

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

Onur Mutlu

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

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

December 1, 2017

Institute of Neuroinformatics Seminar

# Brief Self Introduction



## ■ Onur Mutlu

- Professor @ ETH Zurich CS, since September'15, started May'16
- Strecker Professor @ Carnegie Mellon University ECE (CS), 2009-2016, 2016-...
- PhD from UT-Austin, worked @ Google, VMware, Microsoft Research, Intel, AMD
- <https://people.inf.ethz.ch/omutlu/>
- [omutlu@gmail.com](mailto:omutlu@gmail.com) (Best way to reach me)
- Publications: <https://people.inf.ethz.ch/omutlu/projects.htm>

## ■ Research, Education, Consulting in

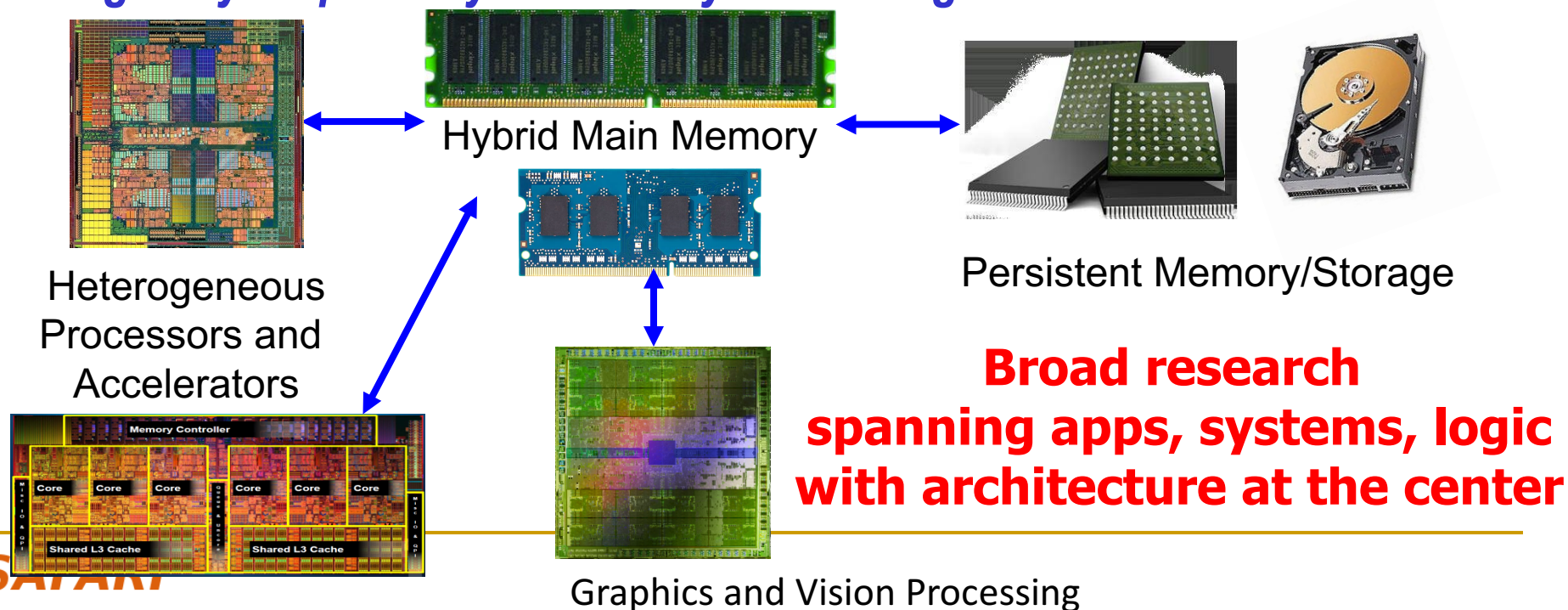
- Computer architecture and systems, bioinformatics
- Memory and storage systems, emerging technologies
- Many-core systems, heterogeneous systems, core design
- Interconnects
- Hardware/software interaction and co-design (PL, OS, Architecture)
- Predictable and QoS-aware systems
- Hardware fault tolerance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



# Current Research Focus Areas

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

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



# Four Key Current Directions

---

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

# In-Memory DNA Sequence Analysis

---

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

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

Jeremie S. Kim<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>

# New Genome Sequencing Technologies

---

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

**Damla Senol Cali<sup>1,\*</sup>, Jeremie Kim<sup>1,3</sup>, Saugata Ghose<sup>1</sup>, Can Alkan<sup>2\*</sup>  
and Onur Mutlu<sup>3,1\*</sup>**

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

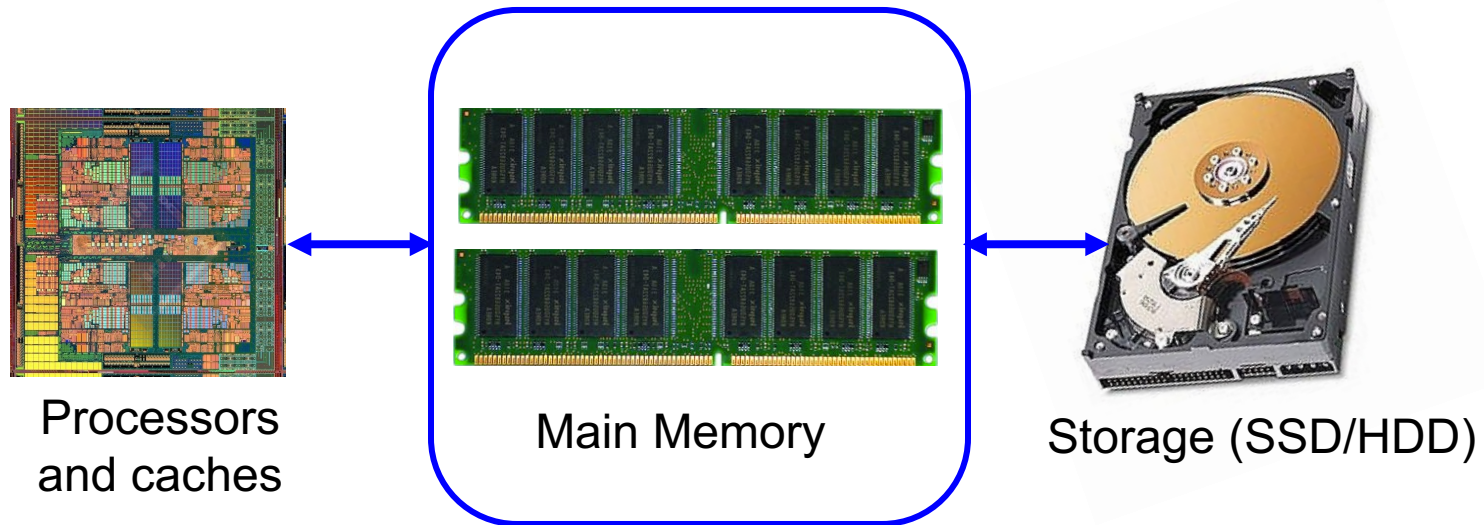
<sup>2</sup>Department of Computer Engineering, Bilkent University, Bilkent, Ankara, Turkey

<sup>3</sup>Department of Computer Science, Systems Group, ETH Zürich, Zürich, Switzerland

# Memory & Storage

# The Main Memory System

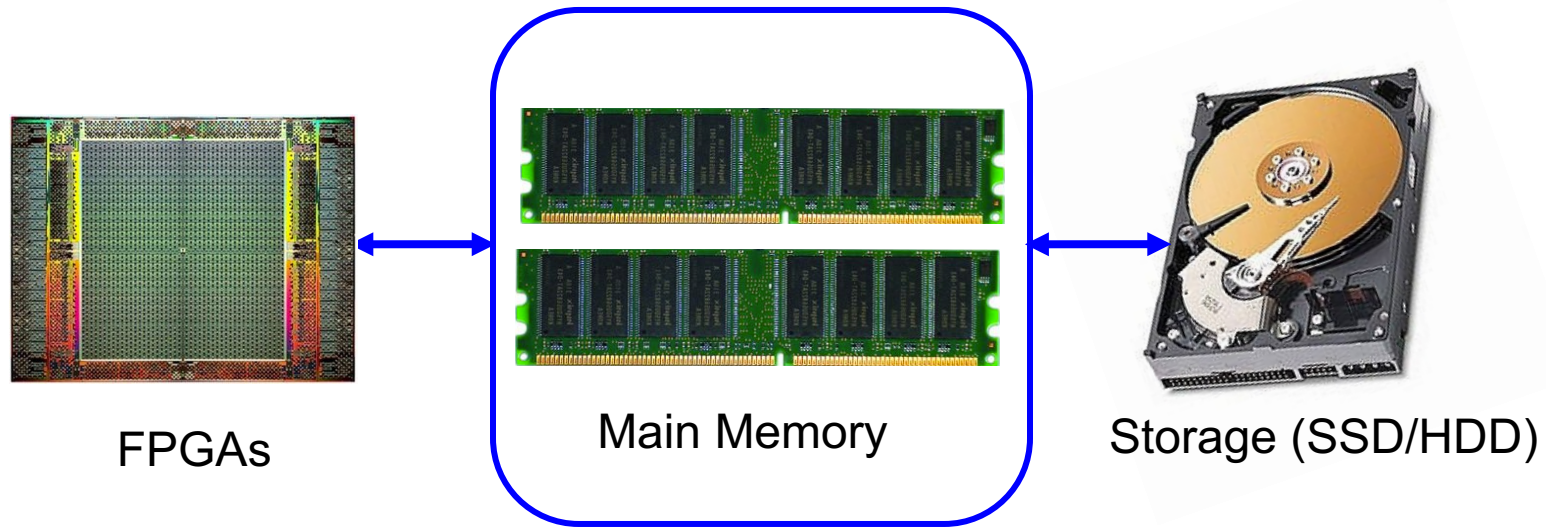
---



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

# The Main Memory System

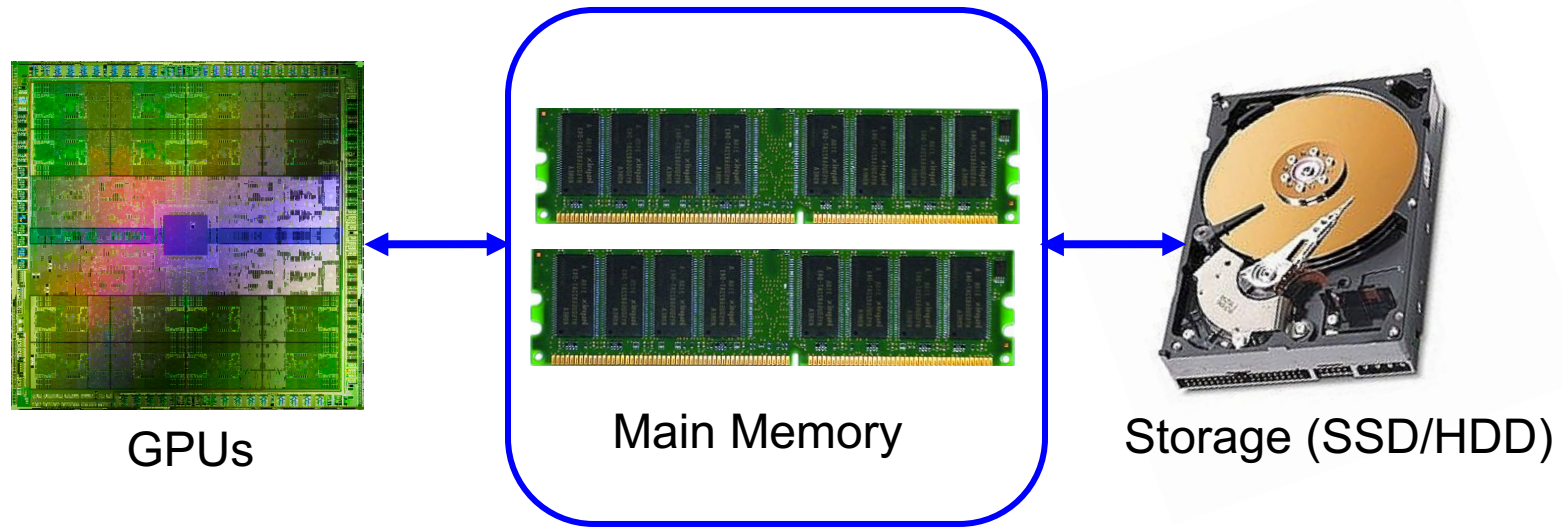
---



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



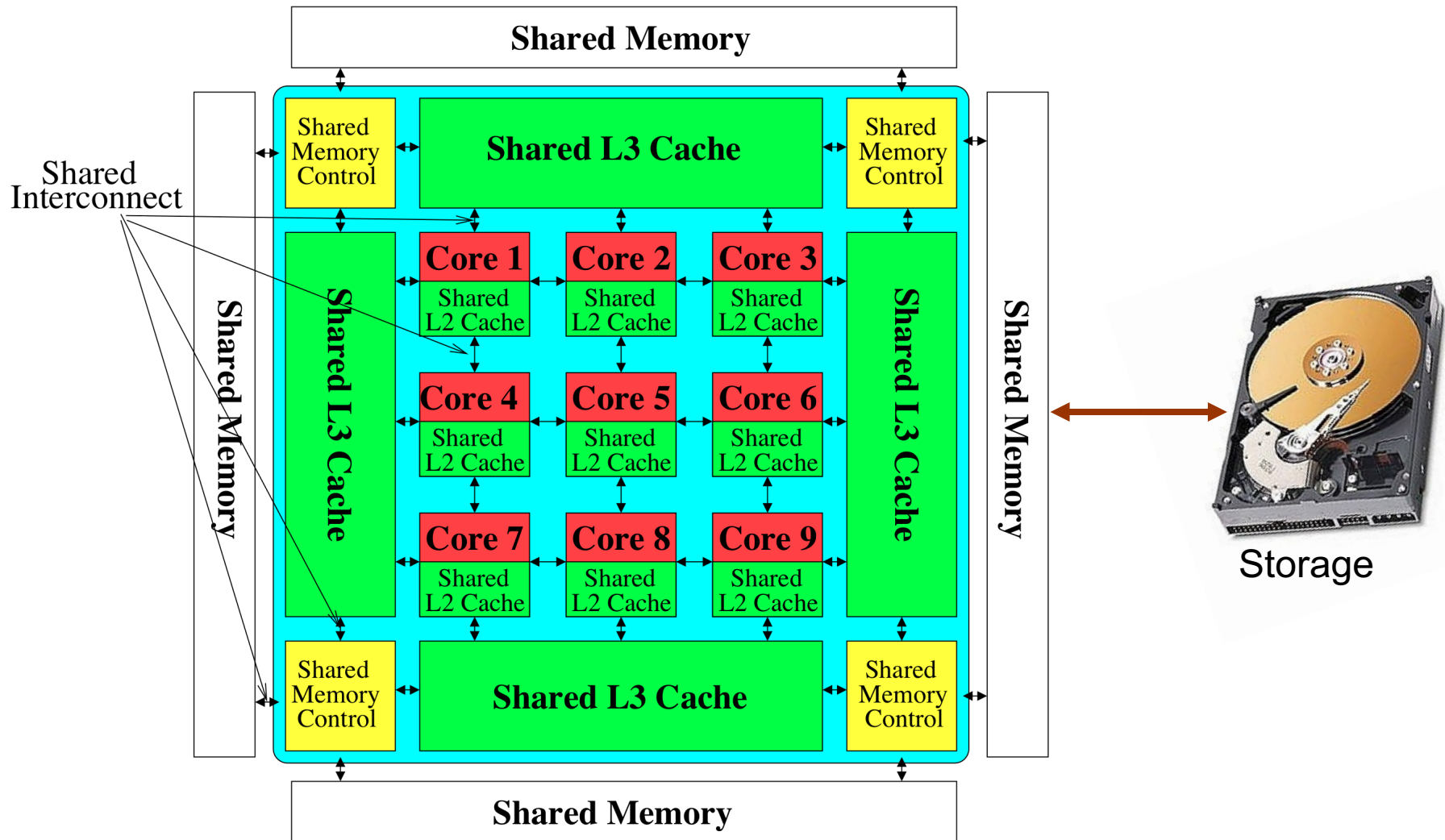
# The Main Memory System



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



# Memory System: A *Shared Resource* View



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

# State of the Main Memory System

---

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

# Major Trends Affecting Main Memory (I)

---

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

# Major Trends Affecting Main Memory (II)

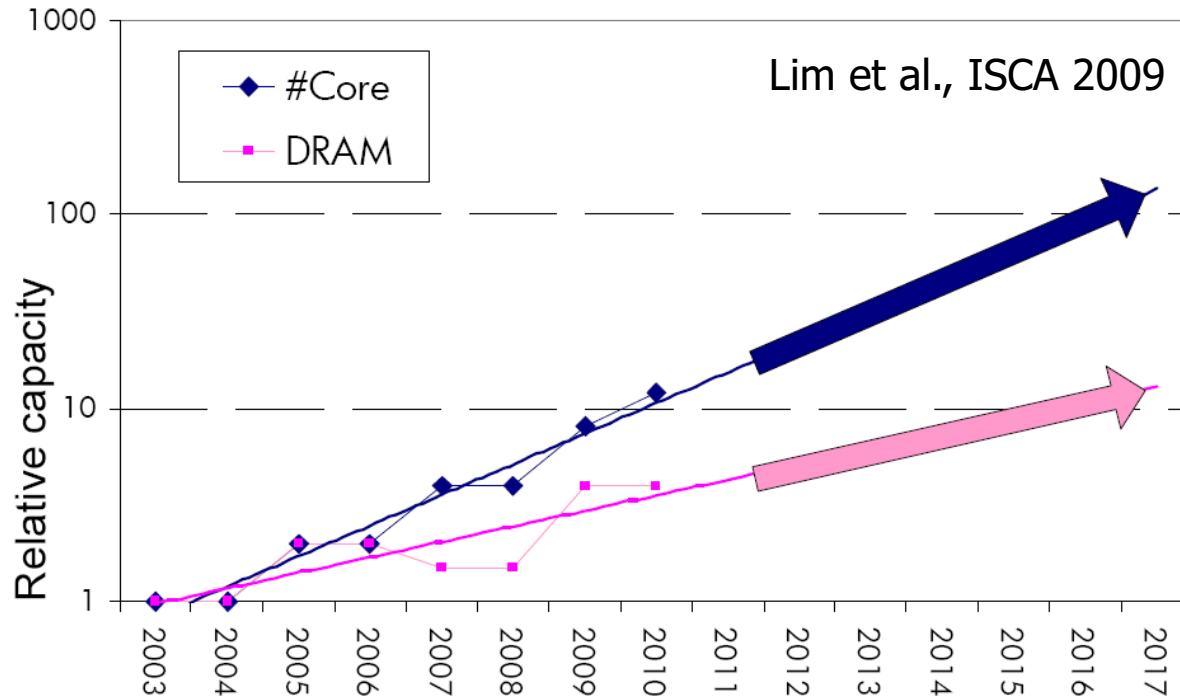
---

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

# Example: The Memory Capacity Gap

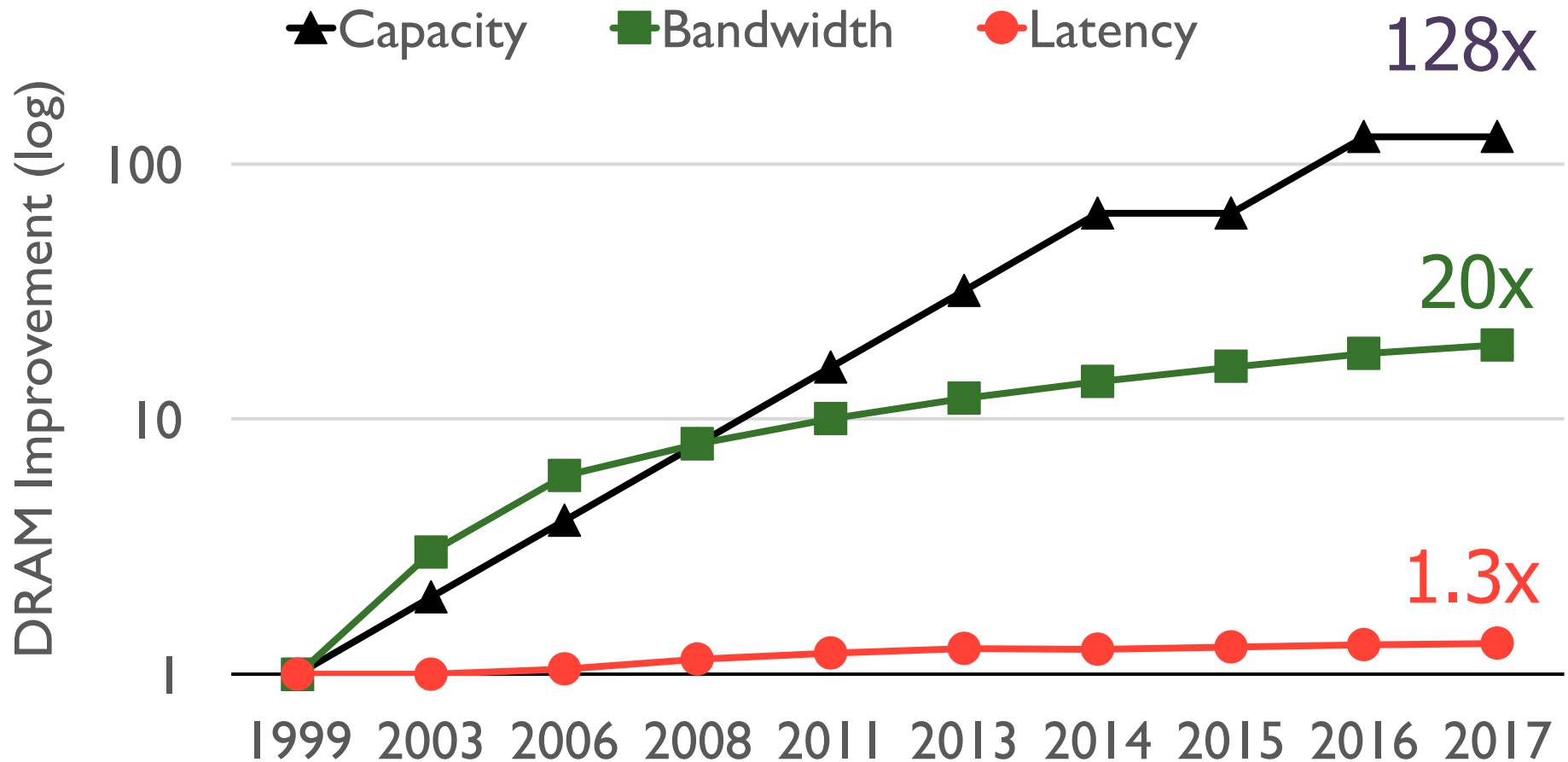
Core count doubling ~ every 2 years

DRAM DIMM capacity doubling ~ every 3 years



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

# Example: Capacity, Bandwidth & Latency



Memory latency remains almost constant

# DRAM Latency Is Critical for Performance

---



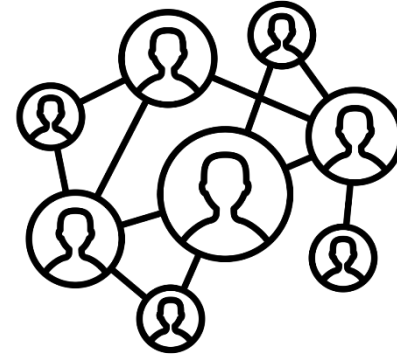
## In-memory Databases

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



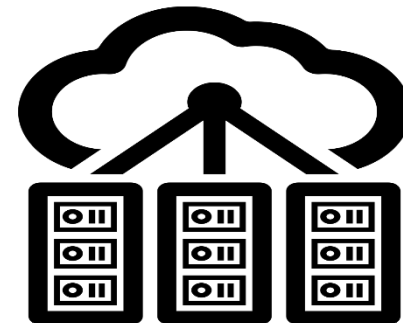
## In-Memory Data Analytics

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



## Graph/Tree Processing

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



## Datacenter Workloads

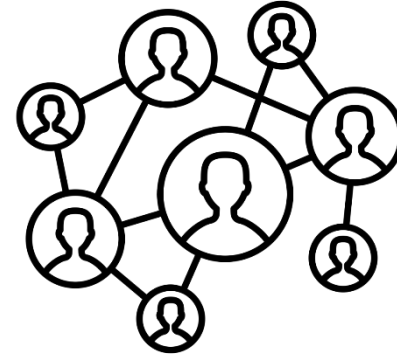
[Kanev+ (Google), ISCA'15]

# DRAM Latency Is Critical for Performance

---



**In-memory Databases**



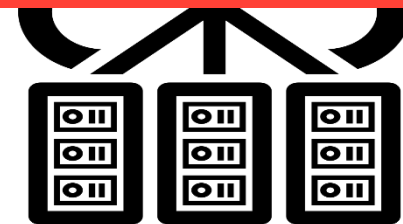
**Graph/Tree Processing**

Long memory latency → performance bottleneck



**In-Memory Data Analytics**

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



**Datacenter Workloads**

[Kanev+ (Google), ISCA'15]



# Major Trends Affecting Main Memory (III)

---

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

# Major Trends Affecting Main Memory (IV)

---

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

# Major Trends Affecting Main Memory (V)

---

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

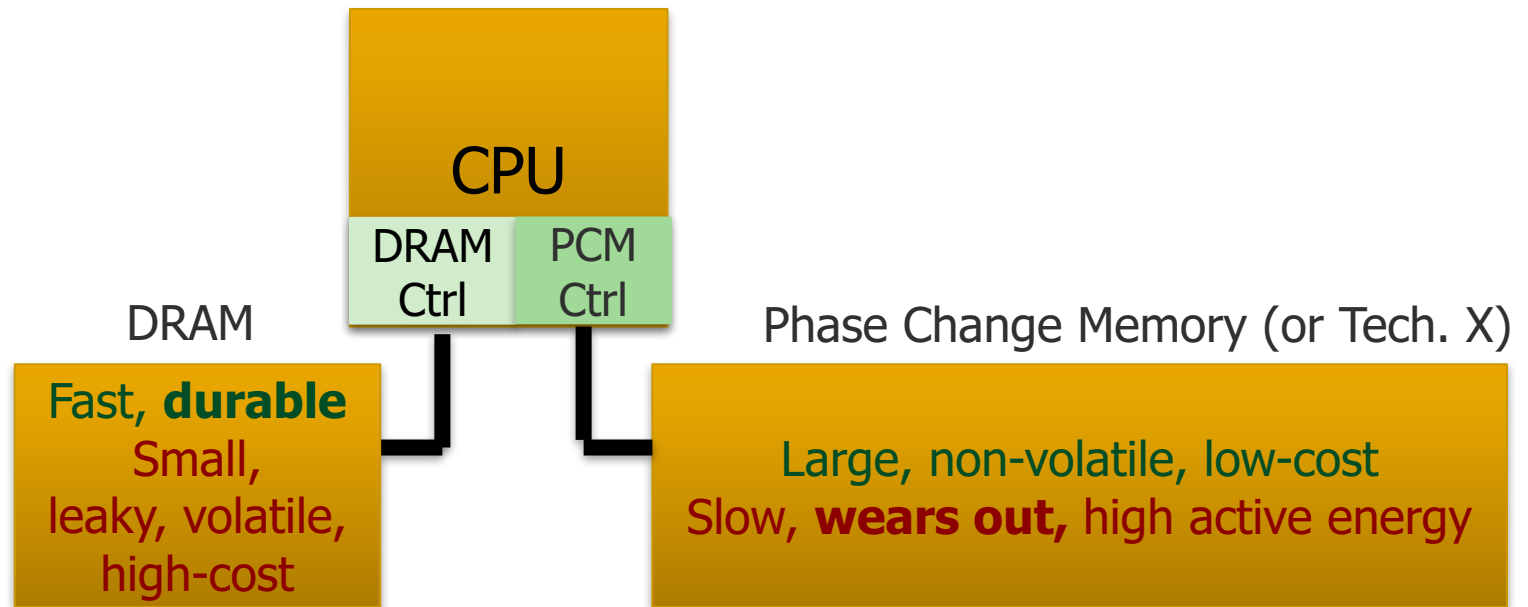

# Major Trends Affecting Main Memory (V)

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

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

# Major Trend: Hybrid Main Memory

---



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

Meza+, "[Enabling Efficient and Scalable Hybrid Memories](#)," IEEE Comp. Arch. Letters, 2012.

Yoon+, "[Row Buffer Locality Aware Caching Policies for Hybrid Memories](#)," ICCD 2012 Best Paper Award.

## Main Memory Needs Intelligent Controllers

# Industry Is Writing Papers About It, Too

## DRAM Process Scaling Challenges

### ❖ Refresh

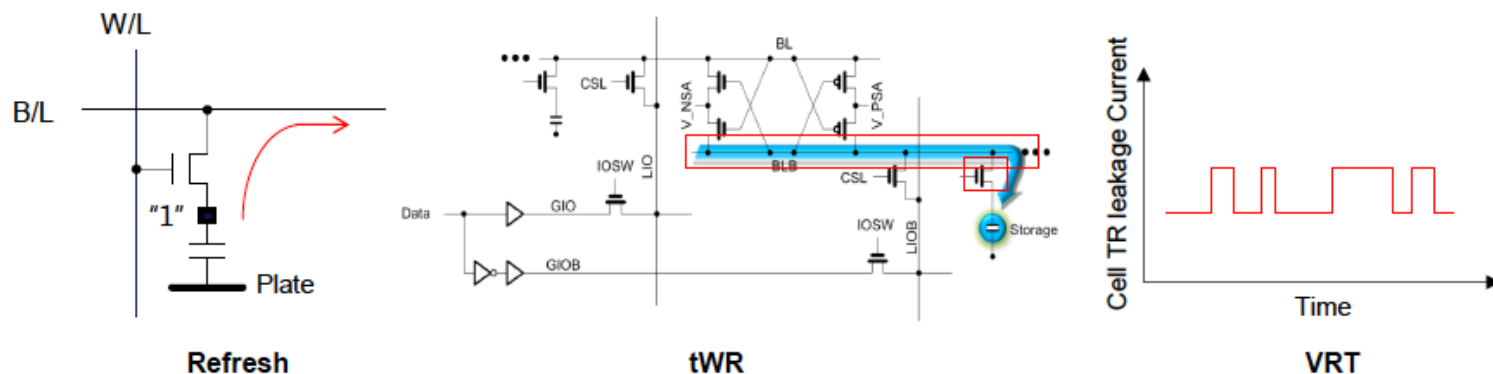
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance
- Leakage current of cell access transistors increasing

### ❖ tWR

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

### ❖ VRT

- Occurring more frequently with cell capacitance decreasing



# Call for Intelligent Memory Controllers

## DRAM Process Scaling Challenges

### ❖ Refresh

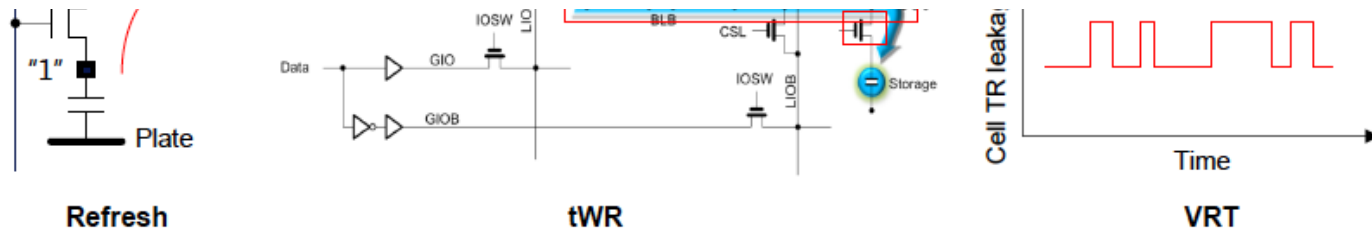
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

THE MEMORY FORUM 2014

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

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

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





# Agenda

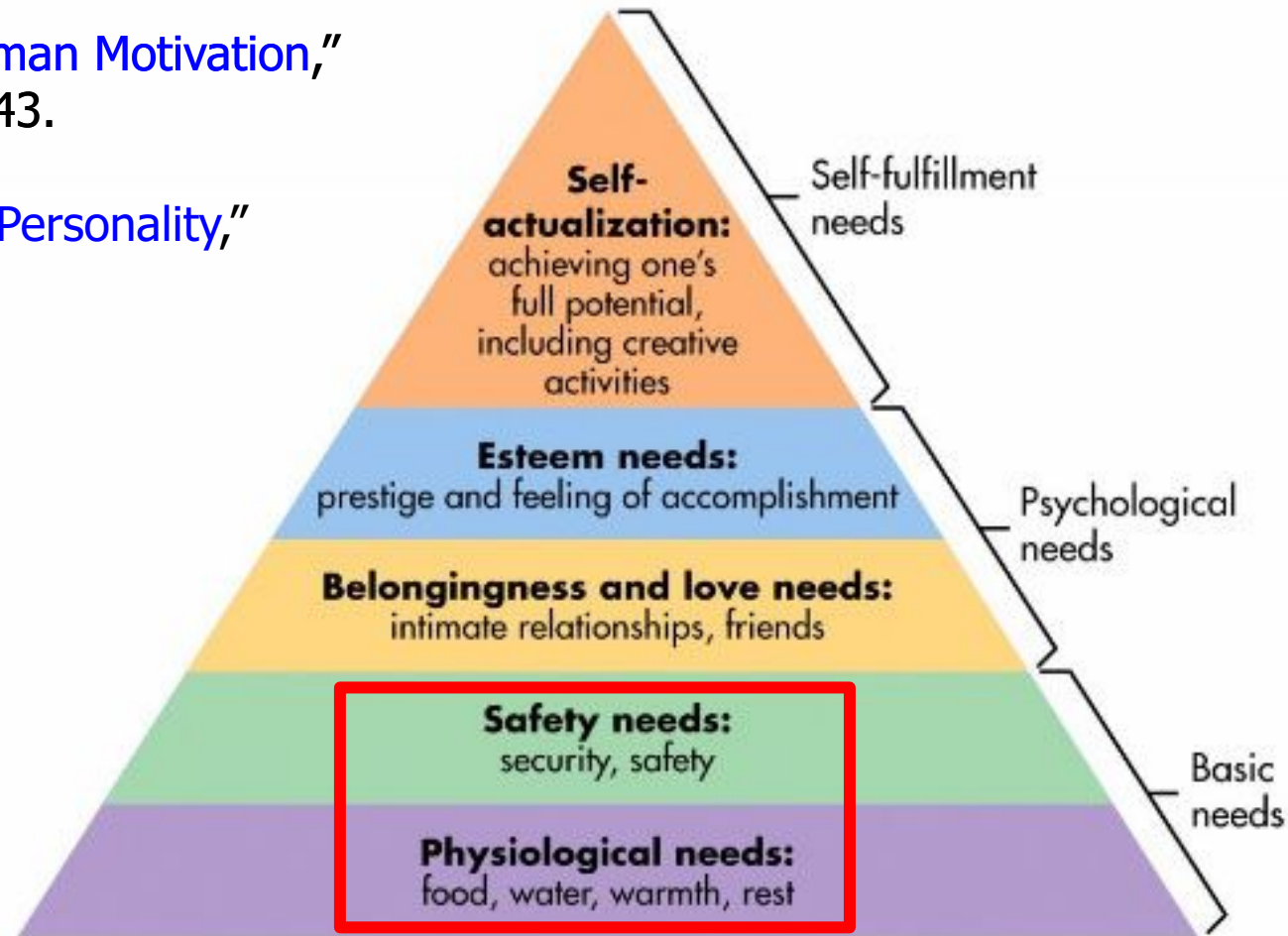
---

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

# Maslow's (Human) Hierarchy of Needs

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

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

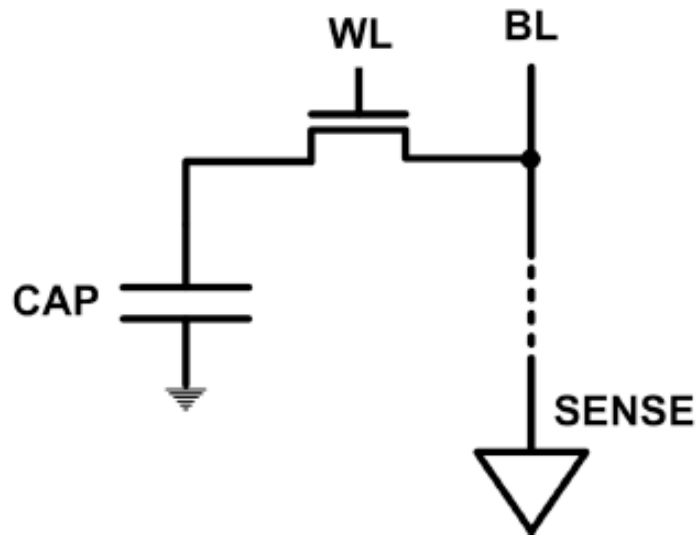


- We need to start with **reliability and security**...

