

P&S Processing-in-Memory

Exploring the Processing-in-Memory Paradigm
for Future Computing Systems

Dr. Juan Gómez Luna

Prof. Onur Mutlu

ETH Zürich

Fall 2021

5 October 2021

P&S: Processing-in-Memory (I)

227-0085-37L Projects & Seminars: Exploring the Processing-in-Memory Paradigm for Future Computing Systems

Semester	Autumn Semester 2021
Lecturers	J. Gómez Luna
Periodicity	every semester recurring course
Language of instruction	English
Comment	Only for Electrical Engineering and Information Technology BSc. The course unit can only be taken once. Repeated enrollment in a later semester is not creditable.

Courses	Catalogue data	Performance assessment	Learning materials	Groups	Restrictions	Offered in	► Overview
Abstract	The category of "Laboratory Courses, Projects, Seminars" includes courses and laboratories in various formats designed to impart practical knowledge and skills. Moreover, these classes encourage independent experimentation and design, allow for explorative learning and teach the methodology of project work.						
Objective	<p>Data movement between the memory units and the compute units of current computing systems is a major performance and energy bottleneck. From large-scale servers to mobile devices, data movement costs dominate computation costs in terms of both performance and energy consumption. For example, data movement between the main memory and the processing cores accounts for 62% of the total system energy in consumer applications. As a result, the data movement bottleneck is a huge burden that greatly limits the energy efficiency and performance of modern computing systems. This phenomenon is an undesired effect of the dichotomy between memory and the processor, which leads to the data movement bottleneck.</p> <p>Many modern and important workloads such as machine learning, computational biology, graph processing, databases, video analytics, and real-time data analytics suffer greatly from the data movement bottleneck. These workloads are exemplified by irregular memory accesses, relatively low data reuse, low cache line utilization, low arithmetic intensity (i.e., ratio of operations per accessed byte), and large datasets that greatly exceed the main memory size. The computation in these workloads cannot usually compensate for the data movement costs. In order to alleviate this data movement bottleneck, we need a paradigm shift from the traditional processor-centric design, where all computation takes place in the compute units, to a more data centric design where processing elements are placed closer to or inside where the data resides. This paradigm of computing is known as Processing-in Memory (PIM).</p> <p>This is your perfect P&S if you want to become familiar with the main PIM technologies, which represent "the next big thing" in Computer Architecture. You will work hands-on with the first real-world PIM architecture, will explore different PIM architecture designs for important workloads, and will develop tools to enable research of future PIM systems. Projects in this course span software and hardware as well as the software/hardware interface. You can potentially work on developing and optimizing new workloads for the first real world PIM hardware or explore new PIM designs in simulators, or do something else that can forward our understanding of the PIM paradigm.</p>						

P&S: Processing-in-Memory (II)

Data movement between the memory units and the compute units of current computing systems is a major performance and energy bottleneck. From large-scale servers to mobile devices, data movement costs dominate computation costs in terms of both performance and energy consumption. For example, data movement between the main memory and the processing cores accounts for 62% of the total system energy in consumer applications. As a result, the data movement bottleneck is a huge burden that greatly limits the energy efficiency and performance of modern computing systems. This phenomenon is an undesired effect of the dichotomy between memory and the processor, which leads to the data movement bottleneck.

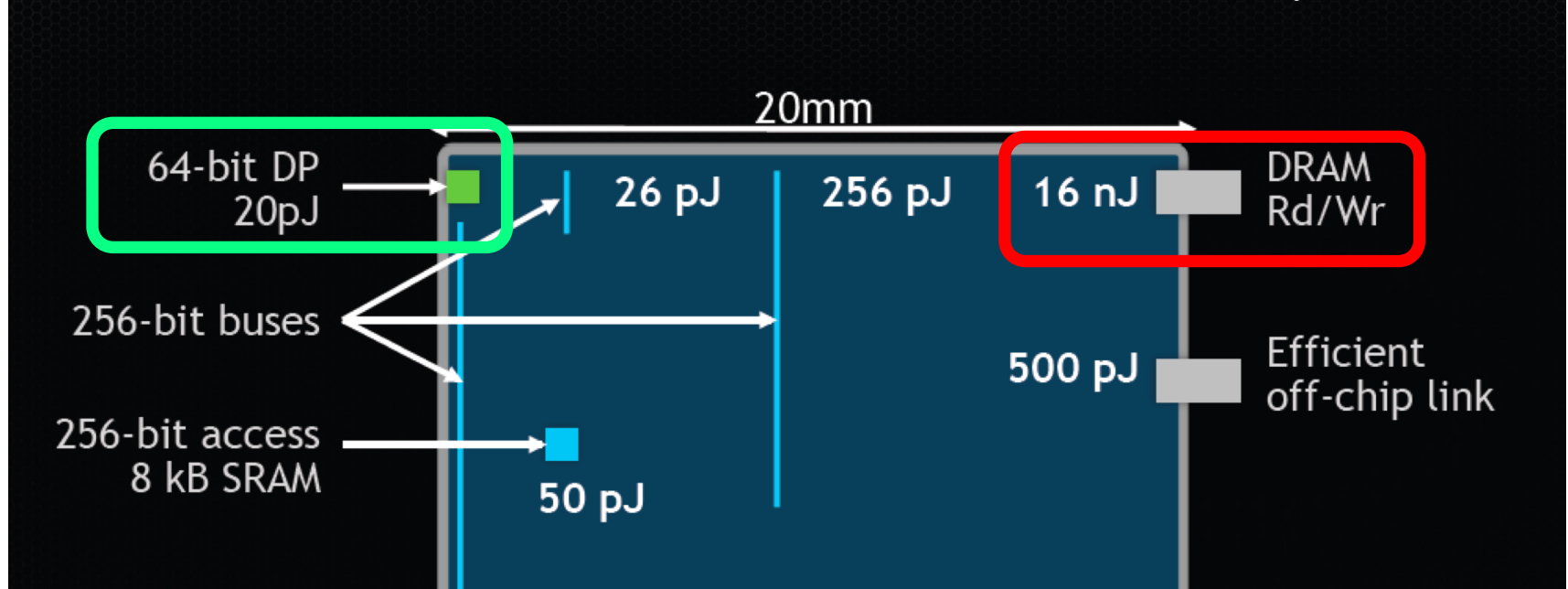
Many modern and important workloads such as machine learning, computational biology, graph processing, databases, video analytics, and real-time data analytics suffer greatly from the data movement bottleneck. These workloads are exemplified by irregular memory accesses, relatively low data reuse, low cache line utilization, low arithmetic intensity (i.e., ratio of operations per accessed byte), and large datasets that greatly exceed the main memory size. The computation in these workloads cannot usually compensate for the data movement costs. In order to alleviate this data movement bottleneck, we need a paradigm shift from the traditional processor-centric design, where all computation takes place in the compute units, to a more data centric design where processing elements are placed closer to or inside where the data resides. This paradigm of computing is known as Processing-in Memory (PIM).

This is your perfect P&S if you want to become familiar with the main PIM technologies, which represent "the next big thing" in Computer Architecture. You will work hands-on with the first real-world PIM architecture, will explore different PIM architecture designs for important workloads, and will develop tools to enable research of future PIM systems. Projects in this course span software and hardware as well as the software/hardware interface. You can potentially work on developing and optimizing new workloads for the first real world PIM hardware or explore new PIM designs in simulators, or do something else that can forward our understanding of the PIM paradigm.

Data Movement vs. Computation Energy

Communication Dominates Arithmetic

Dally, HiPEAC 2015



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

Goals of this P&S Course

P&S Processing-in-Memory: Contents

- We will introduce the **data movement bottleneck**, which is a major threat to high performance and energy efficiency of current computing systems
- You will learn what are **key workload characteristics** that make them more prone to the data movement bottleneck
- You will review traditional approaches to alleviating data movement and will **get familiar with new research proposals**: processing-in-memory solutions
- You will **work hands-on**: analyzing workloads, programming PIM architectures, simulating new PIM proposals, etc.

A +50-Year-Old Paradigm

■ Kautz, "Cellular Logic-in-Memory Arrays", IEEE TC 1969

IEEE TRANSACTIONS ON COMPUTERS, VOL. C-18, NO. 8, AUGUST 1969

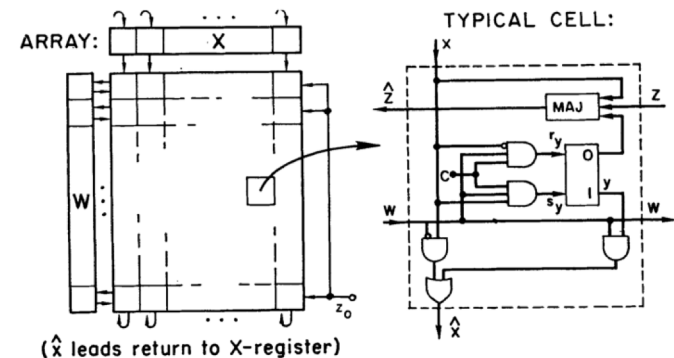
Cellular Logic-in-Memory Arrays

WILLIAM H. KAUTZ, MEMBER, IEEE

Abstract—As a direct consequence of large-scale integration, many advantages in the design, fabrication, testing, and use of digital circuitry can be achieved if the circuits can be arranged in a two-dimensional iterative, or cellular, array of identical elementary networks, or cells. When a small amount of storage is included in each cell, the same array may be regarded either as a logically enhanced memory array, or as a logic array whose elementary gates and connections can be "programmed" to realize a desired logical behavior.

In this paper the specific engineering features of such cellular logic-in-memory (CLIM) arrays are discussed, and one such special-purpose array, a cellular sorting array, is described in detail to illustrate how these features may be achieved in a particular design. It is shown how the cellular sorting array can be employed as a single-address, multiword memory that keeps in order all words stored within it. It can also be used as a content-addressed memory, a pushdown memory, a buffer memory, and (with a lower logical efficiency) a programmable array for the realization of arbitrary switching functions. A second version of a sorting array, operating on a different sorting principle, is also described.

Index Terms—Cellular logic, large-scale integration, logic arrays logic in memory, push-down memory, sorting, switching functions.



$$\begin{aligned}\hat{x} &= \bar{w}x + wy \\ s_y &= wcx, r_y = wc\bar{x} \\ \hat{z} &= M(x, \bar{y}, z) = x\bar{y} + z(x + \bar{y})\end{aligned}$$

Fig. 1. Cellular sorting array I.

Processing in/near Memory: An Old Idea

- Stone, “A Logic-in-Memory Computer,” IEEE TC 1970

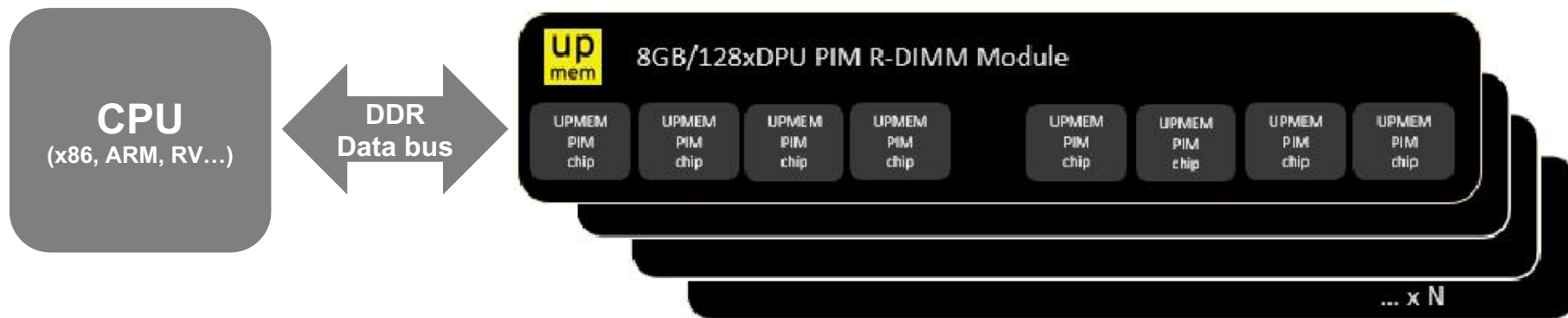
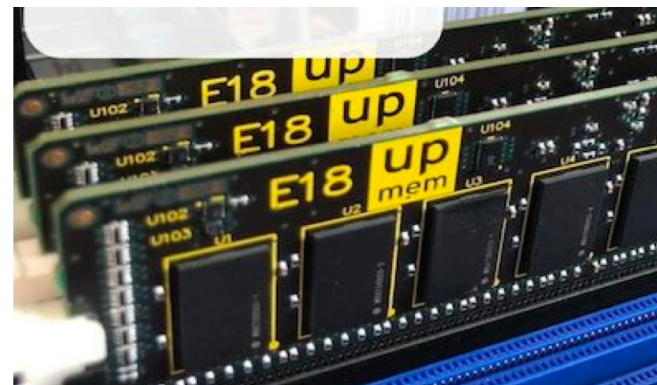
A Logic-in-Memory Computer

HAROLD S. STONE

Abstract—If, as presently projected, the cost of microelectronic arrays in the future will tend to reflect the number of pins on the array rather than the number of gates, the logic-in-memory array is an extremely attractive computer component. Such an array is essentially a microelectronic memory with some combinational logic associated with each storage element.

UPMEM Processing-in-DRAM Engine (2019)

- **Processing in DRAM Engine**
- Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.
- Replaces **standard DIMMs**
 - DDR4 R-DIMM modules
 - 8GB+128 DPUs (16 PIM chips)
 - Standard 2x-nm DRAM process
 - **Large amounts of** compute & memory bandwidth



Experimental Analysis of the UPMEM PIM Engine

Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

IZZAT EL HAJJ, American University of Beirut, Lebanon

IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain

CHRISTINA GIANNOULA, ETH Zürich, Switzerland and NTUA, Greece

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland

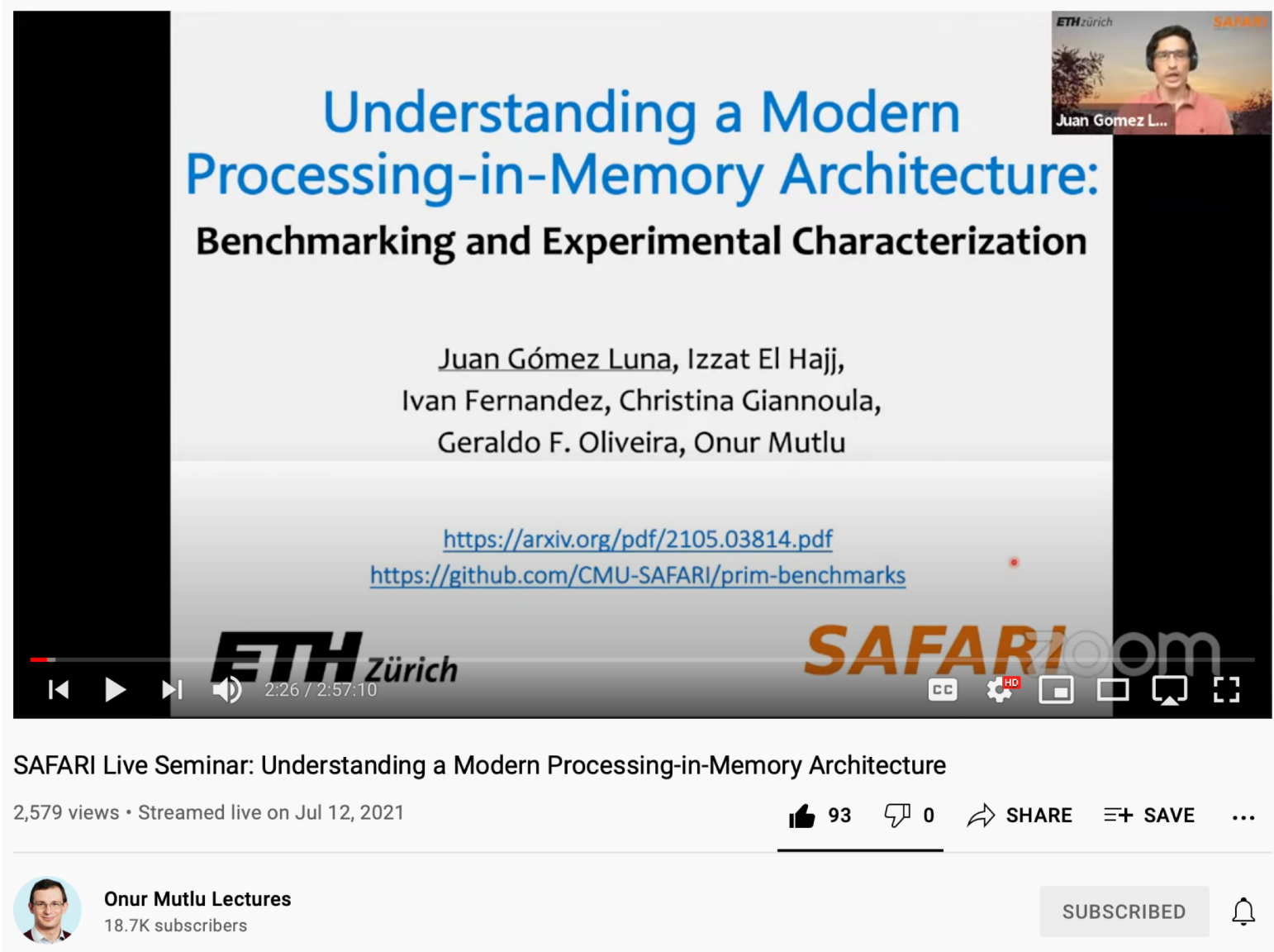
ONUR MUTLU, ETH Zürich, Switzerland

Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens through a narrow bus with high latency and limited bandwidth, and the low data reuse in memory-bound workloads is insufficient to amortize the cost of main memory access. Fundamentally addressing this *data movement bottleneck* requires a paradigm where the memory system assumes an active role in computing by integrating processing capabilities. This paradigm is known as *processing-in-memory* (PIM).

Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware prototypes. In contrast, the UPMEM company has designed and manufactured the first publicly-available real-world PIM architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called *DRAM Processing Units* (DPUs), integrated in the same chip.

This paper provides the first comprehensive analysis of the first publicly-available real-world PIM architecture. We make two key contributions. First, we conduct an experimental characterization of the UPMEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present *PrIM* (*Processing-In-Memory benchmarks*), a benchmark suite of 16 workloads from different application domains (e.g., dense/sparse linear algebra, databases, data analytics, graph processing, neural networks, bioinformatics, image processing), which we identify as memory-bound. We evaluate the performance and scaling characteristics of PrIM benchmarks on the UPMEM PIM architecture, and compare their performance and energy consumption to their state-of-the-art CPU and GPU counterparts. Our extensive evaluation conducted on two real UPMEM-based PIM systems with 640 and 2,556 DPUs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.

Understanding a Modern PIM Architecture



The video player shows a lecture titled "Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization". The speaker is Juan Gómez Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, and Geraldo F. Oliveira, Onur Mutlu. The video is from the SAFARI Live Seminar series, hosted by ETH Zürich. The video player includes a progress bar at 2:26 / 2:57:10, a volume icon, and a full screen button. The video player also shows the video title, view count (2,579 views), and the channel name (Onur Mutlu Lectures, 18.7K subscribers). The video player includes a progress bar at 2:26 / 2:57:10, a volume icon, and a full screen button. The video player also shows the video title, view count (2,579 views), and the channel name (Onur Mutlu Lectures, 18.7K subscribers).

Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization

Juan Gómez Luna, Izzat El Hajj,
Ivan Fernandez, Christina Giannoula,
Geraldo F. Oliveira, Onur Mutlu

<https://arxiv.org/pdf/2105.03814.pdf>
<https://github.com/CMU-SAFARI/prim-benchmarks>

ETH Zürich SAFARI

2:26 / 2:57:10

SAFARI Live Seminar: Understanding a Modern Processing-in-Memory Architecture

2,579 views • Streamed live on Jul 12, 2021

93 0 SHARE SAVE ...

Onur Mutlu Lectures
18.7K subscribers

SUBSCRIBED

Samsung Function-in-Memory DRAM (2021)



Samsung Develops Industry's First High Bandwidth Memory with AI Processing Power

Korea on February 17, 2021

Audio



Share



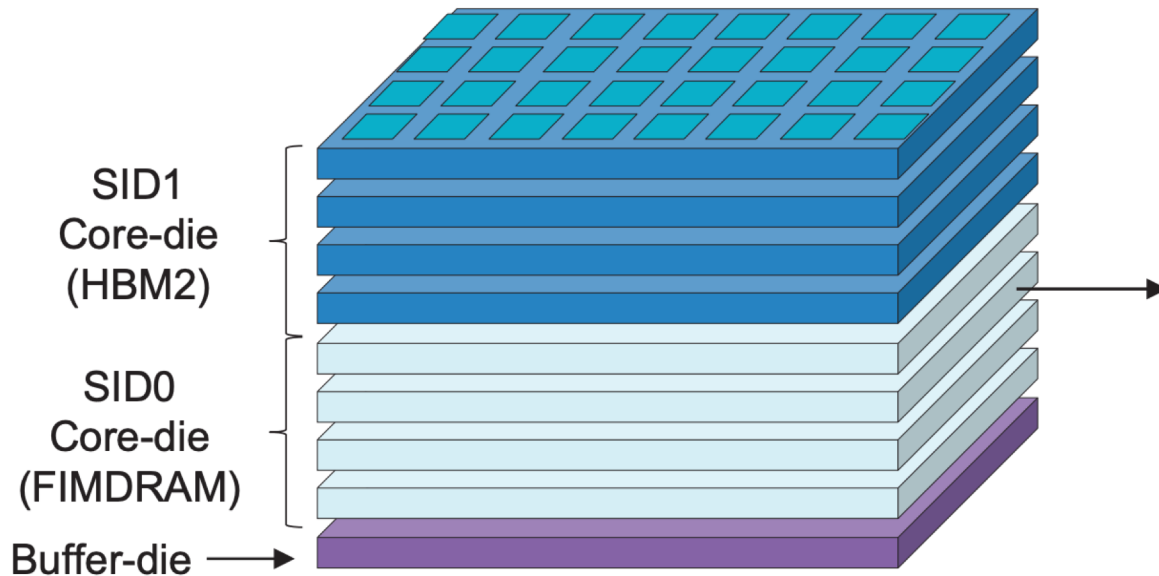
The new architecture will deliver over twice the system performance and reduce energy consumption by more than 70%

Samsung Electronics, the world leader in advanced memory technology, today announced that it has developed the industry's first High Bandwidth Memory (HBM) integrated with artificial intelligence (AI) processing power – the HBM-PIM. **The new processing-in-memory (PIM) architecture brings powerful AI computing capabilities inside high-performance memory, to accelerate large-scale processing in data centers, high performance computing (HPC) systems and AI-enabled mobile applications.**

Kwangil Park, senior vice president of Memory Product Planning at Samsung Electronics stated, "Our groundbreaking HBM-PIM is the industry's first programmable PIM solution tailored for diverse AI-driven workloads such as HPC, training and inference. We plan to build upon this breakthrough by further collaborating with AI solution providers for even more advanced PIM-powered applications."

Samsung Function-in-Memory DRAM (2021)

■ FIMDRAM based on HBM2



[3D Chip Structure of HBM with FIMDRAM]

Chip Specification

128DQ / 8CH / 16 banks / BL4

32 PCU blocks (1 FIM block/2 banks)

1.2 TFLOPS (4H)

**FP16 ADD /
Multiply (MUL) /
Multiply-Accumulate (MAC) /
Multiply-and- Add (MAD)**

ISSCC 2021 / SESSION 25 / DRAM / 25.4

25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

Young-Cheon Kwon¹, Suk Han Lee¹, Jaehoon Lee¹, Sang-Hyuk Kwon¹, Je Min Ryu¹, Jong-Pil Son¹, Seongil O¹, Hak-Soo Yu¹, Haesuk Lee¹, Soo Young Kim¹, Youngmin Cho¹, Jin Guk Kim¹, Jongyoon Choi¹, Hyun-Sung Shin¹, Jin Kim¹, BengSeng Phuah¹, HyoungMin Kim¹, Myeong Jun Song¹, Ahn Choi¹, Daeho Kim¹, SooYoung Kim¹, Eun-Bong Kim¹, David Wang², Shinhaeng Kang¹, Yuhwan Ro³, Seungwoo Seo³, JoonHo Song³, Jaeyoun Youn¹, Kyomin Sohn¹, Nam Sung Kim¹

¹Samsung Electronics, Hwaseong, Korea

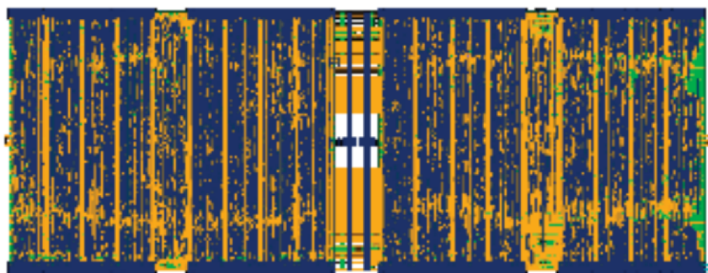
²Samsung Electronics, San Jose, CA

³Samsung Electronics, Suwon, Korea

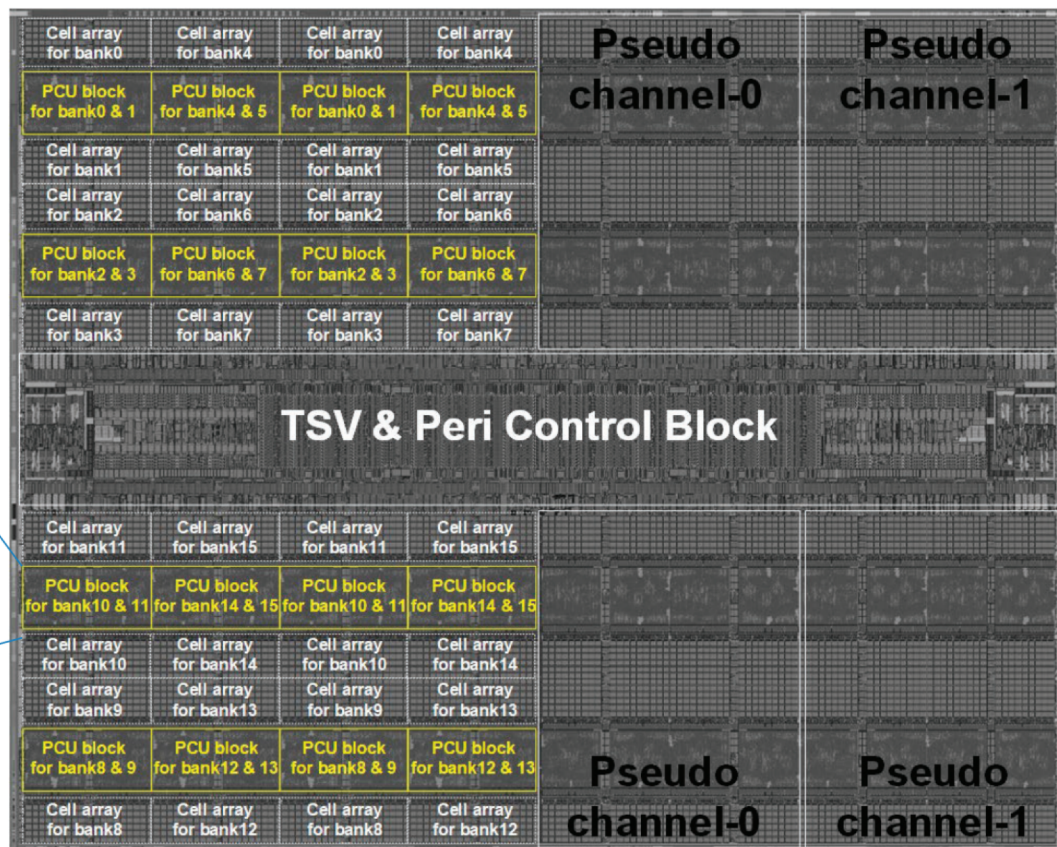
Samsung Function-in-Memory DRAM (2021)

Chip Implementation

- Mixed design methodology to implement FIMDRAM
 - Full-custom + Digital RTL



[Digital RTL design for PCU block]



ISSCC 2021 / SESSION 25 / DRAM / 25.4

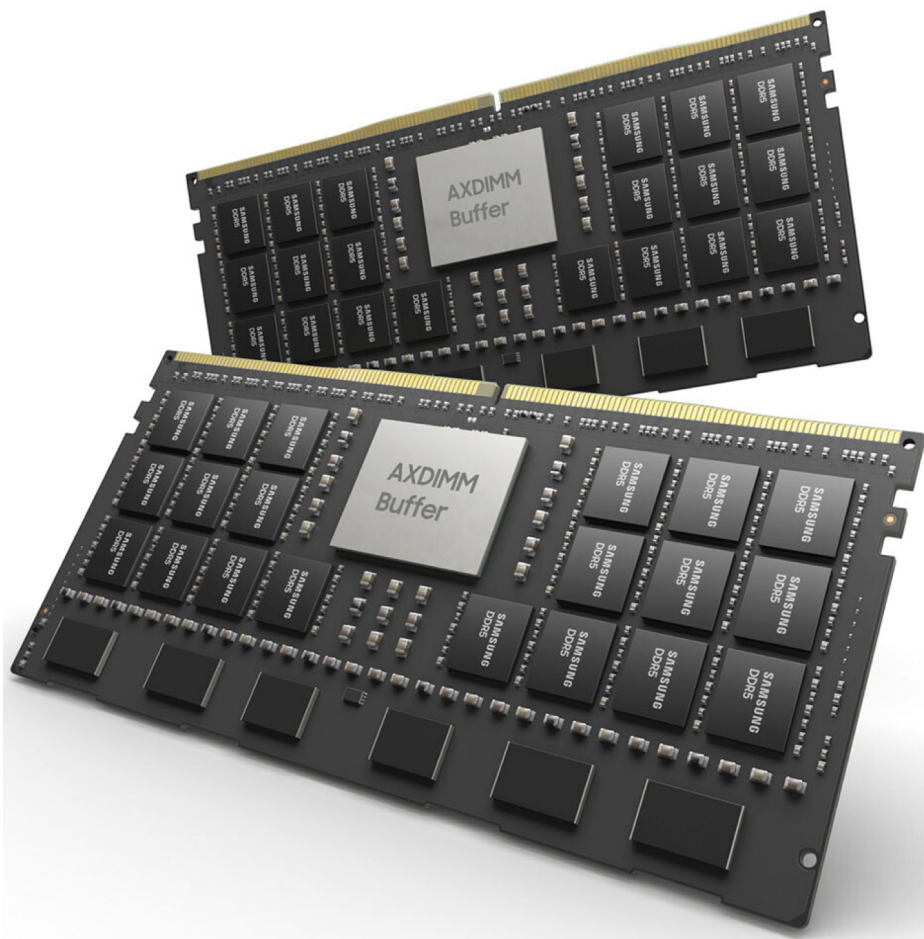
25.4 A 20nm 6Gb Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

Young-Cheon Kwon¹, Suk Han Lee¹, Jaehoon Lee¹, Sang-Hyuk Kwon¹, Je Min Ryu¹, Jong-Pil Son¹, Seongil O¹, Hak-Soo Yu¹, Haesuk Lee¹, Soo Young Kim¹, Youngmin Cho¹, Jin Guk Kim¹, Jongyeon Choi¹, Hyun-Sung Shim¹, Jin Kim¹, BengSeng Phuah¹, HyounMin Kim¹, Myeong Jun Song¹, Ahn Chai¹, Daeho Kim¹, SooYoung Kim¹, Eun-Bong Kim¹, David Wang¹, Shinhaeng Kang¹, Yuhwan Ro¹, Seungwoo Seo¹, JoonHo Song¹, Jaeyoun Yoon¹, Kyomin Sohn¹, Nam Sung Kim¹

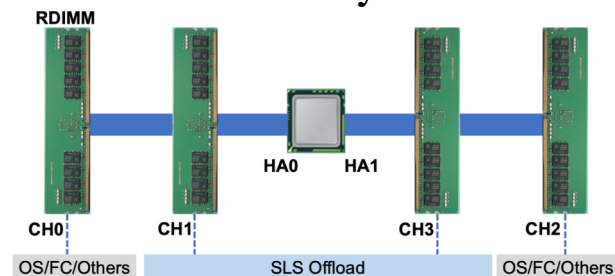
¹Samsung Electronics, Hwaseong, Korea
²Samsung Electronics, San Jose, CA
³Samsung Electronics, Suwon, Korea

Samsung AxDIMM (2021)

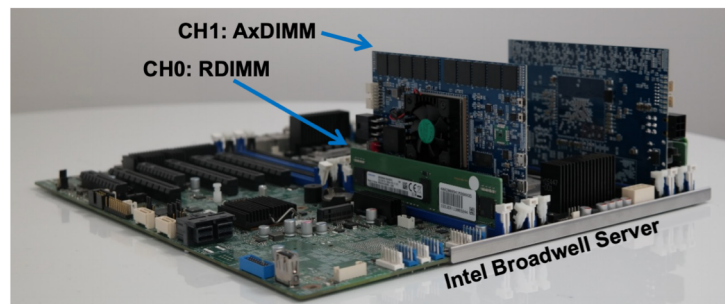
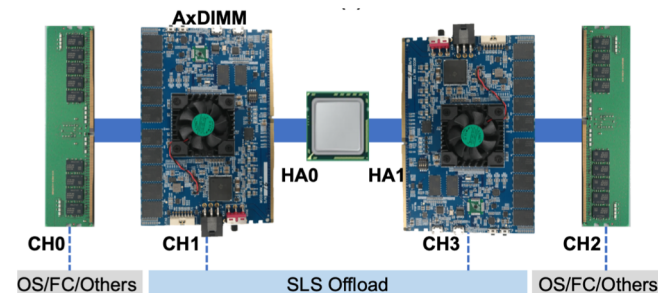
- DIMM-based PIM
 - DLRM recommendation system



Baseline System



AxDIMM System



Key Takeaways

- This P&S is aimed at improving your
 - **Knowledge** in Computer Architecture and Processing-in-Memory
 - **Technical skills** in programming parallel (PIM) architectures and CompArch simulation
 - **Critical thinking and analysis**
 - **Interaction** with a nice group of researchers
 - Familiarity with key **research directions**
 - **Technical presentation** of your project

Key Goal

(Learn how to) overcome
the data movement bottleneck
by programming, benchmarking,
exploring different designs of
the PIM computing paradigm

Prerequisites of the Course

- Digital Design and Computer Architecture (or equivalent course)
 - <https://safari.ethz.ch/digitaltechnik/spring2021/doku.php?id=schedule>
- Familiarity with C/C++ programming
 - FPGA implementation or GPU programming (desirable)
- Interest in
 - future computer architectures and computing paradigms
 - discovering why things do or do not work and solving problems
 - making systems efficient and usable

Course Info: Who Are We? (I)



■ Onur Mutlu

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

■ Research and Teaching in:

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

Course Info: Who Are We? (II)

- Lead Supervisor:

- Dr. Juan Gómez Luna



- Supervisors:

- Dr. Haiyu Mao
- Geraldo F. Oliveira
- Konstantinos Kanellopoulos
- Nika Mansouri Ghiasi



