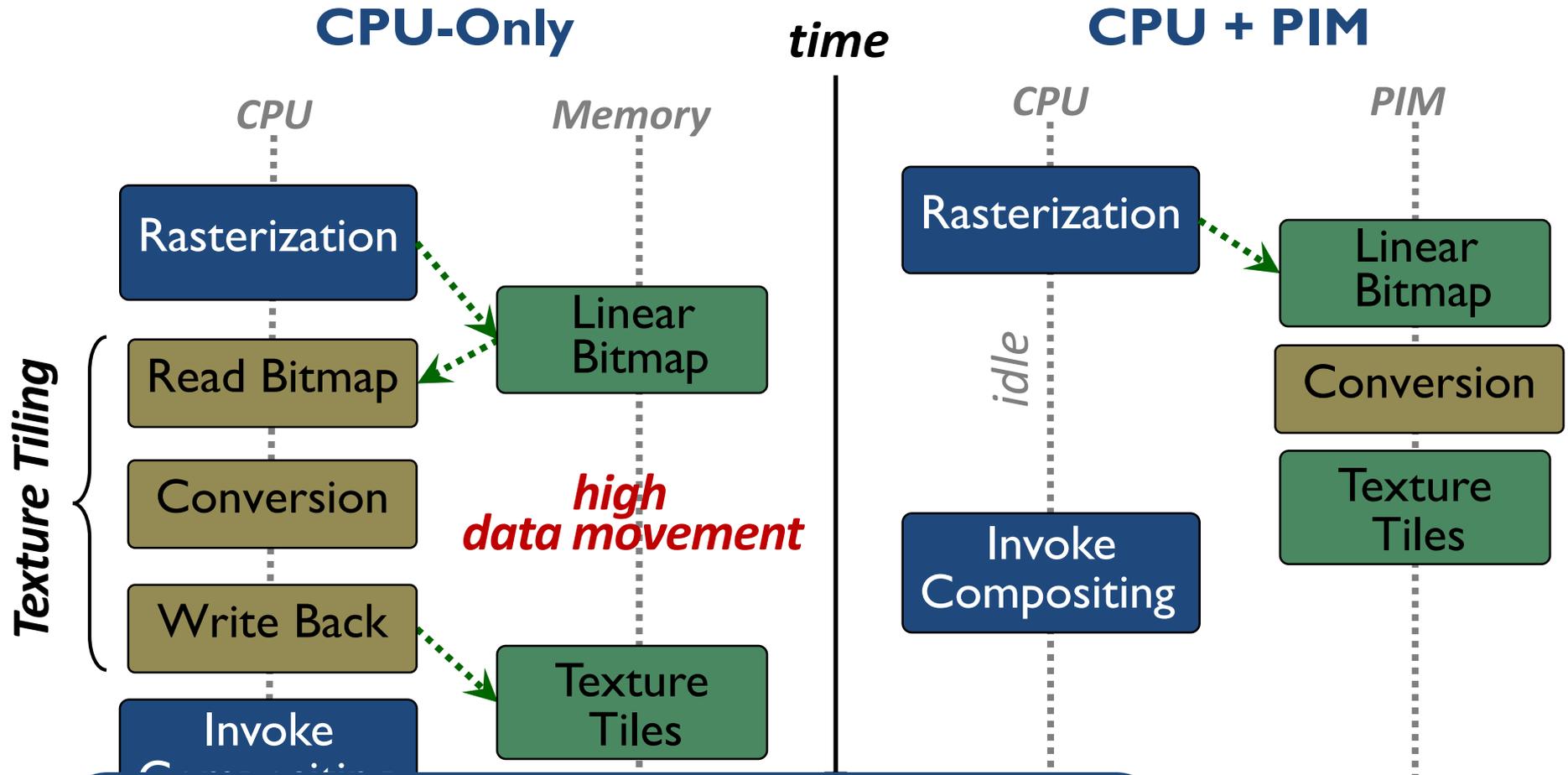
Can we use **PIM** to mitigate the **data movement cost** for **texture tiling** and **color blitting**?

A significant portion of total data movement comes from **texture tiling** and **color blitting**



**37.7%** of total system energy

# Can We Use PIM for Texture Tiling?



**Texture tiling is a good candidate for PIM execution**

# Can We Implement Texture Tiling in PIM Logic?



Requires simple primitives: **memcpy**, **bitwise operations**, and simple **arithmetic operations**

**PIM Core**

**9.4%** of the area  
available for PIM logic

**PIM  
Accelerator**

**7.1%** of the area  
available for PIM logic

**PIM core and PIM accelerator are feasible to  
implement in-memory Texture Tiling**

# Color Blitting Analysis

Generates a large amount of **data movement**

Accounts for **19.1%** of the **total system energy** during **scrolling**

**Color blitting** is a good candidate  
for **PIM execution**

Requires **low-cost** operations:

**Memset**, **simple arithmetic**, and **shift operations**

It is **feasible** to implement **color blitting**  
in **PIM core** and **PIM accelerator**

# Scrolling Wrap Up

**Texture tiling** and **color blitting** account for a significant portion (**41.9%**) of energy consumption



**37.7%** of total system energy goes to data movement generated by these functions

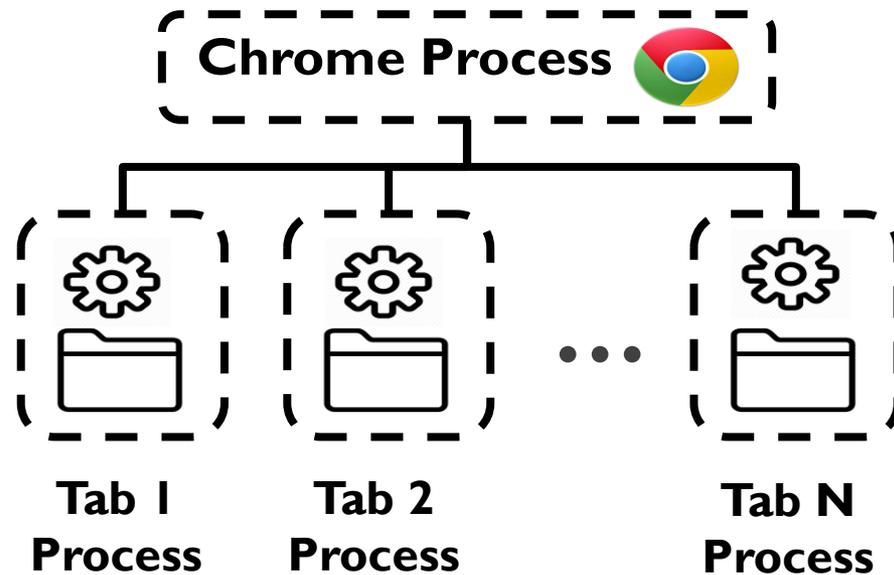
**1** Both functions can benefit significantly from PIM execution

**2** Both functions are feasible to implement as PIM logic

# Tab Switching

# What Happens During Tab Switching?

- Chrome employs a **multi-process** architecture
  - Each tab is a separate process

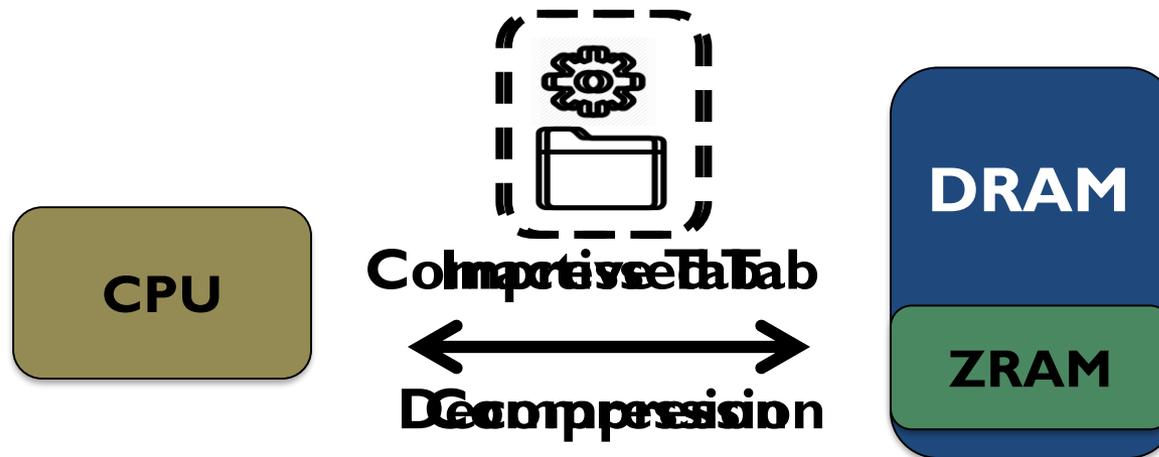


- Main operations during **tab switching**:
  - Context switch
  - Load the new page

