

# Intelligent Architectures for Intelligent Machines

Onur Mutlu

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

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

20 November 2019

Khalifa System on Chip Center Open House Keynote

**SAFARI**

**ETH** zürich

**Carnegie Mellon**

Computing

is Bottlenecked by Data



# Data is Key for AI, ML, Genomics, ...

---

- Important workloads are all data intensive
- They require rapid and efficient processing of large amounts of data
- Data is increasing
  - We can generate more than we can process

# Data is Key for Future Workloads

---



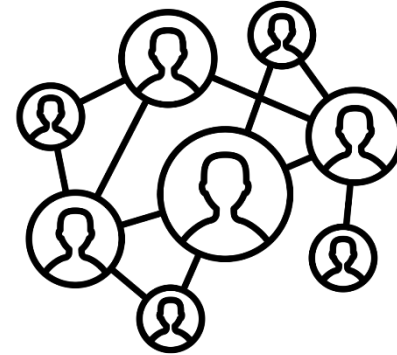
## In-memory Databases

[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

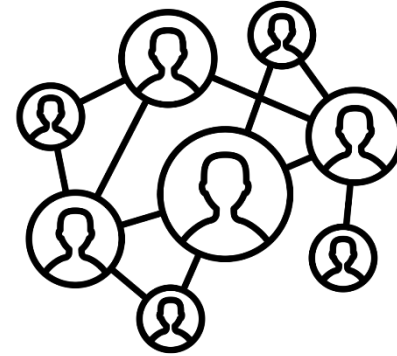
[Kanev+ (Google), ISCA'15]

# Data Overwhelms Modern Machines

---



**In-memory Databases**



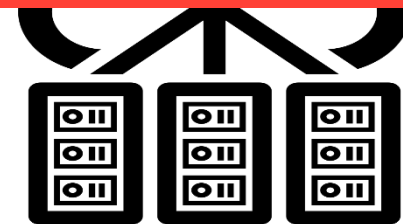
**Graph/Tree Processing**

**Data → performance & energy bottleneck**



**In-Memory Data Analytics**

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



**Datacenter Workloads**

[Kanev+ (Google), ISCA'15]

# Data is Key for Future 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**

# Data Overwhelms Modern Machines



**Chrome**



**TensorFlow Mobile**

Data → performance & energy bottleneck

**VP9**



**Video Playback**

Google's **video codec**

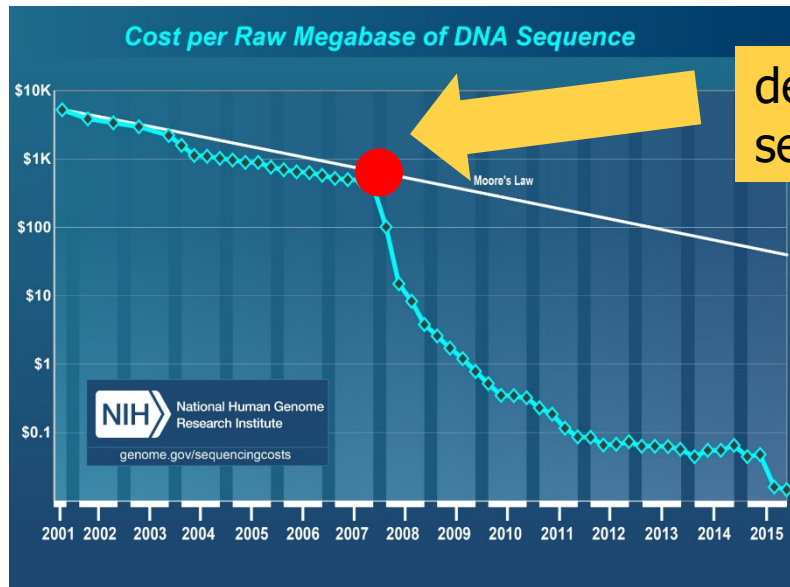
**VP9**



**Video Capture**

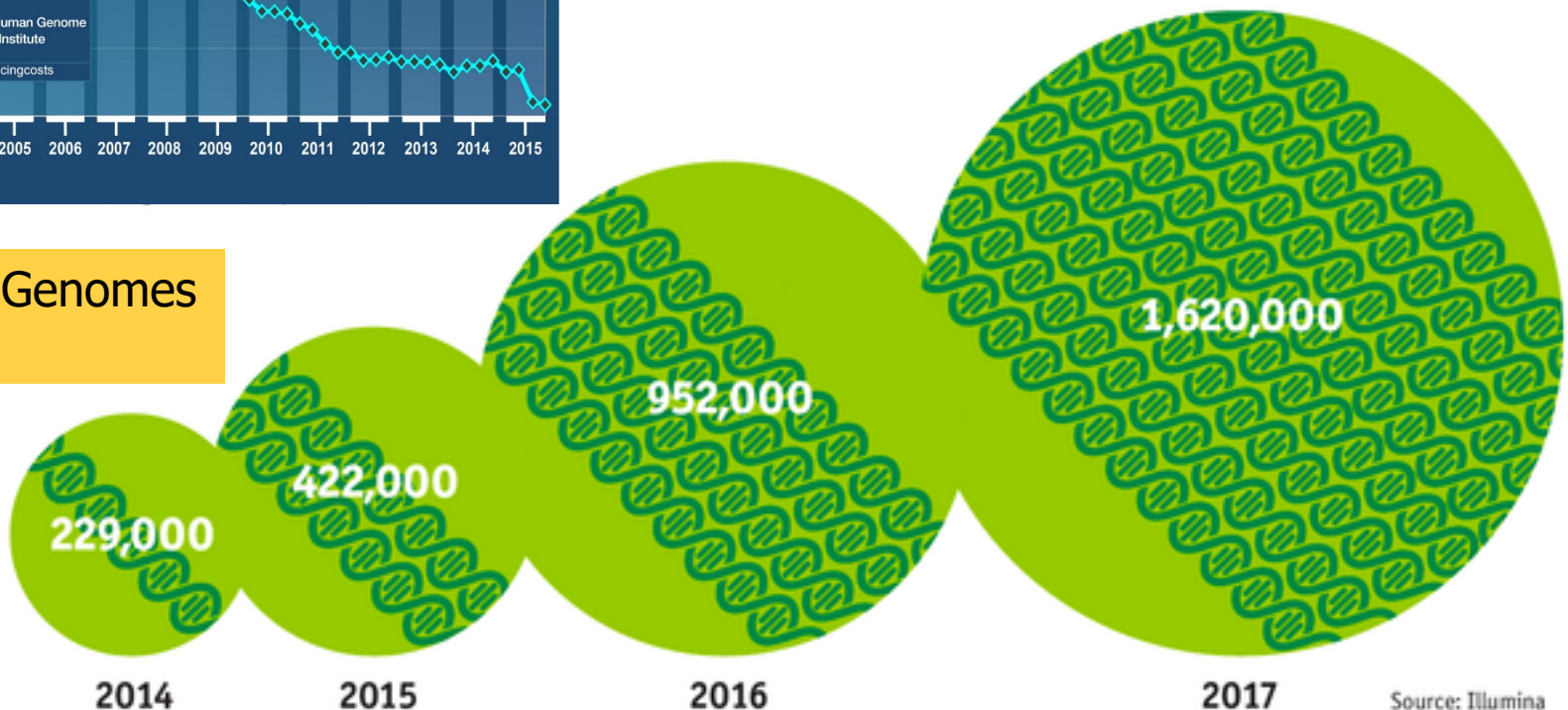
Google's **video codec**

# Data is Key for Future Workloads

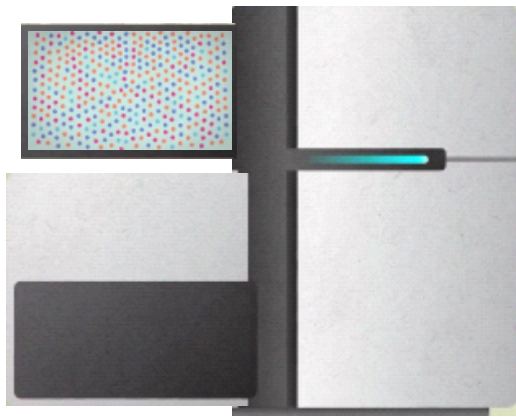


development of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced



The Economist



Billions of Short Reads

ATATATACGTACTAGTACGT  
 TTTAGTACGTACGT  
 ATACGTACTAGTACGT  
 CGCCCCTACGTA  
 ACGTACTAGTACGT  
 TTAGTACGTACGT  
 TACGTACTAAAGTACGT  
 TACGTACTAGTACGT  
 TTTAAACGTA  
 CGTACTAGTACGT  
 GGGAGTACGTACGT



## 1 Sequencing

# Genome Analysis

## 2 Read Mapping

Data → performance & energy bottleneck

read4: CGCTTCCAT  
 read5: CCATGACGC  
 read6: TTCCATGAC



## 3 Variant Calling

## 4 Scientific Discovery

# 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

Data → performance & energy bottleneck



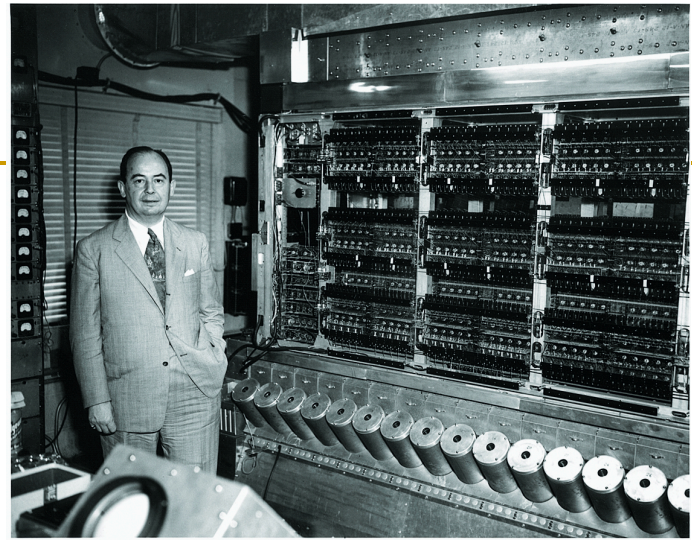
# Data Overwhelms Modern Machines ...

---

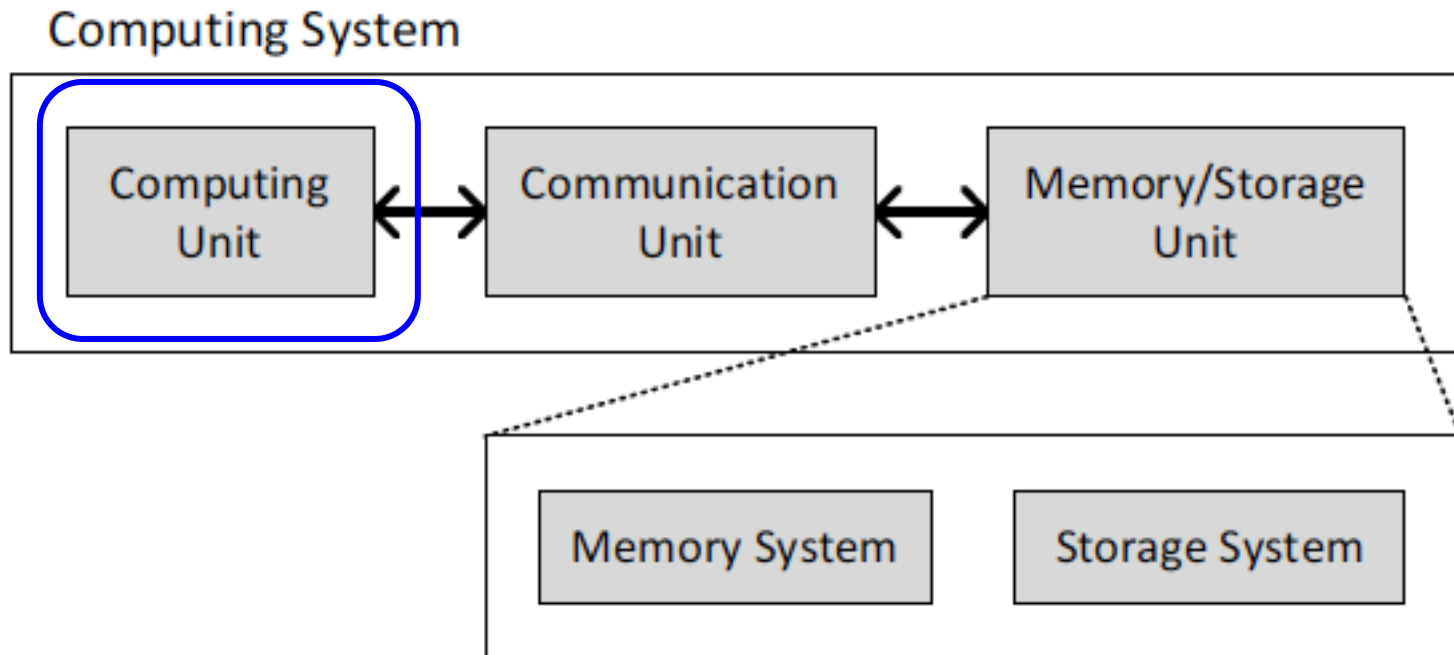
- Storage/memory capability
- Communication capability
- Computation capability
- Greatly impacts robustness, energy, performance, cost

# 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.





# Data Overwhelms Modern Machines



**Chrome**



**TensorFlow Mobile**

Data → performance & energy bottleneck

**VP9**



**Video Playback**

Google's **video codec**

**VP9**



**Video Capture**

Google's **video codec**

# Data Movement Overwhelms Modern Machines

- 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>

## An Intelligent Architecture Handles Data Well

# How to Handle Data Well

---

- Ensure data does not overwhelm the components
  - via intelligent algorithms
  - via intelligent architectures
  - via whole system designs: algorithm-architecture-devices
- Take advantage of vast amounts of data and metadata
  - to improve architectural & system-level decisions
- Understand and exploit properties of (different) data
  - to improve algorithms & architectures in various metrics

# Corollaries: Architectures Today ...

---

- Architectures are **terrible at dealing with data**
  - ❑ Designed to mainly store and move data vs. to compute
  - ❑ They are **processor-centric** as opposed to **data-centric**
- Architectures are **terrible at taking advantage of vast amounts of data** (and metadata) available to them
  - ❑ Designed to make simple decisions, ignoring lots of data
  - ❑ They make **human-driven decisions** vs. **data-driven** decisions
- Architectures are **terrible at knowing and exploiting different properties of application data**
  - ❑ Designed to treat all data as the same
  - ❑ They make **component-aware decisions** vs. **data-aware**



# Data-Centric (Memory-Centric) Architectures

# Low Latency Data Access

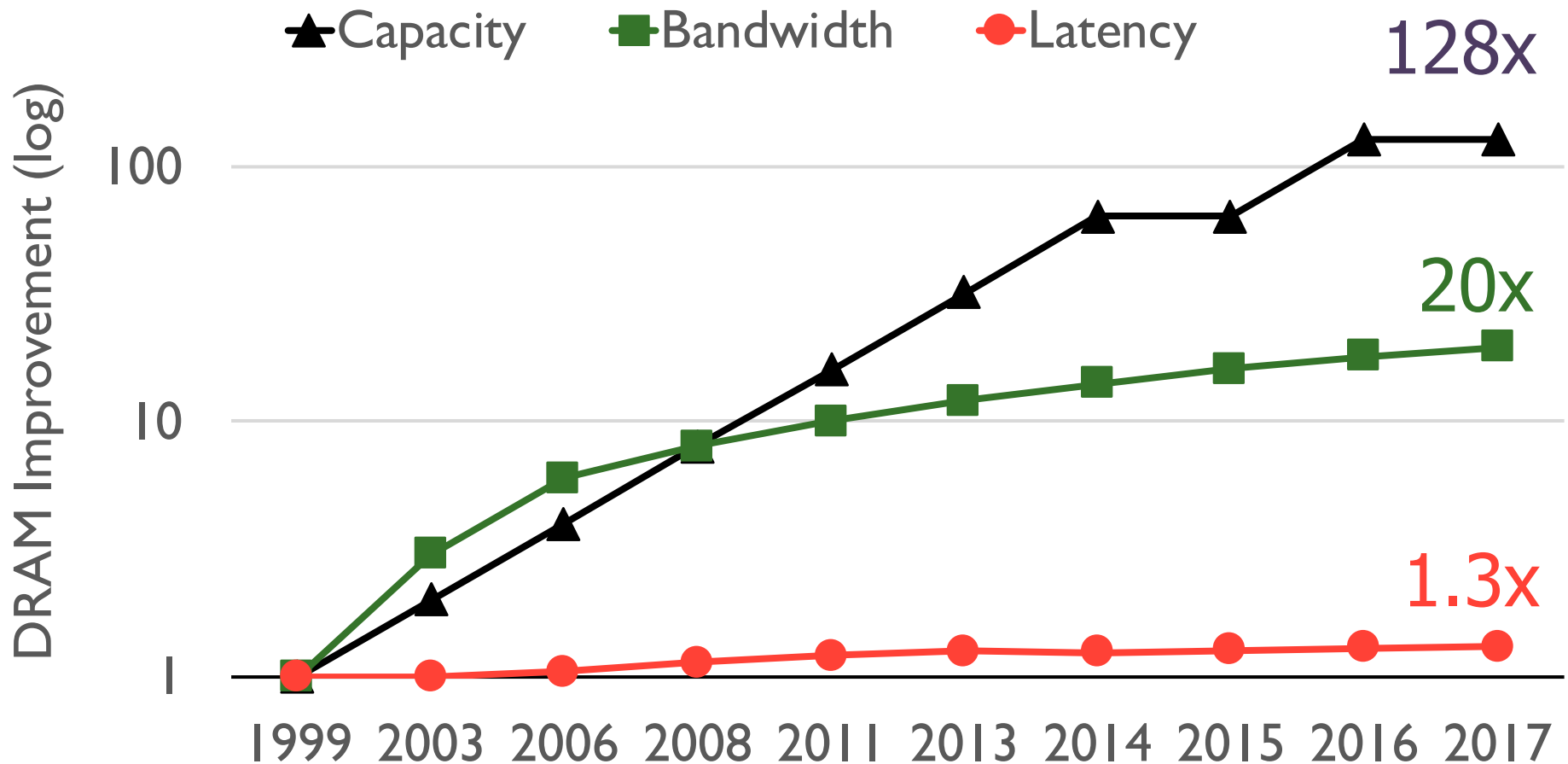
# Data-Centric Architectures: Properties

---

- **Process data where it resides** (where it makes sense)
  - Processing in and near memory structures
- **Low-latency & low-energy data access**
  - Low latency memory
  - Low energy memory
- **Low-cost data storage & processing**
  - High capacity memory at low cost: hybrid memory, compression
- **Intelligent data management**
  - Intelligent controllers handling robustness, security, cost, scaling

# Low-Latency & Low-Energy Data Access

# Main Memory Latency Lags Behind



Memory latency remains almost constant

# A Closer Look ...

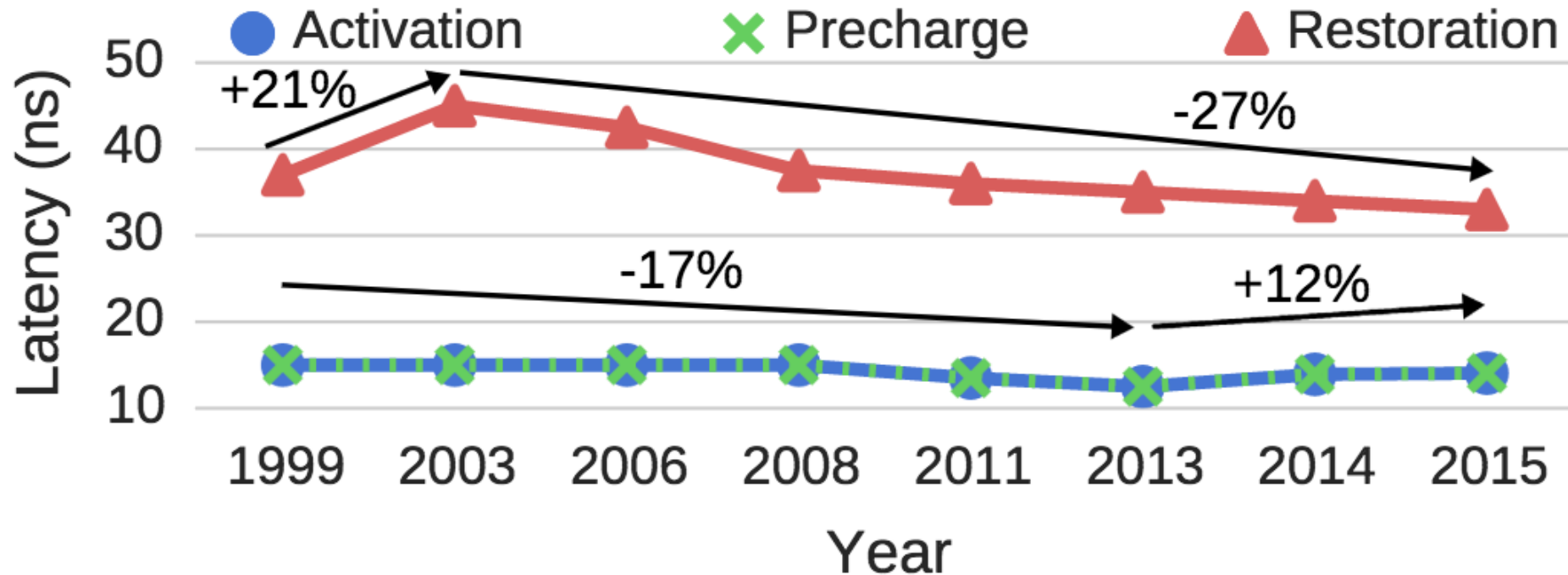


Figure 1: DRAM latency trends over time [20, 21, 23, 51].

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

# DRAM Latency Is Critical for Performance

---



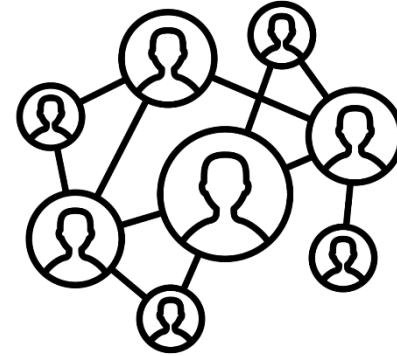
## In-memory Databases

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



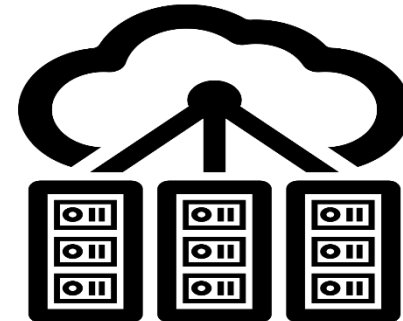
## In-Memory Data Analytics

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



## Graph/Tree Processing

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



## Datacenter Workloads

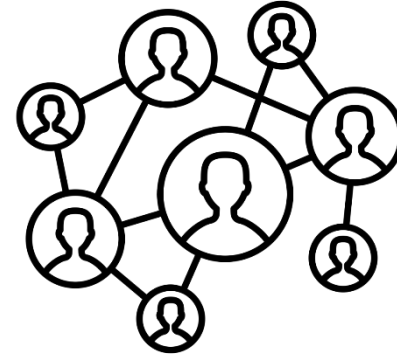
[Kanev+ (Google), ISCA'15]

# DRAM Latency Is Critical for Performance

---



**In-memory Databases**



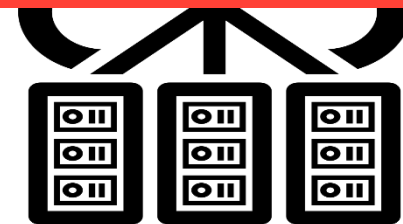
**Graph/Tree Processing**

Long memory latency → performance bottleneck



**In-Memory Data Analytics**

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



**Datacenter Workloads**

[Kanev+ (Google), ISCA'15]



# DRAM Latency Is Critical for Performance



**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**

# DRAM Latency Is Critical for Performance



**Chrome**



**TensorFlow Mobile**

Long memory latency → performance bottleneck

**VP9**



**Video Playback**

Google's **video codec**

**VP9**



**Video Capture**

Google's **video codec**

# New DRAM Types Increase Latency!

---

- Saugata Ghose, Tianshi Li, Nastaran Hajinazar, Damla Senol Cali, and Onur Mutlu,  
**"Demystifying Workload–DRAM Interactions: An Experimental Study"**  
*Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**), Phoenix, AZ, USA, June 2019.*  
[[Preliminary arXiv Version](#)]  
[[Abstract](#)]  
[[Slides \(pptx\)](#) ([pdf](#))]

## Demystifying Complex Workload–DRAM Interactions: An Experimental Study

Saugata Ghose<sup>†</sup>

Tianshi Li<sup>†</sup>

Nastaran Hajinazar<sup>‡†</sup>

Damla Senol Cali<sup>†</sup>

Onur Mutlu<sup>§†</sup>

<sup>†</sup>Carnegie Mellon University

<sup>‡</sup>Simon Fraser University

<sup>§</sup>ETH Zürich

# The Memory Latency Problem

---

- High memory latency is a significant **limiter of system performance and energy-efficiency**
- It is becoming increasingly so with **higher memory contention** in multi-core and heterogeneous architectures
  - Exacerbating the bandwidth need
  - Exacerbating the QoS problem
- It increases **processor design complexity** due to the mechanisms incorporated to tolerate memory latency

# Retrospective: Conventional Latency Tolerance Techniques

---

- Caching [initially by Wilkes, 1965]
  - Widely used, simple, effective, but inefficient, passive
  - Not all applications/phases exhibit temporal or spatial locality
- Prefetching [initially in IBM 360/91, 1967]

**None of These  
Fundamentally Reduce  
Memory Latency**

ongoing research effort

- Out-of-order execution [initially by Tomasulo, 1967]
  - **Tolerates cache misses that cannot be prefetched**
  - Requires extensive hardware resources for tolerating long latencies

# Two Major Sources of Latency Inefficiency

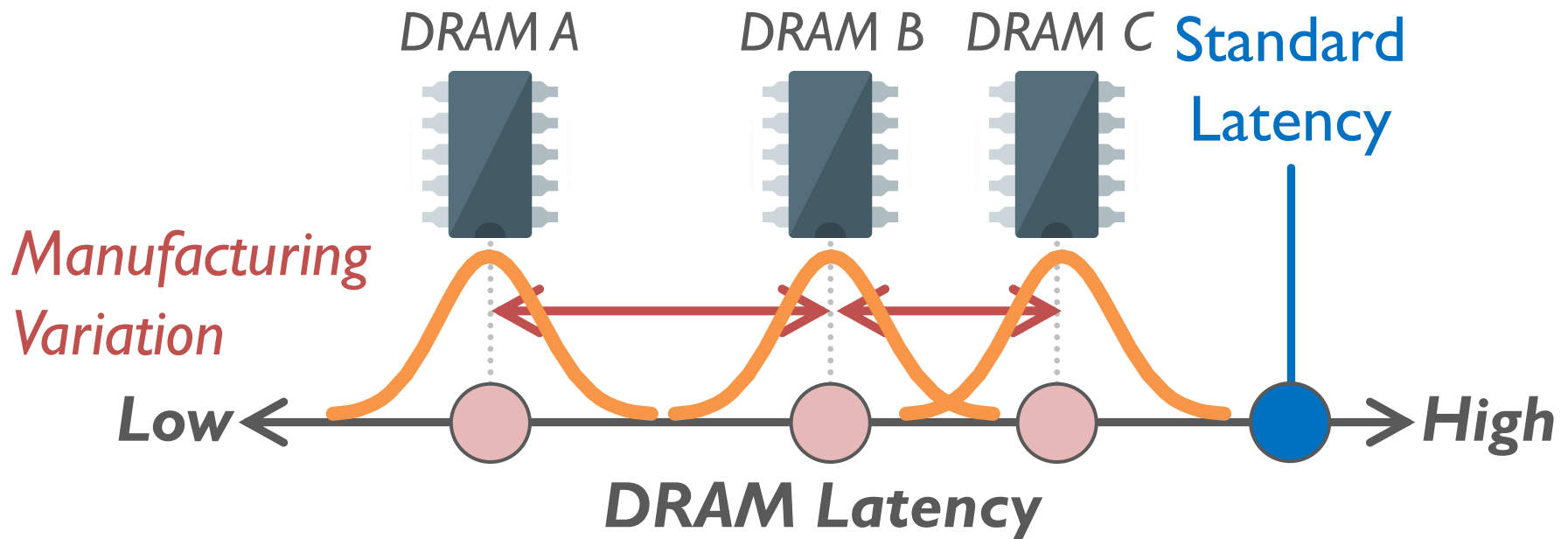
---

- Modern DRAM is **not** designed for low latency
  - Main focus is cost-per-bit (capacity)
- Modern DRAM latency is determined by **worst case** conditions and **worst case** devices
  - Much of memory latency is unnecessary

**Our Goal: Reduce Memory Latency  
at the Source of the Problem**

# Why is Memory Latency High?

- DRAM latency: Delay as specified in DRAM standards
  - Doesn't reflect true DRAM device latency
- Imperfect manufacturing process → latency variation
- **High standard latency** chosen to increase yield

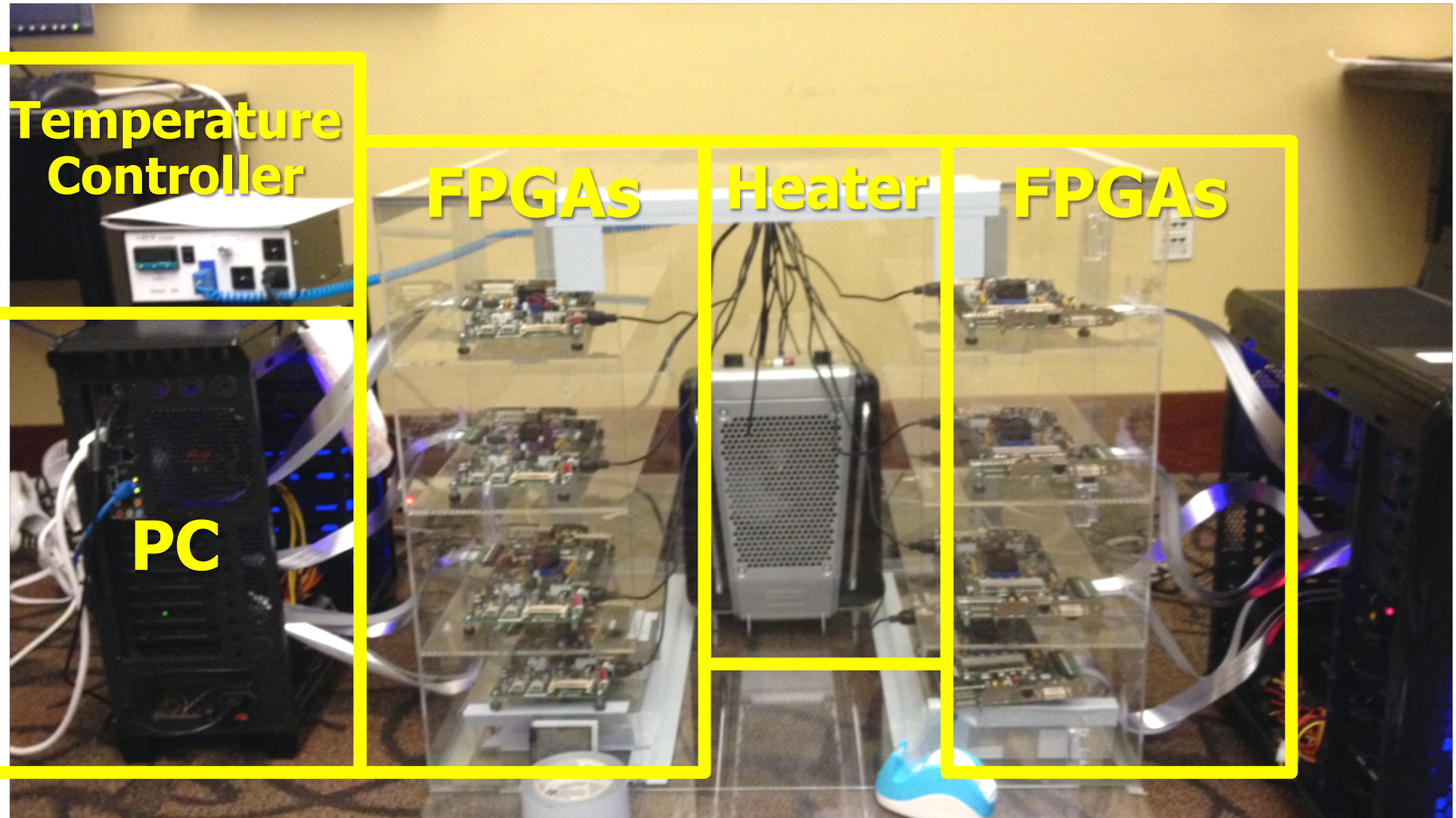


# Adaptive-Latency DRAM

- *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

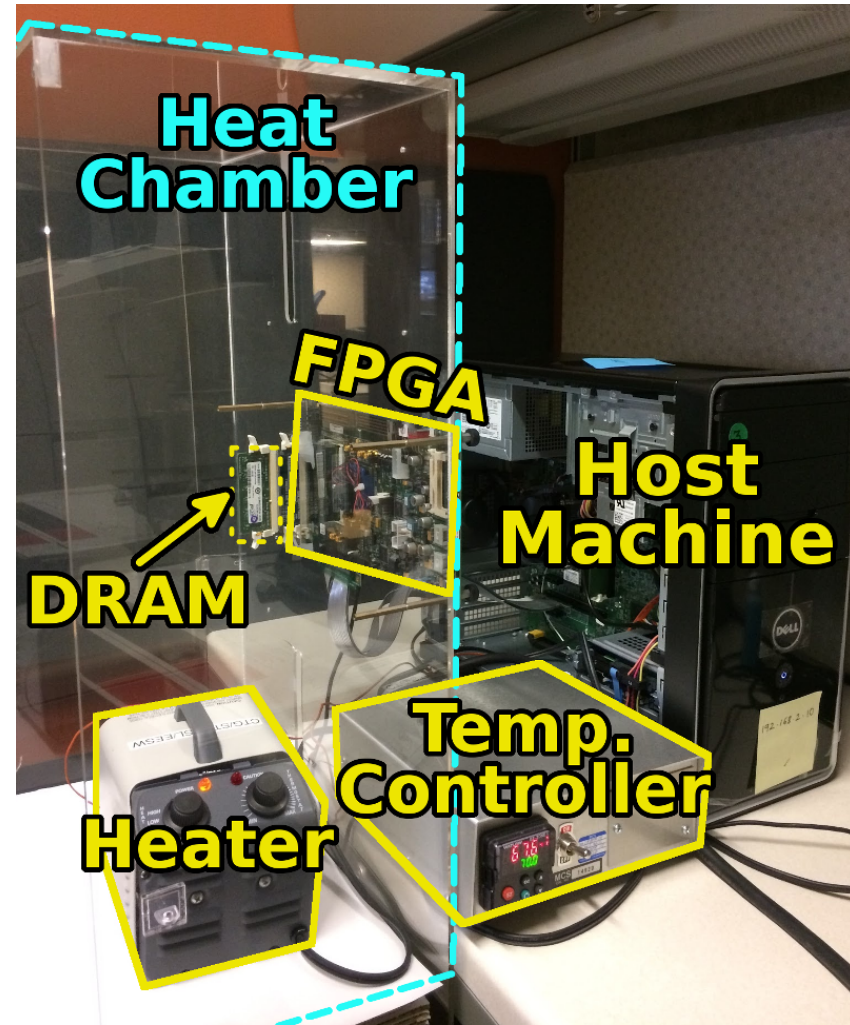


# Infrastructures to Understand Such Issues



# 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>

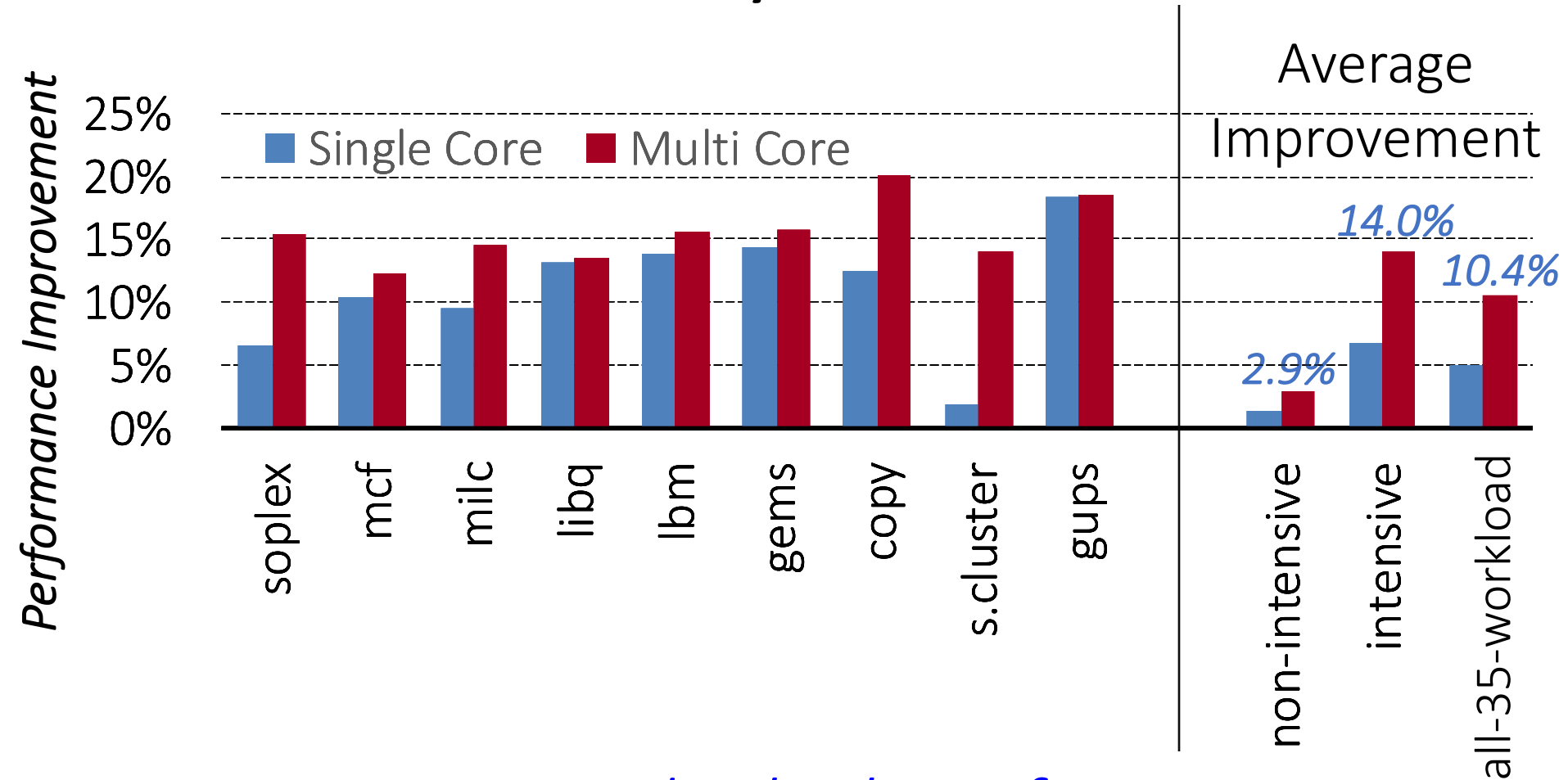
<sup>1</sup>*ETH Zürich*   <sup>2</sup>*TOBB University of Economics & Technology*   <sup>3</sup>*Carnegie Mellon University*  
<sup>4</sup>*University of Virginia*   <sup>5</sup>*Microsoft Research*   <sup>6</sup>*NVIDIA Research*

# 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 Performance



*AL-DRAM provides high performance on memory-intensive workloads*

# Reducing Latency Also Reduces Energy

---

- AL-DRAM reduces DRAM power consumption
- 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

# 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]
  - DRAM Latency PUF [HPCA 2018]
  - DRAM Latency True Random Number Generator [HPCA 2019]
  - ...
- We would like to find sources of latency heterogeneity and exploit them to minimize latency (or create other benefits)

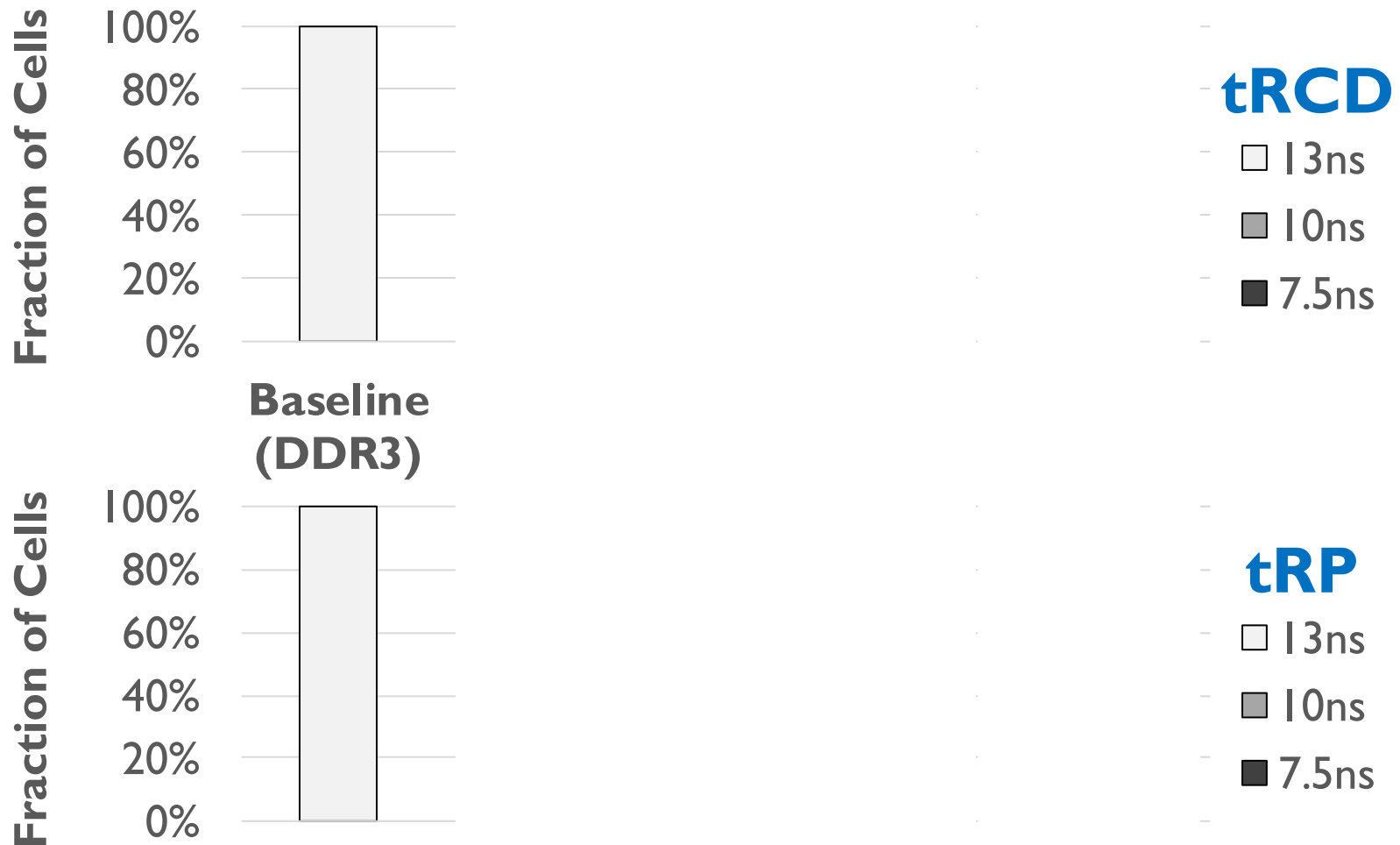


# FLY-DRAM: Reducing DRAM Latency

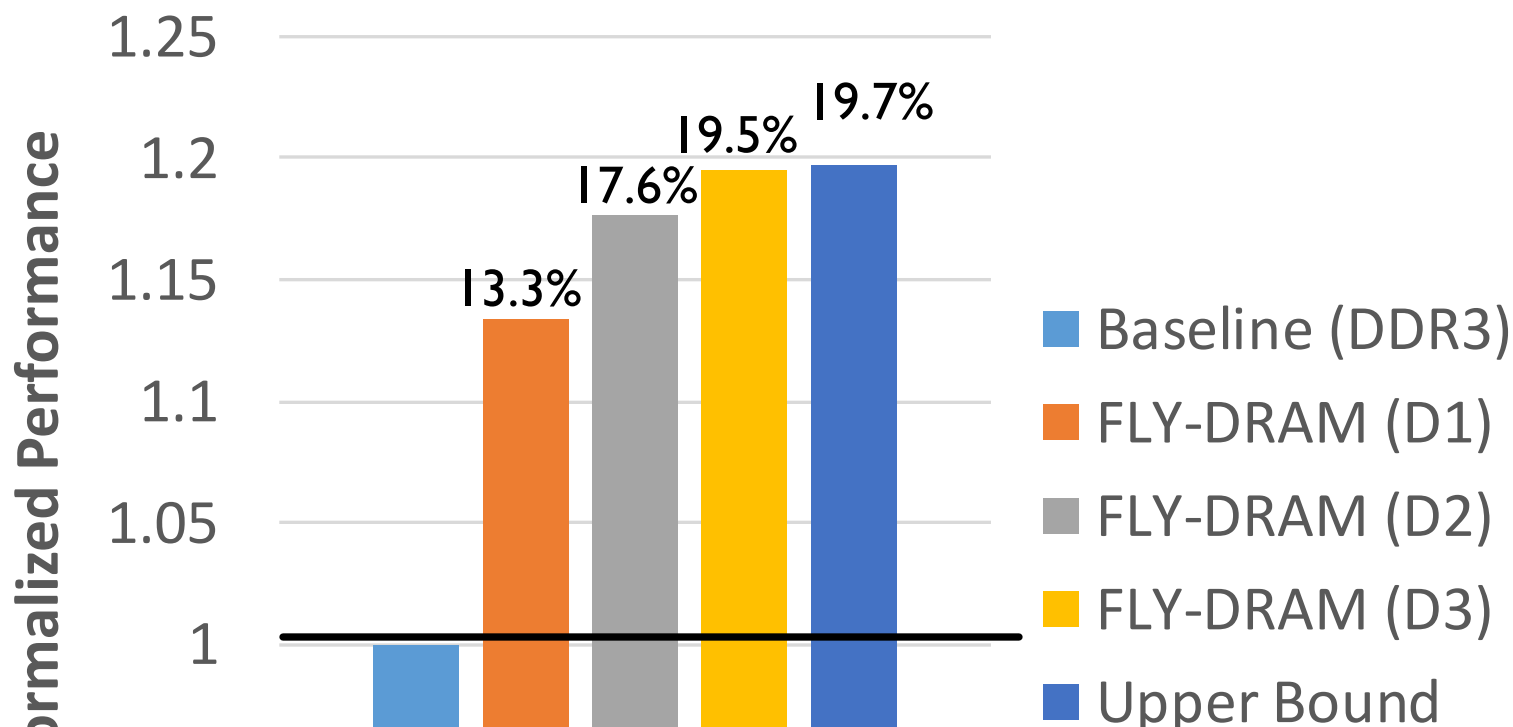
---

- **Observation:** DRAM timing errors (slow DRAM cells) are concentrated on certain DRAM regions
- **Flexible-Latency (FLY) DRAM**
  - A software-transparent design that reduces latency
- **Key idea:**
  - 1) Divide memory into regions of different latencies
  - 2) *Memory controller:* Use lower latency for regions without slow cells; higher latency for other regions

# FLY-DRAM Configurations



# FLY-DRAM Performance Results



**FLY-DRAM improves performance  
by exploiting spatial latency variation in DRAM**

# 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

# 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

# Solar-DRAM: Exploiting Spatial Variation

---

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu,  
**"Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines"**  
*Proceedings of the 36th IEEE International Conference on Computer Design (ICCD)*, Orlando, FL, USA, October 2018.

## Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines

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

# 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

# DRAM Latency True Random Number Generator

---

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, Lois Orosa, and Onur Mutlu, **"D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput"**  
*Proceedings of the 25th International Symposium on High-Performance Computer Architecture (HPCA)*, Washington, DC, USA, February 2019.

## D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

Jeremie S. Kim<sup>‡§</sup>

Minesh Patel<sup>§</sup>

Hasan Hassan<sup>§</sup>

Lois Orosa<sup>§</sup>

Onur Mutlu<sup>§‡</sup>

<sup>‡</sup>Carnegie Mellon University

<sup>§</sup>ETH Zürich



# ChargeCache: Exploiting Access Patterns

---

- Hasan Hassan, Gennady Pekhimenko, Nandita Vijaykumar, Vivek Seshadri, Donghyuk Lee, Oguz Ergin, and Onur Mutlu,  
**"ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality"**  
*Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (HPCA)*, Barcelona, Spain, March 2016.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Source Code](#)]

## ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality

Hasan Hassan<sup>†\*</sup>, Gennady Pekhimenko<sup>†</sup>, Nandita Vijaykumar<sup>†</sup>  
Vivek Seshadri<sup>†</sup>, Donghyuk Lee<sup>†</sup>, Oguz Ergin<sup>\*</sup>, Onur Mutlu<sup>†</sup>

# Exploiting Subarray Level Parallelism

---

- Yoongu Kim, Vivek Seshadri, Donghyuk Lee, Jamie Liu, and Onur Mutlu,  
**"A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM"**  
*Proceedings of the 39th International Symposium on Computer Architecture (ISCA)*, Portland, OR, June 2012. [Slides \(pptx\)](#)

## A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM

Yoongu Kim

Vivek Seshadri

Donghyuk Lee

Jamie Liu

Onur Mutlu

Carnegie Mellon University

# Tiered-Latency DRAM

---

- Donghyuk Lee, Yoongu Kim, Vivek Seshadri, Jamie Liu, Lavanya Subramanian, and Onur Mutlu,  
**"Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture"**  
*Proceedings of the 19th International Symposium on High-Performance Computer Architecture (HPCA)*, Shenzhen, China, February 2013. [Slides \(pptx\)](#)

## Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture

Donghyuk Lee   Yoongu Kim   Vivek Seshadri   Jamie Liu   Lavanya Subramanian   Onur Mutlu  
Carnegie Mellon University

# LISA: Low-cost Inter-linked Subarrays

---

- Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,  
**"Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM"**  
*Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (HPCA)*, Barcelona, Spain, March 2016.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Source Code](#)]

## Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM

Kevin K. Chang<sup>†</sup>, Prashant J. Nair<sup>\*</sup>, Donghyuk Lee<sup>†</sup>, Saugata Ghose<sup>†</sup>, Moinuddin K. Qureshi<sup>\*</sup>, and Onur Mutlu<sup>†</sup>

<sup>†</sup>Carnegie Mellon University    <sup>\*</sup>Georgia Institute of Technology

# The CROW Substrate for DRAM

---

- Hasan Hassan, Minesh Patel, Jeremie S. Kim, A. Giray Yaglikci, Nandita Vijaykumar, Nika Mansourighiasi, Saugata Ghose, and Onur Mutlu,  
**"CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability"**  
*Proceedings of the 46th International Symposium on Computer Architecture (ISCA), Phoenix, AZ, USA, June 2019.*

## **CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability**

Hasan Hassan<sup>†</sup>   Minesh Patel<sup>†</sup>   Jeremie S. Kim<sup>†§</sup>   A. Giray Yaglikci<sup>†</sup>  
Nandita Vijaykumar<sup>†§</sup>   Nika Mansouri Ghiasi<sup>†</sup>   Saugata Ghose<sup>§</sup>   Onur Mutlu<sup>†§</sup>

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

# Reducing Refresh Latency

---

- Anup Das, Hasan Hassan, and Onur Mutlu,  
**"VRL-DRAM: Improving DRAM Performance via  
Variable Refresh Latency"**  
*Proceedings of the 55th Design Automation  
Conference (DAC)*, San Francisco, CA, USA, June 2018.

## VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency

Anup Das  
Drexel University  
Philadelphia, PA, USA  
anup.das@drexel.edu

Hasan Hassan  
ETH Zürich  
Zürich, Switzerland  
hhasan@ethz.ch

Onur Mutlu  
ETH Zürich  
Zürich, Switzerland  
omutlu@gmail.com

# Parallelizing Refreshes and Accesses

---

- 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<sup>†</sup>

Alaa R. Alameldeen<sup>†</sup>   Chris Wilkerson<sup>†</sup>   Yoongu Kim   Onur Mutlu

Carnegie Mellon University   <sup>†</sup>Intel Labs

# Eliminating Refreshes

---

- 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



# 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

# VAMPIRE DRAM Power Model

---

- Saugata Ghose, A. Giray Yaglikci, Raghav Gupta, Donghyuk Lee, Kais Kudrolli, William X. Liu, Hasan Hassan, Kevin K. Chang, Niladrish Chatterjee, Aditya Agrawal, Mike O'Connor, and Onur Mutlu,  
**"What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study"**

*Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**), Irvine, CA, USA, June 2018.*

[[Abstract](#)]

## What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study

Saugata Ghose <sup>†</sup>	Abdullah Giray Yağlıkçı <sup>‡†</sup>	Raghav Gupta <sup>†</sup>	Donghyuk Lee <sup>§</sup>
Kais Kudrolli <sup>†</sup>	William X. Liu <sup>†</sup>	Hasan Hassan <sup>‡</sup>	Kevin K. Chang <sup>†</sup>
Niladrish Chatterjee <sup>§</sup>	Aditya Agrawal <sup>§</sup>	Mike O'Connor <sup>§¶</sup>	Onur Mutlu <sup>‡†</sup>

<sup>†</sup>Carnegie Mellon University

<sup>‡</sup>ETH Zürich

<sup>§</sup>NVIDIA

<sup>¶</sup>University of Texas at Austin

We Can Reduce  
Memory Latency  
with Change of Mindset

Main Memory Needs  
Intelligent Controllers  
to Reduce Latency

# Data-Centric Architectures: Properties

---

- **Process data where it resides** (where it makes sense)
  - Processing in and near memory structures
- **Low-latency and low-energy data access**
  - Low latency memory
  - Low energy memory
- **Low-cost data storage and processing**
  - High capacity memory at low cost: hybrid memory, compression
- **Intelligent data management**
  - Intelligent controllers handling robustness, security, cost

# Processing Data Where It Makes Sense

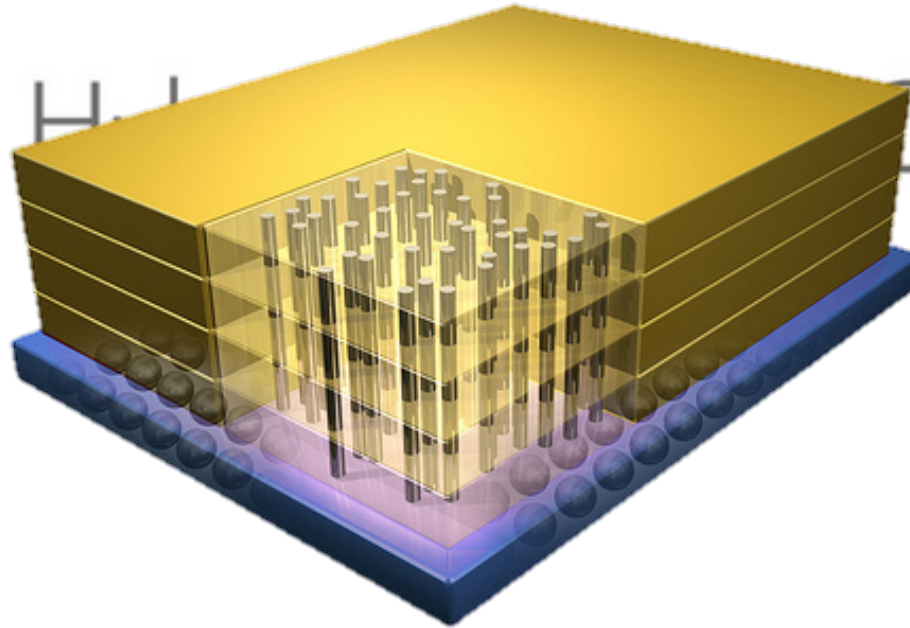
# Why In-Memory Computation Today?

---

- Push from Technology
  - DRAM Scaling at jeopardy
    - Controllers close to DRAM
    - Industry open to new memory architectures

# Why In-Memory Computation Today?

---





# Memory Scaling Issues **Were** Real

---

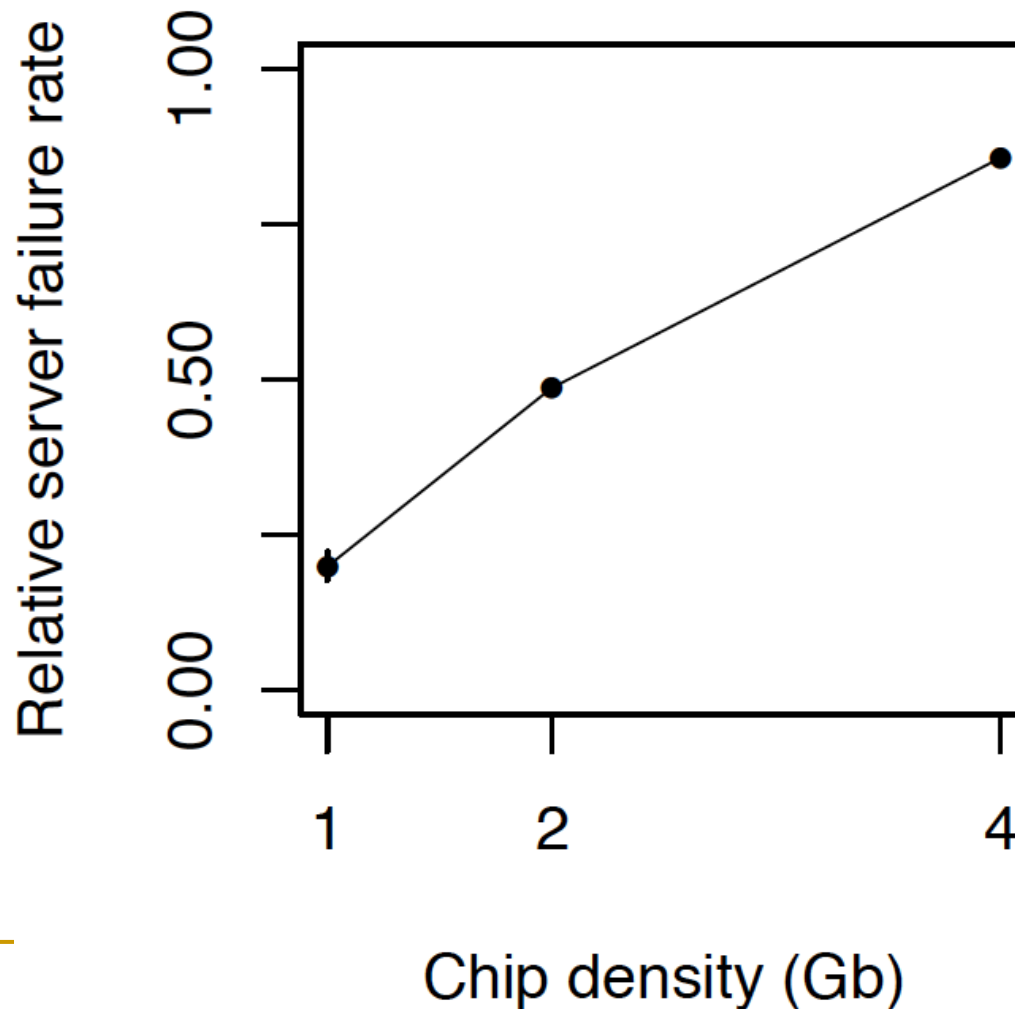
- Onur Mutlu,  
**"Memory Scaling: A Systems Architecture Perspective"**  
*Proceedings of the 5th International Memory Workshop (IMW)*, Monterey, CA, May 2013. Slides  
(pptx) (pdf)  
EETimes Reprint

## Memory Scaling: A Systems Architecture Perspective

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

# 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



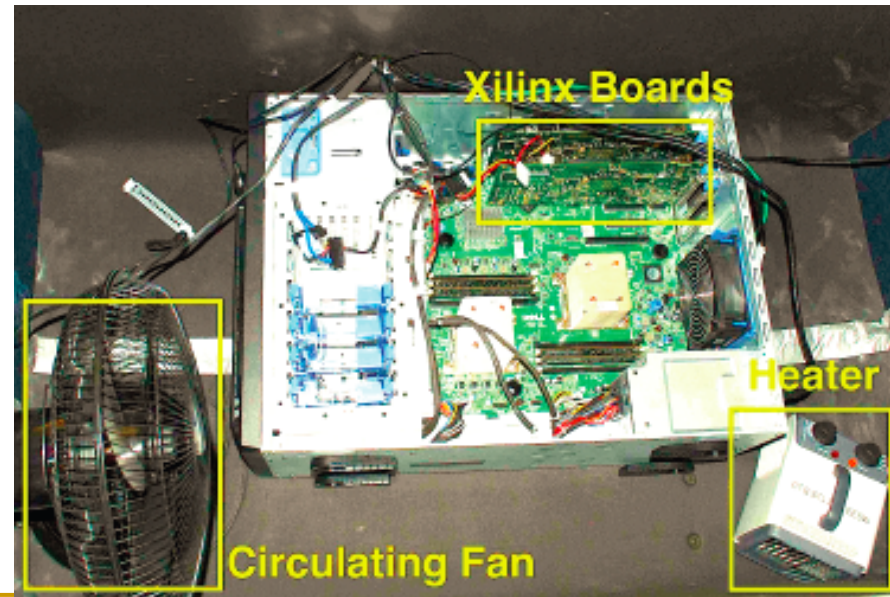
An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

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)

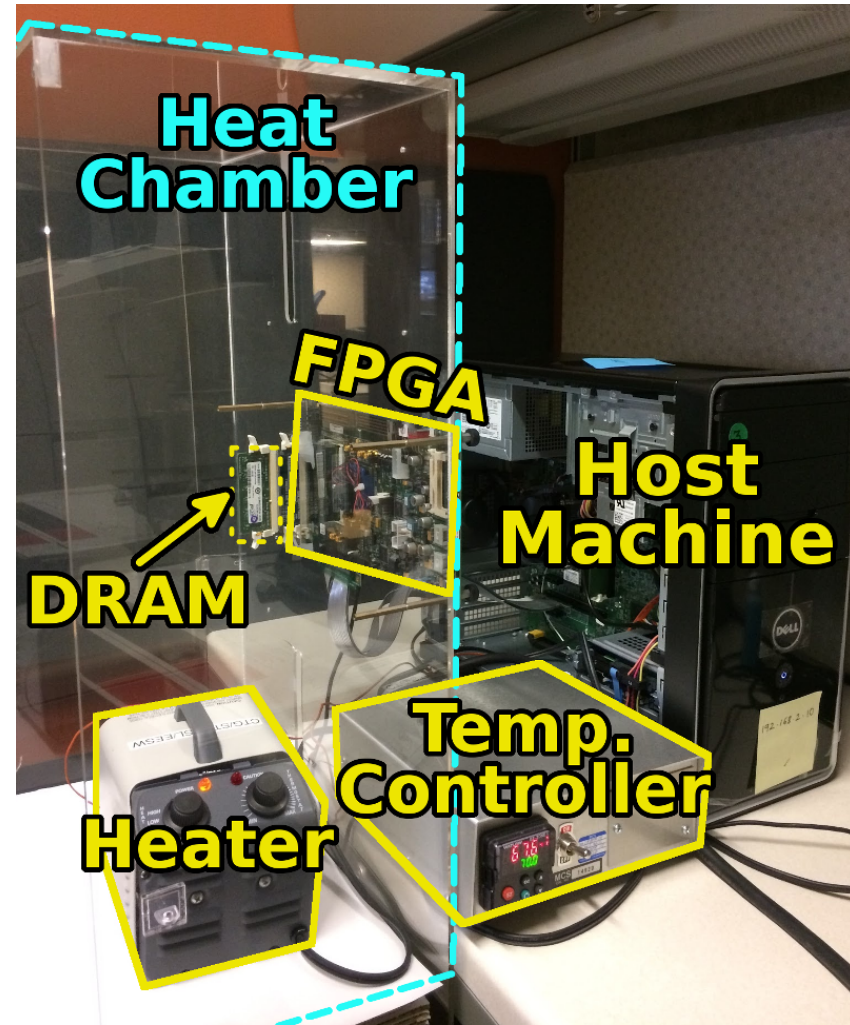






# SoftMC: Open Source DRAM Infrastructure

- 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**

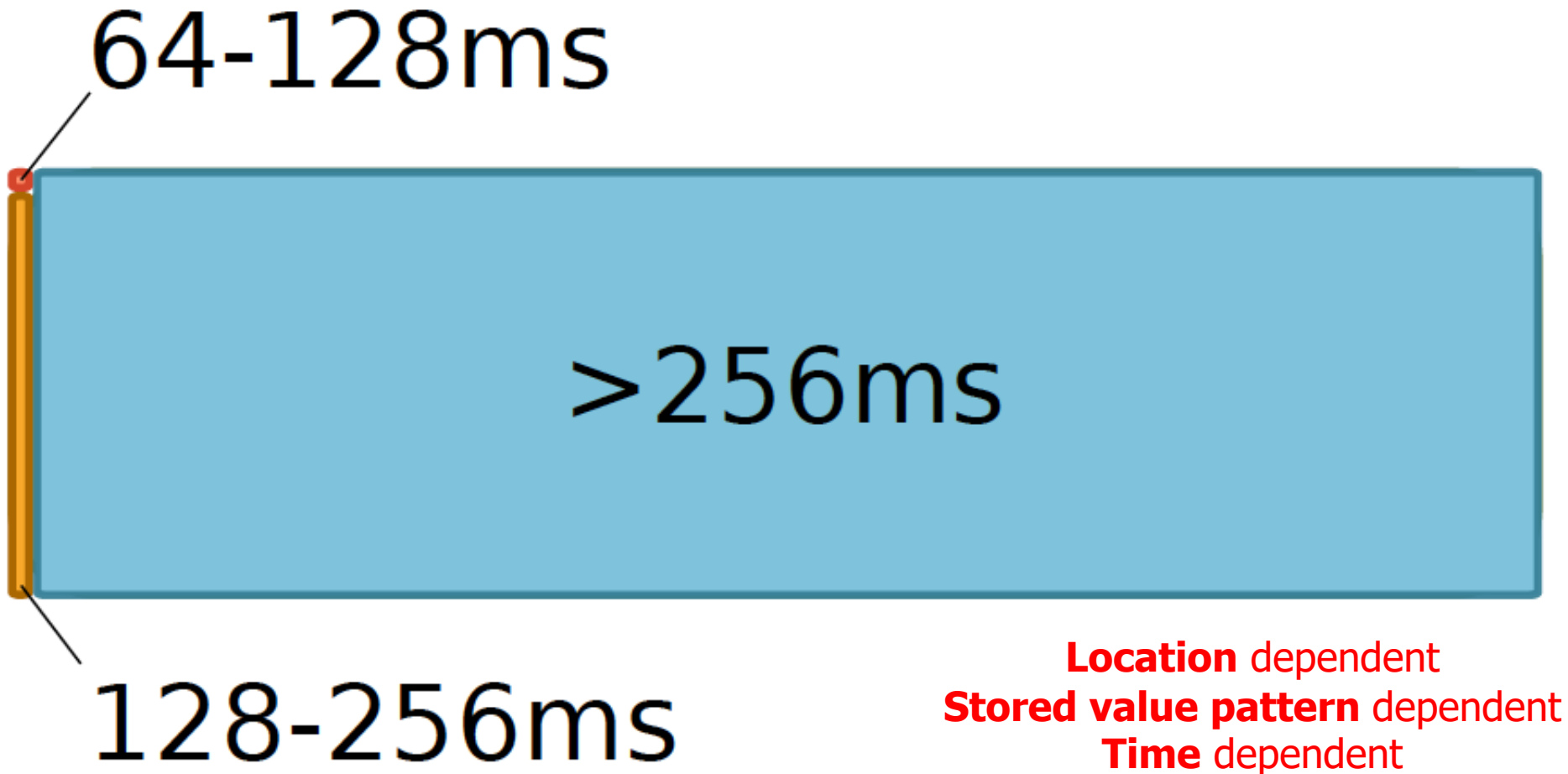
Hasan Hassan<sup>1,2,3</sup> Nandita Vijaykumar<sup>3</sup> Samira Khan<sup>4,3</sup> Saugata Ghose<sup>3</sup> Kevin Chang<sup>3</sup>  
Gennady Pekhimenko<sup>5,3</sup> Donghyuk Lee<sup>6,3</sup> Oguz Ergin<sup>2</sup> Onur Mutlu<sup>1,3</sup>

<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:





## Main Memory Needs Intelligent Controllers

# More on DRAM Refresh (I)

---

- 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

# More on DRAM Refresh (II)

---

- 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<sup>\*</sup>  
Carnegie Mellon University  
5000 Forbes Ave.  
Pittsburgh, PA 15213  
[jamiel@alumni.cmu.edu](mailto:jamiel@alumni.cmu.edu)

Ben Jaiyen<sup>\*</sup>  
Carnegie Mellon University  
5000 Forbes Ave.  
Pittsburgh, PA 15213  
[bjaiyen@alumni.cmu.edu](mailto:bjaiyen@alumni.cmu.edu)

Yoongu Kim  
Carnegie Mellon University  
5000 Forbes Ave.  
Pittsburgh, PA 15213  
[yoonguk@ece.cmu.edu](mailto:yoonguk@ece.cmu.edu)

Chris Wilkerson  
Intel Corporation  
2200 Mission College Blvd.  
Santa Clara, CA 95054  
[chris.wilkerson@intel.com](mailto:chris.wilkerson@intel.com)

Onur Mutlu  
Carnegie Mellon University  
5000 Forbes Ave.  
Pittsburgh, PA 15213  
[onur@cmu.edu](mailto:onur@cmu.edu)

# More on DRAM Refresh (III)

---

- 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

# More on DRAM Refresh (IV)

---

- 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 { <i>moin, dhkim, pnair6</i> }@ece.gatech.edu			<sup>‡</sup> Carnegie Mellon University { <i>samirakhan, onur</i> }@cmu.edu	

# More on DRAM Refresh (V)

---

- Samira Khan, Donghyuk Lee, and Onur Mutlu,  
**"PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM"**  
*Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Toulouse, France, June 2016.  
[[Slides \(pptx\)](#)] [[pdf](#)]

## 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

# More on DRAM Refresh (VI)

---

- Samira Khan, Chris Wilkerson, Zhe Wang, Alaa R. Alameldeen, Donghyuk Lee, and Onur Mutlu,  
**"Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content"**  
*Proceedings of the 50th International Symposium on Microarchitecture (MICRO)*, Boston, MA, USA, October 2017.  
[\[Slides \(pptx\) \(pdf\)\]](#) [\[Lightning Session Slides \(pptx\) \(pdf\)\]](#) [\[Poster \(pptx\) \(pdf\)\]](#)

## Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content

Samira Khan<sup>\*</sup> Chris Wilkerson<sup>†</sup> Zhe Wang<sup>†</sup> Alaa R. Alameldeen<sup>†</sup> Donghyuk Lee<sup>‡</sup> Onur Mutlu<sup>\*</sup>  
<sup>\*</sup>University of Virginia    <sup>†</sup>Intel Labs    <sup>‡</sup>Nvidia Research    <sup>\*</sup>ETH Zürich

# More on DRAM Refresh (VII)

---

- 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.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Lightning Session Slides \(pptx\)](#)] [[pdf](#)]
- First experimental analysis of (mobile) LPDDR4 chips
- Analyzes the complex tradeoff space of retention time profiling
- Idea: enable fast and robust profiling at higher refresh intervals & temperatures

## 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>  
<sup>§</sup>ETH Zürich   <sup>‡</sup>Carnegie Mellon University



# More on DRAM Refresh (VIII)

---

- Minesh Patel, Jeremie S. Kim, Hasan Hassan, and Onur Mutlu,  
**"Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices"**  
*Proceedings of the 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Portland, OR, USA, June 2019.  
[[Source Code for EINSim, the Error Inference Simulator](#)]  
***Best paper session.***

## Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices

Minesh Patel<sup>†</sup>   Jeremie S. Kim<sup>‡†</sup>   Hasan Hassan<sup>†</sup>   Onur Mutlu<sup>†‡</sup>

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

# Memory Scaling Issues **Are** Real

---

- Onur Mutlu and Jeremie Kim,  
**"RowHammer: A Retrospective"**  
*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) Special Issue on Top Picks in Hardware and Embedded Security*, 2019.  
[[Preliminary arXiv version](#)]

## RowHammer: A Retrospective

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

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

---

One can  
predictably induce errors  
in most DRAM memory chips

# The Story of RowHammer

- One can **predictably induce bit flips** in commodity DRAM chips
  - >80% of the tested DRAM chips are vulnerable
- First example of how 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



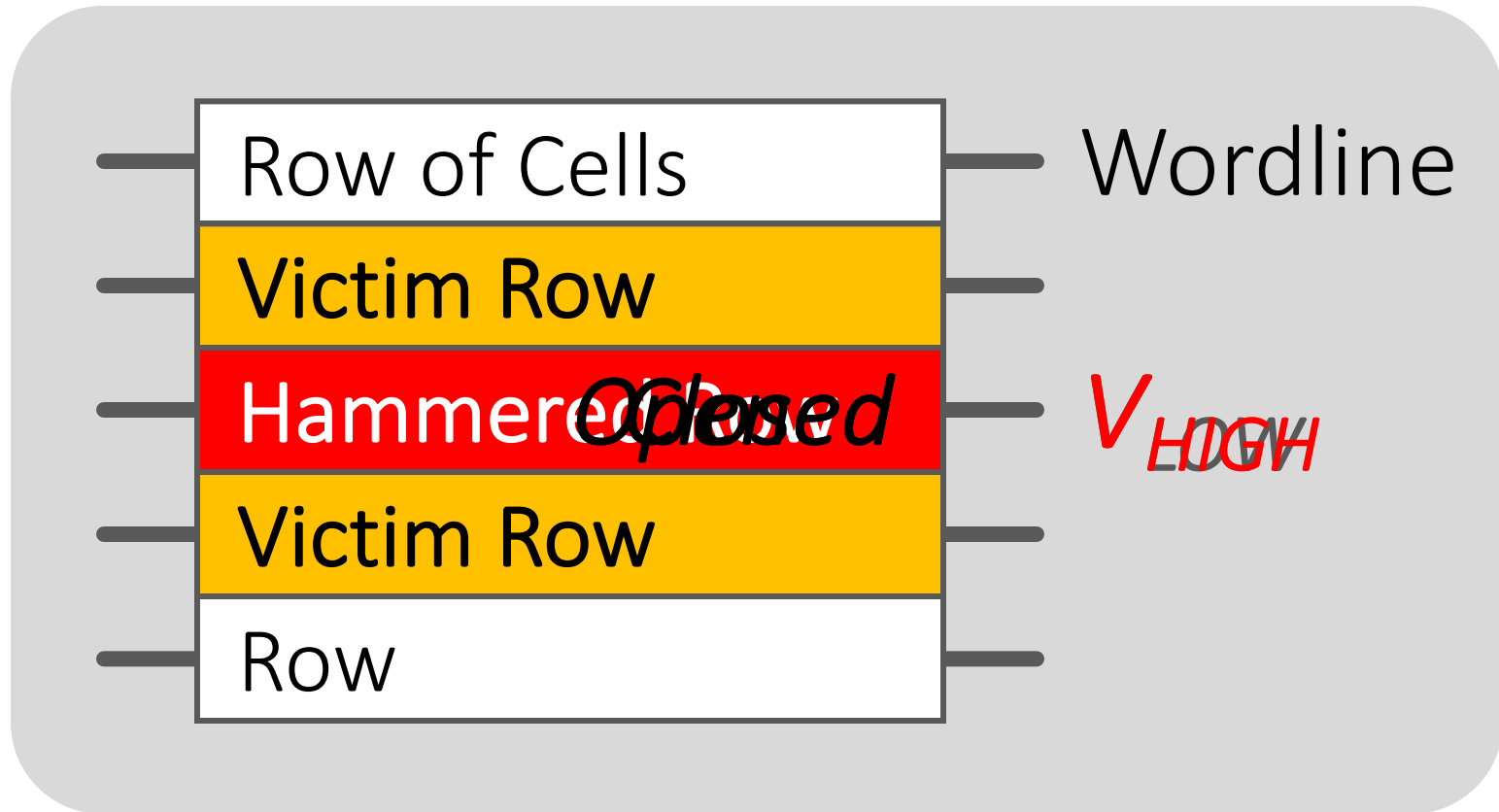
SHARE  
18276



TWEET

# FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS

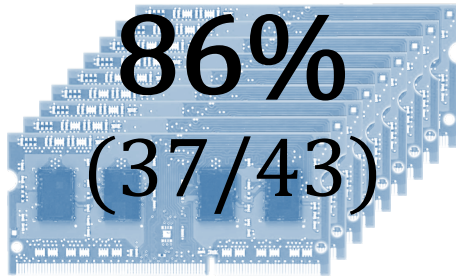
# Modern DRAM is Prone to Disturbance Errors



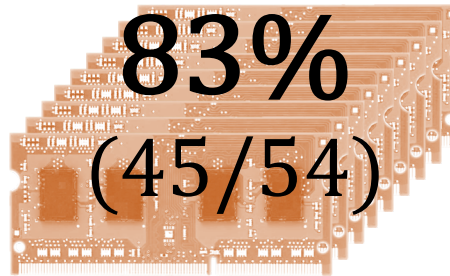
Repeatedly reading a row enough times (before memory gets refreshed) induces **disturbance errors** in adjacent rows in **most real DRAM chips you can buy today**

# Most DRAM Modules Are Vulnerable

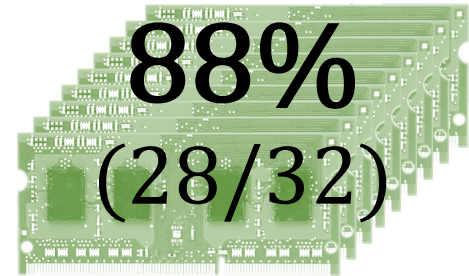
A company



B company



C company



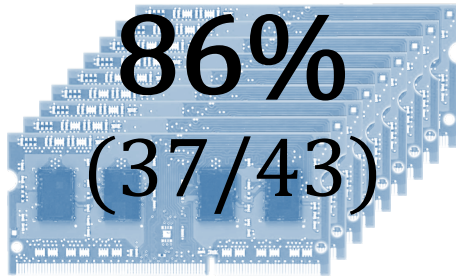
Up to  
 $1.0 \times 10^7$   
errors

Up to  
 $2.7 \times 10^6$   
errors

Up to  
 $3.3 \times 10^5$   
errors

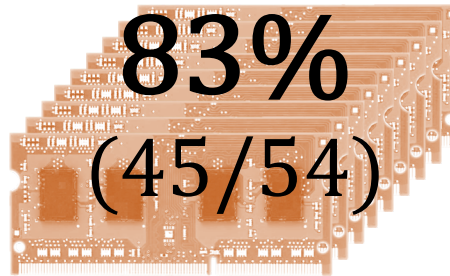
# Most DRAM Modules Are Vulnerable

A company



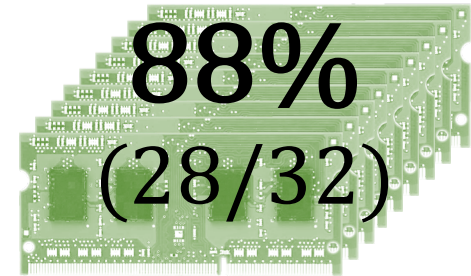
Up to  
 $1.0 \times 10^7$   
errors

B company



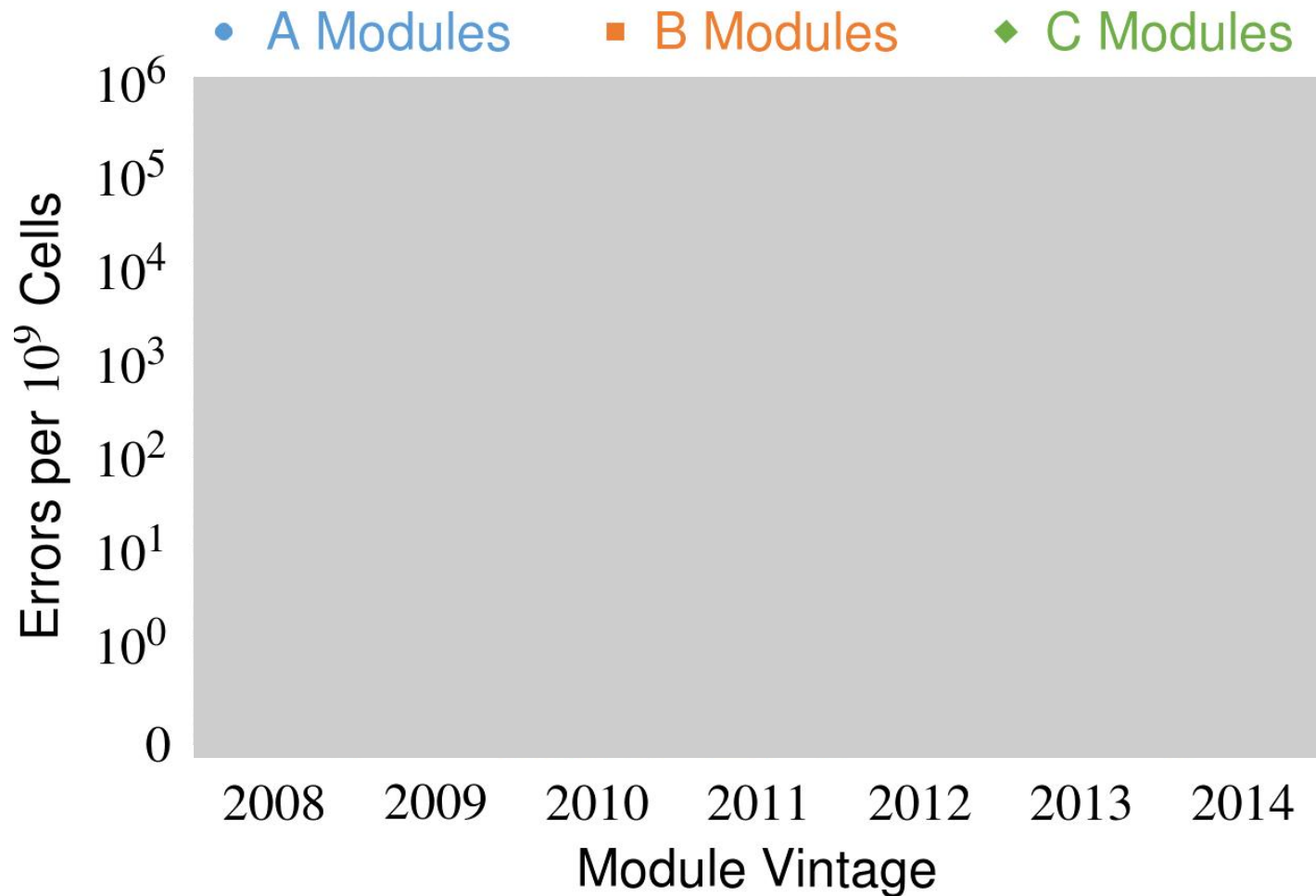
Up to  
 $2.7 \times 10^6$   
errors

C company



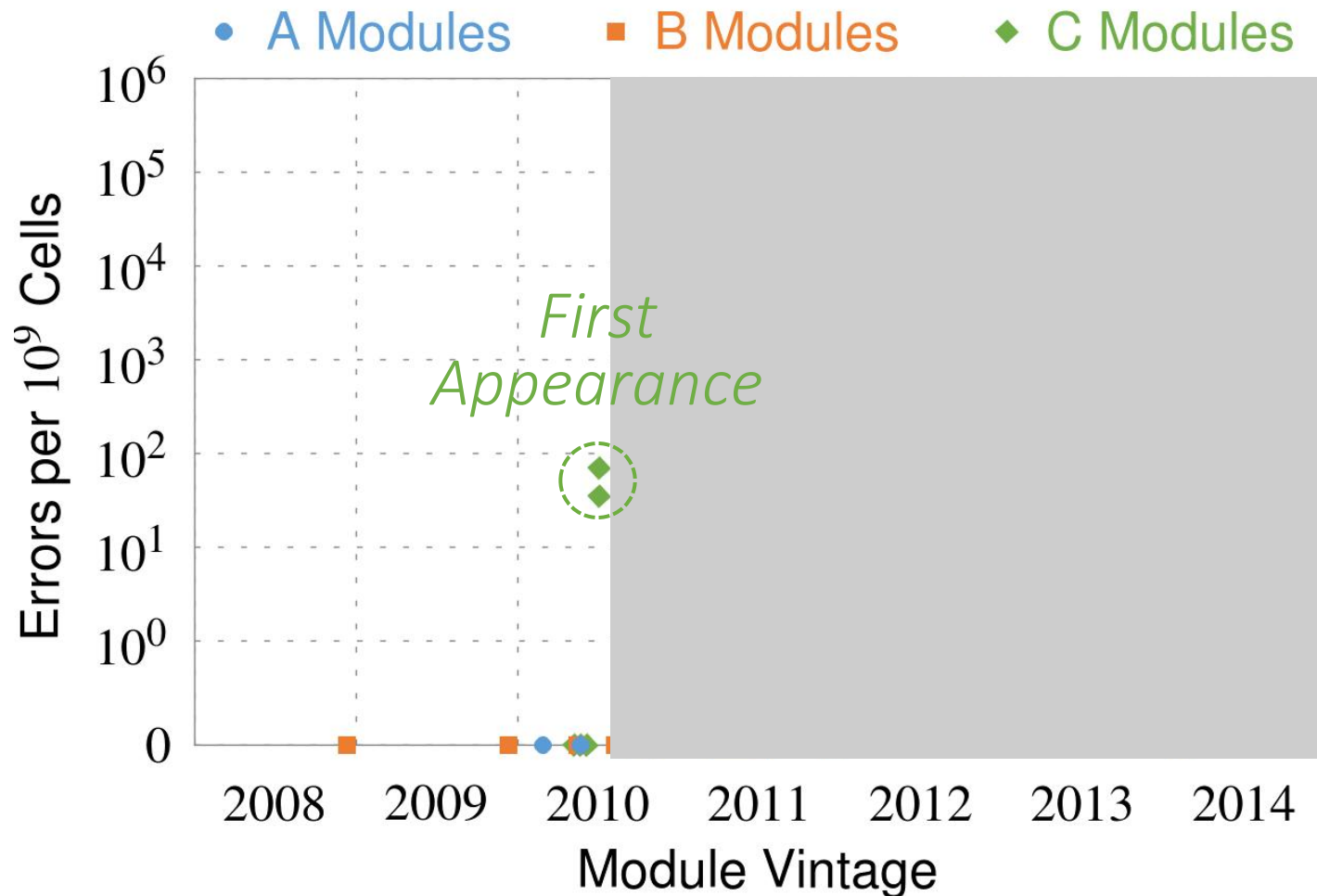
Up to  
 $3.3 \times 10^5$   
errors

# Recent DRAM Is More Vulnerable

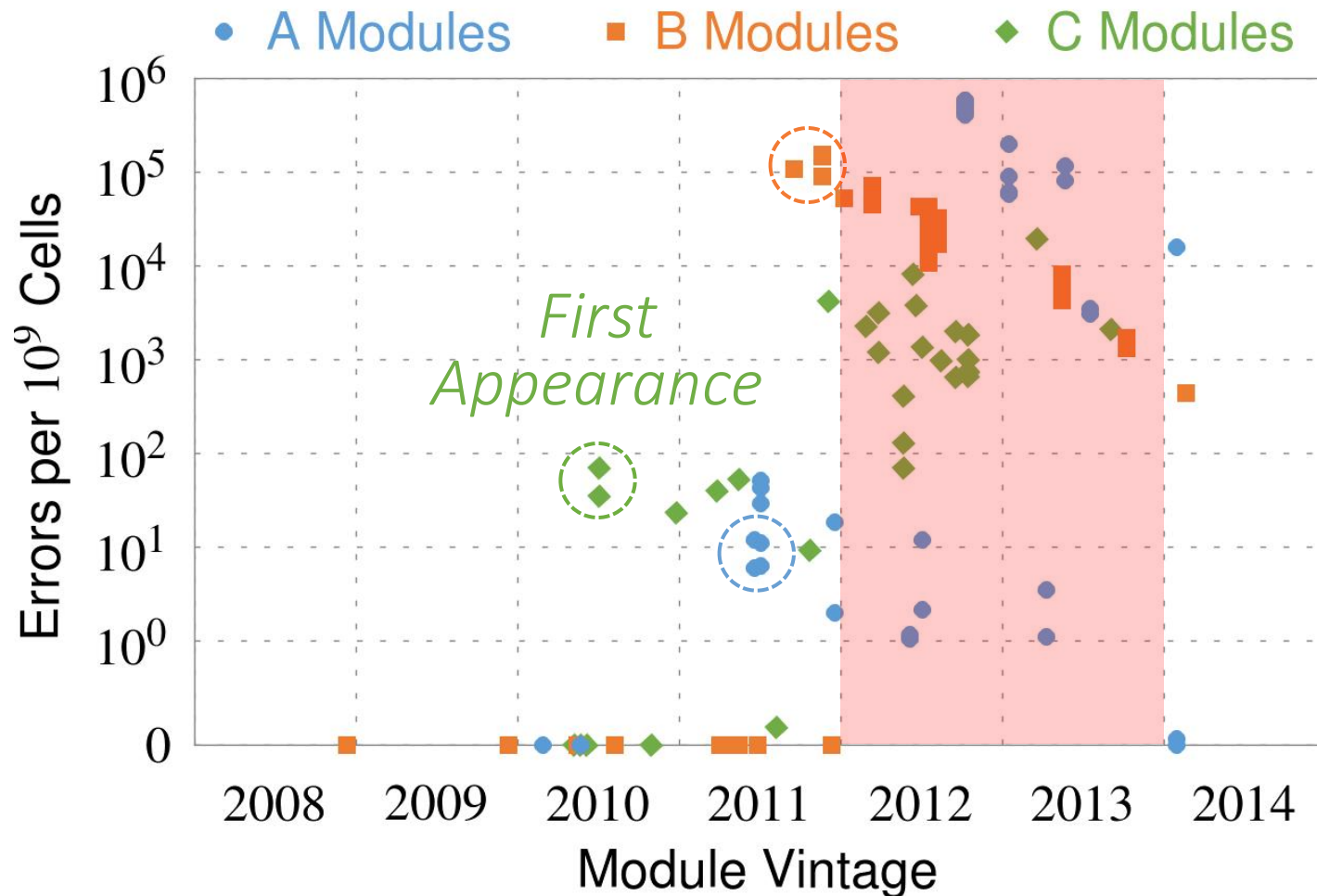




# Recent DRAM Is More Vulnerable

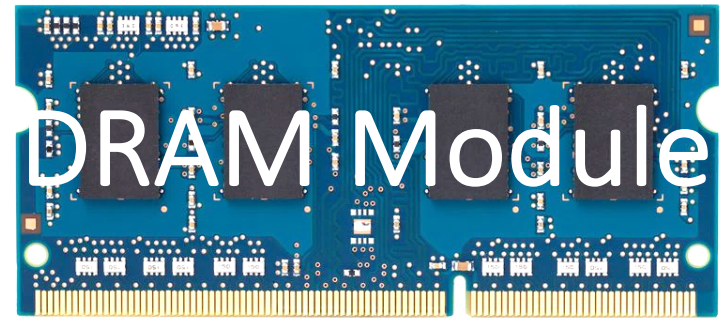
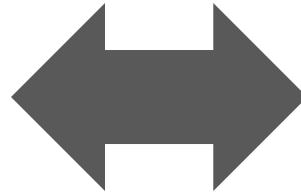
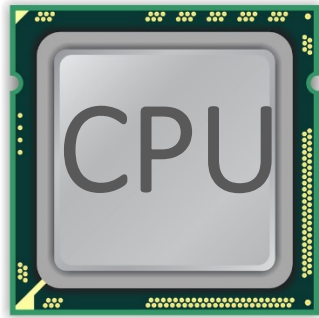