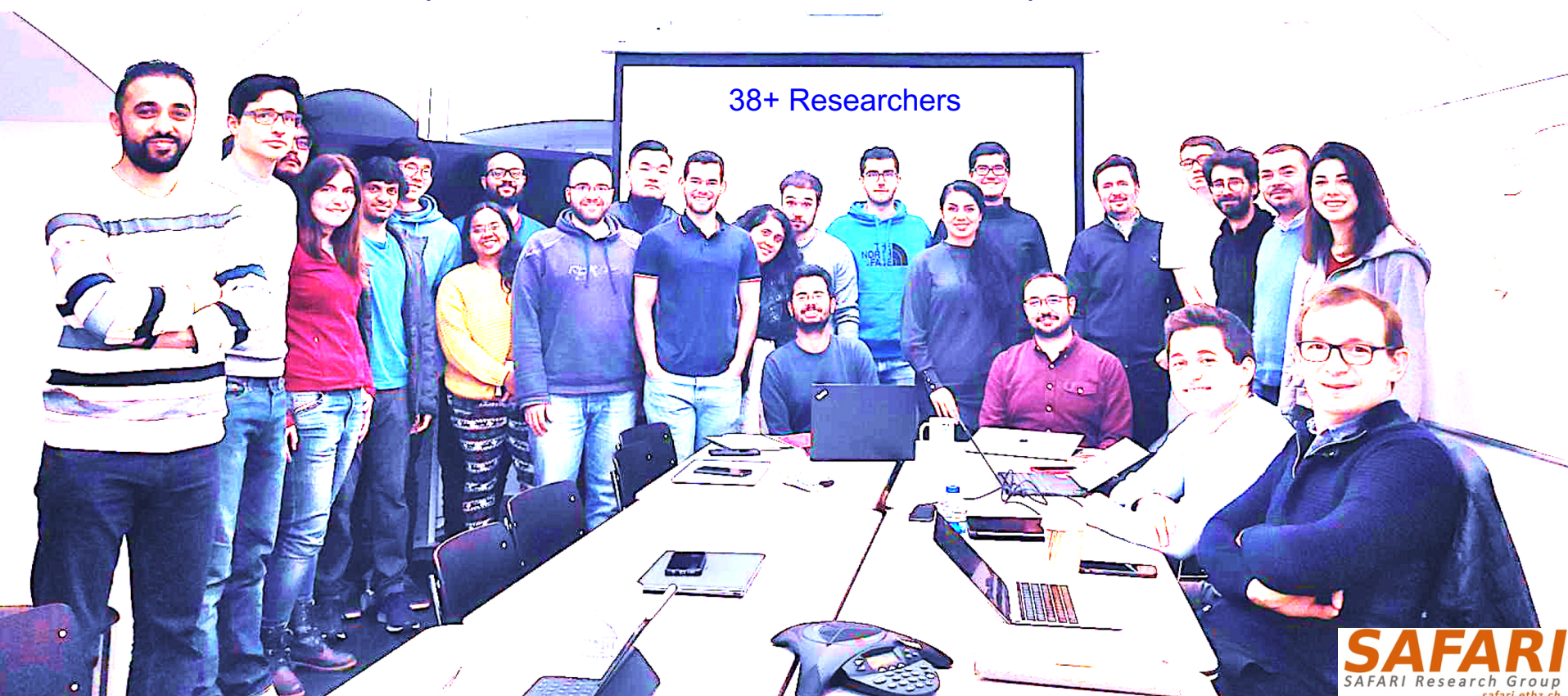
- Get to know us and our research

- <https://safari.ethz.ch/safari-group/>

Onur Mutlu's SAFARI Research Group

Computer architecture, HW/SW, systems, bioinformatics, security, memory

<https://safari.ethz.ch/safari-newsletter-april-2020/>



SAFARI
SAFARI Research Group
safari.ethz.ch

Think BIG, Aim HIGH!

<https://safari.ethz.ch>

SAFARI Newsletter January 2021 Edition

- <https://safari.ethz.ch/safari-newsletter-january-2021/>



SAFARI
SAFARI Research Group

Newsletter
January 2021

*Think Big, Aim High, and
Have a Wonderful 2021!*



Dear SAFARI friends,

Happy New Year! We are excited to share our group highlights with you in this second edition of the SAFARI newsletter (You can find the first edition from April 2020 [here](#)). 2020 has

SAFARI Live Seminars (I)

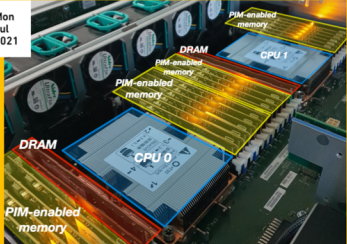

SAFARI Live Seminars in Computer Architecture

Dr. Juan Gómez Luna, ETH Zurich

Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization

SAFARI
SAFARI Research Group

12 Mon Jul 2021



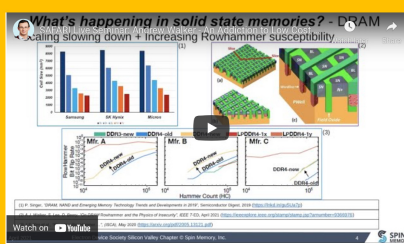

SAFARI Live Seminars in Computer Architecture

Dr. Andrew Walker, Schiltron Corporation & Nexgen Power Systems

An Addition to Low Cost Per Memory Bit – How to Recognize It and What to Do About It

SAFARI
SAFARI Research Group

19 Mo Jul 2021



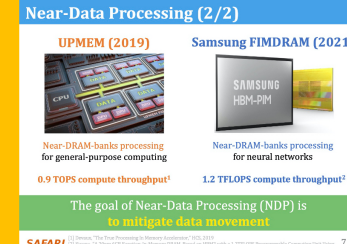

SAFARI Live Seminars in Computer Architecture

Geraldo F. Oliveira, ETH Zurich

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

SAFARI
SAFARI Research Group

22 Do Jul 2021



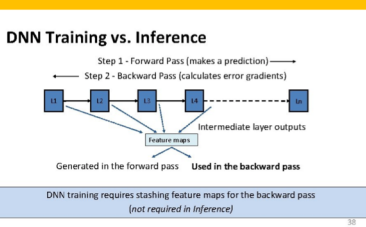

SAFARI Live Seminars in Computer Architecture

Gennady Pekhimenko, University of Toronto

Efficient DNN Training at Scale: from Algorithms to Hardware

SAFARI
SAFARI Research Group

5 Do Aug 2021



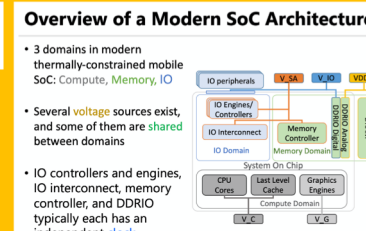

SAFARI Live Seminars in Computer Architecture

Jawad Haj-Yahya, Huawei Research Center Zurich

Power Management Mechanisms in Modern Microprocessors and Their Security Implications

SAFARI
SAFARI Research Group

16 Mo Aug 2021



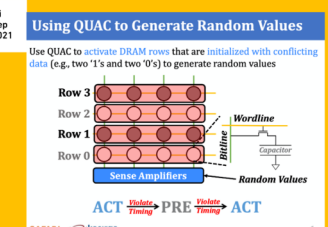

SAFARI Live Seminars in Computer Architecture

Ataberk Olgun, TOBB & ETH Zurich

QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips

SAFARI
SAFARI Research Group

15 Mi Sep 2021



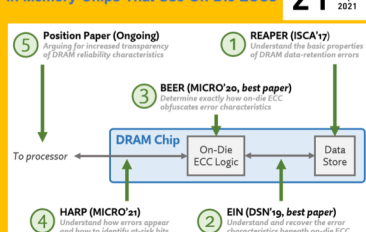

SAFARI Live Seminars in Computer Architecture

Minesh Patel, ETH Zurich

Enabling Effective Error Mitigation in Memory Chips That Use On-Die ECCs

SAFARI
SAFARI Research Group

21 Tues Sep 2021

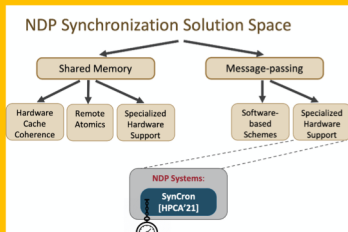



SAFARI Live Seminars in Computer Architecture

Christina Giannoula, National Technical University of Athens
Efficient Synchronization Support for Near-Data-Processing Architectures

SAFARI
SAFARI Research Group

27 Mo Sep 2021



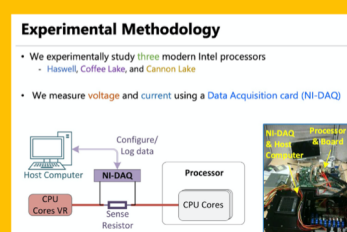

SAFARI Live Seminars in Computer Architecture

Jawad Haj-Yahya, Huawei Research Center Zurich

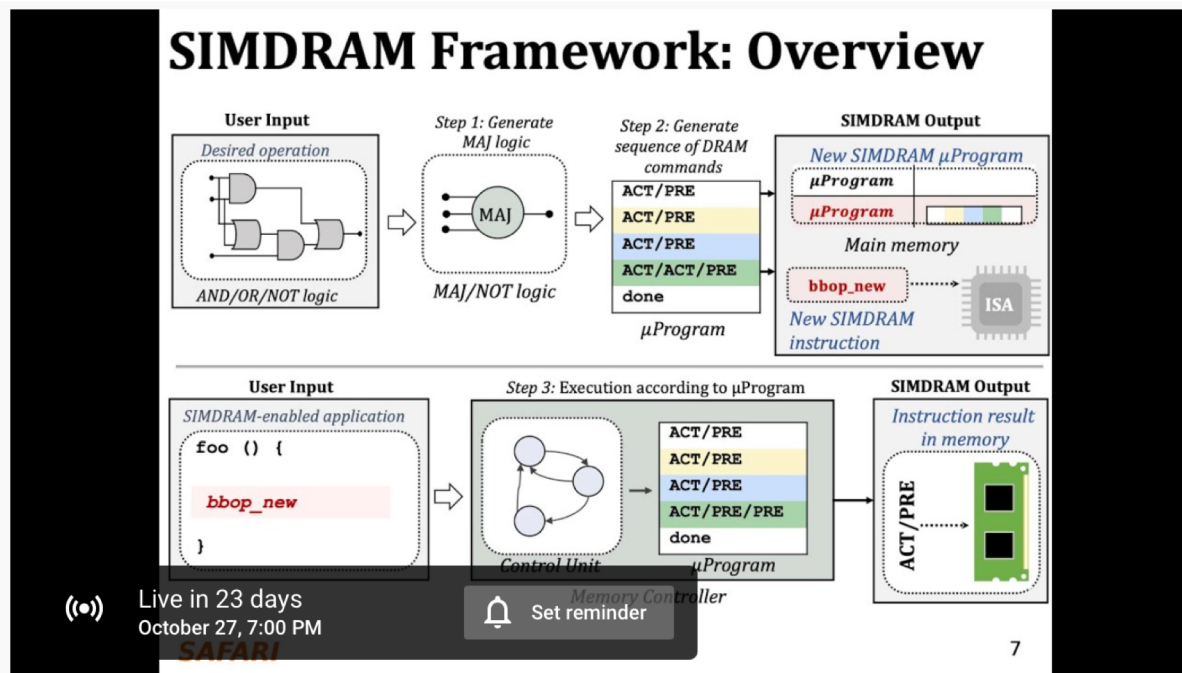
Security Implications of Power Management Mechanisms in Modern Processors, Current Studies and Future Trends

SAFARI
SAFARI Research Group

4 Mo Okt 2021



SAFARI Live Seminars (II)



SAFARI Live Seminar - Data-Centric & Data-Aware Frameworks for Fundamentally Efficient Data Handling

2 waiting • Scheduled for Oct 27, 2021

👍 4 💬 0 ➦ SHARE ⚙️ SAVE ...



Onur Mutlu Lectures
19K subscribers

SUBSCRIBED



Title: Data-Centric and Data-Aware Frameworks for Fundamentally Efficient Data Handling in Modern Computing Systems

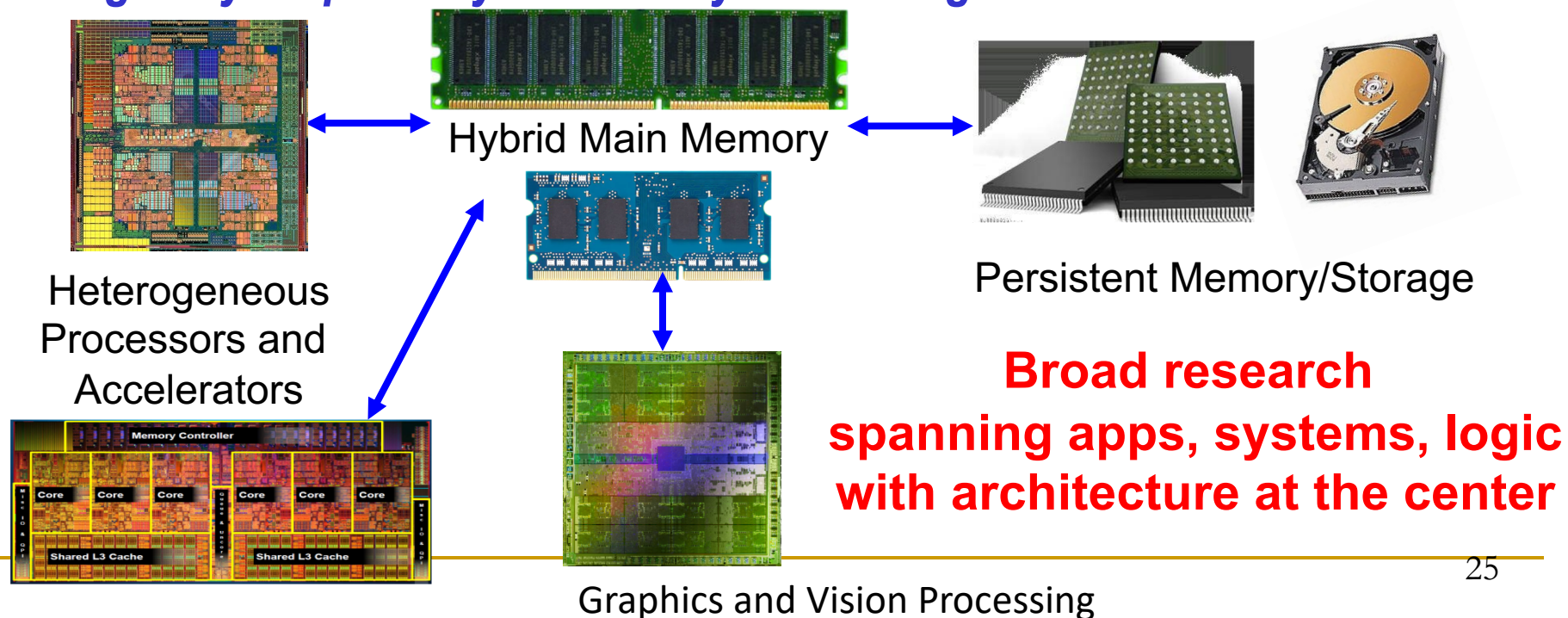
Speaker: Nastaran Hajinazar, SAFARI Research Group, <https://www.linkedin.com/in/nastaran-...>

<https://youtu.be/XIfPHtvA9rw>

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**



Course Info: How About You?

- Let us know your background, interests
- Why did you join this P&S?

Course Requirements and Expectations

- Attendance required for all meetings
- Study the learning materials
- Each student will carry out a hands-on project
 - Build, implement, code, and design with close engagement from the supervisors
- Participation
 - Ask questions, contribute thoughts/ideas
 - Read relevant papers

We will help in all projects!

If your work is really good, you may get it published!

Course Website

- https://safari.ethz.ch/projects_and_seminars/doku.php?id=processing_in_memory
- Useful information about the course
- Check your email frequently for announcements
- We will also have Moodle for Q&A

Meeting 1

■ Required materials:

1. Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,

"A Modern Primer on Processing in Memory"

*Invited Book Chapter in **Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann**, Springer, to be published in 2021.*

[[Tutorial Video on "Memory-Centric Computing Systems"](#) (1 hour 51 minutes)]

2. Onur Mutlu,

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

Keynote talk at 37th IEEE International Conference on Computer Design (ICCD), Abu Dhabi, UAE, 19 November 2019.

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

[[Related Overview Paper I](#)]

[[Related Overview Paper II](#)]

[[Talk Video \(1 hour 18 minutes\)](#)]

■ Recommended materials:

3. 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, November/December 2019.