# Memory Consumption

- **Primary concerns during tab switching:**
  - How fast a new tab **loads** and **becomes interactive**
  - **Memory consumption**

Chrome uses **compression** to reduce each tab's **memory footprint**



# Data Movement Study

- To study **data movement** during tab switching, we emulate a user switching through 50 tabs

We make two **key observations**:

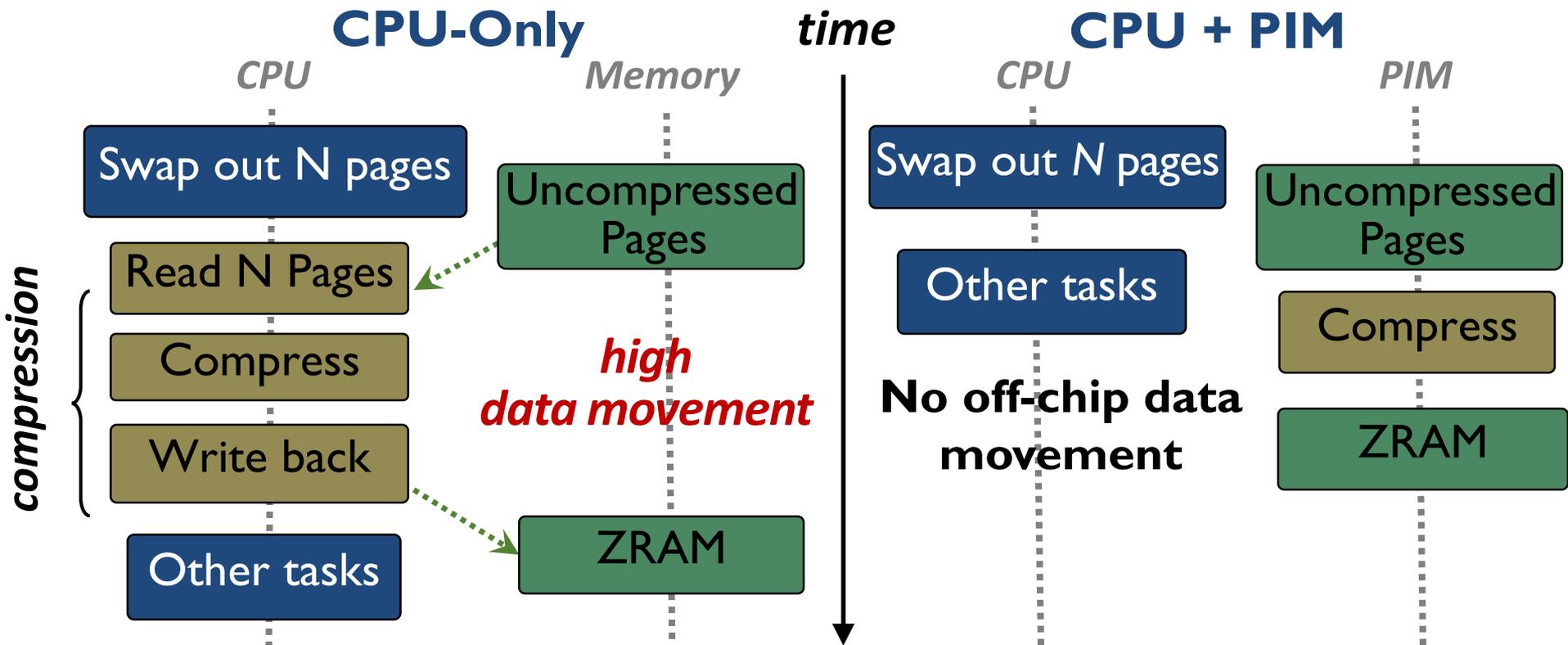
1

**Compression and decompression** contribute to **18.1%** of the total system energy

2

**19.6 GB** of data moves between **CPU** and **ZRAM**

# Can We Use PIM to Mitigate the Cost?



**PIM core and PIM accelerator are feasible to implement in-memory compression/decompression**

# Tab Switching Wrap Up

A large amount of **data movement** happens during **tab switching** as Chrome attempts to **compress** and **decompress** tabs

**Both functions can benefit from PIM execution and can be implemented as PIM logic**

# Workload Analysis



**Chrome**

Google's web browser



**TensorFlow**

Google's machine learning framework

**VP9**



**Video Playback**

Google's **video codec**

**VP9**



**Video Capture**

Google's **video codec**

# Workload Analysis



**Chrome**

Google's web browser



**TensorFlow**

Google's machine learning framework

**VP9**



**Video Playback**

Google's **video codec**

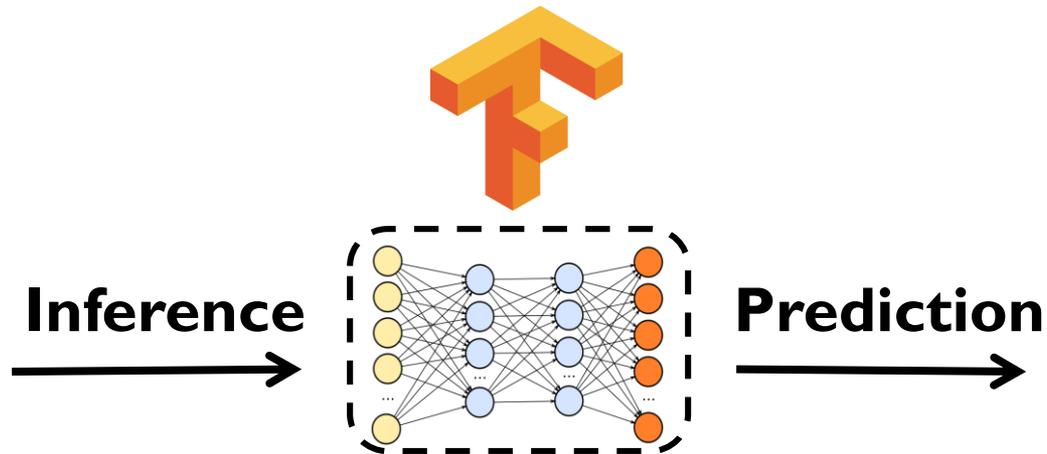
**VP9**



**Video Capture**

Google's **video codec**

# TensorFlow Mobile



**57.3%** of the inference energy is spent on data movement



**54.4%** of the **data movement** energy comes from packing/unpacking and quantization

# Packing



**Reorders** elements of matrices to minimize **cache misses** during **matrix multiplication**



Up to **40%** of the inference **energy** and **31%** of inference **execution time**



Packing's **data movement** accounts for up to **35.3%** of the inference **energy**

A simple **data reorganization** process that requires **simple arithmetic**

# Quantization



Converts 32-bit floating point to 8-bit integers to improve inference execution time and energy consumption



Up to **16.8%** of the inference **energy** and **16.1%** of inference **execution time**



Majority of **quantization** energy comes from **data movement**

A simple **data conversion** operation that requires **shift, addition, and multiplication** operations

# Quantization



Converts 32-bit floating point to 8-bit integers to improve inference execution time and energy consumption