# Recent DRAM Is More Vulnerable

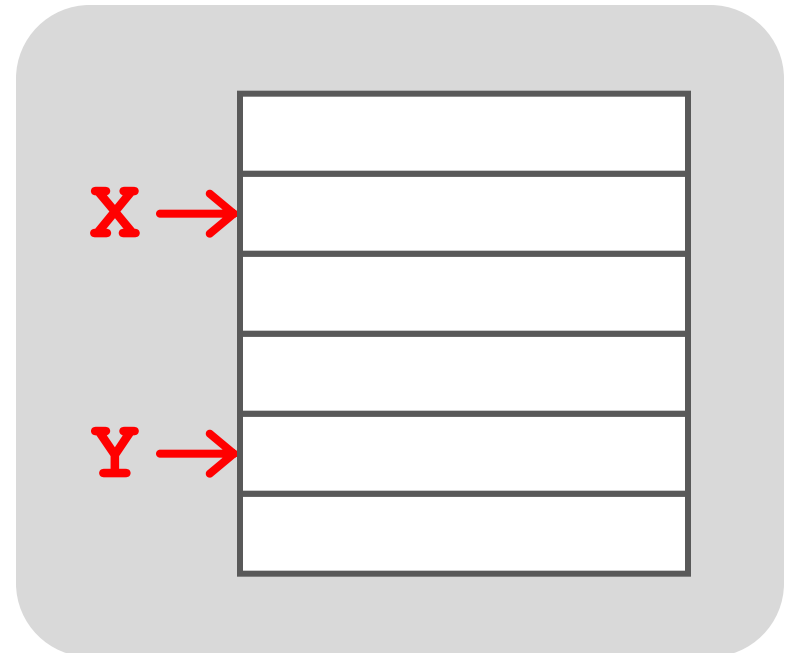


*All modules from 2012-2013 are vulnerable*

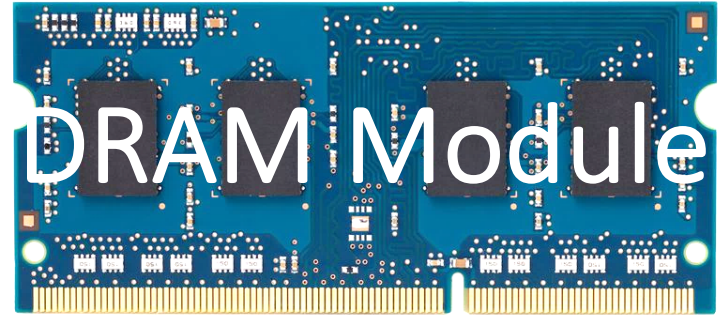
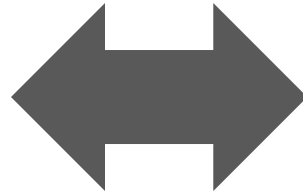
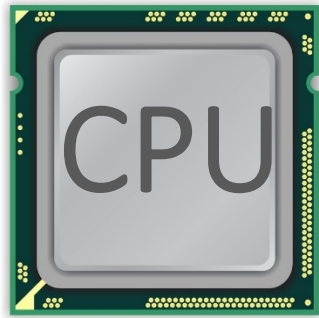
# A Simple Program Can Induce Many Errors



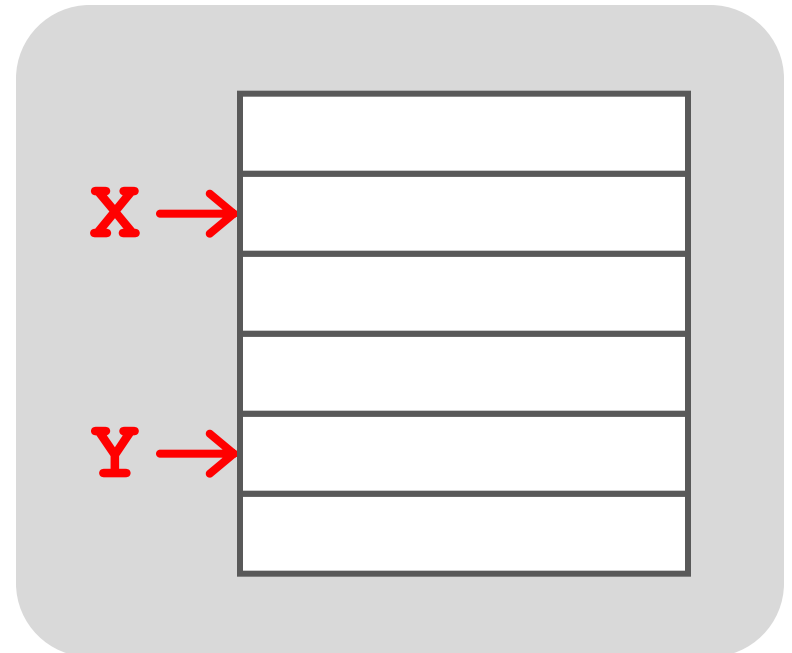
```
loop:  
  mov  (X), %eax  
  mov  (Y), %ebx  
  clflush (X)  
  clflush (Y)  
  mfence  
  jmp  loop
```



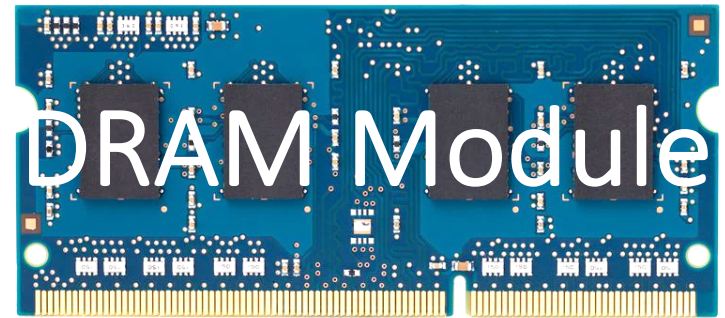
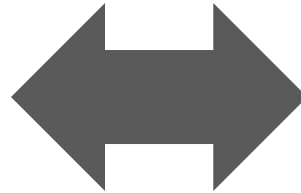
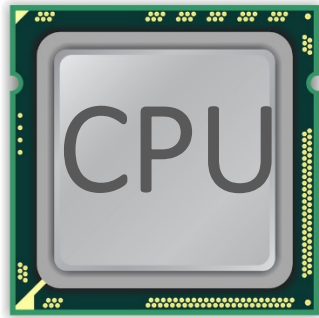
# A Simple Program Can Induce Many Errors



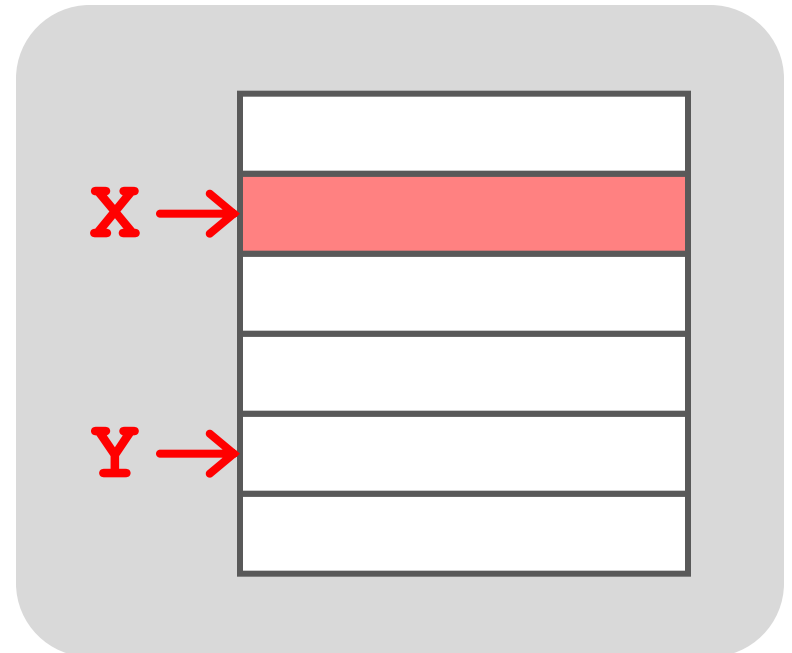
1. Avoid *cache hits*
  - Flush **X** from cache
2. Avoid *row hits* to **X**
  - Read **Y** in another row



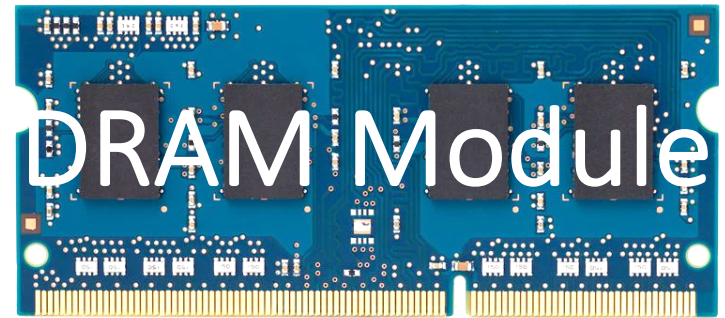
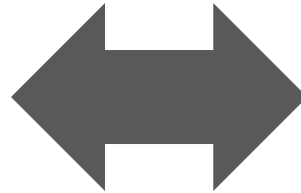
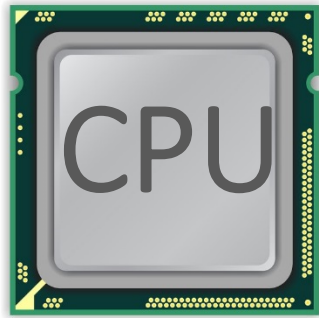
# A Simple Program Can Induce Many Errors



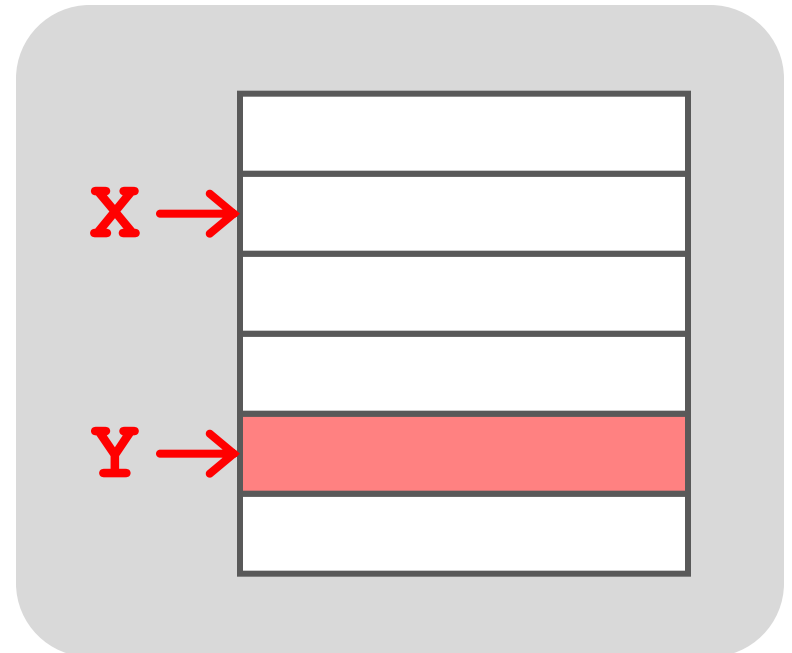
```
loop:  
  mov  (X), %eax  
  mov  (Y), %ebx  
  clflush (X)  
  clflush (Y)  
  mfence  
  jmp  loop
```



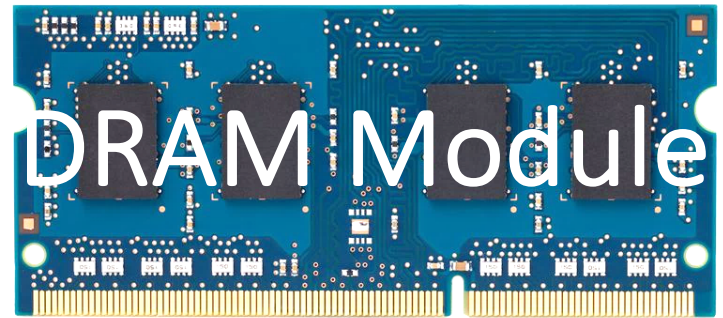
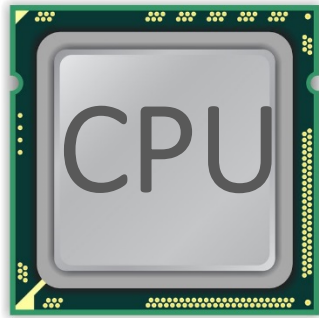
# A Simple Program Can Induce Many Errors



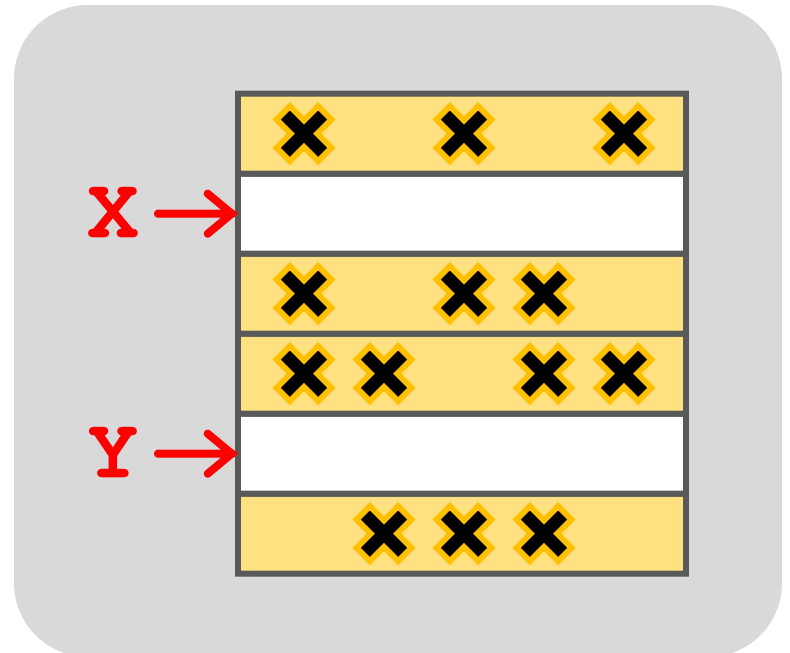
```
loop:  
  mov  (X), %eax  
  mov  (Y), %ebx  
  clflush (X)  
  clflush (Y)  
  mfence  
  jmp  loop
```



# 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



Rowhammer

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

# More Security Implications (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<sup>104</sup>

# More Security Implications (III)

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



TECHNICA

BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

"GRAND PWINING 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)

**ars** TECHNICA

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?

---



# Memory Scaling Issues **Are** Real

---

- 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

# More on RowHammer

---

- Onur Mutlu and Jeremie Kim,  
**"RowHammer: A Retrospective"**  
*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) Special Issue on Top Picks in Hardware and Embedded Security*, 2019.  
[[Preliminary arXiv version](#)]

## RowHammer: A Retrospective

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

# 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

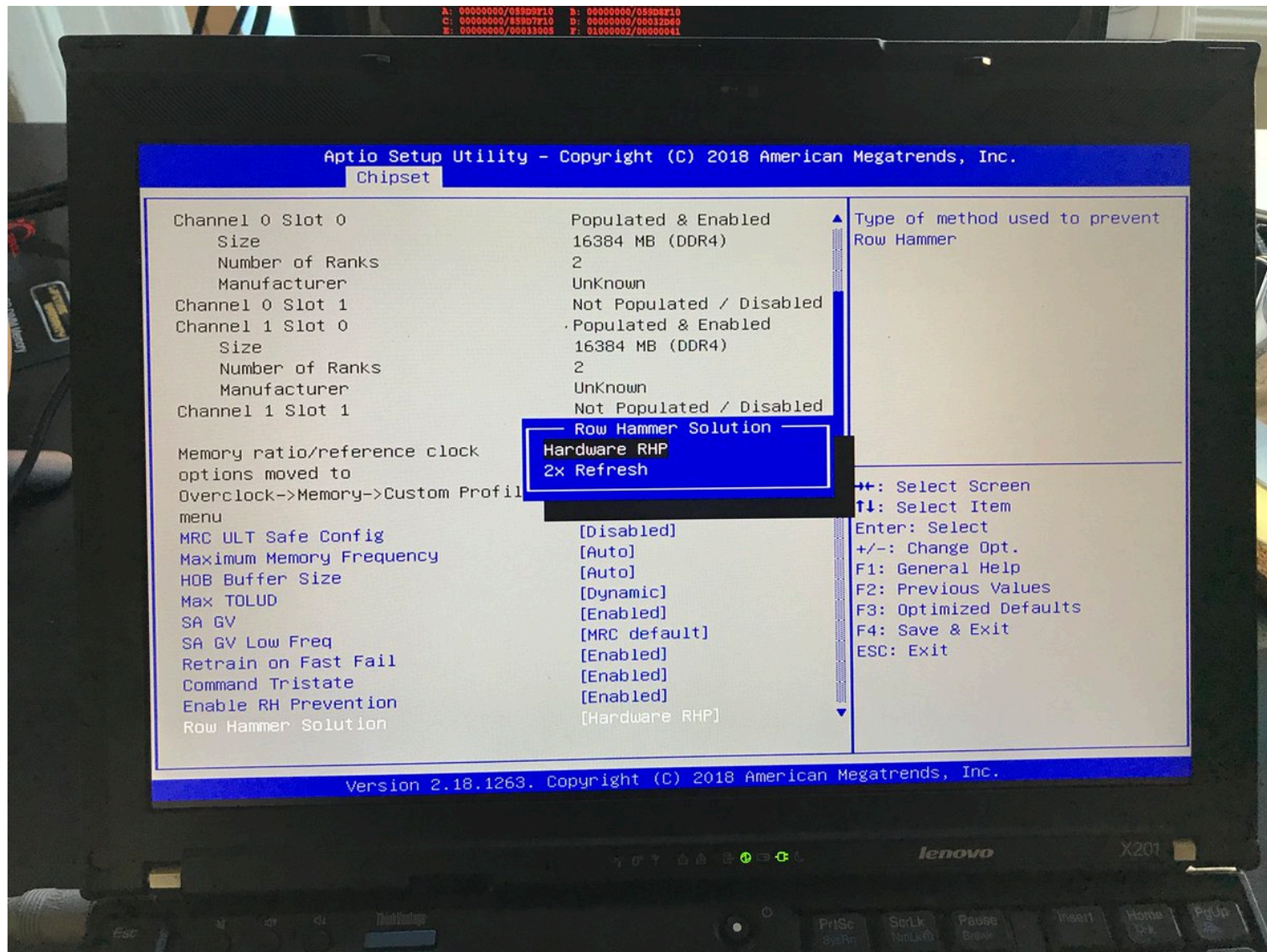
- *PARA refreshes rows infrequently*
  - Low power
  - Low performance-overhead
    - Average slowdown: **0.20%** (for 29 benchmarks)
    - Maximum slowdown: **0.75%**
- *PARA is stateless*
  - Low cost
  - Low complexity
- *PARA is an effective and low-overhead solution to prevent disturbance errors*

# Requirements for PARA

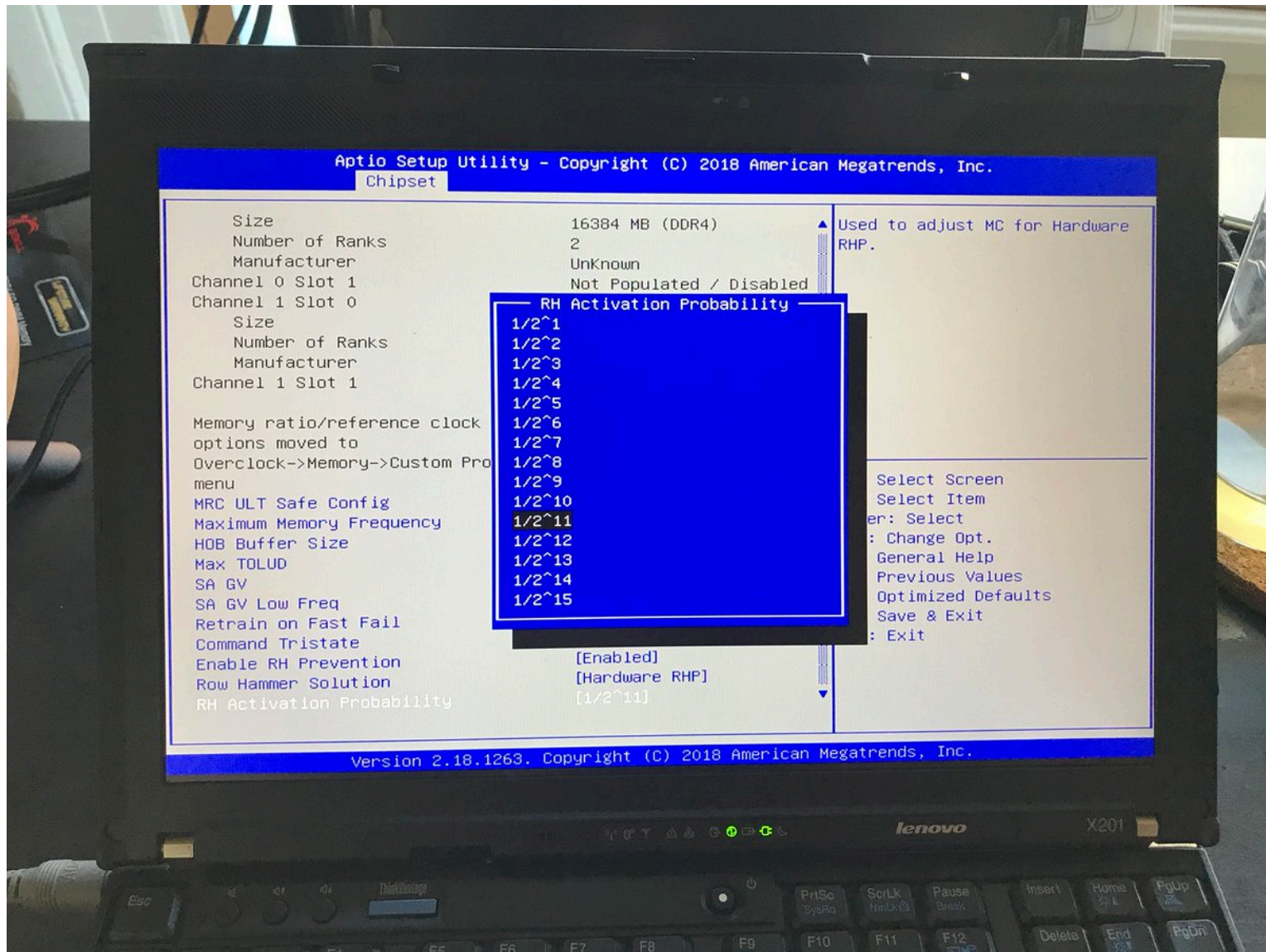
- If implemented in **DRAM chip** (done today)
  - Enough slack in timing and refresh parameters
  - Plenty of slack today:
    - Lee et al., “**Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common Case**,” HPCA 2015.
    - Chang et al., “**Understanding Latency Variation in Modern DRAM Chips**,” 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.
    - Ghose et al., “**What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study**,” SIGMETRICS 2018.
- If implemented in **memory controller**
  - Better coordination between memory controller and DRAM
  - Memory controller should know which rows are physically adjacent



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

# A RowHammer Retrospective

---

- Onur Mutlu and Jeremie Kim,  
**"RowHammer: A Retrospective"**  
*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) Special Issue on Top Picks in Hardware and Embedded Security*, 2019.  
[[Preliminary arXiv version](#)]

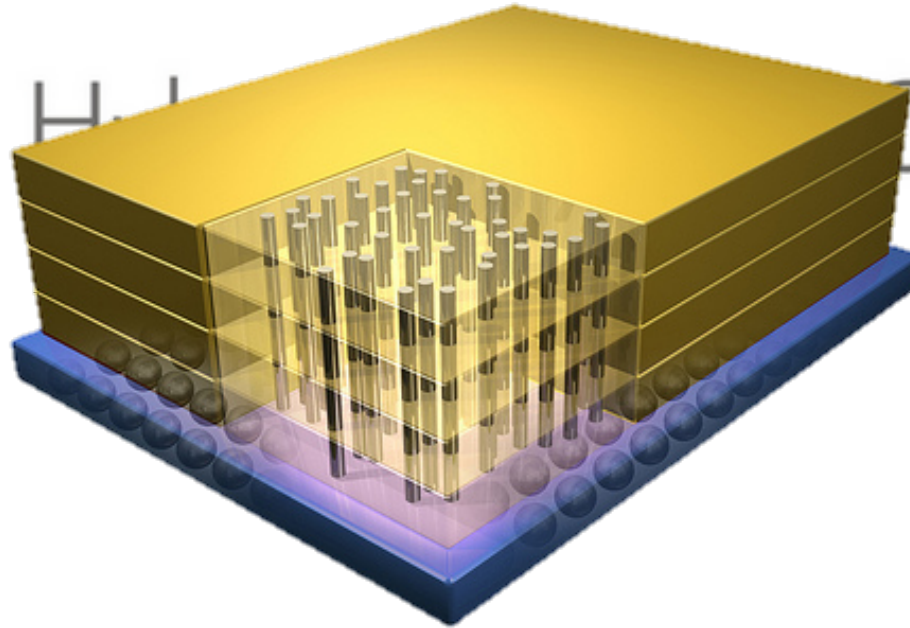
## RowHammer: A Retrospective

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

## Main Memory Needs Intelligent Controllers

# Why In-Memory Computation Today?

---



- 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

# Do We Want This?

---





# Or This?

---





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)

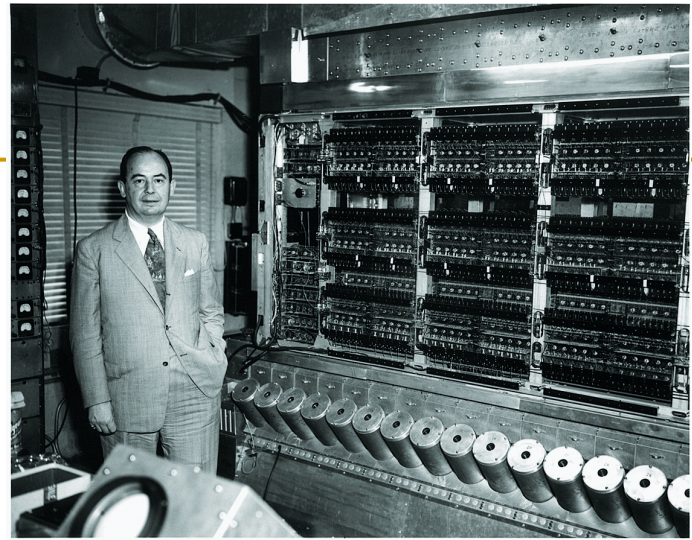
# The Problem

---

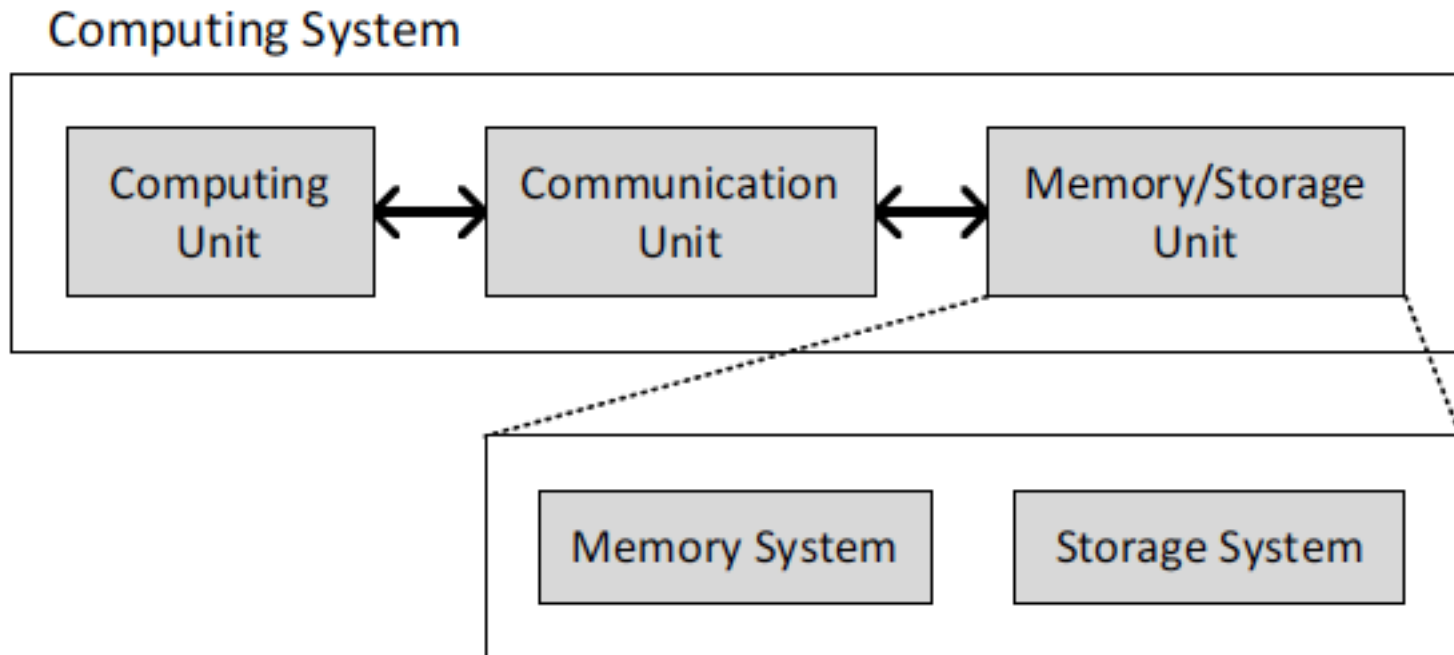
Processing of data  
is performed  
far away from the data

# A Computing System

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

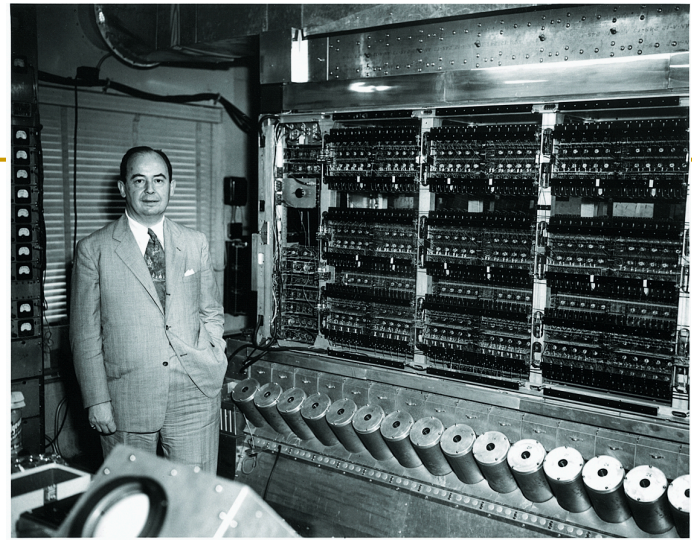


Burks, Goldstein, von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument," 1946.



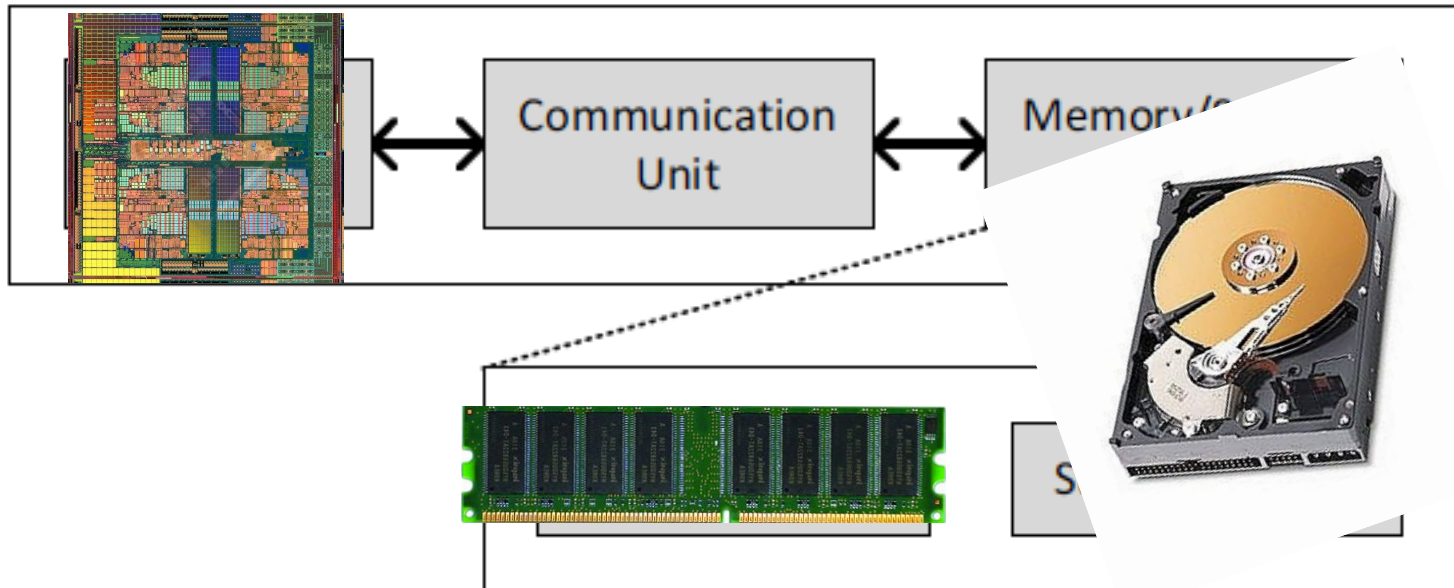
# A Computing System

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



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

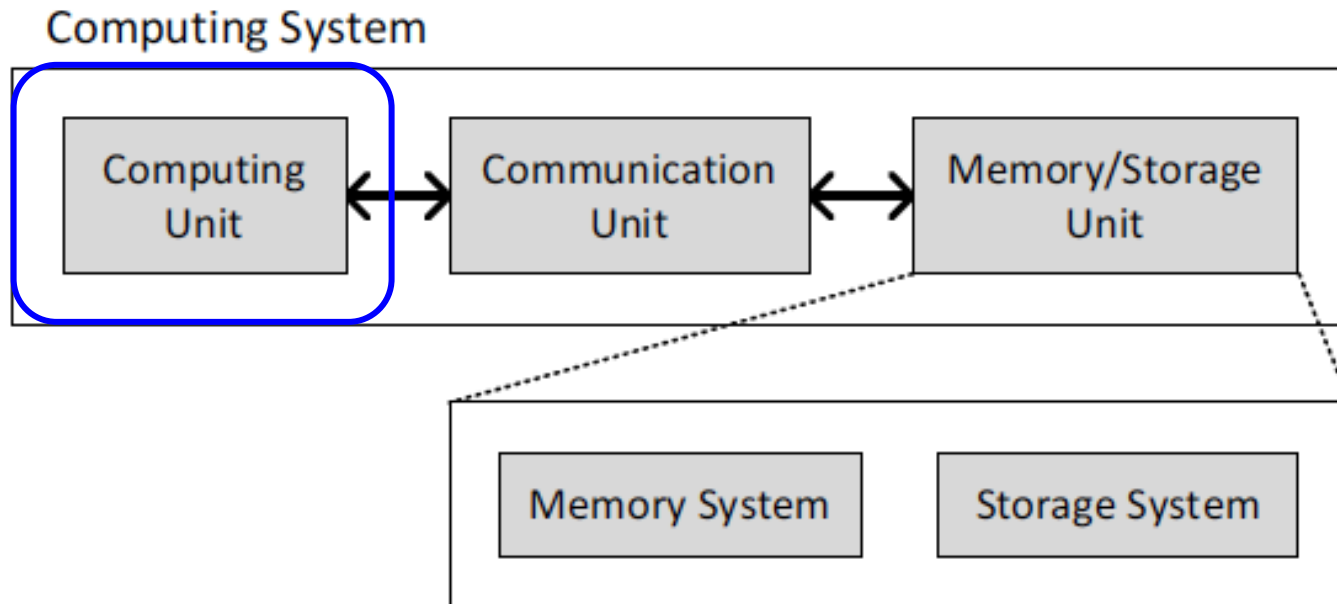
## Computing System



# Today's Computing Systems

---

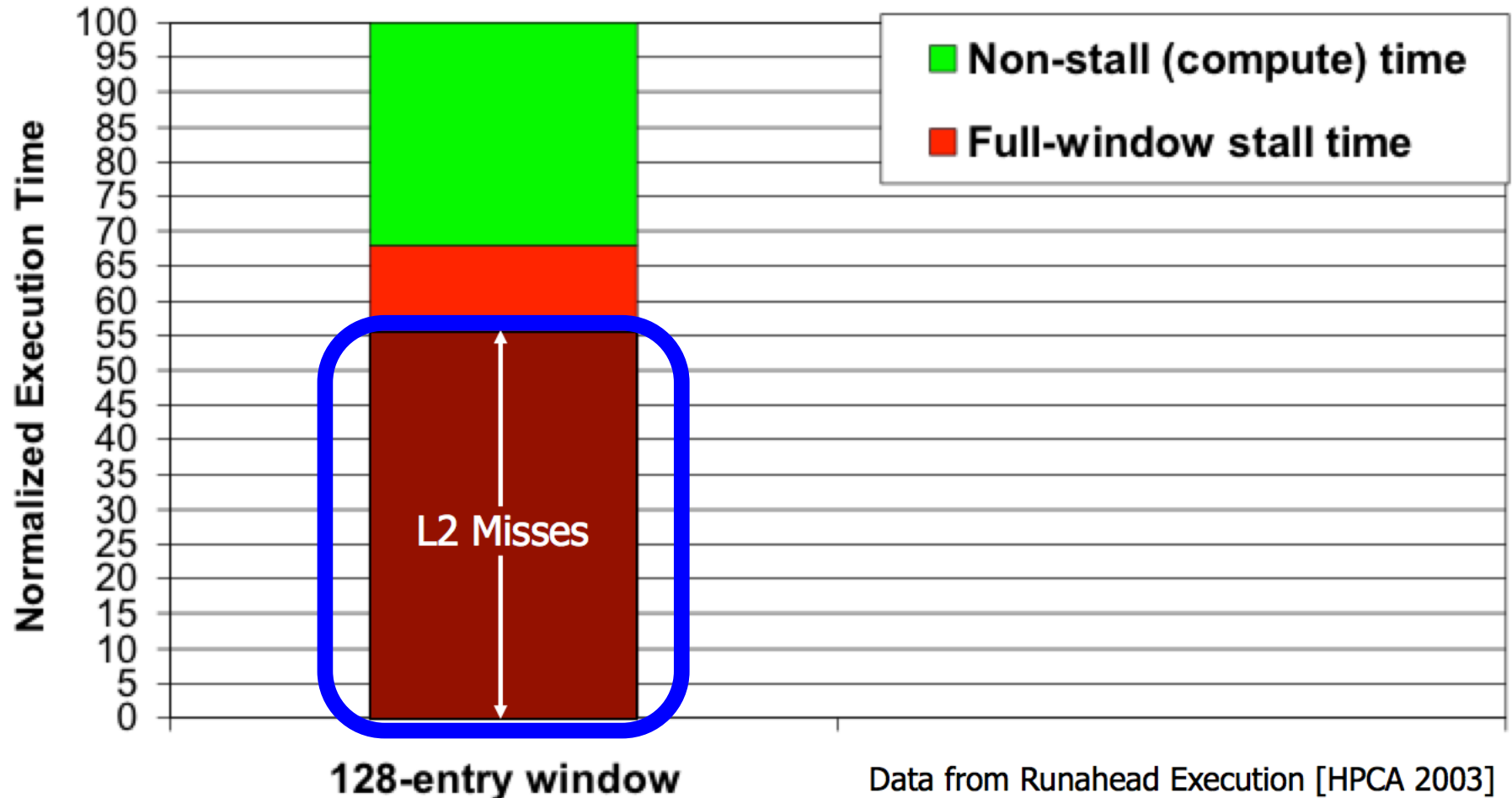
- Are overwhelmingly processor centric
- All data processed in the processor → at great system cost
- Processor is heavily optimized and is considered the master
- Data storage units are dumb and are largely unoptimized (except for some that are on the processor die)



# Yet ...

I expect that over the coming decade memory subsystem design will be the *only* important design issue for microprocessors.