# The DRAM Scaling Problem

---

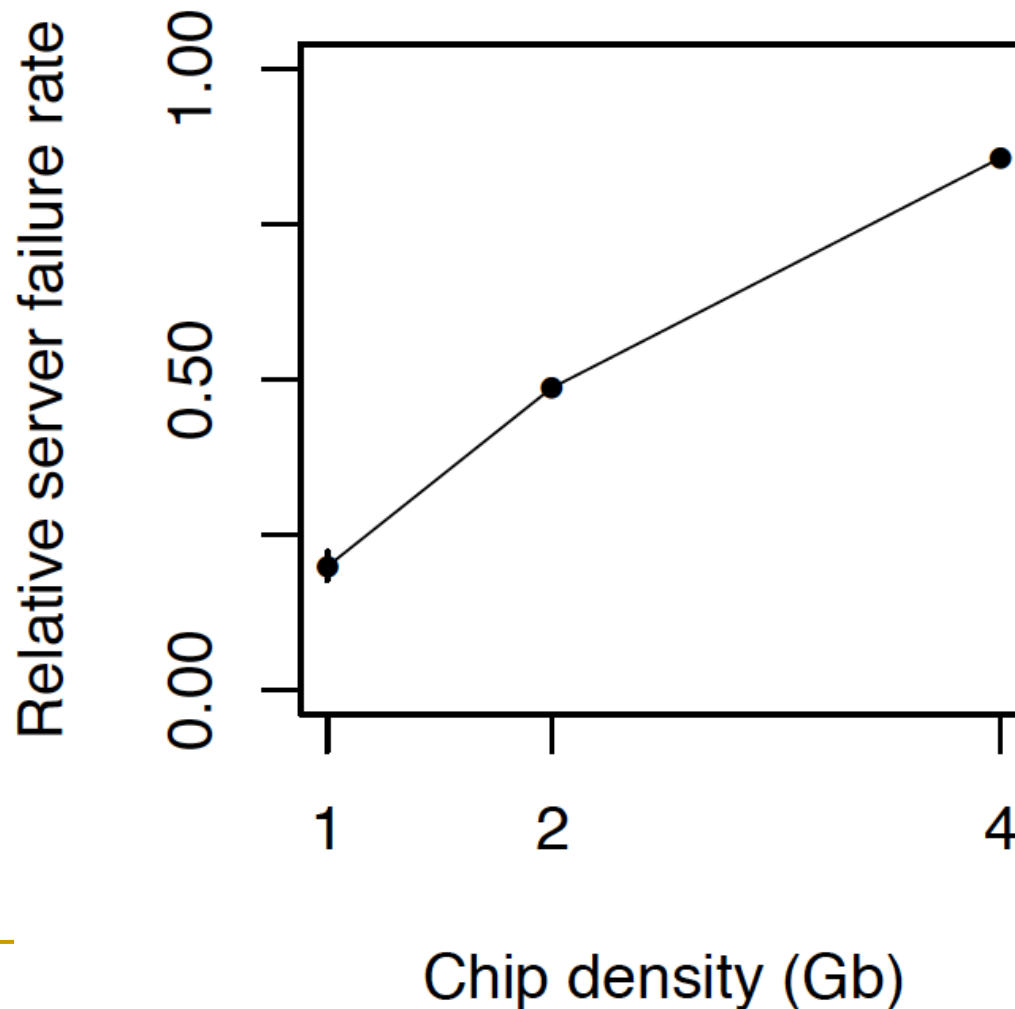
- DRAM stores charge in a capacitor (charge-based memory)
  - Capacitor must be large enough for reliable sensing
  - Access transistor should be large enough for low leakage and high retention time
  - Scaling beyond 40-35nm (2013) is challenging [ITRS, 2009]



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

# As Memory Scales, It Becomes Unreliable

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



*Intuition:  
quadratic  
increase  
in  
capacity*

# Large-Scale Failure Analysis of DRAM Chips

---

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

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

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

# Infrastructures to Understand Such Issues



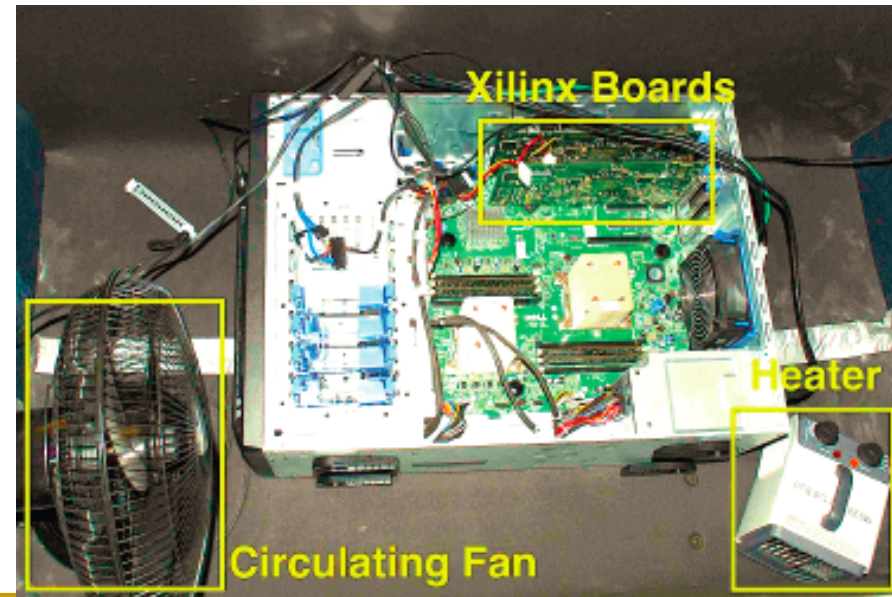
An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

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

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

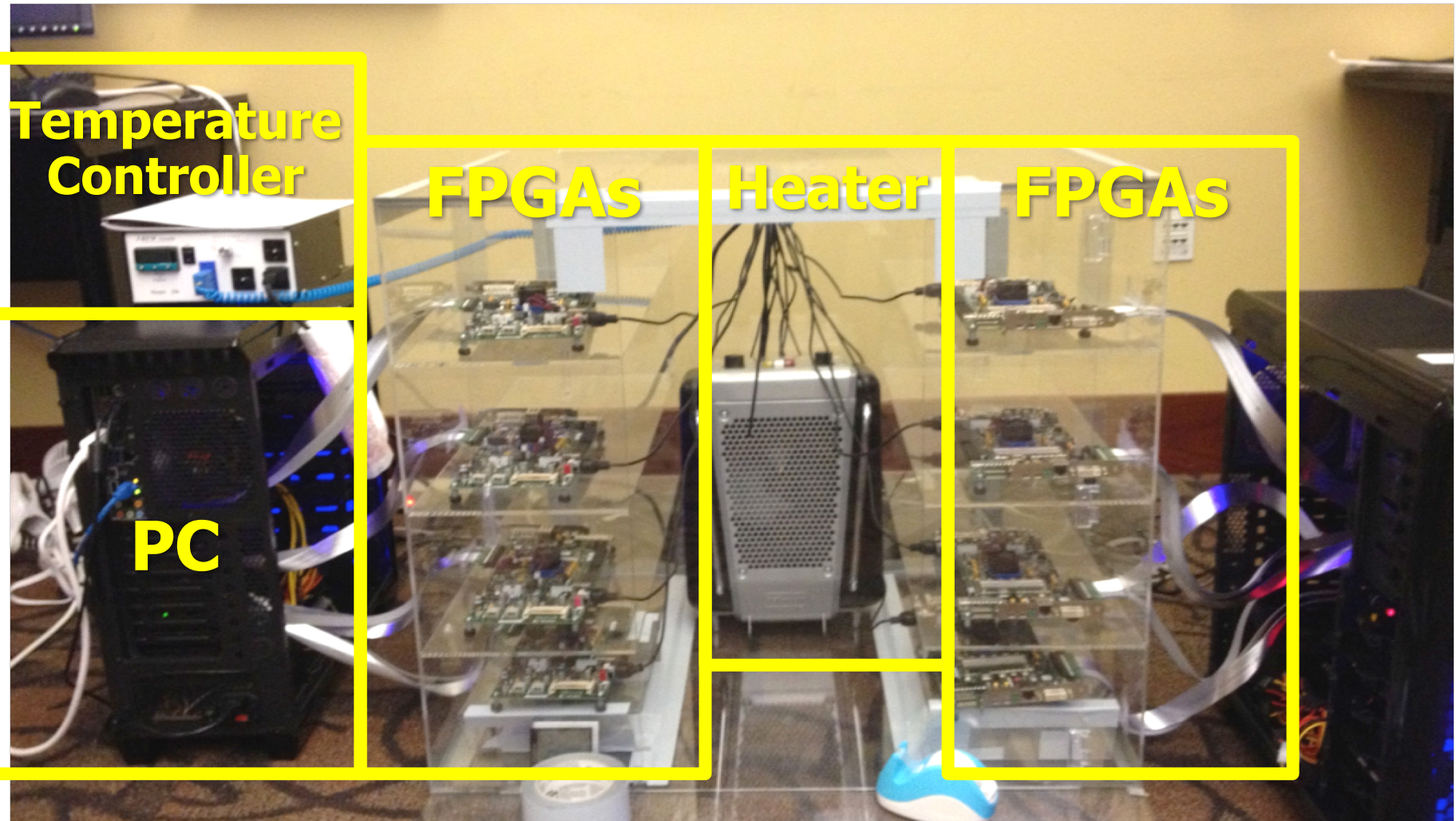
Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case (Lee et al., HPCA 2015)

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





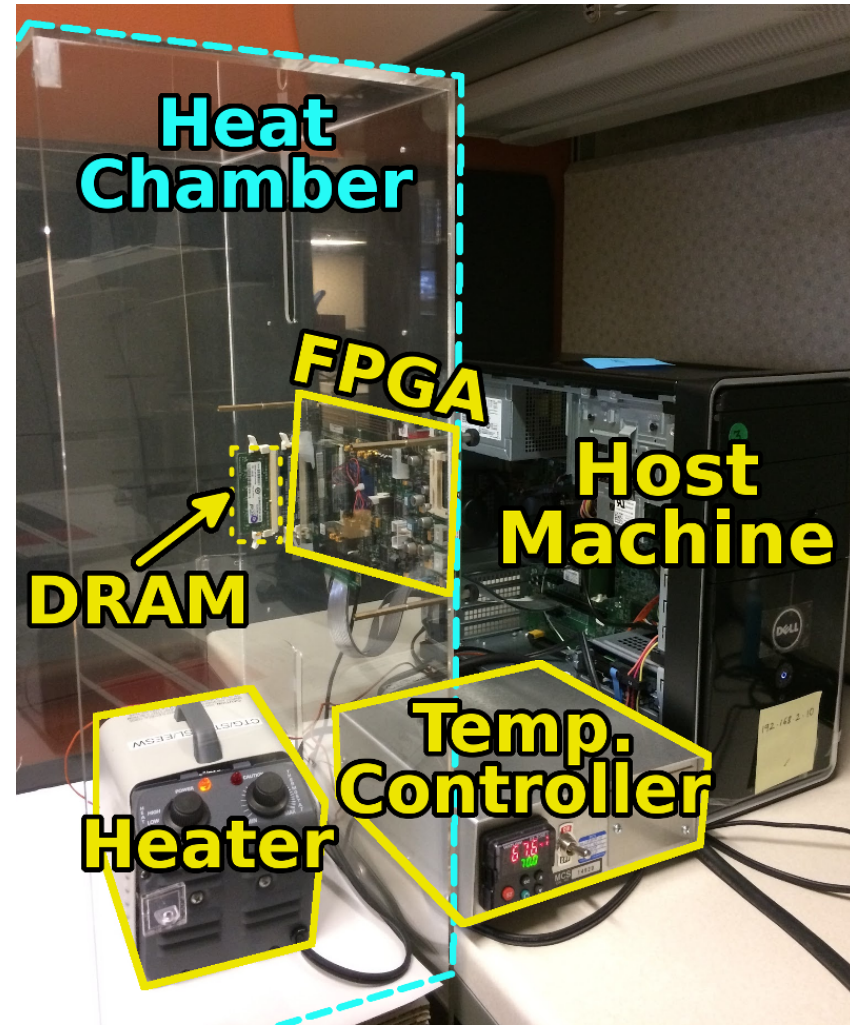
# Infrastructures to Understand Such Issues



# 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**

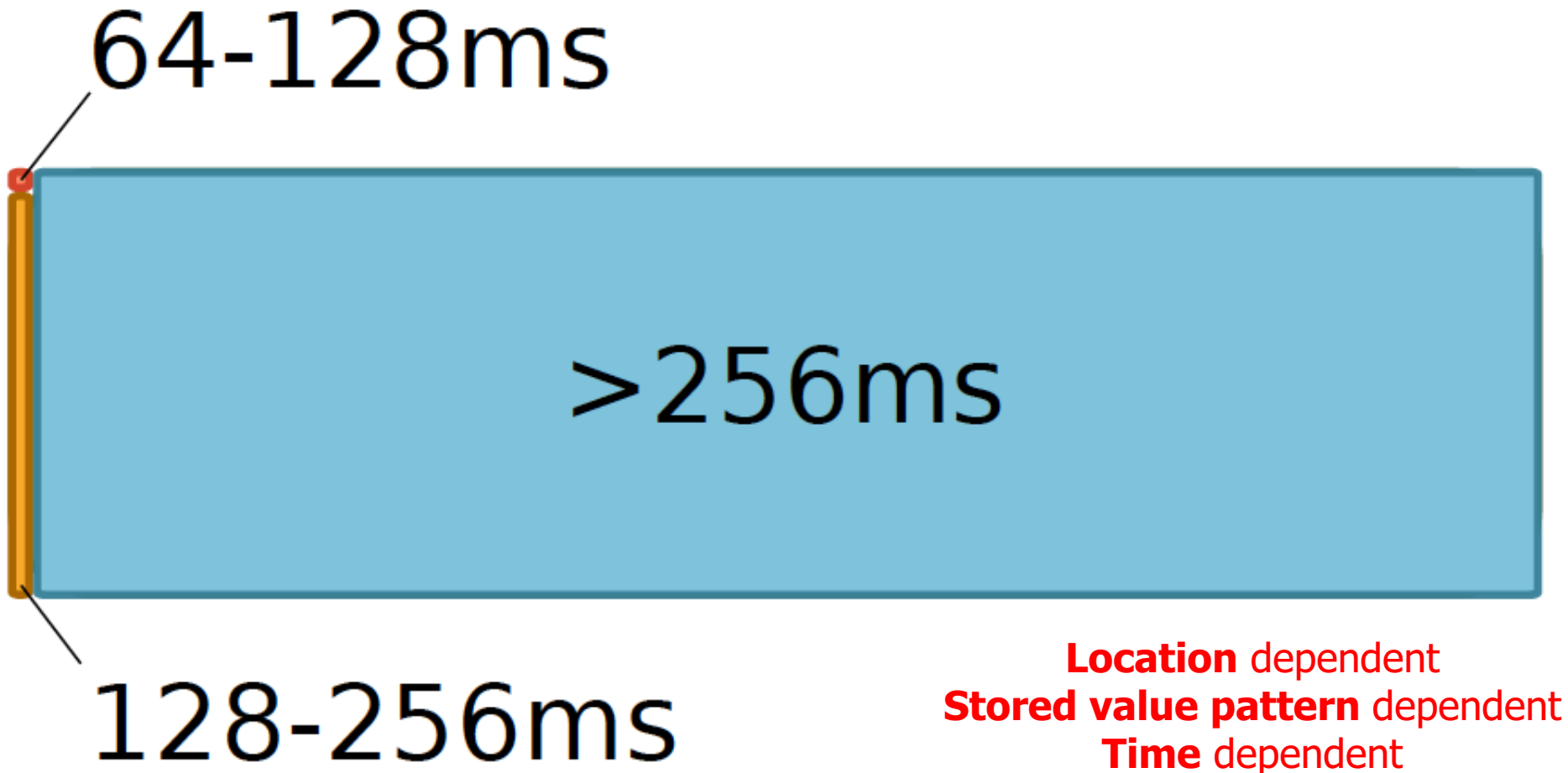
Hasan Hassan<sup>1,2,3</sup> Nandita Vijaykumar<sup>3</sup> Samira Khan<sup>4,3</sup> Saugata Ghose<sup>3</sup> Kevin Chang<sup>3</sup>  
Gennady Pekhimenko<sup>5,3</sup> Donghyuk Lee<sup>6,3</sup> Oguz Ergin<sup>2</sup> Onur Mutlu<sup>1,3</sup>

<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:



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

---

One can  
predictably induce errors  
in most DRAM memory chips

# DRAM RowHammer

---

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

**WIRED**

Forget Software—Now Hackers Are Exploiting Physics

BUSINESS

CULTURE

DESIGN

GEAR

SCIENCE

ANDY GREENBERG SECURITY 08.31.16 7:00 AM

SHARE



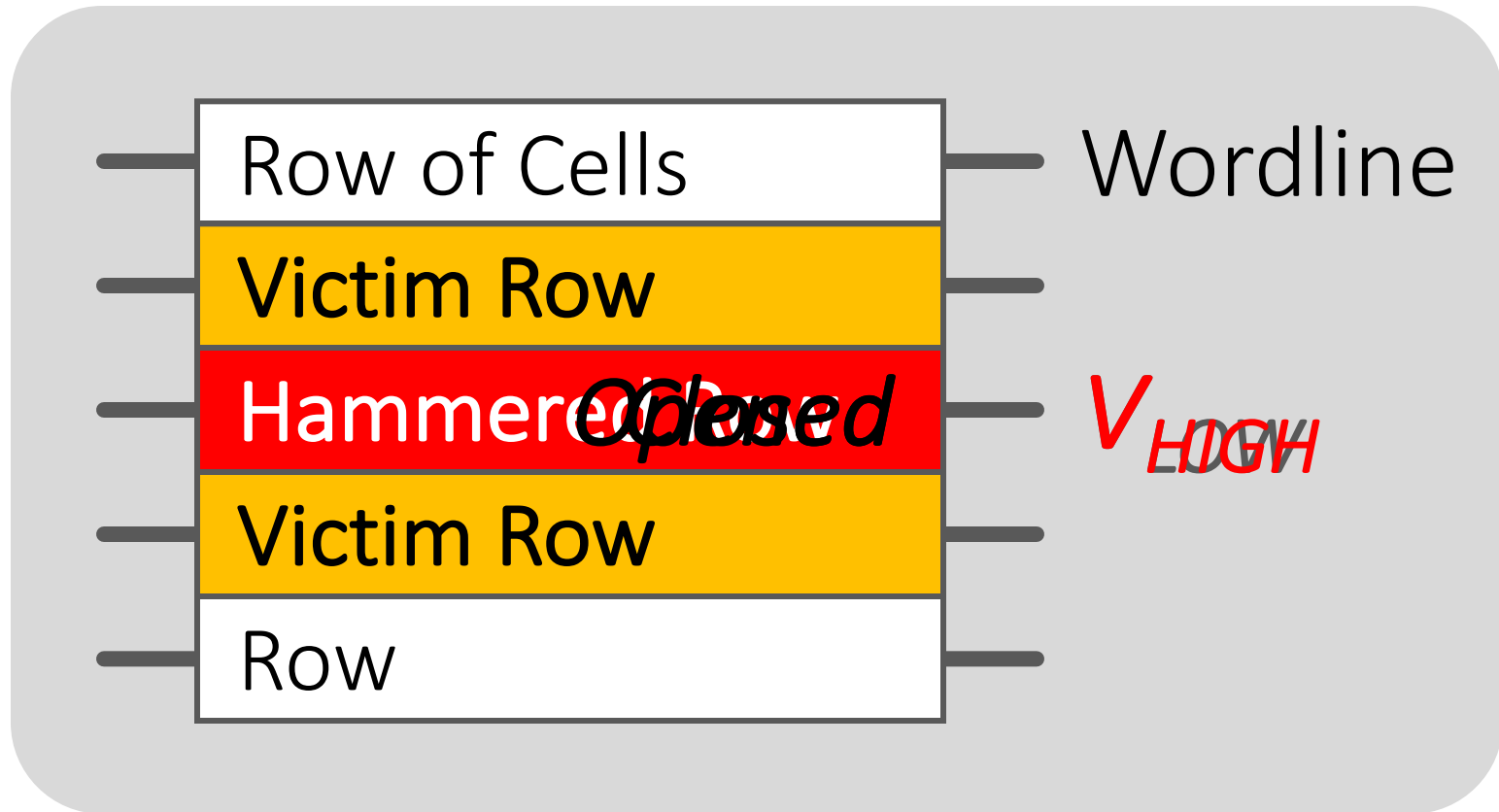
SHARE  
18276



TWEET

# FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS

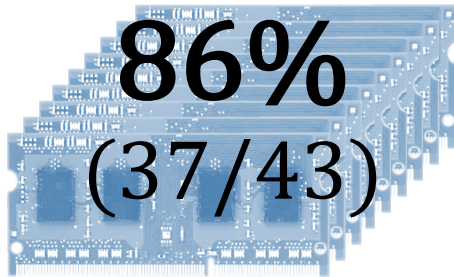
# Modern DRAM is Prone to Disturbance Errors



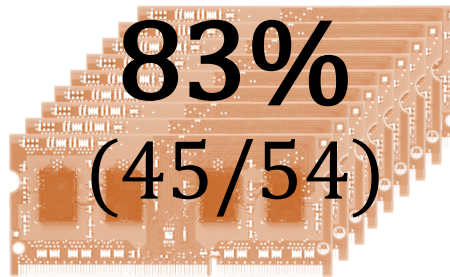
**Repeatedly reading** a row enough times (before memory gets refreshed) induces **disturbance errors** in **adjacent rows** in **most real DRAM chips you can buy today**

# Most DRAM Modules Are Vulnerable

A company



B company



C company

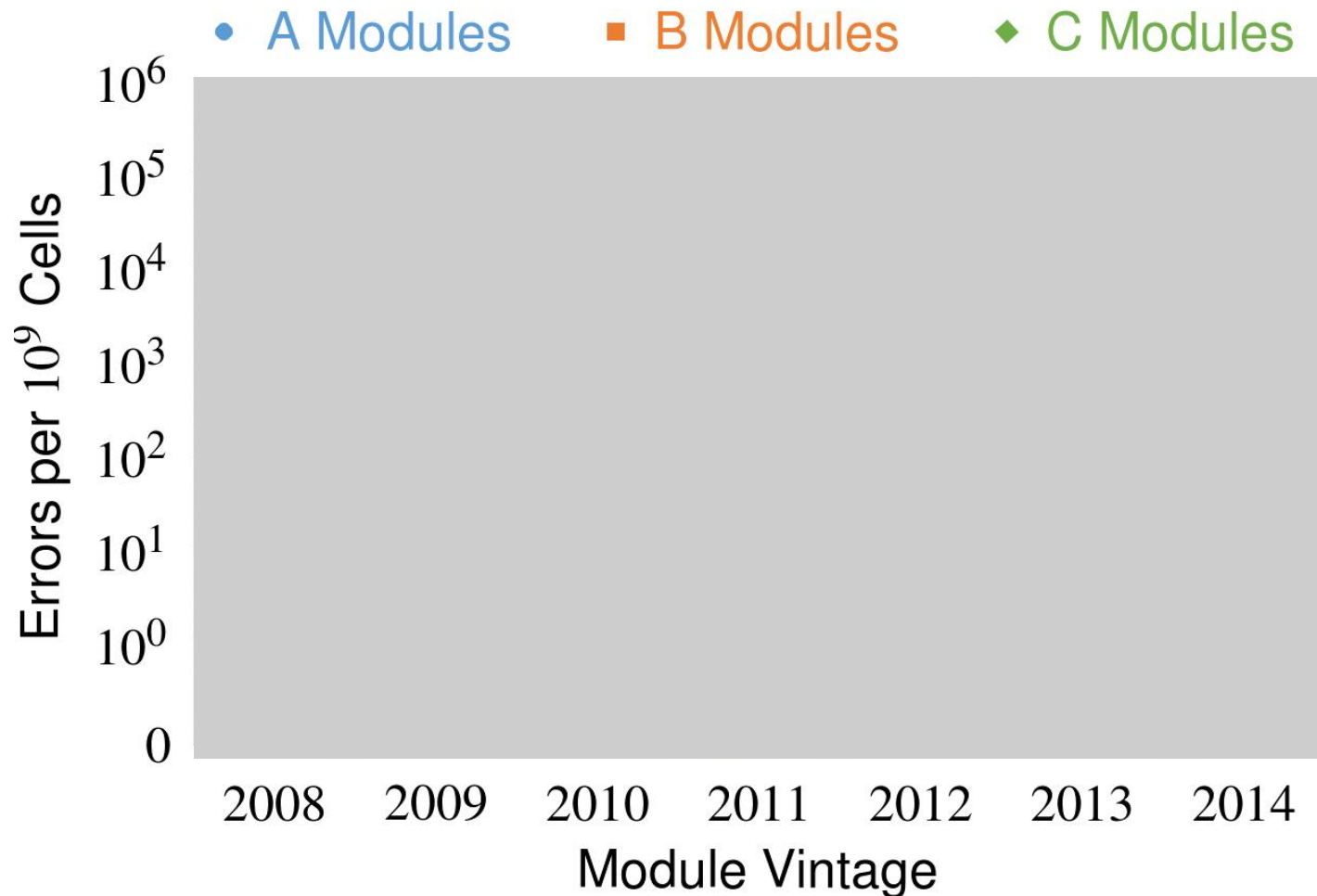


Up to  
 $1.0 \times 10^7$   
errors

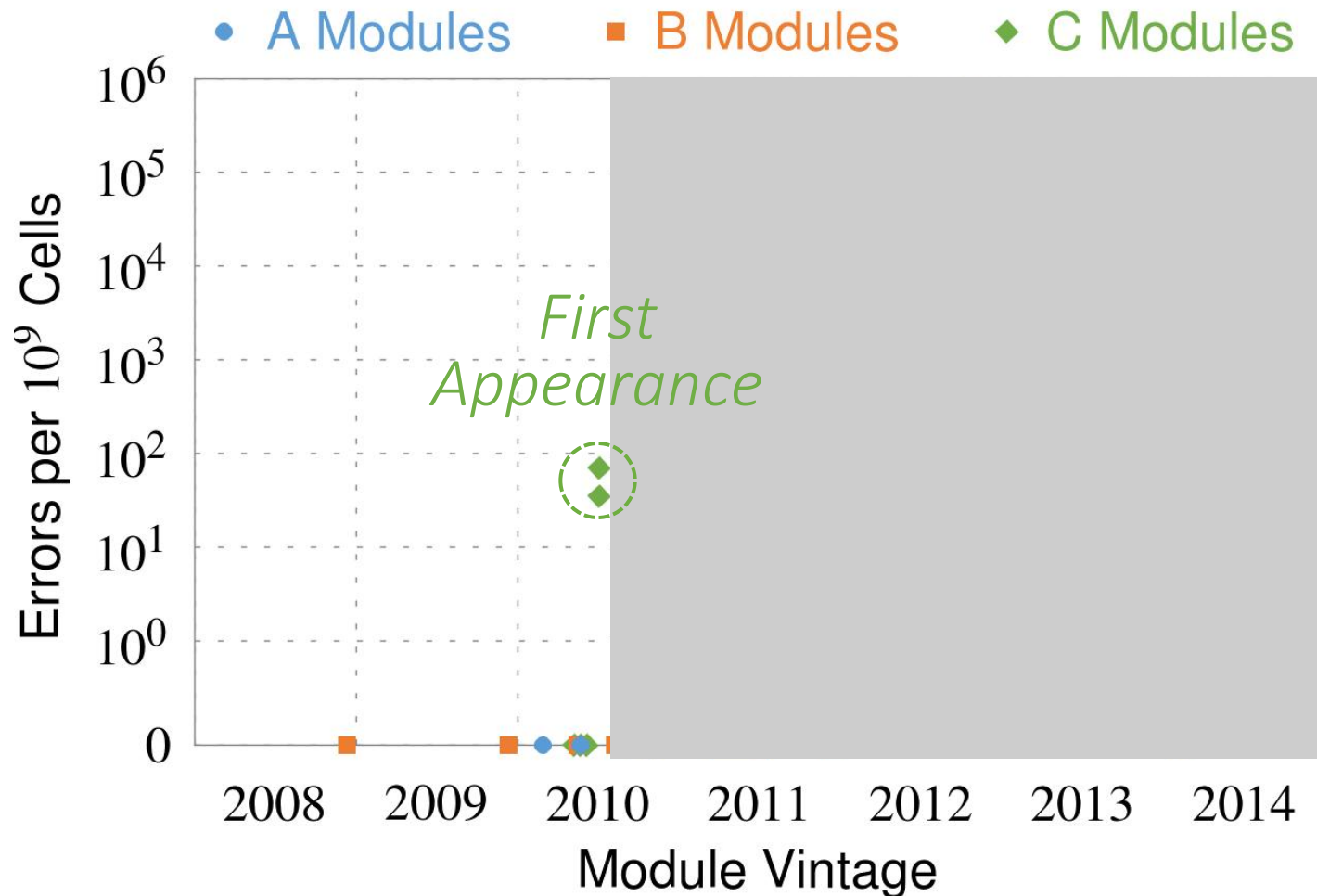
Up to  
 $2.7 \times 10^6$   
errors

Up to  
 $3.3 \times 10^5$   
errors

# Recent DRAM Is More Vulnerable

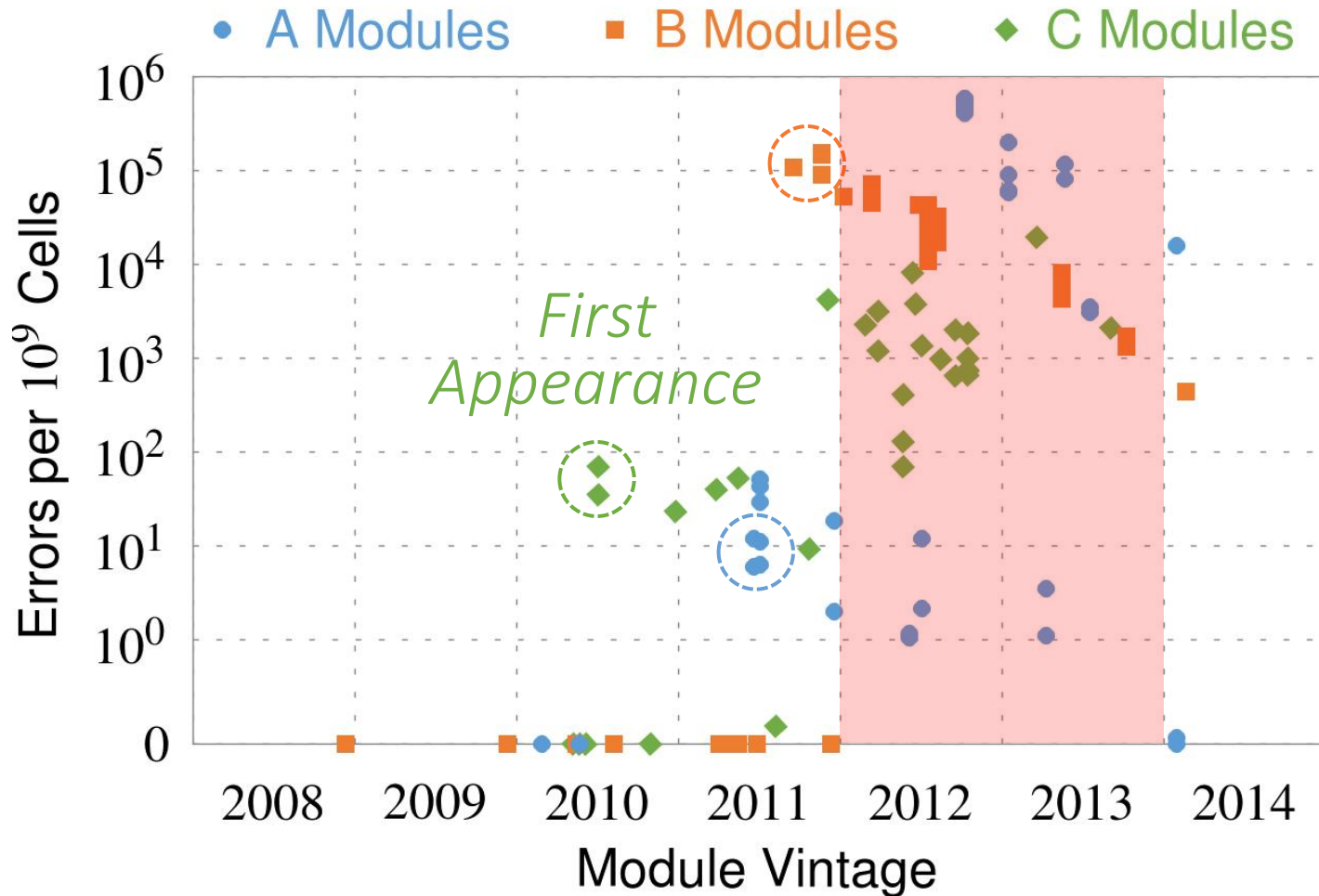


# Recent DRAM Is More Vulnerable





# Recent DRAM Is More Vulnerable



*All modules from 2012-2013 are vulnerable*

# 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

# Security Implications



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

# More Security Implications

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

www.iaik.tugraz.at ■

Not there yet, but ...



ROOT privileges for web apps!