**Based on our analysis, we conclude that:**

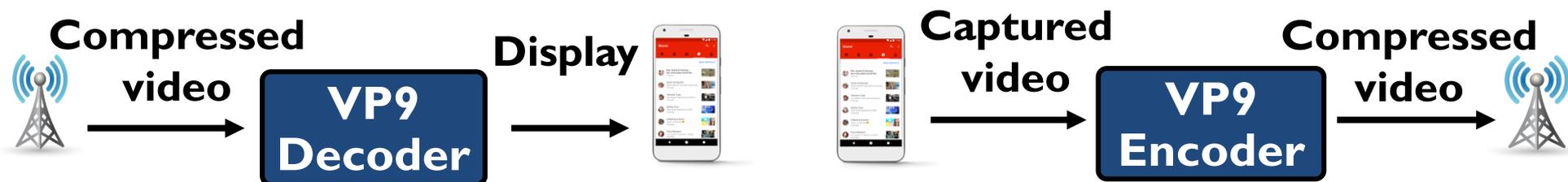
- Both functions are good candidates for **PIM execution**
- It is **feasible** to implement them in **PIM logic**

inference execution time

A simple **data conversion** operation that requires **shift, addition, and multiplication** operations

# Video Playback and Capture

## VP9



Majority of energy is spent on **data movement**

Majority of **data movement** comes from **simple functions** in **decoding** and **encoding** pipelines

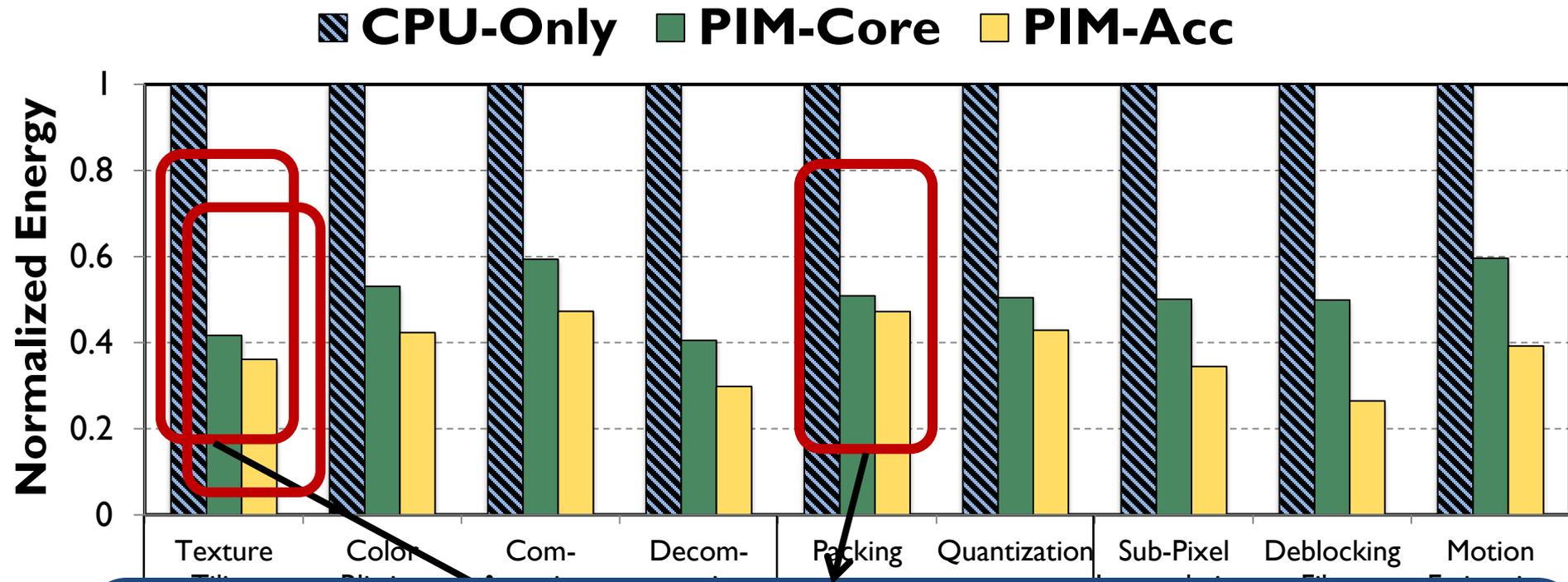
# Outline

- Introduction
- Background
- Analysis Methodology
- Workload Analysis
- **Evaluation**
- Conclusion

# Evaluation Methodology

- **System Configuration (gem5 Simulator)**
  - **SoC:** 4 OoO cores, 8-wide issue, 64 kB L1 cache, 2MB L2 cache
  - **PIM Core:** 1 core per vault, 1-wide issue, 4-wide SIMD, 32kB L1 cache
  - **3D-Stacked Memory:** 2GB cube, 16 vaults per cube
    - Internal Bandwidth: 256GB/S
    - Off-Chip Channel Bandwidth: 32 GB/s
  - **Baseline Memory:** LPDDR3, 2GB, FR-FCFS scheduler
- We study each target **in isolation** and emulate each separately and run them in our simulator

# Normalized Energy

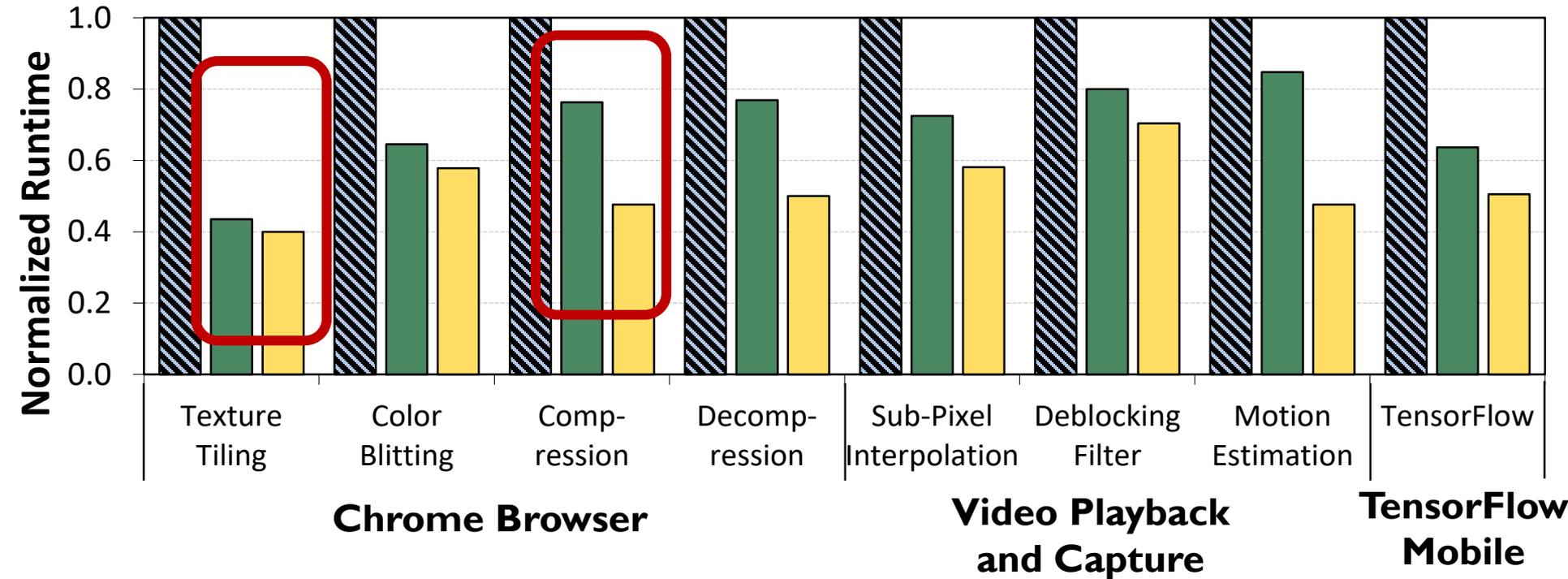


**77.7%** and **82.6%** of energy reduction for **texture tiling** and **packing** comes from eliminating **data movement**