[[Preliminary arXiv version](#)]

4. Computation in Memory (Professor Onur Mutlu, lecture, Fall 2020).

([PDF](#)) ([PPT](#))[Video](#)

5. Near-data Processing (Professor Onur Mutlu, lecture, Fall 2020).

([PDF](#)) ([PPT](#))[Video](#)

6. Real Processing-in-DRAM with UPMEM (Dr. Juan Gomez Luna, lecture, Fall 2020).

([PDF](#)) ([PPT](#))[Video](#)

Meeting 2 (October 12th)

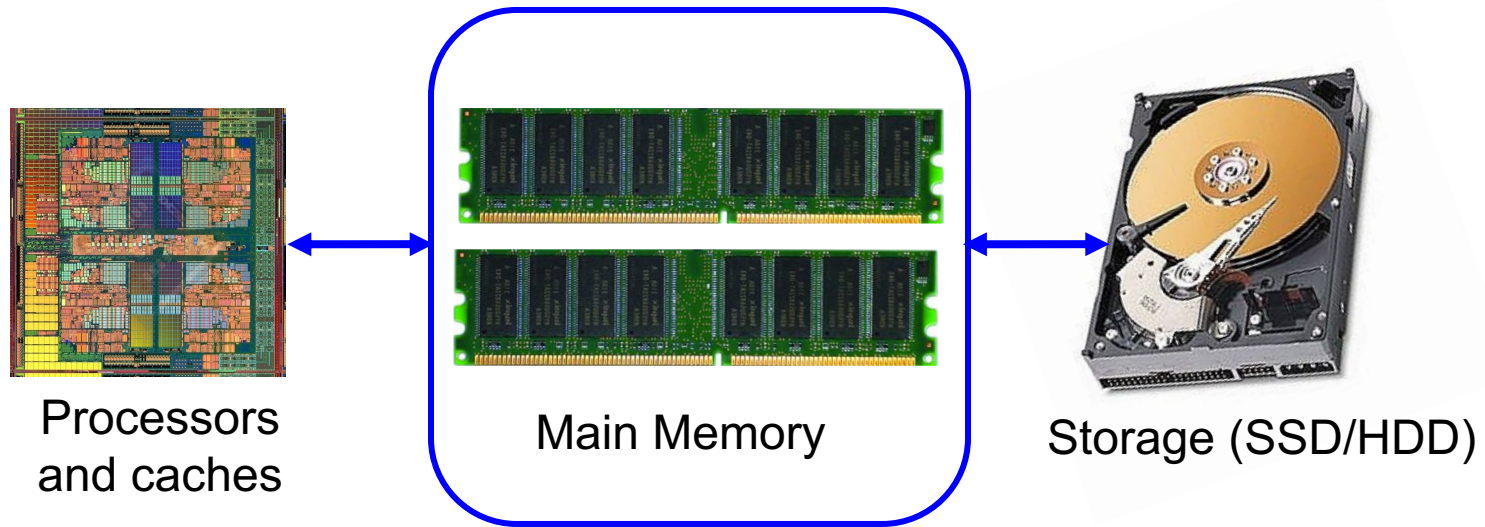
- We will **announce the projects** and will give you some description about them
- We will give you a chance to select a project
- Then, we will have **1-1 meetings** to match your interests, skills, and background with a suitable project
- It is important that you **study the learning materials** before our next meeting!

Next Meetings

- Individual meetings with your mentor/s
- Tutorials and short talks
 - PIM programming
 - Recent research works
- Presentation of your work

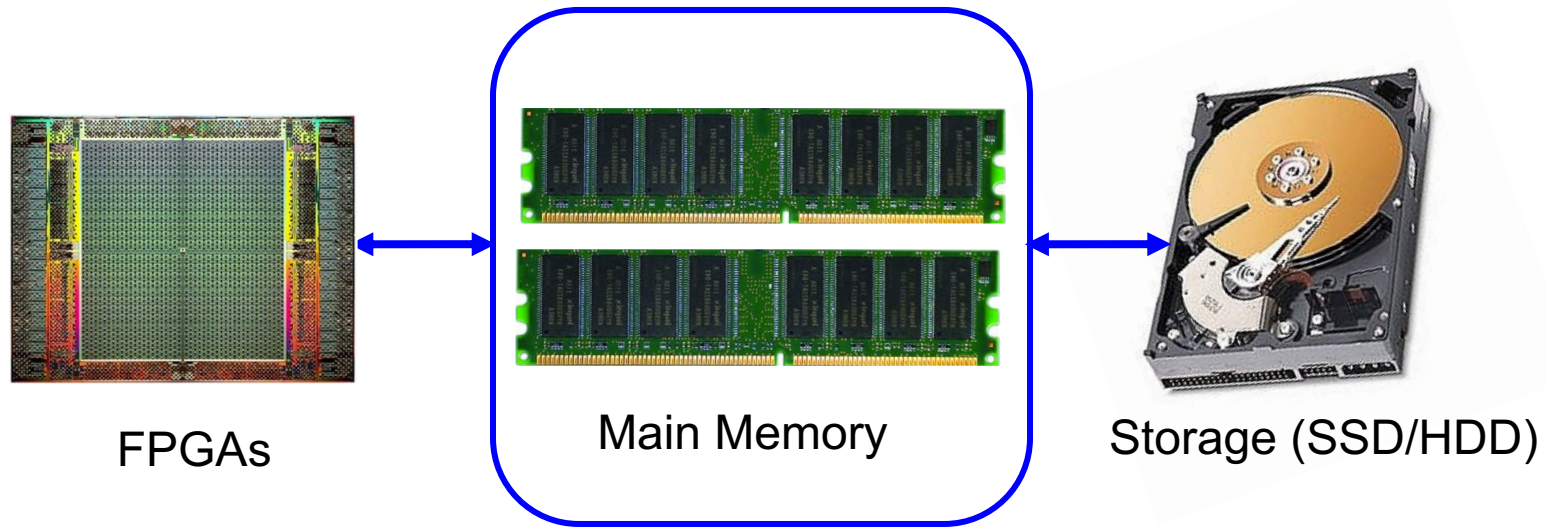
An Introduction to Processing-in-Memory

The Main Memory System



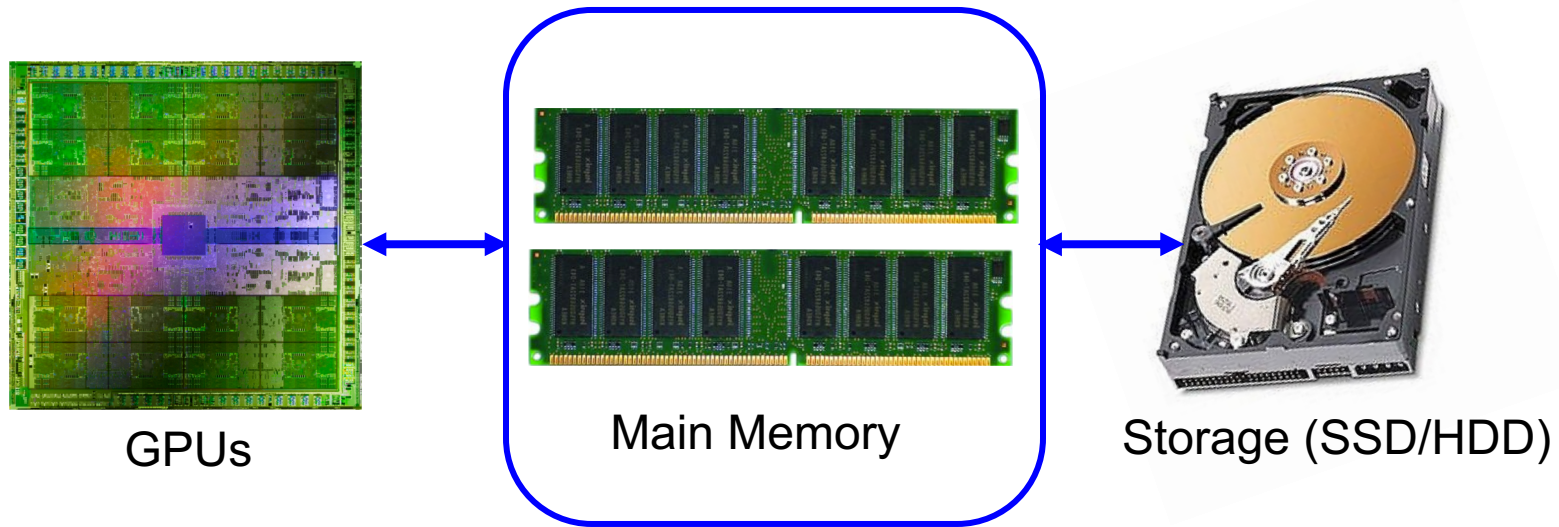
- Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor
- Main memory system must scale (in *size, technology, efficiency, cost, and management algorithms*) to maintain performance growth and technology scaling benefits

The Main Memory System



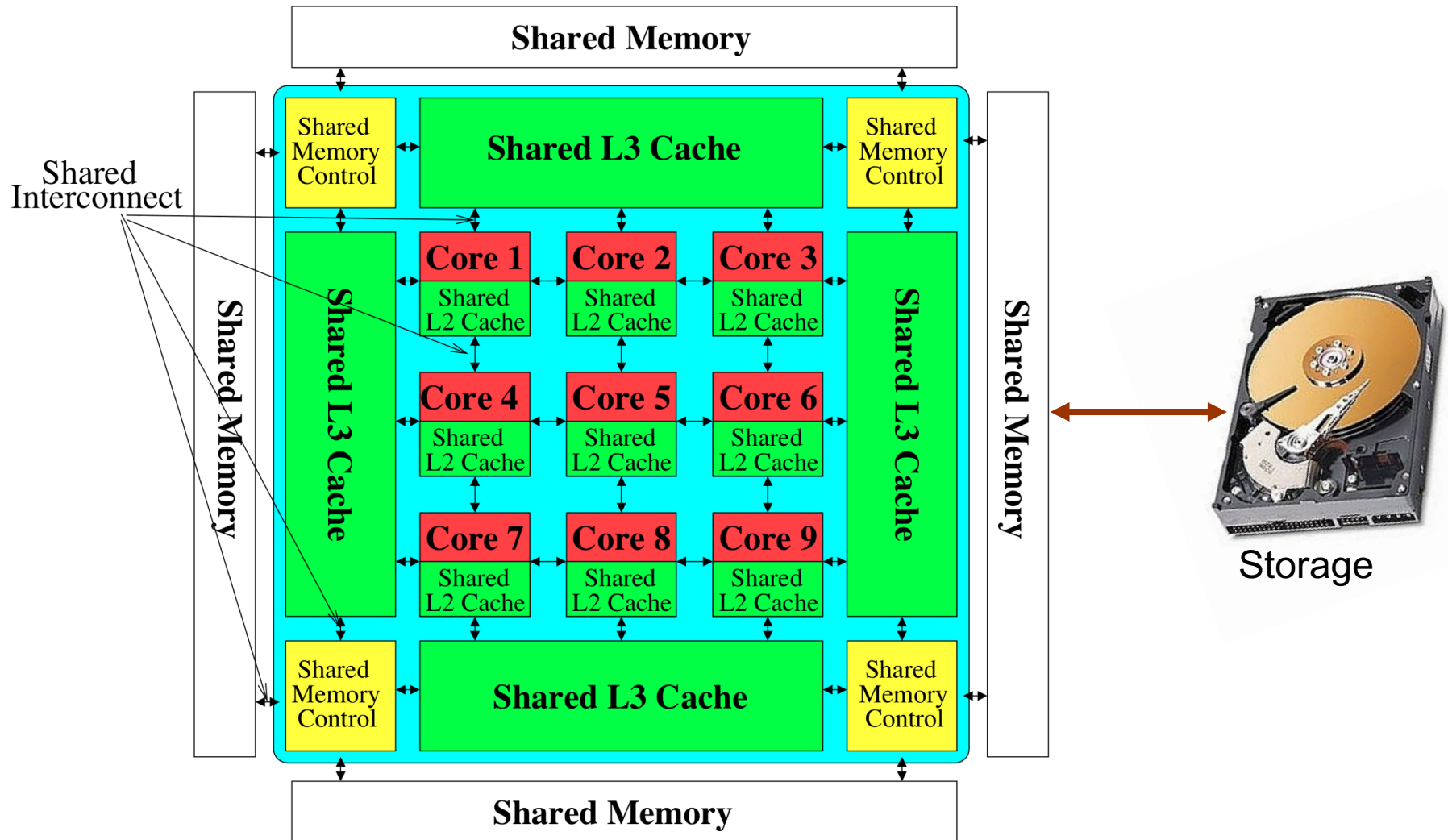
- Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor
- Main memory system must scale (in *size, technology, efficiency, cost, and management algorithms*) to maintain performance growth and technology scaling benefits

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

Three Key Systems Trends

1. Data access is a major bottleneck

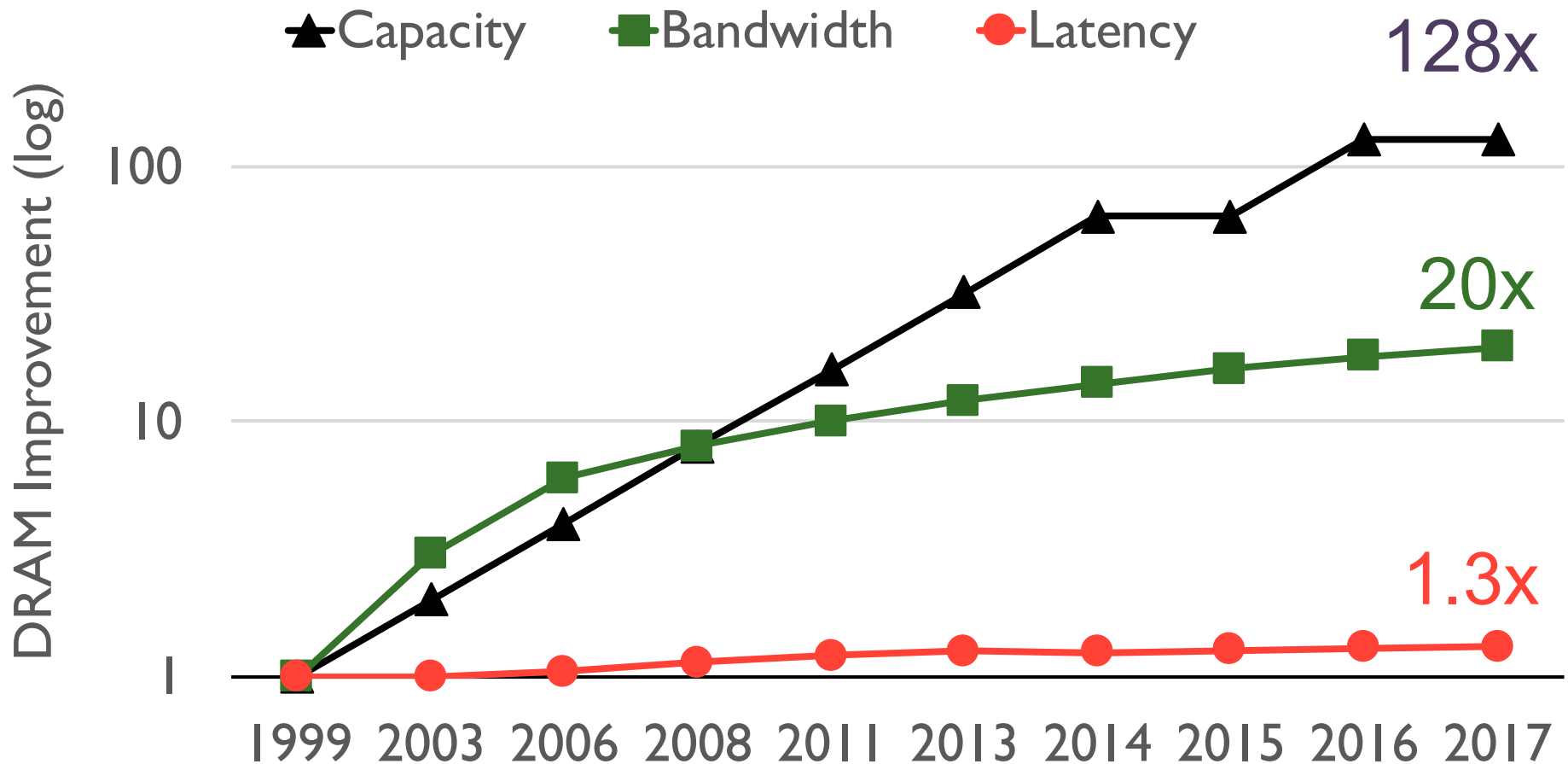
- Applications are increasingly data hungry

2. Energy consumption is a key limiter

3. Data movement energy dominates compute

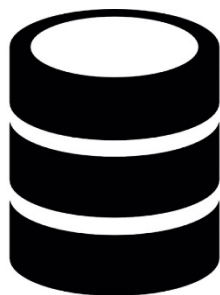
- Especially true for off-chip to on-chip movement

Example: Capacity, Bandwidth & Latency



Memory latency remains almost constant

The Need for More Memory Performance



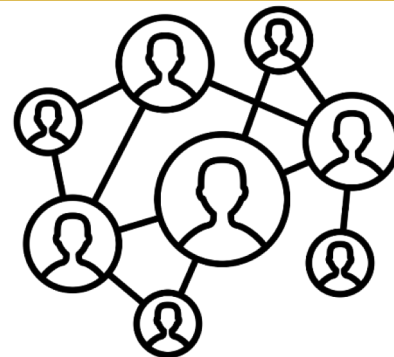
In-memory Databases

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



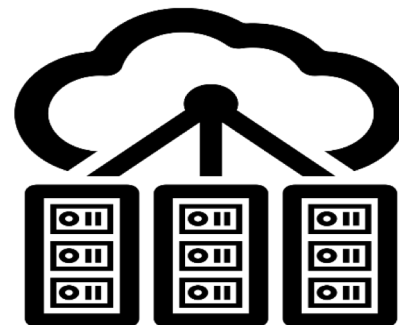
In-Memory Data Analytics

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



Graph/Tree Processing

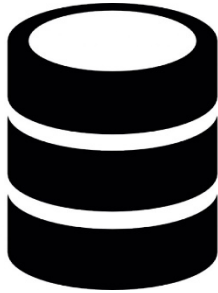
[Xu+, IISWC'12; Umuroglu+, FPL'15]



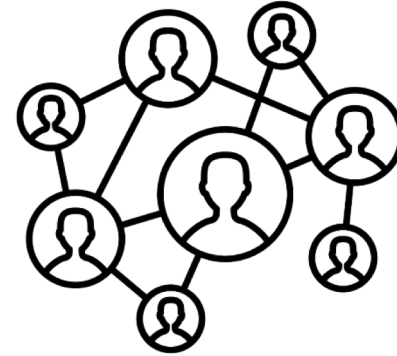
Datacenter Workloads

[Kanev+ (Google), ISCA'15]