29

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



GATED  
COMMUNITIES

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



# More Security Implications

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



Drammer: Deterministic Rowhammer  
Attacks on Mobile Platforms, CCS'16 <sup>47</sup>

# More Security Implications?

---



# 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>



# Call for Intelligent Memory Controllers

## DRAM Process Scaling Challenges

### ❖ Refresh

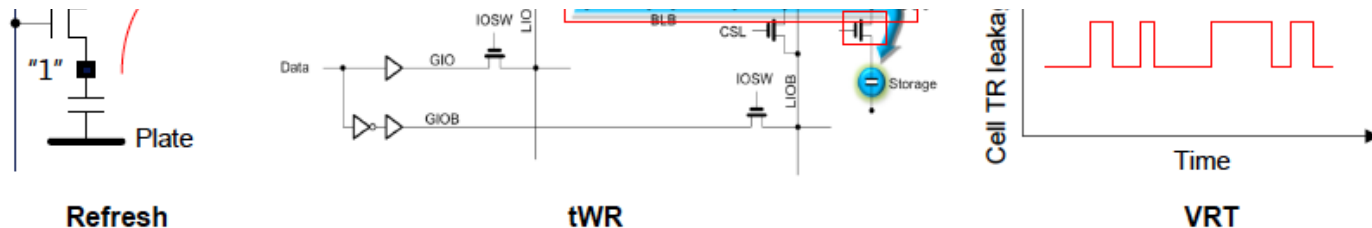
- Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

THE MEMORY FORUM 2014

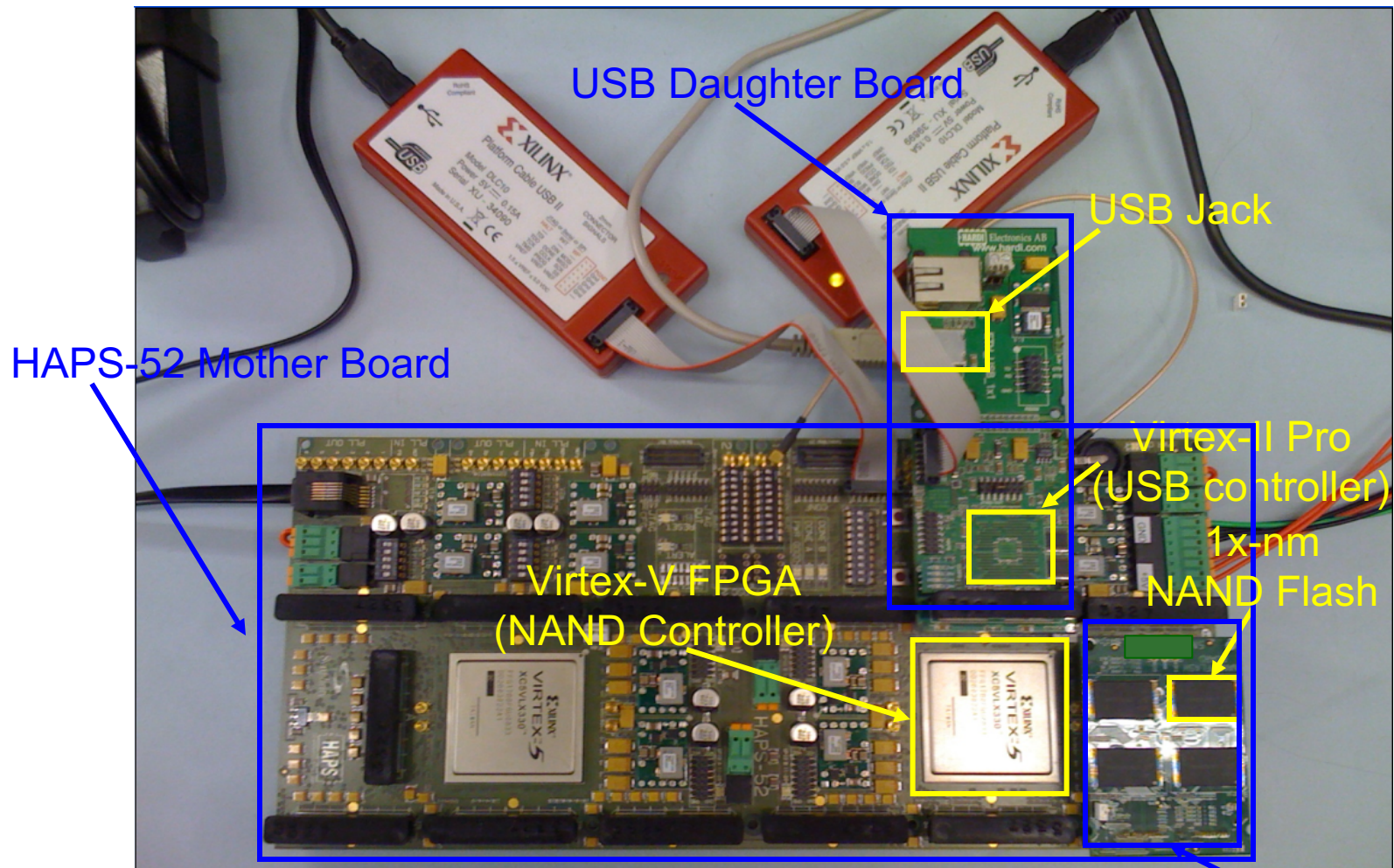
## Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling

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

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



# Aside: Intelligent Controller for NAND Flash



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

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



*Proceedings of the IEEE, Sept. 2017*



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

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

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

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

## Main Memory Needs Intelligent Controllers

# Agenda

---

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

# Three Key Systems Trends

---

## 1. Data access is a major bottleneck

- ▣ Applications are increasingly data hungry

## 2. Energy consumption is a key limiter

## 3. Data movement energy dominates compute

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

# The Need for More Memory Performance

---



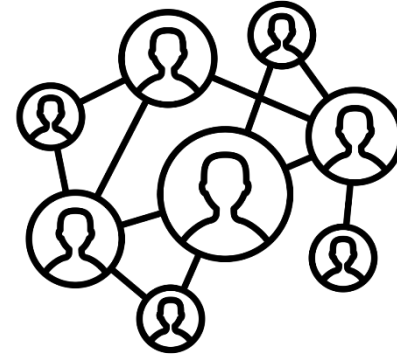
## In-memory Databases

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



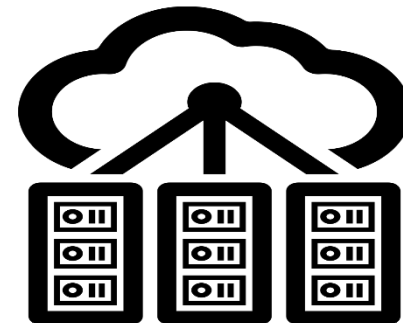
## In-Memory Data Analytics

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



## Graph/Tree Processing

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



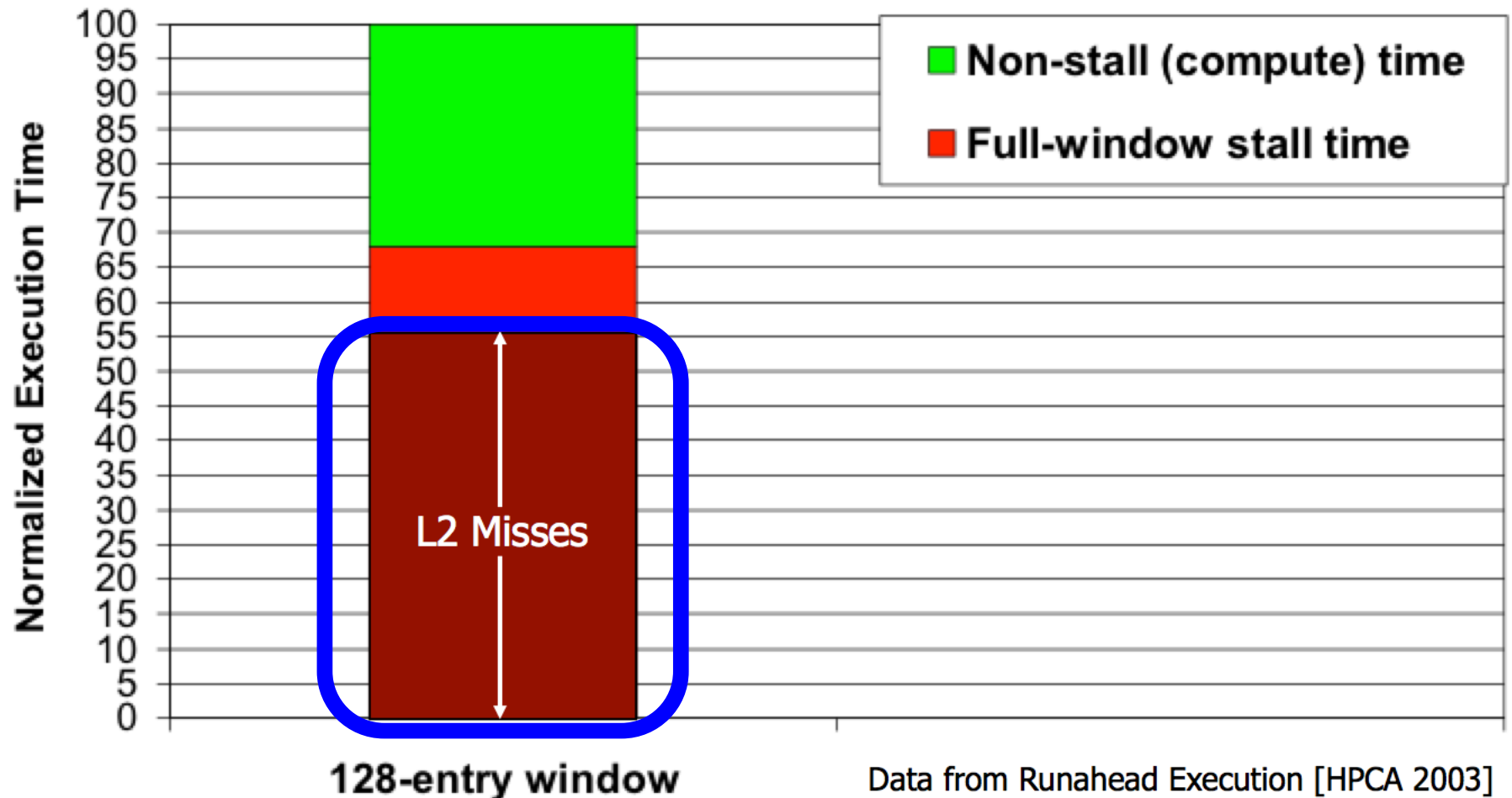
## Datacenter Workloads

[Kanev+ (Google), ISCA'15]



# The Performance Perspective (1996-2005)

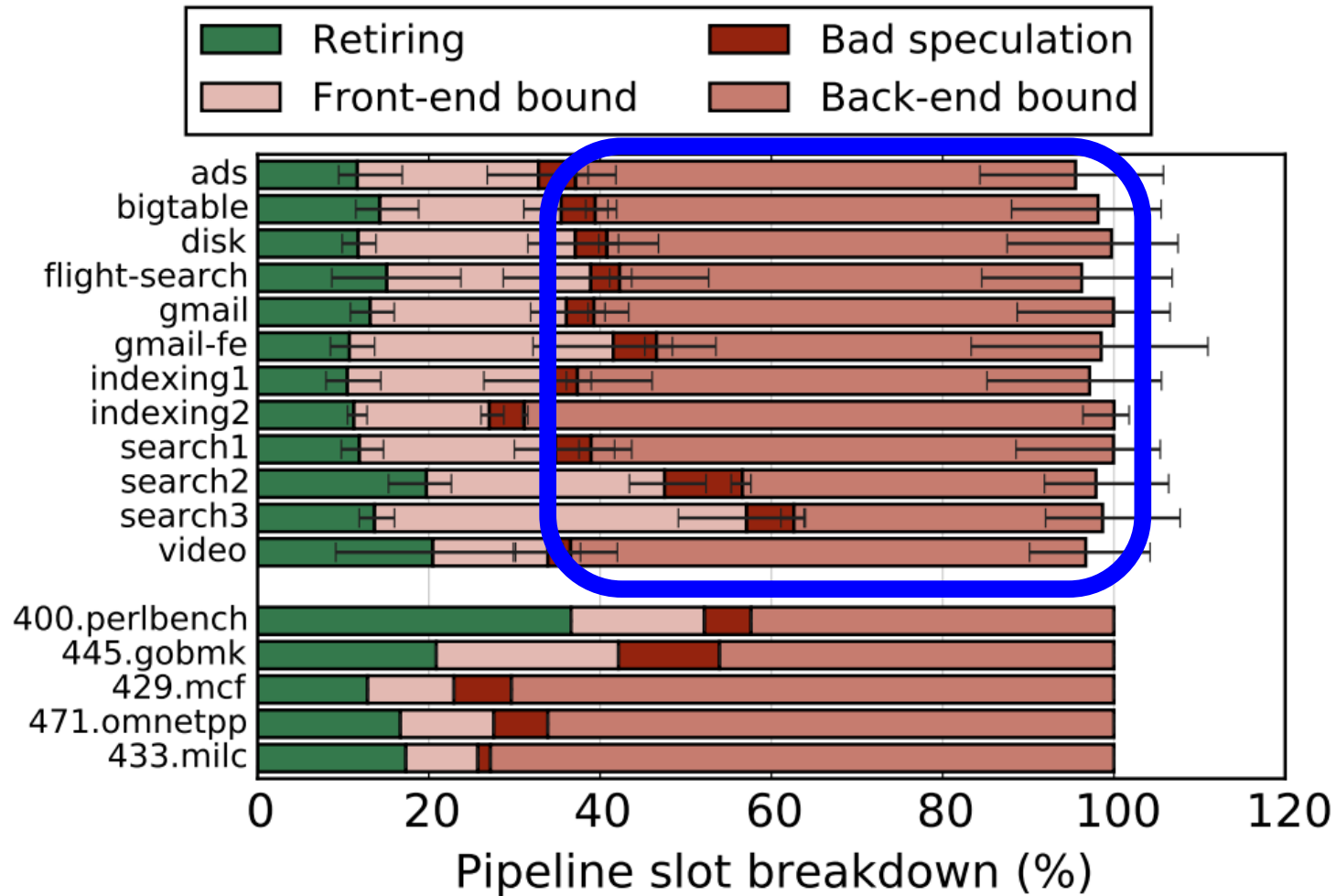
- **“It’s the Memory, Stupid!”** (Richard Sites, MPR, 1996)





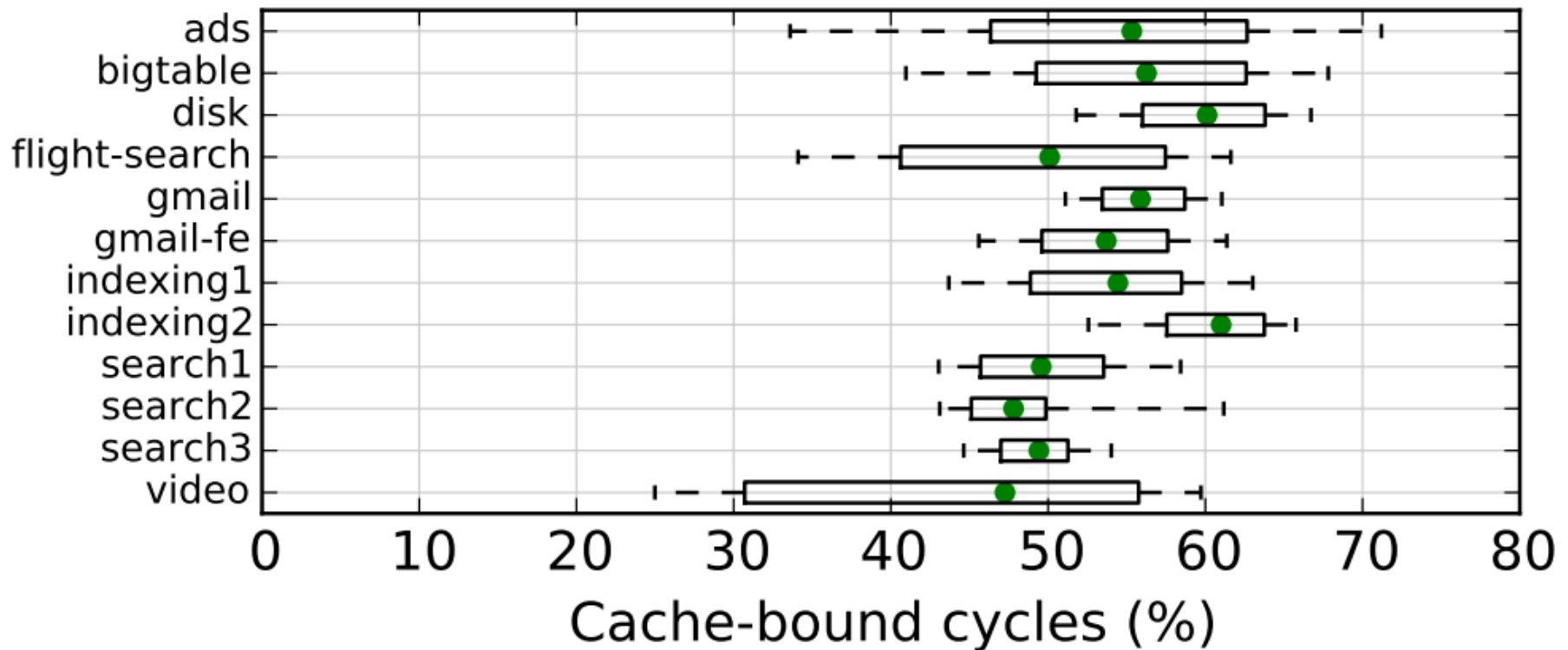
# The Performance Perspective (Today)

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



# The Performance Perspective (Today)

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



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

# The Performance Perspective

---

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

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

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

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

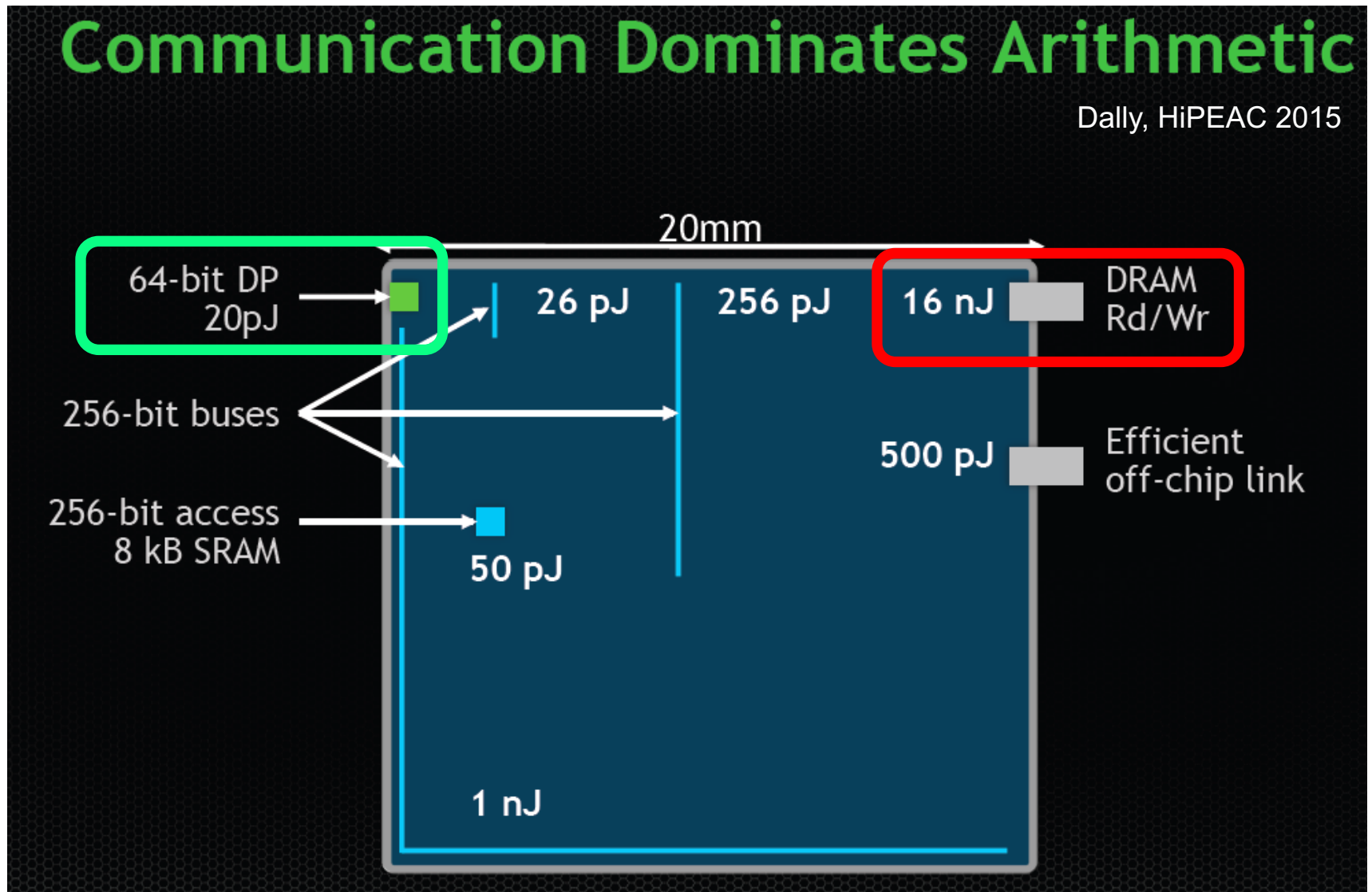
†Microprocessor Research  
Intel Labs  
jared.w.stark@intel.com

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

# The Energy Perspective

## Communication Dominates Arithmetic

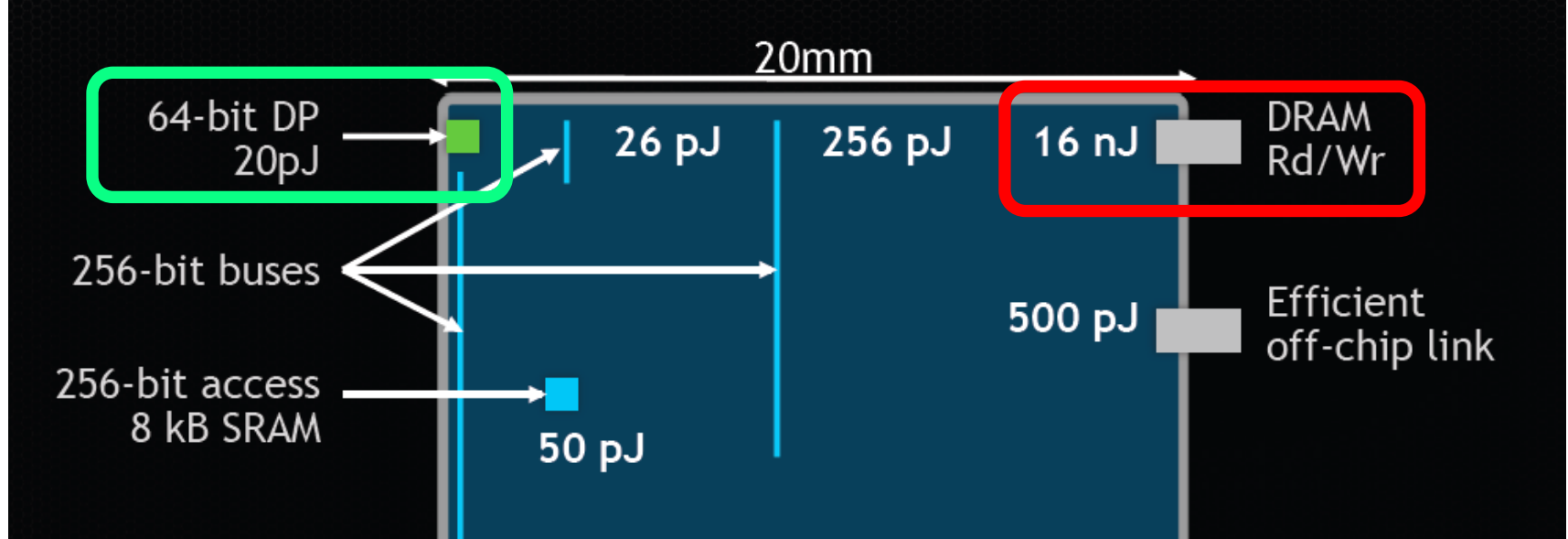
Dally, HiPEAC 2015



# Data Movement vs. Computation Energy

## Communication Dominates Arithmetic

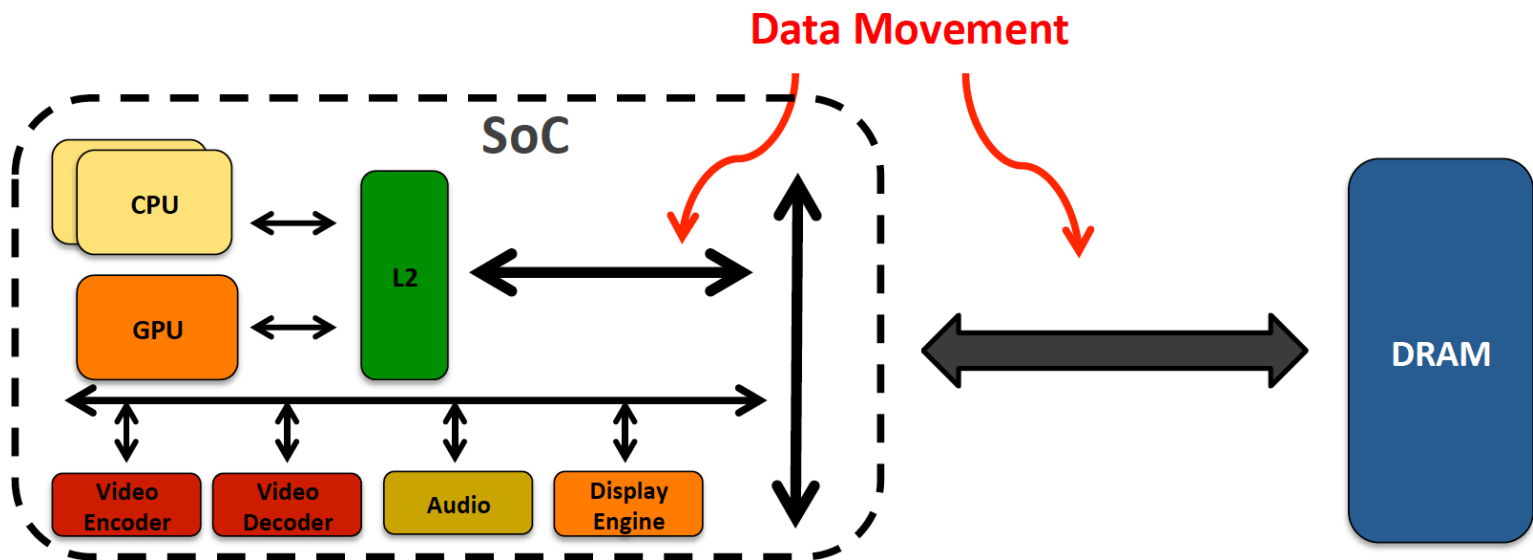
Dally, HiPEAC 2015



A memory access consumes  $\sim 1000\times$  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)

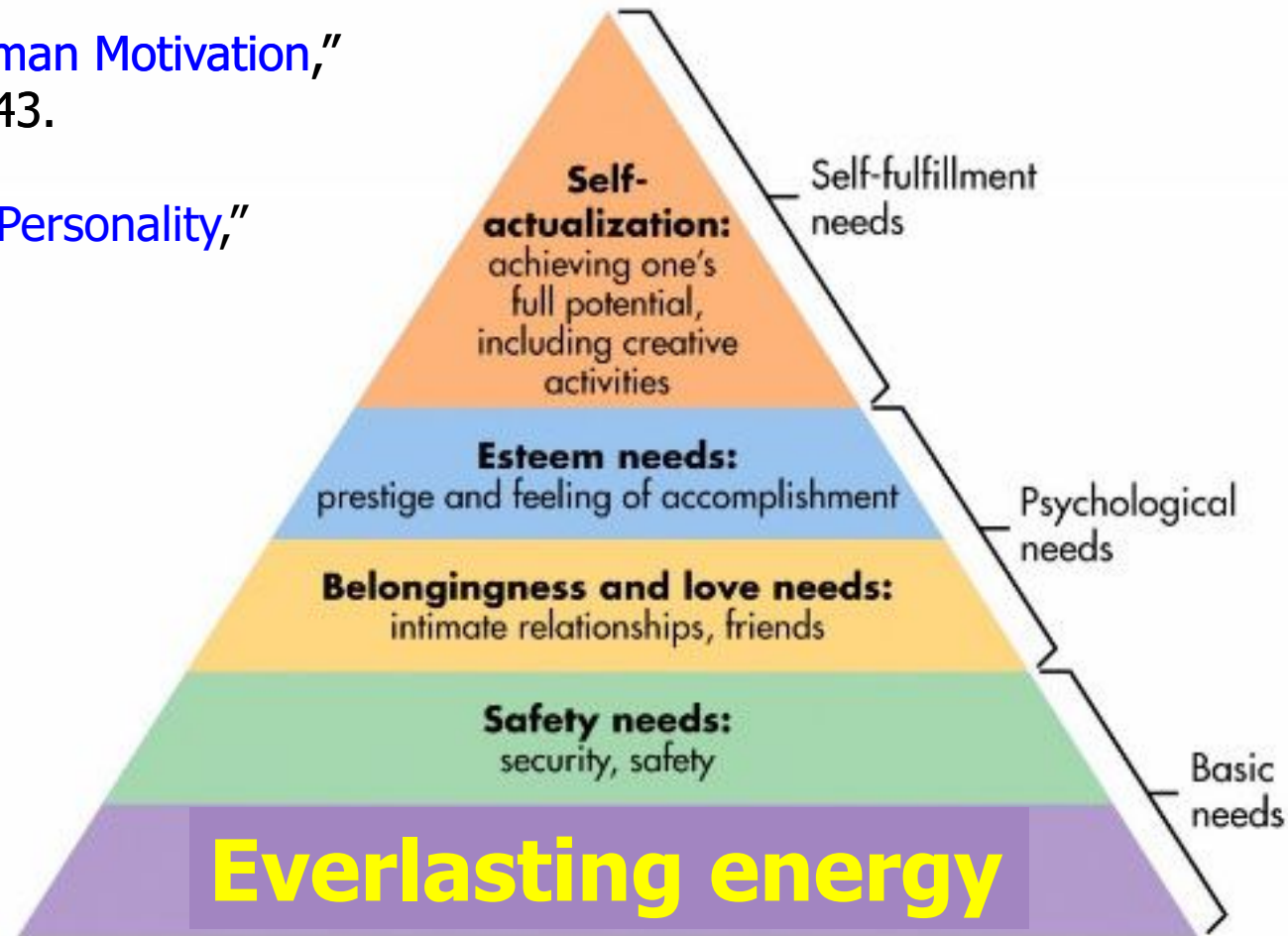
[2]: Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms (IISWC'14)

High Performance  
and  
Energy Efficient

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

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

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





# The Problem

---

Data access is the major performance and energy bottleneck

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

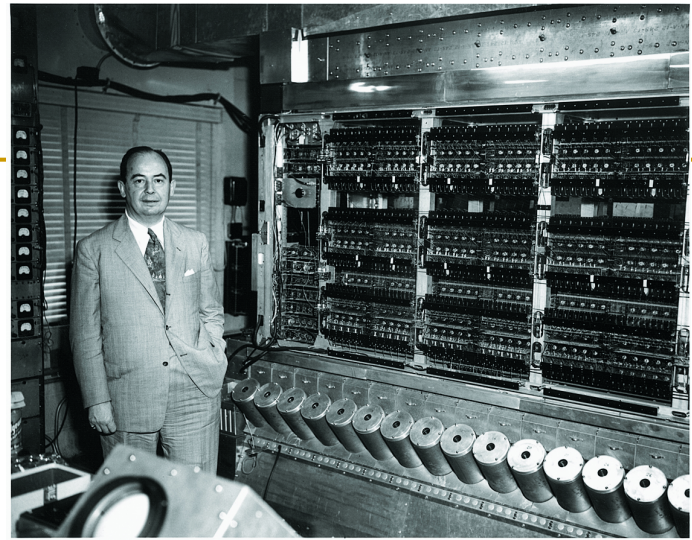
# The Problem

---

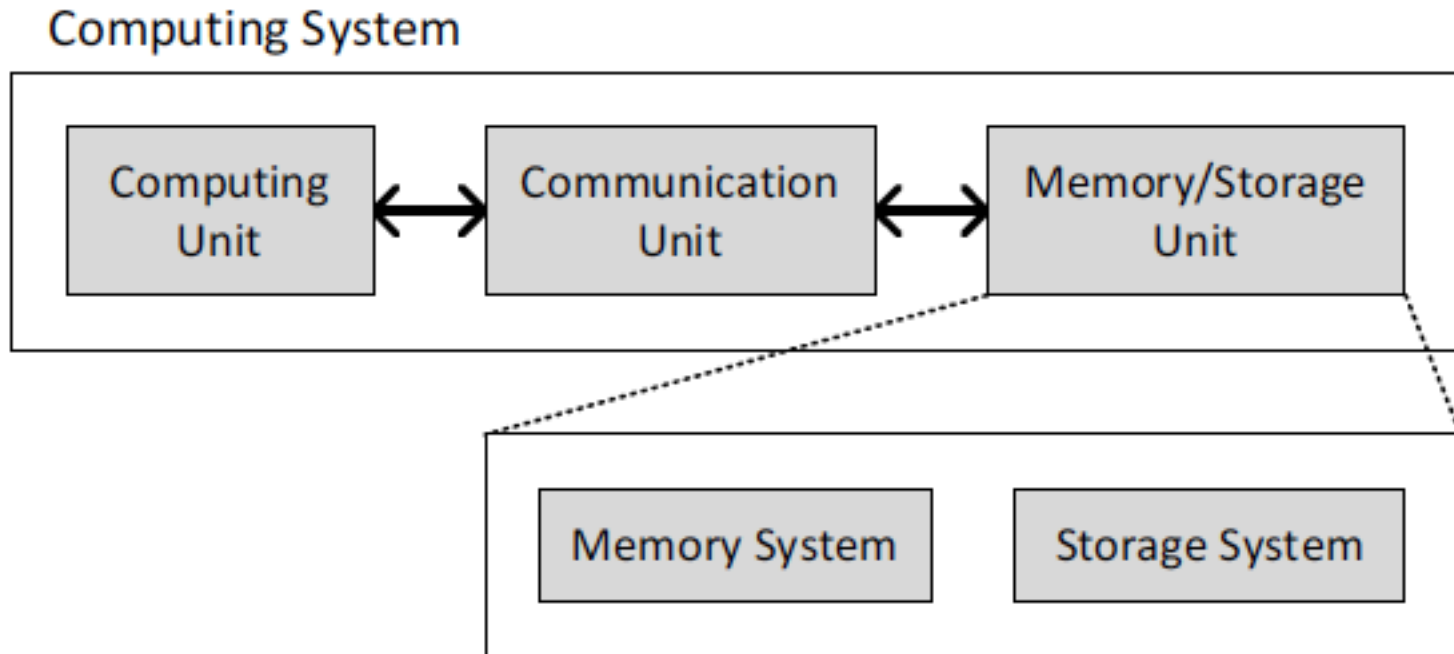
Processing of data  
is performed  
far away from the data

# A Computing System

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

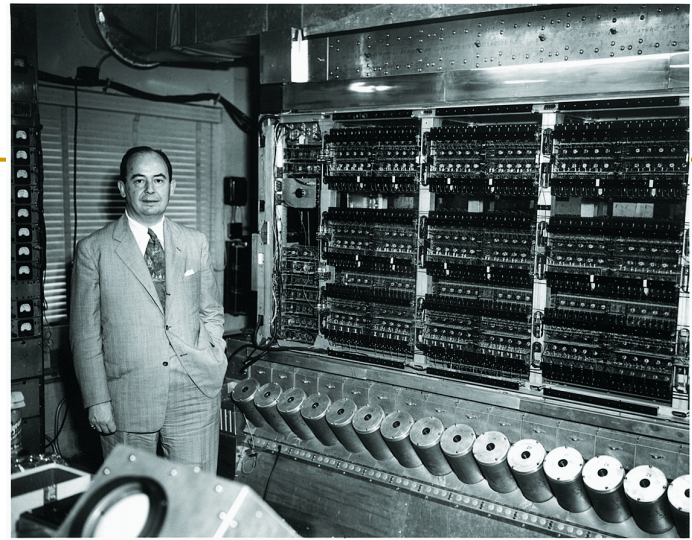


Burks, Goldstein, von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument," 1946.