- **“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 §

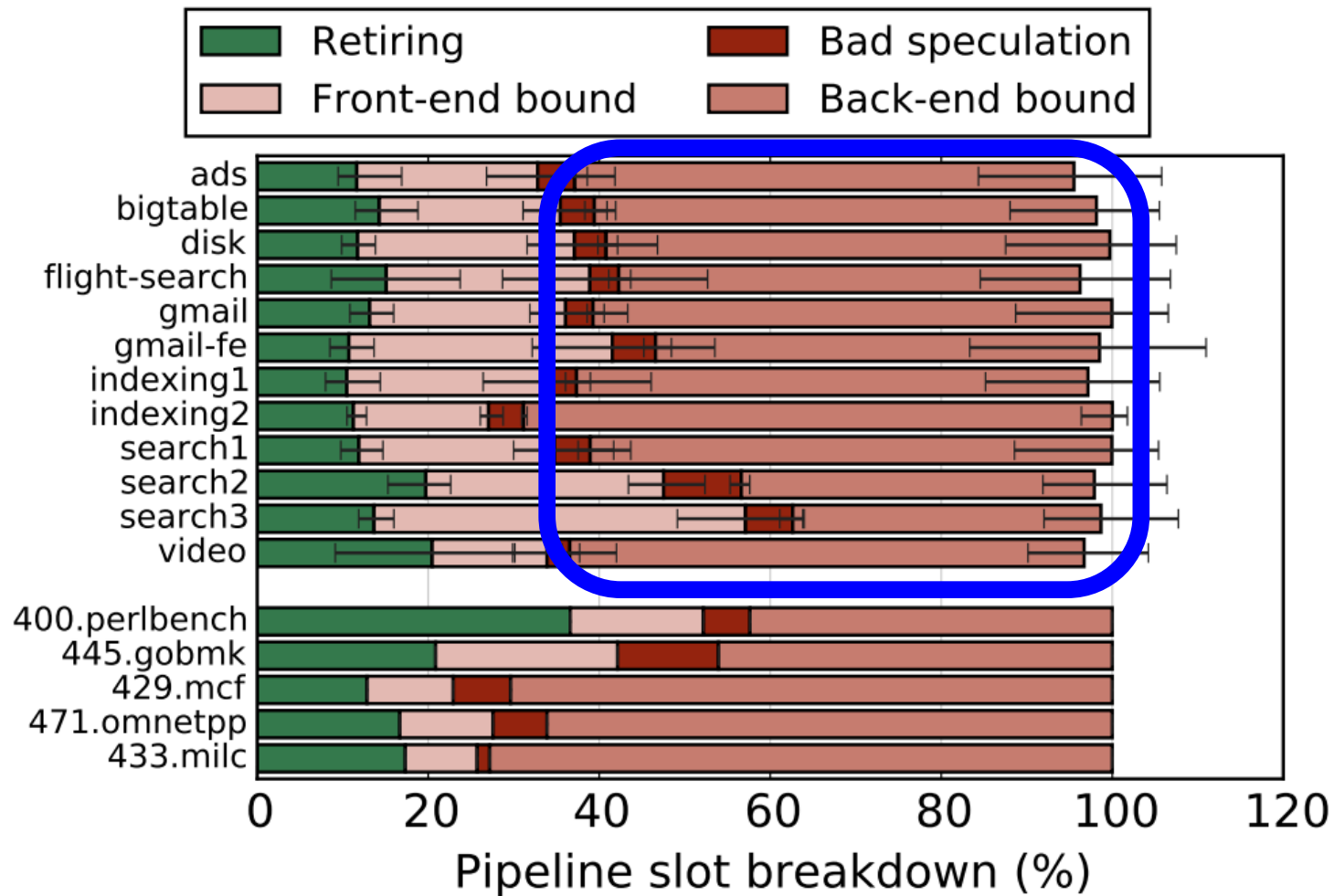
§ECE Department  
The University of Texas at Austin  
{onur,patt}@ece.utexas.edu

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

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

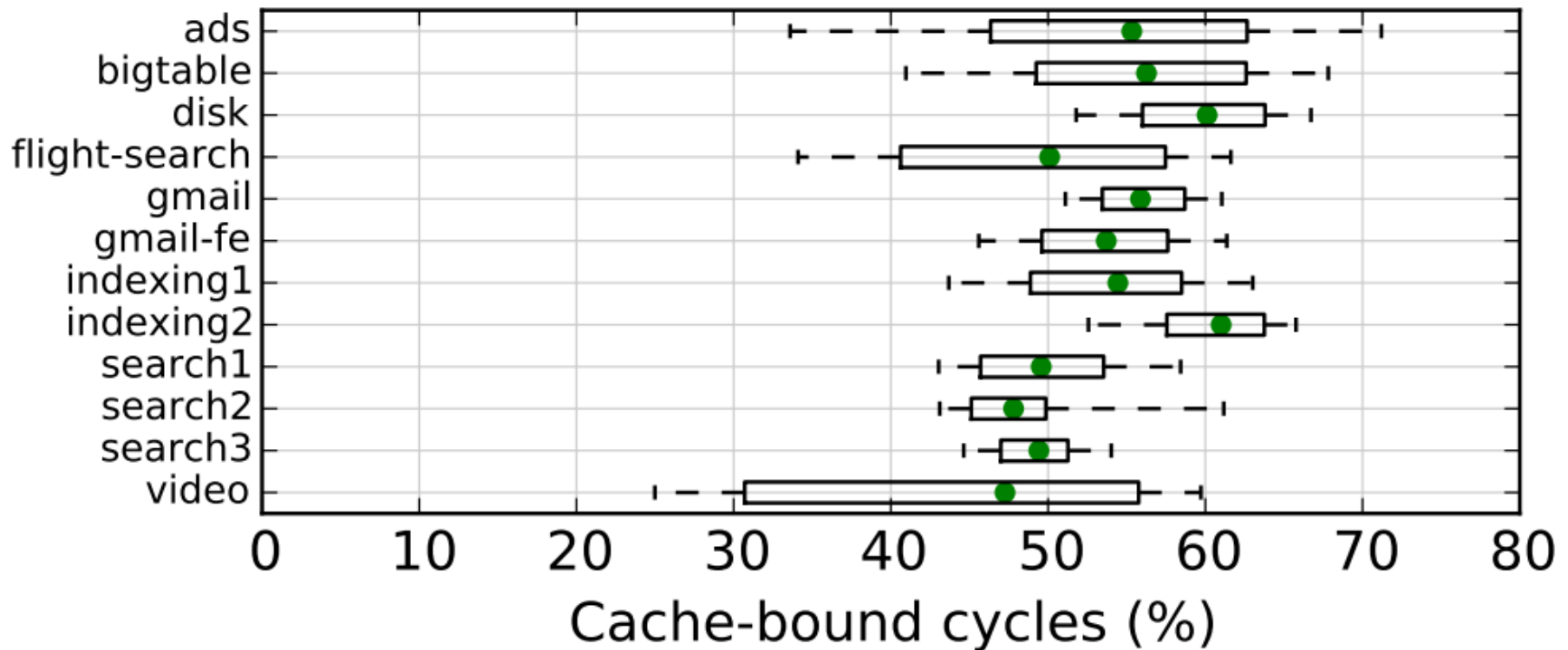
# The Performance Perspective (Today)

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



# The Performance Perspective (Today)

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



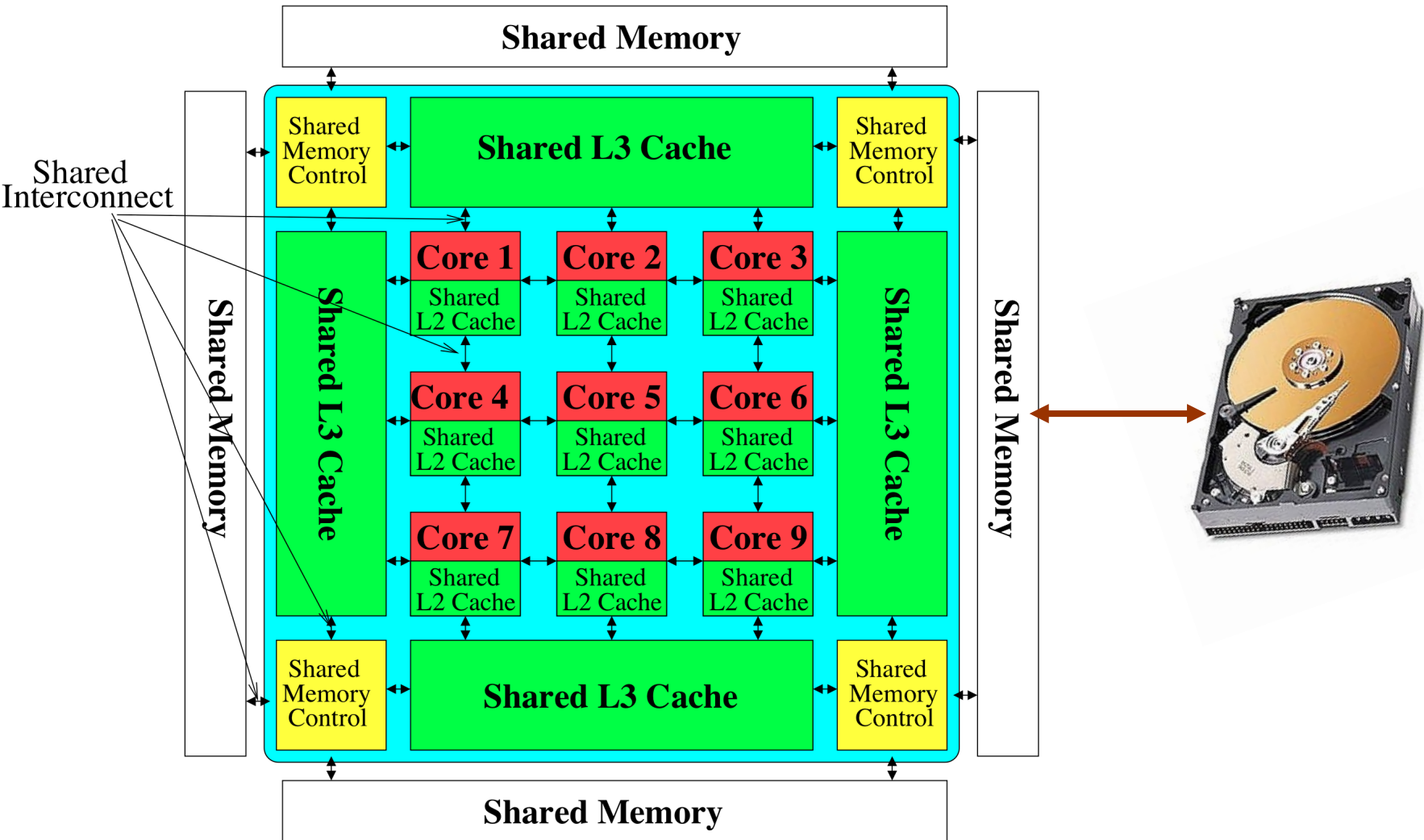
**Figure 11: Half of cycles are spent stalled on caches.**

# 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

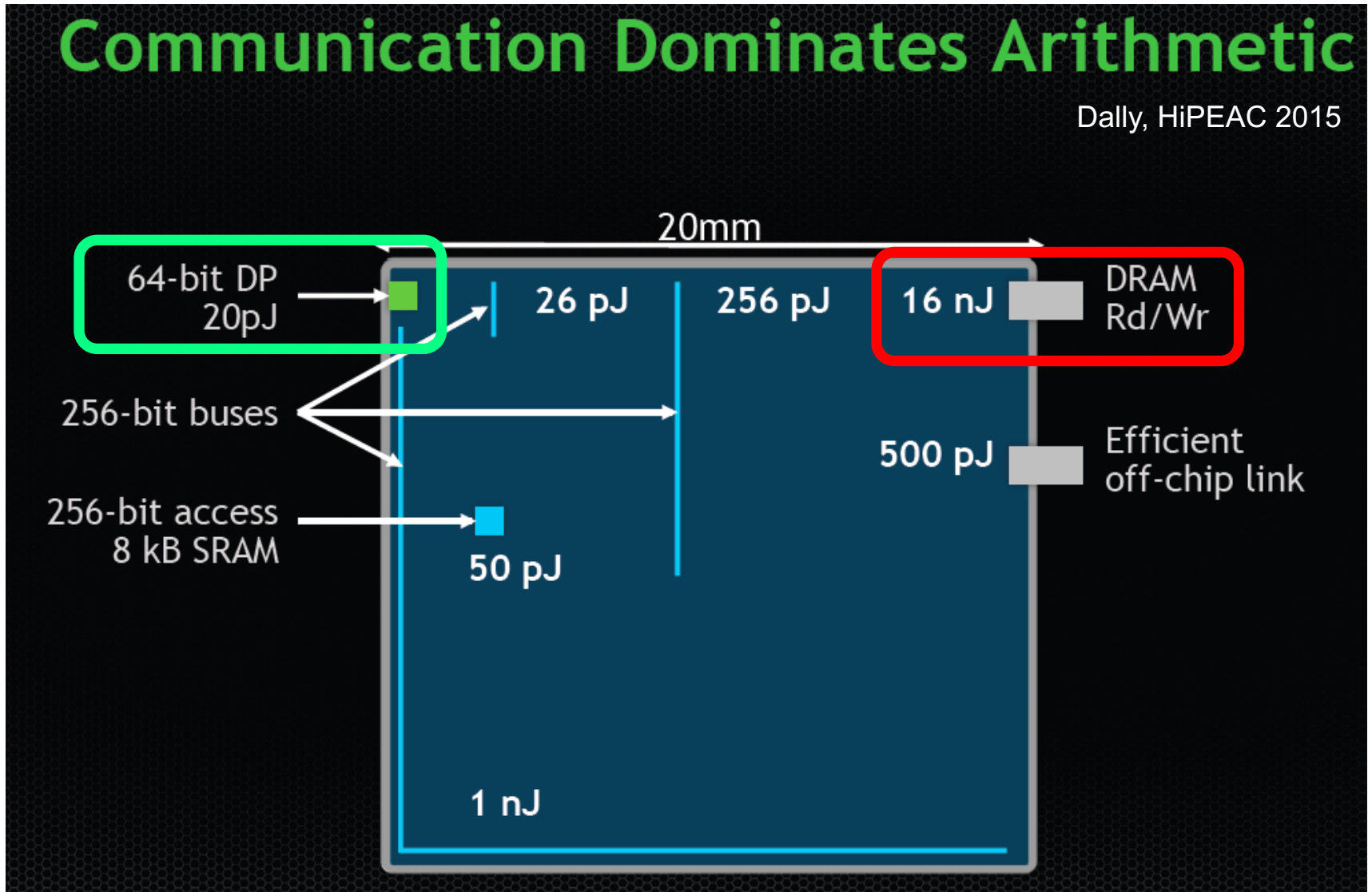


**Most of the system is dedicated to storing and moving data**

# The Energy Perspective

## Communication Dominates Arithmetic

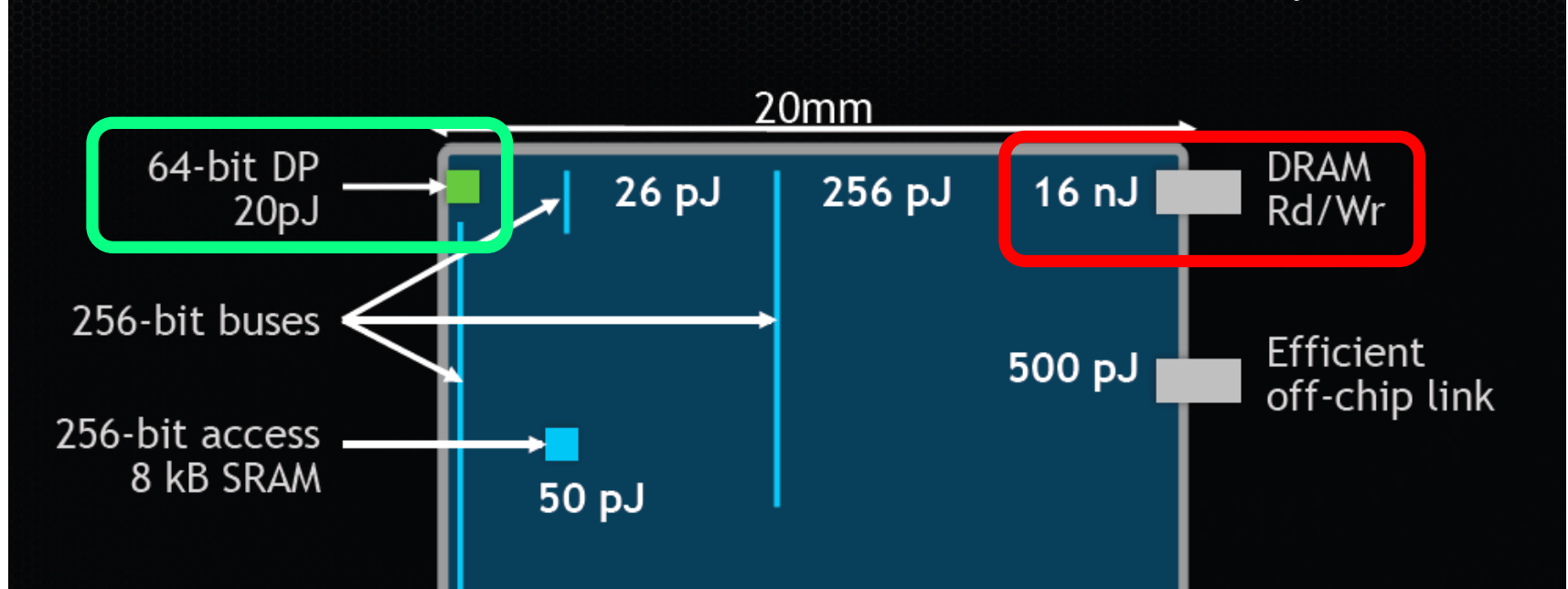
Dally, HiPEAC 2015



# Data Movement vs. Computation Energy

## Communication Dominates Arithmetic

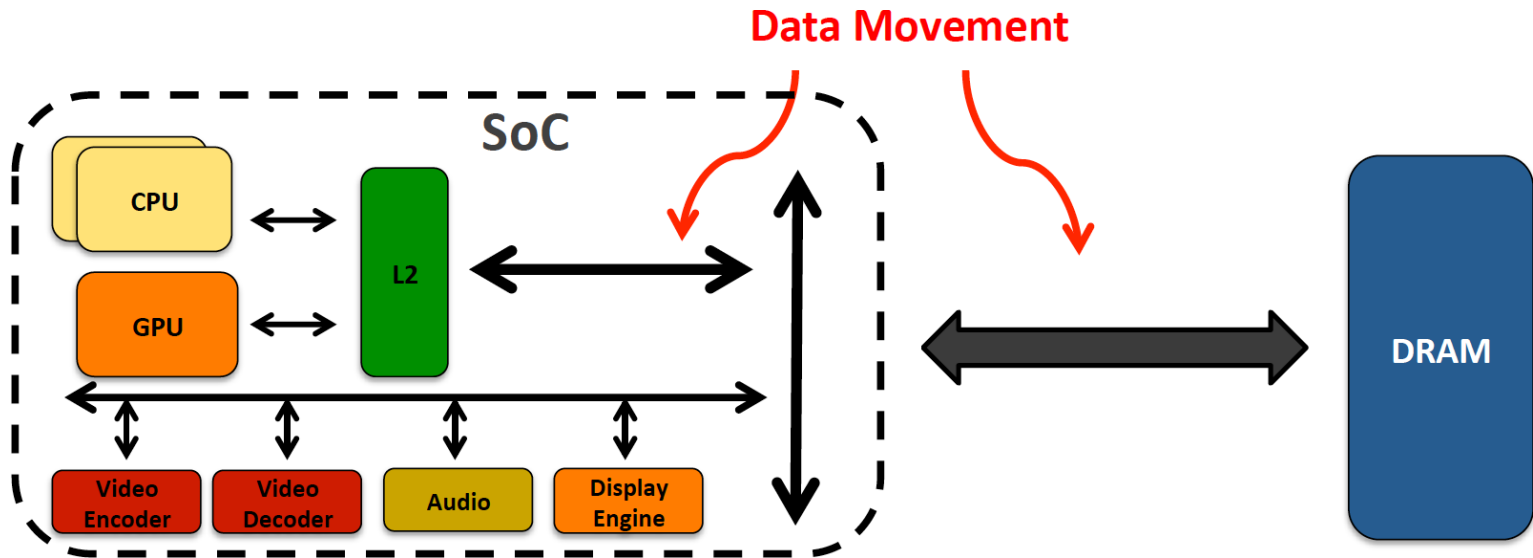
Dally, HiPEAC 2015



A memory access consumes  $\sim 100-1000\times$  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>

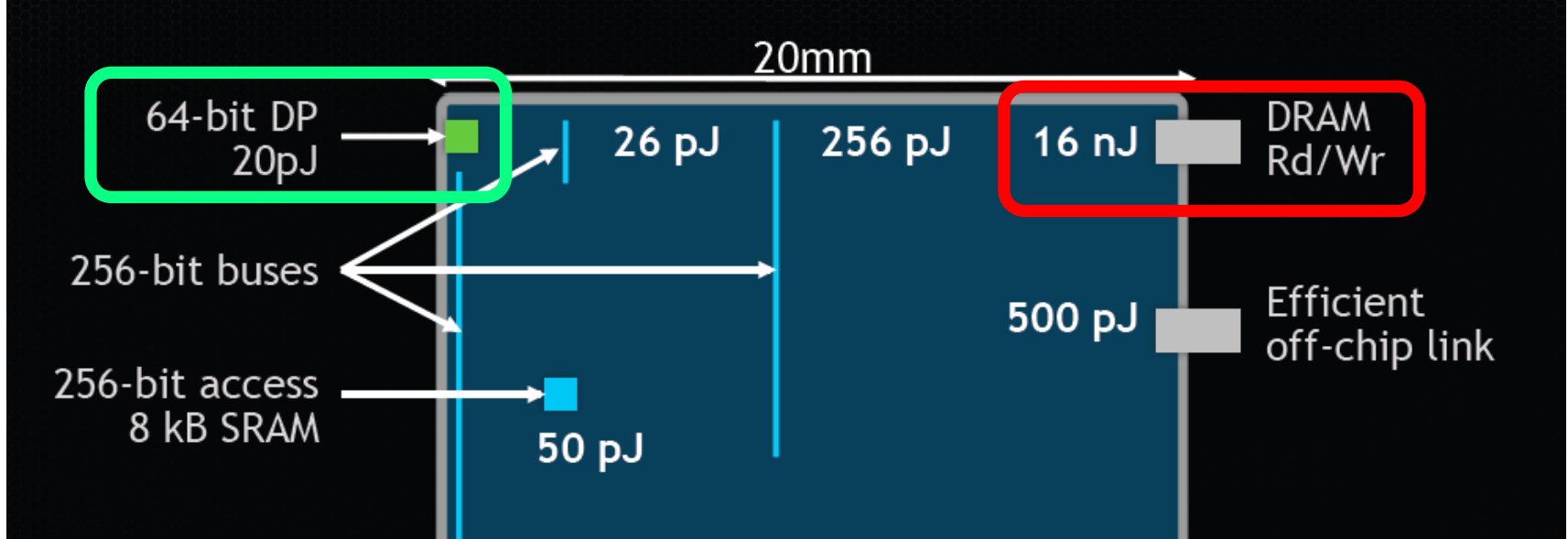
Parthasarathy Ranganathan<sup>3</sup>

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

# We Do Not Want to Move Data!

## Communication Dominates Arithmetic

Dally, HiPEAC 2015



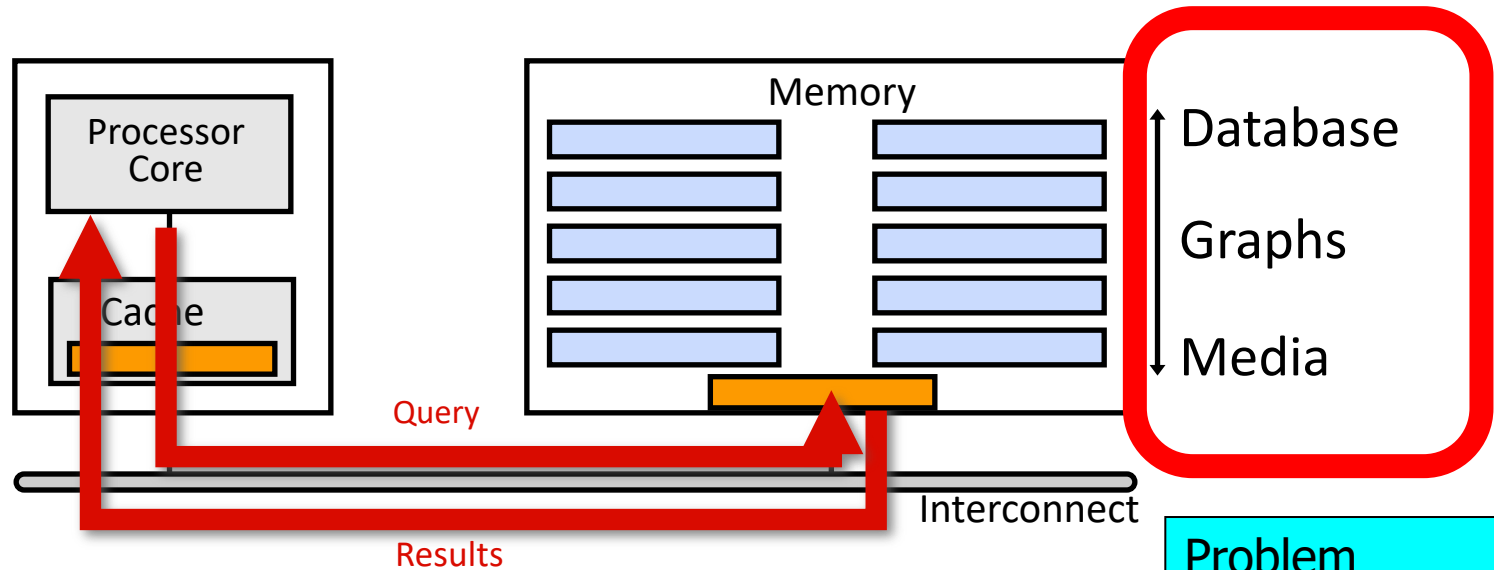
A memory access consumes  $\sim 100\text{-}1000\times$  the energy of a complex addition

# 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, compilers, languages?
  - ❑ algorithms and theoretical foundations?

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

# Processing in Memory: Two Approaches

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

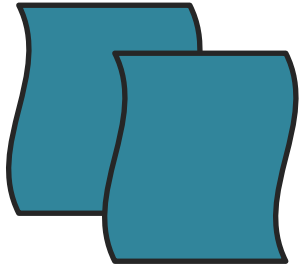
# Approach 1: Minimally Changing Memory

---

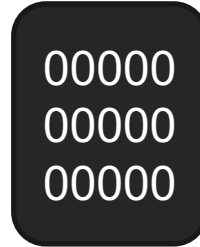
- DRAM has great capability to perform **bulk data movement and computation** internally with small changes
  - Can **exploit internal connectivity** to move data
  - Can **exploit analog computation capability**
  - ...
- Examples: RowClone, In-DRAM AND/OR, Gather/Scatter DRAM
  - RowClone: Fast and 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

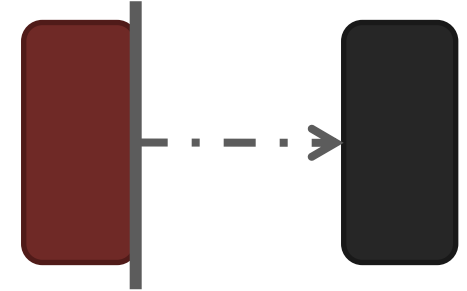
*memmove & memcpy: 5% cycles in Google's datacenter [Kanev+ ISCA'15]*



**Forking**



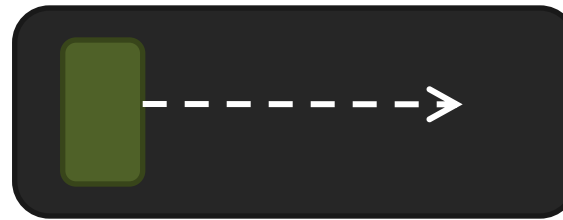
**Zero initialization  
(e.g., security)**



**Checkpointing**



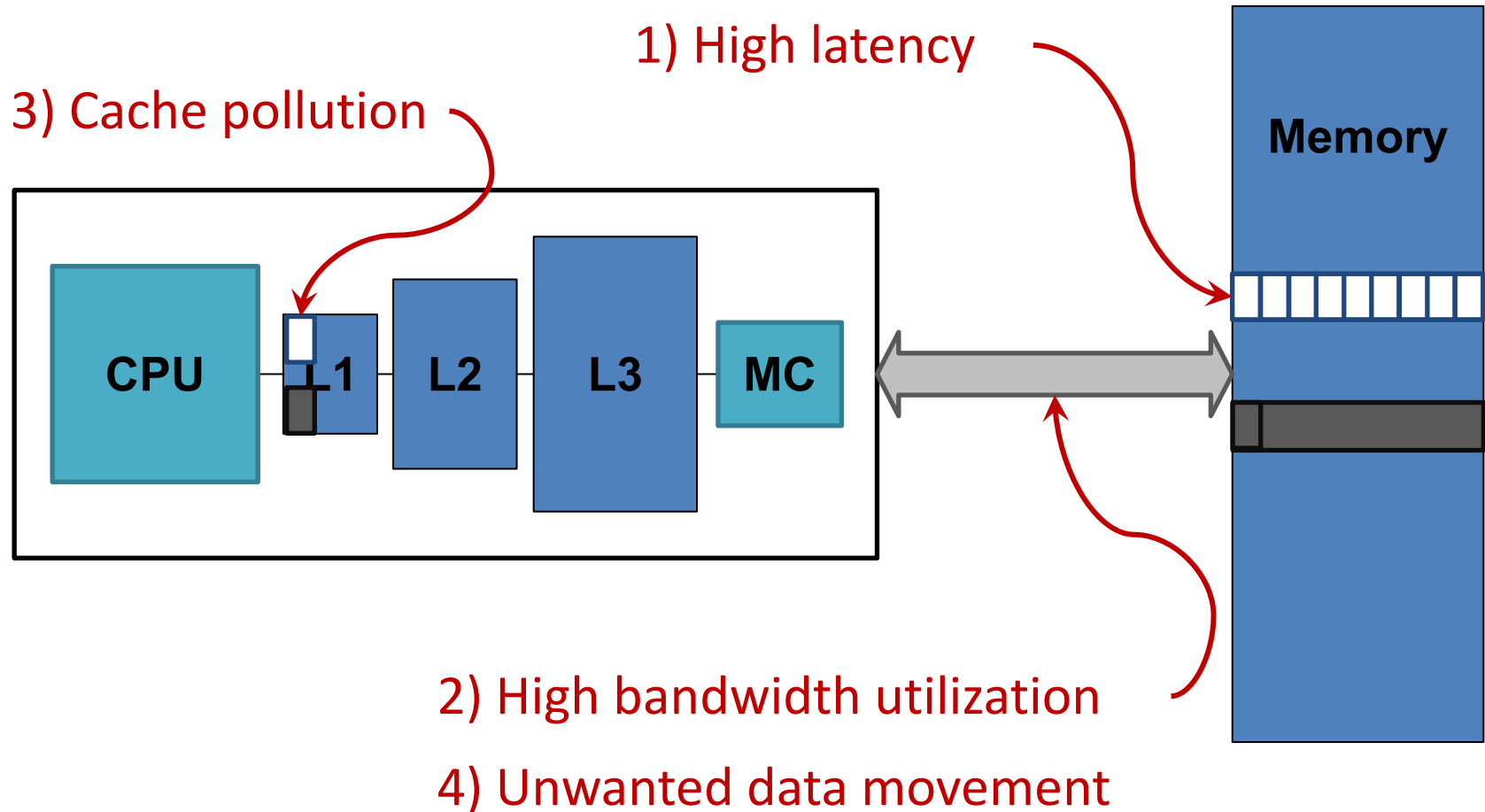
**VM Cloning  
Deduplication**



**Page Migration**

...  
**Many more**

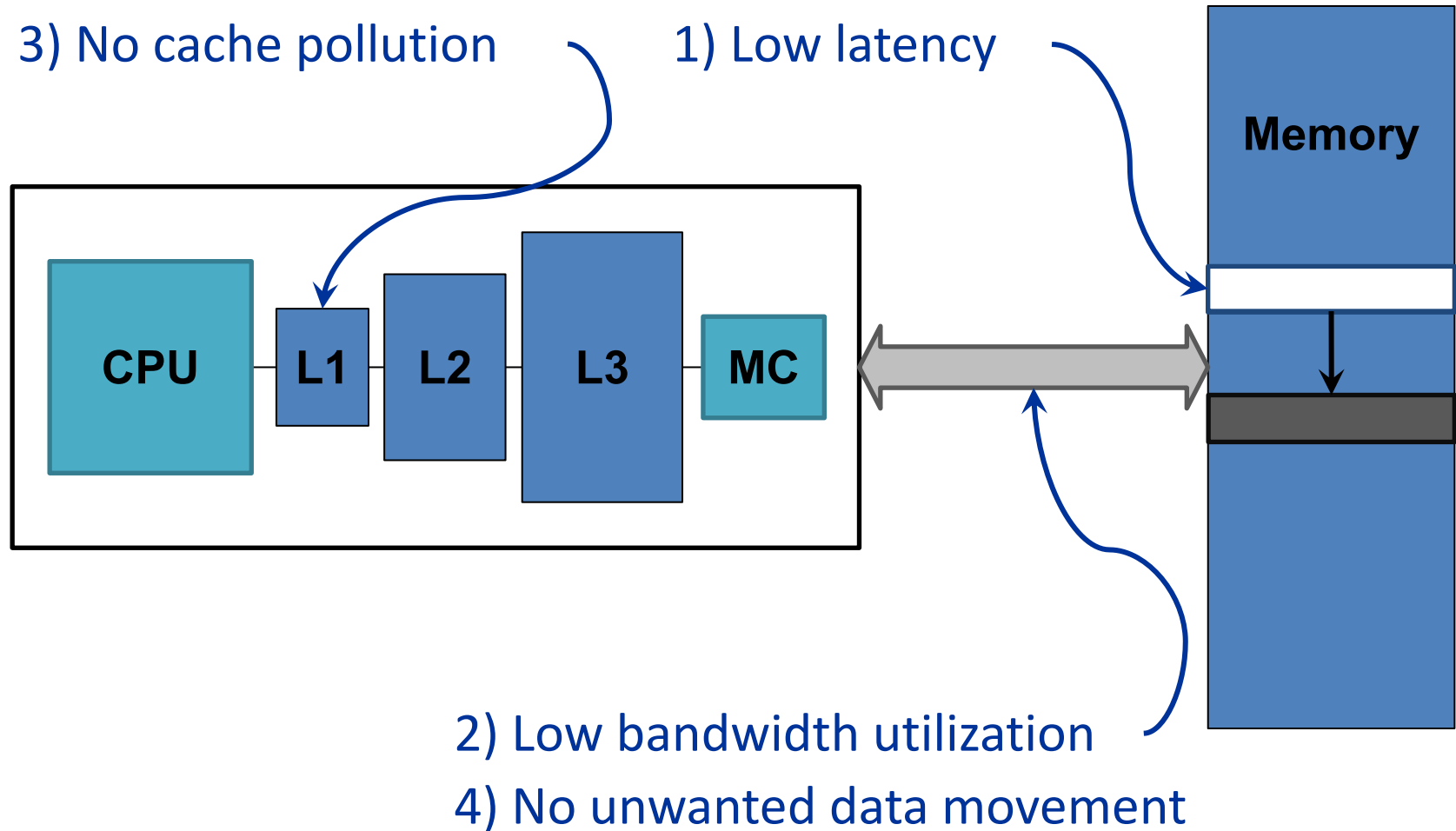
# Today's Systems: Bulk Data Copy



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



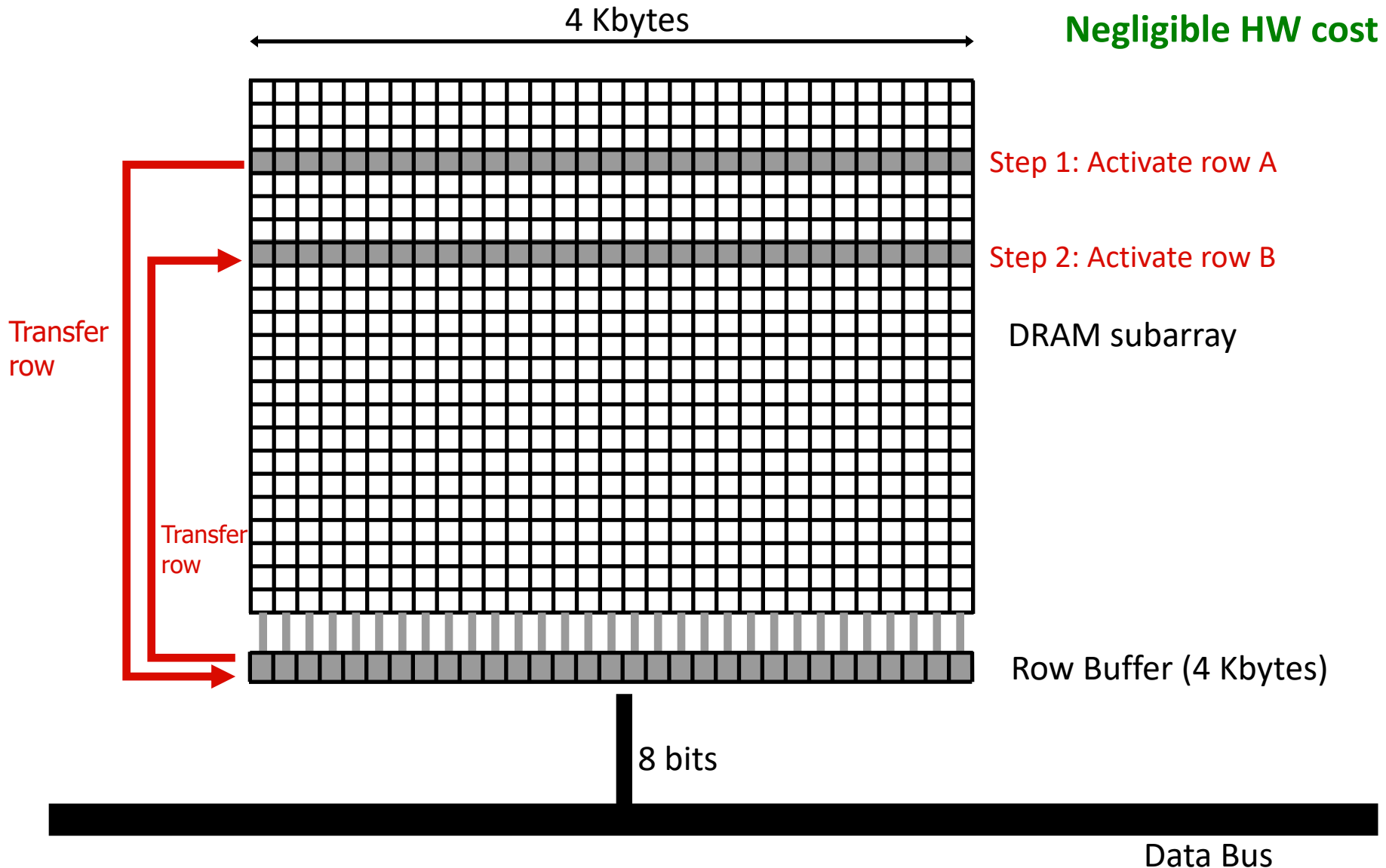
# Future Systems: In-Memory Copy



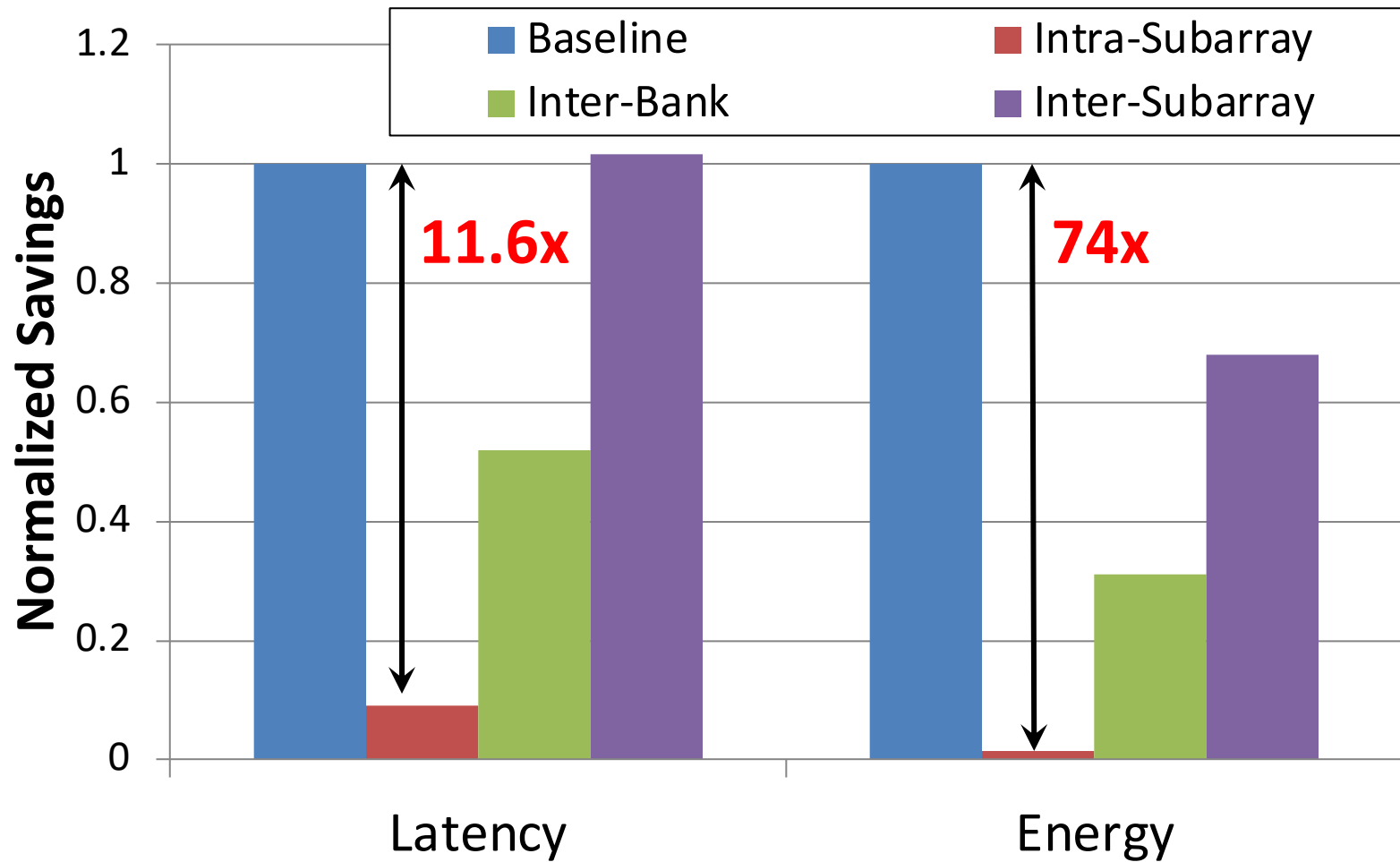
1046ns, 3.6uJ → 90ns, 0.04uJ

# RowClone: In-DRAM Row Copy

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



# RowClone: Latency and Energy Savings



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

# More on RowClone

---

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

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

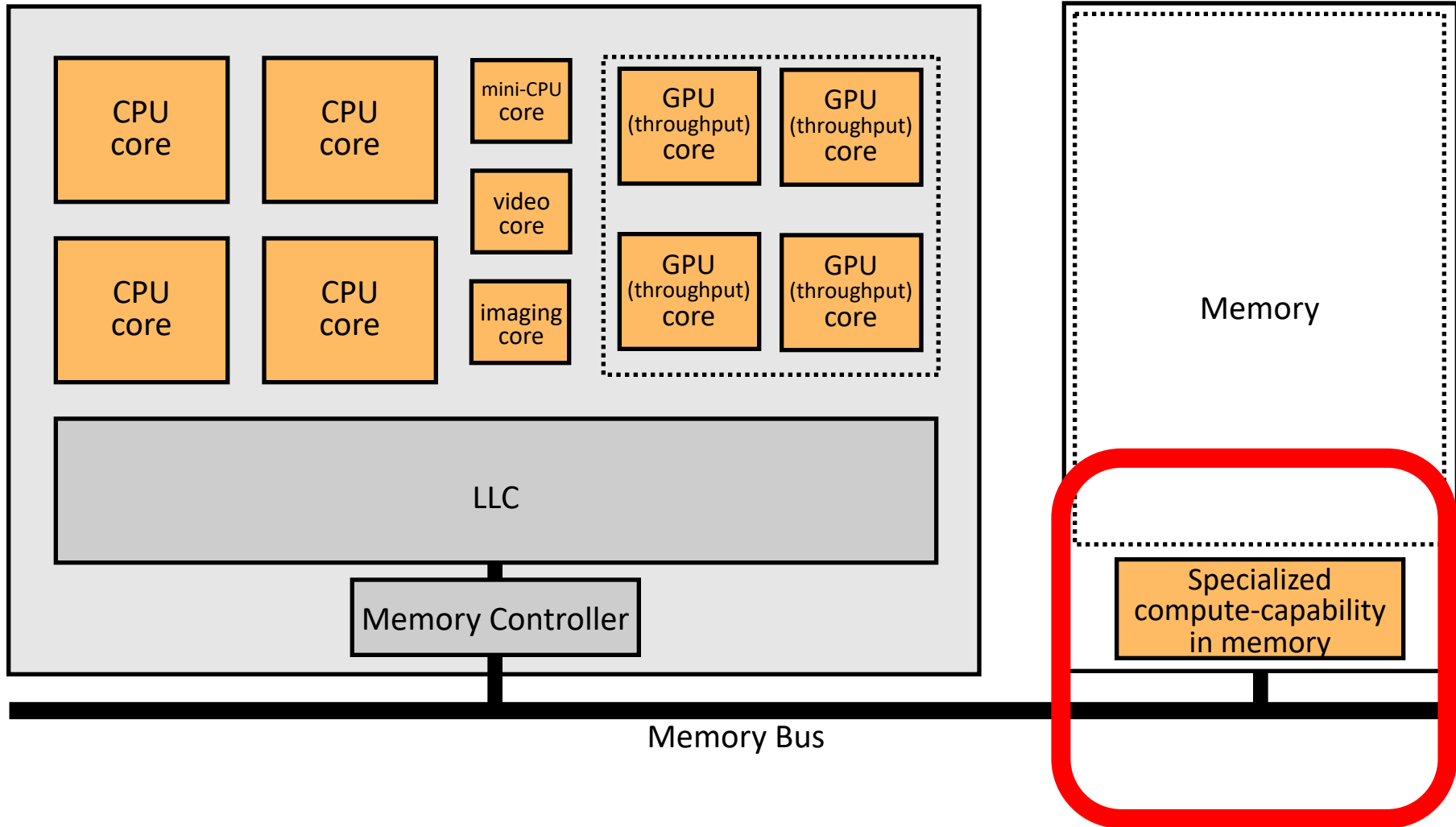
Vivek Seshadri      Yoongu Kim      Chris Fallin\*      Donghyuk Lee  
vseshadr@cs.cmu.edu    yoongukim@cmu.edu    cfallin@c1f.net    donghyuk1@cmu.edu