**PIM core** and **PIM accelerator** reduces **energy consumption** on average by **49.1%** and **55.4%**

# Normalized Runtime

▨ CPU-Only █ PIM-Core █ PIM-Acc



Offloading these kernels to **PIM core** and **PIM accelerator** improves **performance** on average by **44.6%** and **54.2%**

# Conclusion

- Energy consumption is a **major challenge** in consumer devices
- We conduct an in-depth analysis of popular **Google consumer workloads**
  - **62.7%** of the total system energy is spent on **data movement**
  - Most of the **data movement** comes from simple functions that consist of simple operations
- We use **PIM** to reduce **data movement cost**
  - We design **lightweight logic** to implement **simple operations** in DRAM



- Reduces **total energy** by **55.4%** on average
- Reduces **execution time** by **54.2%** on average

# Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

**Amirali Boroumand**

Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun,  
Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela,  
Allan Knies, Parthasarathy Ranganathan, Onur Mutlu

**SAFARI**

**Carnegie Mellon**

**Google**



SEOUL  
NATIONAL  
UNIVERSITY

**ETH** zürich

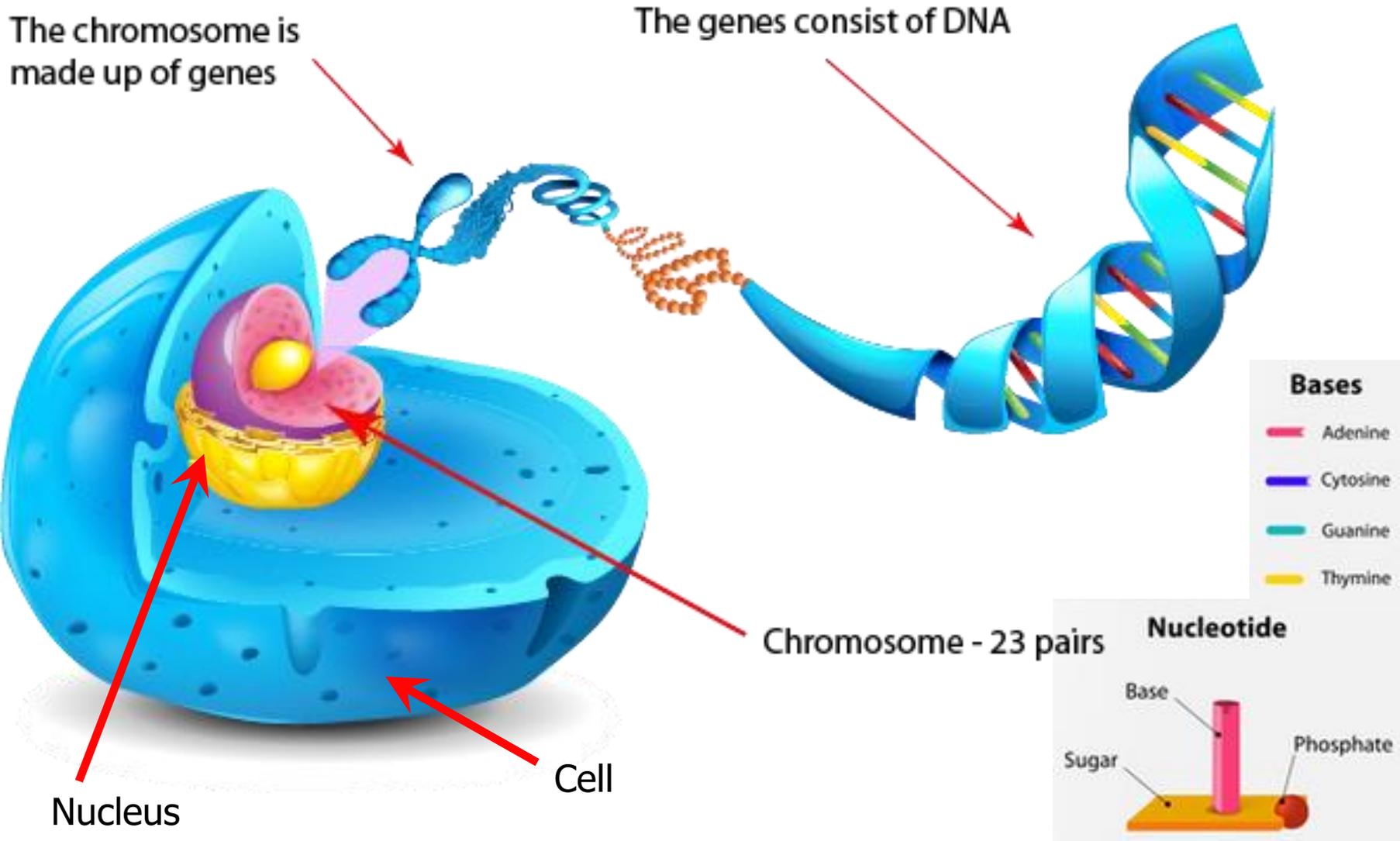
# A Motivating Detour: Genome Sequence Analysis

# Our Dream

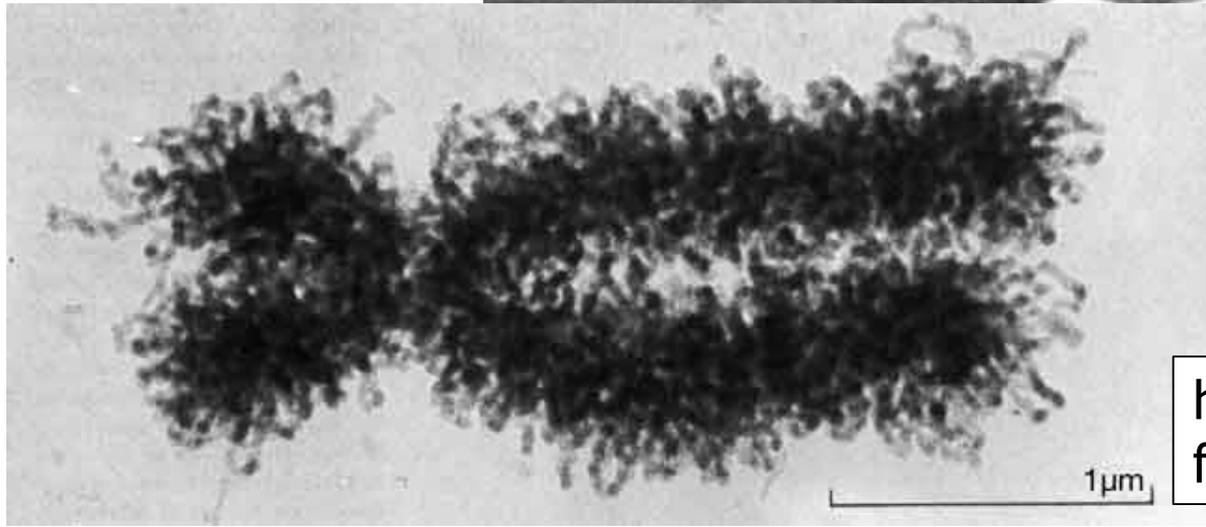
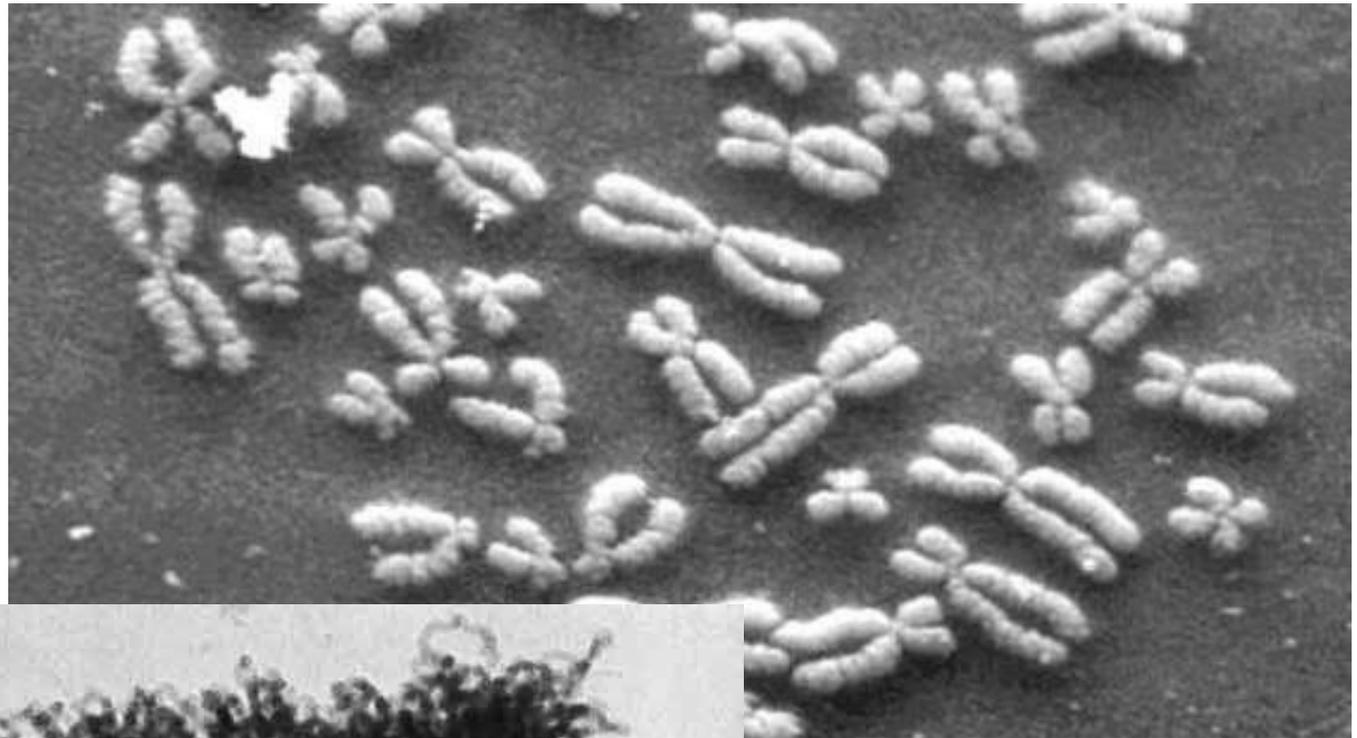
---