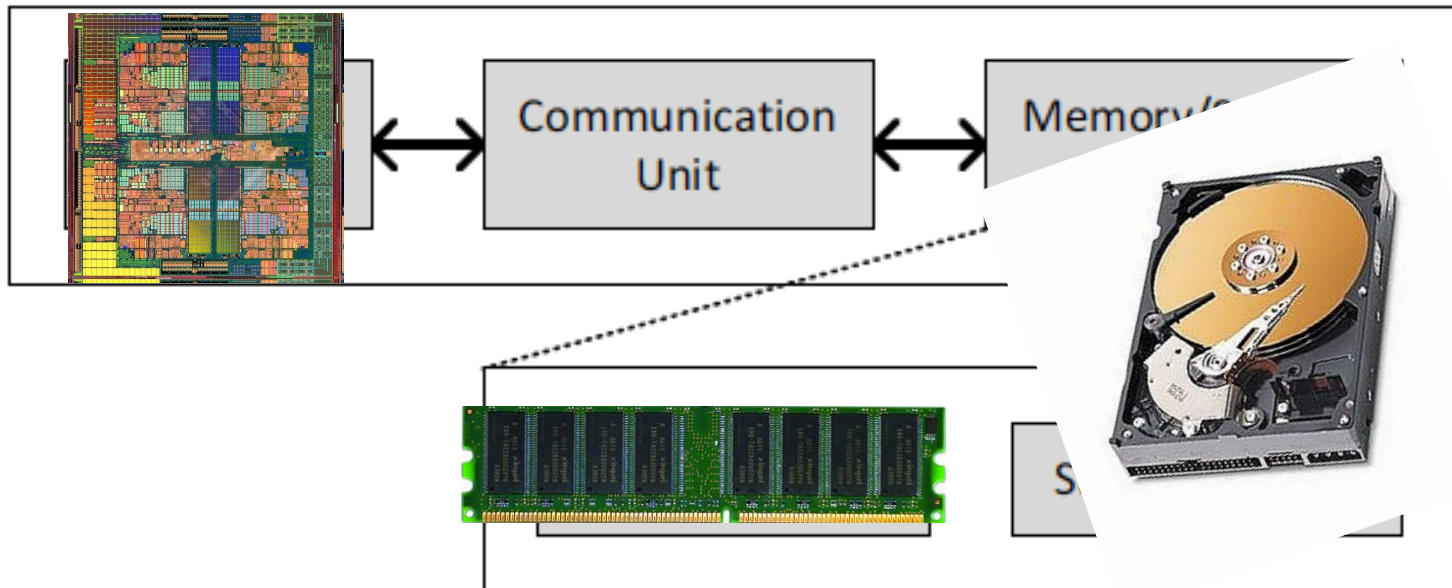
# A Computing System

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



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

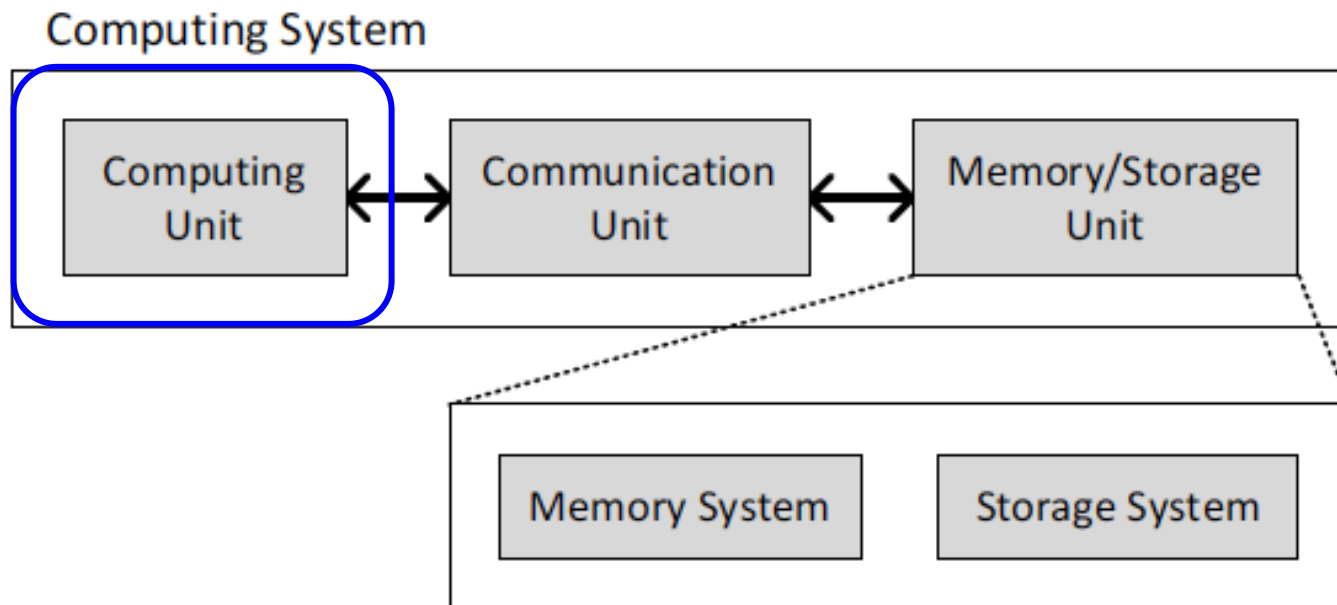
## Computing System



# Today's Computing Systems

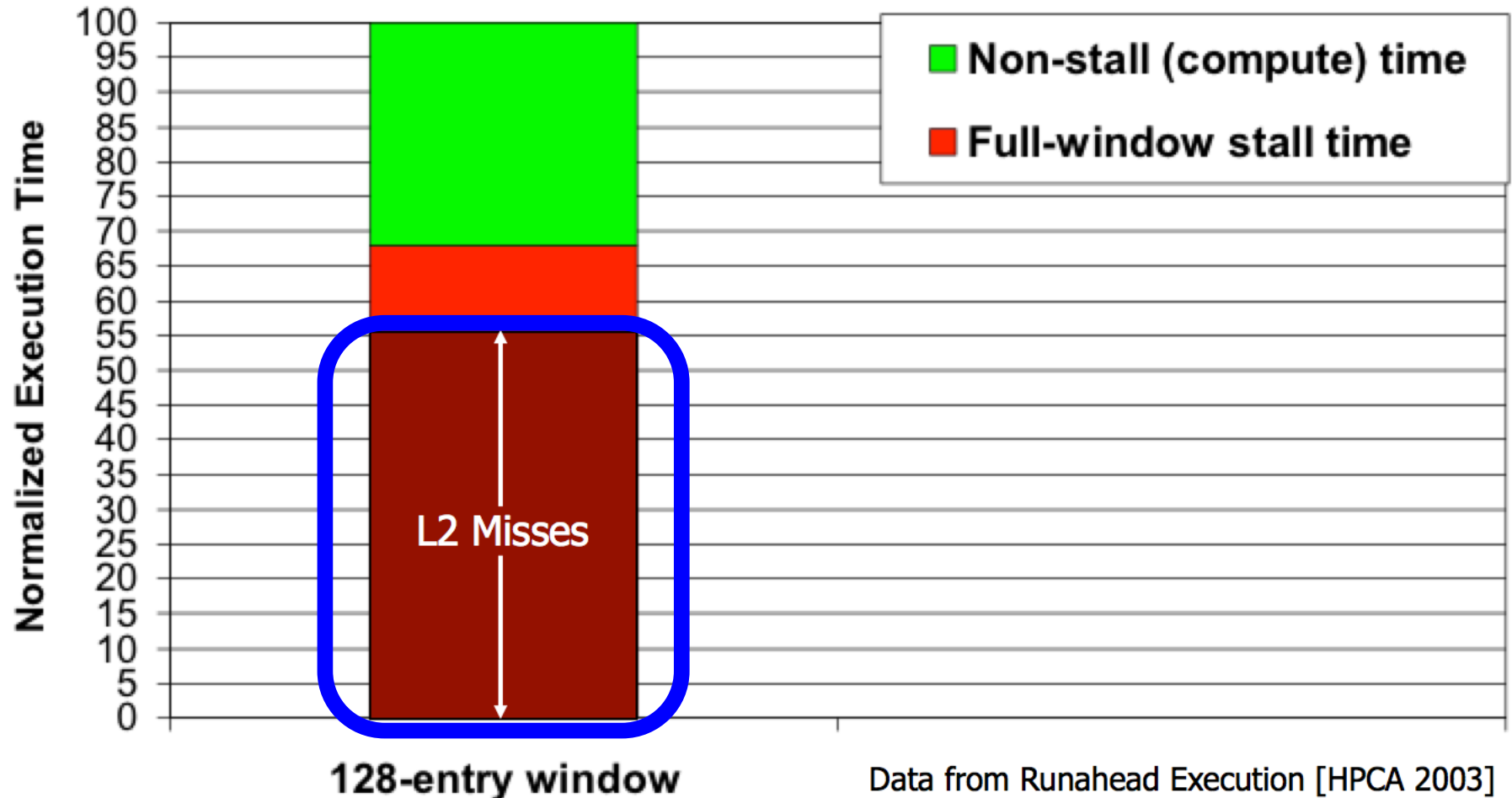
---

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



# Yet ...

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

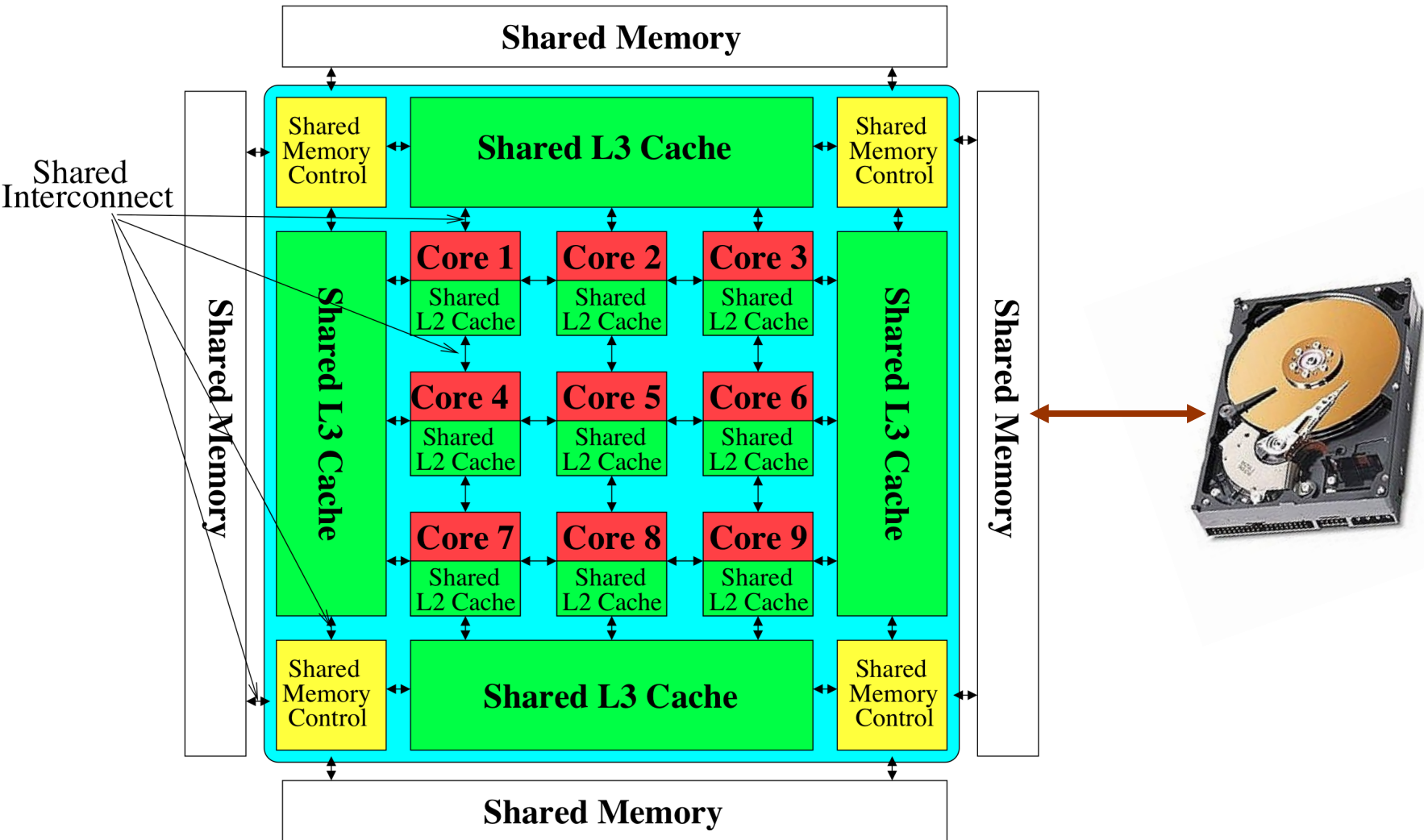


# Perils of Processor-Centric Design

---

- **Grossly-imbalanced systems**
  - ❑ Processing done only in **one place**
  - ❑ Everything else just stores and moves data: **data moves a lot**
    - Energy inefficient
    - Low performance
    - Complex
- **Overly complex and bloated processor (and accelerators)**
  - ❑ To tolerate data access from memory
  - ❑ Complex hierarchies and mechanisms
    - Energy inefficient
    - Low performance
    - Complex

# Perils of Processor-Centric Design



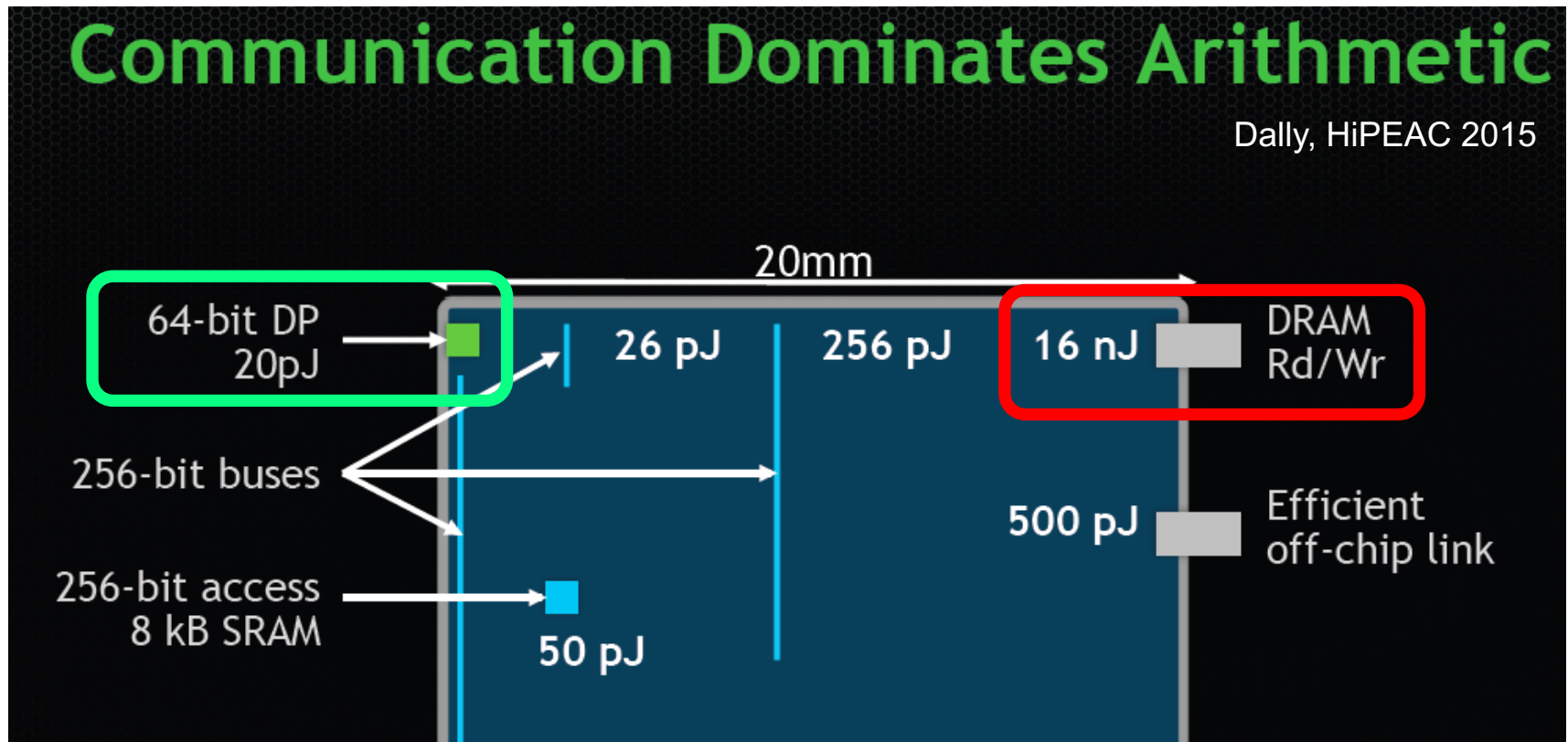
**Most of the system is dedicated to storing and moving data**



# We Do Not Want to Move Data!

## Communication Dominates Arithmetic

Dally, HiPEAC 2015



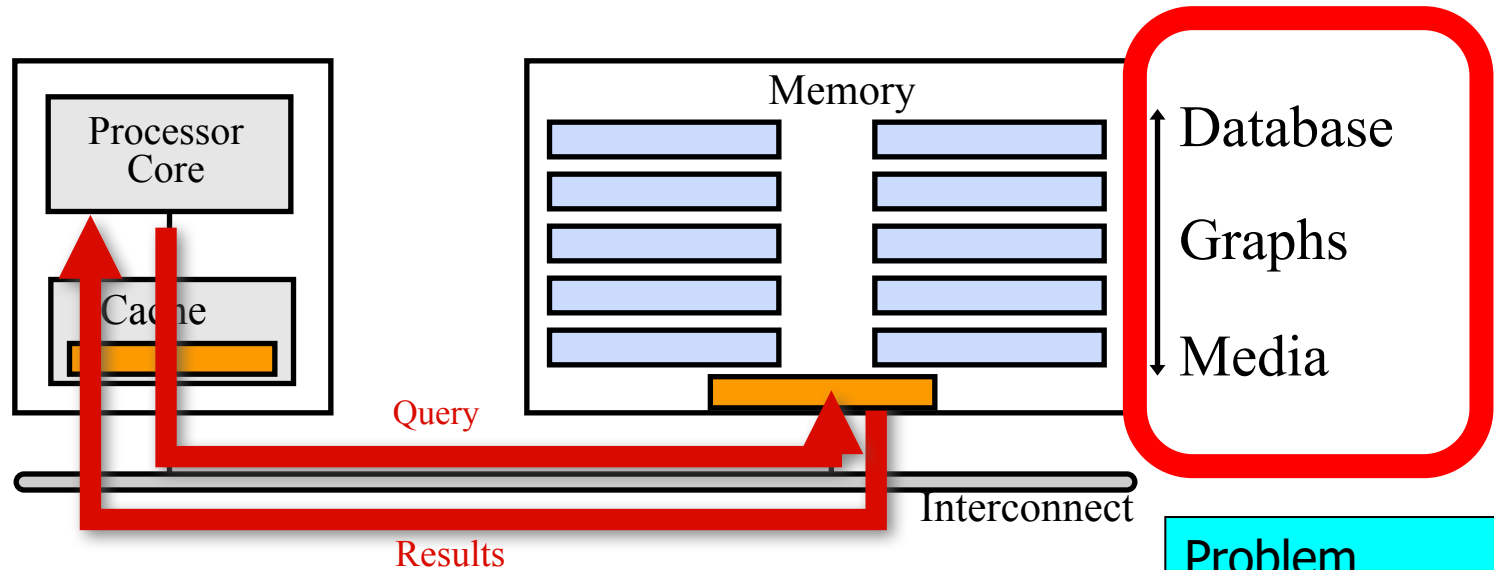
A memory access consumes  $\sim 1000X$   
the energy of a complex addition

# We Need A Paradigm Shift To ...

---

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

# Goal: Processing Inside Memory



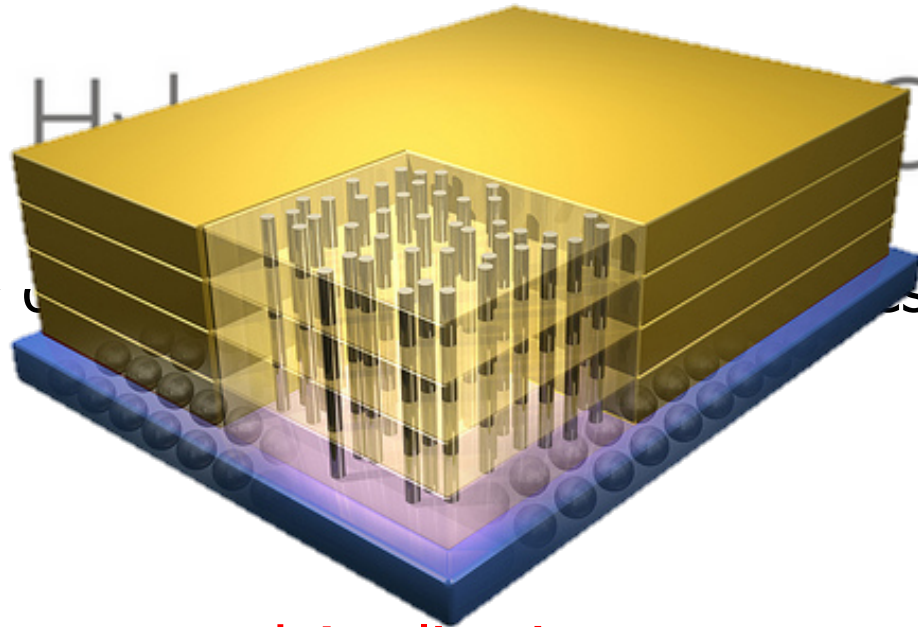
- Many questions ... How do we design the:
  - ❑ compute-capable memory & controllers?
  - ❑ processor chip?
  - ❑ software and hardware interfaces?
  - ❑ system software and languages?
  - ❑ algorithms?

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

# Why In-Memory Computation Today?



→ Industry C



## ■ Pull from Systems and Applications

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

# Agenda

---

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

# Processing in Memory: Two Approaches

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

# Approach 1: Minimally Changing DRAM

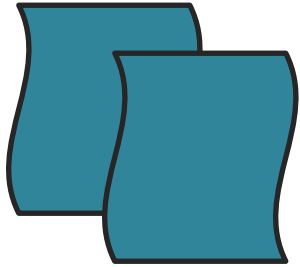
---

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

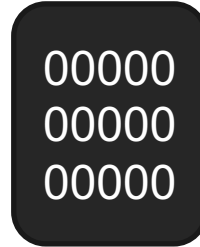


# Starting Simple: Data Copy and Initialization

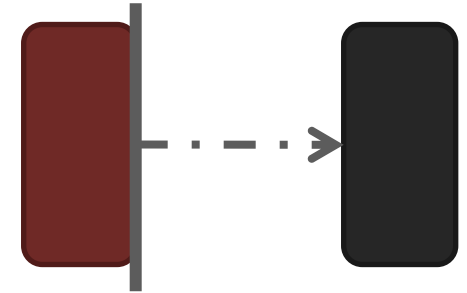
*memmove & memcpy: 5% cycles in Google's datacenter [Kanev+ ISCA'15]*



**Forking**



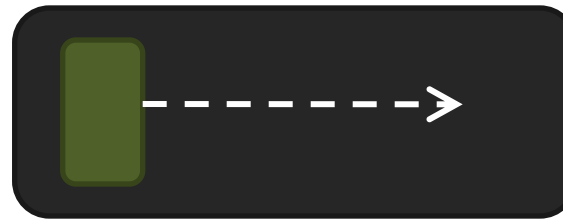
**Zero initialization  
(e.g., security)**



**Checkpointing**



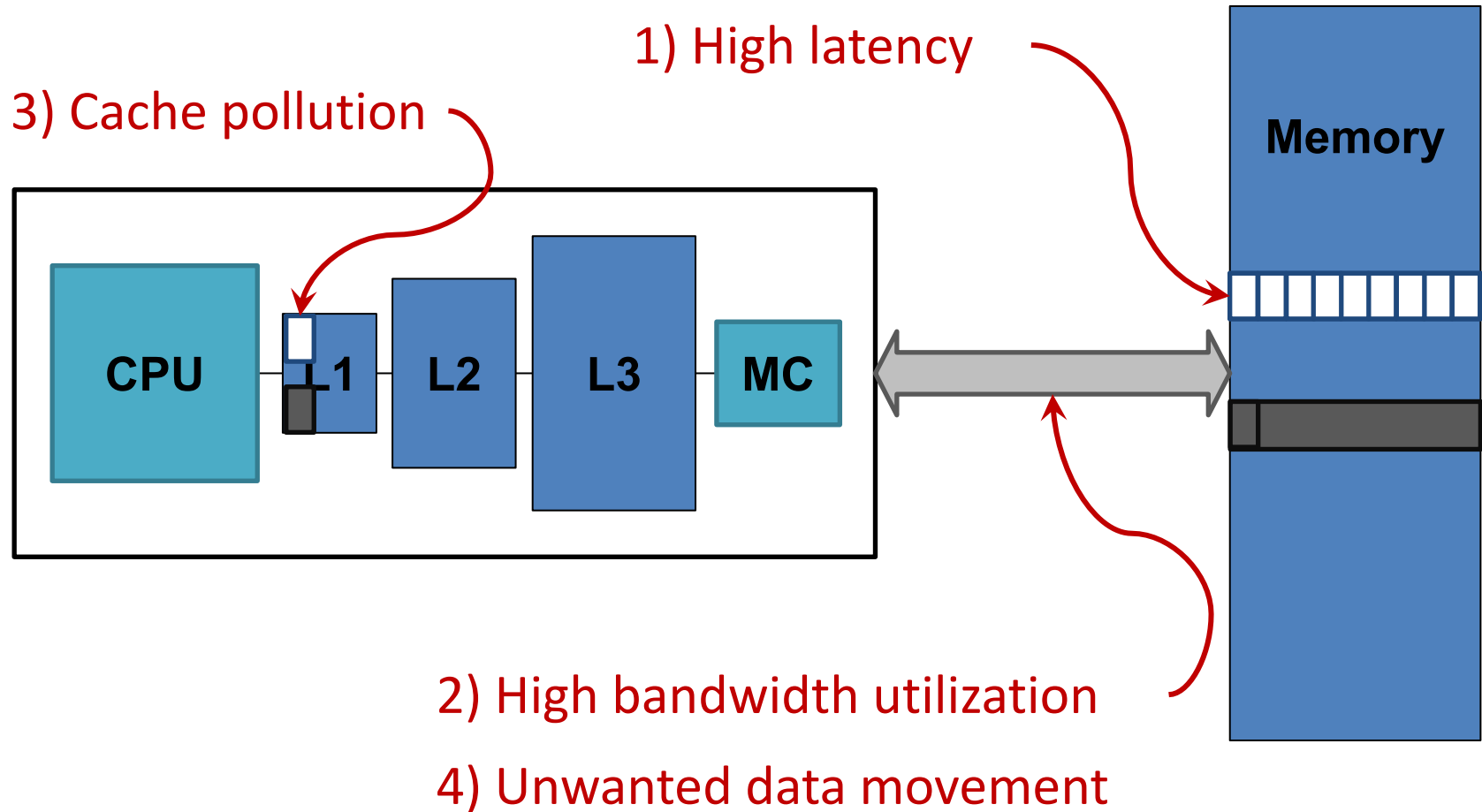
**VM Cloning  
Deduplication**



**Page Migration**

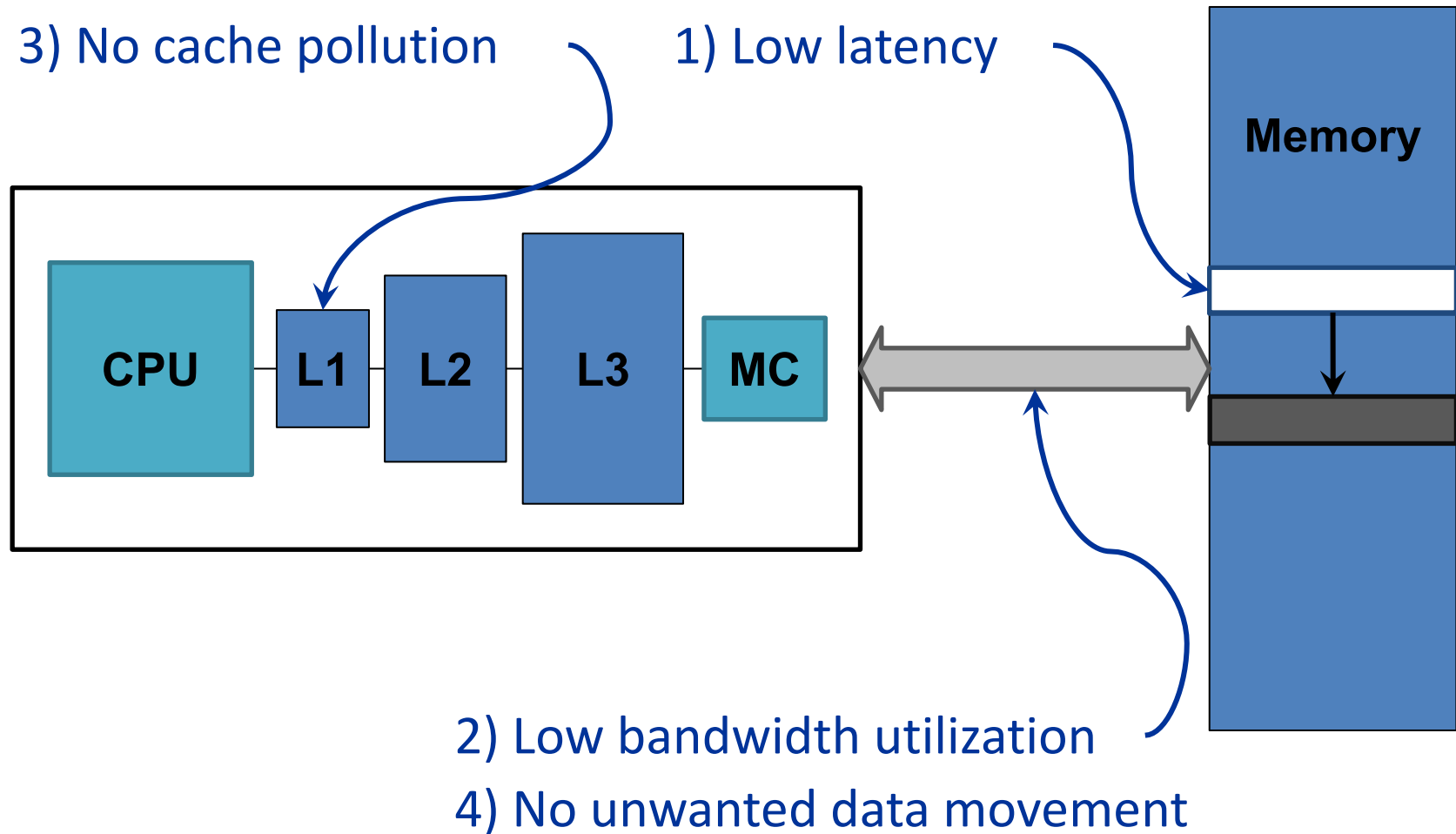
...  
**Many more**

# Today's Systems: Bulk Data Copy



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

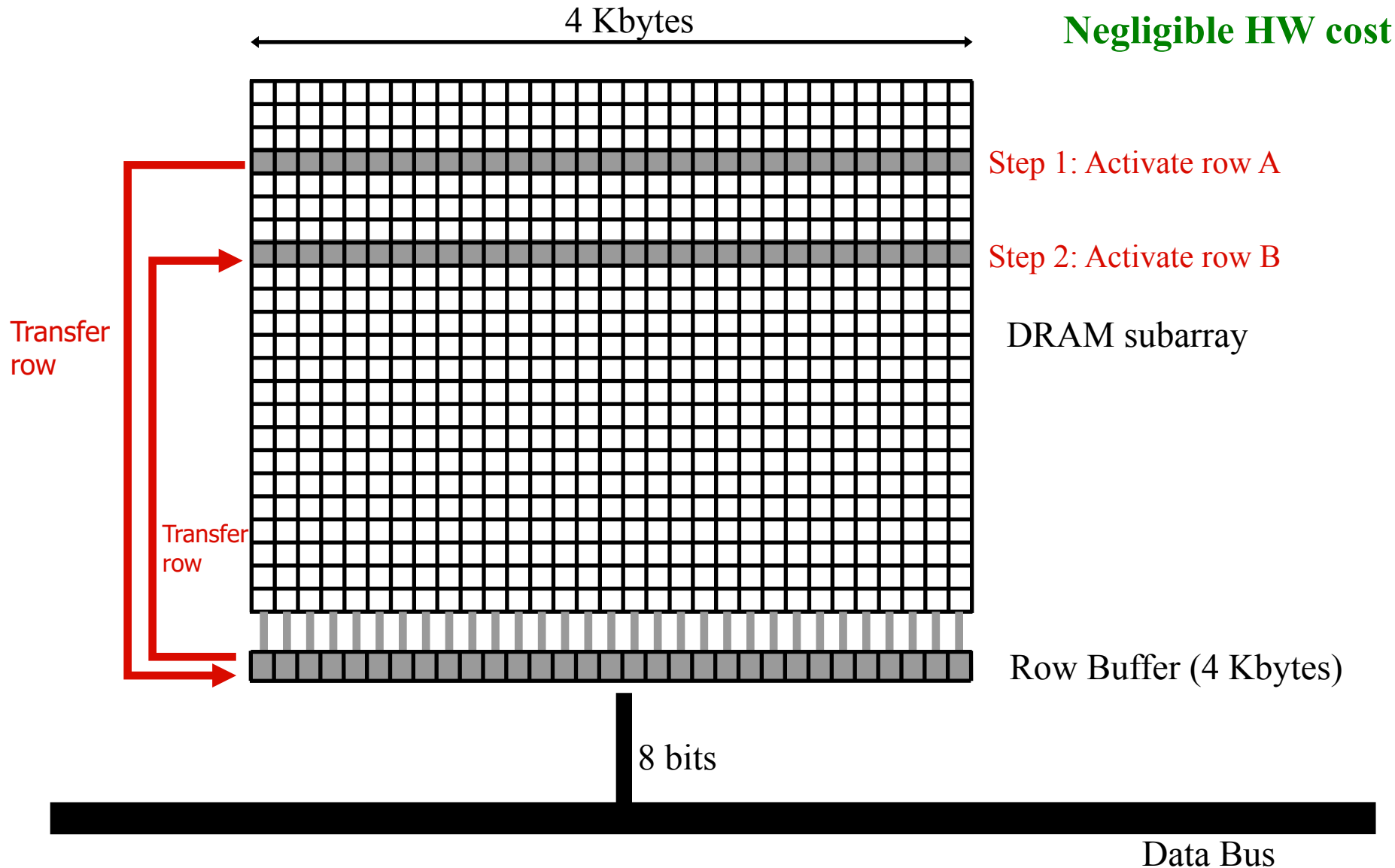
# Future Systems: In-Memory Copy



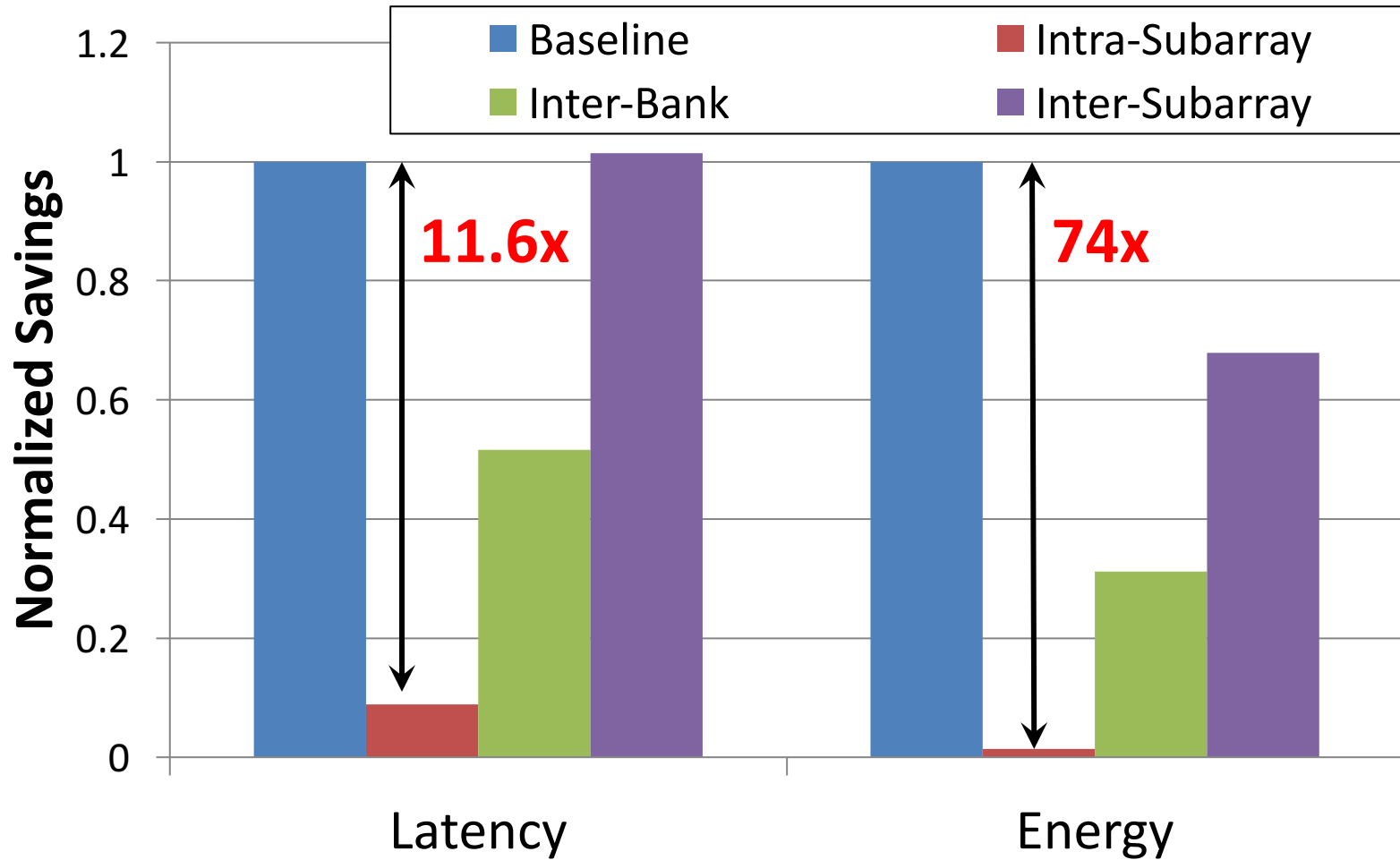
1046ns, 3.6uJ → 90ns, 0.04uJ

# 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

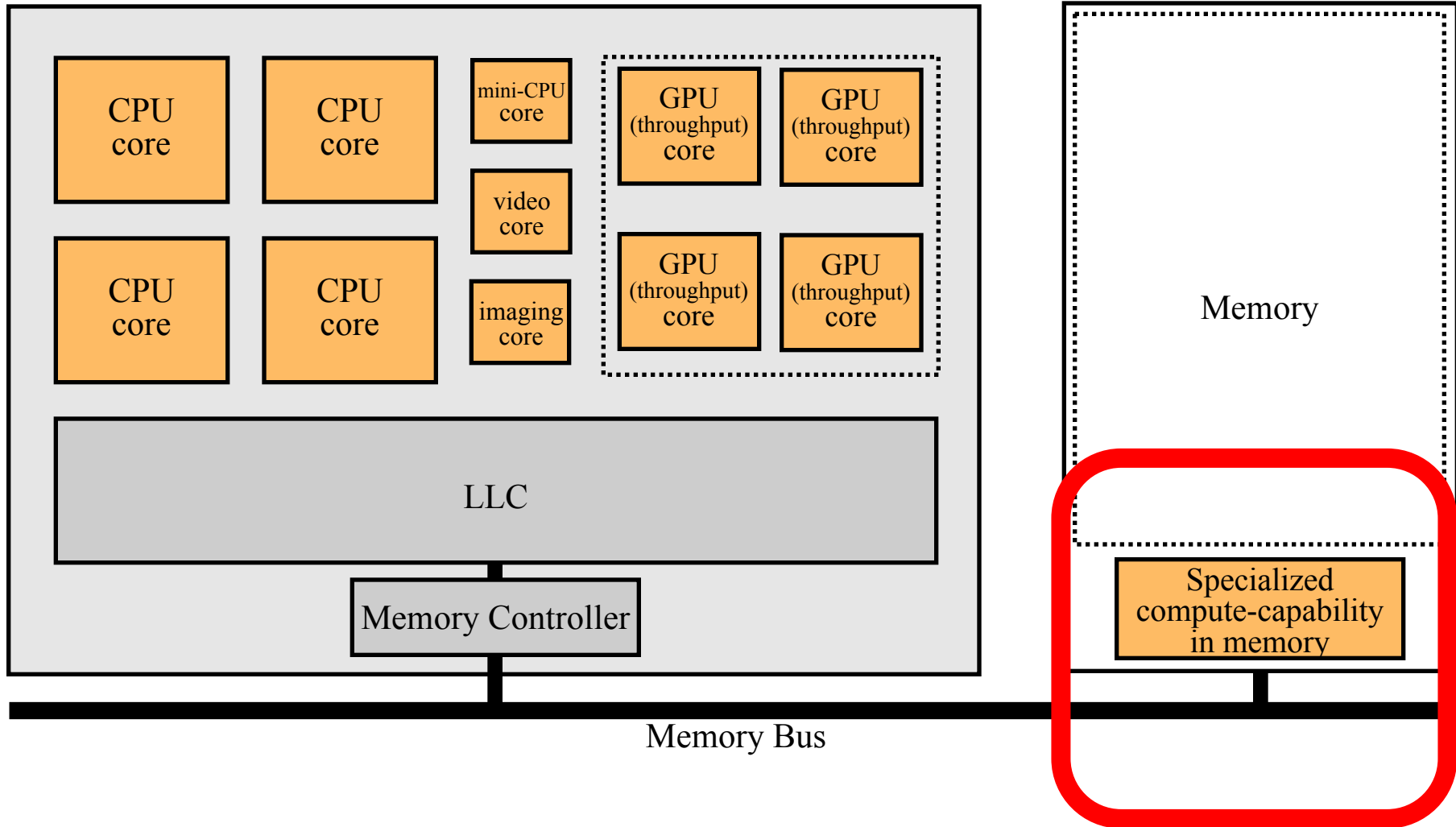
Vivek Seshadri      Yoongu Kim      Chris Fallin\*      Donghyuk Lee  
vseshadr@cs.cmu.edu    yoongukim@cmu.edu    cfallin@c1f.net    donghyuk1@cmu.edu