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



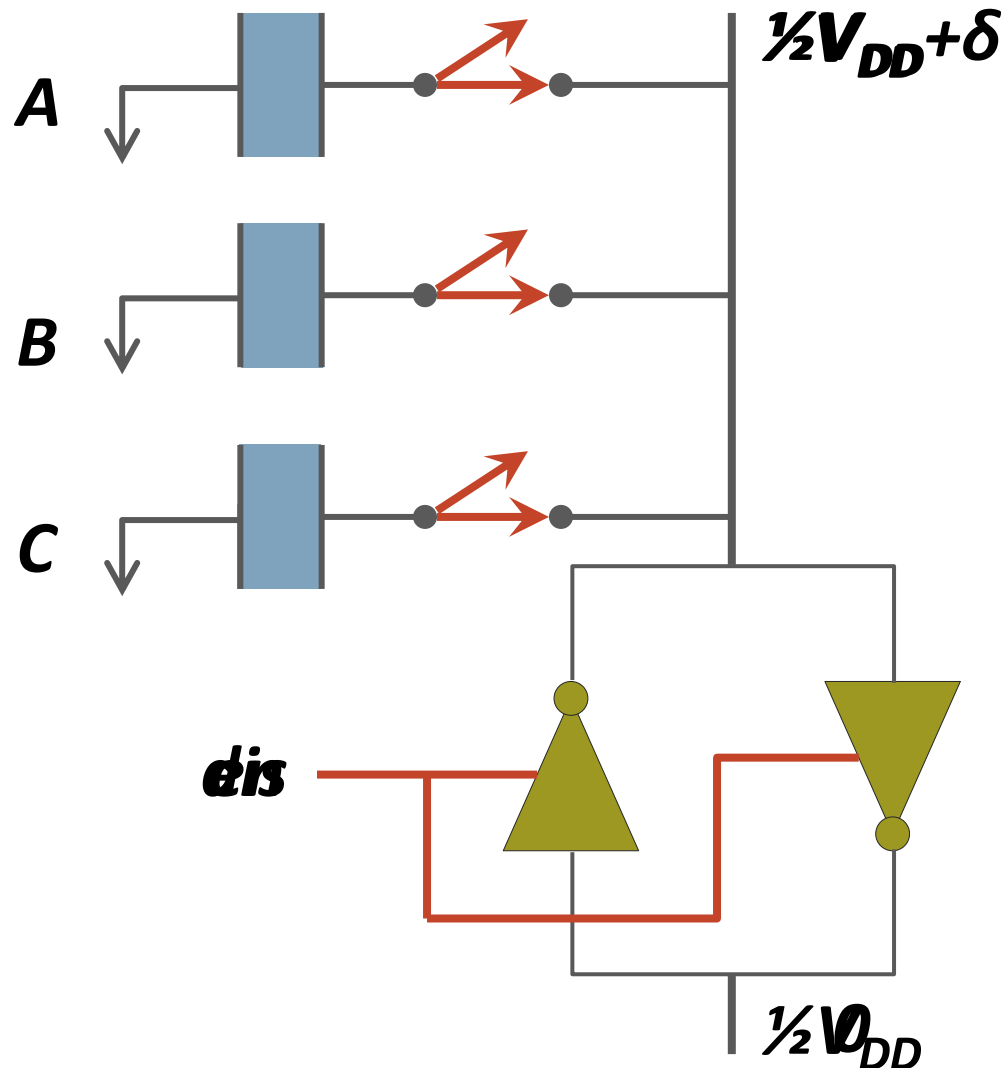
**Memory similar to a "conventional" accelerator**

# 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

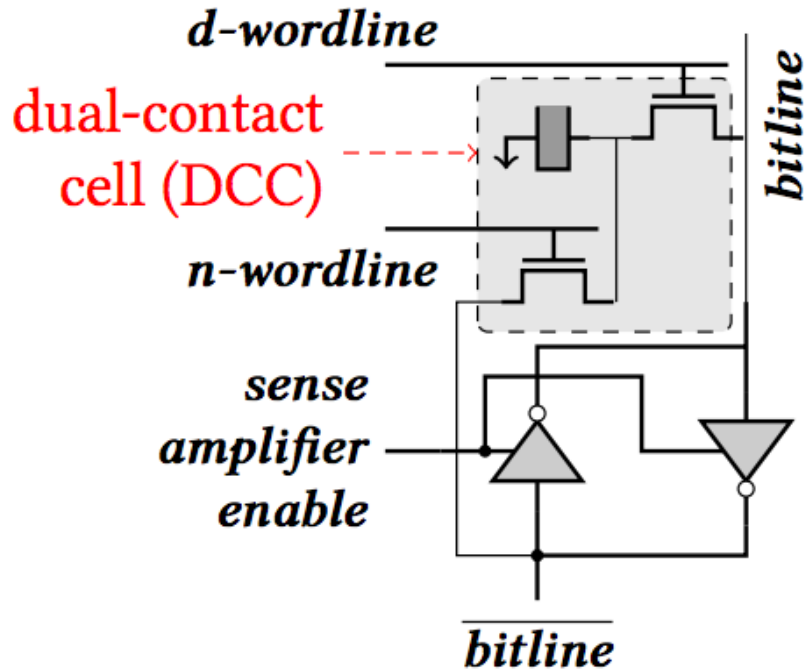


**Final State**  
 **$AB + BC + AC$**

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



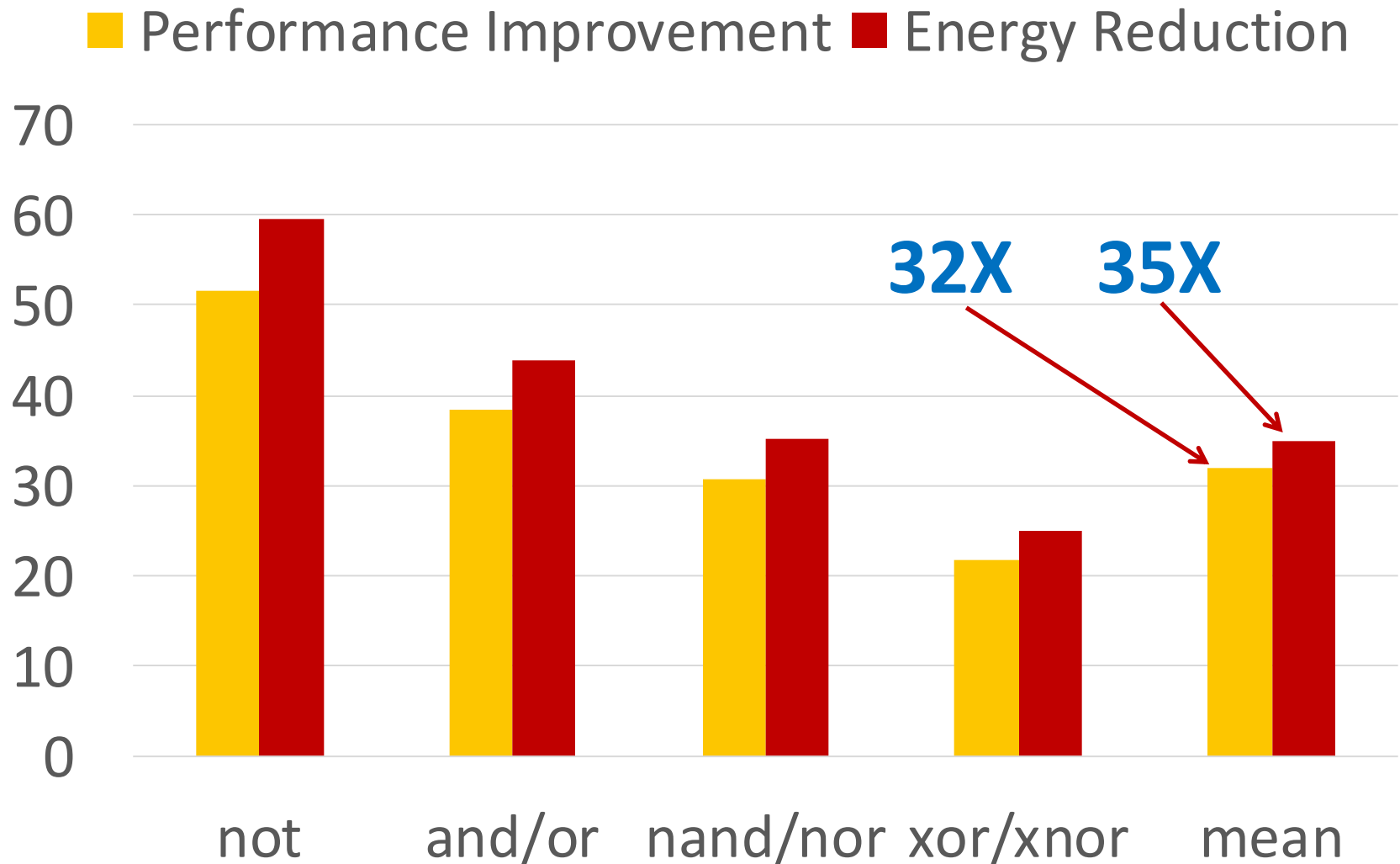
# In-DRAM NOT: Dual Contact Cell



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

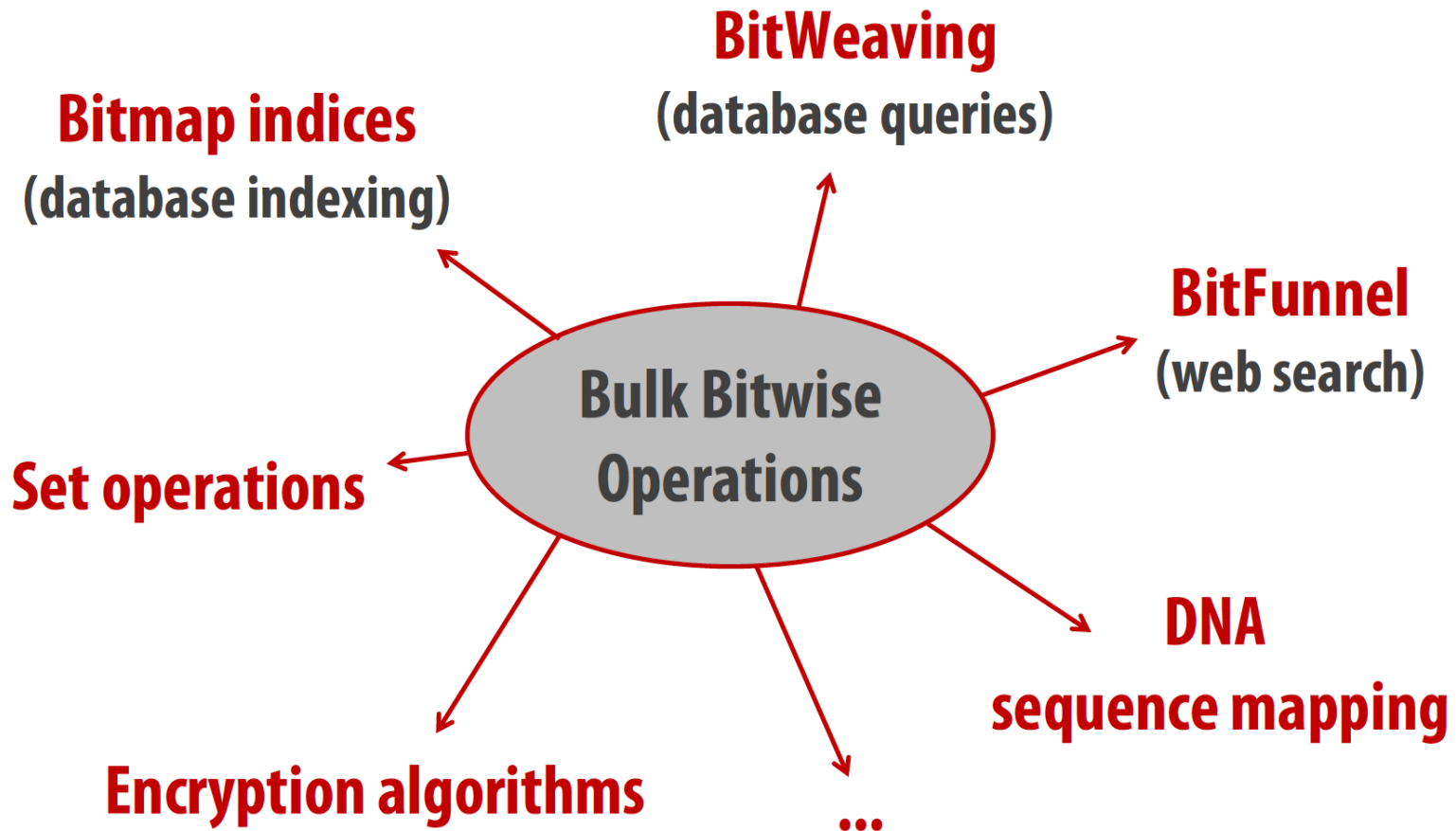
Idea:  
Feed the  
negated value  
in the sense amplifier  
into a special row

# Ambit vs. DDR3: Performance and Energy

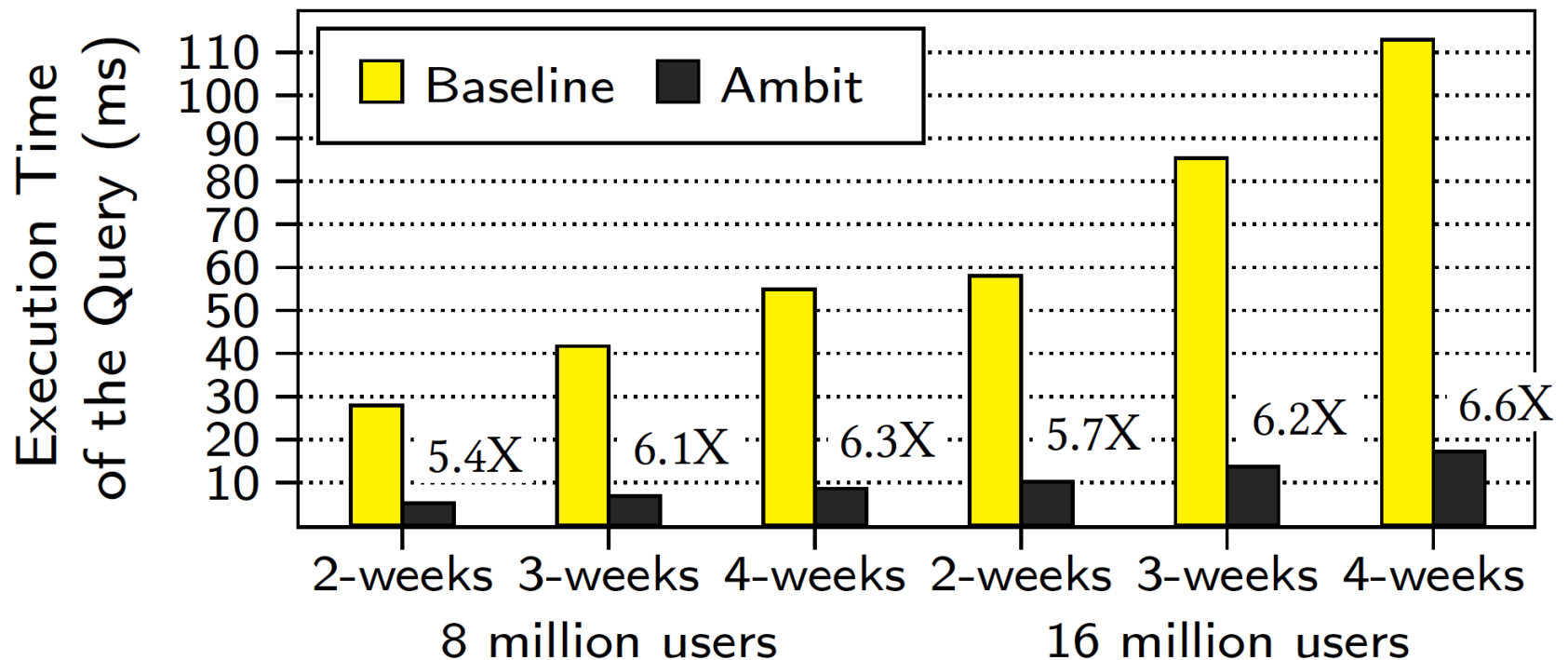


# Bulk Bitwise Operations in Workloads

---



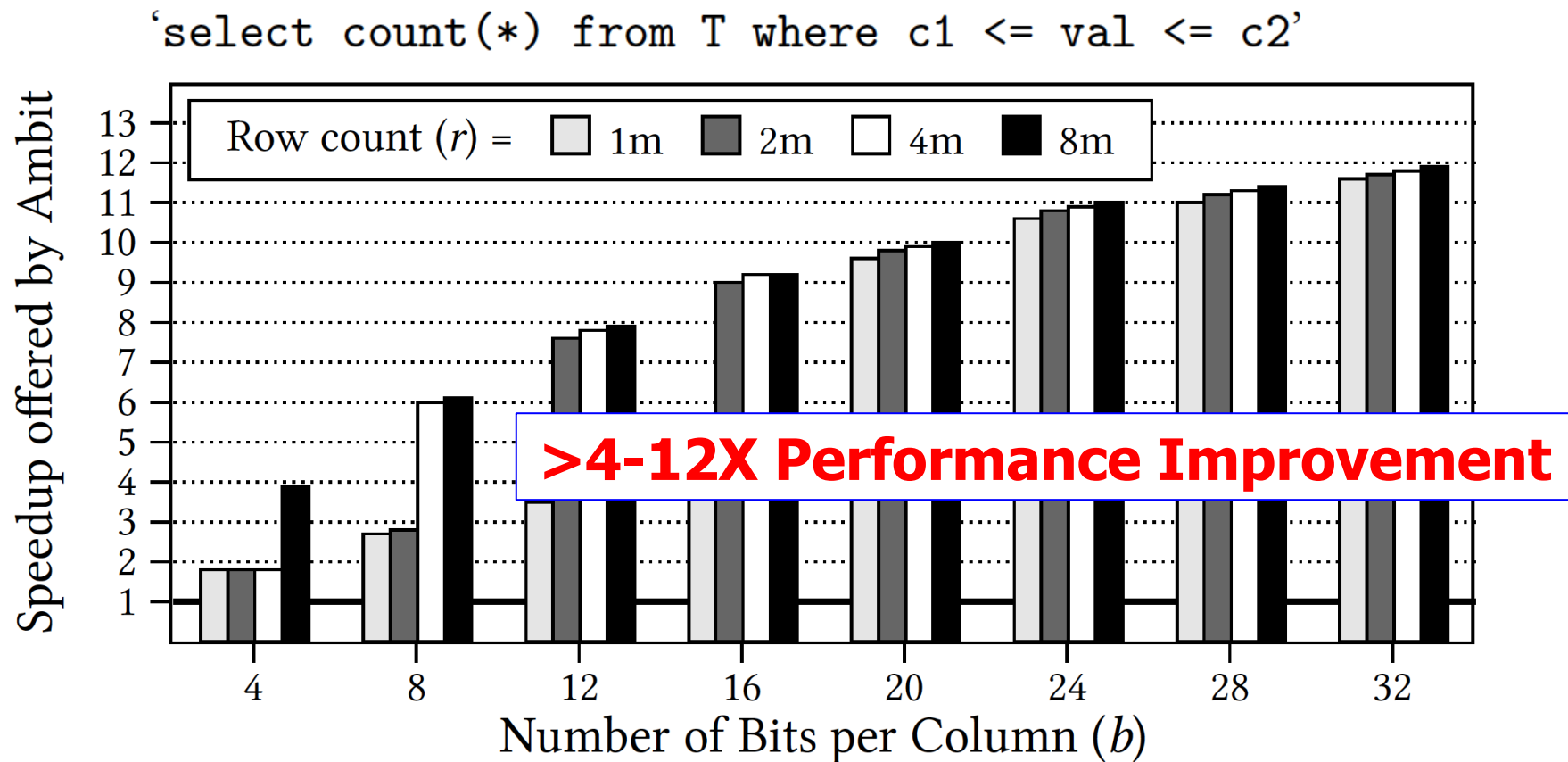
# Performance: Bitmap Index on Ambit



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

**>5.4-6.6X Performance Improvement**

# Performance: BitWeaving on Ambit



**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 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

# In-DRAM Bulk Bitwise Execution

---

- Vivek Seshadri and Onur Mutlu,  
**"In-DRAM Bulk Bitwise Execution Engine"**  
*Invited Book Chapter in Advances in Computers*, to appear  
in 2020.  
[[Preliminary arXiv version](#)]

## In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri  
Microsoft Research India  
visesha@microsoft.com

Onur Mutlu  
ETH Zürich  
onur.mutlu@inf.ethz.ch



# Sounds Good, No?

---

## Review from ISCA 2016

### Paper summary

The paper proposes to extend DRAM to include bulk, bit-wise logical operations directly between rows within the DRAM.

---

### Strengths

- Very clever/novel idea.
  - Great potential speedup and efficiency gains.
- 

### Weaknesses

- Probably won't ever be built. Not practical to assume DRAM manufacturers with change DRAM in this way.
-

# Another Review

---

## Another Review from ISCA 2016

### Strengths

The proposed mechanisms effectively exploit the operation of the DRAM to perform efficient bitwise operations across entire rows of the DRAM.

---

### Weaknesses

This requires a modification to the DRAM that will only help this type of bitwise operation. It seems unlikely that something like that will be adopted.

# Yet Another Review

---

## Yet Another Review from ISCA 2016

### Weaknesses

The core novelty of Buddy RAM is almost all circuits-related (by exploiting sense amps). I do not find architectural innovation even though the circuits technique benefits architecturally by mitigating memory bandwidth and relieving cache resources within a subarray. The only related part is the new ISA support for bitwise operations at DRAM side and its induced issue on cache coherence.

# The Reviewer Accountability Problem

---

## Acknowledgments

We thank the reviewers of ISCA 2016/2017, MICRO 2016/2017, and HPCA 2017 for their valuable comments. We

# We Have a Mindset Issue...

---

- There are many other similar examples from reviews...
  - For many other papers...
- And, we are not even talking about JEDEC yet...
- How do we fix the mindset problem?
- By doing more research, education, implementation in alternative processing paradigms

**We need to work on enabling the better future...**

## We Need to Fix the Reviewer Accountability Problem

## Main Memory Needs Intelligent Controllers

# Our Community Needs Accountable Reviewers



# RowClone & Bitwise Ops in Real DRAM Chips

---

## ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs

Fei Gao

feig@princeton.edu

Department of Electrical Engineering  
Princeton University

Georgios Tziantzioulis

georgios.tziantzioulis@princeton.edu

Department of Electrical Engineering  
Princeton University

David Wentzlaff

wentzlaf@princeton.edu

Department of Electrical Engineering  
Princeton University

# Pinatubo: RowClone and Bitwise Ops in PCM

---

## **Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories**

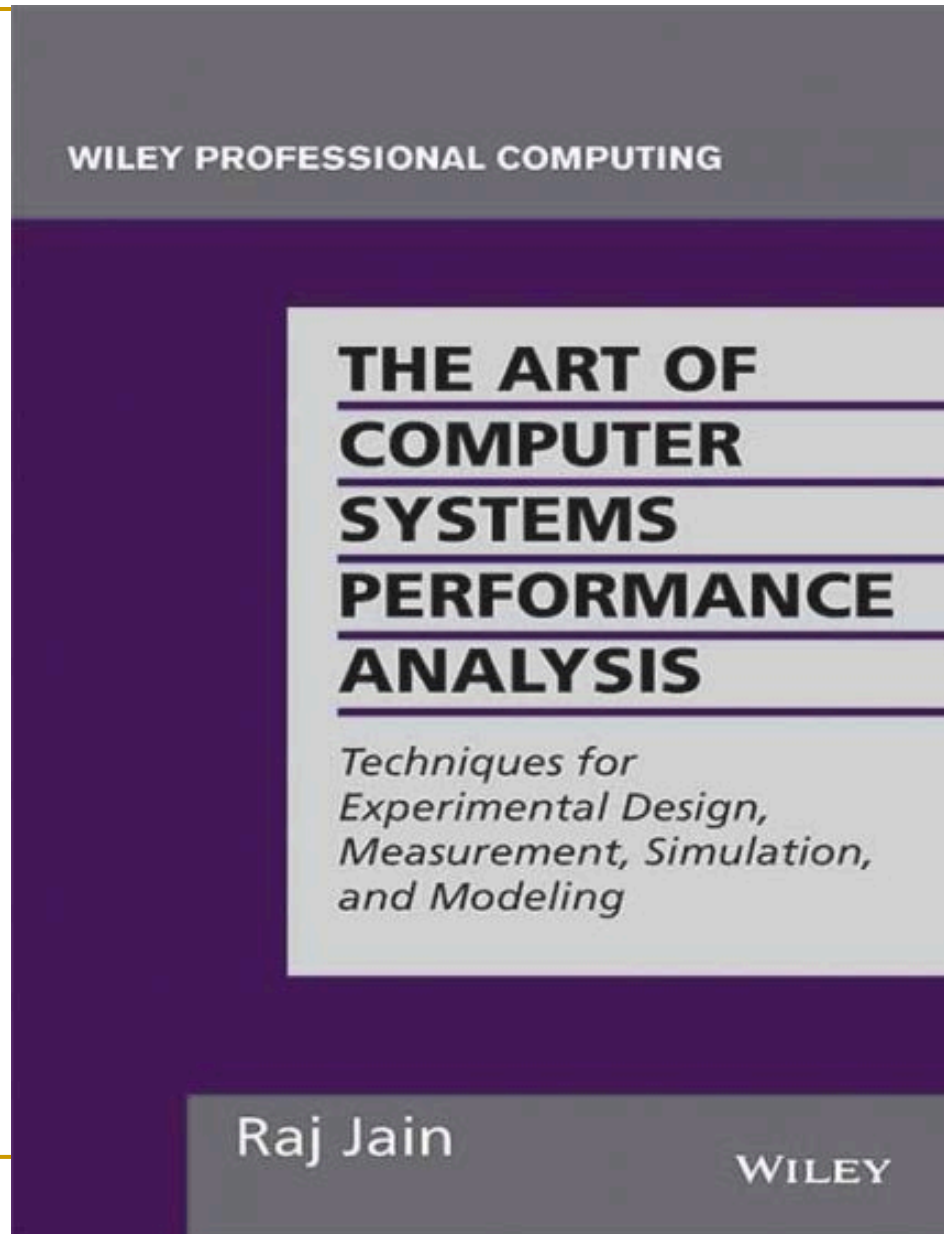
Shuangchen Li<sup>1\*</sup>, Cong Xu<sup>2</sup>, Qiaosha Zou<sup>1,5</sup>, Jishen Zhao<sup>3</sup>, Yu Lu<sup>4</sup>, and Yuan Xie<sup>1</sup>

University of California, Santa Barbara<sup>1</sup>, Hewlett Packard Labs<sup>2</sup>

University of California, Santa Cruz<sup>3</sup>, Qualcomm Inc.<sup>4</sup>, Huawei Technologies Inc.<sup>5</sup>  
{shuangchenli, yuanxie}@ece.ucsb.edu<sup>1</sup>

# Aside: A Recommended Book

---



Raj Jain, "[The Art of Computer Systems Performance Analysis](#)," Wiley, 1991.

## 10.8 DECISION MAKER'S GAMES

Even if the performance analysis is correctly done and presented, it may not be enough to persuade your audience—the decision makers—to follow your recommendations. The list shown in Box 10.2 is a compilation of reasons for rejection heard at various performance analysis presentations. You can use the list by presenting it immediately and pointing out that the reason for rejection is not new and that the analysis deserves more consideration. Also, the list is helpful in getting the competing proposals rejected!

There is no clear end of an analysis. Any analysis can be rejected simply on the grounds that the problem needs more analysis. This is the first reason listed in Box 10.2. The second most common reason for rejection of an analysis and for endless debate is the workload. Since workloads are always based on the past measurements, their applicability to the current or future environment can always be questioned. Actually workload is one of the four areas of discussion that lead a performance presentation into an endless debate. These “rat holes” and their relative sizes in terms of time consumed are shown in Figure 10.26. Presenting this cartoon at the beginning of a presentation helps to avoid these areas.

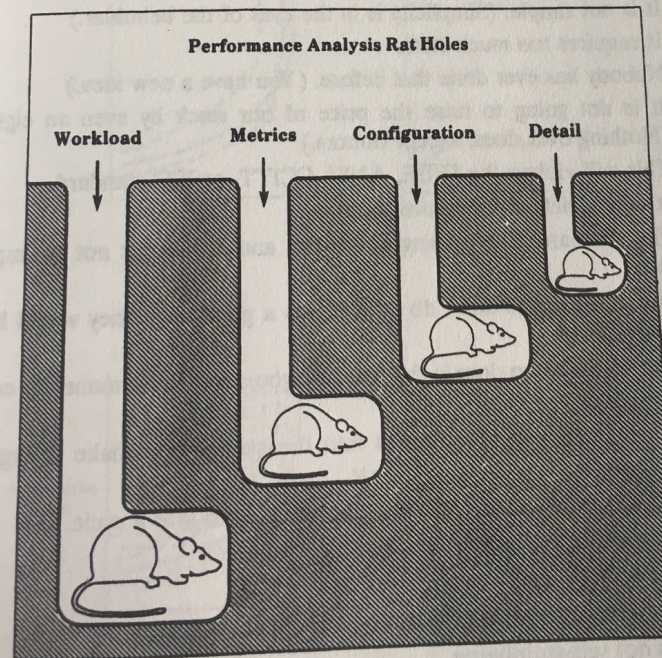


FIGURE 10.26 Four issues in performance presentations that commonly lead to endless discussion.

Raj Jain, "The Art of Computer Systems Performance Analysis," Wiley, 1991.



**Box 10.2 Reasons for Not Accepting the Results of an Analysis**


1. This needs more analysis.
2. You need a better understanding of the workload.
3. It improves performance only for long I/O's, packets, jobs, and files, and most of the I/O's, packets, jobs, and files are short.
4. It improves performance only for short I/O's, packets, jobs, and files, but who cares for the performance of short I/O's, packets, jobs, and files; its the long ones that impact the system.
5. It needs too much memory/CPU/bandwidth and memory/CPU/bandwidth isn't free.
6. It only saves us memory/CPU/bandwidth and memory/CPU/bandwidth is cheap.
7. There is no point in making the networks (similarly, CPUs/disks/...) faster; our CPUs/disks (any component other than the one being discussed) aren't fast enough to use them.
8. It improves the performance by a factor of  $x$ , but it doesn't really matter at the user level because everything else is so slow.
9. It is going to increase the complexity and cost.
10. Let us keep it simple stupid (and your idea is not stupid).
11. It is not simple. (Simplicity is in the eyes of the beholder.)
12. It requires too much state.
13. Nobody has ever done that before. (You have a new idea.)
14. It is not going to raise the price of our stock by even an eighth. (Nothing ever does, except rumors.)
15. This will violate the IEEE, ANSI, CCITT, or ISO standard.
16. It may violate some future standard.
17. The standard says nothing about this and so it must not be important.
18. Our competitors don't do it. If it was a good idea, they would have done it.
19. Our competition does it this way and you don't make money by copying others.
20. It will introduce randomness into the system and make debugging difficult.
21. It is too deterministic; it may lead the system into a cycle.
22. It's not interoperable.
23. This impacts hardware.
24. That's beyond today's technology.
25. It is not self-stabilizing.
26. Why change—it's working OK.

Raj Jain, "The Art of Computer Systems Performance Analysis," Wiley, 1991.

# Initial RowHammer Reviews

## Disturbance Errors in DRAM: Demonstration, Characterization, and Prevention

Rejected (R2)

 863kB

Friday 31 May 2013 2:00:53pm PDT

b9bf06021da54cddf4cd0b3565558a181868b972

You are an **author** of this paper.

+ ABSTRACT

+ AUTHORS

	OveMer	Nov	WriQua	RevExp
<a href="#">Review #66A</a>	1	4	4	4
<a href="#">Review #66B</a>	5	4	5	3
<a href="#">Review #66C</a>	2	3	5	4
<a href="#">Review #66D</a>	1	2	3	4
<a href="#">Review #66E</a>	4	4	4	3
<a href="#">Review #66F</a>	2	4	4	3

# Missing the Point **Reviews from Micro 2013**

## **PAPER WEAKNESSES**

This is an excellent test methodology paper, but there is no micro-architectural or architectural content.

## **PAPER WEAKNESSES**

- Whereas they show disturbance may happen in DRAM array, authors don't show it can be an issue in realistic DRAM usage scenario
- Lacks architectural/microarchitectural impact on the DRAM disturbance analysis

## **PAPER WEAKNESSES**

The mechanism investigated by the authors is one of many well known disturb mechanisms. The paper does not discuss the root causes to sufficient depth and the importance of this mechanism compared to others. Overall the length of the sections restating known information is much too long in relation to new work.

### PAPER WEAKNESSES

1) The disturbance error (a.k.a coupling or cross-talk noise induced error) is a known problem to the DRAM circuit community.

2) What you demonstrated in this paper is so called DRAM row hammering issue - you can even find a Youtube video showing this! - <http://www.youtube.com/watch?v=i3-gQSnBcdo>

2) The architectural contribution of this study is too insignificant.

### PAPER WEAKNESSES

- Row Hammering appears to be well-known, and solutions have already been proposed by industry to address the issue.

- The paper only provides a qualitative analysis of solutions to the problem. A more robust evaluation is really needed to know whether the proposed solution is necessary.



# Final RowHammer Reviews

---

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

Accepted



639kB

21 Nov 2013 10:53:11pm CST |

f039be2735313b39304ae1c6296523867a485610

You are an **author** of this paper.

	OveMer	Nov	WriQua	RevConAnd
<a href="#">Review #41A</a>	8	4	5	3
<a href="#">Review #41B</a>	7	4	4	3
<a href="#">Review #41C</a>	6	4	4	3
<a href="#">Review #41D</a>	2	2	5	4
<a href="#">Review #41E</a>	3	2	3	3
<a href="#">Review #41F</a>	7	4	4	3

# Suggestions to Reviewers

---

- Be fair; you do not know it all
- Be open-minded; you do not know it all
- Be accepting of diverse research methods: there is no single way of doing research
- Be constructive, not destructive
- Do not have double standards...

**Do not block or delay scientific progress for non-reasons**

# Processing in Memory: Two Approaches

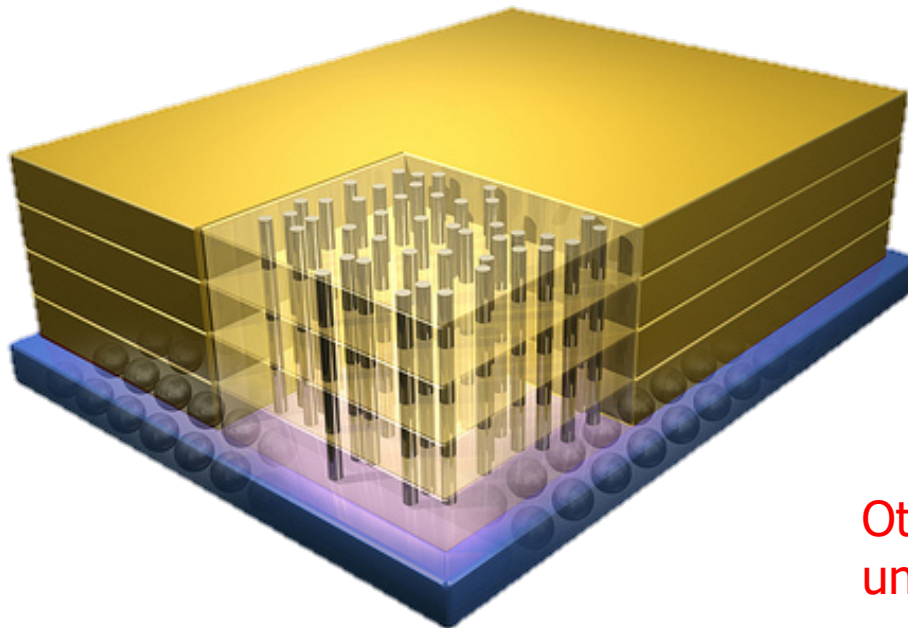
1. Minimally changing memory chips
2. Exploiting 3D-stacked memory

# 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]; RLDram3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

Table 1. Landscape of DRAM-based memory

Kim+, “[Ramulator: A Flexible and Extensible DRAM Simulator](#)”, IEEE CAL 2015.