- An embedded device that can perform comprehensive genome analysis in real time (within a minute)
  - Which of these DNAs does this DNA segment match with?
  - What is the likely genetic disposition of this patient to this drug?
  - . . .

# What Is a Genome Made Of?



# DNA Under Electron Microscope



human chromosome #12  
from HeLa's cell

# DNA Sequencing

---

- Goal:
  - Find the complete sequence of A, C, G, T's in DNA.
- Challenge:
  - There is no machine that takes long DNA as an input, and gives the complete sequence as output
  - All sequencing machines chop DNA into pieces and identify relatively small pieces (but not how they fit together)

# Untangling Yarn Balls & DNA Sequencing

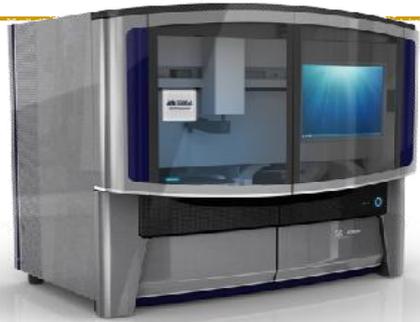
---



# Genome Sequencers



Roche/454



AB SOLiD



Illumina MiSeq



Complete Genomics



Illumina HiSeq2000



Pacific Biosciences RS



Oxford Nanopore MinION



Illumina NovaSeq 6000



Ion Torrent PGM



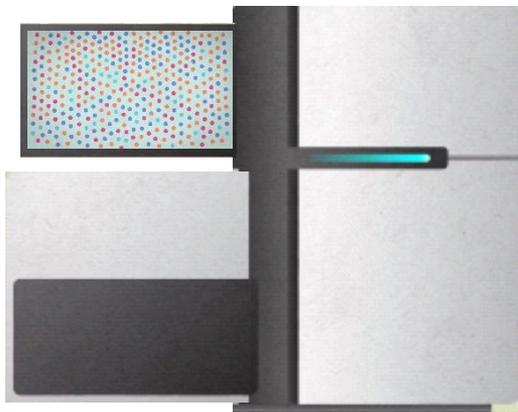
Ion Torrent Proton



Oxford Nanopore GridION

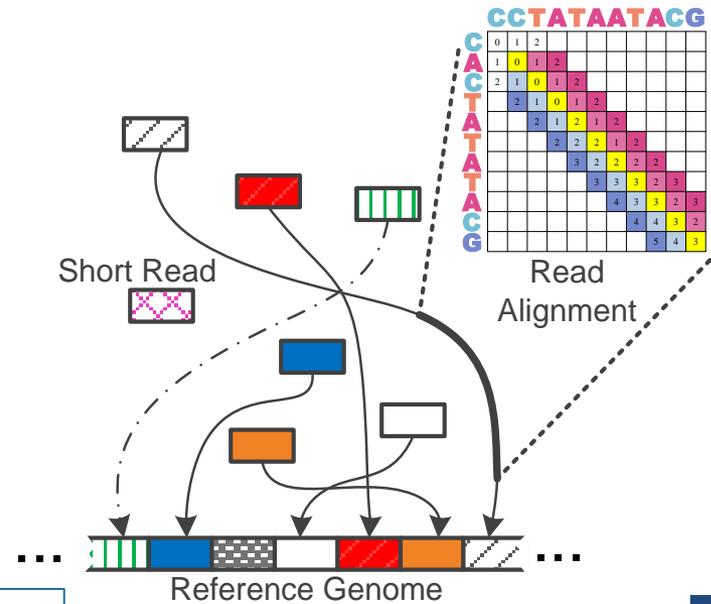
**SAFARI**

**... and more! All produce data with different properties.**



Billions of Short Reads

TATATACGCTACTAGTACGT  
 TTTAGTACGTACGT  
 ATACGCTACTAGTACGT  
 ACG CCCCTACGTA  
 ACGTACTAGTACGT  
 TTAGTACGTACGT  
 TACGTAATAAGTACGT  
 TACGCTACTAGTACGT  
 TTTAAAACGTA  
 CGTACTAGTACGT  
 GGGAGTACGTACGT