DRAM Latency Is Critical for Performance



In-memory Databases



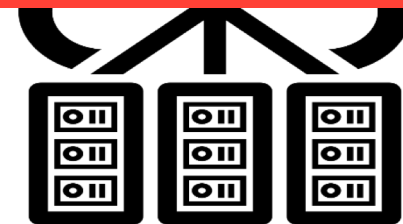
Graph/Tree Processing

Long memory latency → performance bottleneck



In-Memory Data Analytics

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



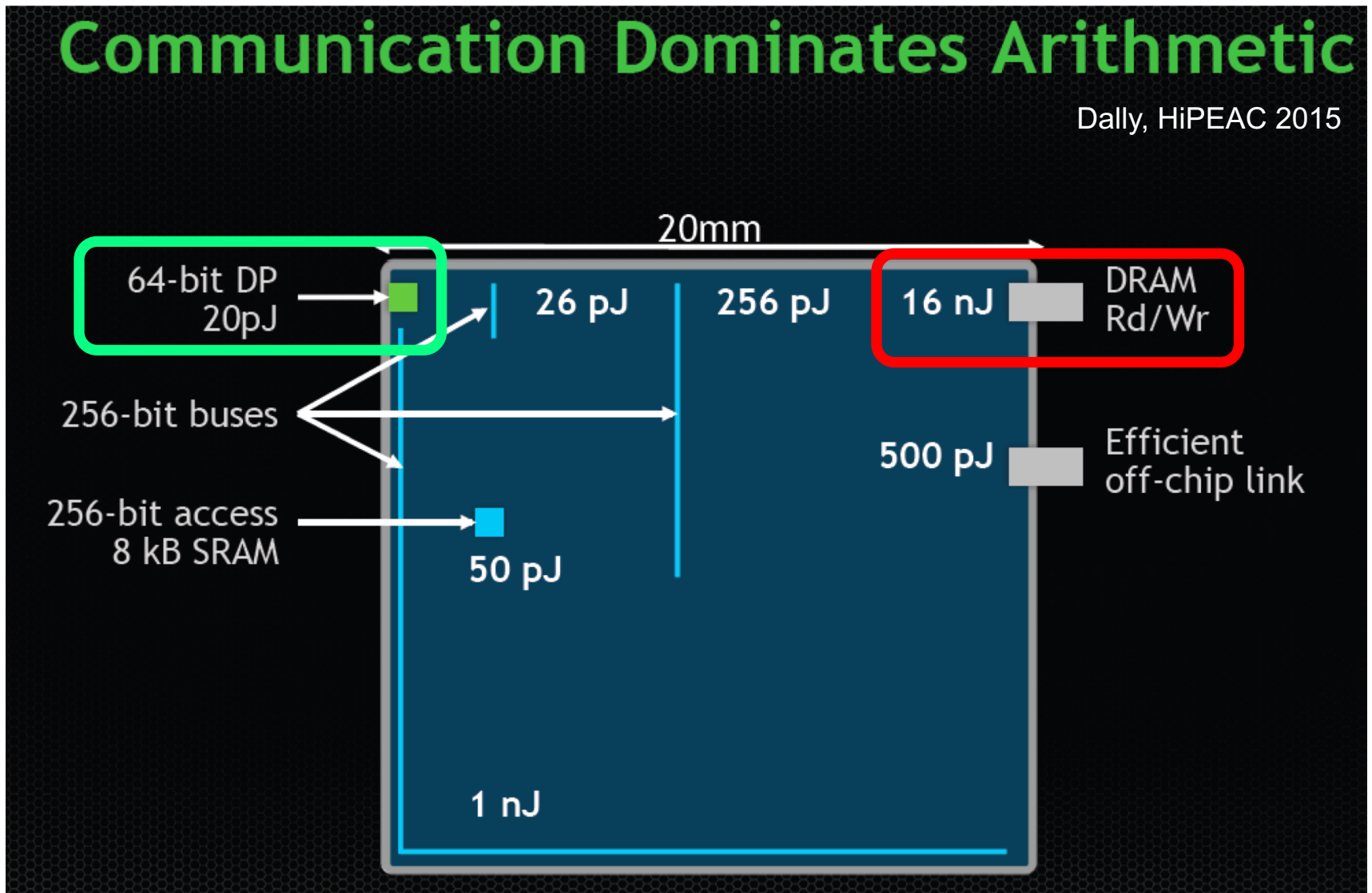
Datacenter Workloads

[Kanev+ (Google), ISCA'15]

The Energy Perspective

Communication Dominates Arithmetic

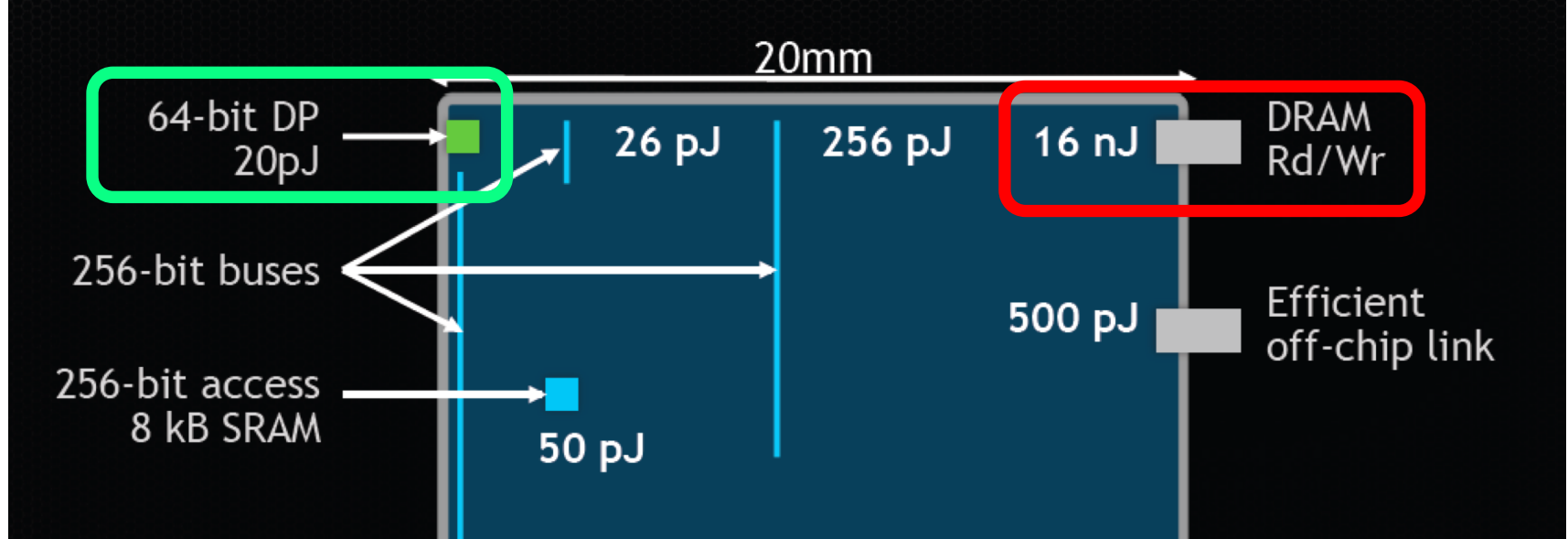
Dally, HiPEAC 2015



Data Movement vs. Computation Energy

Communication Dominates Arithmetic

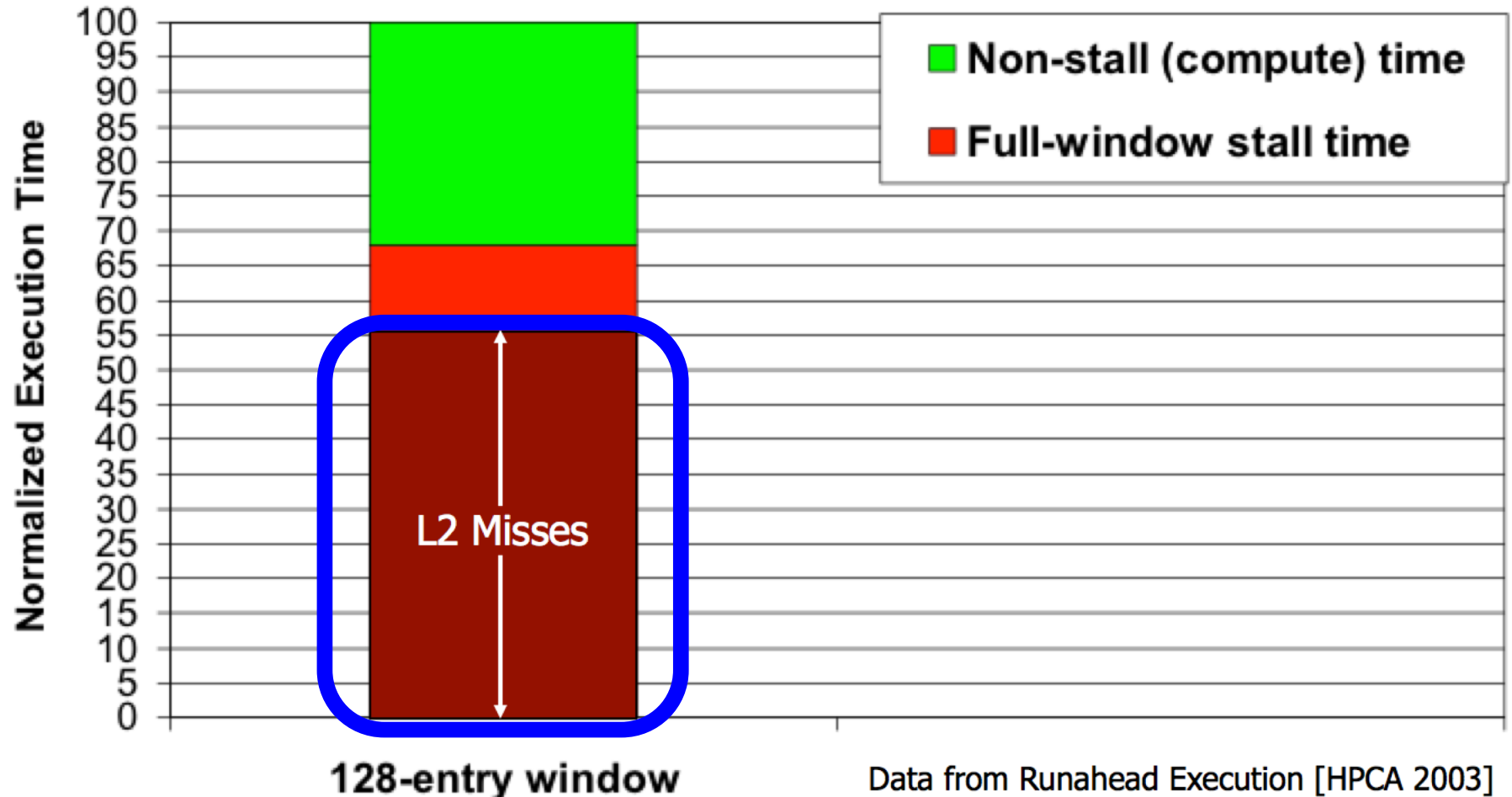
Dally, HiPEAC 2015



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

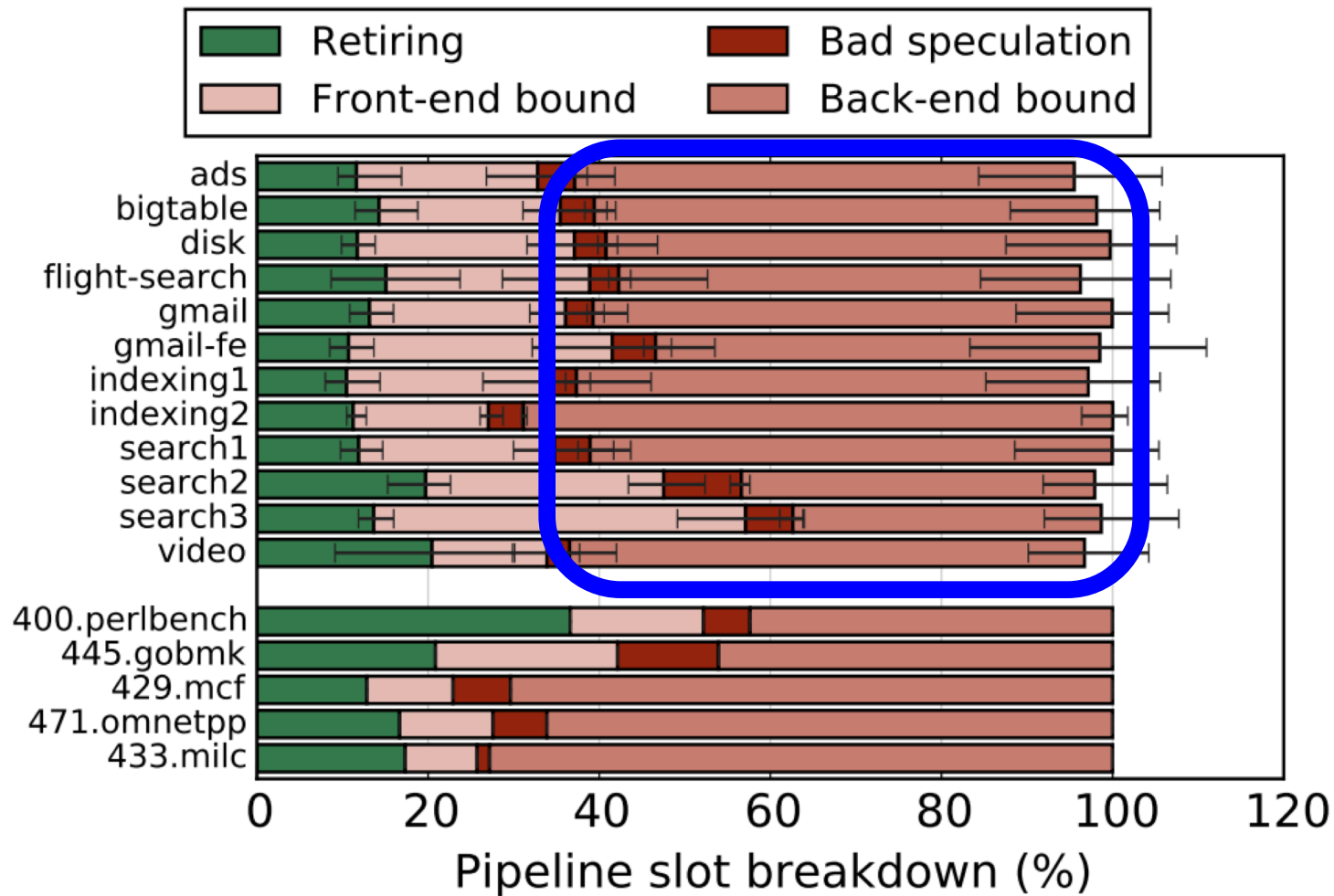
The Performance Perspective (1996-2005)

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



The Performance Perspective (Today)

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



The Problem

Data access is the major performance and energy bottleneck

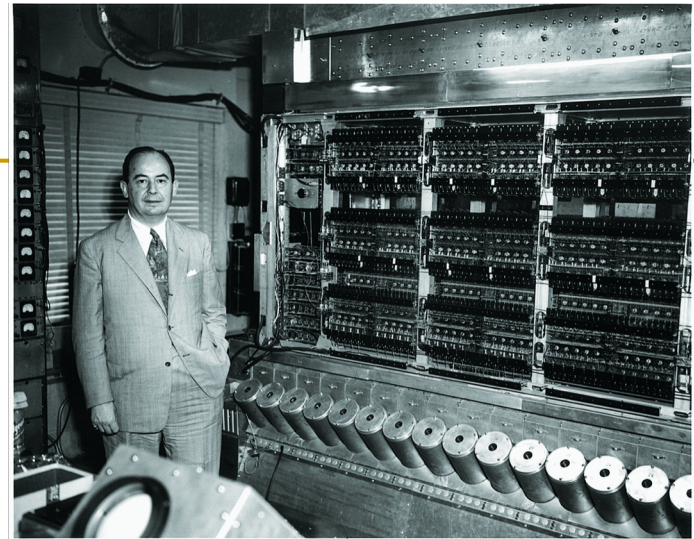
Our current
design principles
cause great energy waste
(and great performance loss)