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**

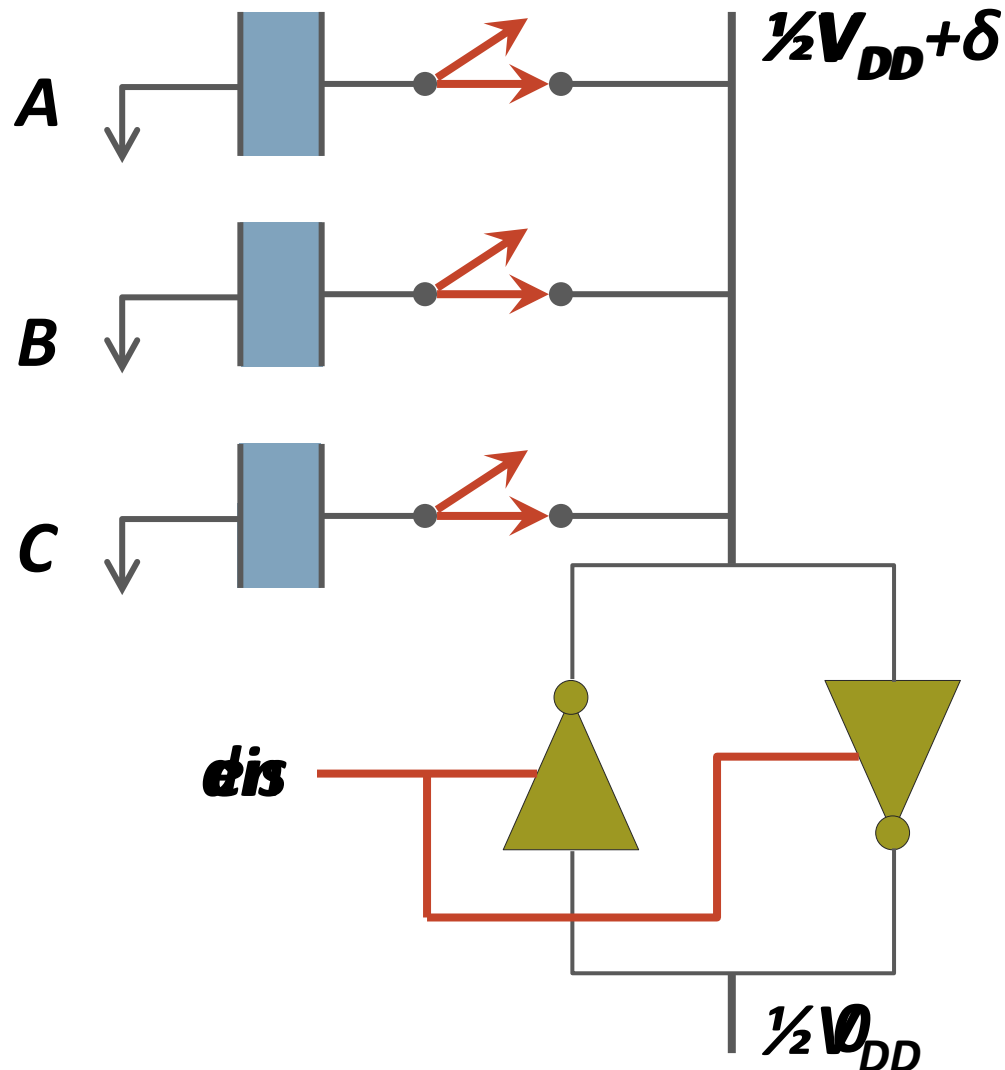


# In-Memory Bulk Bitwise Operations

---

- 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

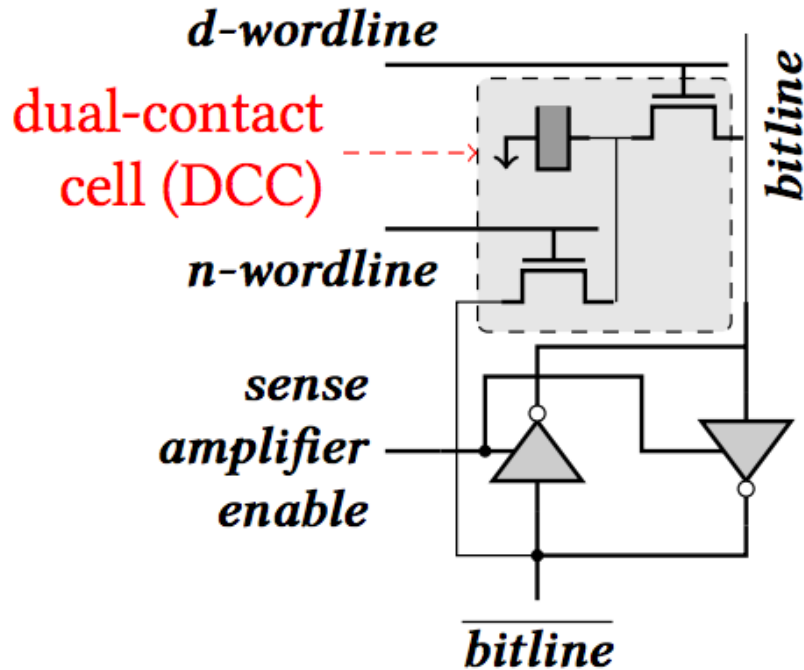
# In-DRAM AND/OR: Triple Row Activation



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

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

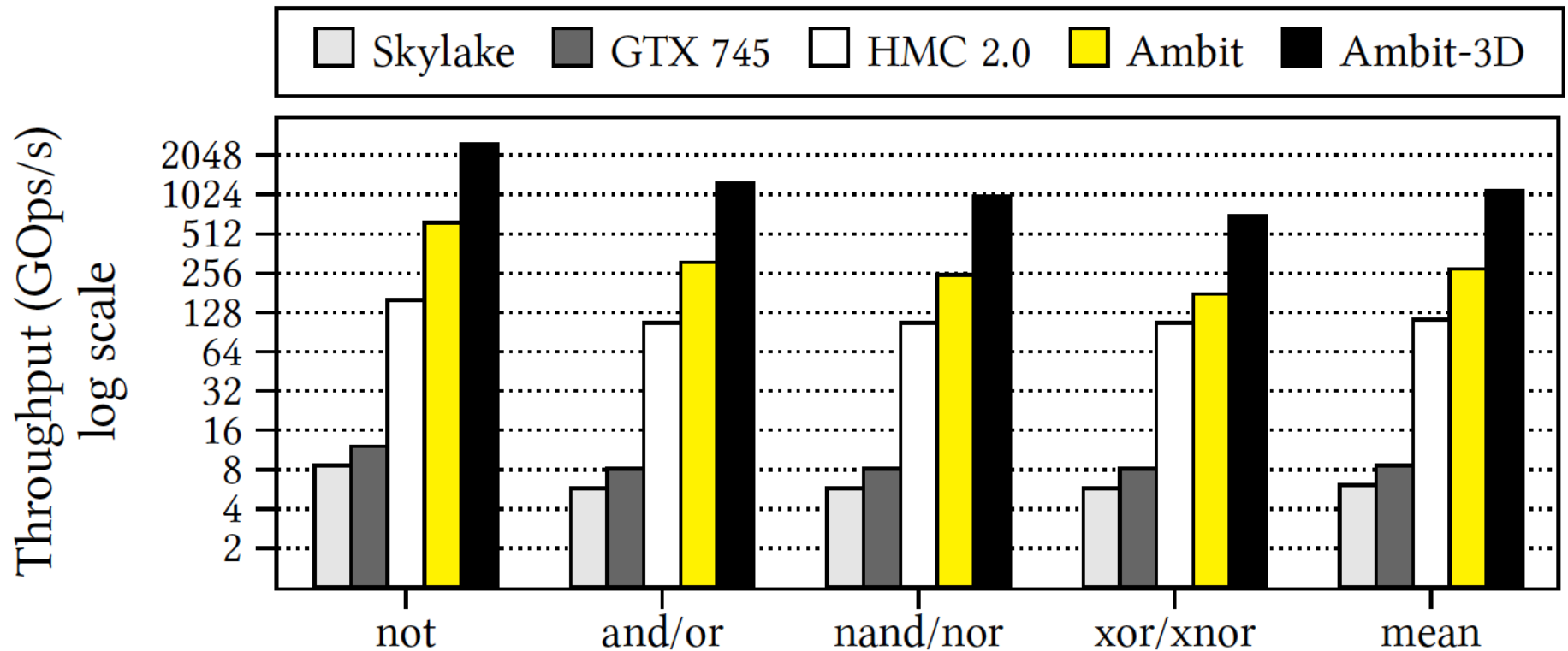
# In-DRAM NOT: Dual Contact Cell



**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

# Performance: In-DRAM Bitwise Operations



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

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

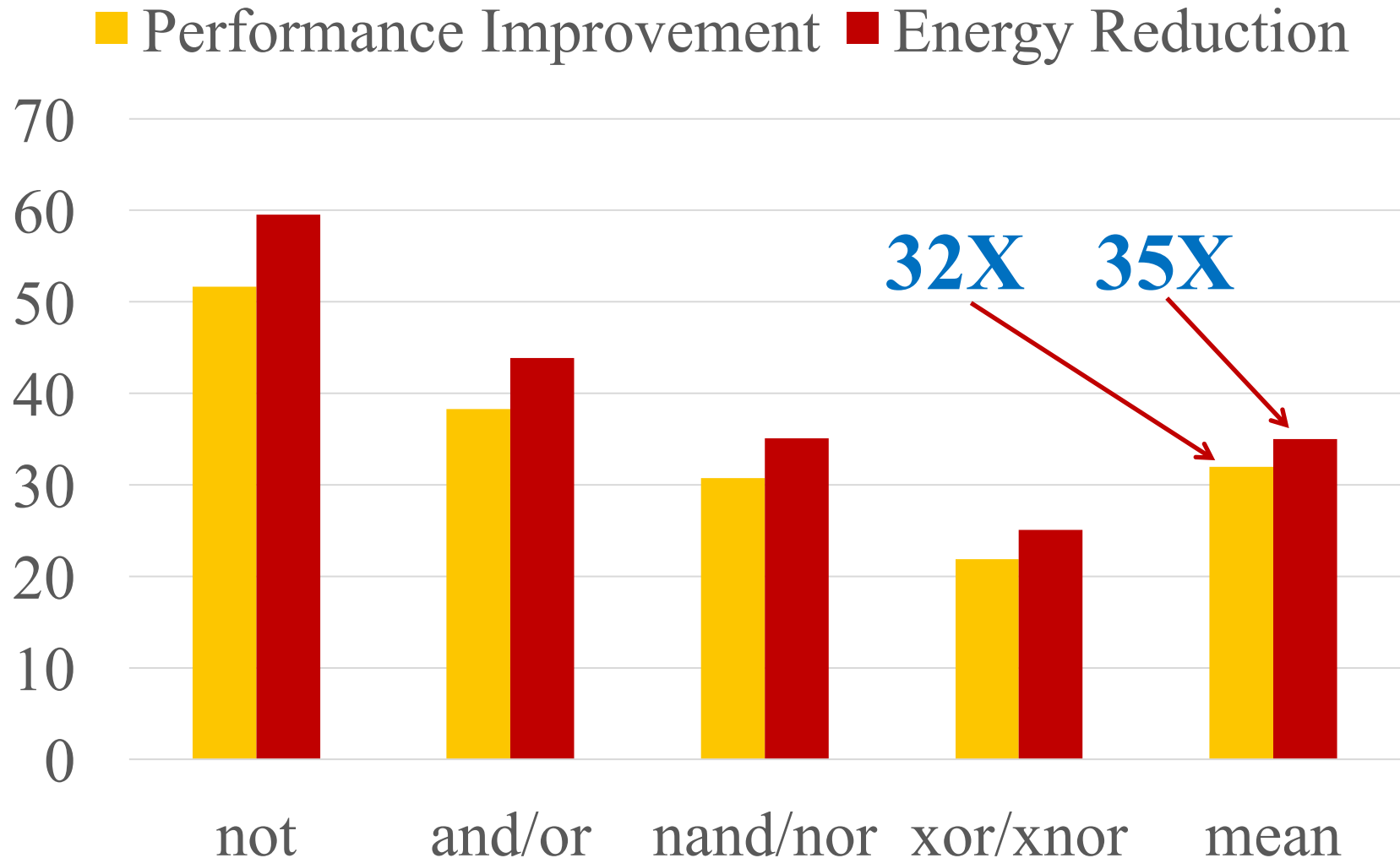
# Energy of In-DRAM Bitwise Operations

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

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

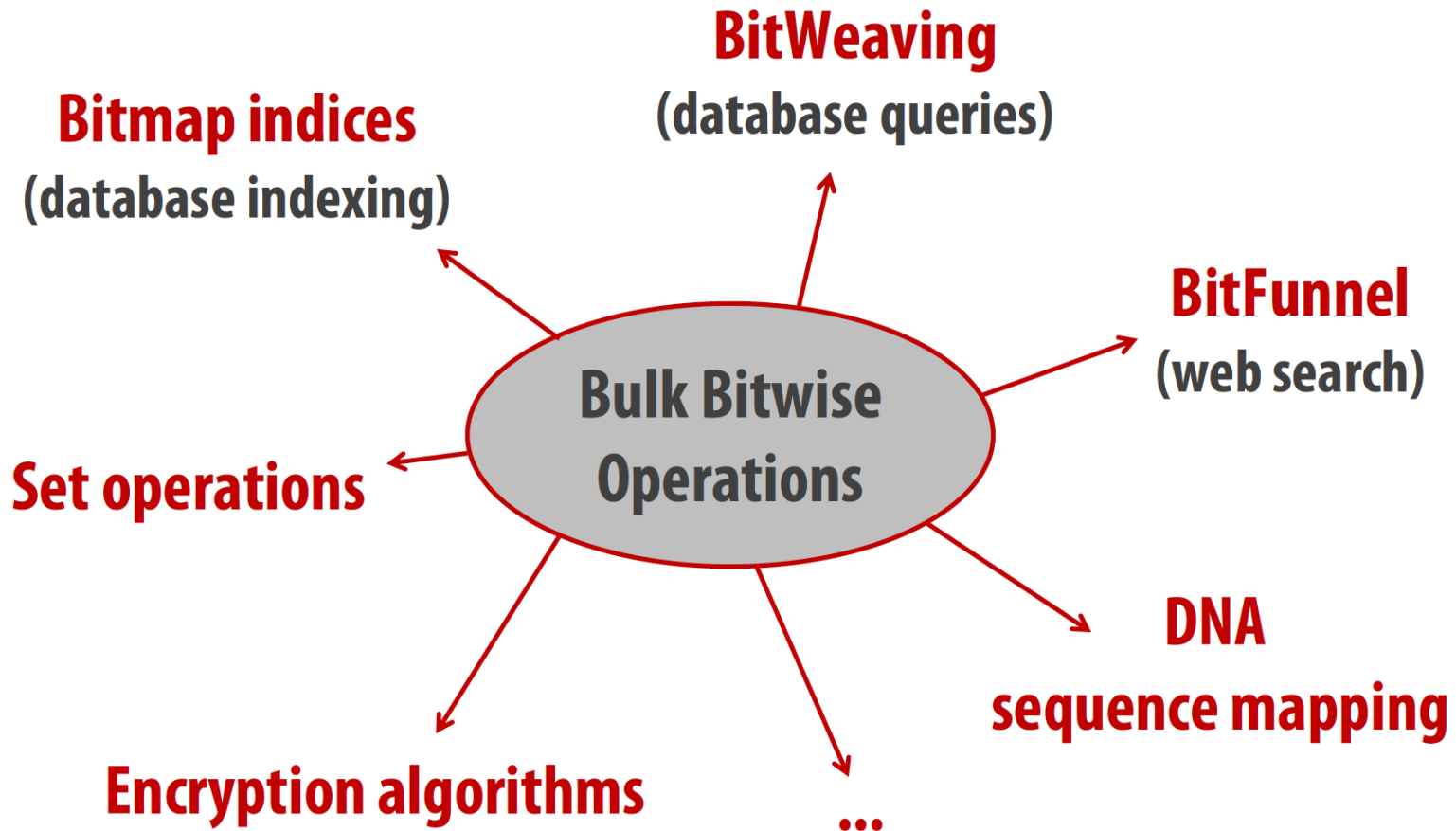
Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

# Ambit vs. DDR3: Performance and Energy



# Bulk Bitwise Operations in Workloads

---





# Example Data Structure: Bitmap Index

---

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

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

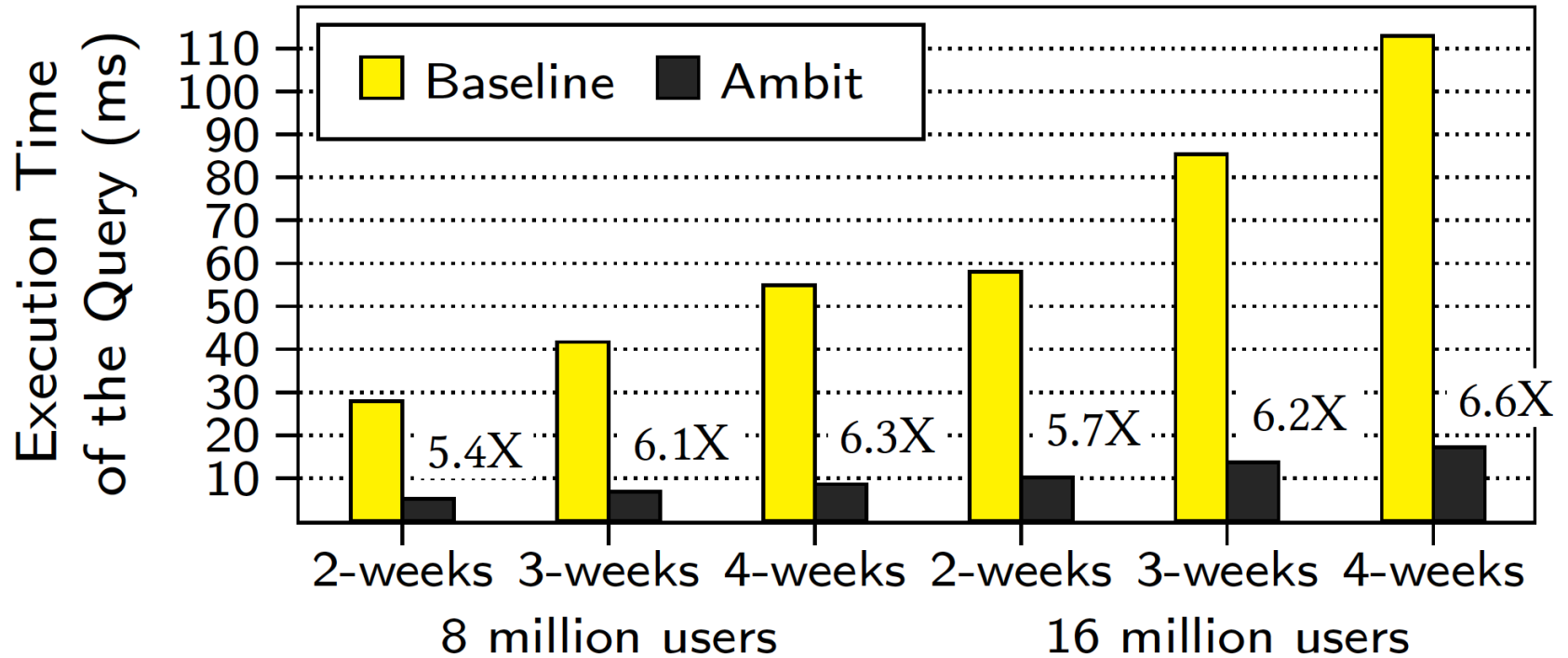
Bitmap 1

Bitmap 2

Bitmap 3

Bitmap 4

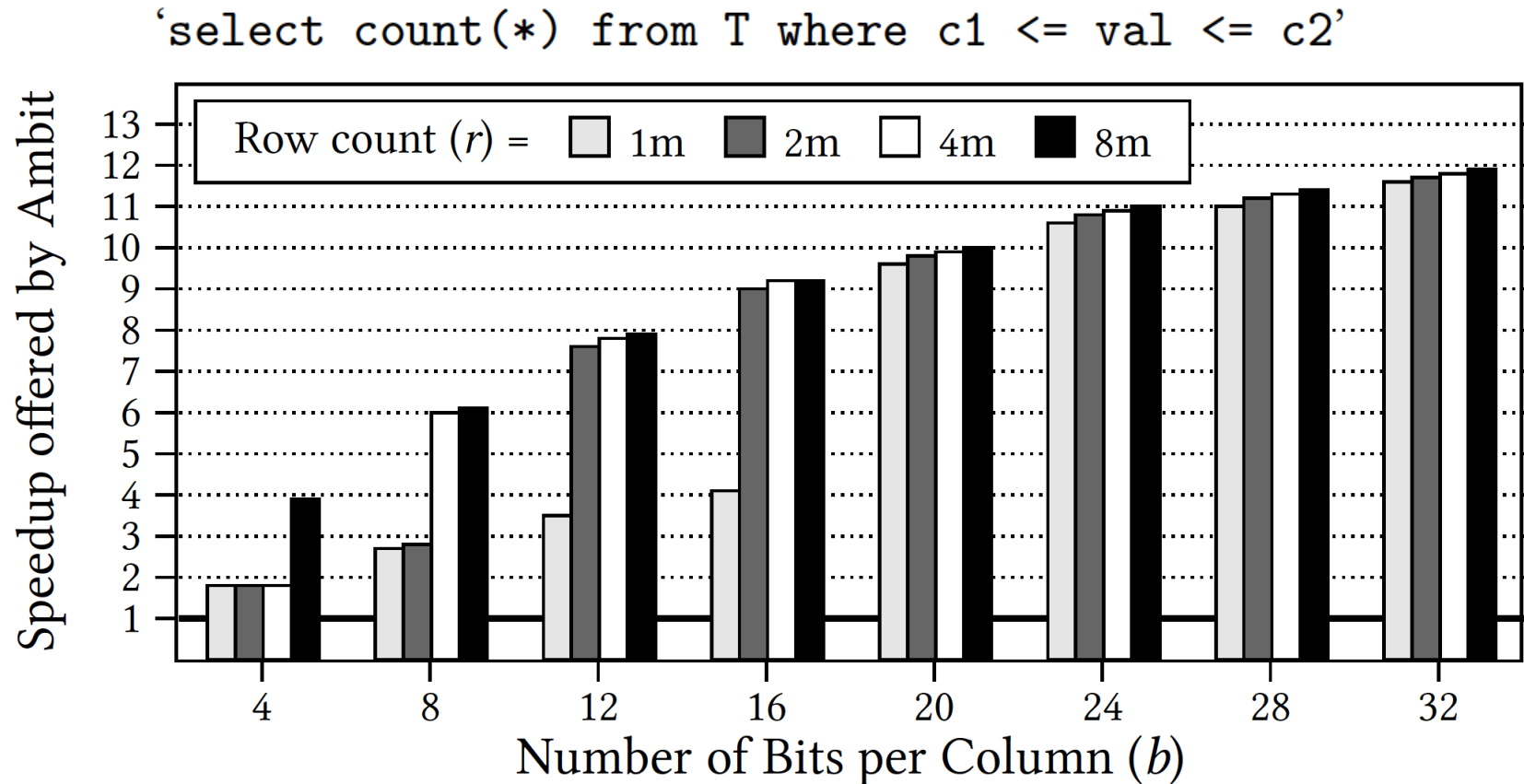
# Performance: Bitmap Index on Ambit



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

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

# Performance: BitWeaving on Ambit



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

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

# More on In-DRAM Bulk AND/OR

---

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

## Fast Bulk Bitwise AND and OR in DRAM

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

\*Carnegie Mellon University

†Intel Pittsburgh

# More on Ambit

---

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

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

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

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

# Computing Architectures with Minimal Data Movement

# Challenge: Intelligent Memory Device

---

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



# Agenda

---

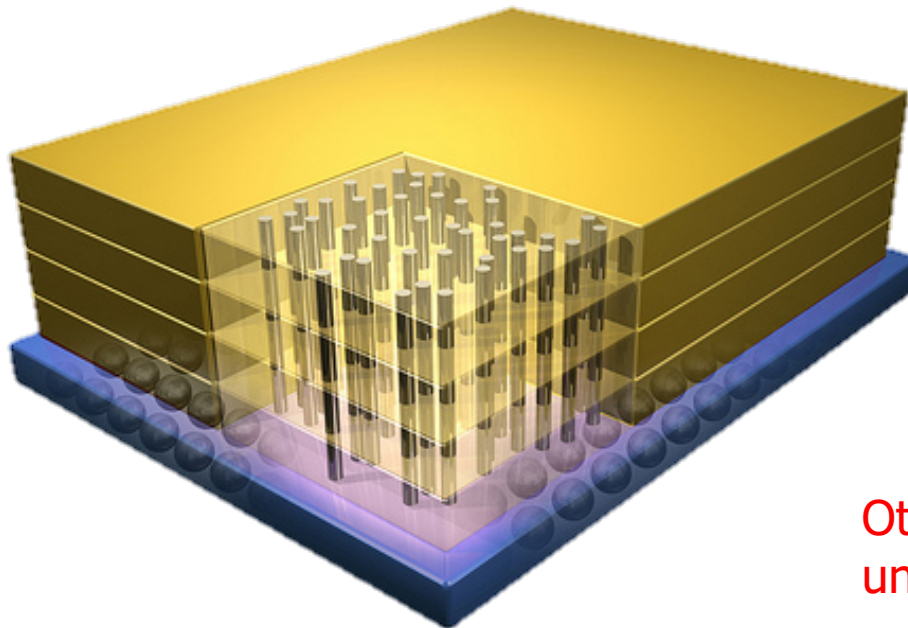
- Major Trends Affecting Main Memory
- The Need for Intelligent Memory Controllers
  - Bottom Up: Push from Circuits and Devices
  - Top Down: Pull from Systems and Applications
- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips
  - Exploiting 3D-Stacked Memory
- How to Enable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

# 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

---

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

# Graph Processing

- Large graphs are everywhere (circa 2015)



36 Million  
Wikipedia Pages



1.4 Billion  
Facebook Users

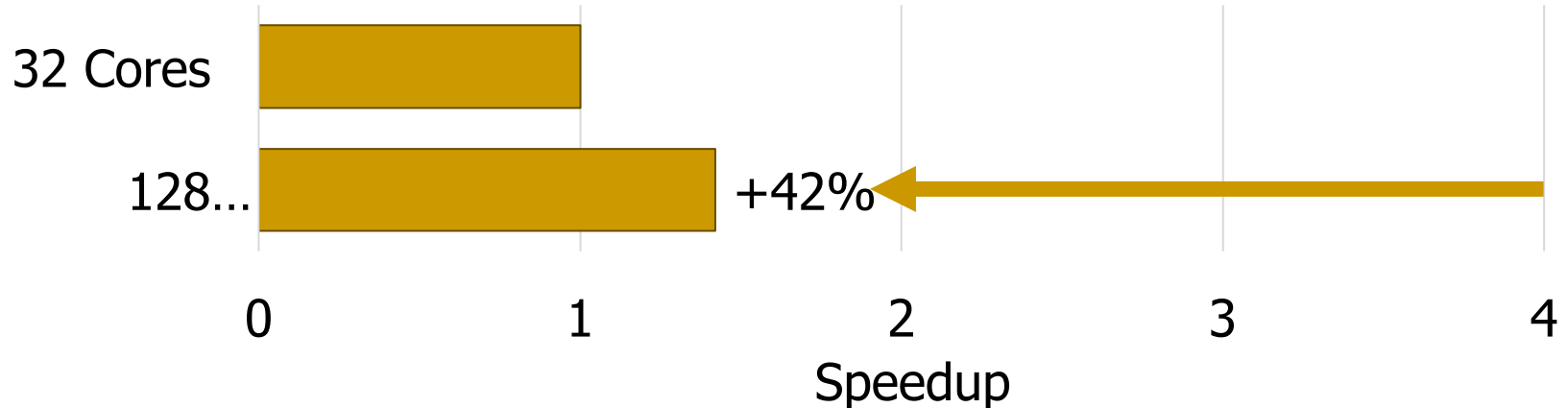


300 Million  
Twitter Users



30 Billion  
Instagram Photos

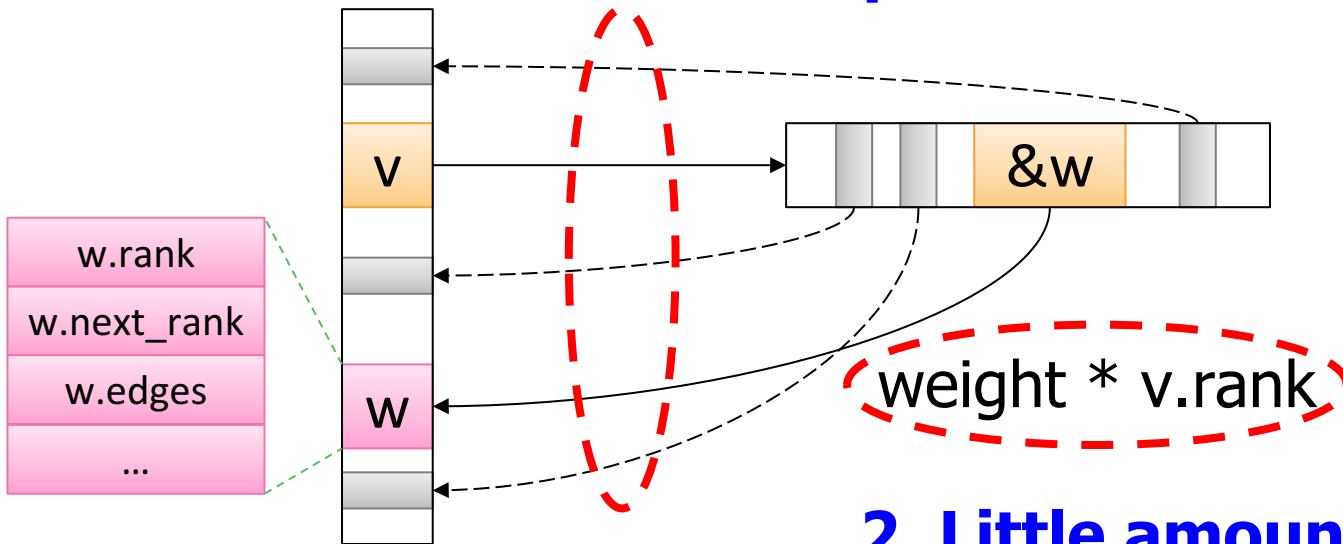
- Scalable large-scale graph processing is challenging



# Key Bottlenecks in Graph Processing

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

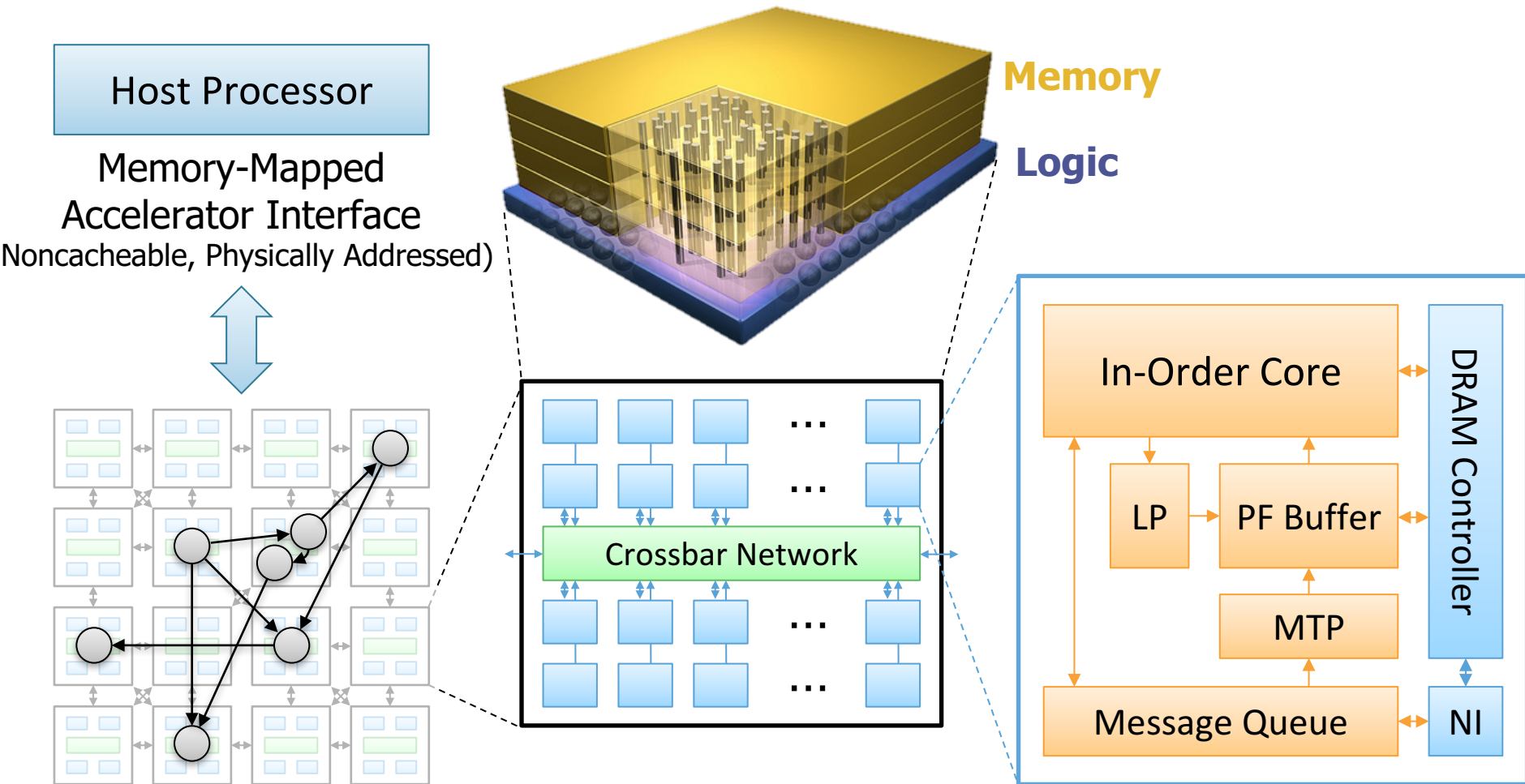
**1. Frequent random memory accesses**



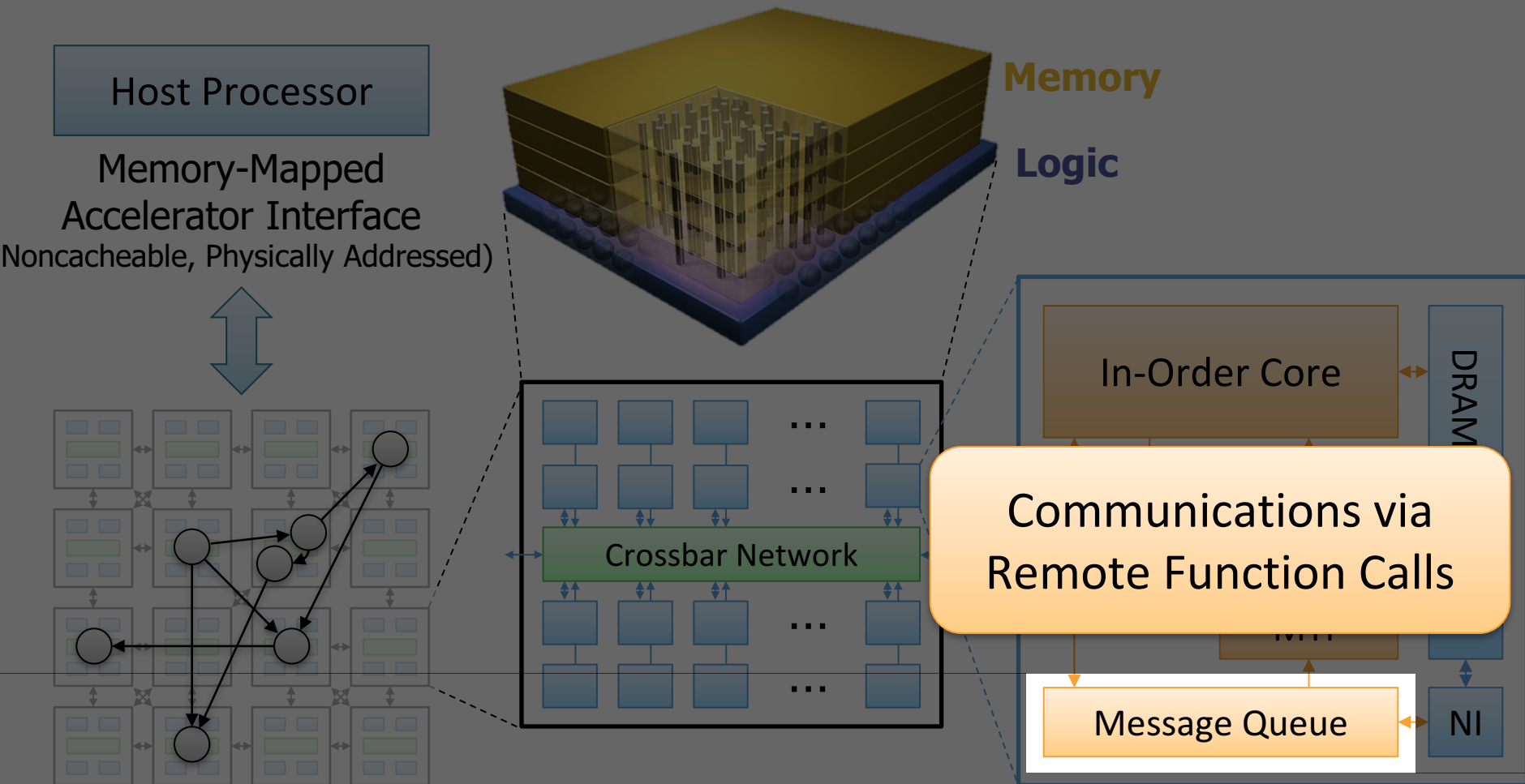
**2. Little amount of computation**

# Tesseract System for Graph Processing

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



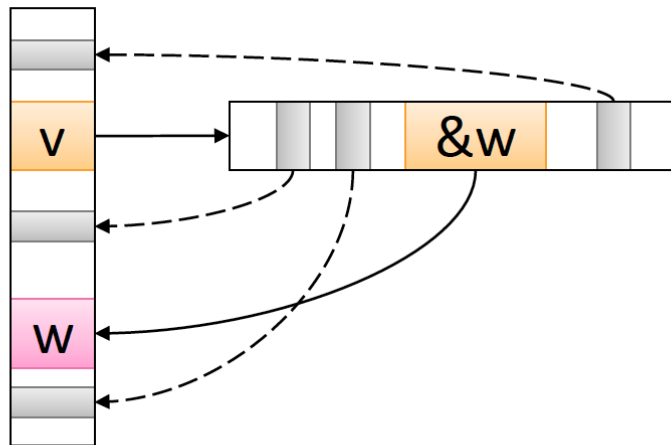
# Tesseract System for Graph Processing





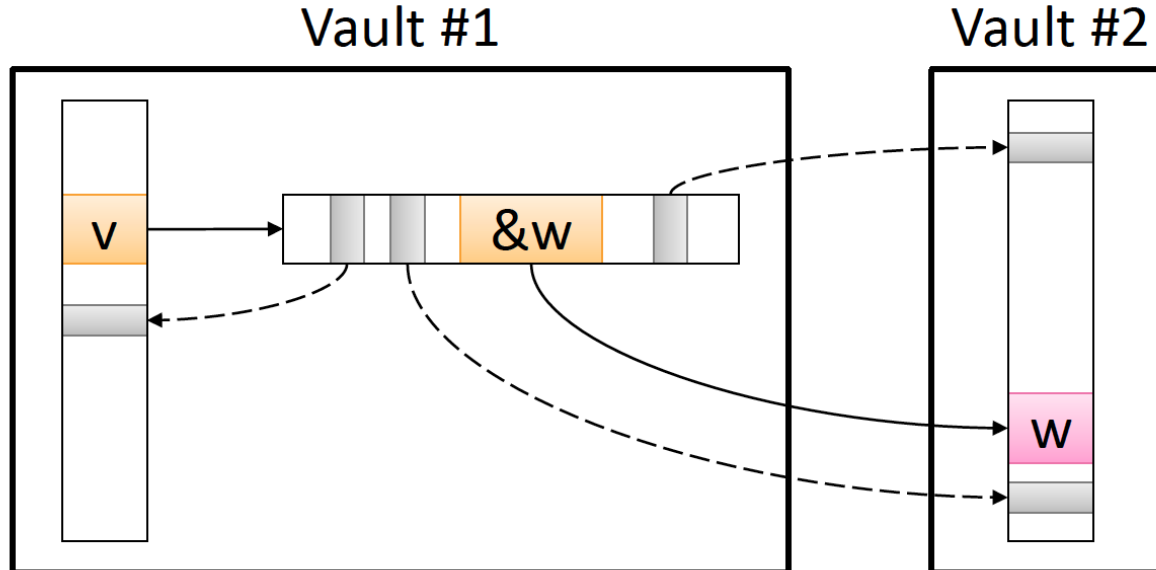
# Communications In Tesseract (I)