# 1 Sequencing

# Genome Analysis

# 2 Read Mapping

reference: TTTATCGCTTCCATGACGCAG

read1: ATCGCATCC

read2: TATCGCATC

read3: CATCCATGA

read4: CGCTTCCAT

read5: CCATGACGC

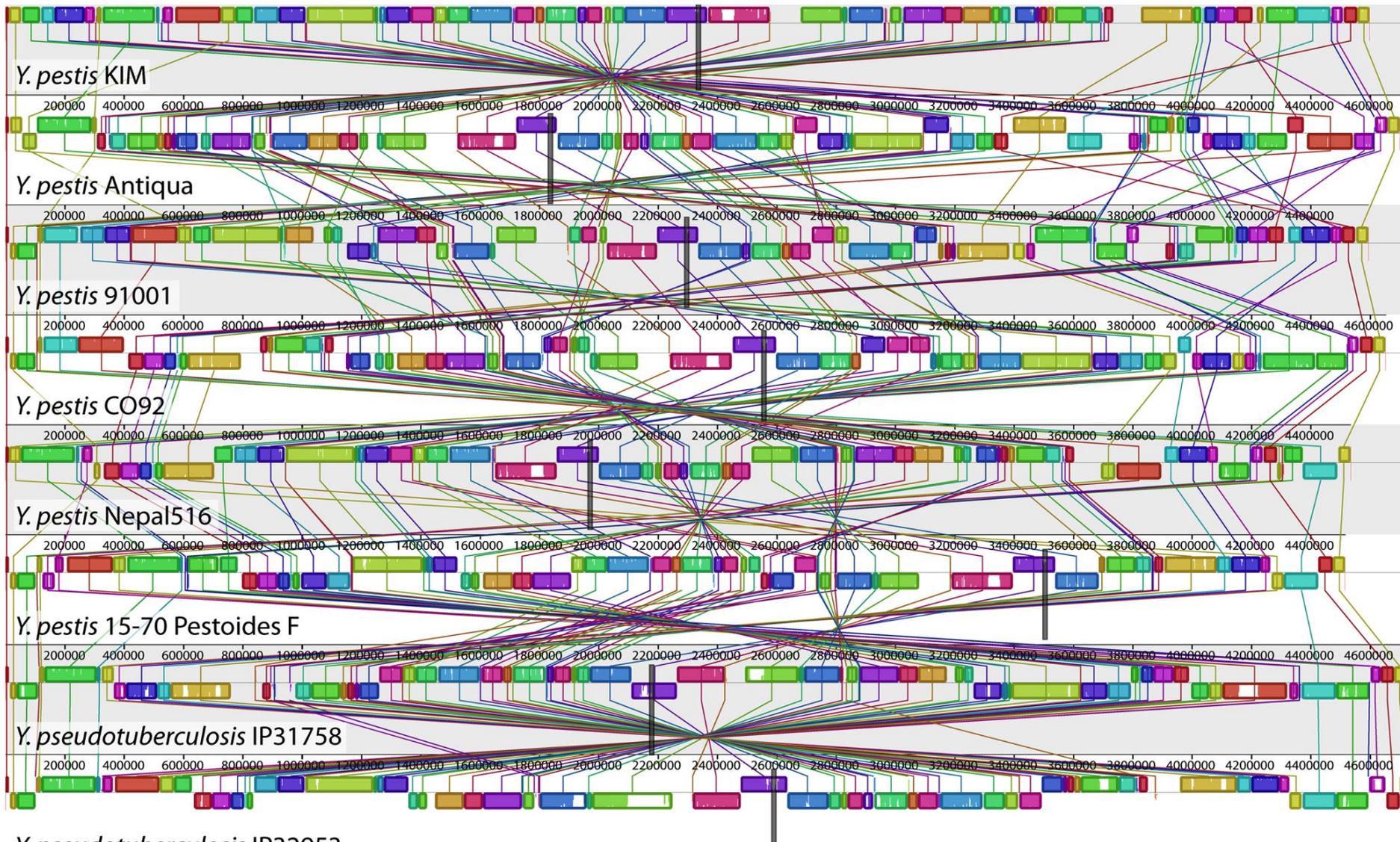
read6: TTCCATGAC



# 3 Variant Calling

# 4 Scientific Discovery

# Genome Sequence Alignment: Example



Source: By Aaron E. Darling, István Miklós, Mark A. Ragan - Figure 1 from Darling AE, Miklós I, Ragan MA (2008).

"Dynamics of Genome Rearrangement in Bacterial Populations". PLOS Genetics. DOI:10.1371/journal.pgen.1000128., CC BY 2.5, <https://commons.wikimedia.org/w/index.php?curid=30550950>



# Hash Table Based Read Mappers

---

- + Guaranteed to find *a//* mappings → sensitive
- + Can tolerate up to *e* errors

nature  
genetics

<http://mrfast.sourceforge.net/>

---

## Personalized copy number and segmental duplication maps using next-generation sequencing

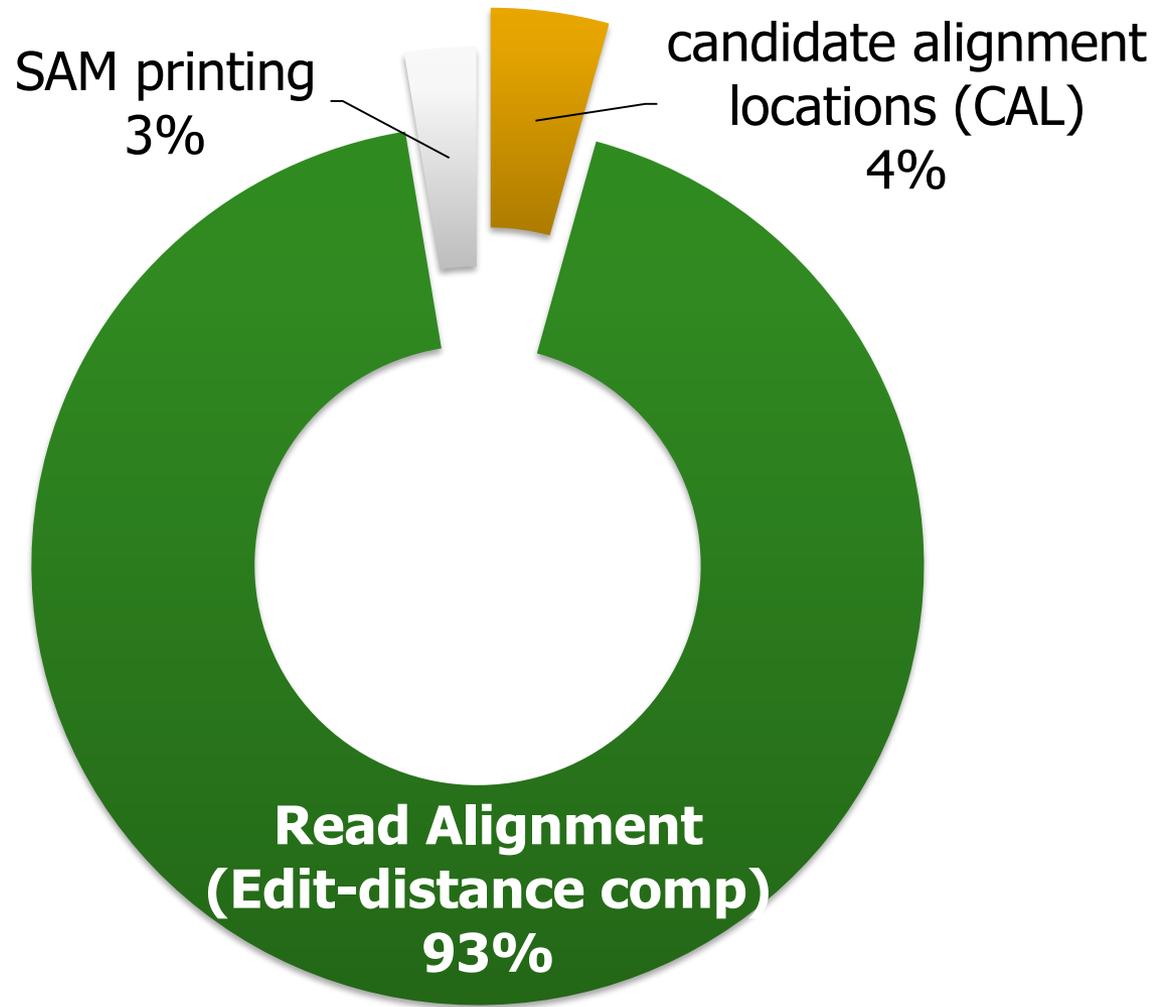
Can Alkan<sup>1,2</sup>, Jeffrey M Kidd<sup>1</sup>, Tomas Marques-Bonet<sup>1,3</sup>, Gozde Aksay<sup>1</sup>, Francesca Antonacci<sup>1</sup>, Fereydoun Hormozdiari<sup>4</sup>, Jacob O Kitzman<sup>1</sup>, Carl Baker<sup>1</sup>, Maika Malig<sup>1</sup>, Onur Mutlu<sup>5</sup>, S Cenk Sahinalp<sup>4</sup>, Richard A Gibbs<sup>6</sup> & Evan E Eichler<sup>1,2</sup>

---

Alkan+, "[Personalized copy number and segmental duplication maps using next-generation sequencing](#)", Nature Genetics 2009.