The Problem

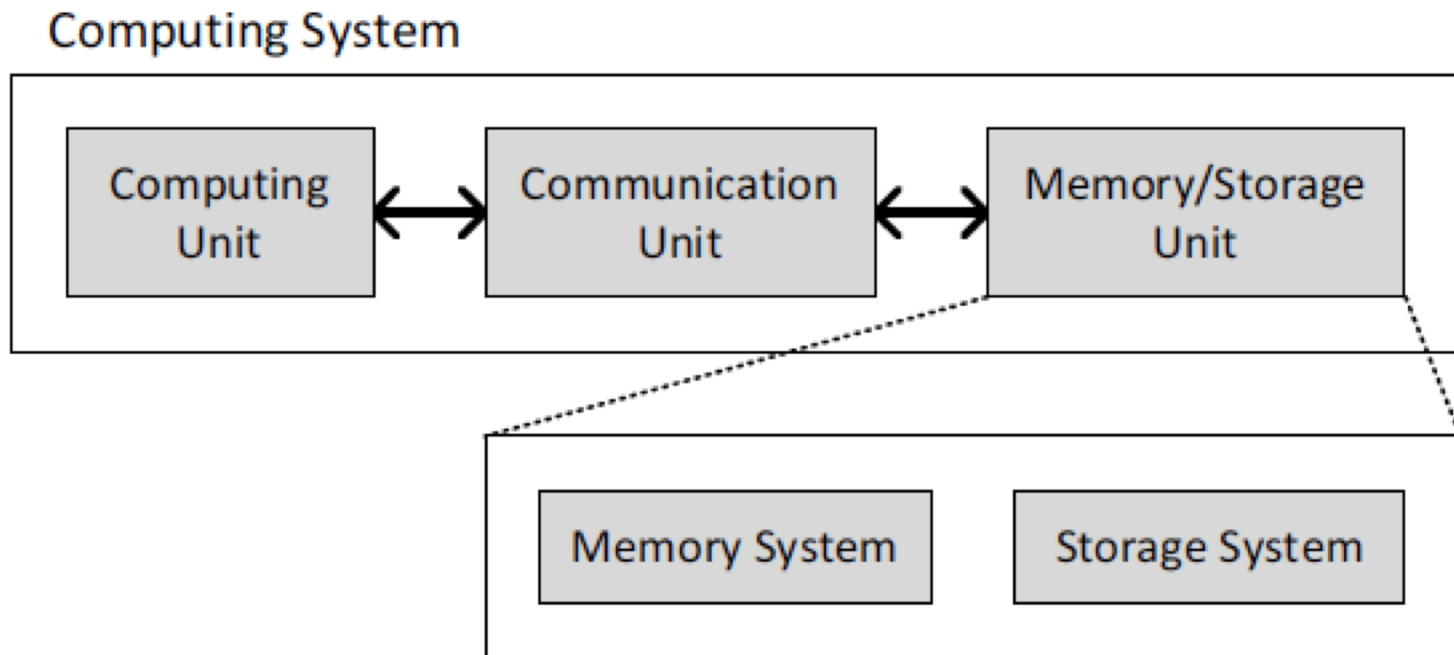
Processing of data
is performed
far away from the data

A Computing System

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

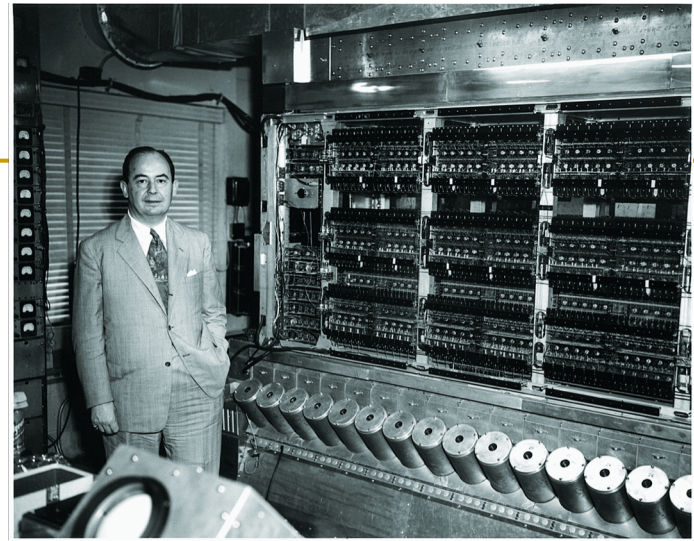


Burks, Goldstein, von Neumann, “Preliminary discussion of the logical design of an electronic computing instrument,” 1946.



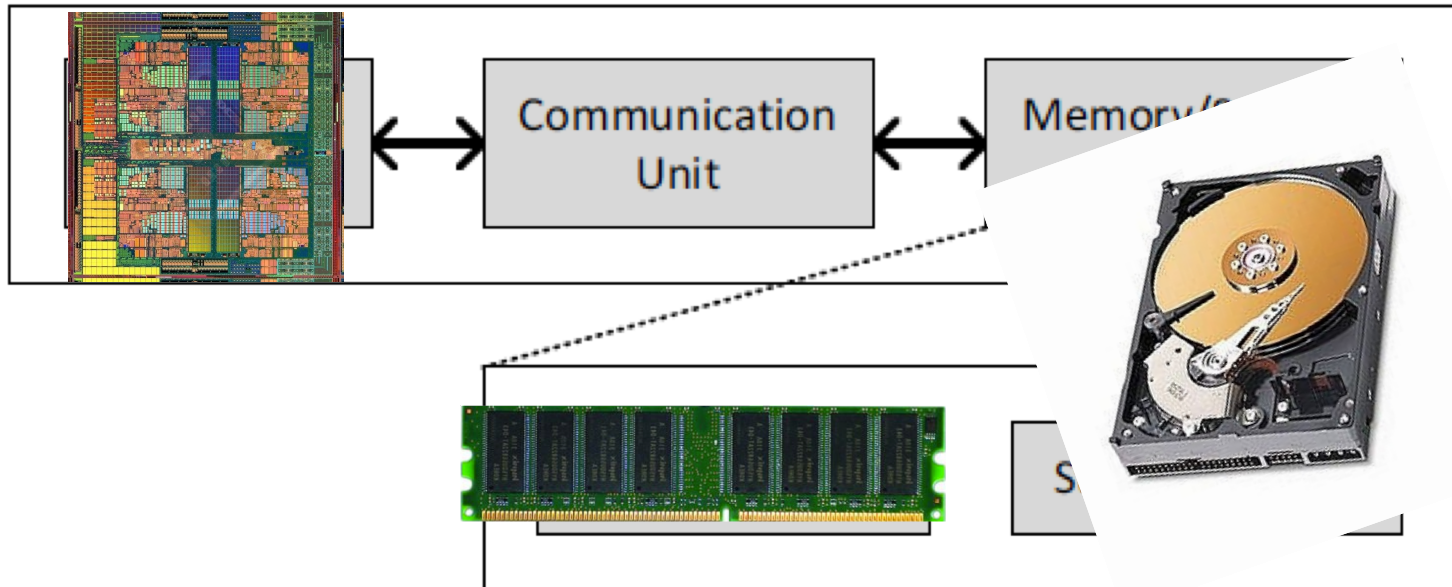
A Computing System

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



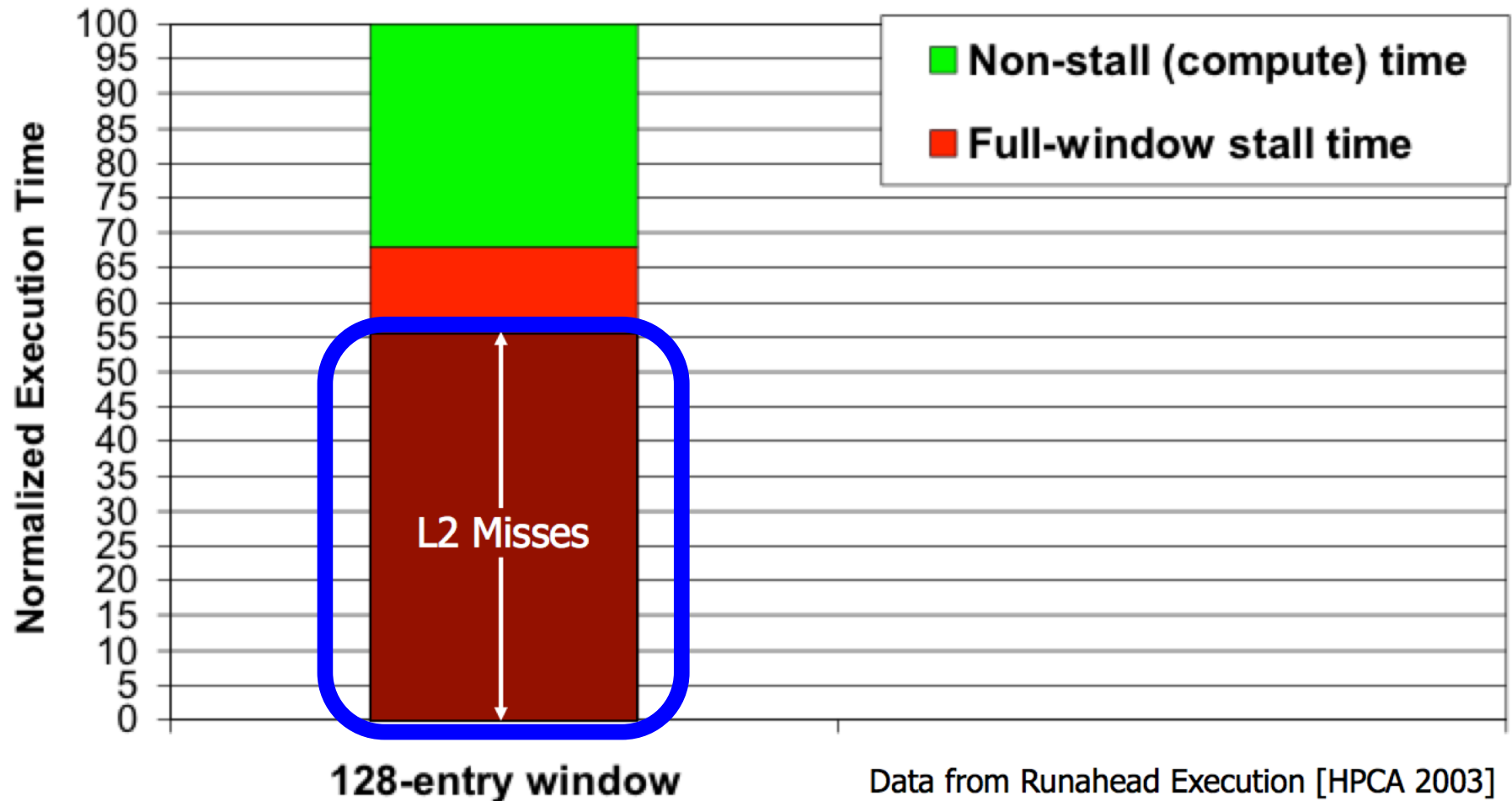
Burks, Goldstein, von Neumann, “Preliminary discussion of the logical design of an electronic computing instrument,” 1946.

Computing System



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)

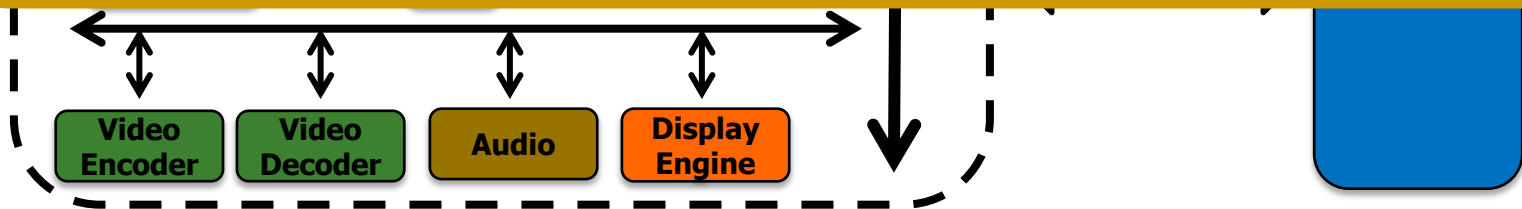
- ❑ To tolerate data access from memory
- ❑ Complex hierarchies and mechanisms
 - Energy inefficient
 - Low performance
 - Complex

Data Movement in Computing Systems

- Data movement dominates performance and is a major system energy bottleneck
 - Comprises 41% of mobile system energy during web browsing*

Compute systems should be more data-centric

Processing-In-Memory proposes computing where it makes sense (where data resides)



*Reducing data Movement Energy via Online Data Clustering and Encoding (MICRO'16)

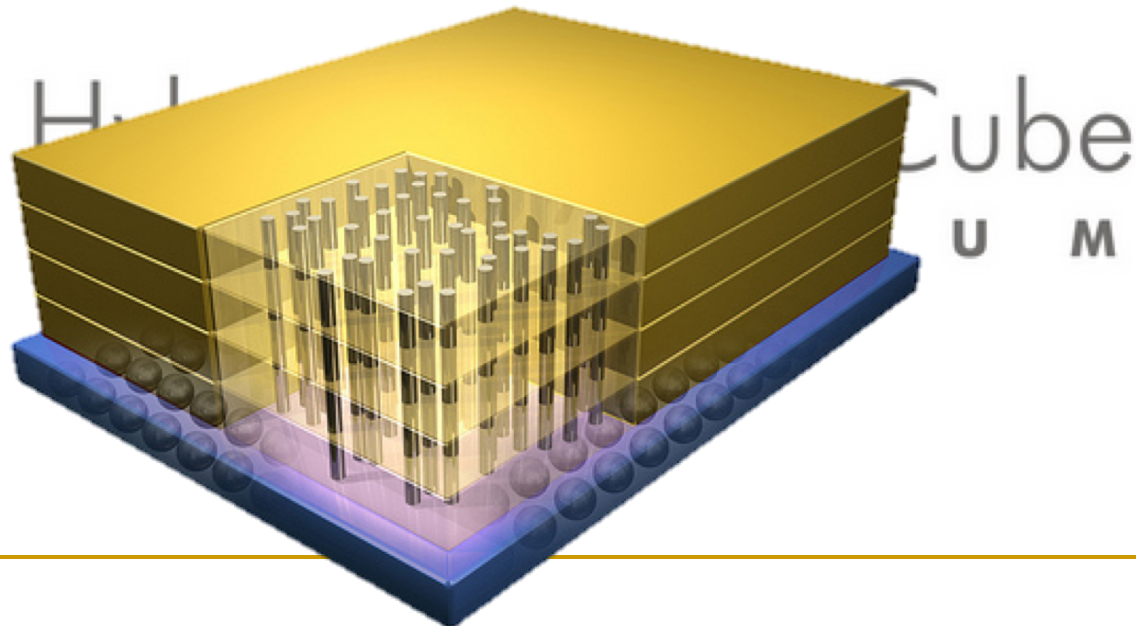
**Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms (IISWC'14)

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

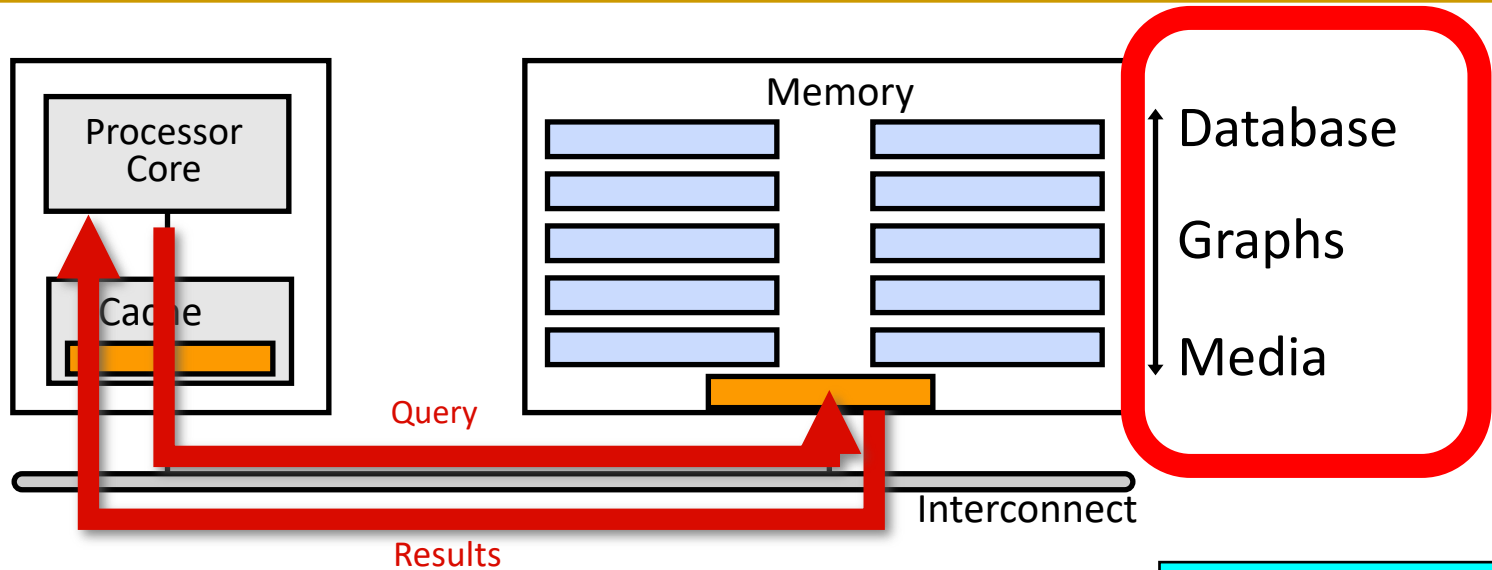
Why In-Memory Computation Today?

- Pull from systems/applications for data-centric execution
- It can be practical today
 - 3D-stacked memories combine logic and memory functionality (relatively) tightly + industry open to new architectures



High Performance and Energy Efficiency

Goal: Processing Inside Memory

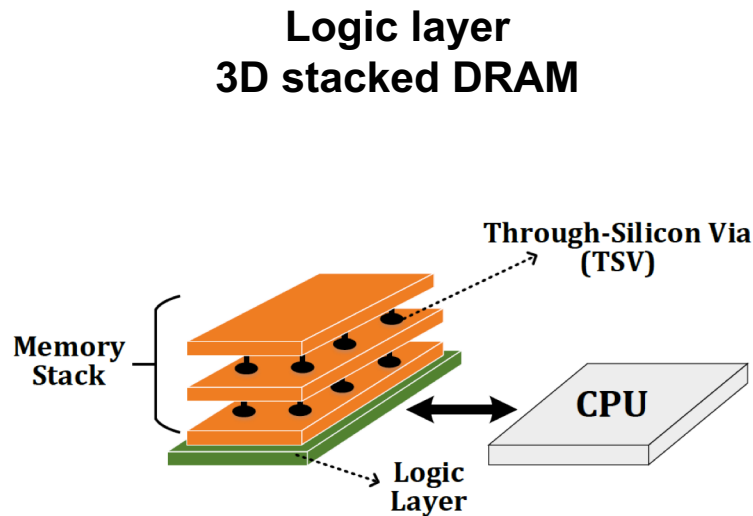


- Many questions... How do we design the:
 - ❑ compute-capable memory & controllers?
 - ❑ processor chip?
 - ❑ software and hardware interfaces?
 - ❑ system software and languages?
 - ❑ algorithms?