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



# Communications In Tesseract (II)

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

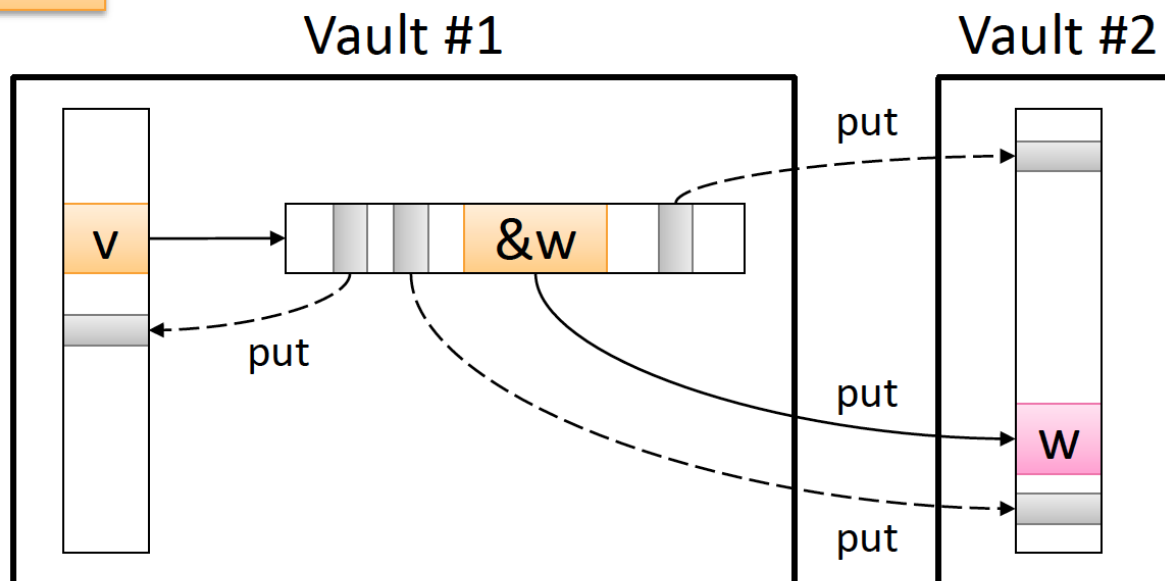


# Communications In Tesseract (III)

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

**Non-blocking Remote Function Call**

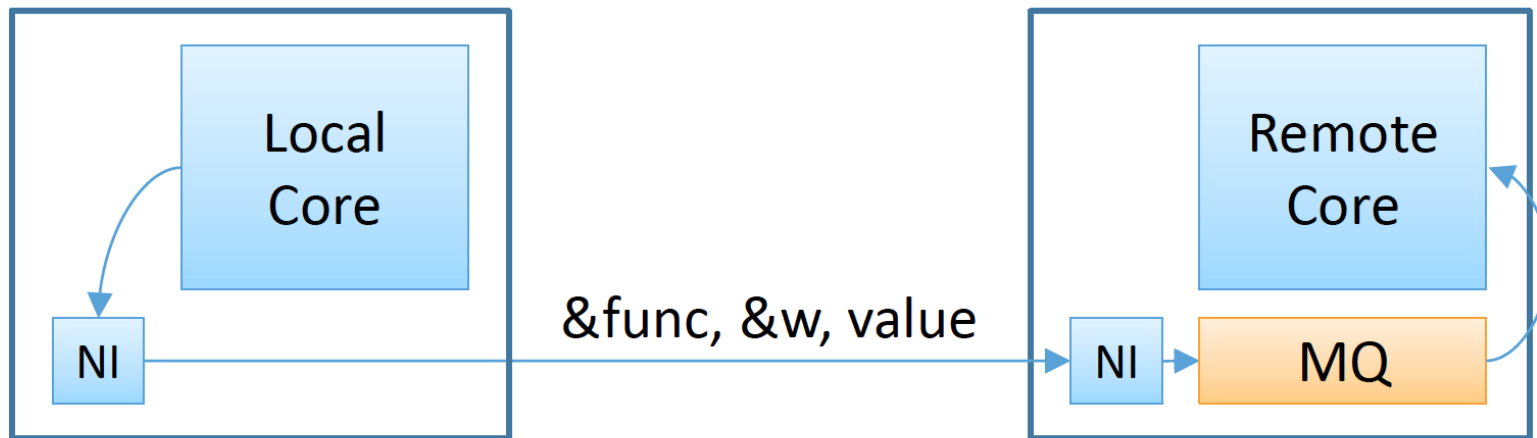
Can be **delayed**  
until the nearest barrier



# Remote Function Call (Non-Blocking)

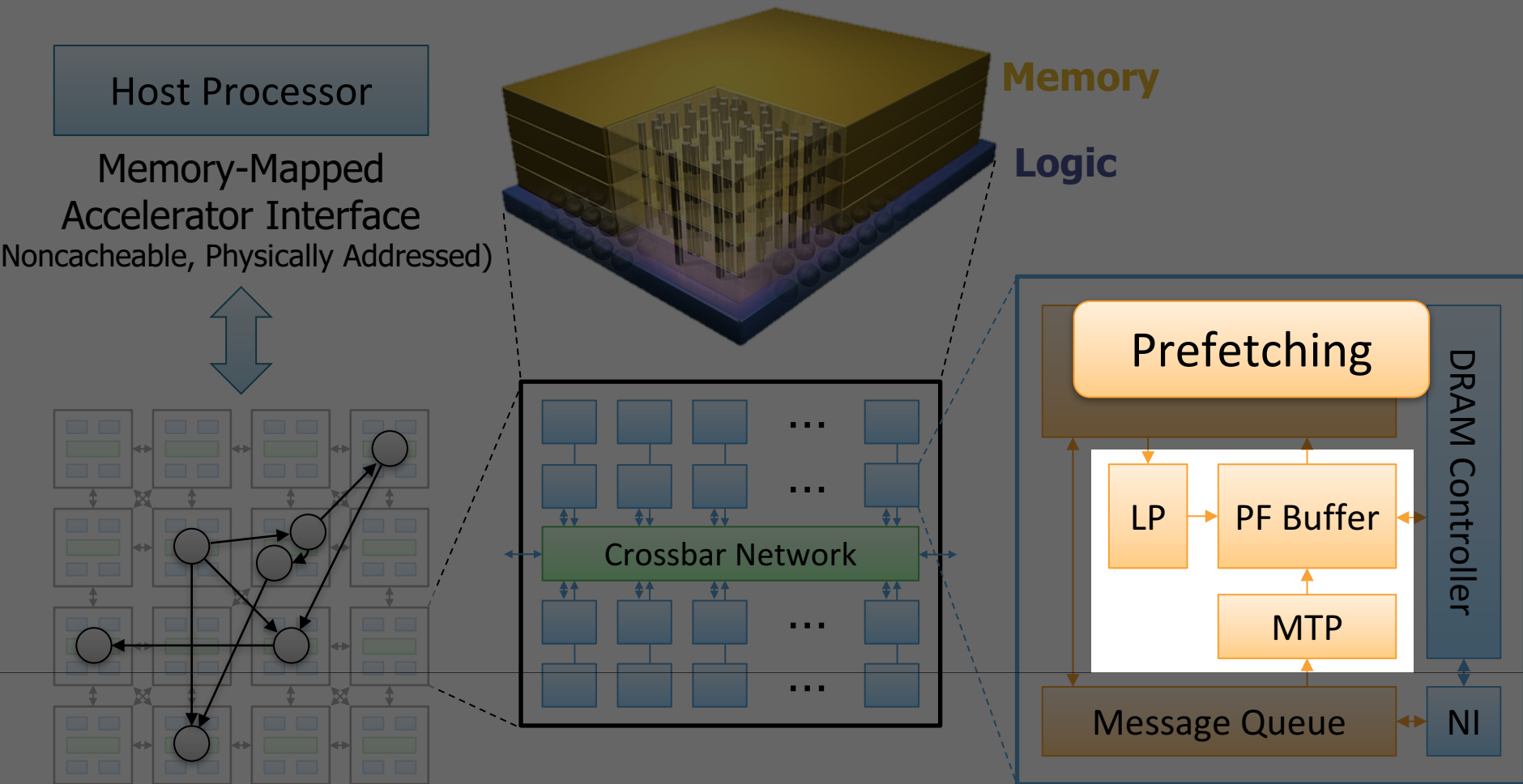
---

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



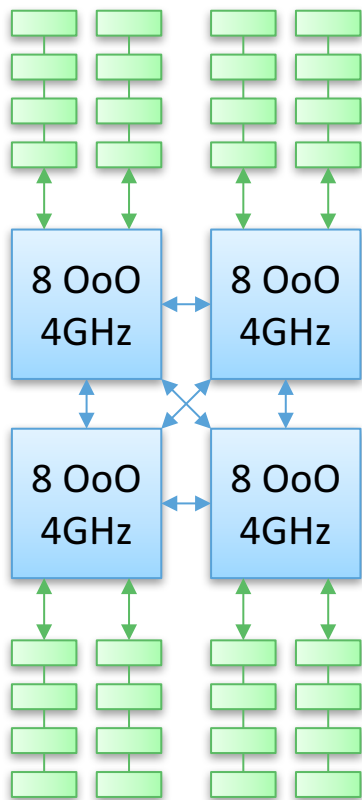
```
put(w.id, function() { w.next_rank += value; })
```

# Tesseract System for Graph Processing



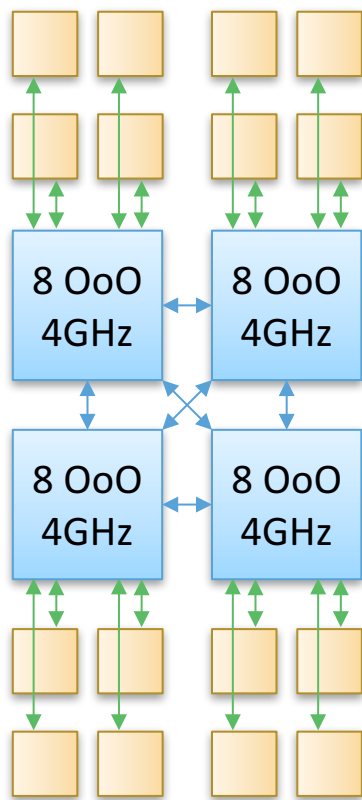
# Evaluated Systems

DDR3-OoO



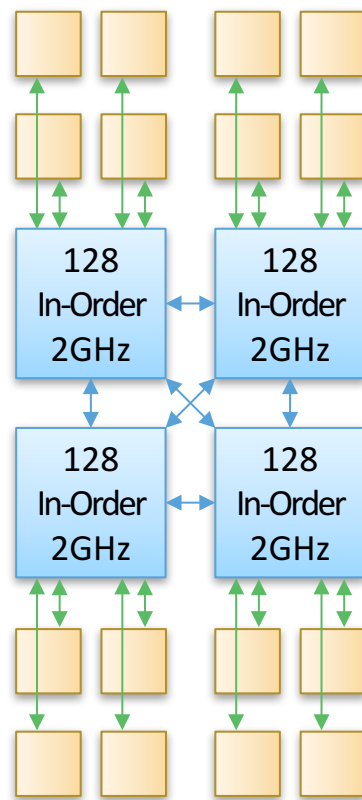
102.4GB/s

HMC-OoO



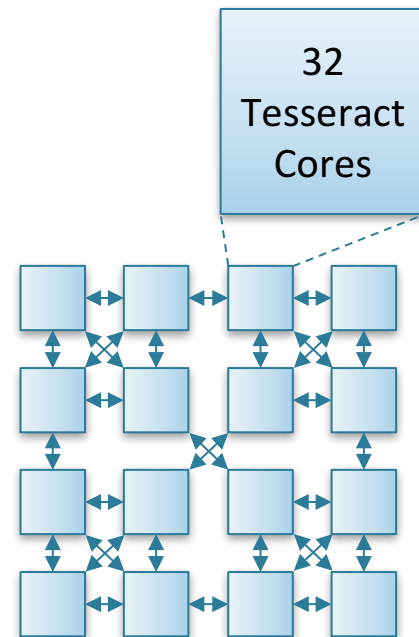
640GB/s

HMC-MC



640GB/s

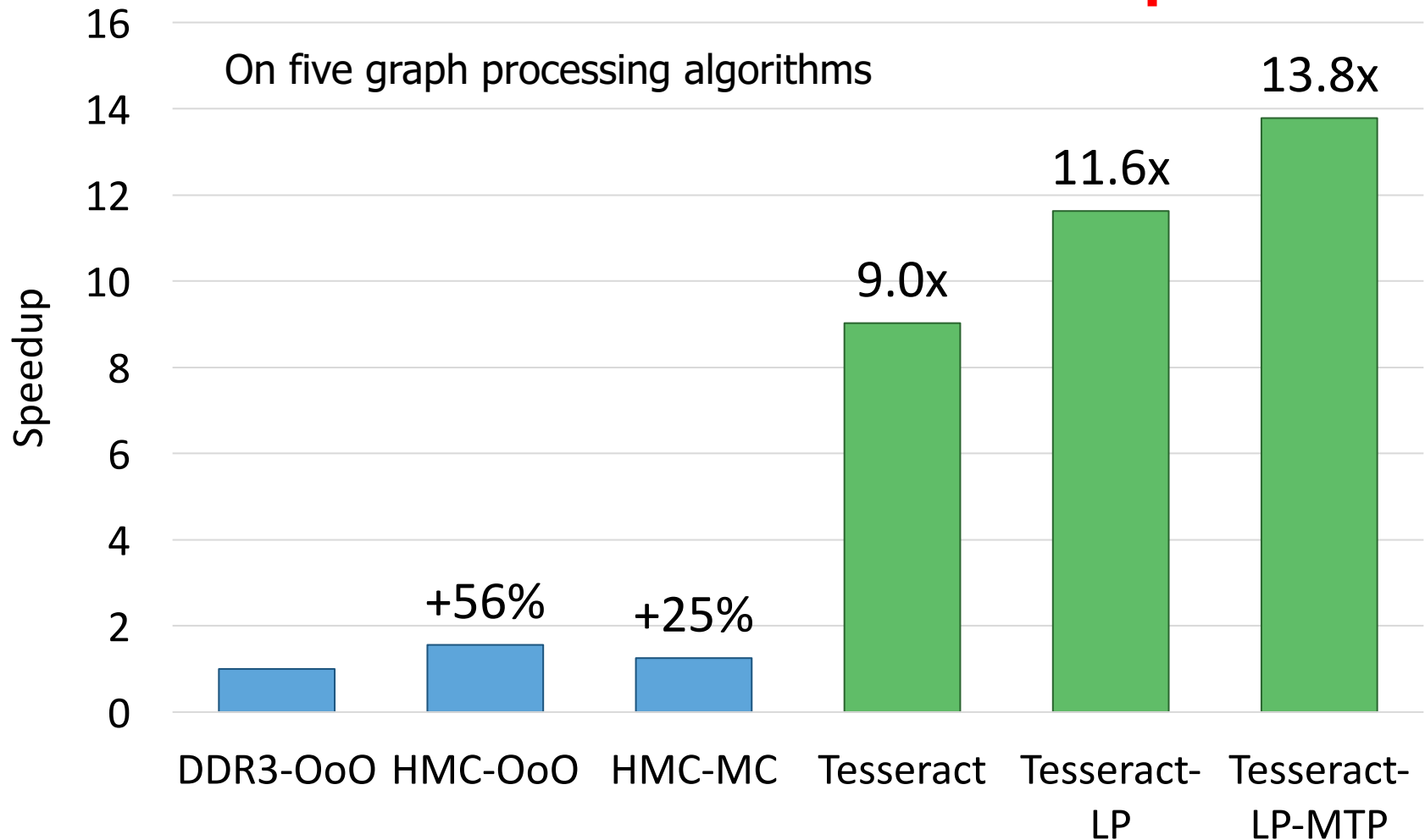
**Tesseract**



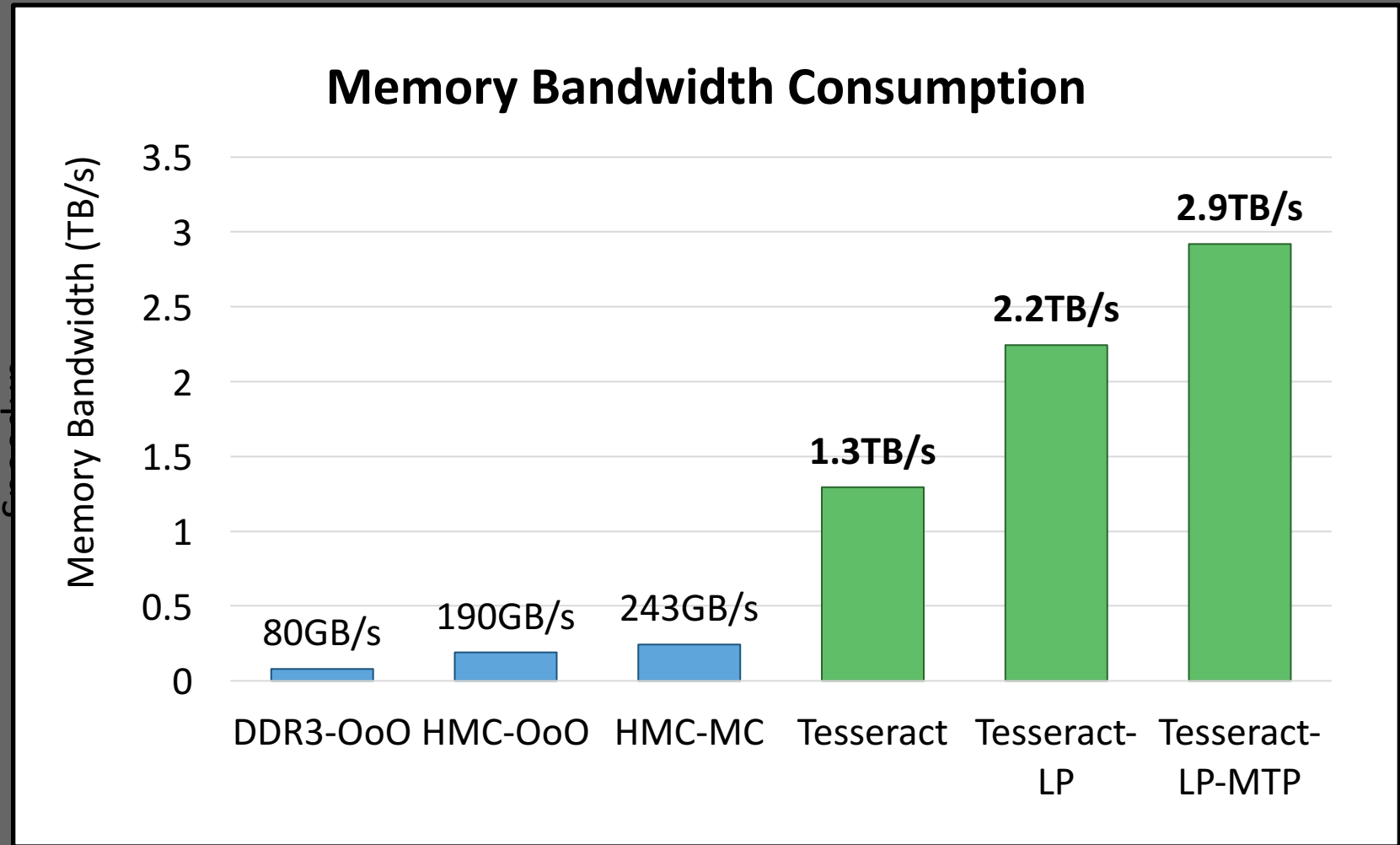
**8TB/s**

# Tesseract Graph Processing Performance

**>13X Performance Improvement**

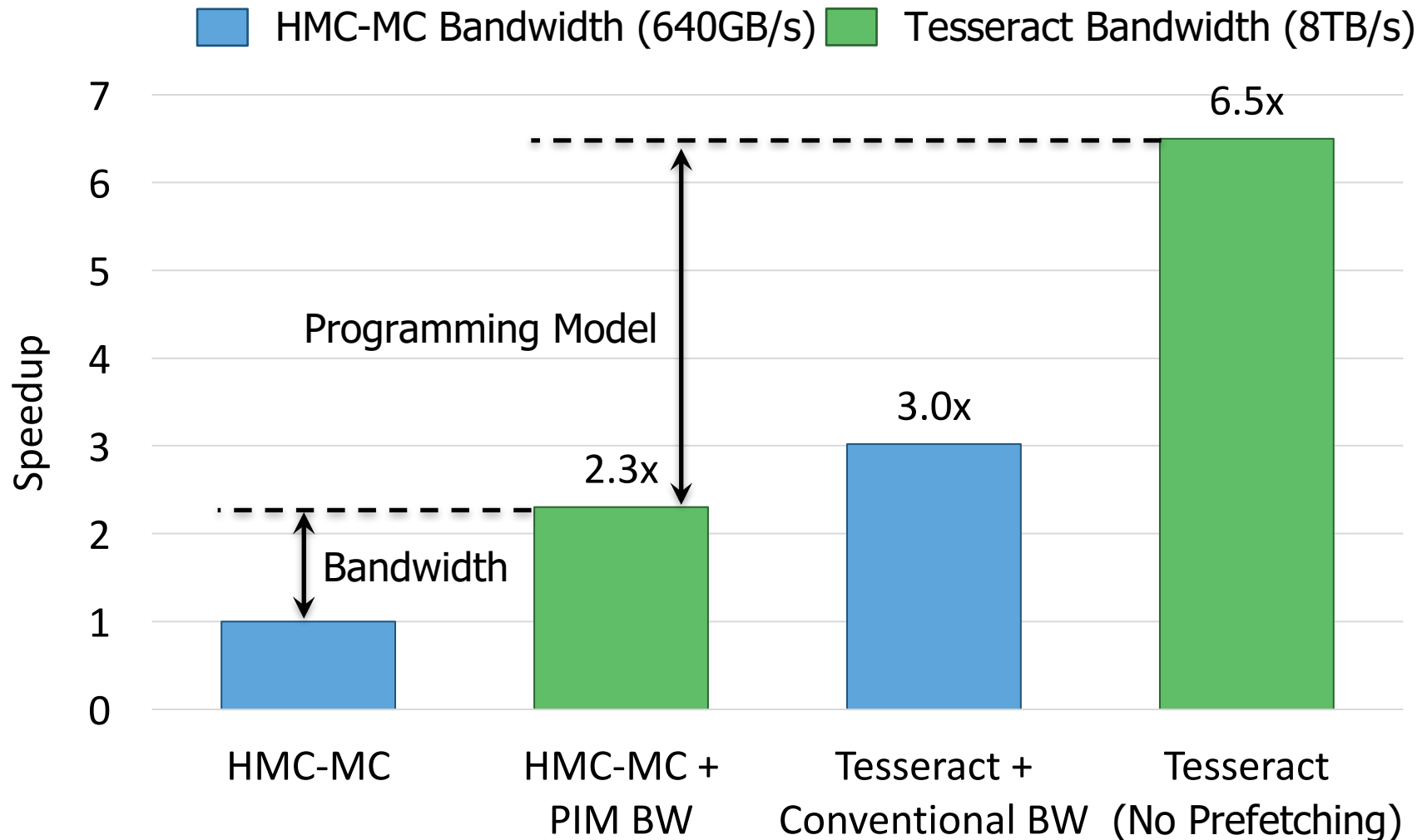


# Tesseract Graph Processing Performance

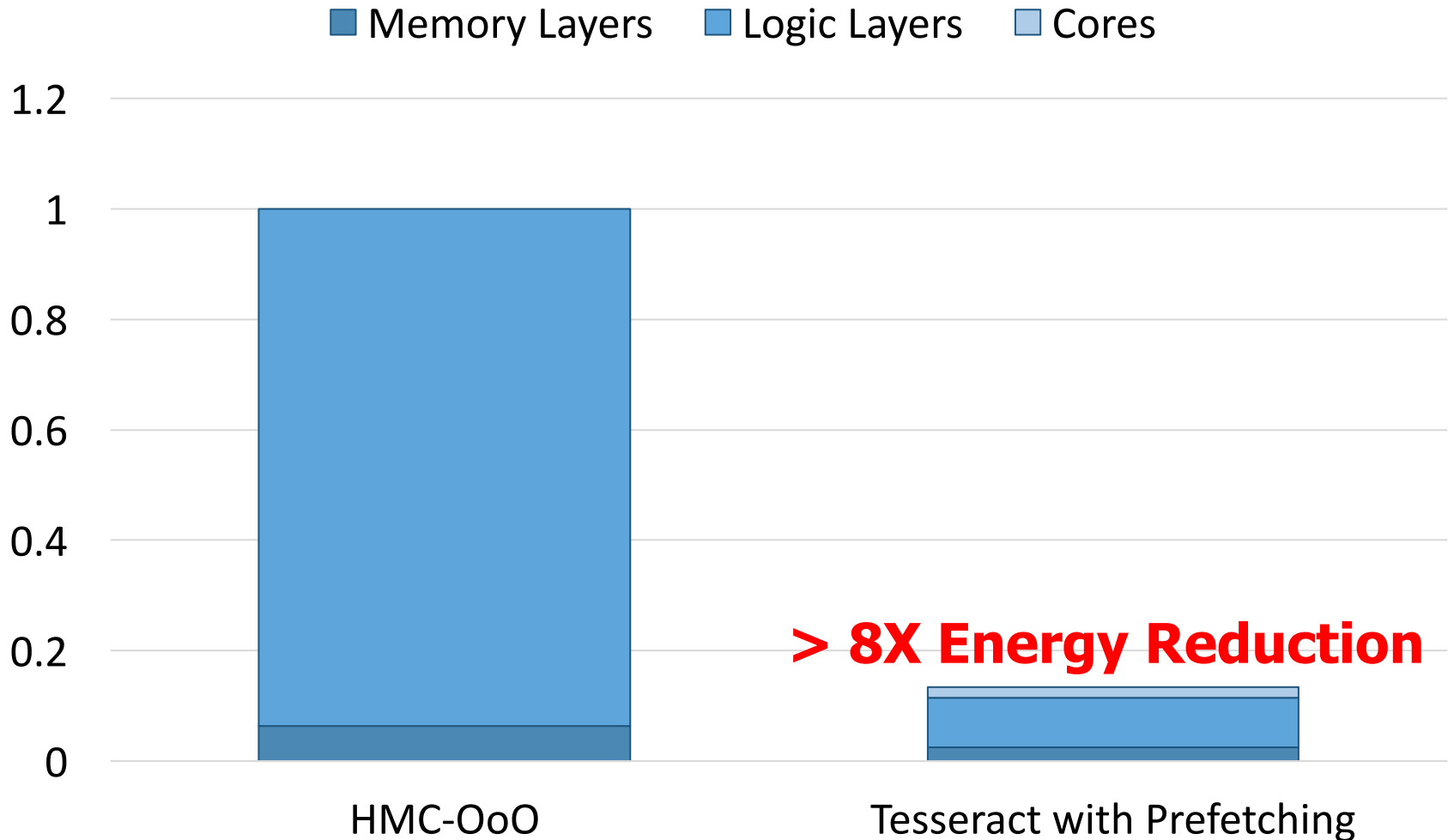




# Effect of Bandwidth & Programming Model



# Tesseract Graph Processing System Energy



# More on Tesseract

---

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

# Truly Distributed GPU Processing with PIM?

```
__global__
void applyScaleFactorsKernel( uint8_T * const out,
    uint8_T const * const in, const double *factor,
    size_t const numRows, size_t const numCols )
{
    // Work out which pixel we are working on.
    const int rowIdx = blockIdx.x * blockDim.x + threadIdx.x;
    const int colIdx = blockIdx.y;
    const int sliceIdx = threadIdx.z;

    // Check this thread isn't off the image
    if( rowIdx >= numRows ) return;

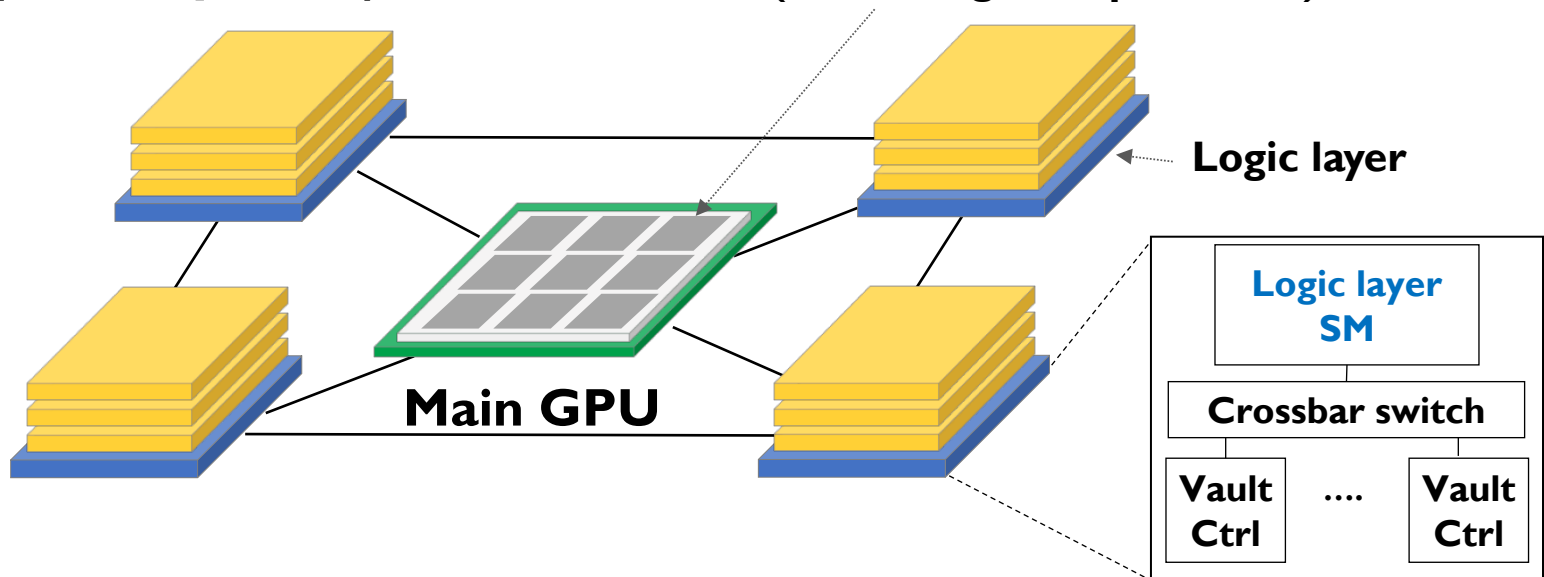
    // Compute the index of my element
    size_t linearIdx = rowIdx + colIdx*numRows +
        sliceIdx*numRows*numCols;
```

## 3D-stacked memory (memory stack)

## SM (Streaming Multiprocessor)

## Logic layer

## Main GPU



# Accelerating GPU Execution with PIM (I)

---

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

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

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

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

## Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems

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

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

# Accelerating GPU Execution with PIM (II)

---

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

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

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

# Two Key Questions in 3D-Stacked PIM

---

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

# PIM-Enabled Instructions

---

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and 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



# Automatic Code and Data Mapping?

---

- Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler, **"Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"**  
*Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)*, Seoul, South Korea, June 2016.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Lightning Session Slides \(pptx\)](#)] [[pdf](#)]

## Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems

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

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

# Fundamentally Energy-Efficient (Data-Centric) Computing Architectures

# Fundamentally Low-Latency (Data-Centric) Computing Architectures

# Agenda

---

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

## How to Enable Adoption of Processing in Memory

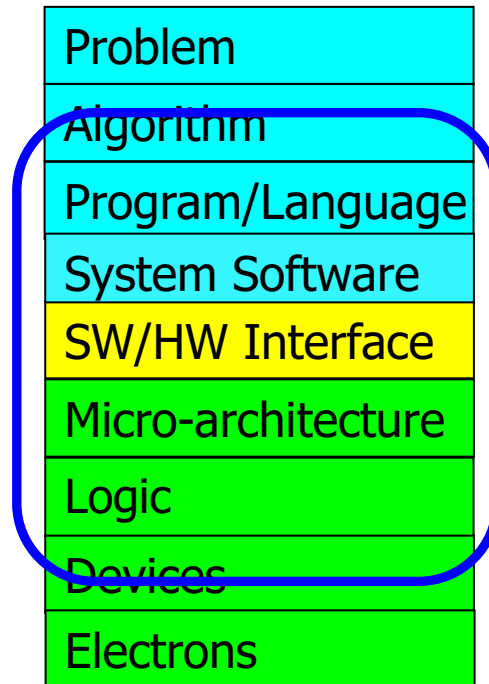
# Barriers to Adoption of PIM

---

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

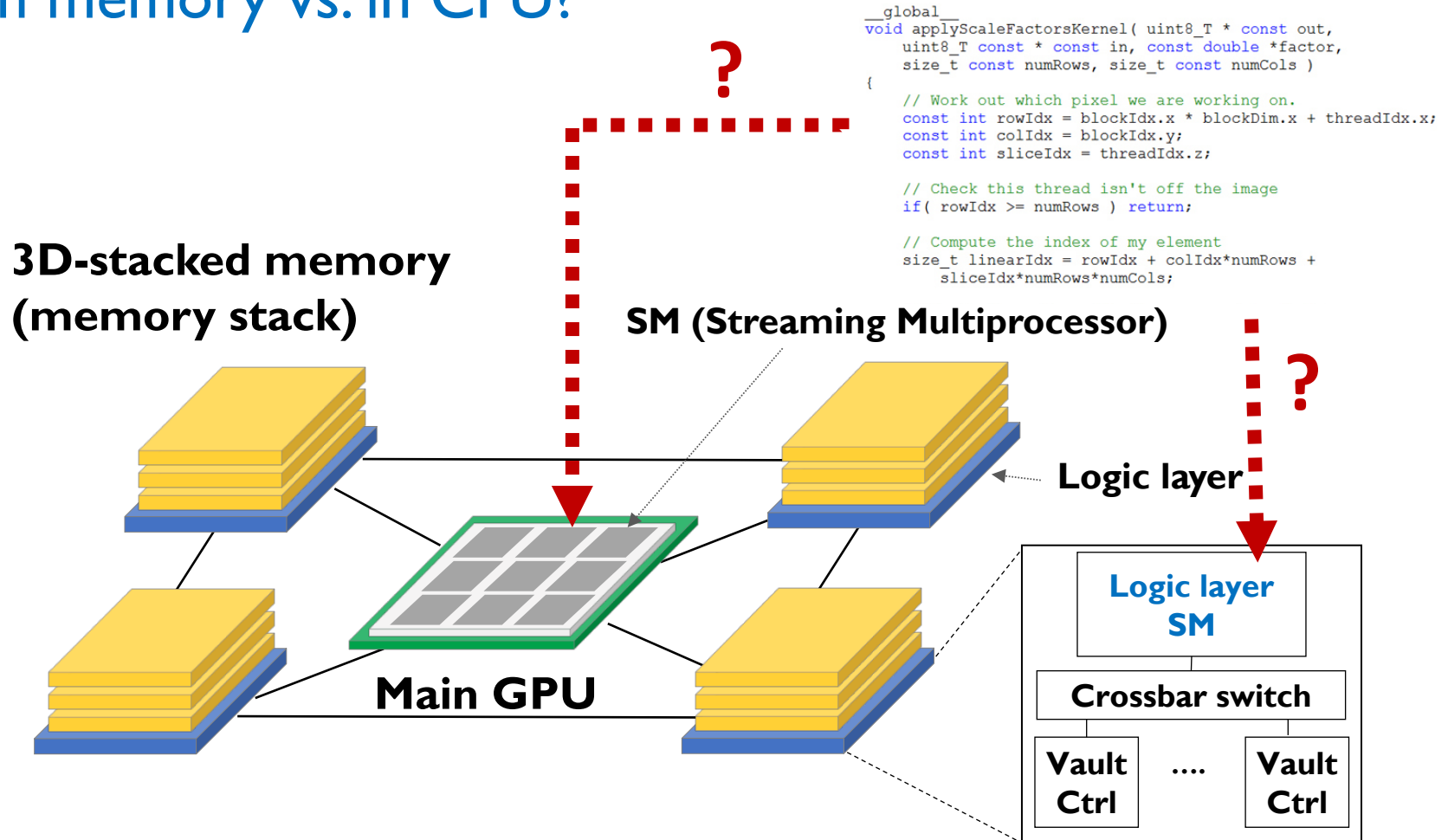
# We Need to Revisit the Entire Stack

---



# Key Challenge 1: Code Mapping

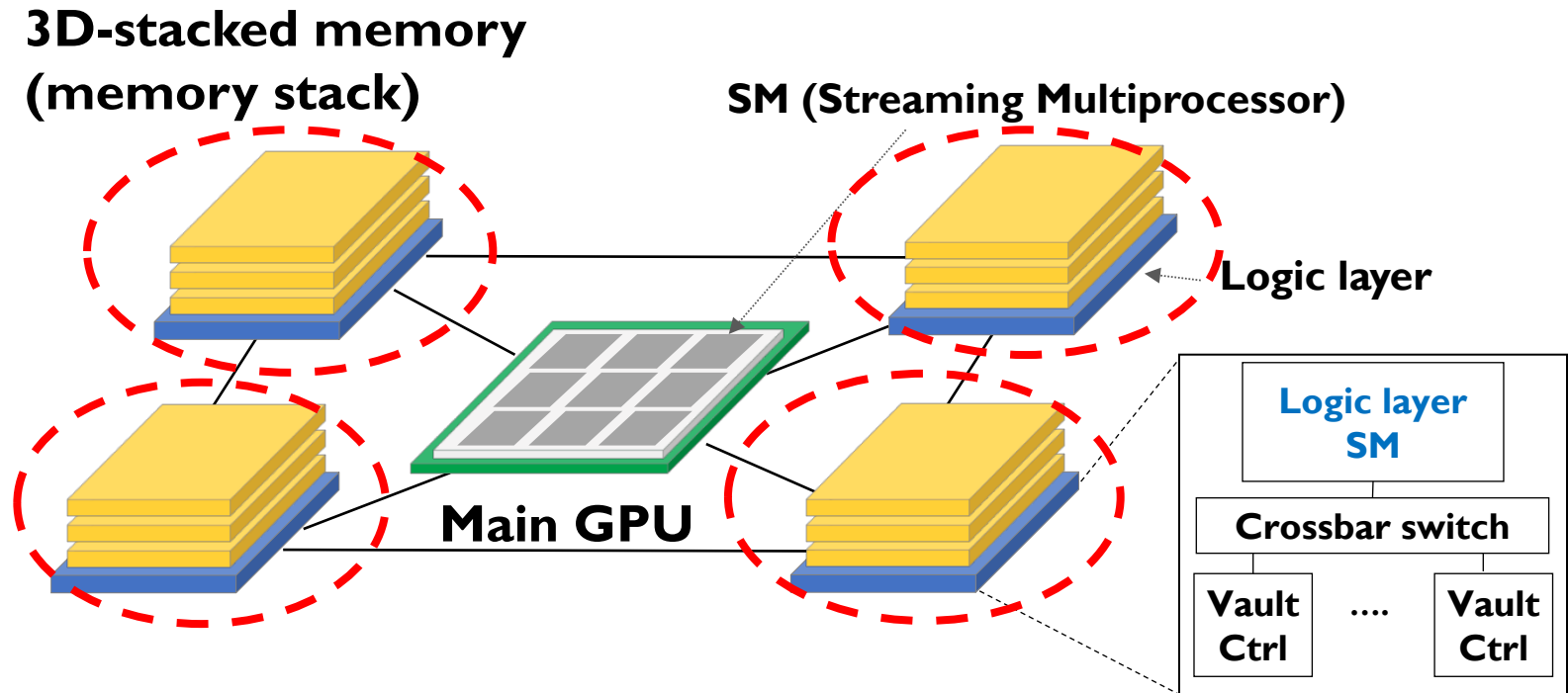
- **Challenge 1: Which operations should be executed in memory vs. in CPU?**





# 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?

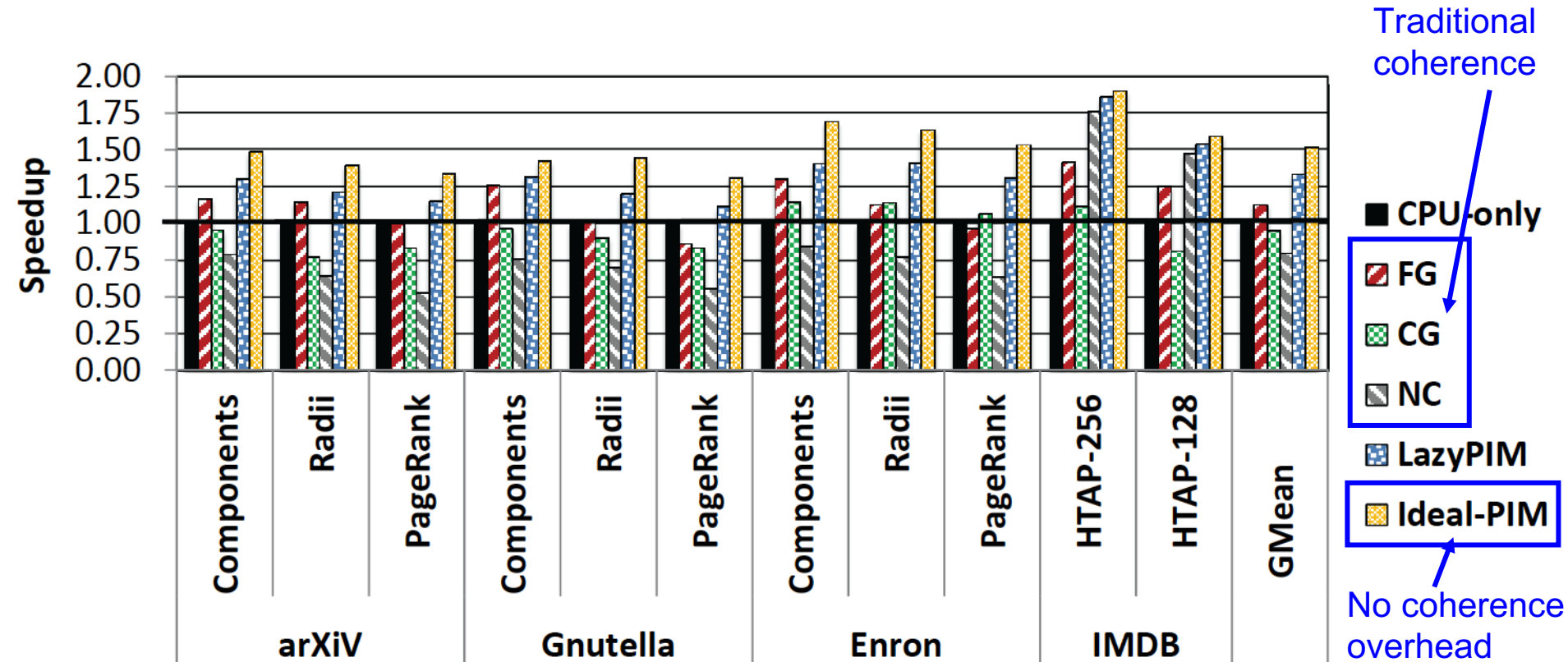
---

- 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 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](mailto:zhiyu.liu@brown.edu)

Maurice Herlihy

Computer Science Department  
Brown University  
[mph@cs.brown.edu](mailto:mph@cs.brown.edu)

Irina Calciu

VMware Research Group  
[icalciu@vmware.com](mailto:icalciu@vmware.com)

Onur Mutlu

Computer Science Department  
ETH Zürich  
[onur.mutlu@inf.ethz.ch](mailto:onur.mutlu@inf.ethz.ch)

# Simulation Infrastructures for PIM

---

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

## Ramulator: A Fast and Extensible DRAM Simulator

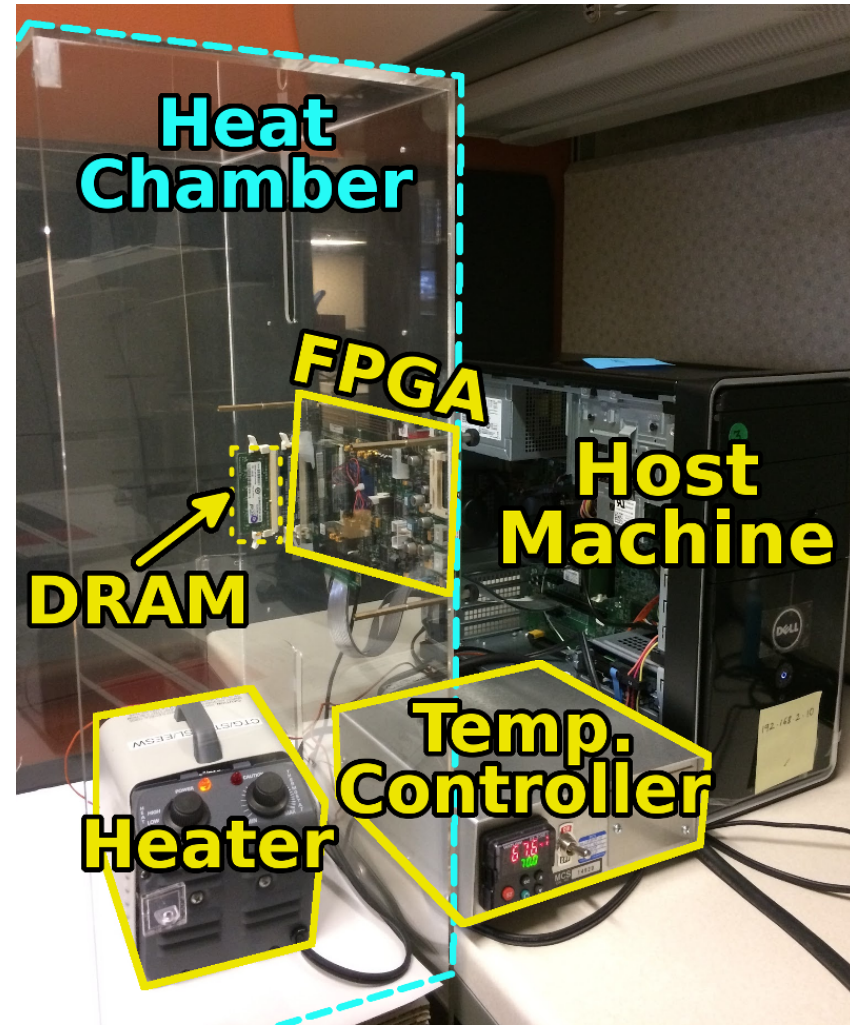
Yoongu Kim<sup>1</sup>    Weikun Yang<sup>1,2</sup>    Onur Mutlu<sup>1</sup>  
<sup>1</sup>Carnegie Mellon University    <sup>2</sup>Peking University



# An FPGA-based Test-bed for PIM?

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

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



# Some PIM Applications

# Goals

---

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

# Genome Read In-Memory (GRIM) Filter:

## Fast Location Filtering in DNA Read Mapping with Emerging Memory Technologies

**Jeremie Kim,**

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

**Carnegie Mellon**



**ETH** zürich

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

# GRIM-Filter in 3D-stacked DRAM

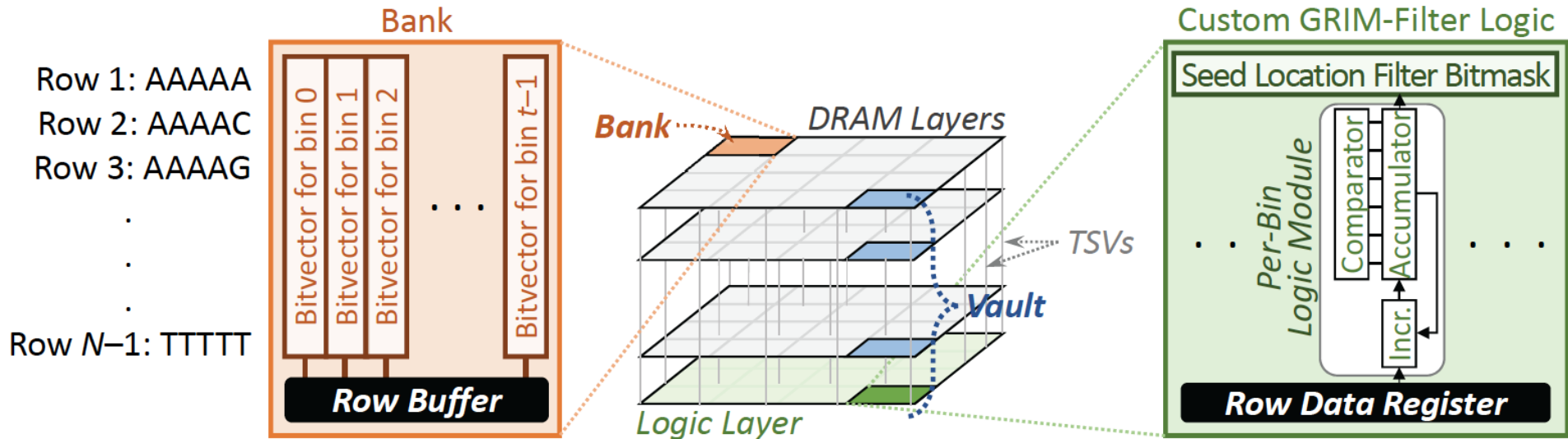
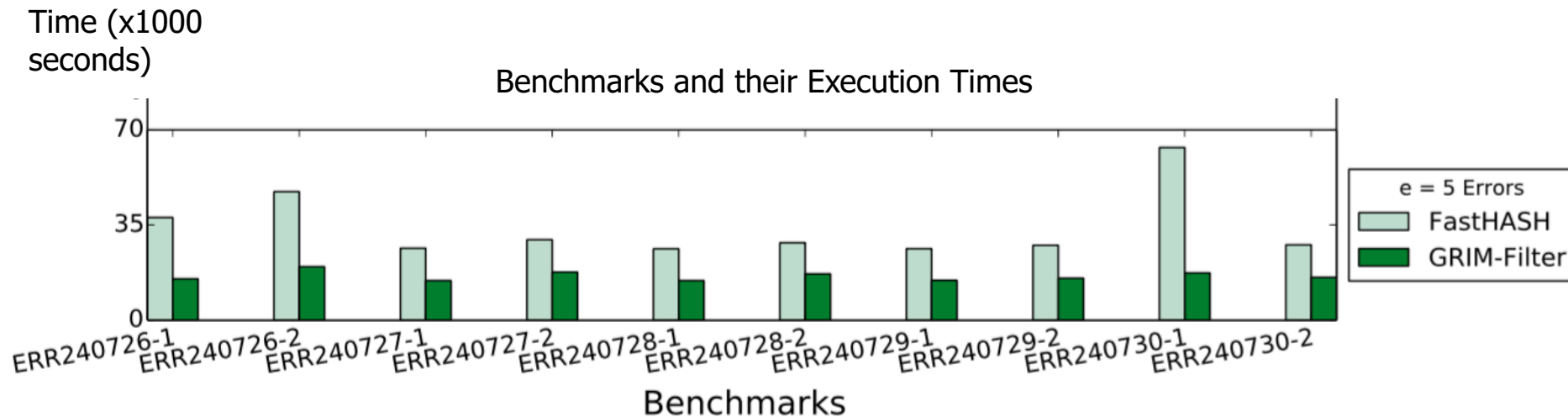


Figure 7: *Left block:* GRIM-Filter bitvector layout within a DRAM bank. *Center block:* 3D-stacked DRAM with tightly integrated logic layer stacked underneath with TSVs for a high intra-DRAM data transfer bandwidth. *Right block:* Custom GRIM-Filter logic placed in the logic layer.

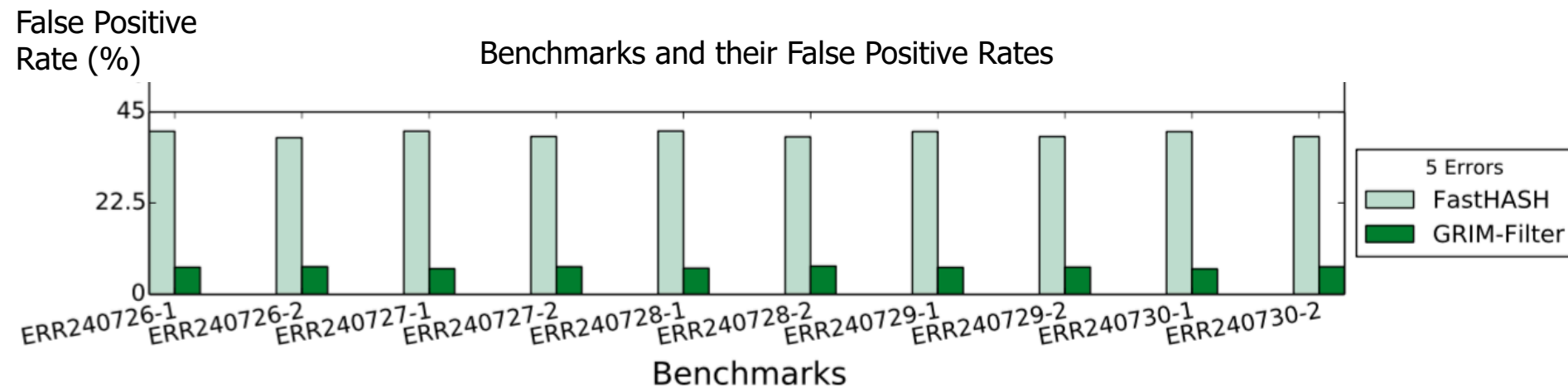
- The layout of bit vectors in a bank enables filtering many bins in parallel
- Customized logic for accumulation and comparison per genome segment
  - Low area overhead, simple implementation

# GRIM-Filter Performance



**1.8x-3.7x performance benefit across real data sets**

# GRIM-Filter False Positive Rate



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



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

# PIM-Based DNA Sequence Analysis

---

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

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

Jeremie S. Kim<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>

# Agenda

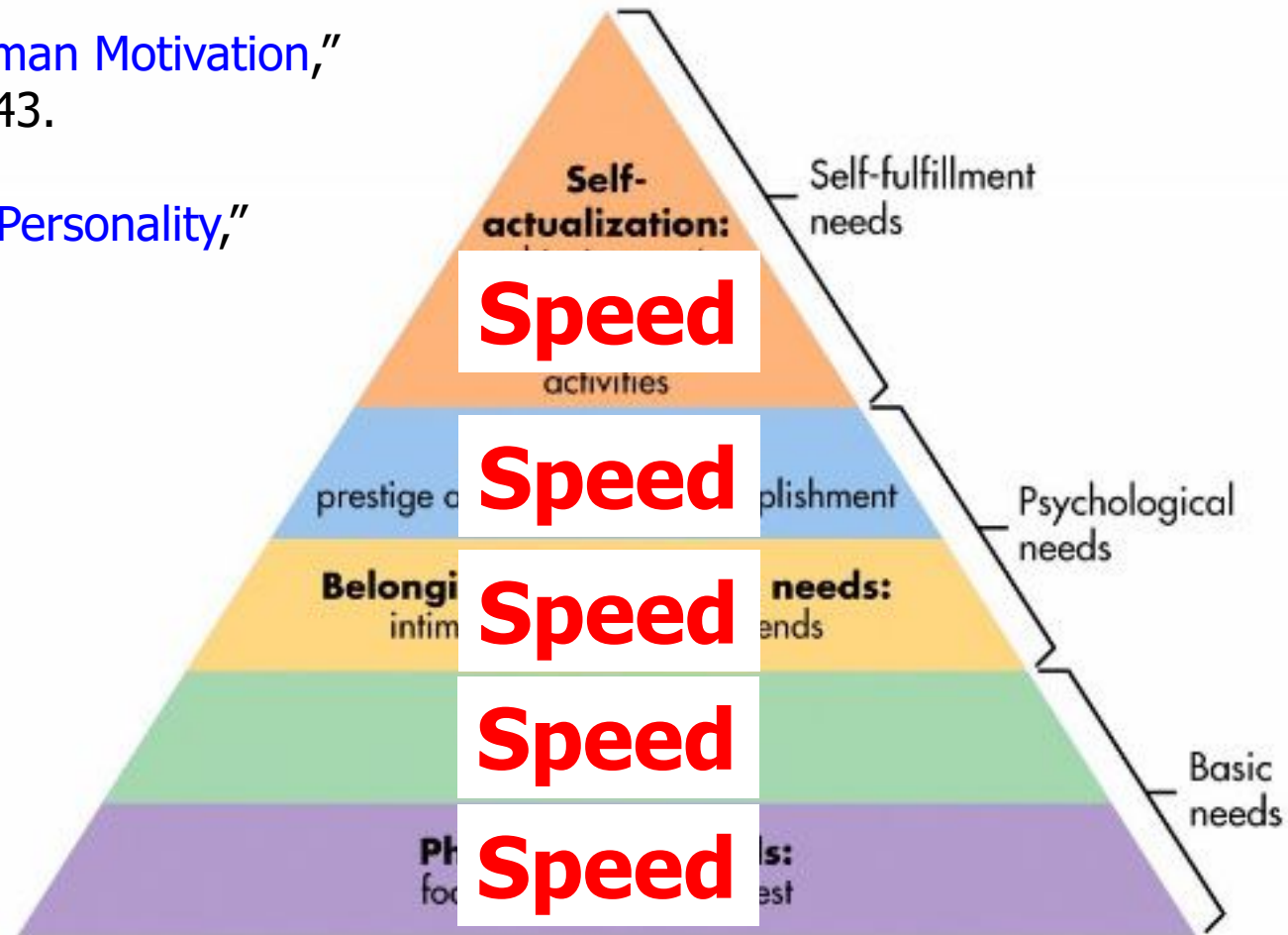
---

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

# Maslow's Hierarchy of Needs, A Third Time

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

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



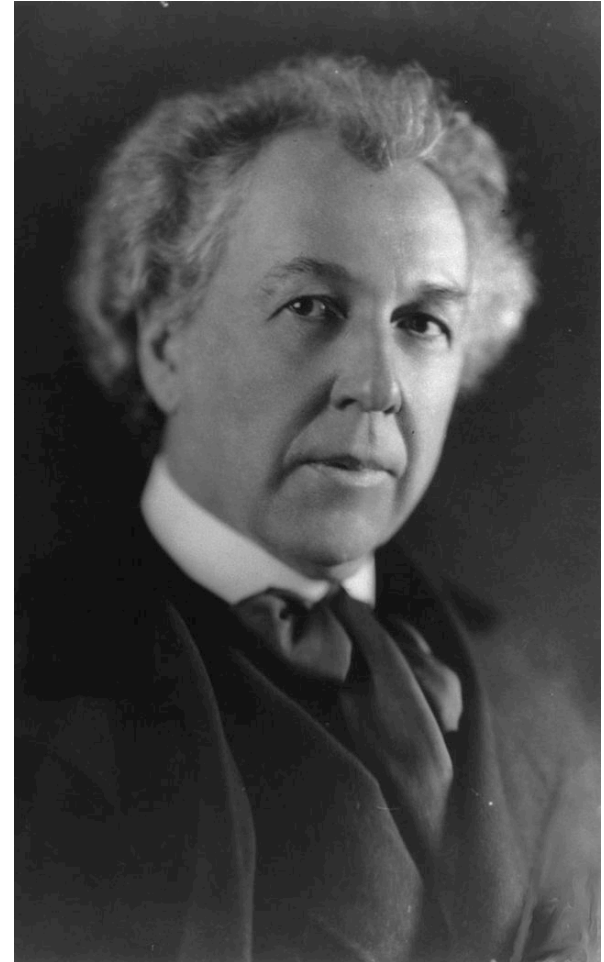
# Fundamentally Energy-Efficient (Data-Centric) Computing Architectures

# Fundamentally Low-Latency (Data-Centric) Computing Architectures

# A Quote from A Famous Architect

---

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





# Precedent-Based Design?

---

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





# Principled Design

---

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







# The Overarching Principle

---

## Organic architecture

---

From Wikipedia, the free encyclopedia

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

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



# Another Example: Precedent-Based Design

---





# Principled Design





# Another Principled Design



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

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



# Principle Applied to Another Structure



Source: By 準建築人手机网站 Forgemind ArchiMedia - Flickr: IMG\_2489.JPG, CC BY 2.0,

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

# The Overarching Principle

---

## Zoomorphic architecture

---

From Wikipedia, the free encyclopedia

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

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



# Overarching Principle for Computing?

---



# Concluding Remarks

---

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

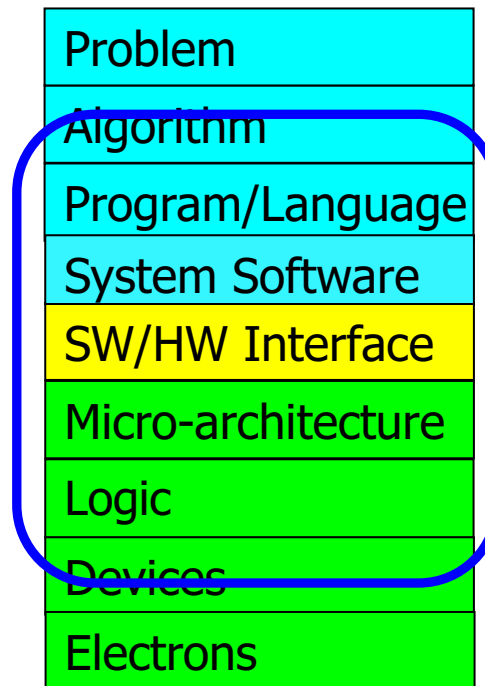
# The Future of Processing in Memory is Bright

---

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

Can enable:

- Orders of magnitude improvements
- New applications and computing systems



Yet, we have to

- Think across the stack
- Design enabling systems

# If In Doubt, See Other Doubtful Technologies

---

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



*Proceedings of the IEEE, Sept. 2017*

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

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

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

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

Onur Mutlu

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

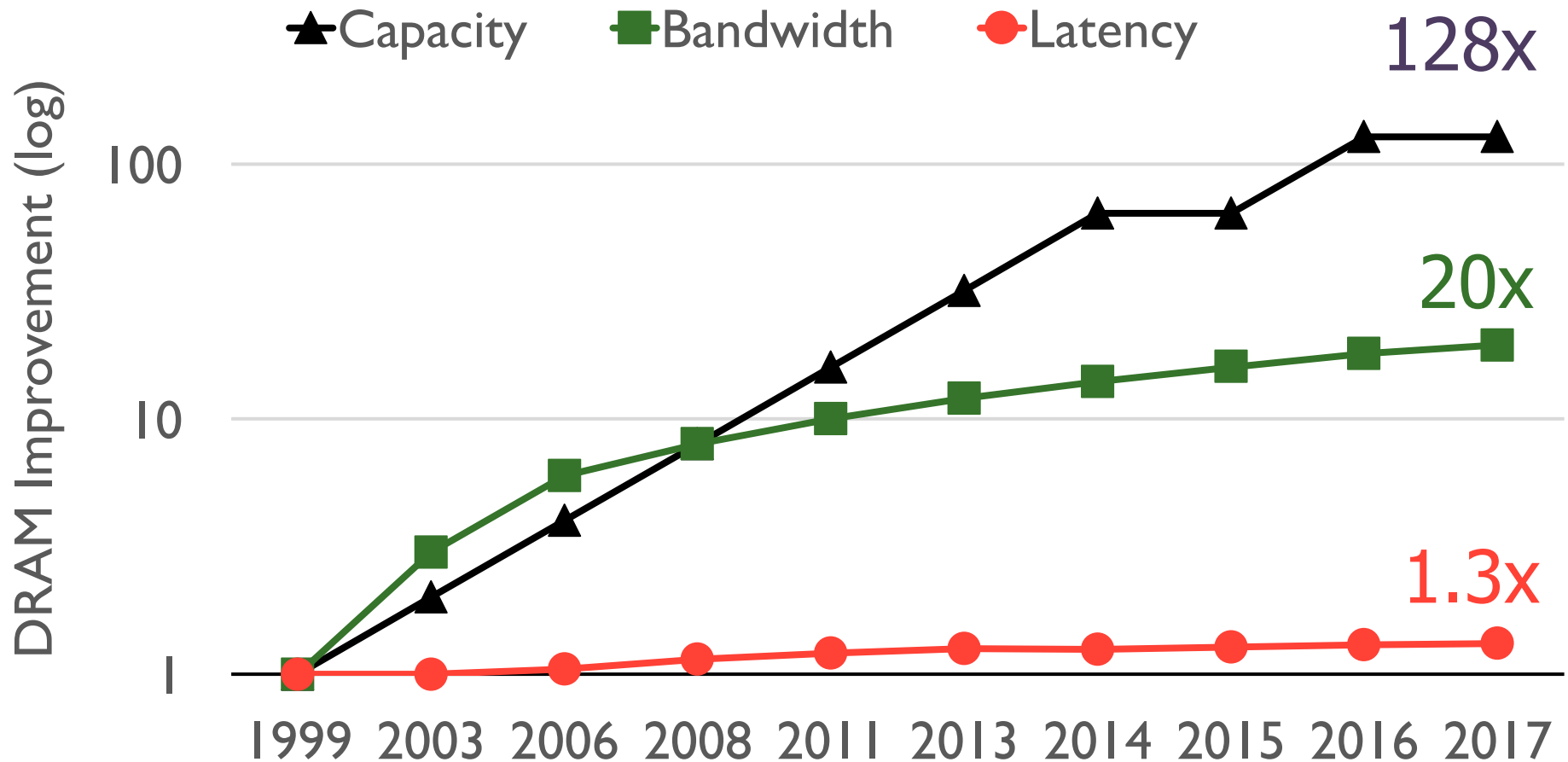
<https://people.inf.ethz.ch/omutlu>

December 1, 2017

Institute of Neuroinformatics Seminar

# Reducing Memory Latency

# Main Memory Latency Lags Behind



Memory latency remains almost constant



# A Closer Look ...

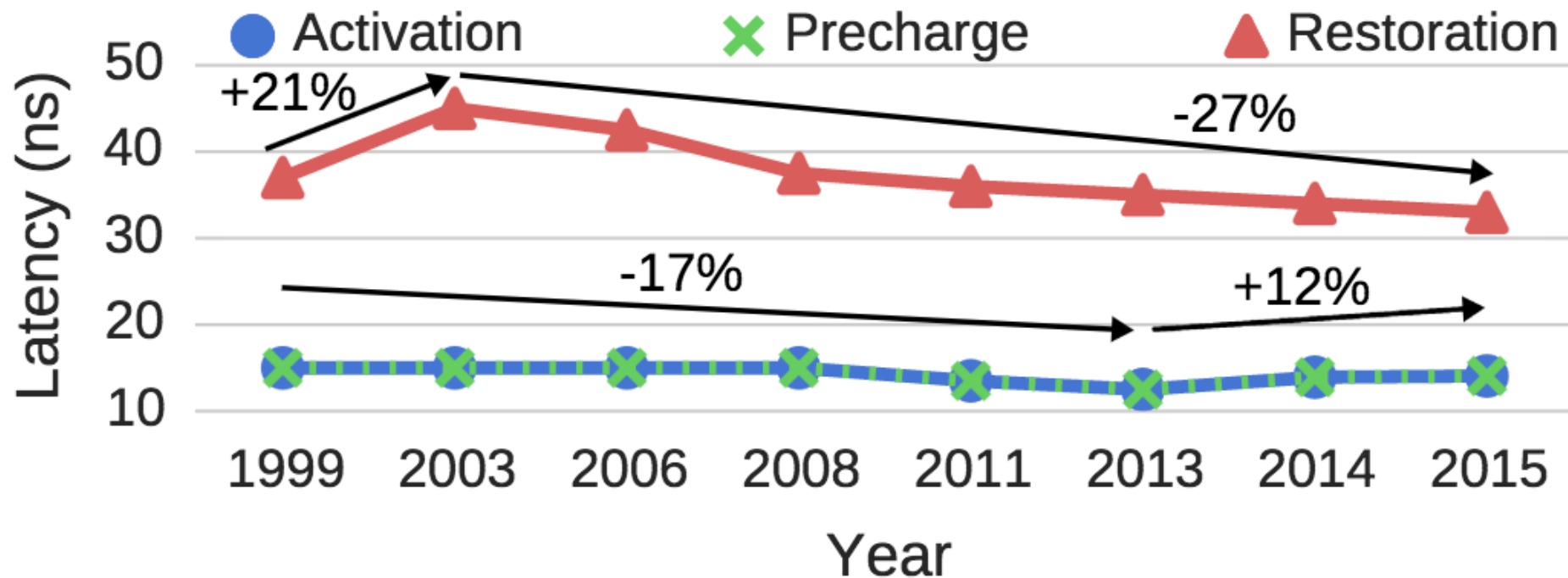


Figure 1: DRAM latency trends over time [20, 21, 23, 51].

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

# DRAM Latency Is Critical for Performance

---



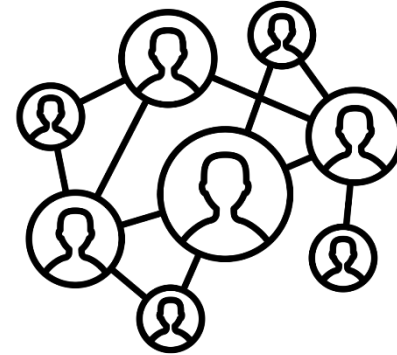
## In-memory Databases

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



## In-Memory Data Analytics

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



## Graph/Tree Processing

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



## Datacenter Workloads

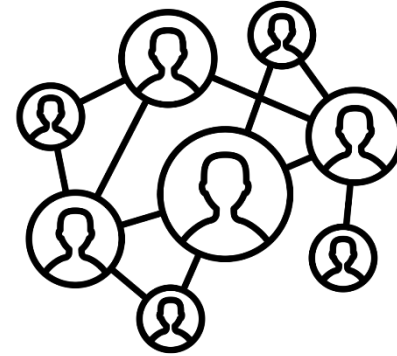
[Kanev+ (Google), ISCA'15]

# DRAM Latency Is Critical for Performance

---



**In-memory Databases**



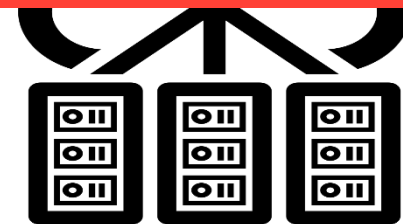
**Graph/Tree Processing**

Long memory latency → performance bottleneck



**In-Memory Data Analytics**

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



**Datacenter Workloads**

[Kanev+ (Google), ISCA'15]

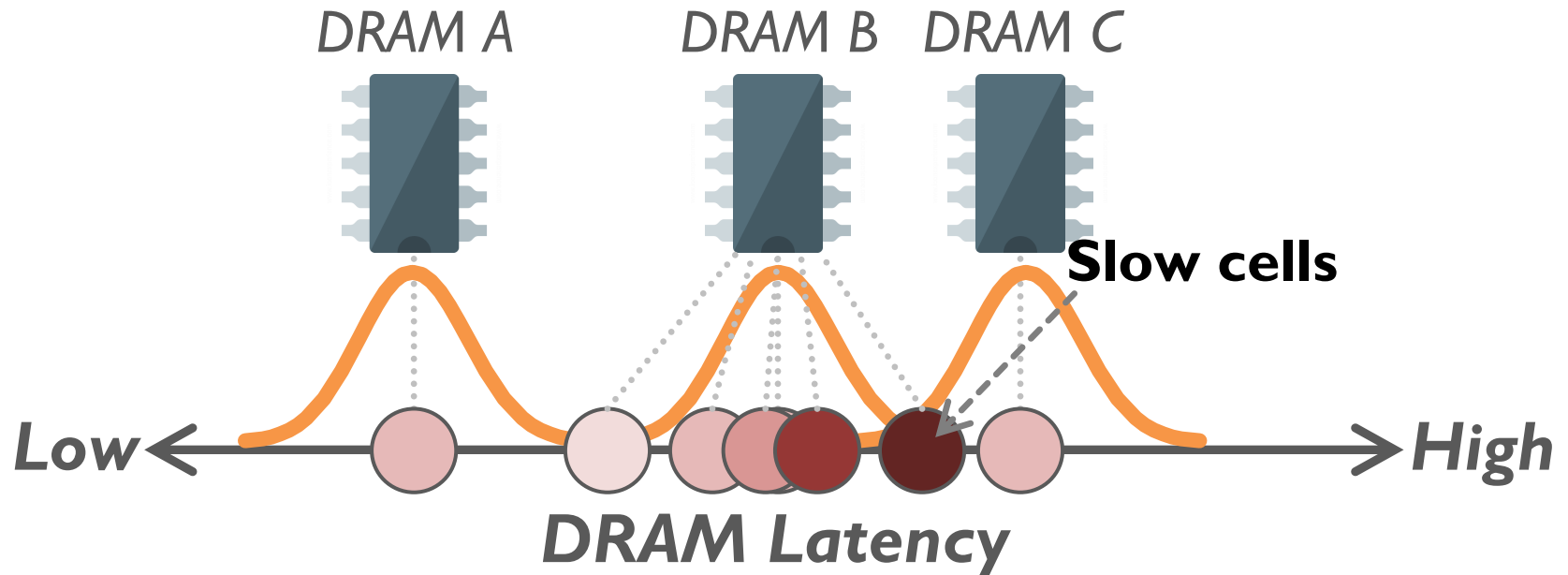
# Why the Long Latency?