# Read Mapping Execution Time Breakdown

---



**Filter fast** before you align

Minimize costly

“approximate string comparisons”

# Our First Filter: Pure Software Approach

---

- Download source code and try for yourself
  - [Download link to FastHASH](#)

Xin *et al.* *BMC Genomics* 2013, **14**(Suppl 1):S13  
<http://www.biomedcentral.com/1471-2164/14/S1/S13>



**PROCEEDINGS**

**Open Access**

## Accelerating read mapping with FastHASH

Hongyi Xin<sup>1</sup>, Donghyuk Lee<sup>1</sup>, Farhad Hormozdiari<sup>2</sup>, Samihan Yedkar<sup>1</sup>, Onur Mutlu<sup>1\*</sup>, Can Alkan<sup>3\*</sup>

*From* The Eleventh Asia Pacific Bioinformatics Conference (APBC 2013)  
Vancouver, Canada. 21-24 January 2013

# Shifted Hamming Distance: SIMD Acceleration

---

*Bioinformatics*, 31(10), 2015, 1553–1560

doi: 10.1093/bioinformatics/btu856

Advance Access Publication Date: 10 January 2015

Original Paper

OXFORD

---

Sequence analysis

## **Shifted Hamming distance: a fast and accurate SIMD-friendly filter to accelerate alignment verification in read mapping**

Hongyi Xin<sup>1,\*</sup>, John Greth<sup>2</sup>, John Emmons<sup>2</sup>, Gennady Pekhimenko<sup>1</sup>,  
Carl Kingsford<sup>3</sup>, Can Alkan<sup>4,\*</sup> and Onur Mutlu<sup>2,\*</sup>

Xin+, **"Shifted Hamming Distance: A Fast and Accurate SIMD-friendly Filter to Accelerate Alignment Verification in Read Mapping"**, **Bioinformatics 2015.**

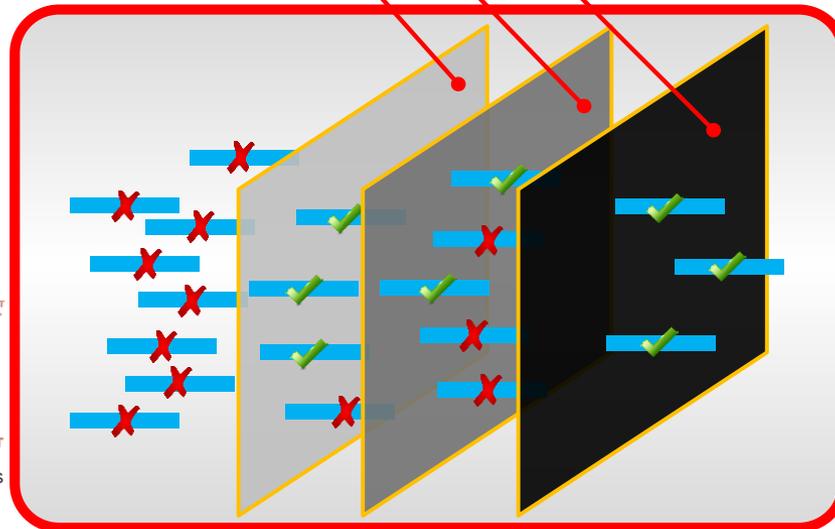
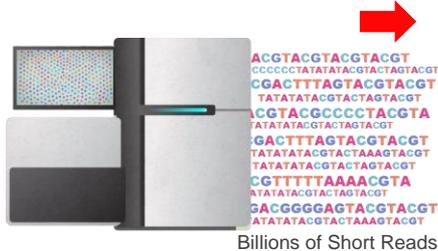
---

# An Example Solution: GateKeeper



Low Speed & High Accuracy  
 Medium Speed, Medium Accuracy  
 High Speed, Low Accuracy

$\times 10^{12}$   
mappings



$\times 10^3$   
mappings

CTATAATACG

C	0	1	2																
A	1	0	1	2															
C	2	1	0	1	2														
T		2	1	0	1	2													
A			2	1	2	1	2												
T				2	2	2	1	2											
A					3	2	2	2	2										
T						3	3	3	2	3									
A							4	3	3	2	3								
C								4	4	3	2								
G									5	4	3								

- 1 High throughput DNA sequencing (HTS) technologies
- 2 Read Pre-Alignment Filtering  
Fast & Low False Positive Rate
- 3 Read Alignment  
Slow & Zero False Positives

# FPGA-Based Alignment Filtering

---

- Mohammed Alser, Hasan Hassan, Hongyi Xin, Oguz Ergin, Onur Mutlu, and Can Alkan  
**"GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping"**  
***Bioinformatics***, [published online, May 31], 2017.  
[[Source Code](#)]  
[[Online link at Bioinformatics Journal](#)]

## GateKeeper: a new hardware architecture for accelerating pre-alignment in DNA short read mapping

Mohammed Alser ✉, Hasan Hassan, Hongyi Xin, Oğuz Ergin, Onur Mutlu ✉, Can Alkan ✉

*Bioinformatics*, Volume 33, Issue 21, 1 November 2017, Pages 3355–3363,

<https://doi.org/10.1093/bioinformatics/btx342>

**Published:** 31 May 2017    **Article history** ▼

# DNA Read Mapping & Filtering

---

- **Problem: Heavily bottlenecked by Data Movement**
- GateKeeper FPGA performance limited by DRAM bandwidth [Alser+, Bioinformatics 2017]
- Ditto for SHD on SIMD [Xin+, Bioinformatics 2015]
- **Solution: Processing-in-memory can alleviate the bottleneck**
- However, we need to design mapping & filtering algorithms to fit processing-in-memory

# In-Memory DNA Sequence Analysis

---

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

## GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies

Jeremie S. Kim<sup>1,6\*</sup>, Damla Senol Cali<sup>1</sup>, Hongyi Xin<sup>2</sup>, Donghyuk Lee<sup>3</sup>, Saugata Ghose<sup>1</sup>, Mohammed Alser<sup>4</sup>, Hasan Hassan<sup>6</sup>, Oguz Ergin<sup>5</sup>, Can Alkan<sup>4\*</sup> and Onur Mutlu<sup>6,1\*</sup>

From The Sixteenth Asia Pacific Bioinformatics Conference 2018  
Yokohama, Japan. 15-17 January 2018

# Quick Note: Key Principles and Results

---

- Two key principles:
  - **Exploit the structure of the genome** to minimize computation
  - **Morph and exploit the structure of the underlying hardware** to maximize performance and efficiency
  
- **Algorithm-architecture co-design** for DNA read mapping
  - **Speeds up** read mapping by **~200X (sometimes more)**
  - **Improves accuracy** of read mapping in the presence of errors

Xin et al., "Accelerating Read Mapping with FastHASH," BMC Genomics 2013.

Xin et al., "Shifted Hamming Distance: A Fast and Accurate SIMD-friendly Filter to Accelerate Alignment Verification in Read Mapping," Bioinformatics 2015.

Alser et al., "GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping," Bioinformatics 2017.

Kim et al., "Genome Read In-Memory (GRIM) Filter," BMC Genomics 2018.