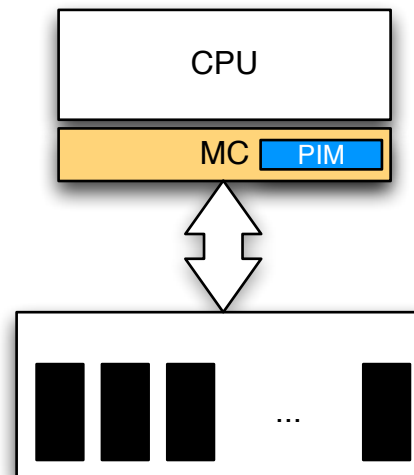
Problem
Algorithm
Program/Language
System Software
SW/HW Interface
Micro-architecture
Logic
Devices
Electrons

Processing In-Memory (PIM)

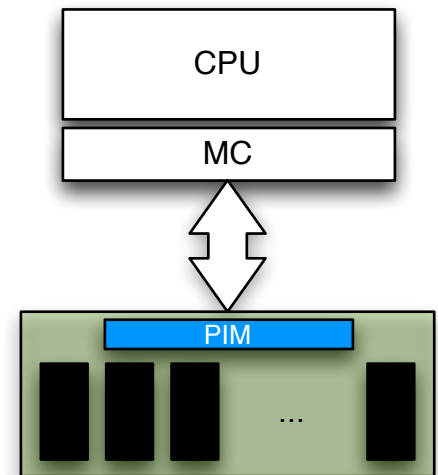
- Near-Data Processing or Processing In-Memory (PIM)
 - Move **computation** closer to **where the data resides**



Memory controller

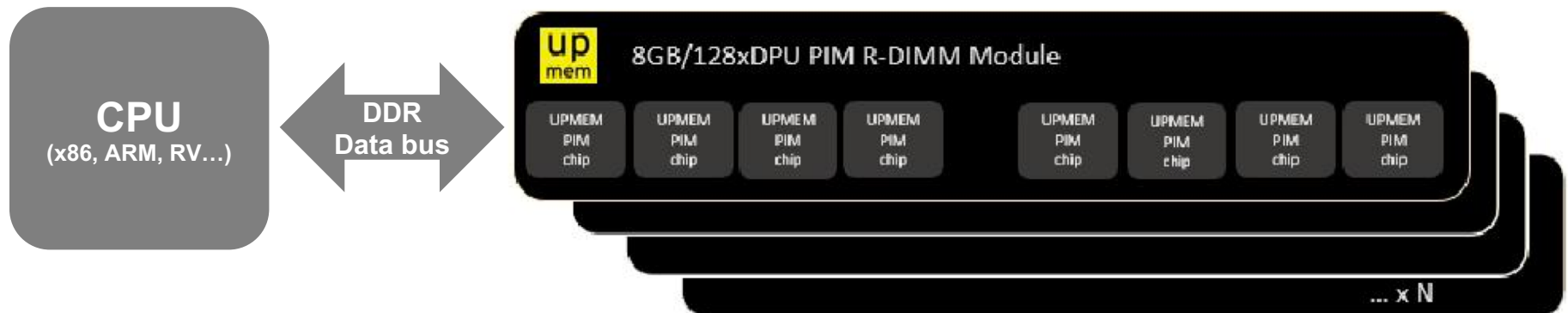
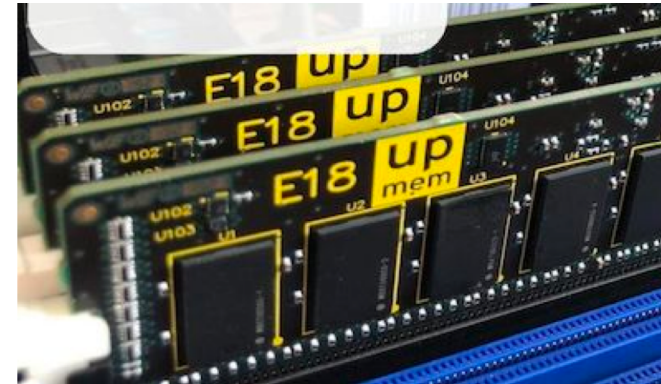


**Memory module
(DIMM)**



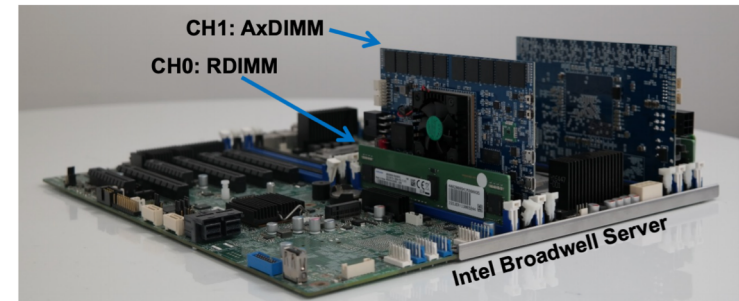
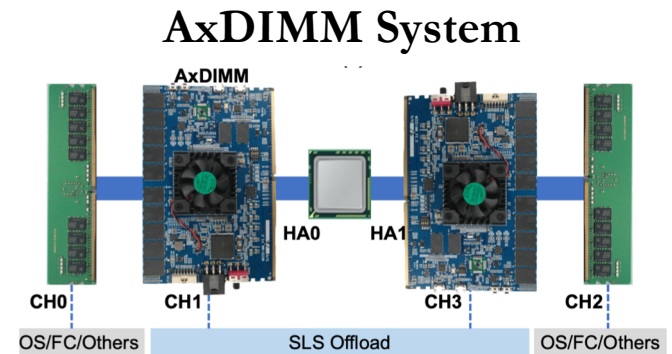
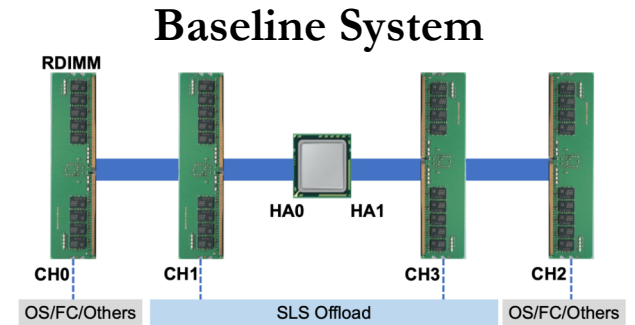
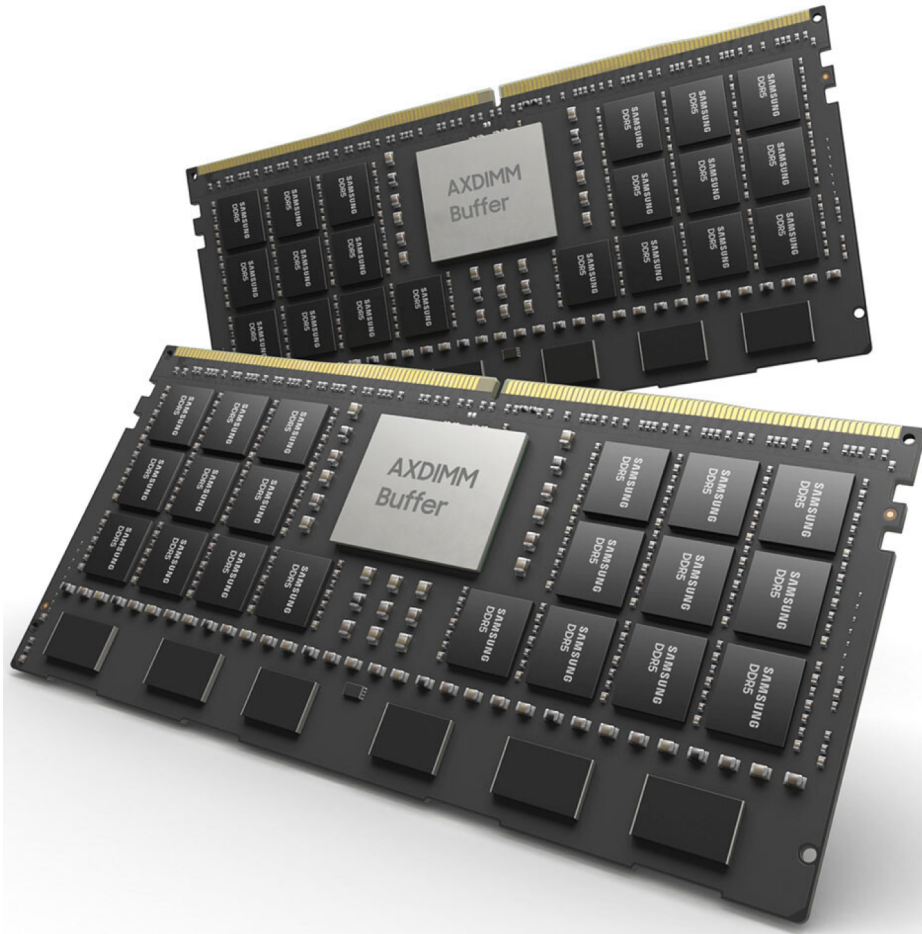
UPMEM Processing-in-DRAM Engine (2019)

- **Processing in DRAM Engine**
- Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.
- Replaces **standard DIMMs**
 - DDR4 R-DIMM modules
 - 8GB+128 DPUs (16 PIM chips)
 - Standard 2x-nm DRAM process
 - **Large amounts of** compute & memory bandwidth



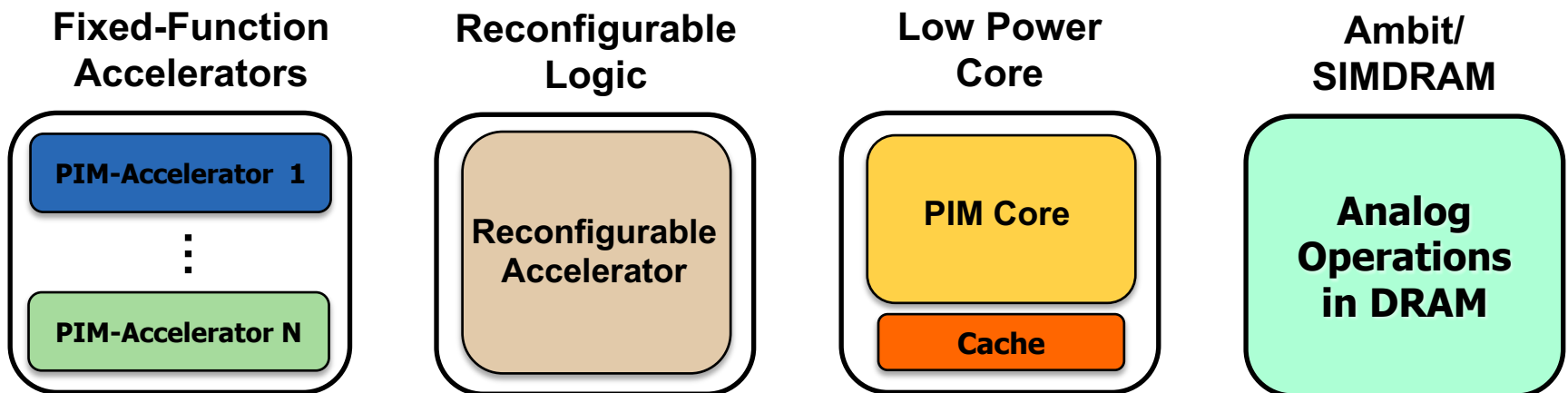
Samsung AxDIMM (2021)

- DIMM-based PIM
 - DLRM recommendation system



Possible Designs

- Fixed-function units
- Reconfigurable architectures
 - FPGAs, CGRA
- General-purpose programmable cores
 - E.g., ARM Cortex R-8, ARM Cortex A-35 (+SIMD units)
 - Possibility of running any workload
- Processing-using-memory:
 - Ambit: In-DRAM bulk bitwise operations (Seshadri+, MICRO'17)
 - SIMDAM: End-to-end framework for SIMD in DRAM (Hajinazar+, ASPLOS'21)



Agenda

- Major Trends Affecting Memory
- Processing in Memory: Two Directions
 - Minimally Changing Memory Chips
 - Exploiting 3D-Stacked Memory

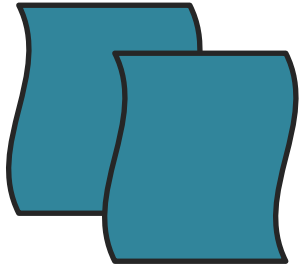
Approach 1: Minimally Changing DRAM

- DRAM has great capability to perform **bulk data movement and computation** internally with small changes
 - Can exploit internal bandwidth 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)
 - "SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM" (Hajinazar et al., ASPLOS 2021)

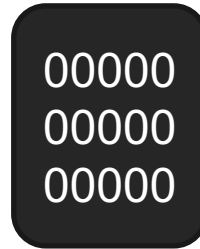
RowClone: In-Memory Copy and Initialization

Starting Simple: Data Copy and Initialization

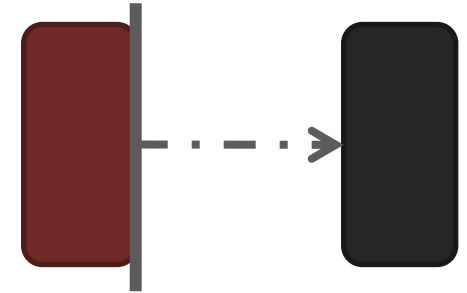
memmove & memcpy: 5% cycles in Google's datacenter [Kanev+ ISCA'15]