# Two Key Questions in 3D-Stacked PIM

---

- What are the performance and energy benefits of using 3D-stacked memory as a coarse-grained accelerator?
  - By changing the entire system
  - By performing simple function offloading
  
- What is the minimal processing-in-memory support we can provide?
  - With minimal changes to system and programming

# Graph Processing

- Large graphs are everywhere (circa 2015)



36 Million  
Wikipedia Pages



1.4 Billion  
Facebook Users

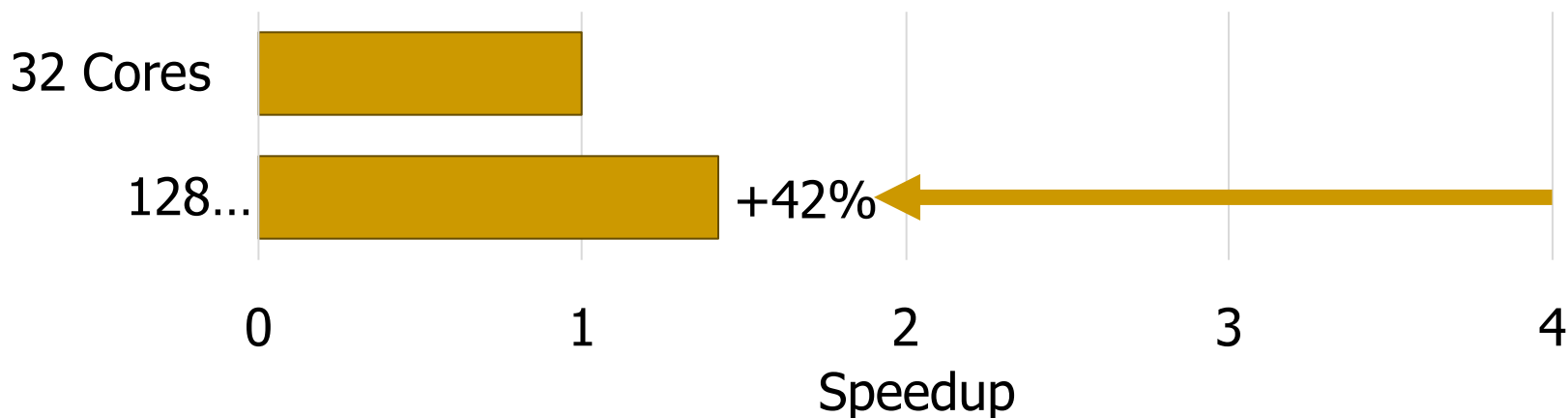


300 Million  
Twitter Users



30 Billion  
Instagram Photos

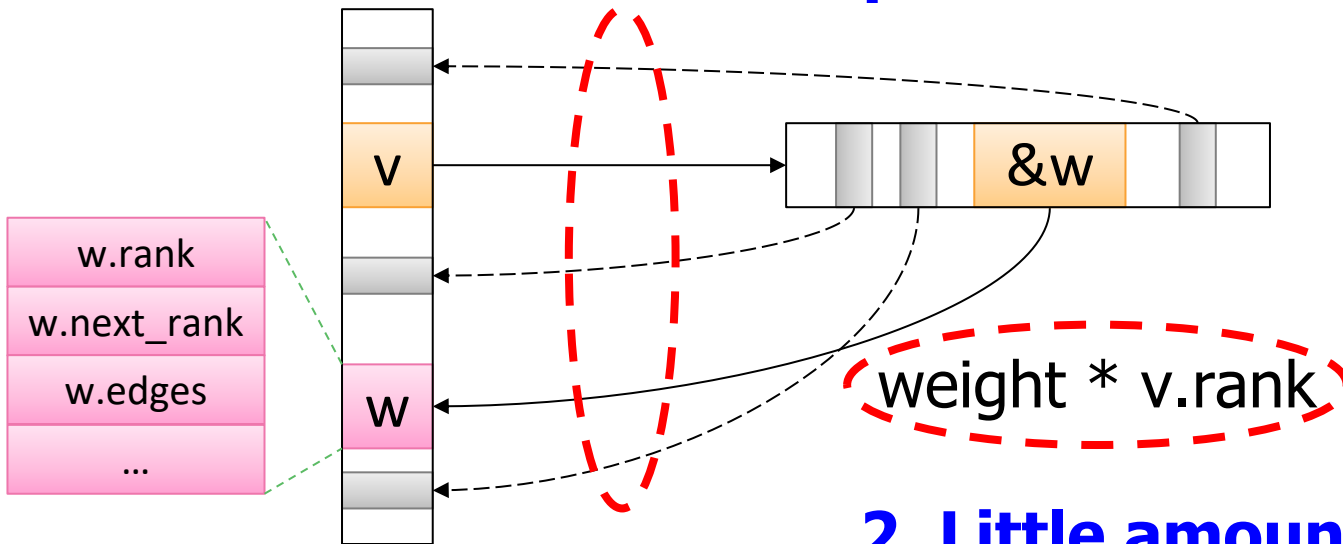
- Scalable large-scale graph processing is challenging



# Key Bottlenecks in Graph Processing

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

## 1. Frequent random memory accesses

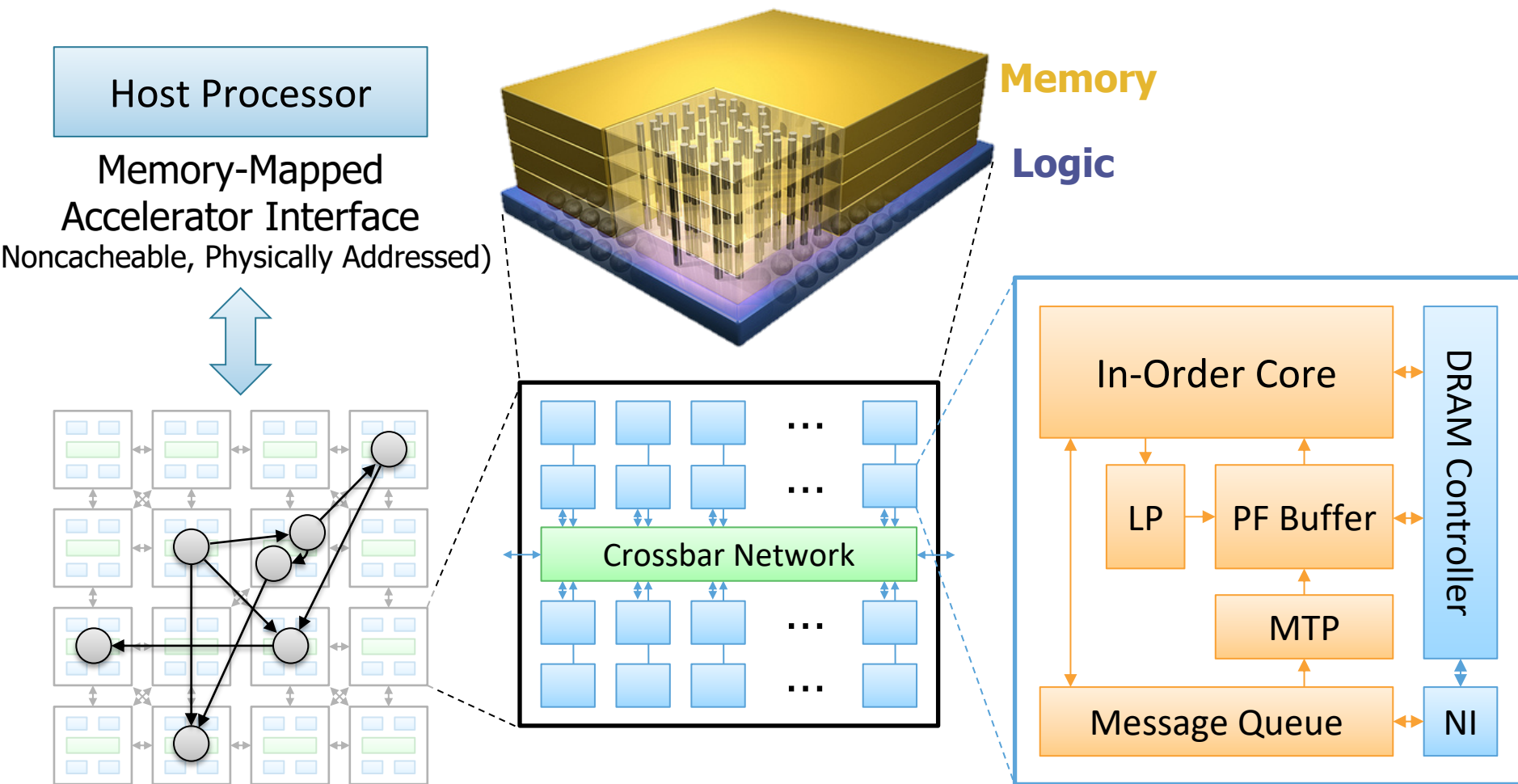


## 2. Little amount of computation

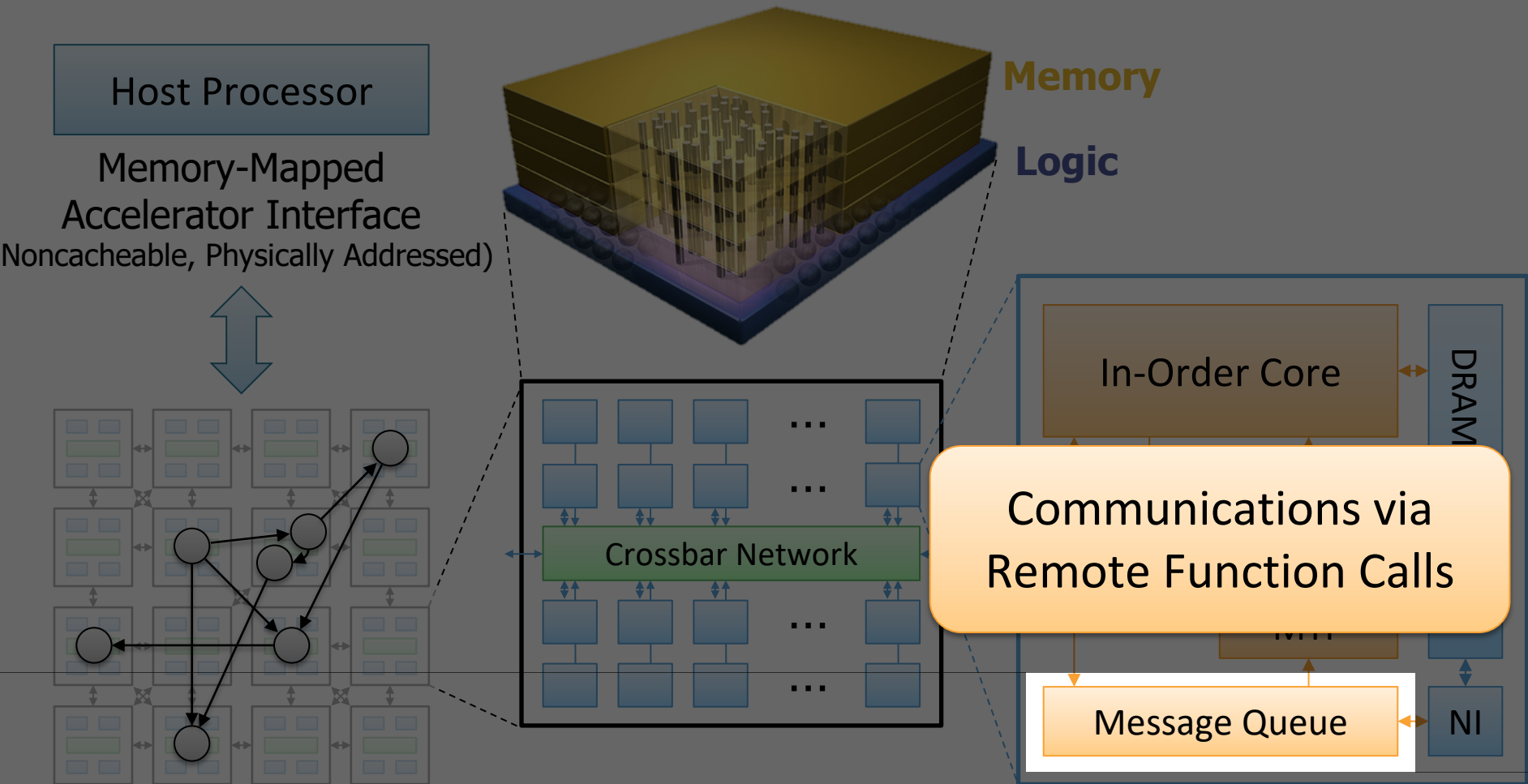


# Tesseract System for Graph Processing

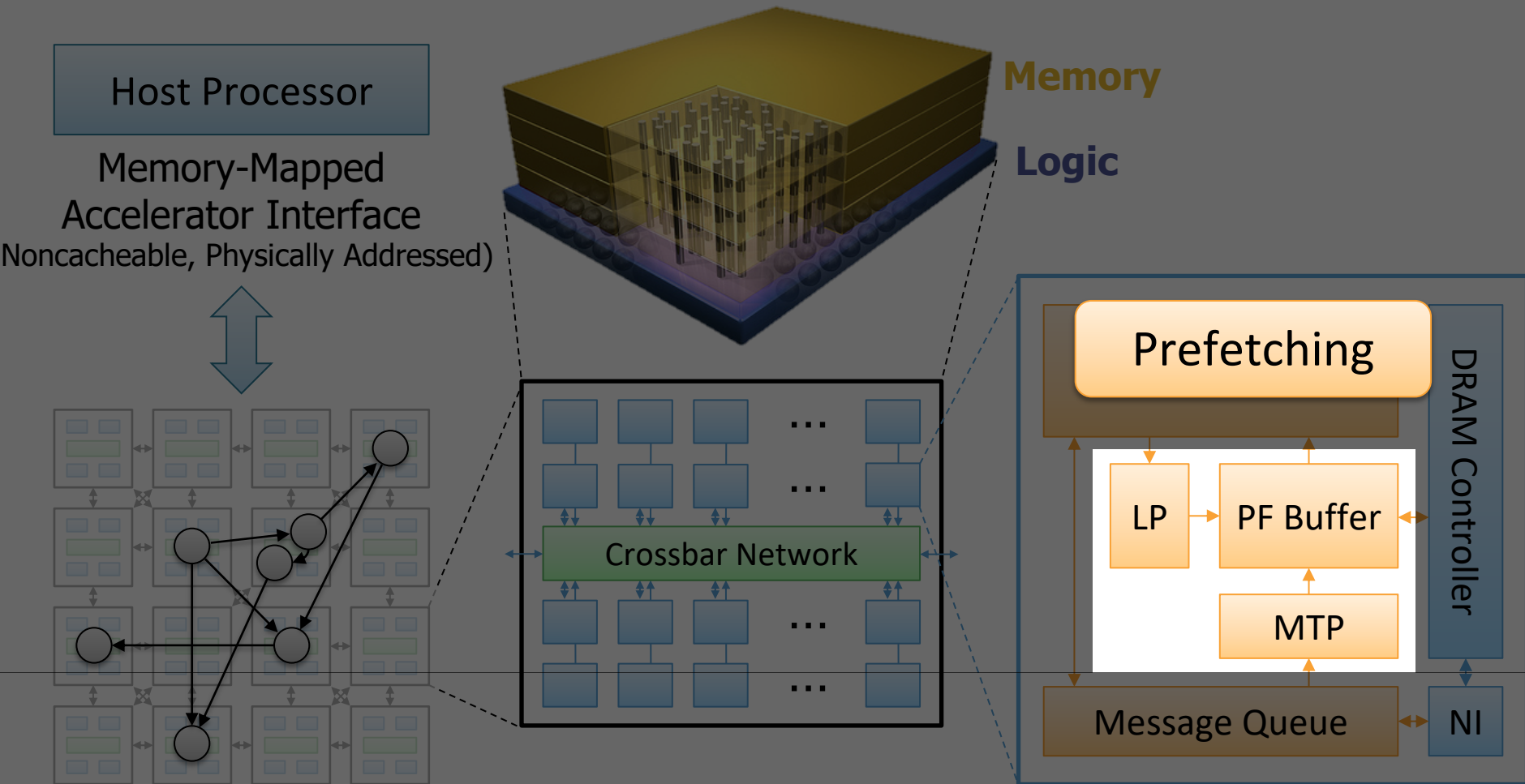
Interconnected set of 3D-stacked memory+logic chips with simple cores



# Tesseract System for Graph Processing

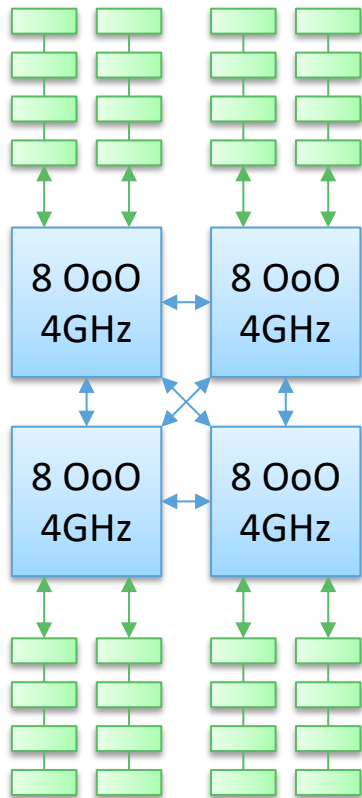


# Tesseract System for Graph Processing



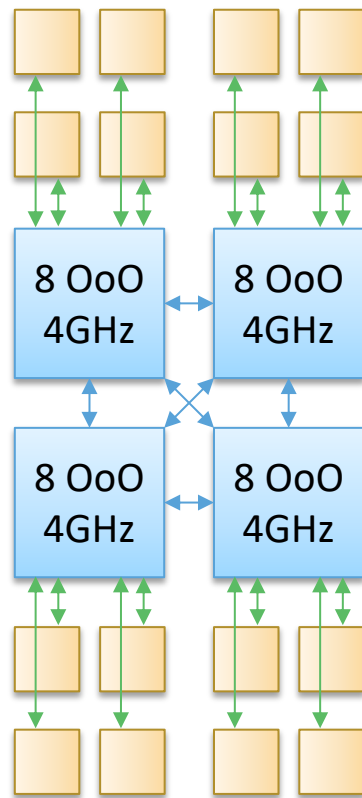
# Evaluated Systems

DDR3-OoO



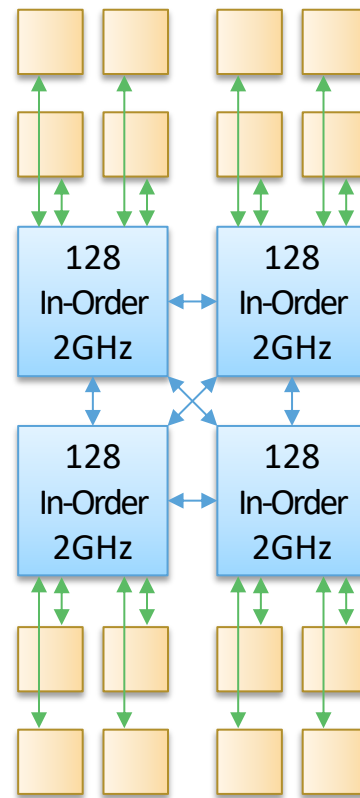
102.4GB/s

HMC-OoO



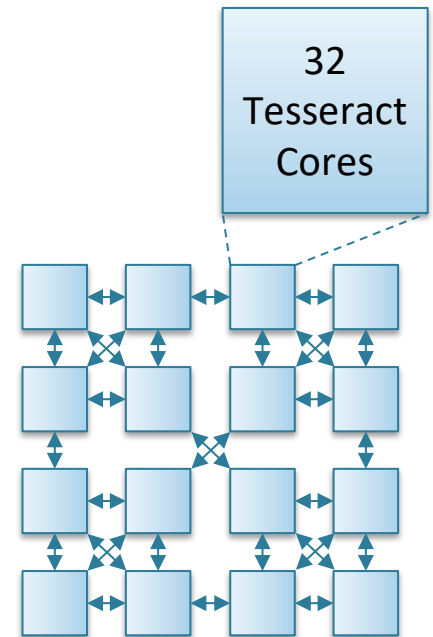
640GB/s

HMC-MC



640GB/s

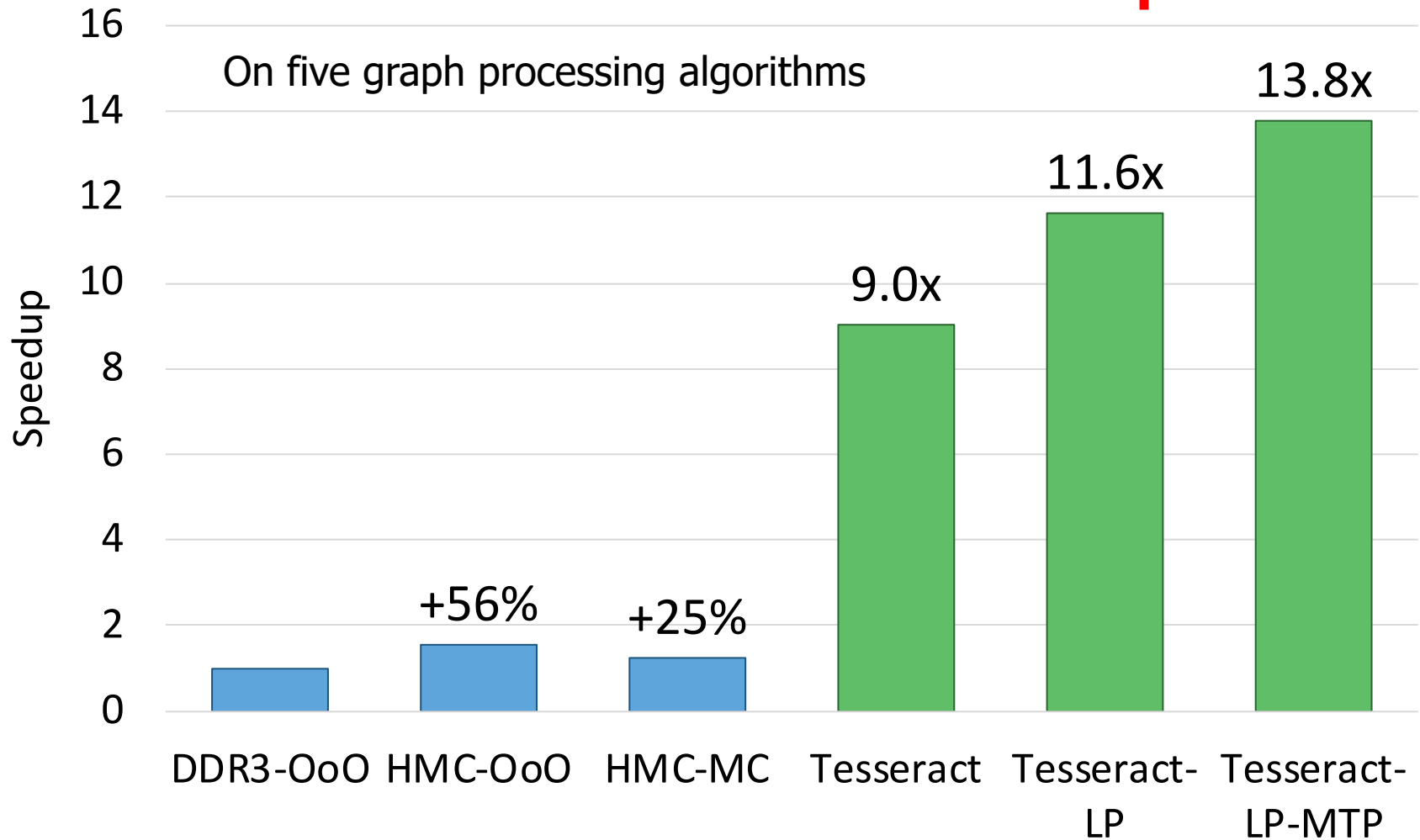
**Tesseract**



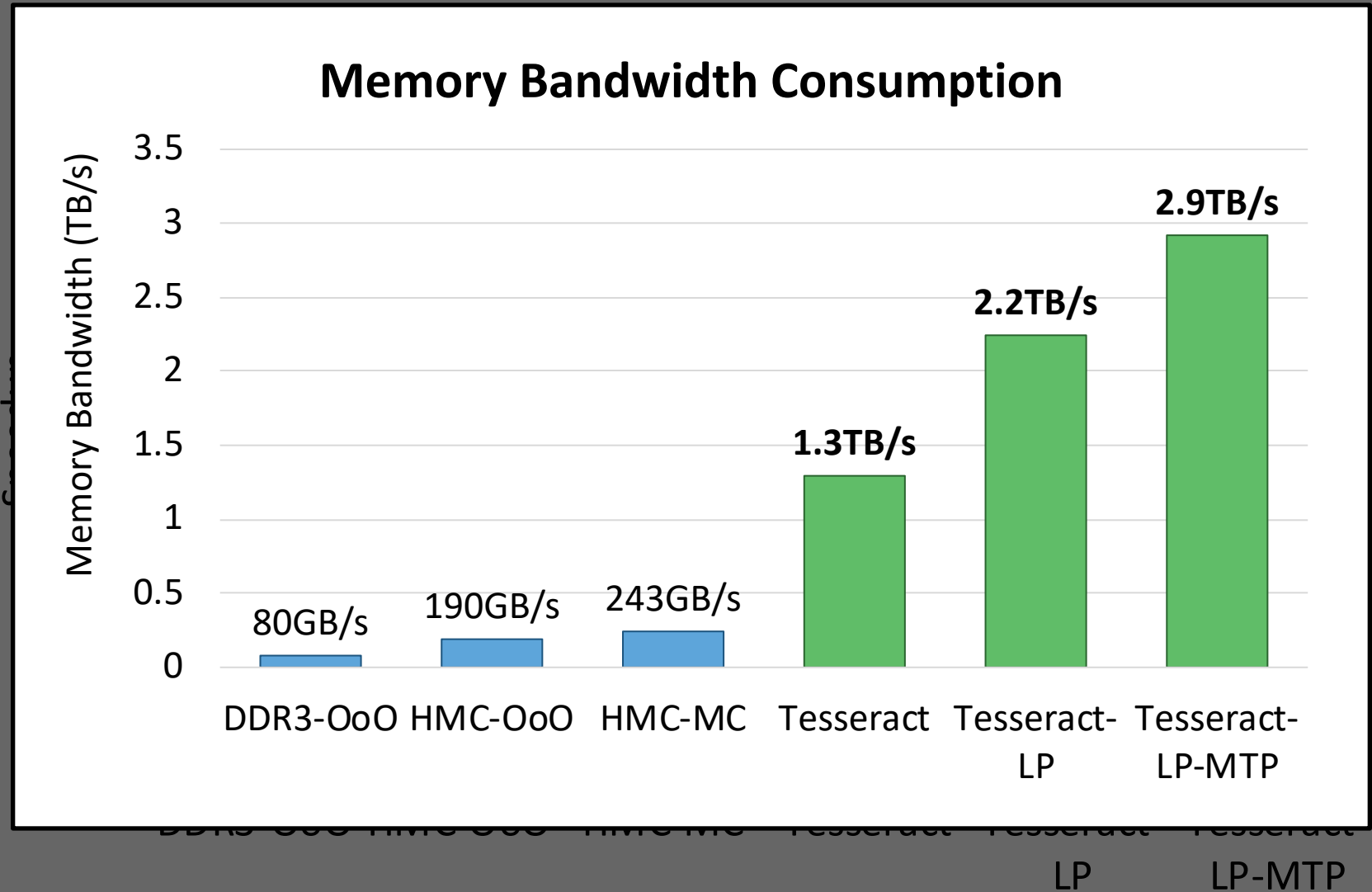
**8TB/s**

# Tesseract Graph Processing Performance

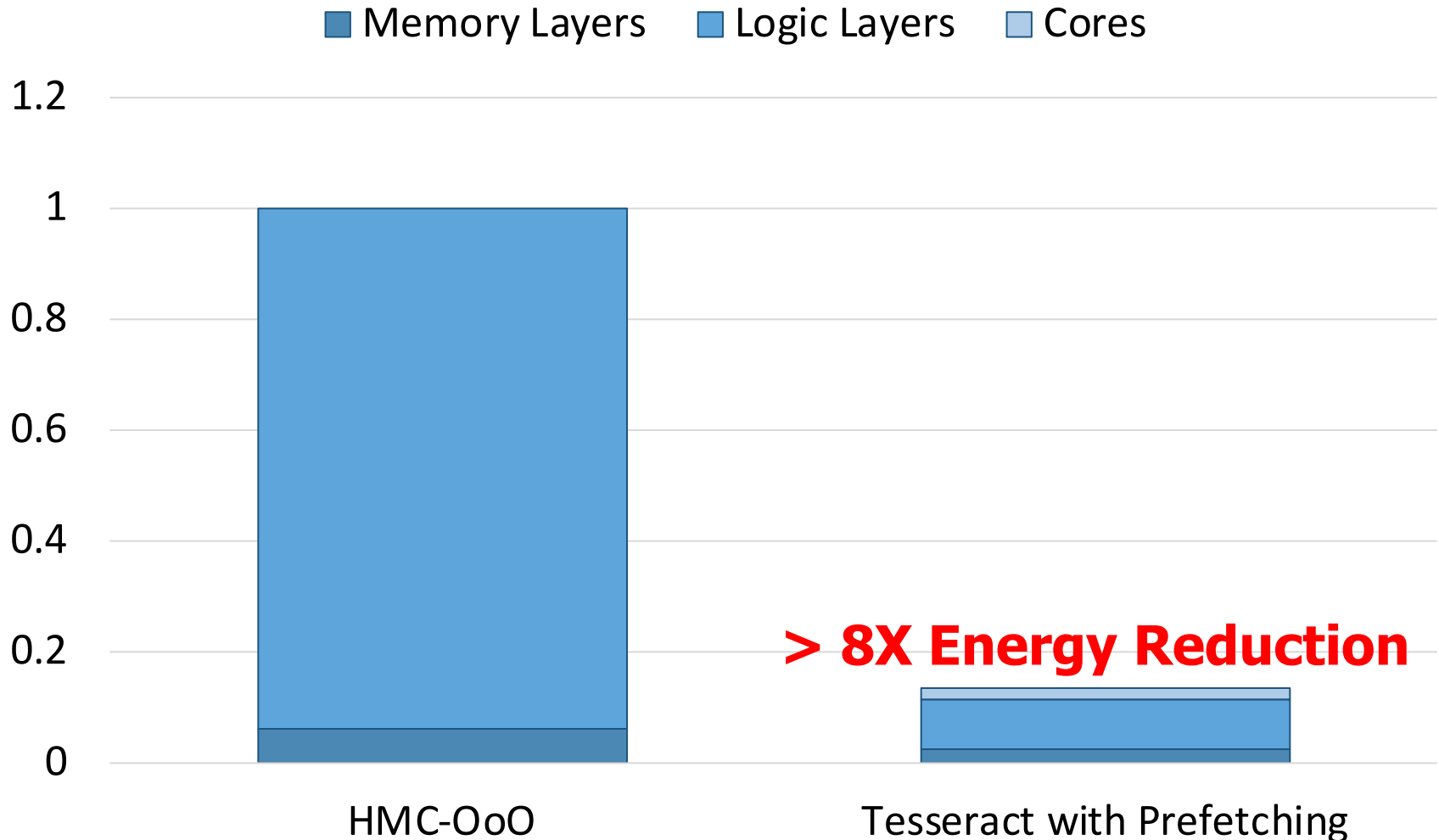
**>13X Performance Improvement**



# Tesseract Graph Processing Performance



# Tesseract Graph Processing System Energy



# More on Tesseract

---

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoungh 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>   Kiyoungh 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



# Two Key Questions in 3D-Stacked PIM

---

- What are the performance and energy benefits of using 3D-stacked memory as a coarse-grained accelerator?
  - By changing the entire system
  - By performing simple function offloading
  
- What is the minimal processing-in-memory support we can provide?
  - With minimal changes to system and programming

# Another Example: 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>

# Four Important Workloads



**Chrome**

Google's web browser



**TensorFlow Mobile**

Google's machine learning  
framework

**VP9**



**Video Playback**

Google's **video codec**

**VP9**

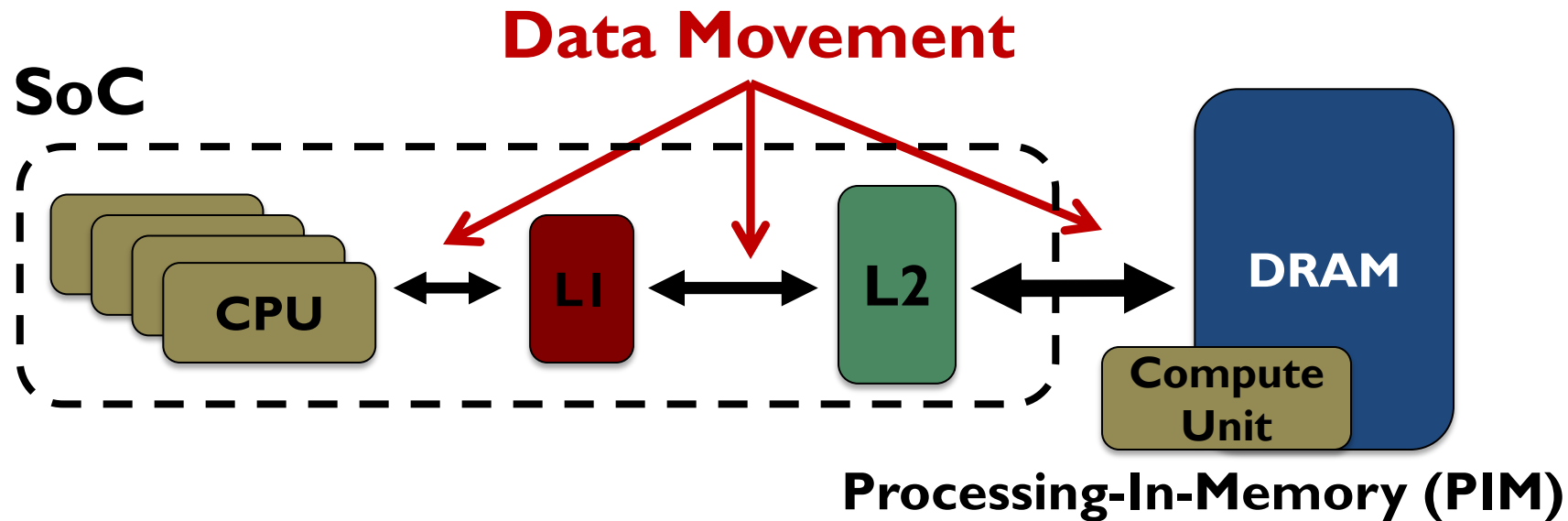


**Video Capture**

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

# Simple PIM on Mobile Workloads

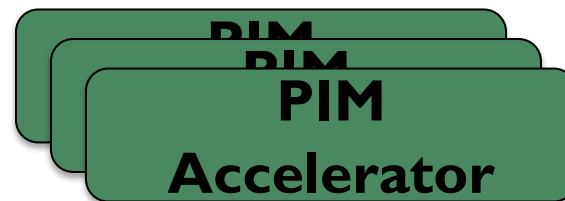
**2<sup>nd</sup> key observation:** a significant fraction of the **data movement** often comes from **simple functions**

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 execution time, 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**

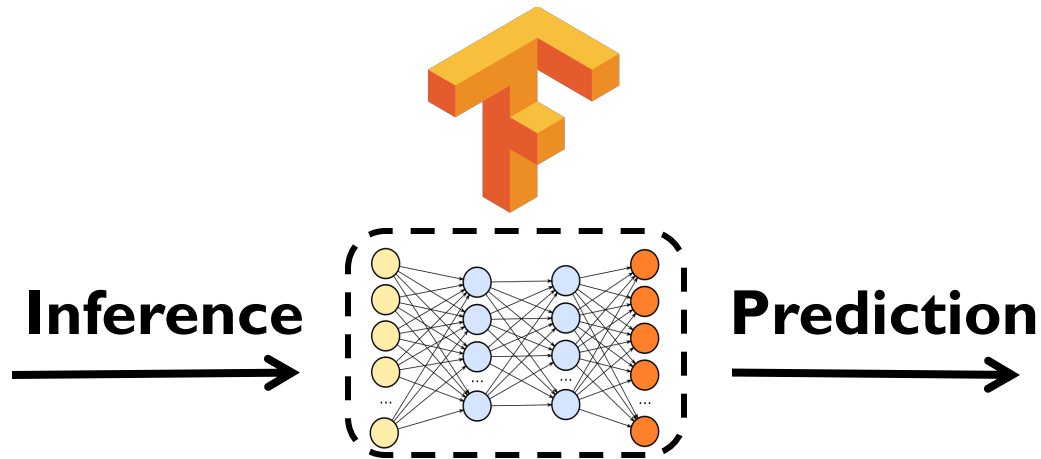
**VP9**



**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



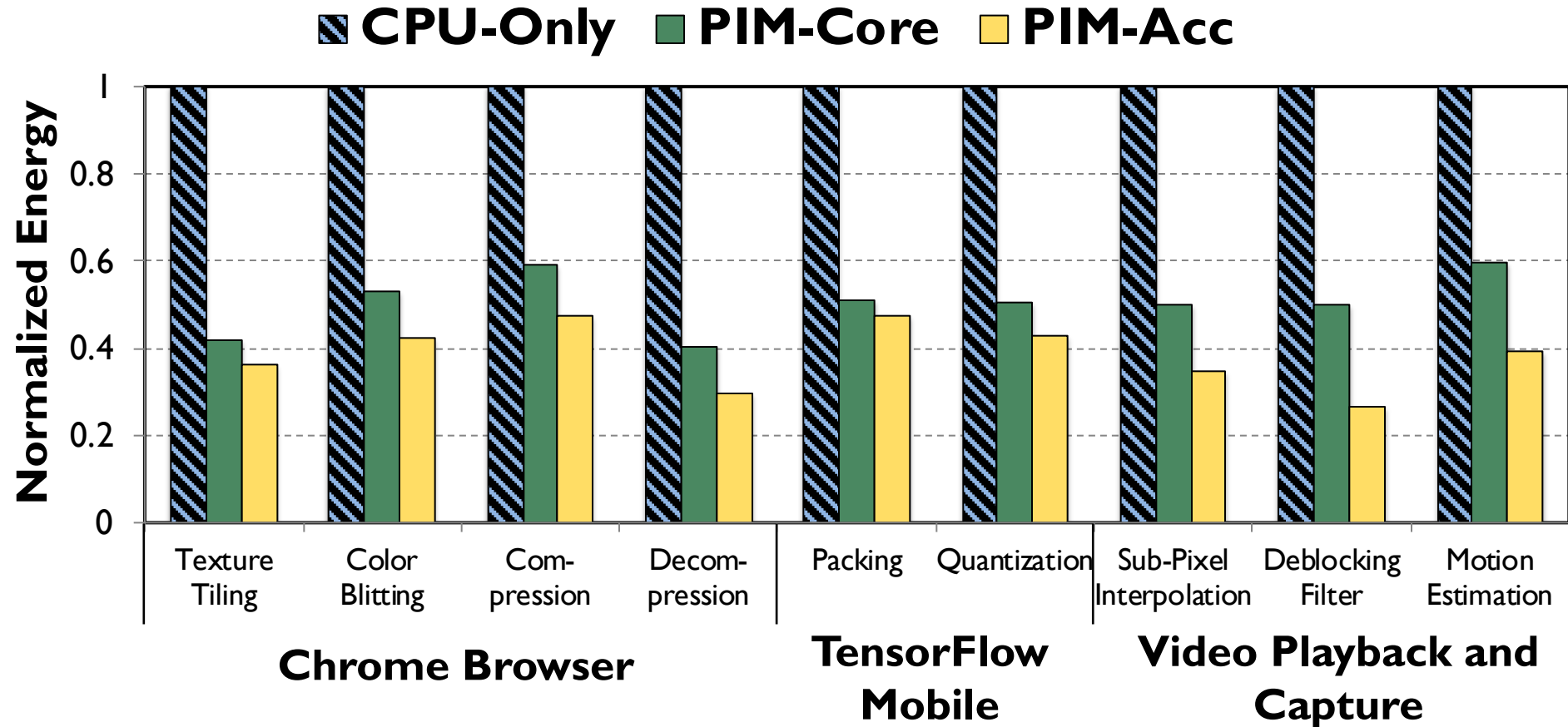
Up to **16.8%** of the inference **energy** and **16.1%** of inference **execution time**



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

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

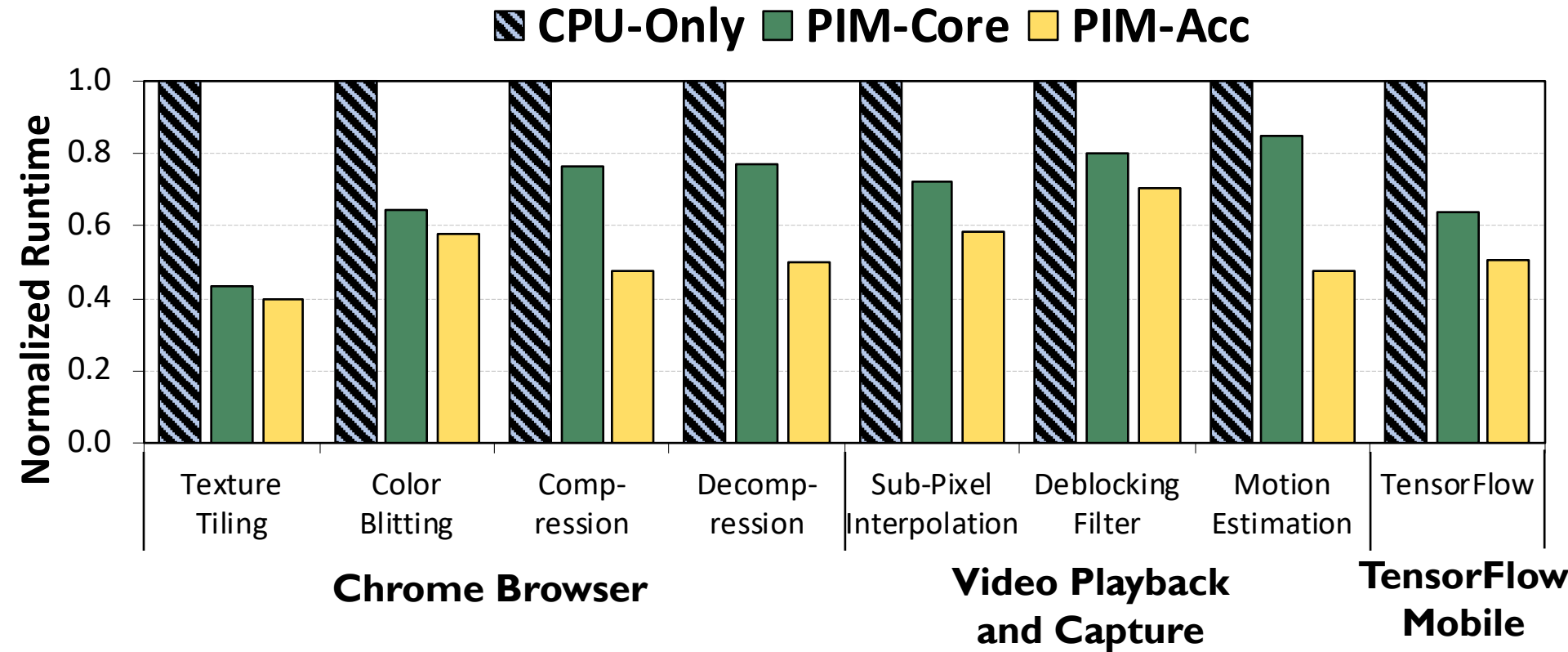
# Normalized Energy



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

**SAFARI**

# Normalized Runtime

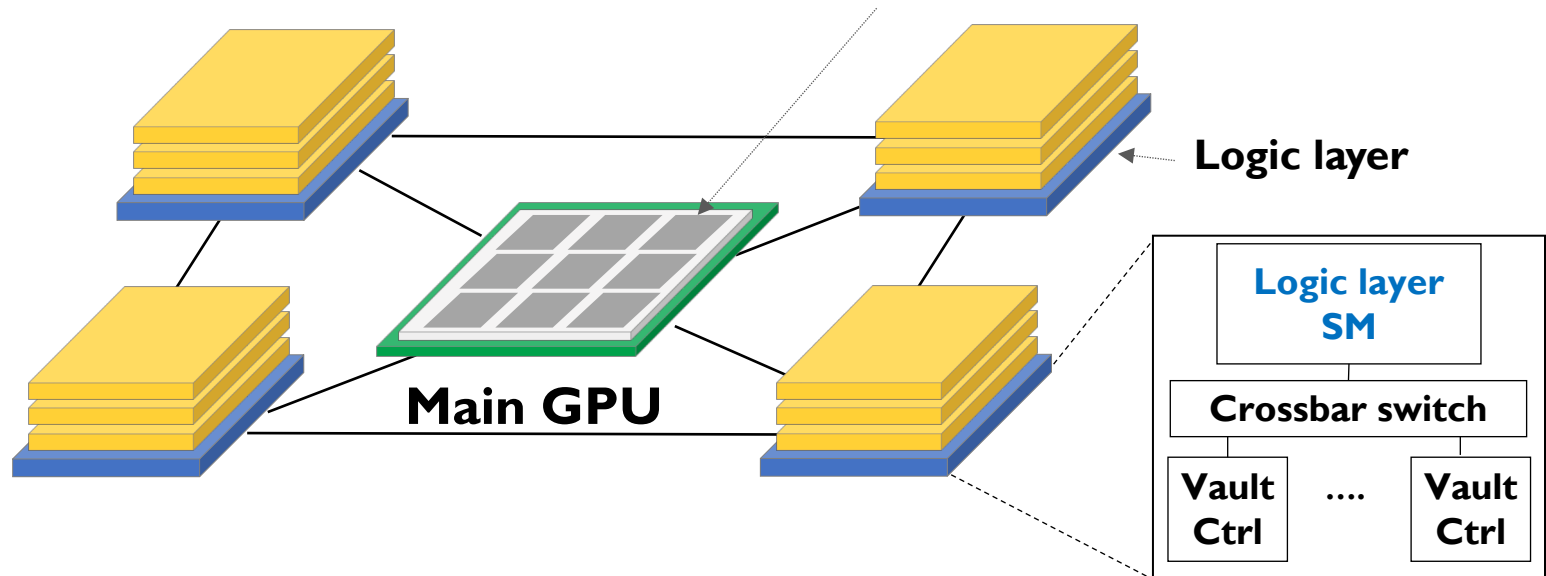


Offloading these kernels to **PIM core** and **PIM accelerator** reduces **program runtime** on average by **44.6%** and **54.2%**

# Truly Distributed GPU Processing with PIM?

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

**SM (Streaming Multiprocessor)**



```
__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;
```

# 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 \*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

## How to Enable Adoption of Processing in Memory

# Barriers to Adoption of PIM

---

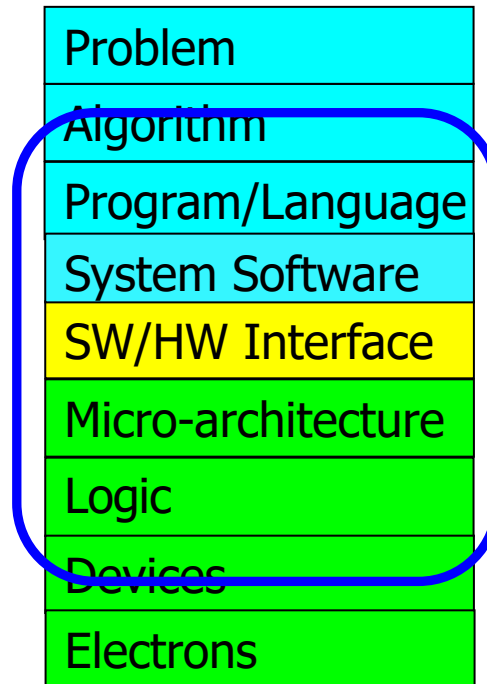
1. Functionality of and applications & software for PIM
2. Ease of programming (interfaces and compiler/HW support)
3. System support: coherence & virtual memory
4. Runtime and compilation systems for adaptive scheduling, data mapping, access/sharing control
5. Infrastructures to assess benefits and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

**All can be solved with change of mindset**



# We Need to Revisit the Entire Stack

---



**We can get there step by step**

# PIM Review and Open Problems

---

## Processing Data Where It Makes Sense: Enabling In-Memory Computation

Onur Mutlu<sup>a,b</sup>, Saugata Ghose<sup>b</sup>, Juan Gómez-Luna<sup>a</sup>, Rachata Ausavarungnirun<sup>b,c</sup>

<sup>a</sup>*ETH Zürich*

<sup>b</sup>*Carnegie Mellon University*

<sup>c</sup>*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,  
**"Processing Data Where It Makes Sense: Enabling In-Memory  
Computation"**

*Invited paper in Microprocessors and Microsystems (**MICPRO**), June 2019.  
[arXiv version]*

# PIM Review and Open Problems (II)

---

## **A Workload and Programming Ease Driven Perspective of Processing-in-Memory**

Saugata Ghose<sup>†</sup>   Amirali Boroumand<sup>†</sup>   Jeremie S. Kim<sup>†§</sup>   Juan Gómez-Luna<sup>§</sup>   Onur Mutlu<sup>§†</sup>

<sup>†</sup>*Carnegie Mellon University*

<sup>§</sup>*ETH Zürich*

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu,

**"Processing-in-Memory: A Workload-Driven Perspective"**

*Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence, to appear in November 2019.*

[Preliminary arXiv version]

# Computing Architectures with Minimal Data Movement

# Corollaries: Architectures Today ...

---

- Architectures are **terrible at dealing with data**
  - ❑ Designed to mainly store and move data vs. to compute
  - ❑ They are **processor-centric** as opposed to **data-centric**
- Architectures are **terrible at taking advantage of vast amounts of data** (and metadata) available to them
  - ❑ Designed to make simple decisions, ignoring lots of data
  - ❑ They make **human-driven decisions** vs. **data-driven** decisions
- Architectures are **terrible at knowing and exploiting different properties of application data**
  - ❑ Designed to treat all data as the same
  - ❑ They make **component-aware decisions** vs. **data-aware**

# Exploiting Data to Design Intelligent Architectures

# System Architecture Design Today

---

- Human-driven
  - Humans design the policies (how to do things)
- Many (too) simple, short-sighted policies all over the system
- No automatic data-driven policy learning
- (Almost) no learning: cannot take lessons from past actions

**Can we design  
fundamentally intelligent architectures?**

# An Intelligent Architecture

---

- Data-driven
  - Machine learns the “best” policies (how to do things)
- Sophisticated, workload-driven, changing, far-sighted policies
- Automatic data-driven policy learning
- All controllers are intelligent data-driven agents

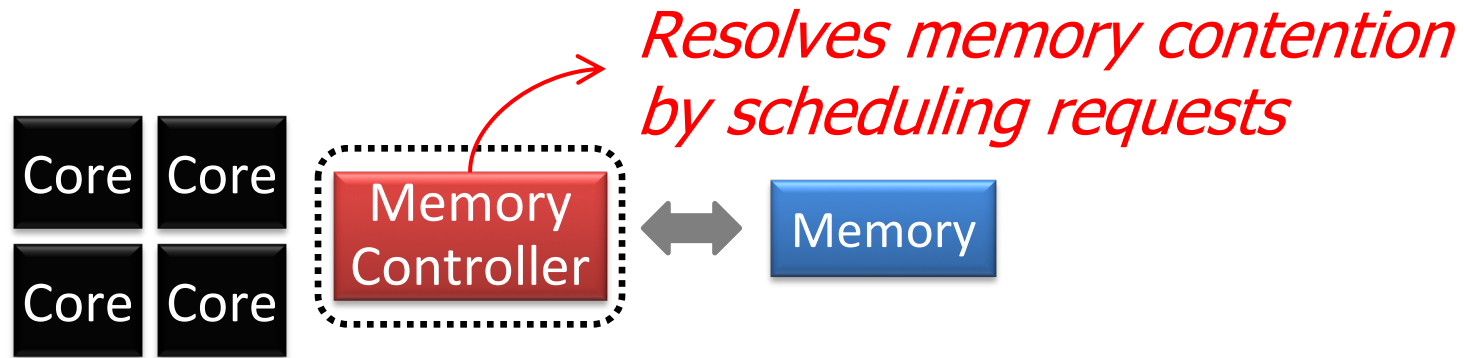
**How do we start?**



# Self-Optimizing Memory Controllers

# Memory Controller

---



How to schedule requests to maximize system performance?