---

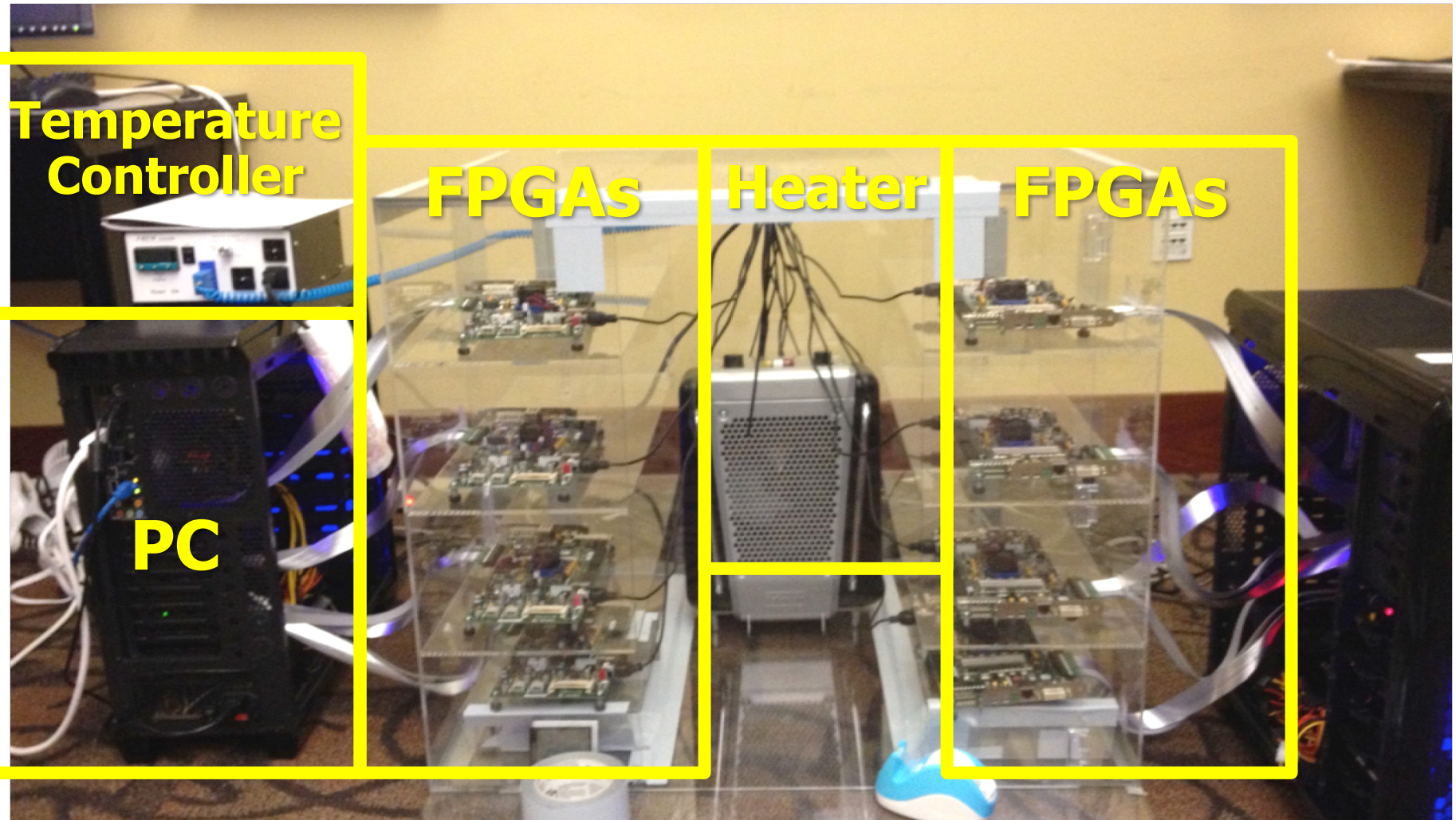
- Design of DRAM uArchitecture
  - Goal: Maximize capacity/area, not minimize latency
- “One size fits all” approach to latency specification
  - Same latency parameters for all temperatures
  - Same latency parameters for all DRAM chips (e.g., rows)
  - Same latency parameters for all parts of a DRAM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

# Latency Variation in Memory Chips

Heterogeneous manufacturing & operating conditions →  
latency variation in timing parameters



# DRAM Characterization Infrastructure

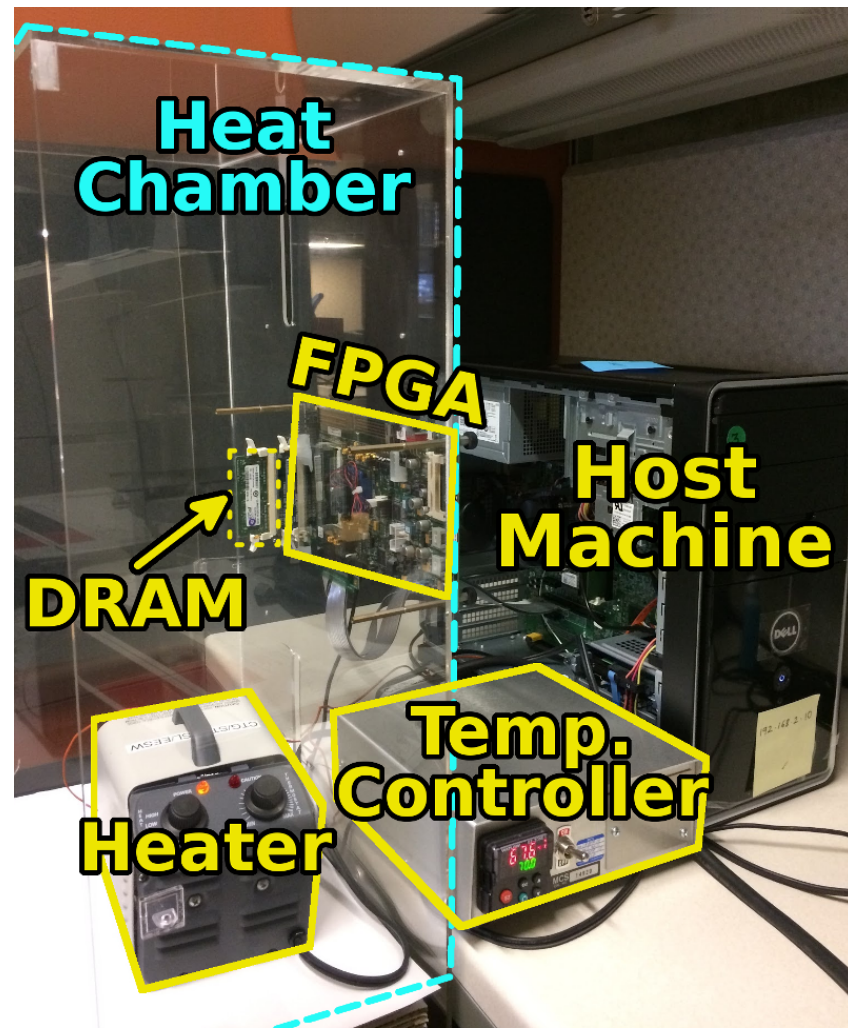




# DRAM Characterization Infrastructure

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

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





# SoftMC: Open Source DRAM Infrastructure

---

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

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

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

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

# Tackling the Fixed Latency Mindset

---

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

# Adaptive-Latency DRAM

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

# Latency Reduction Summary of 115 DIMMs

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

# AL-DRAM: Real System Evaluation

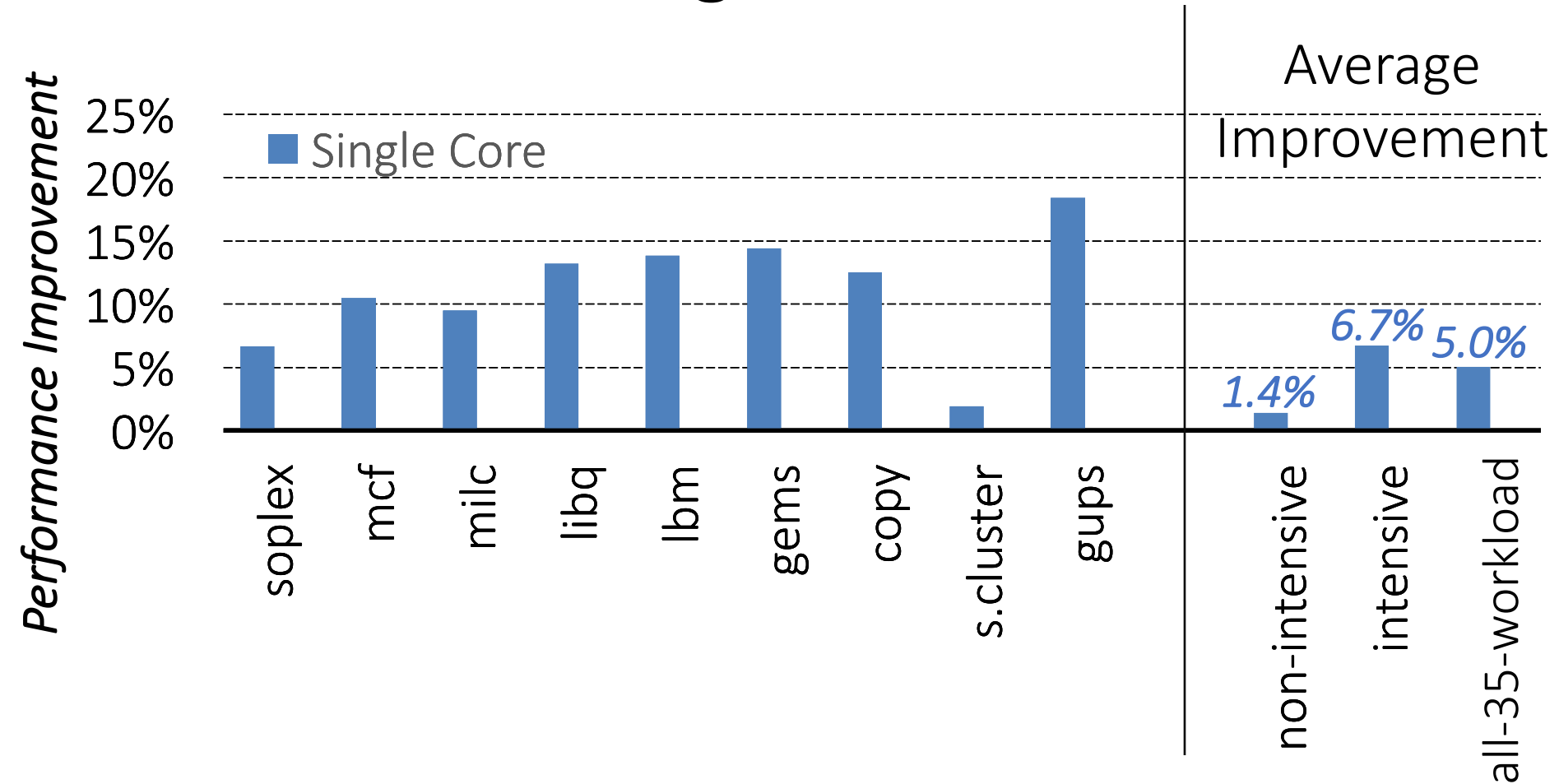
- *System*
  - *CPU: AMD 4386 ( 8 Cores, 3.1GHz, 8MB LLC)*

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

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

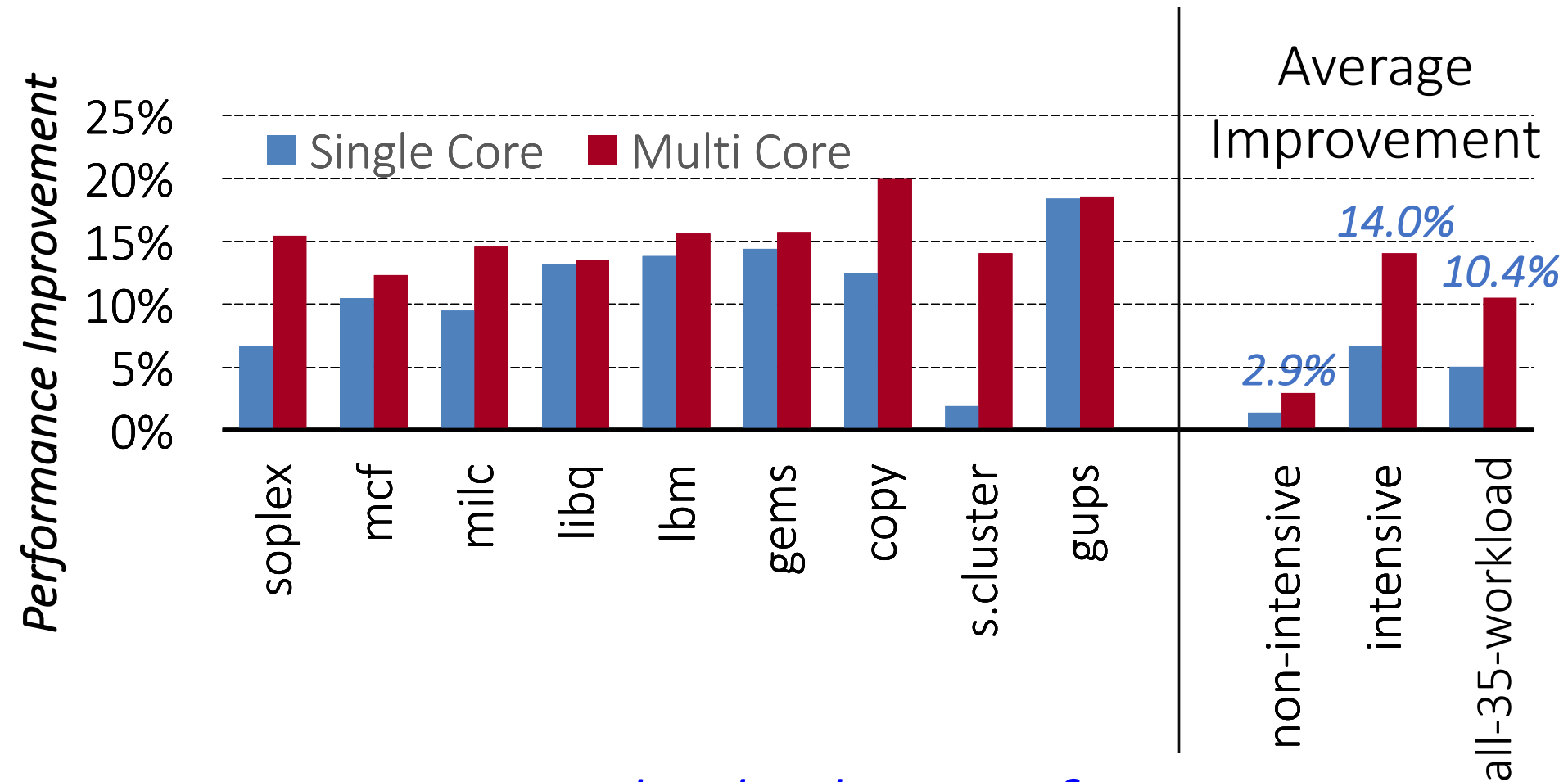
Bits	Description								
31:30	Reserved.								
29:24	<b>Tras: row active strobe.</b> Read-write. BIOS: See <a href="#">2.9.7.5 [SPD ROM-Based Configuration]</a> . Specifies the minimum time in memory clock cycles from an activate command to a precharge command, both to the same chip select bank. <table>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

# AL-DRAM: Single-Core Evaluation



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

# AL-DRAM: Multi-Core Evaluation



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



# Reducing Latency Also Reduces Energy

---

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

# More on Adaptive-Latency DRAM

---

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

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

*Proceedings of the 21st International Symposium on High-Performance Computer Architecture (**HPCA**), Bay Area, CA, February 2015.*

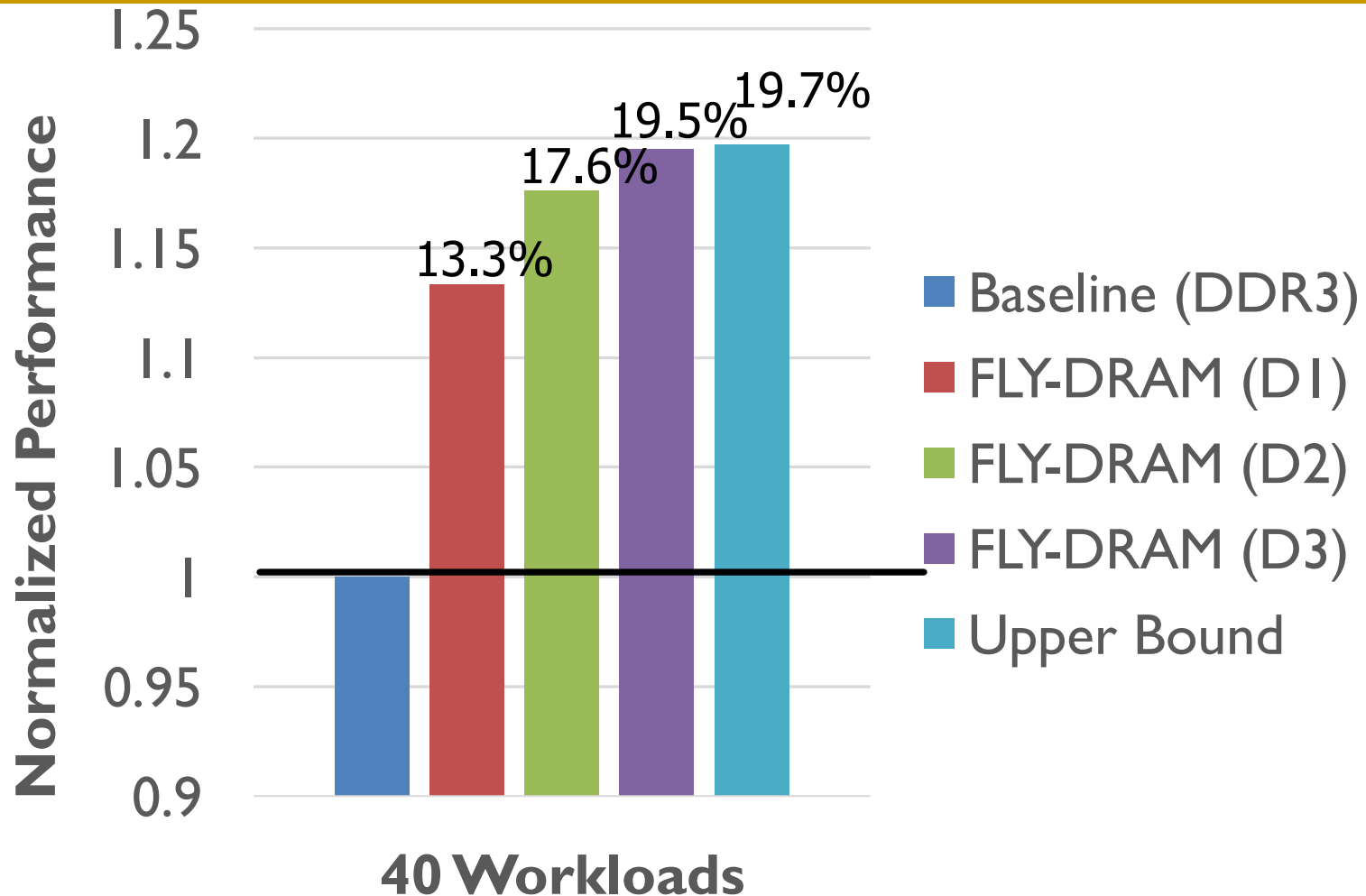
[[Slides \(pptx\)](#)] [[pdf](#)] [[Full data sets](#)]

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

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

Carnegie Mellon University

# Heterogeneous Latency within A Chip



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

# Analysis of Latency Variation in DRAM Chips

---

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

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

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

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

[[Source Code](#)]

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

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

Abhijith Kashyap<sup>1</sup>

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

Saugata Ghose<sup>1</sup>

Kevin Hsieh<sup>1</sup>

Donghyuk Lee<sup>1</sup>

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

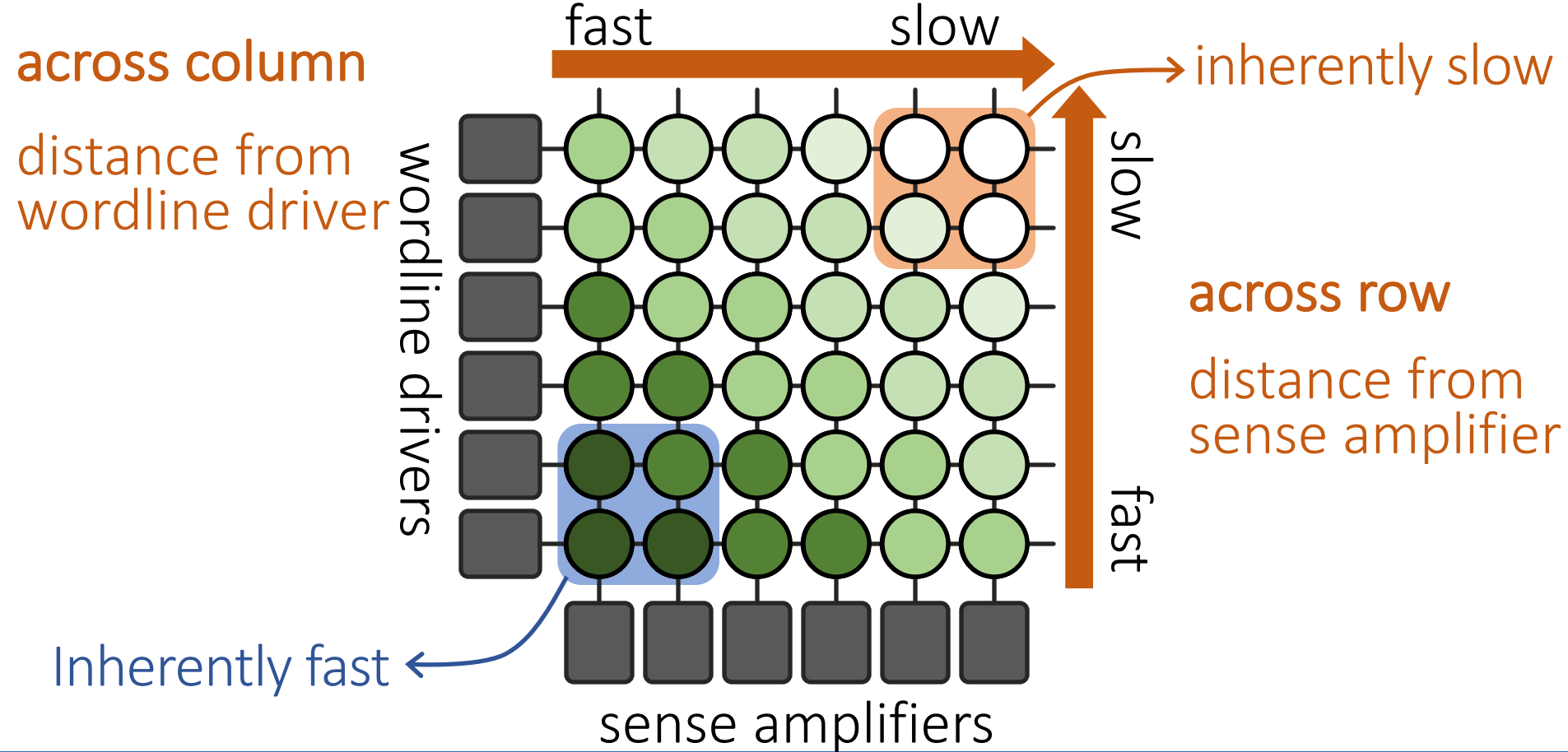
Gennady Pekhimenko<sup>1</sup>

Samira Khan<sup>4</sup>

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

<sup>1</sup>Carnegie Mellon University   <sup>2</sup>TOBB ETÜ   <sup>3</sup>Peking University   <sup>4</sup>University of Virginia   <sup>5</sup>ETH Zürich

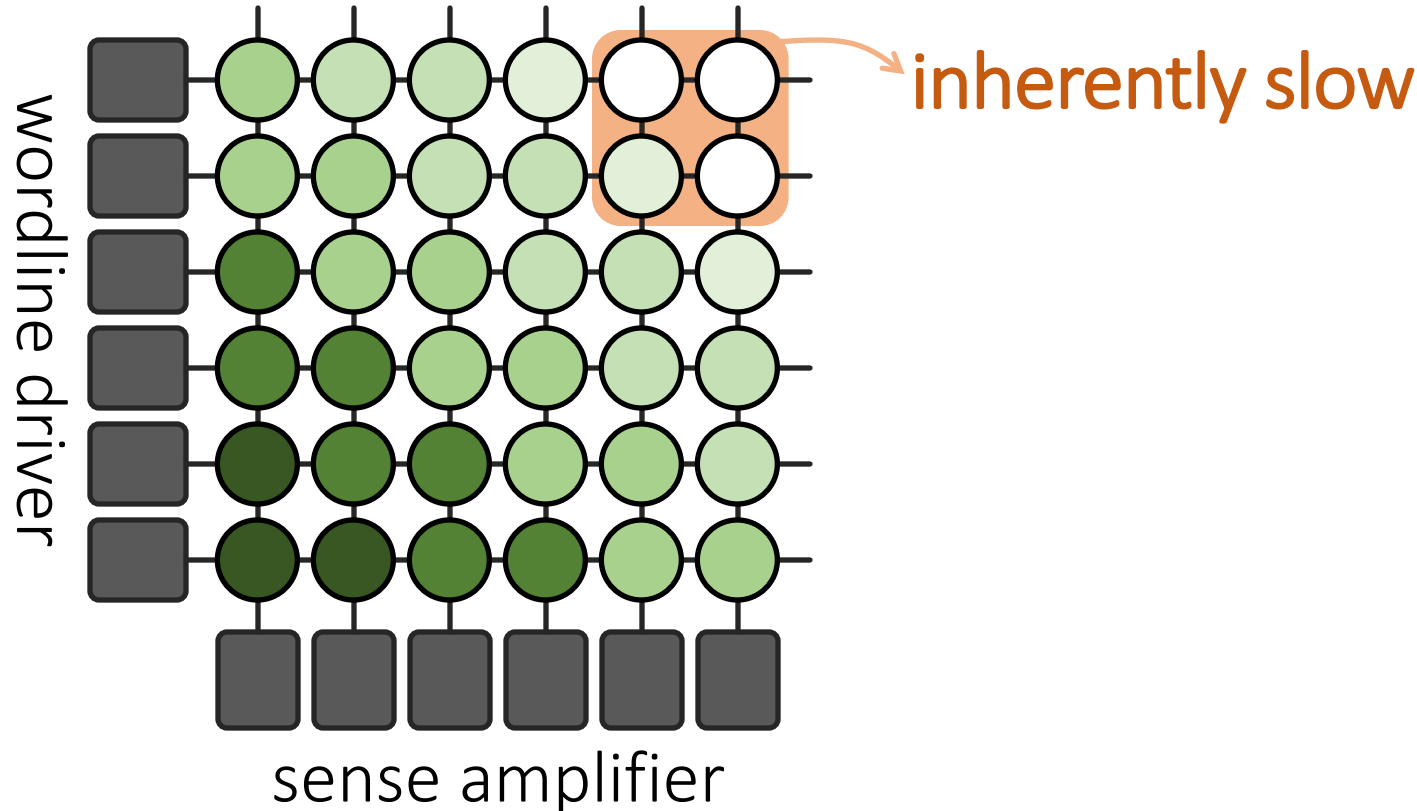
# What Is Design-Induced Variation?



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

# DIVA Online Profiling

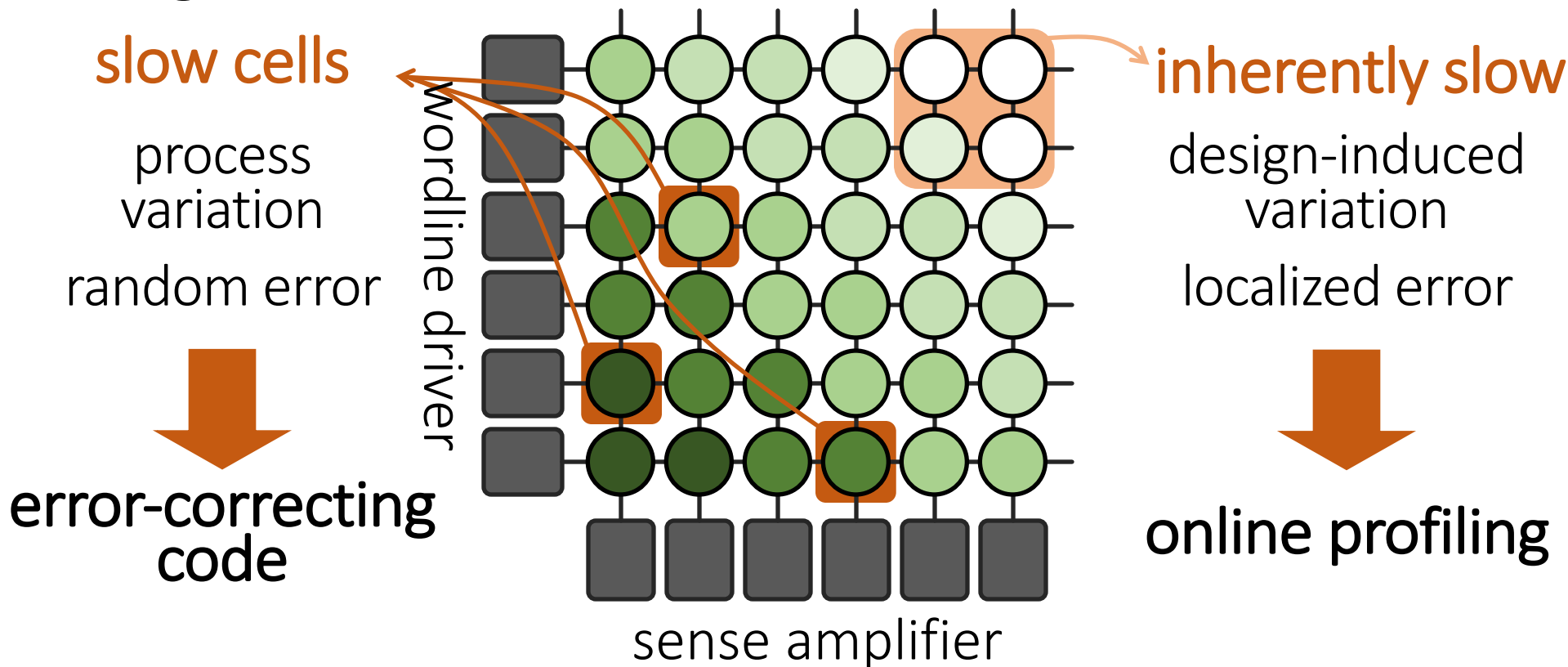
Design-Induced-Variation-Aware



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

# DIVA Online Profiling

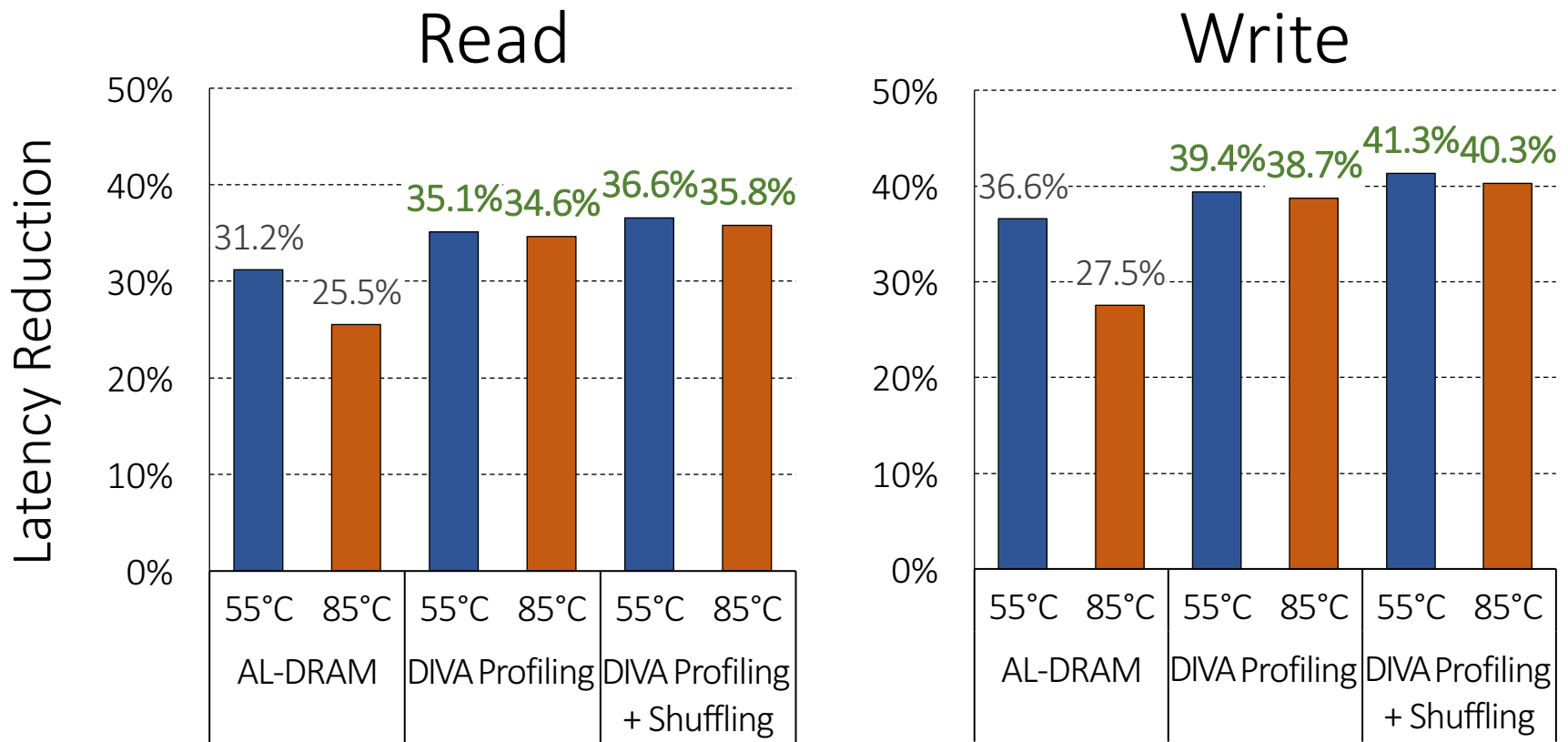
Design-Induced-Variation-Aware



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



# DIVA-DRAM Reduces Latency



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

# Design-Induced Latency Variation in DRAM

---

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

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

Donghyuk Lee, NVIDIA and Carnegie Mellon University

Samira Khan, University of Virginia

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

Gennady Pekhimenko, Vivek Seshadri, Microsoft Research

Onur Mutlu, ETH Zürich and Carnegie Mellon University

# Voltron: Exploiting the Voltage-Latency-Reliability Relationship

# Executive Summary

---

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

# Analysis of Latency-Voltage in DRAM Chips

---

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

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

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

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

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

<sup>†</sup>Carnegie Mellon University

<sup>¶</sup>NVIDIA

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

<sup>§</sup>ETH Zürich

# And, What If ...

---

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

# Fundamentally Low Latency Computing Architectures