# New Genome Sequencing Technologies

---

## Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions

Damla Senol Cali ✉, Jeremie S Kim, Saugata Ghose, Can Alkan, Onur Mutlu

*Briefings in Bioinformatics*, bby017, <https://doi.org/10.1093/bib/bby017>

**Published:** 02 April 2018    **Article history** ▼

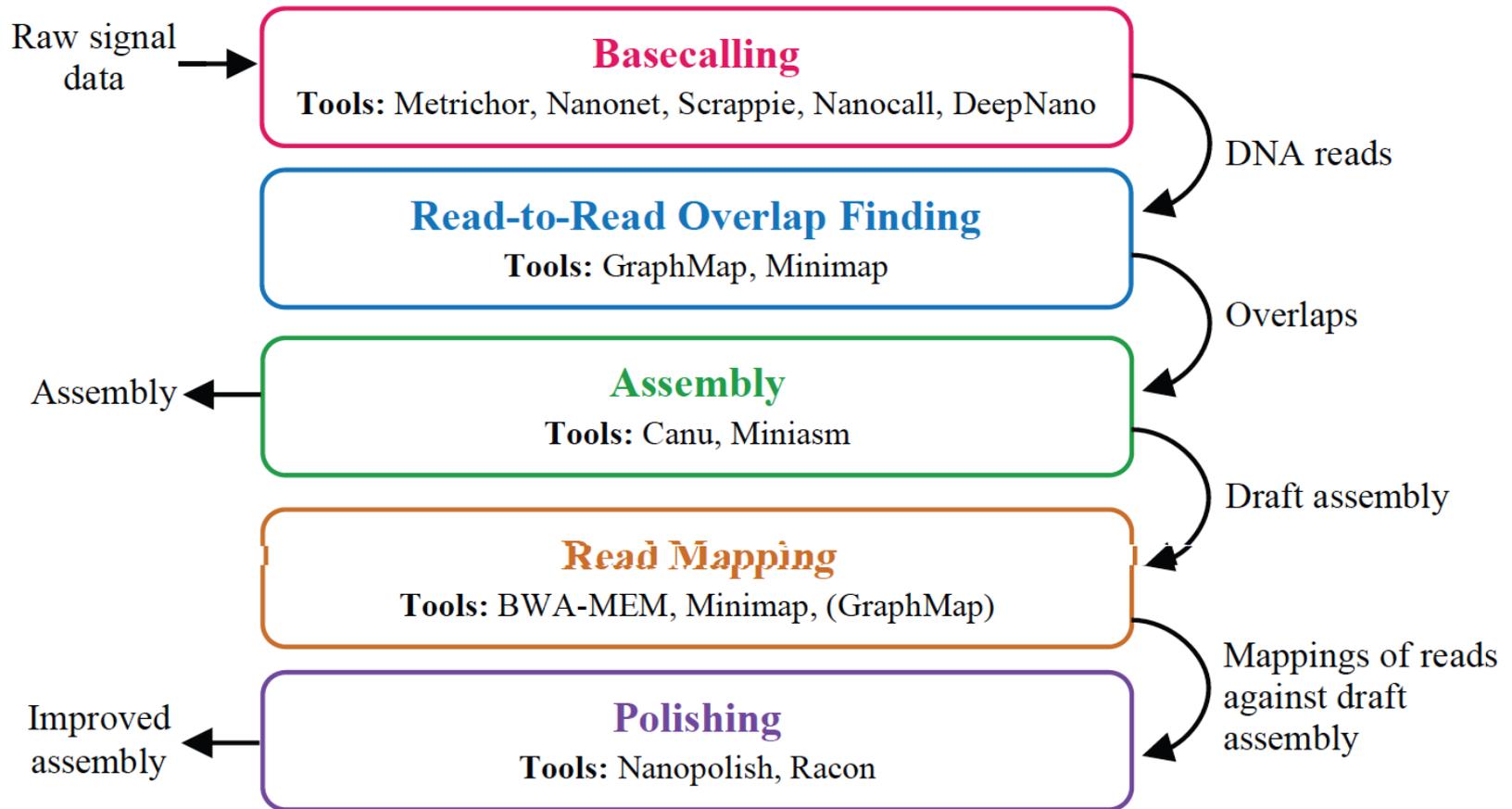


Oxford Nanopore MinION

Senol Cali+, “**Nanopore Sequencing Technology and Tools for Genome Assembly: Computational Analysis of the Current State, Bottlenecks and Future Directions**,” *Briefings in Bioinformatics*, 2018.

[\[Preliminary arxiv.org version\]](#)

# Nanopore Genome Assembly Pipeline



**Figure 1. The analyzed genome assembly pipeline using nanopore sequence data, with its five steps and the associated tools for each step.**

# More on Genome Analysis: Another Talk

---

## Accelerating Genome Analysis A Primer on an Ongoing Journey

Onur Mutlu

[omutlu@gmail.com](mailto:omutlu@gmail.com)

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

May 21, 2018

HiCOMB-17 Keynote Talk



**ETH** zürich

# Recall Our Dream

---

- An embedded device that can perform comprehensive genome analysis in real time (within a minute)
- Still a long ways to go
  - Energy efficiency
  - Performance (latency)
  - Security
  - Huge memory bottleneck

End of Backup Slides

# Brief Self Introduction

---



## ■ Onur Mutlu

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

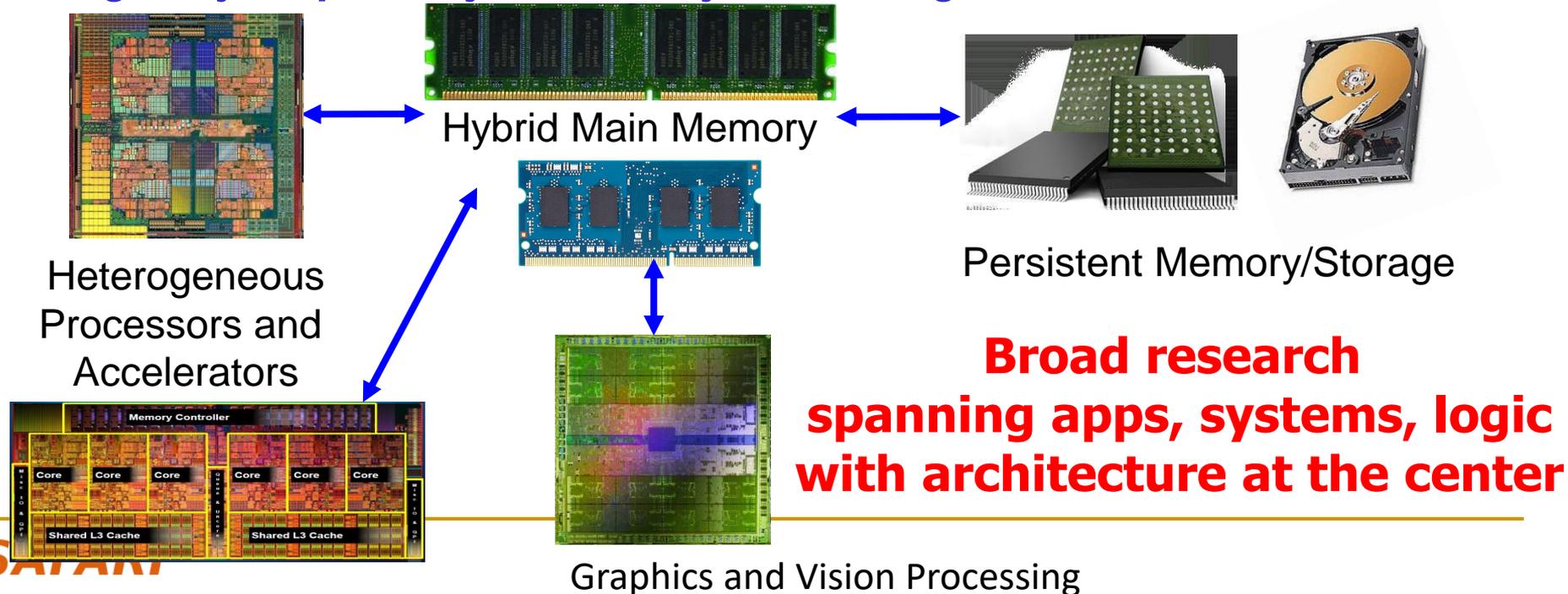
## ■ Research and Teaching in:

- ❑ Computer architecture, computer systems, hardware security, bioinformatics
- ❑ Memory and storage systems
- ❑ Hardware security, safety, predictability
- ❑ Fault tolerance
- ❑ Hardware/software cooperation
- ❑ Architectures for bioinformatics, health, medicine
- ❑ ...

# Current Research Focus Areas

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

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



# Four Key Directions

---

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