# Why are Memory Controllers Difficult to Design?

---

- Need to obey **DRAM timing constraints** for correctness
  - There are many (50+) timing constraints in DRAM
  - tWTR: Minimum number of cycles to wait before issuing a read command after a write command is issued
  - tRC: Minimum number of cycles between the issuing of two consecutive activate commands to the same bank
  - ...
- Need to **keep track of many resources** to prevent conflicts
  - Channels, banks, ranks, data bus, address bus, row buffers, ...
- Need to handle **DRAM refresh**
- Need to **manage power** consumption
- Need to **optimize performance & QoS** (in the presence of constraints)
  - Reordering is not simple
  - Fairness and QoS needs complicates the scheduling problem
- ...

# Many Memory Timing Constraints

---

Latency	Symbol	DRAM cycles	Latency	Symbol	DRAM cycles
Precharge	$t_{RP}$	11	Activate to read/write	$t_{RCD}$	11
Read column address strobe	$CL$	11	Write column address strobe	$CWL$	8
Additive	$AL$	0	Activate to activate	$t_{RC}$	39
Activate to precharge	$t_{RAS}$	28	Read to precharge	$t_{RTP}$	6
Burst length	$t_{BL}$	4	Column address strobe to column address strobe	$t_{CCD}$	4
Activate to activate (different bank)	$t_{RRD}$	6	Four activate windows	$t_{FAW}$	24
Write to read	$t_{WTR}$	6	Write recovery	$t_{WR}$	12

Table 4. DDR3 1600 DRAM timing specifications

- From Lee et al., “[DRAM-Aware Last-Level Cache Writeback: Reducing Write-Caused Interference in Memory Systems](#),” HPS Technical Report, April 2010.

# Many Memory Timing Constraints

- Kim et al., "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," ISCA 2012.
- Lee et al., "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013.

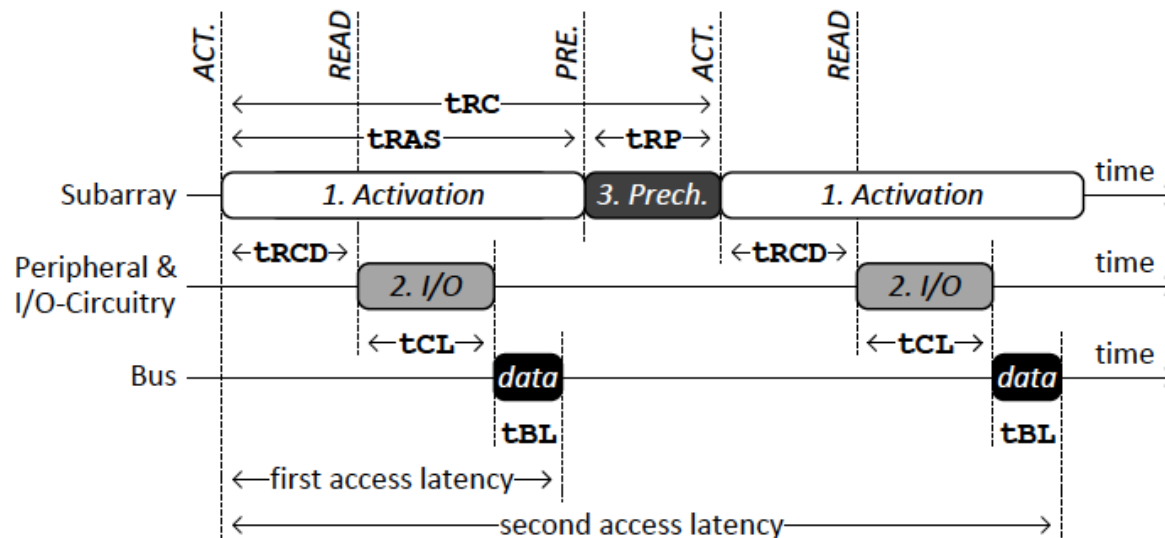
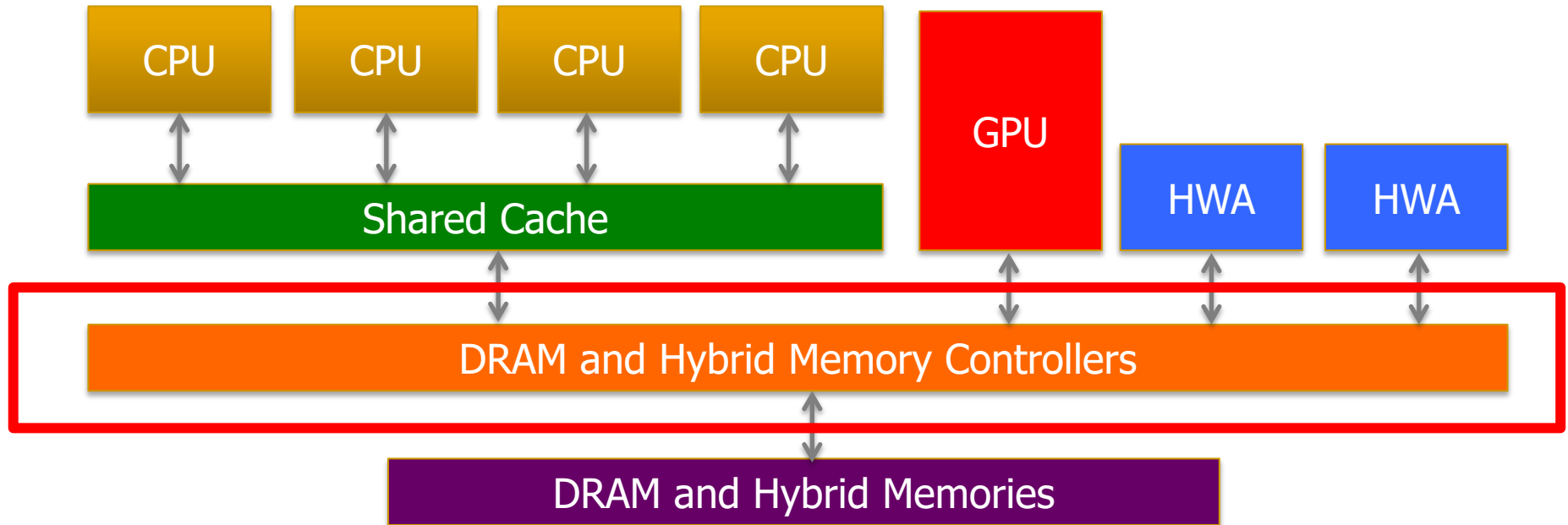


Figure 5. Three Phases of DRAM Access

Table 2. Timing Constraints (DDR3-1066) [43]

Phase	Commands	Name	Value
1	ACT → READ	$t_{RCD}$	15ns
	ACT → WRITE		
	ACT → PRE	$t_{RAS}$	37.5ns
2	READ → data	$t_{CL}$	15ns
	WRITE → data	$t_{CWL}$	11.25ns
	data burst	$t_{BL}$	7.5ns
3	PRE → ACT	$t_{RP}$	15ns
1 & 3	ACT → ACT	$t_{RC}$ ( $t_{RAS}+t_{RP}$ )	52.5ns

# Memory Controller Design Is Becoming More Difficult



- Heterogeneous agents: CPUs, GPUs, and HWAs
- Main memory interference between CPUs, GPUs, HWAs
- Many timing constraints for various memory types
- Many goals at the same time: performance, fairness, QoS, energy efficiency, ...

# Reality and Dream

---

- Reality: It difficult to design a policy that maximizes performance, QoS, energy-efficiency, ...
  - Too many things to think about
  - Continuously changing workload and system behavior
  
- Dream: Wouldn't it be nice if the DRAM controller automatically found a good scheduling policy on its own?

# Self-Optimizing DRAM Controllers

---

- Problem: DRAM controllers are difficult to design
  - It is difficult for human designers to design a policy that can adapt itself very well to different workloads and different system conditions
- Idea: A memory controller that adapts its scheduling policy to workload behavior and system conditions using machine learning.
- Observation: Reinforcement learning maps nicely to memory control.
- Design: Memory controller is a reinforcement learning agent
  - It dynamically and continuously learns and employs the best scheduling policy to maximize long-term performance.



# Self-Optimizing DRAM Controllers

---

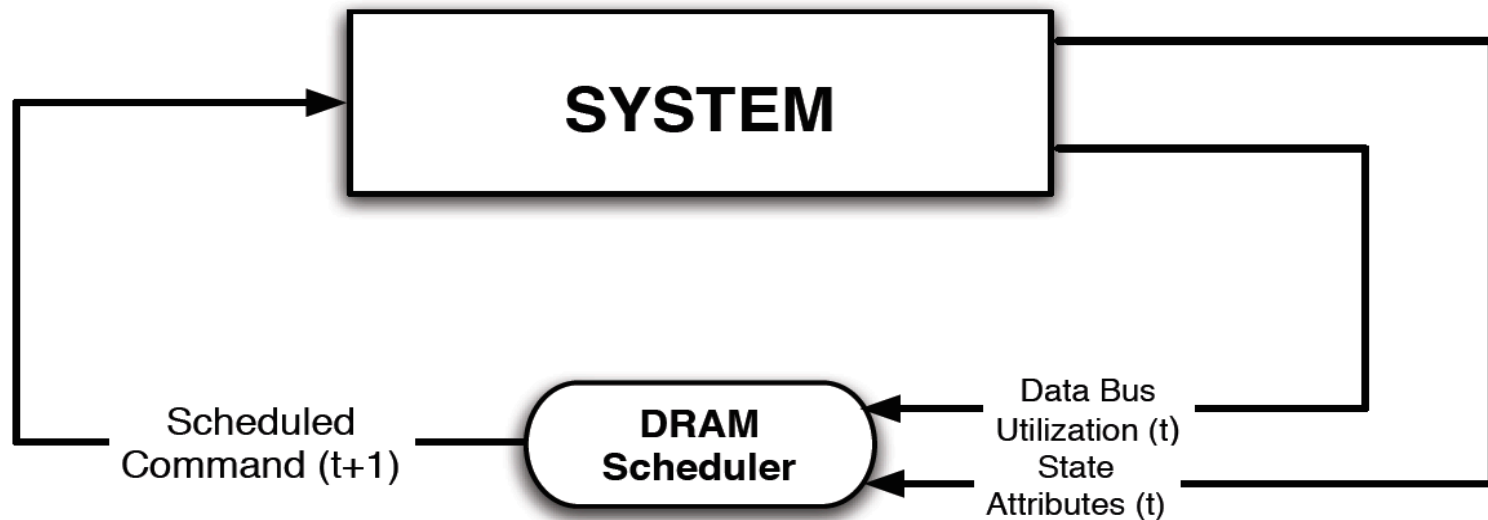


Goal: Learn to choose actions to maximize  $r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$  ( $0 \leq \gamma < 1$ )

**Figure 2:** (a) Intelligent agent based on reinforcement learning principles;

# Self-Optimizing DRAM Controllers

- Dynamically adapt the memory scheduling policy via interaction with the system at runtime
  - Associate system states and actions (commands) with long term reward values: **each action at a given state leads to a learned reward**
  - **Schedule command with highest estimated long-term reward value in each state**
  - **Continuously update reward values for  $\langle \text{state}, \text{action} \rangle$  pairs based on feedback from system**



# Self-Optimizing DRAM Controllers

- Engin Ipek, Onur Mutlu, José F. Martínez, and Rich Caruana,  
**"Self Optimizing Memory Controllers: A Reinforcement Learning Approach"**

*Proceedings of the 35th International Symposium on Computer Architecture (ISCA), pages 39-50, Beijing, China, June 2008.*

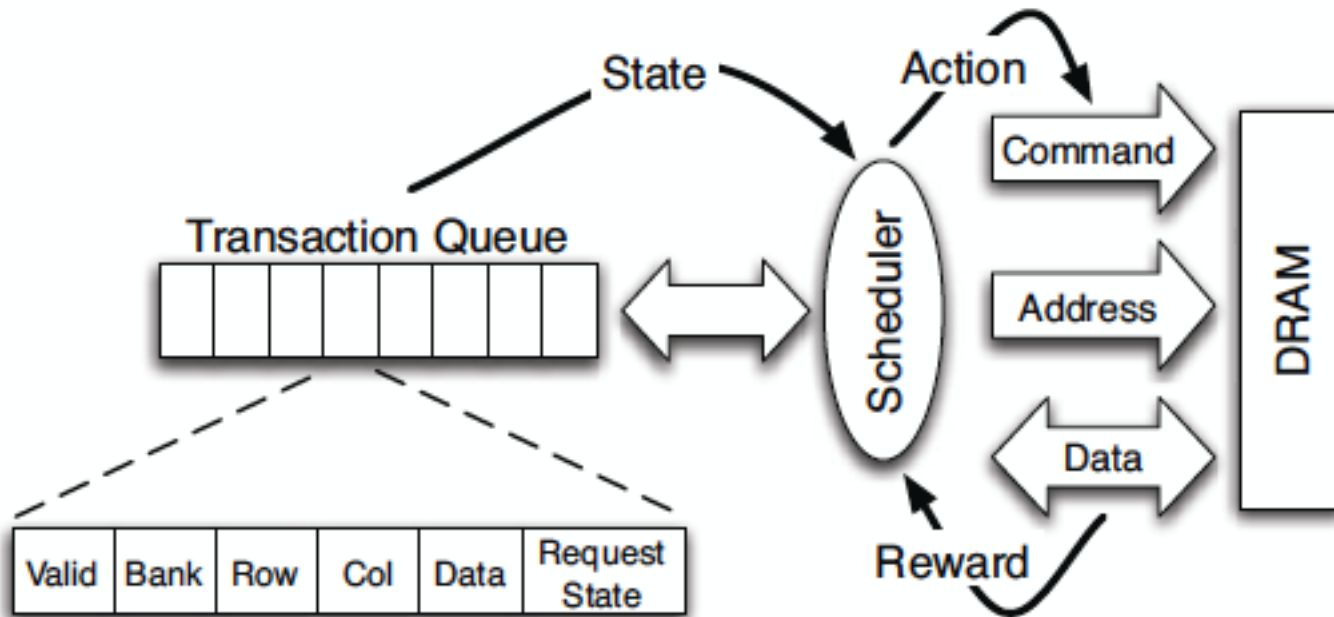


Figure 4: High-level overview of an RL-based scheduler.

# States, Actions, Rewards

---

## ❖ Reward function

- +1 for scheduling Read and Write commands
- 0 at all other times

Goal is to maximize long-term data bus utilization

## ❖ State attributes

- Number of reads, writes, and load misses in transaction queue
- Number of pending writes and ROB heads waiting for referenced row
- Request's relative ROB order

## ❖ Actions

- Activate
- Write
- Read - load miss
- Read - store miss
- Precharge - pending
- Precharge - preemptive
- NOP

# Performance Results

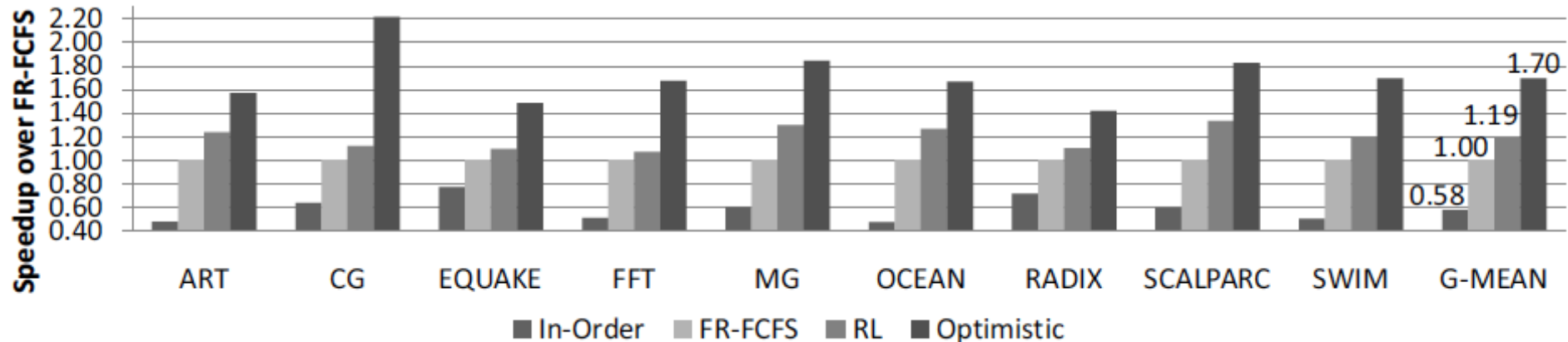


Figure 7: Performance comparison of in-order, FR-FCFS, RL-based, and optimistic memory controllers

**Large, robust performance improvements over many human-designed policies**

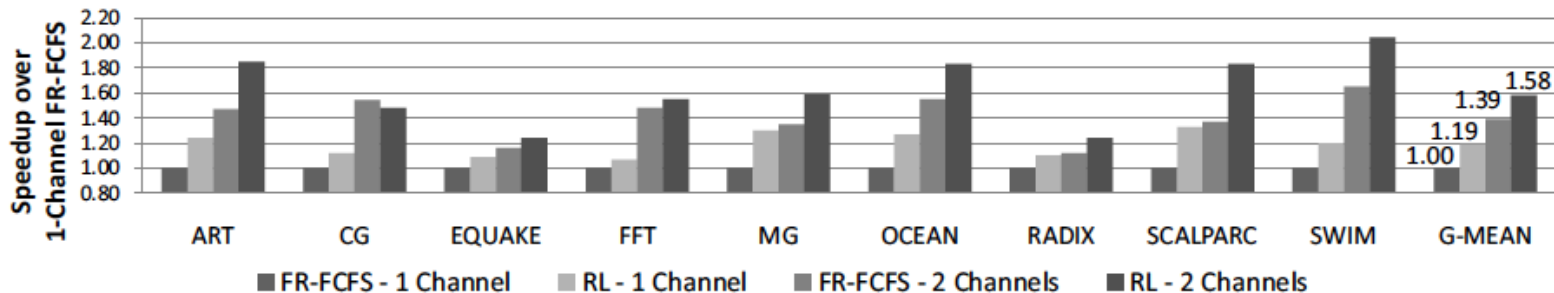


Figure 15: Performance comparison of FR-FCFS and RL-based memory controllers on systems with 6.4GB/s and 12.8GB/s peak DRAM bandwidth

# Self Optimizing DRAM Controllers

---

- + Continuous learning in the presence of changing environment

- + Reduced designer burden in finding a good scheduling policy.

Designer specifies:

- 1) What system variables might be useful

- 2) What target to optimize, but not how to optimize it

- How to specify different objectives? (e.g., fairness, QoS, ...)

- Hardware complexity?

- Design mindset and flow

# More on Self-Optimizing DRAM Controllers

---

- Engin Ipek, Onur Mutlu, José F. Martínez, and Rich Caruana,  
**"Self Optimizing Memory Controllers: A Reinforcement Learning Approach"**  
*Proceedings of the 35th International Symposium on Computer Architecture (ISCA)*, pages 39-50, Beijing, China, June 2008.

## Self-Optimizing Memory Controllers: A Reinforcement Learning Approach

Engin İpek<sup>1,2</sup>   Onur Mutlu<sup>2</sup>   José F. Martínez<sup>1</sup>   Rich Caruana<sup>1</sup>

<sup>1</sup>Cornell University, Ithaca, NY 14850 USA

<sup>2</sup>Microsoft Research, Redmond, WA 98052 USA

# An Intelligent Architecture

---

- Data-driven
  - Machine learns the “best” policies (how to do things)
- Sophisticated, workload-driven, changing, far-sighted policies
- Automatic data-driven policy learning
- All controllers are intelligent data-driven agents

**We need to rethink design  
(of all controllers)**



## Self-Optimizing (Data-Driven) Computing Architectures

# Corollaries: Architectures Today ...

---

- Architectures are **terrible at dealing with data**
  - ❑ Designed to mainly store and move data vs. to compute
  - ❑ They are **processor-centric** as opposed to **data-centric**
- Architectures are **terrible at taking advantage of vast amounts of data** (and metadata) available to them
  - ❑ Designed to make simple decisions, ignoring lots of data
  - ❑ They make **human-driven decisions** vs. **data-driven** decisions
- Architectures are **terrible at knowing and exploiting different properties of application data**
  - ❑ Designed to treat all data as the same
  - ❑ They make **component-aware decisions** vs. **data-aware**

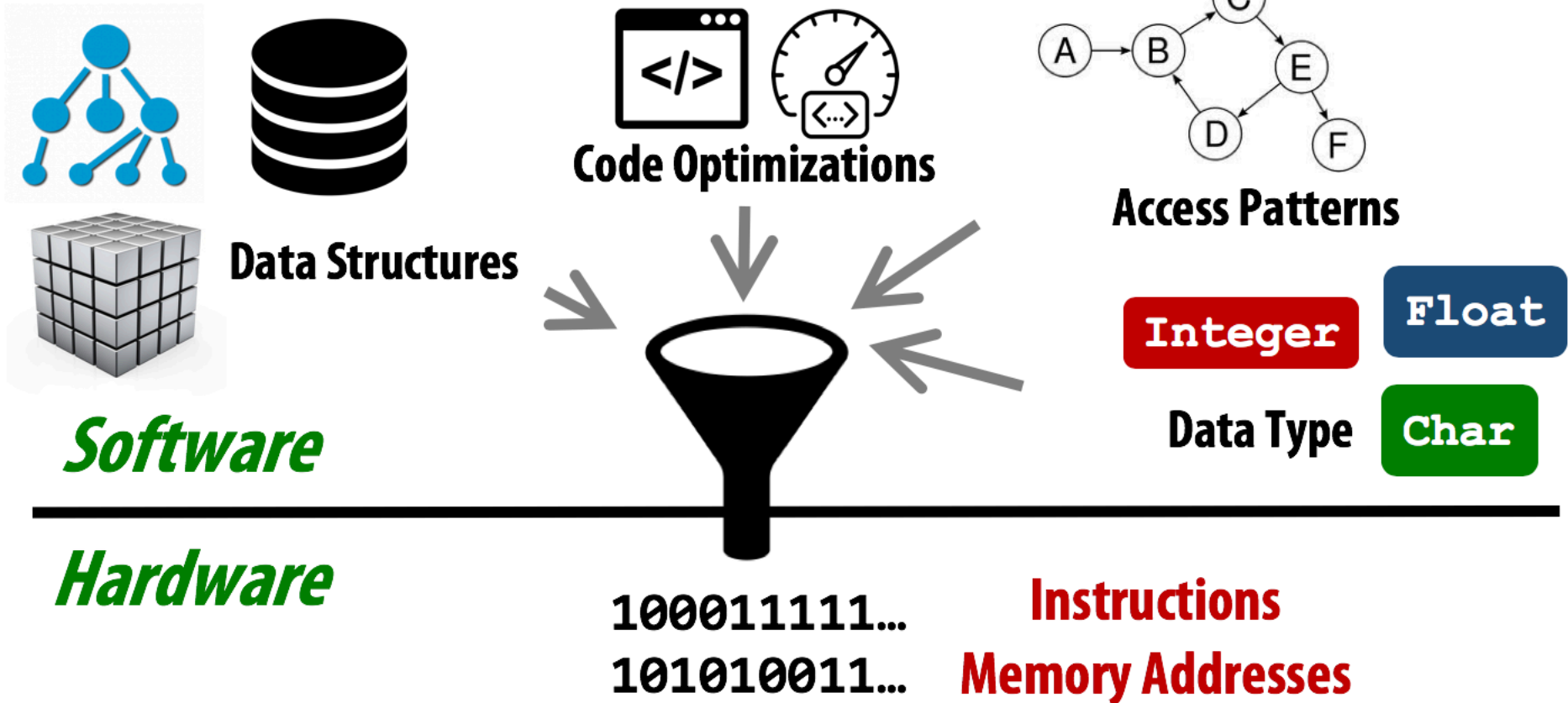
# Data-Aware Architectures

---

- A data-aware architecture understands what it can do with and to each piece of data
- It makes use of different properties of data to improve performance, efficiency and other metrics
  - Compressibility
  - Approximability
  - Locality
  - Sparsity
  - Criticality for Computation X
  - Access Semantics
  - ...

# One Problem: Limited Interfaces

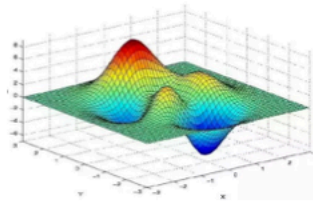
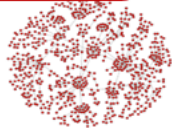
## Higher-level information is not visible to HW



# A Solution: More Expressive Interfaces

**Performance**

**Software**



**Functionality**



**ISA  
Virtual Memory**

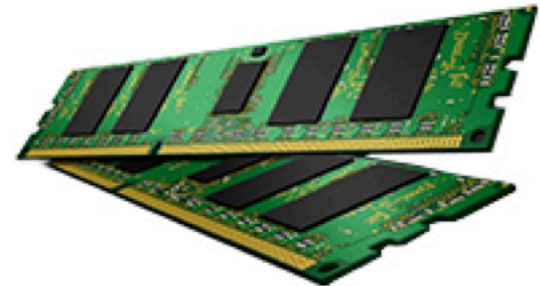
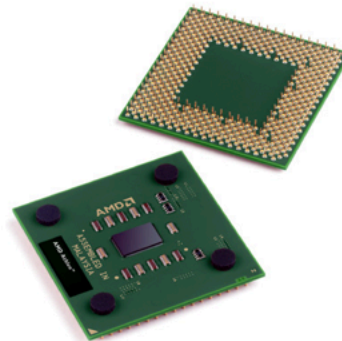
**Higher-level  
Program  
Semantics**

**Expressive  
Memory  
“XMem”**

**Hardware**



wiseGEEK



# Expressive (Memory) Interfaces

---

- Nandita Vijaykumar, Abhilasha Jain, Diptesh Majumdar, Kevin Hsieh, Gennady Pekhimenko, Eiman Ebrahimi, Nastaran Hajinazar, Phillip B. Gibbons and Onur Mutlu, **"A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory"**  
*Proceedings of the 45th International Symposium on Computer Architecture (ISCA)*, Los Angeles, CA, USA, June 2018.  
[[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]  
[[Lightning Talk Video](#)]

## A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory

Nandita Vijaykumar<sup>†§</sup> Abhilasha Jain<sup>†</sup> Diptesh Majumdar<sup>†</sup> Kevin Hsieh<sup>†</sup> Gennady Pekhimenko<sup>‡</sup>  
Eiman Ebrahimi<sup>⌘</sup> Nastaran Hajinazar<sup>†</sup> Phillip B. Gibbons<sup>†</sup> Onur Mutlu<sup>§†</sup>

<sup>†</sup>Carnegie Mellon University

<sup>‡</sup>University of Toronto

<sup>⌘</sup>NVIDIA

<sup>†</sup>Simon Fraser University

<sup>§</sup>ETH Zürich

# X-MeM Aids Many Optimizations

**Table 1: Summary of the example memory optimizations that XMem aids.**

Memory optimization	Example semantics provided by XMem (described in §3.3)	Example Benefits of XMem
Cache management	(i) Distinguishing between data structures or pools of similar data; (ii) Working set size; (iii) Data reuse	Enables: (i) applying different caching policies to different data structures or pools of data; (ii) avoiding cache thrashing by <i>knowing</i> the active working set size; (iii) bypassing/prioritizing data that has no/high reuse. (§5)
Page placement in DRAM e.g., [23, 24]	(i) Distinguishing between data structures; (ii) Access pattern; (iii) Access intensity	Enables page placement at the <i>data structure</i> granularity to (i) isolate data structures that have high row buffer locality and (ii) spread out concurrently-accessed irregular data structures across banks and channels to improve parallelism. (§6)
Cache/memory compression e.g., [25–32]	(i) Data type: integer, float, char; (ii) Data properties: sparse, pointer, data index	Enables using a <i>different compression algorithm</i> for each data structure based on data type and data properties, e.g., sparse data encodings, FP-specific compression, delta-based compression for pointers [27].
Data prefetching e.g., [33–36]	(i) Access pattern: strided, irregular, irregular but repeated (e.g., graphs), access stride; (ii) Data type: index, pointer	Enables (i) <i>highly accurate</i> software-driven prefetching while leveraging the benefits of hardware prefetching (e.g., by being memory bandwidth-aware, avoiding cache thrashing); (ii) using different prefetcher <i>types</i> for different data structures: e.g., stride [33], tile-based [20], pattern-based [34–37], data-based for indices/pointers [38, 39], etc.
DRAM cache management e.g., [40–46]	(i) Access intensity; (ii) Data reuse; (iii) Working set size	(i) Helps avoid cache thrashing by knowing working set size [44]; (ii) Better DRAM cache management via reuse behavior and access intensity information.
Approximation in memory e.g., [47–53]	(i) Distinguishing between pools of similar data; (ii) Data properties: tolerance towards approximation	Enables (i) each memory component to track how approximable data is (at a fine granularity) to inform approximation techniques; (ii) data placement in heterogeneous reliability memories [54].
Data placement: NUMA systems e.g., [55, 56]	(i) Data partitioning across threads (i.e., relating data to threads that access it); (ii) Read-Write properties	Reduces the need for profiling or data migration (i) to co-locate data with threads that access it and (ii) to identify Read-Only data, thereby enabling techniques such as replication.
Data placement: hybrid memories e.g., [16, 57, 58]	(i) Read-Write properties (Read-Only/Read-Write); (ii) Access intensity; (iii) Data structure size; (iv) Access pattern	Avoids the need for profiling/migration of data in hybrid memories to (i) effectively manage the asymmetric read-write properties in NVM (e.g., placing Read-Only data in the NVM) [16, 57]; (ii) make tradeoffs between data structure "hotness" and size to allocate fast/high bandwidth memory [14]; and (iii) leverage row-buffer locality in placement based on access pattern [45].
Managing NUCA systems e.g., [15, 59]	(i) Distinguishing pools of similar data; (ii) Access intensity; (iii) Read-Write or Private-Shared properties	(i) Enables using different cache policies for different data pools (similar to [15]); (ii) Reduces the need for reactive mechanisms that detect sharing and read-write characteristics to inform cache policies.

# Expressive (Memory) Interfaces for GPUs

---

- Nandita Vijaykumar, Eiman Ebrahimi, Kevin Hsieh, Phillip B. Gibbons and Onur Mutlu, **"The Locality Descriptor: A Holistic Cross-Layer Abstraction to Express Data Locality in GPUs"**  
*Proceedings of the 45th International Symposium on Computer Architecture (ISCA)*, Los Angeles, CA, USA, June 2018.  
[[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]  
[[Lightning Talk Video](#)]

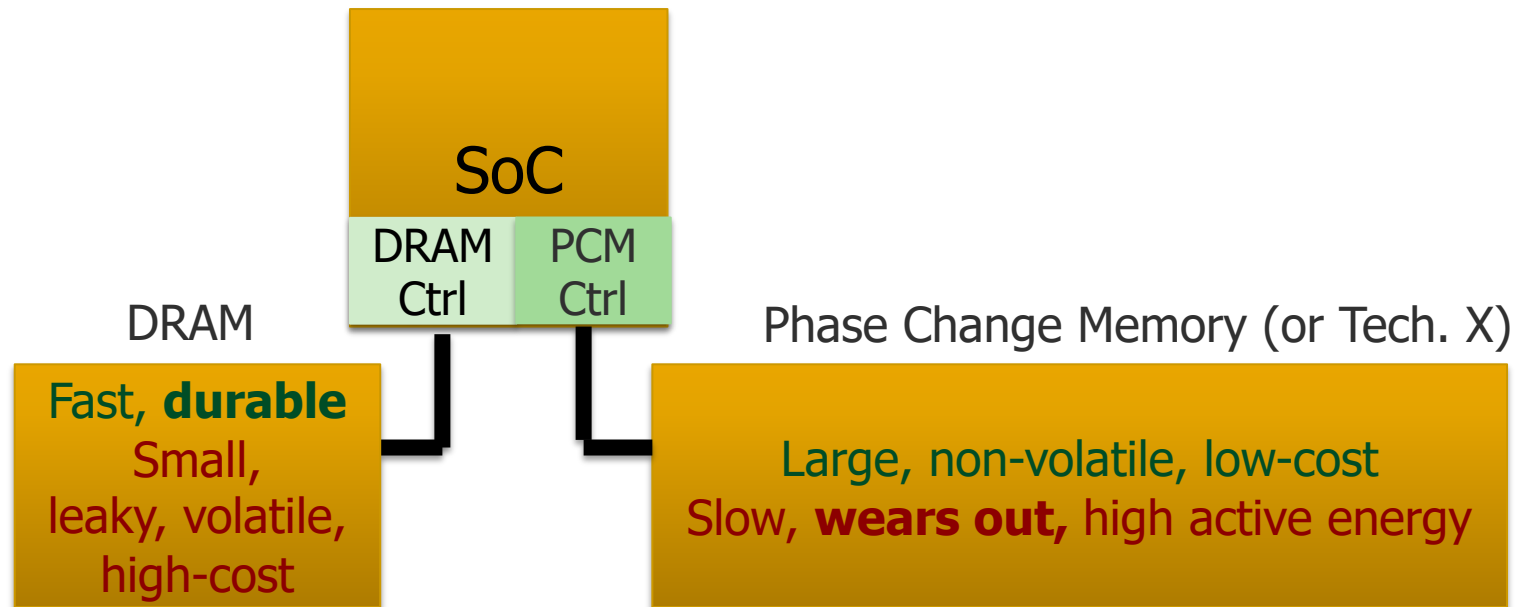
## The Locality Descriptor: A Holistic Cross-Layer Abstraction to Express Data Locality in GPUs

Nandita Vijaykumar <sup>†§</sup>	Eiman Ebrahimi <sup>‡</sup>	Kevin Hsieh <sup>†</sup>
Phillip B. Gibbons <sup>†</sup>	Onur Mutlu <sup>§†</sup>	
<sup>†</sup> Carnegie Mellon University	<sup>‡</sup> NVIDIA	<sup>§</sup> ETH Zürich



# An Example: Hybrid Memory Management

---



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.

# An Example: 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](#)]

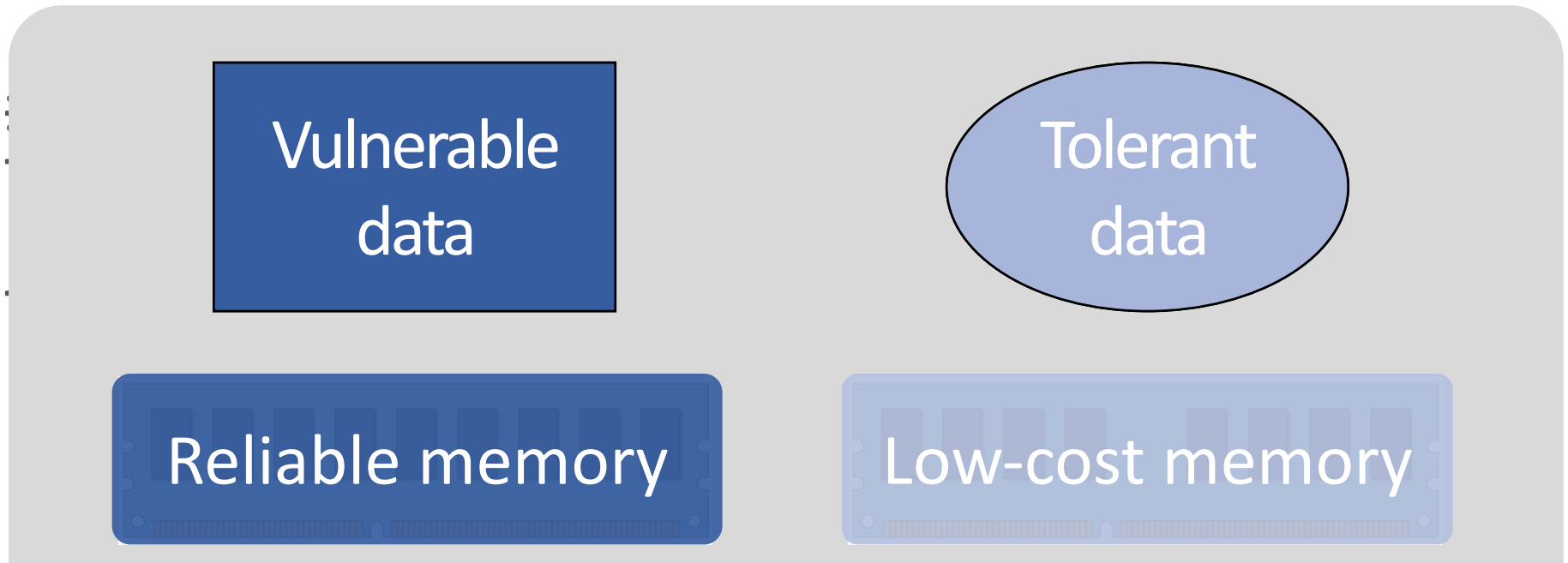
## Characterizing Application Memory Error Vulnerability to Optimize Datacenter Cost via Heterogeneous-Reliability Memory

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, bknessib, kvaid}@microsoft.com

# 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]

# Another Example: EDEN for DNNs

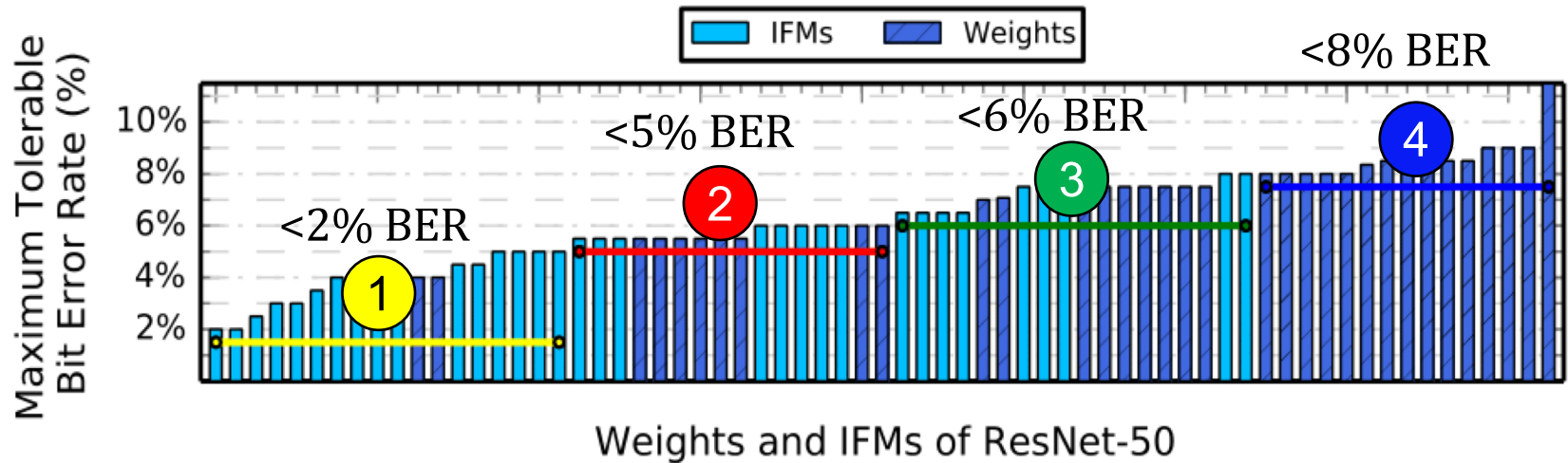
---

- Deep Neural Network evaluation is very DRAM-intensive (especially for large networks)
1. Some data and layers in DNNs are very tolerant to errors
  2. Reduce DRAM latency and voltage on such data and layers
  3. While still achieving a user-specified DNN accuracy target by making training DRAM-error-aware

**Data-aware management of DRAM latency and voltage for Deep Neural Network Inference**

# Example DNN Data Type to DRAM Mapping

Mapping example of ResNet-50:



**Map more error-tolerant DNN layers**  
**to DRAM partitions with lower voltage/latency**

**4 DRAM partitions** with different error rates

# EDEN: Data-Aware Efficient DNN Inference

---

- Skanda Koppula, Lois Orosa, A. Giray Yaglikci, Roknoddin Azizi, Taha Shahroodi, Konstantinos Kanellopoulos, and Onur Mutlu,  
**"EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM"**  
*Proceedings of the 52nd International Symposium on Microarchitecture (MICRO)*, Columbus, OH, USA, October 2019.  
[[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]  
[[Lightning Talk Video](#) (90 seconds)]

## EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM

Skanda Koppula   Lois Orosa   A. Giray Yağlıkçı  
Roknoddin Azizi   Taha Shahroodi   Konstantinos Kanellopoulos   Onur Mutlu  
ETH Zürich

Data-Aware  
(Expressive)

Computing Architectures

# Recap: Corollaries: Architectures Today

---

- Architectures are **terrible at dealing with data**
  - ❑ Designed to mainly store and move data vs. to compute
  - ❑ They are **processor-centric** as opposed to **data-centric**
- Architectures are **terrible at taking advantage of vast amounts of data** (and metadata) available to them
  - ❑ Designed to make simple decisions, ignoring lots of data
  - ❑ They make **human-driven decisions** vs. **data-driven decisions**
- Architectures are **terrible at knowing and exploiting different properties of application data**
  - ❑ Designed to treat all data as the same
  - ❑ They make **component-aware decisions** vs. **data-aware**

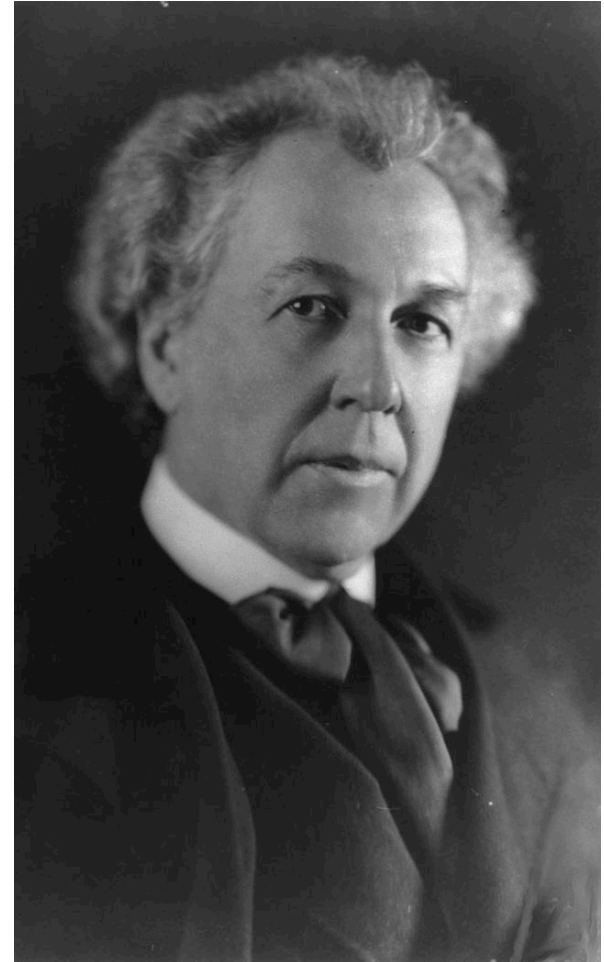


# Epilogue

# 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



# 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 **data handling (i.e., memory/storage) problem**
- **Design complete systems to be truly balanced, high-performance, and energy-efficient** → intelligent architectures
- **Data-centric, data-driven, data-aware**
- **This can**
  - ❑ Lead to **orders-of-magnitude** improvements
  - ❑ **Enable new applications & computing platforms**
  - ❑ **Enable better understanding of nature**
  - ❑ ...

**Data-centric**

**Data-driven**

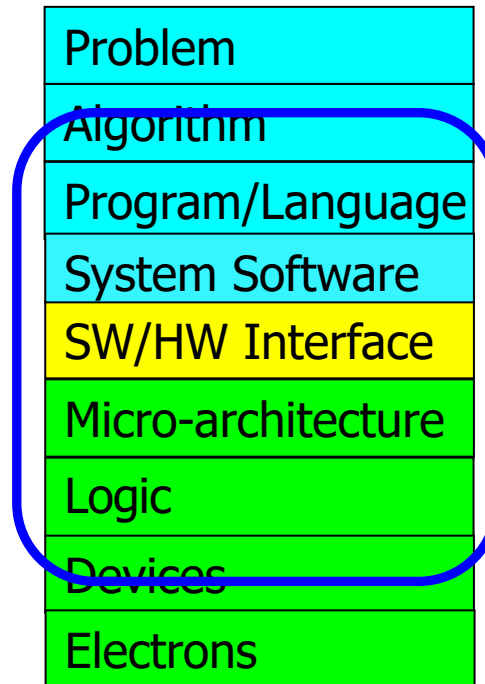
**Data-aware**





# We Need to Think Across the Entire Stack

---



**We can get there step by step**

# We Need to Exploit Good Principles

---

- Data-centric system design
- All components intelligent
- Better cross-layer communication, better interfaces
- Better-than-worst-case design
- Heterogeneity
- Flexibility, adaptability

**Open minds**

# PIM Review and Open Problems

---

## Processing Data Where It Makes Sense: Enabling In-Memory Computation

Onur Mutlu<sup>a,b</sup>, Saugata Ghose<sup>b</sup>, Juan Gómez-Luna<sup>a</sup>, Rachata Ausavarungnirun<sup>b,c</sup>

<sup>a</sup>*ETH Zürich*

<sup>b</sup>*Carnegie Mellon University*

<sup>c</sup>*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,  
**"Processing Data Where It Makes Sense: Enabling In-Memory  
Computation"**

*Invited paper in Microprocessors and Microsystems (**MICPRO**), June 2019.  
[arXiv version]*

# PIM Review and Open Problems (II)

---

## **A Workload and Programming Ease Driven Perspective of Processing-in-Memory**

Saugata Ghose<sup>†</sup>   Amirali Boroumand<sup>†</sup>   Jeremie S. Kim<sup>†§</sup>   Juan Gómez-Luna<sup>§</sup>   Onur Mutlu<sup>§†</sup>

<sup>†</sup>*Carnegie Mellon University*

<sup>§</sup>*ETH Zürich*

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu,

**"Processing-in-Memory: A Workload-Driven Perspective"**

*Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence, to appear in November 2019.*

[Preliminary arXiv version]

# Acknowledgments

---

## ■ My current and past students and postdocs

- ❑ Rachata Ausavarungnirun, Abhishek Bhowmick, Amirali Boroumand, Rui Cai, Yu Cai, Kevin Chang, Saugata Ghose, Kevin Hsieh, Tyler Huberty, Ben Jaiyen, Samira Khan, Jeremie Kim, Yoongu Kim, Yang Li, Jamie Liu, Lavanya Subramanian, Donghyuk Lee, Yixin Luo, Justin Meza, Gennady Pekhimenko, Vivek Seshadri, Lavanya Subramanian, Nandita Vijaykumar, HanBin Yoon, Jishen Zhao, ...

## ■ My collaborators

- ❑ Can Alkan, Chita Das, Phil Gibbons, Sriram Govindan, Norm Jouppi, Mahmut Kandemir, Mike Kozuch, Konrad Lai, Ken Mai, Todd Mowry, Yale Patt, Moinuddin Qureshi, Partha Ranganathan, Bikash Sharma, Kushagra Vaid, Chris Wilkerson, ...

# Funding Acknowledgments

---

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

# Acknowledgments

---

# SAFARI

*SAFARI Research Group*

*safari.ethz.ch*

Think BIG, Aim HIGH!

<https://safari.ethz.ch>

---

# Intelligent Architectures for Intelligent Machines

Onur Mutlu

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

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

20 November 2019

Khalifa System on Chip Center Open House Keynote

**SAFARI**

**ETH** zürich

**Carnegie Mellon**



# Backup Slides

# 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://safari.ethz.ch/memory\\_systems/ACACES2018/](https://safari.ethz.ch/memory_systems/ACACES2018/)
- ❑ <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)

# Longer Memory Course (~18 hours)

---

## ■ Tu Wien 2019

- ❑ Memory Systems and Memory-Centric Computing Systems
- ❑ Taught by Onur Mutlu June 12-19, 2019
- ❑ ~18 hours of lectures

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

- ❑ [https://safari.ethz.ch/memory\\_systems/TUWien2019](https://safari.ethz.ch/memory_systems/TUWien2019)
- ❑ [https://www.youtube.com/playlist?list=PL5Q2soXY2Zi\\_gntM55VoMIKlw7YrXOhbl](https://www.youtube.com/playlist?list=PL5Q2soXY2Zi_gntM55VoMIKlw7YrXOhbl)

## ■ All Papers are at:

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

# Some Overview Talks

---

[https://www.youtube.com/watch?v=kgiZISOcGFM&list=PL5Q2soXY2Zi8D\\_5MGV6EnXEJHnV2YFBJI](https://www.youtube.com/watch?v=kgiZISOcGFM&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI)

## ■ Future Computing Architectures

- [https://www.youtube.com/watch?v=kgiZISOcGFM&list=PL5Q2soXY2Zi8D\\_5MGV6EnXEJHnV2YFBJI&index=1](https://www.youtube.com/watch?v=kgiZISOcGFM&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI&index=1)

## ■ Enabling In-Memory Computation

- [https://www.youtube.com/watch?v=oHqsNbxgdzM&list=PL5Q2soXY2Zi8D\\_5MGV6EnXEJHnV2YFBJI&index=7](https://www.youtube.com/watch?v=oHqsNbxgdzM&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI&index=7)

## ■ Accelerating Genome Analysis

- [https://www.youtube.com/watch?v=hPnSmfwu2-A&list=PL5Q2soXY2Zi8D\\_5MGV6EnXEJHnV2YFBJI&index=9](https://www.youtube.com/watch?v=hPnSmfwu2-A&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI&index=9)

## ■ Rethinking Memory System Design

- [https://www.youtube.com/watch?v=F7xZLNMIY1E&list=PL5Q2soXY2Zi8D\\_5MGV6EnXEJHnV2YFBJI&index=3](https://www.youtube.com/watch?v=F7xZLNMIY1E&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJI&index=3)

# Reference Overview Paper I

---

## Processing Data Where It Makes Sense: Enabling In-Memory Computation

Onur Mutlu<sup>a,b</sup>, Saugata Ghose<sup>b</sup>, Juan Gómez-Luna<sup>a</sup>, Rachata Ausavarungnirun<sup>b,c</sup>

<sup>a</sup>*ETH Zürich*

<sup>b</sup>*Carnegie Mellon University*

<sup>c</sup>*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,  
**"Processing Data Where It Makes Sense: Enabling In-Memory  
Computation"**

*Invited paper in Microprocessors and Microsystems (**MICPRO**), June 2019.  
[arXiv version]*

# Reference Overview Paper II

---

## **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/pdf/1802.00320.pdf)]



# Reference Overview Paper III

---

- 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 IV

---

- 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 V

---

- 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

# Reference Overview Paper VII

---

- Onur Mutlu and Jeremie Kim,  
**"RowHammer: A Retrospective"**  
*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) Special Issue on Top Picks in Hardware and Embedded Security*, 2019.  
[[Preliminary arXiv version](#)]

## RowHammer: A Retrospective

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

# 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 (2018, 2017, 2015, 2013)**
- **Graduate Computer Architecture Course Materials (2018, 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 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

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...

🔗 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>**

# Some Solution Principles (So Far)

---

- Data-centric system design & intelligence spread around
  - Do not center everything around traditional computation units
- Better cooperation across layers of the system
  - Careful co-design of components and layers: system/arch/device
  - Better, richer, more expressive and flexible interfaces
- Better-than-worst-case design
  - Do not optimize for the worst case
  - Worst case should not determine the common case
- Heterogeneity in design (specialization, asymmetry)
  - Enables a more efficient design (No one size fits all)

# Simple PIM and PIM Adoption

# Two Key Questions in 3D-Stacked PIM

---

- What are the performance and energy benefits of using 3D-stacked memory as a coarse-grained accelerator?
  - By changing the entire system
  - By performing simple function offloading
- What is the minimal processing-in-memory support we can provide?
  - With minimal changes to system and programming



# PEI: PIM-Enabled Instructions (Ideas)

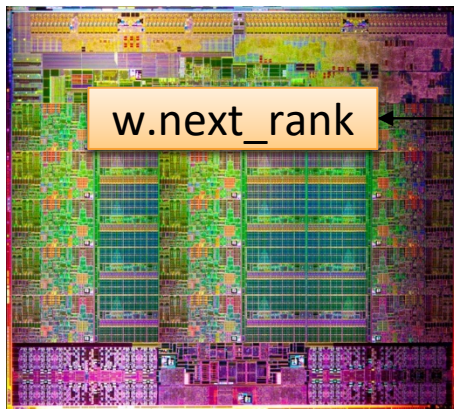
---

- **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;  
    }  
}
```

Host Processor



Main Memory



64 bytes in  
64 bytes out

Conventional Architecture

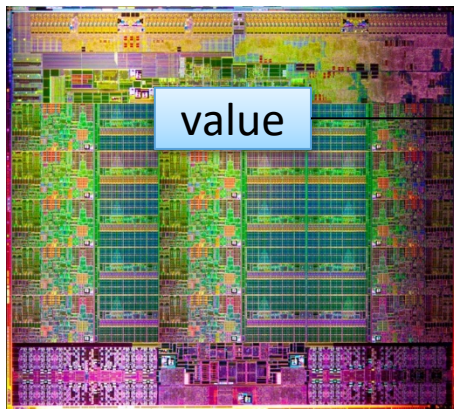
# Simple PIM Operations as ISA Extensions (III)

```
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 r1, (r2)

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

```
}
```

pfence();

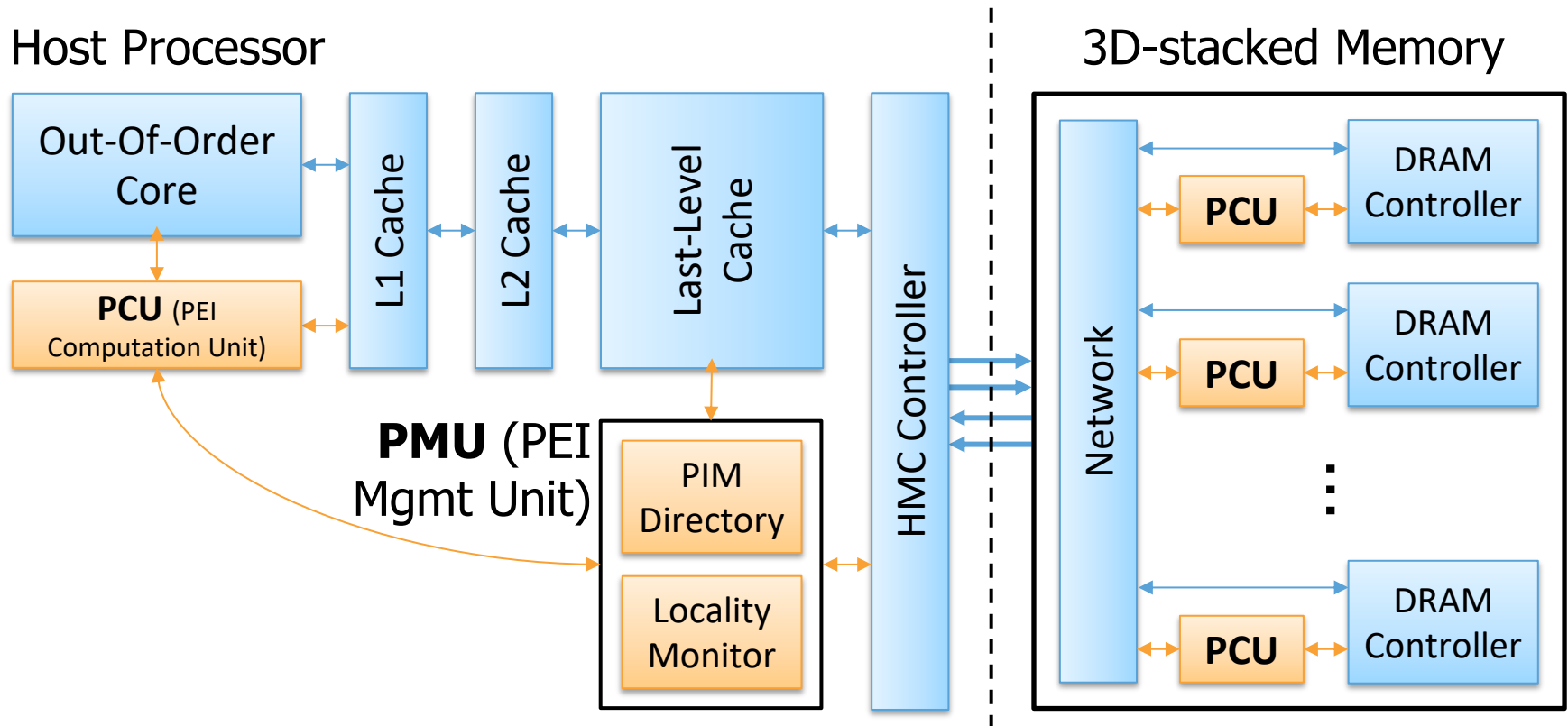
pfence

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

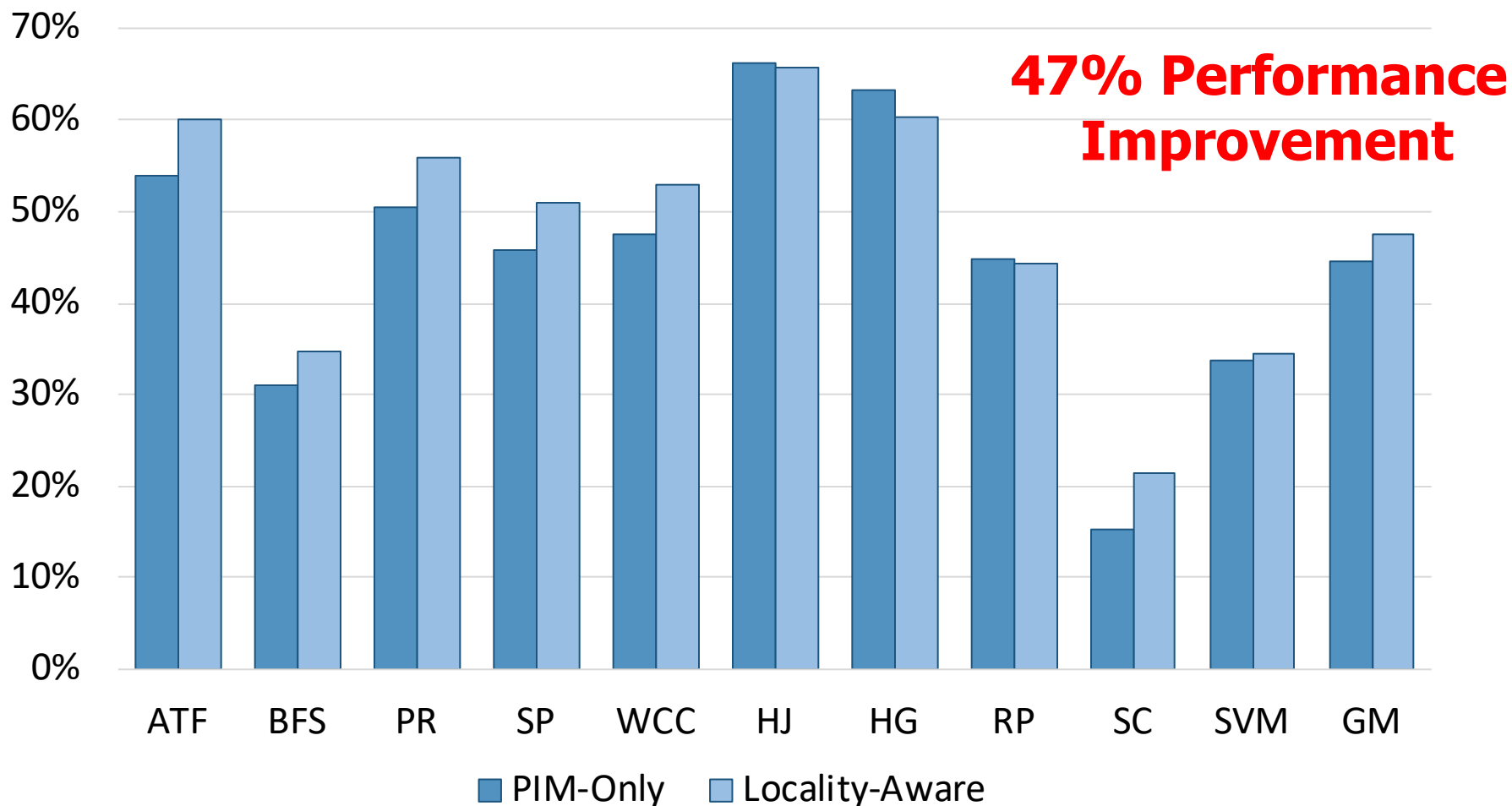
- Initial evaluations with **10 emerging data-intensive workloads**
  - ❑ Large-scale graph processing
  - ❑ In-memory data analytics
  - ❑ Machine learning and data mining
  - ❑ Three input sets (small, medium, large) for each workload to analyze the impact of data locality
- Pin-based cycle-level x86-64 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]

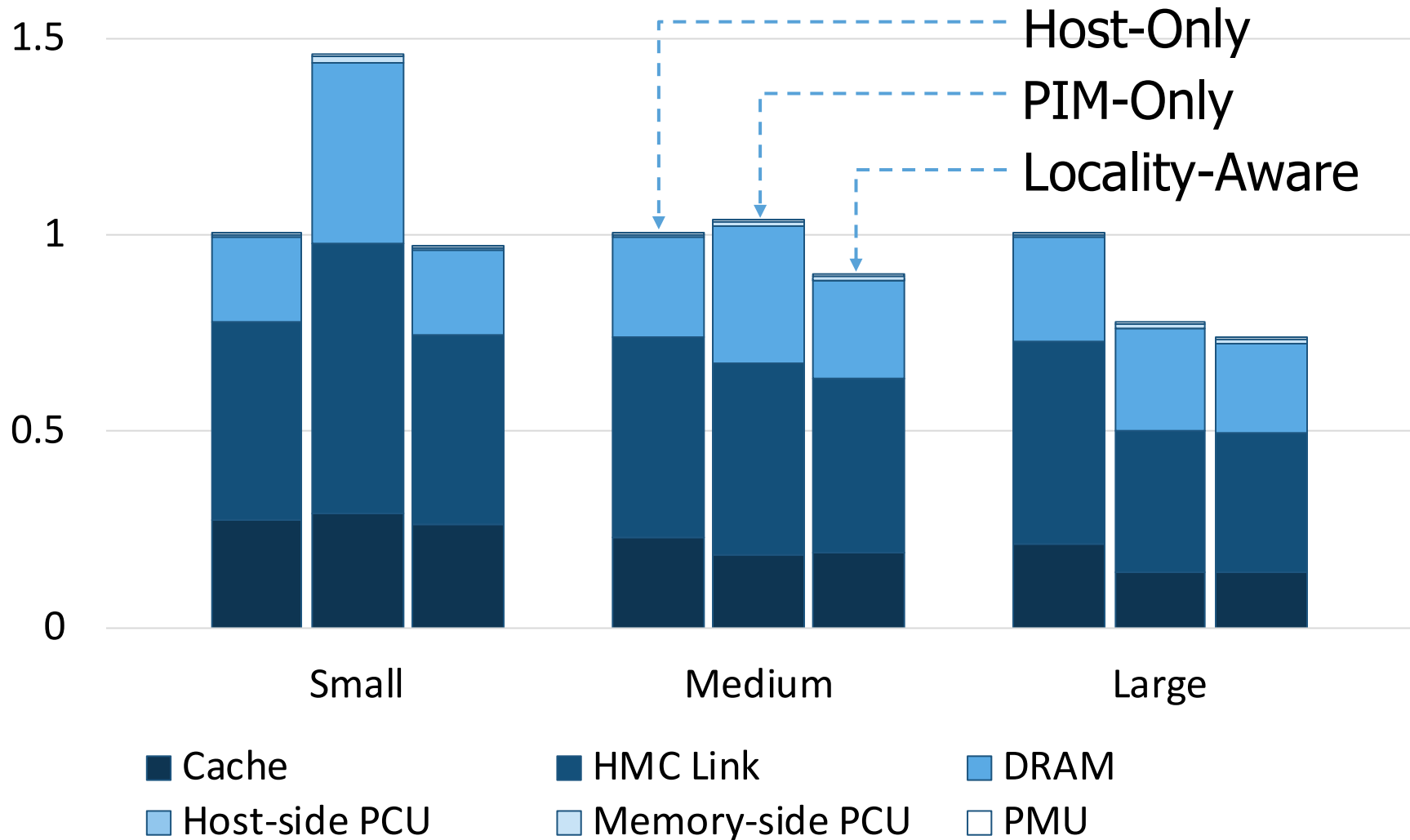
# PEI Performance Delta: Large Data Sets

(Large Inputs, Baseline: Host-Only)



# PEI Energy Consumption

**25% Energy Reduction**





# Simpler PIM: PIM-Enabled Instructions

---

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoungh 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>   Kiyoungh Choi

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

Seoul National University

<sup>†</sup>Carnegie Mellon University

# Simpler PIM: PIM-Enabled Instructions

---

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoungh 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>   Kiyoungh Choi

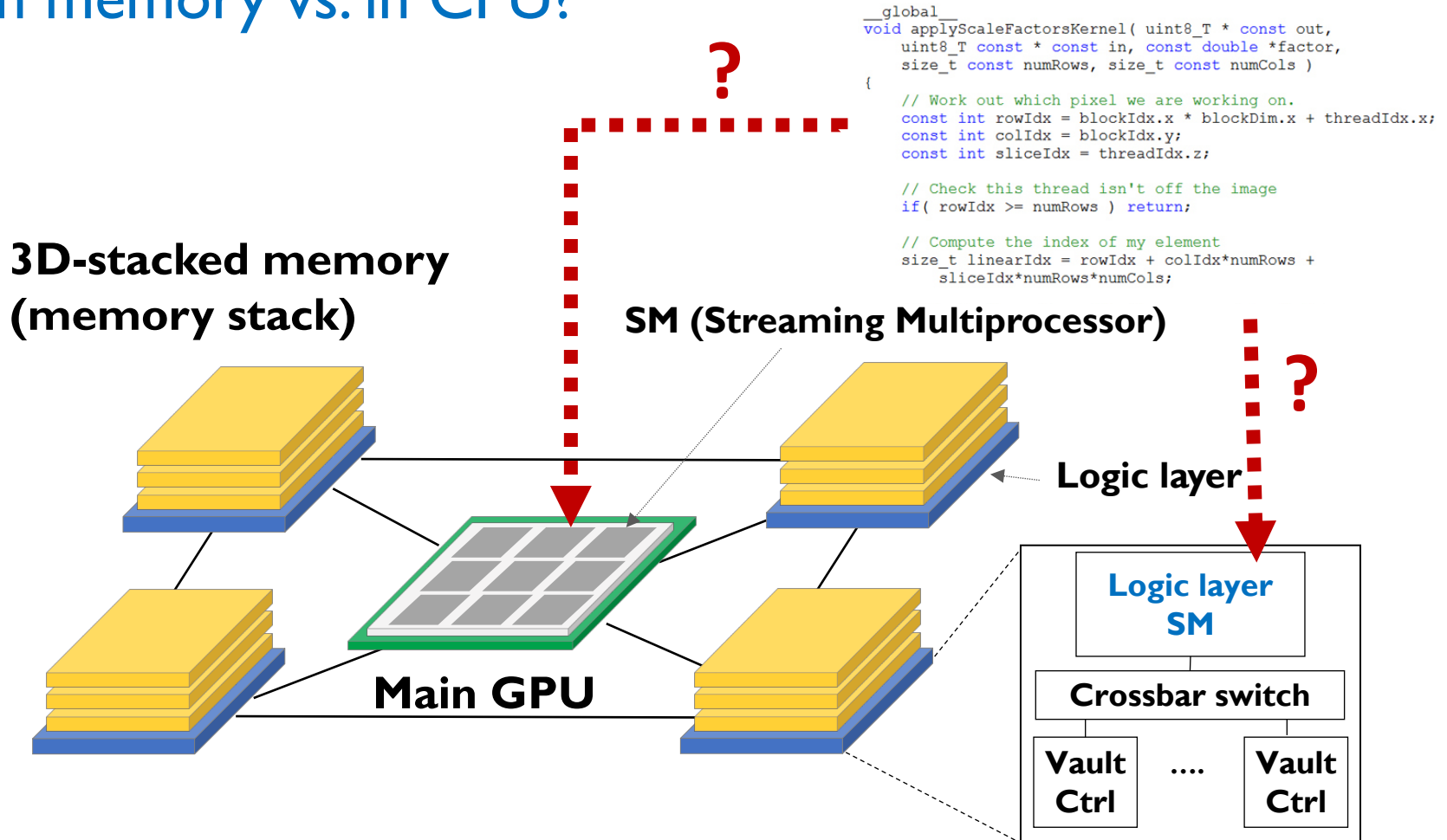
junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University

<sup>†</sup>Carnegie Mellon University

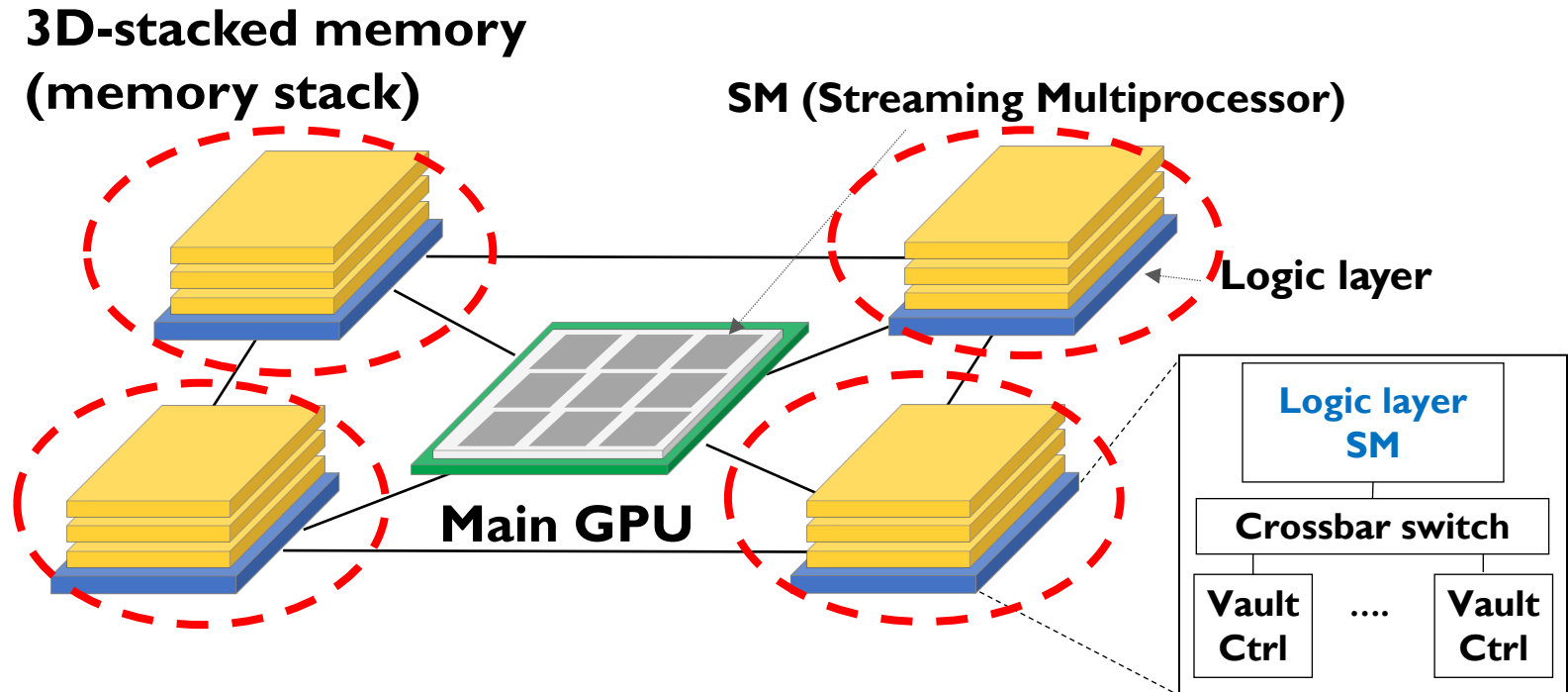
# Key Challenge 1: Code Mapping

- **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 \*KAIST <sup>§</sup>ETH Zürich

# How to Schedule Code?

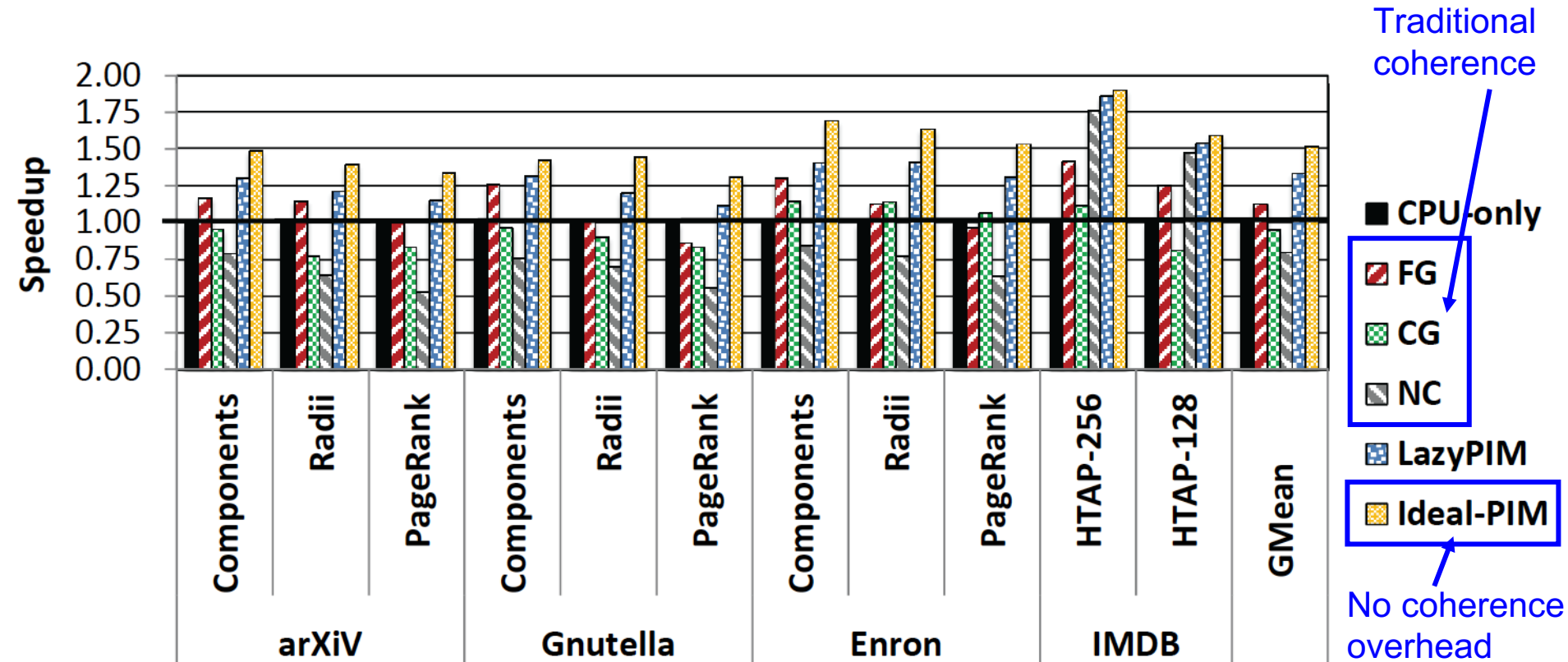
---

- 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 Maintain Coherence? (II)

---

- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,  
**"CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators"**  
*Proceedings of the 46th International Symposium on Computer Architecture (ISCA), Phoenix, AZ, USA, June 2019.*

## CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators

Amirali Boroumand<sup>†</sup>

Saugata Ghose<sup>†</sup>

Minesh Patel<sup>★</sup>

Hasan Hassan<sup>★</sup>

Brandon Lucia<sup>†</sup>

Rachata Ausavarungnirun<sup>†‡</sup>

Kevin Hsieh<sup>†</sup>

Nastaran Hajinazar<sup>◇†</sup>

Krishna T. Malladi<sup>§</sup>

Hongzhong Zheng<sup>§</sup>

Onur Mutlu<sup>★†</sup>

<sup>†</sup>Carnegie Mellon University

<sup>★</sup>ETH Zürich

<sup>‡</sup>KMUTNB

<sup>◇</sup>Simon Fraser University

<sup>§</sup>Samsung Semiconductor, Inc.

# 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

---

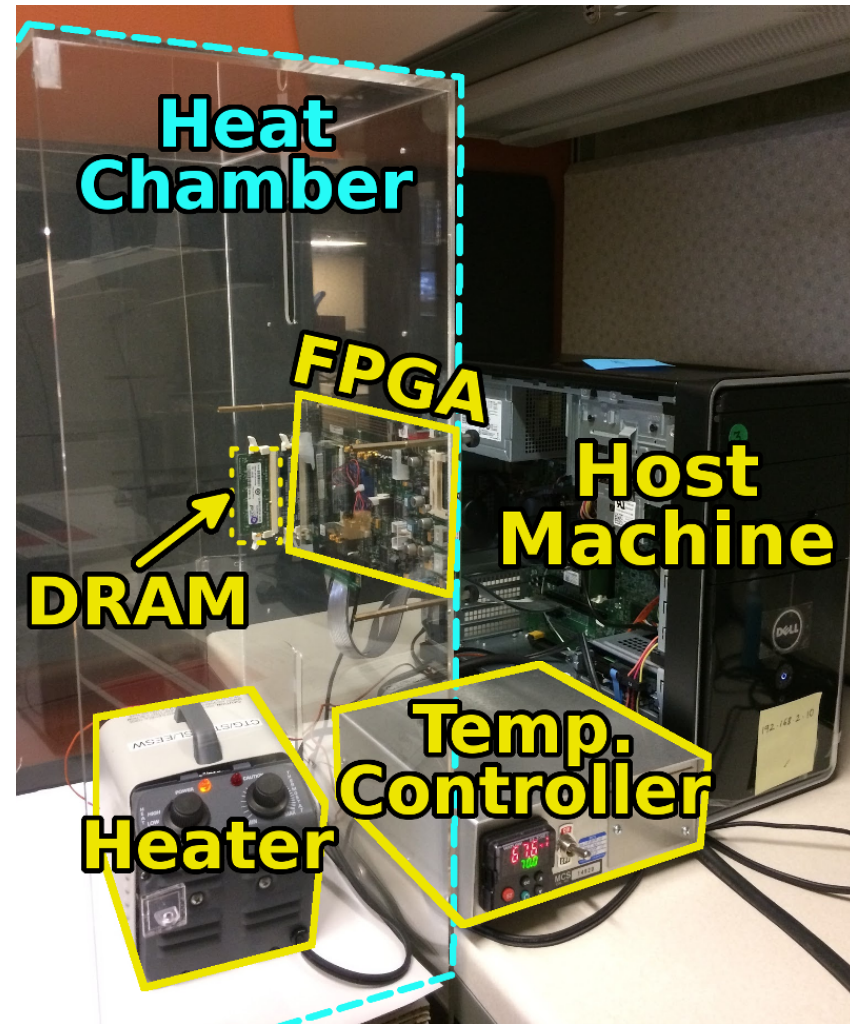
- **Ramulator** extended for PIM
  - Flexible and extensible DRAM simulator
  - Can model many different memory standards and proposals
  - Kim+, “**Ramulator: A Flexible and Extensible DRAM Simulator**”, IEEE CAL 2015.
  - <https://github.com/CMU-SAFARI/ramulator-pim>
  - <https://github.com/CMU-SAFARI/ramulator>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

## 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*

# Performance & Energy Models for PIM

---

- Gagandeep Singh, Juan Gomez-Luna, Giovanni Mariani, Geraldo F. Oliveira, Stefano Corda, Sander Stujik, Onur Mutlu, and Henk Corporaal, **"NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning"**  
*Proceedings of the 56th Design Automation Conference (DAC)*, Las Vegas, NV, USA, June 2019.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Poster \(pptx\)](#)] [[pdf](#)]  
[[Source Code for Ramulator-PIM](#)]

## NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning

Gagandeep Singh <sup>a,c</sup>	Juan Gómez-Luna <sup>b</sup>	Giovanni Mariani <sup>c</sup>	Geraldo F. Oliveira <sup>b</sup>
Stefano Corda <sup>a,c</sup>	Sander Stuijk <sup>a</sup>	Onur Mutlu <sup>b</sup>	Henk Corporaal <sup>a</sup>
<sup>a</sup> Eindhoven University of Technology		<sup>b</sup> ETH Zürich	<sup>c</sup> IBM Research - Zurich



# 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 \(pdf\)](https://arxiv.org/abs/1801.00000)

## 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/pdf/1802.00320.pdf)]

# PIM Review and Open Problems

---

## Processing Data Where It Makes Sense: Enabling In-Memory Computation

Onur Mutlu<sup>a,b</sup>, Saugata Ghose<sup>b</sup>, Juan Gómez-Luna<sup>a</sup>, Rachata Ausavarungnirun<sup>b,c</sup>

<sup>a</sup>*ETH Zürich*

<sup>b</sup>*Carnegie Mellon University*

<sup>c</sup>*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,  
**"Processing Data Where It Makes Sense: Enabling In-Memory  
Computation"**

*Invited paper in Microprocessors and Microsystems (**MICPRO**), June 2019.  
[arXiv version]*

# PIM Review and Open Problems (II)

---

## **A Workload and Programming Ease Driven Perspective of Processing-in-Memory**

Saugata Ghose<sup>†</sup>   Amirali Boroumand<sup>†</sup>   Jeremie S. Kim<sup>†§</sup>   Juan Gómez-Luna<sup>§</sup>   Onur Mutlu<sup>§†</sup>

<sup>†</sup>*Carnegie Mellon University*

<sup>§</sup>*ETH Zürich*

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu,

**"Processing-in-Memory: A Workload-Driven Perspective"**

*Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence, to appear in November 2019.*

[Preliminary arXiv version]

# Computing Architectures with Minimal Data Movement