Forking



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



Checkpointing



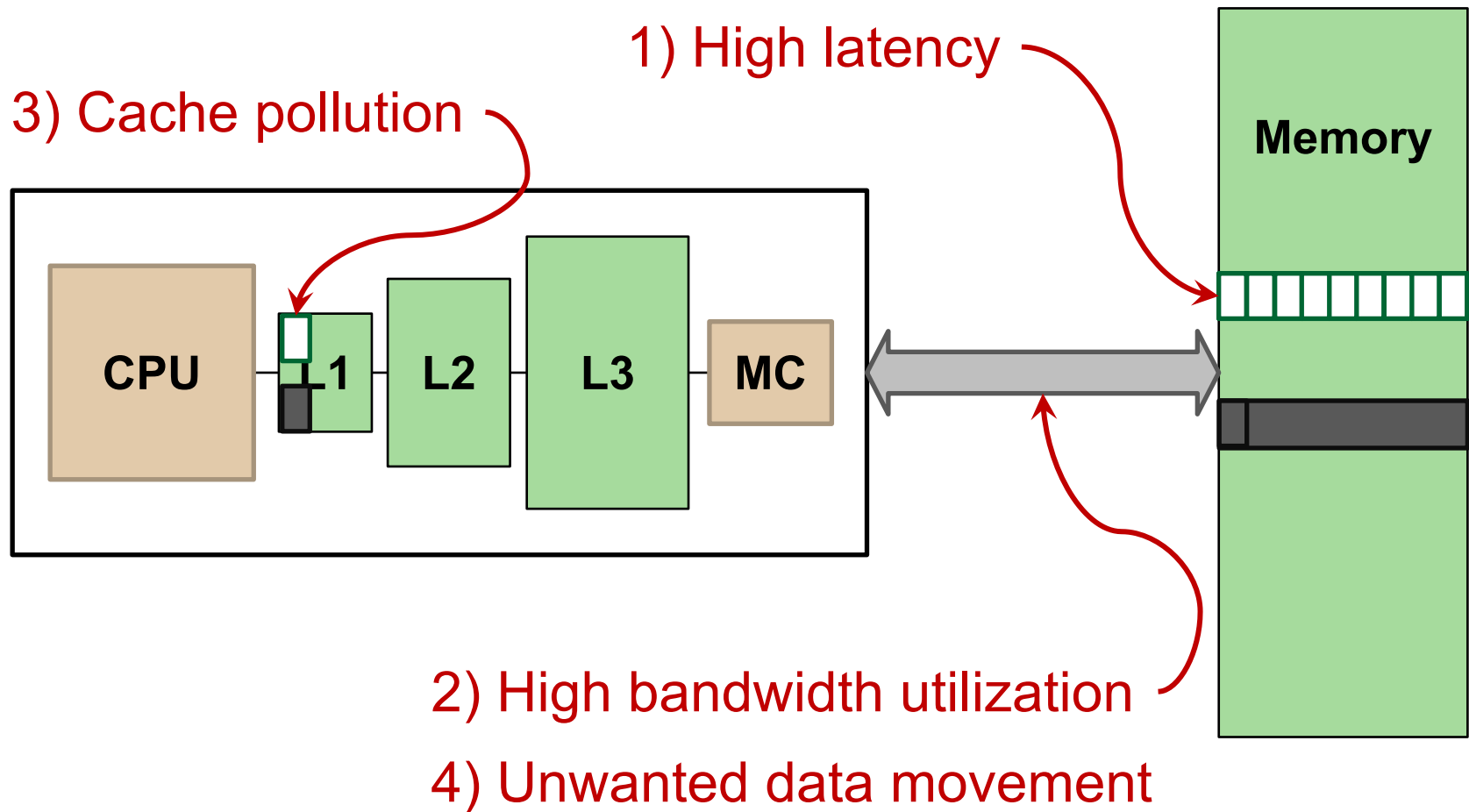
**VM Cloning
Deduplication**



Page Migration

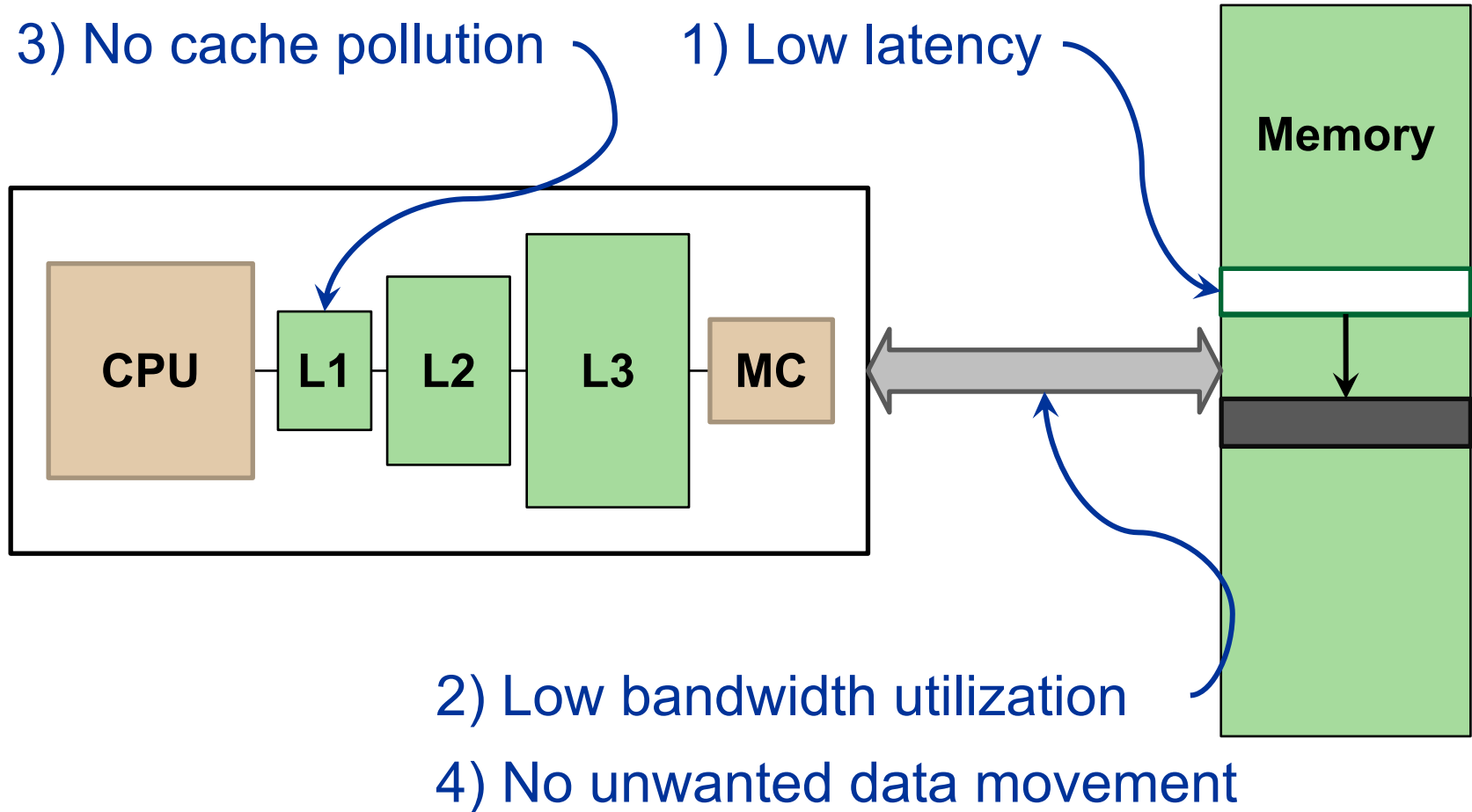
...
Many more

Today's Systems: Bulk Data Copy



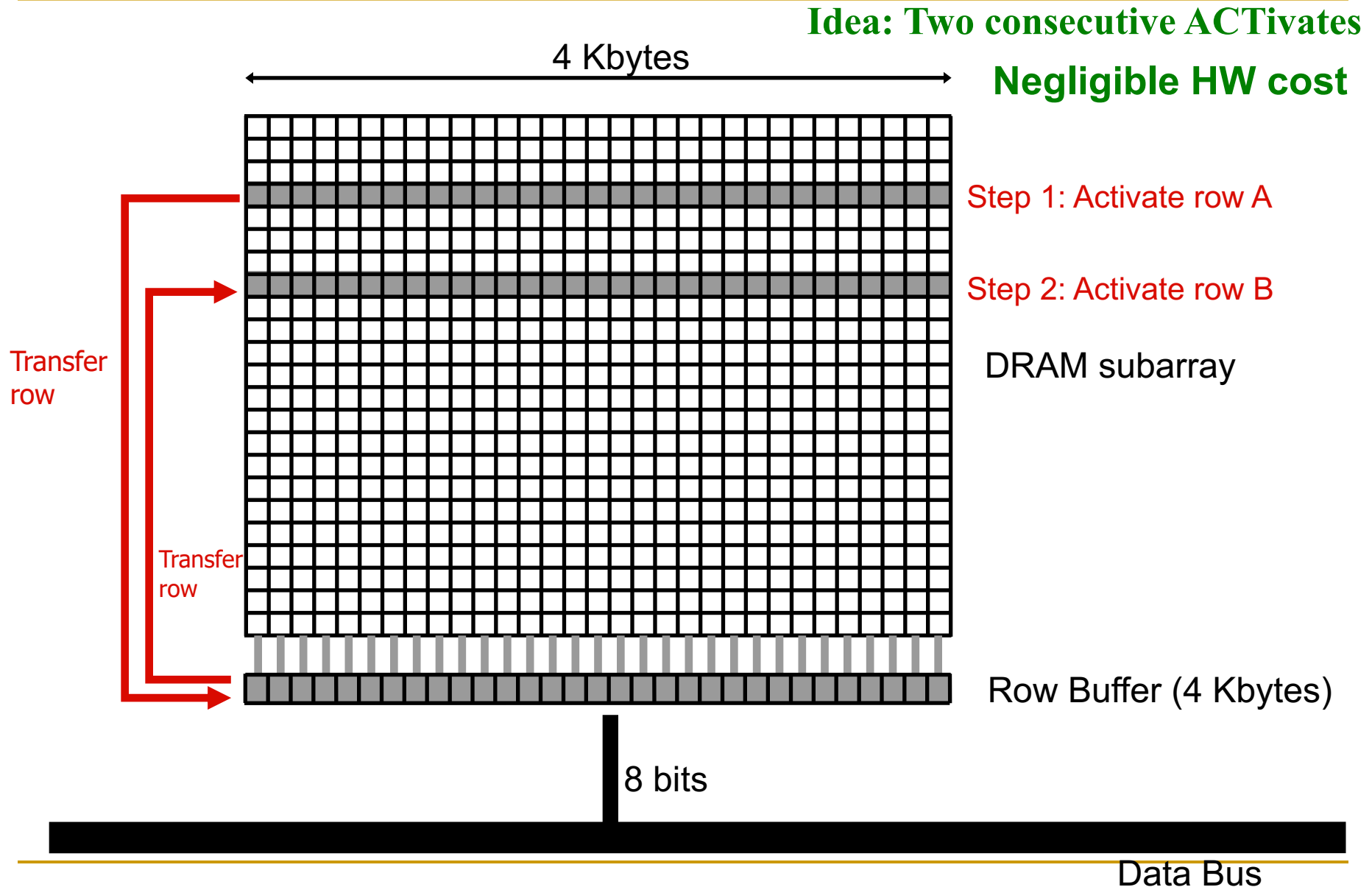
1046ns, 3.6uJ (for 4KB page copy via DMA)

Future Systems: In-Memory Copy

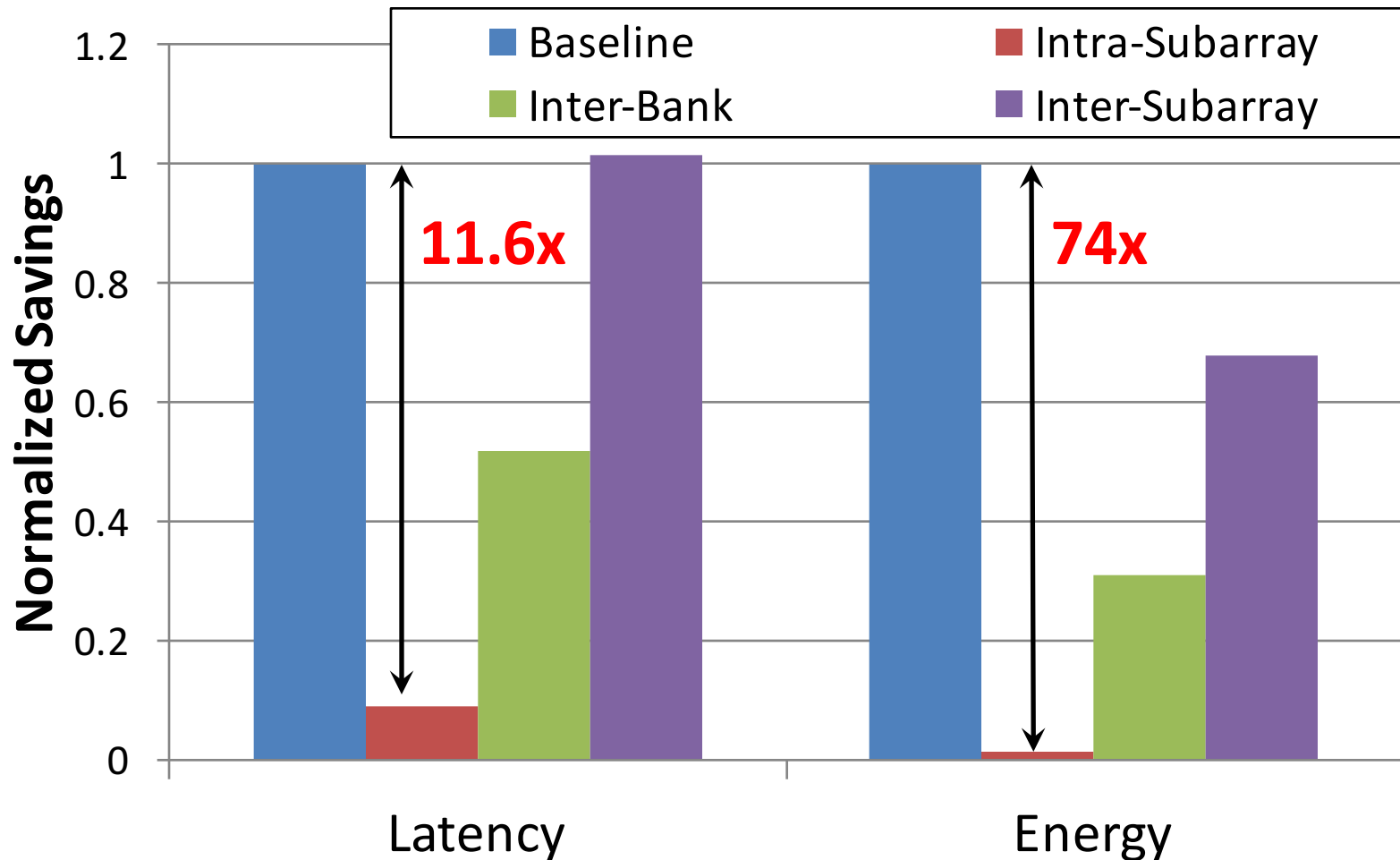


1046ns, 3.6uJ → 90ns, 0.04uJ

RowClone: In-DRAM Row Copy



RowClone: Latency and Energy Savings



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

More on RowClone

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

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

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

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

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

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

Carnegie Mellon University †Intel Pittsburgh

RowClone Demonstration 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

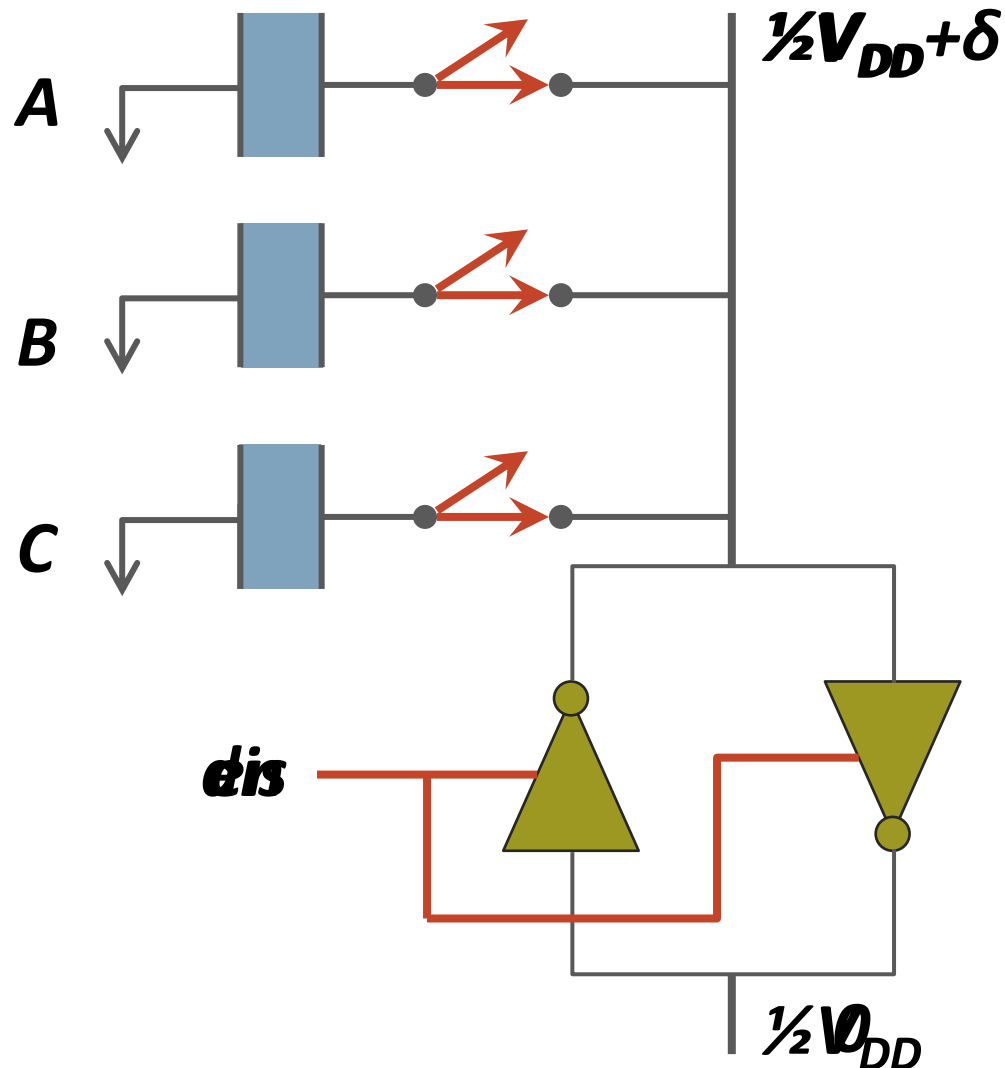
Department of Electrical Engineering
Princeton University

Ambit: In-Memory Bulk Bitwise Operations

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.

In-DRAM AND/OR: Triple Row Activation



Final State
 $AB + BC + AC$

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

In-DRAM Bulk Bitwise AND/OR Operation

- **BULKAND A, B → C**
 - Semantics: Perform a bitwise AND of two rows A and B and store the result in row C
 - R0 – reserved zero row, R1 – reserved one row
 - D1, D2, D3 – Designated rows for triple activation
-
1. RowClone A into D1
 2. RowClone B into D2
 3. RowClone R0 into D3
 4. ACTIVATE D1,D2,D3
 5. RowClone Result into C

More on In-DRAM Bulk AND/OR

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

Fast Bulk Bitwise AND and OR in DRAM

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

*Carnegie Mellon University †Intel Pittsburgh

In-DRAM NOT: Dual Contact Cell

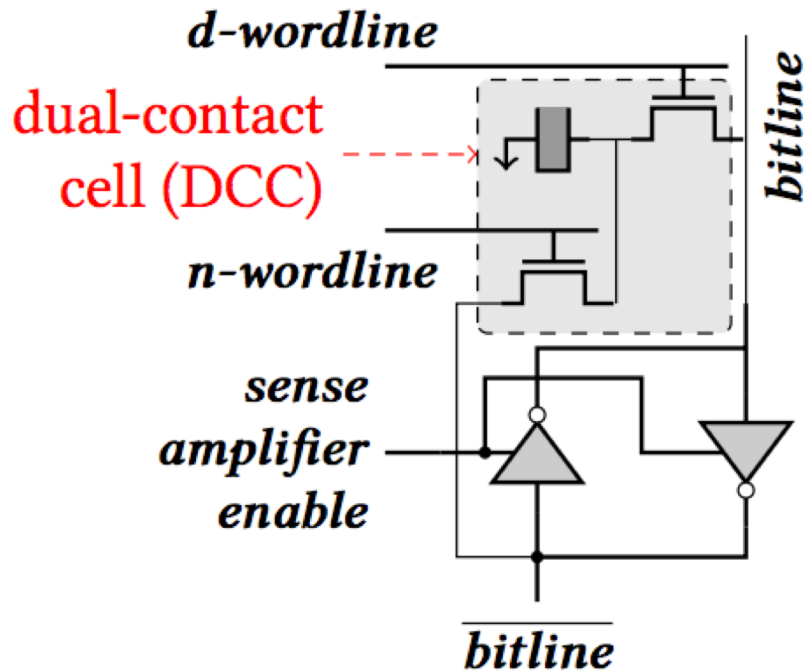


Figure 5: A dual-contact cell connected to both ends of a sense amplifier

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

In-DRAM NOT Operation

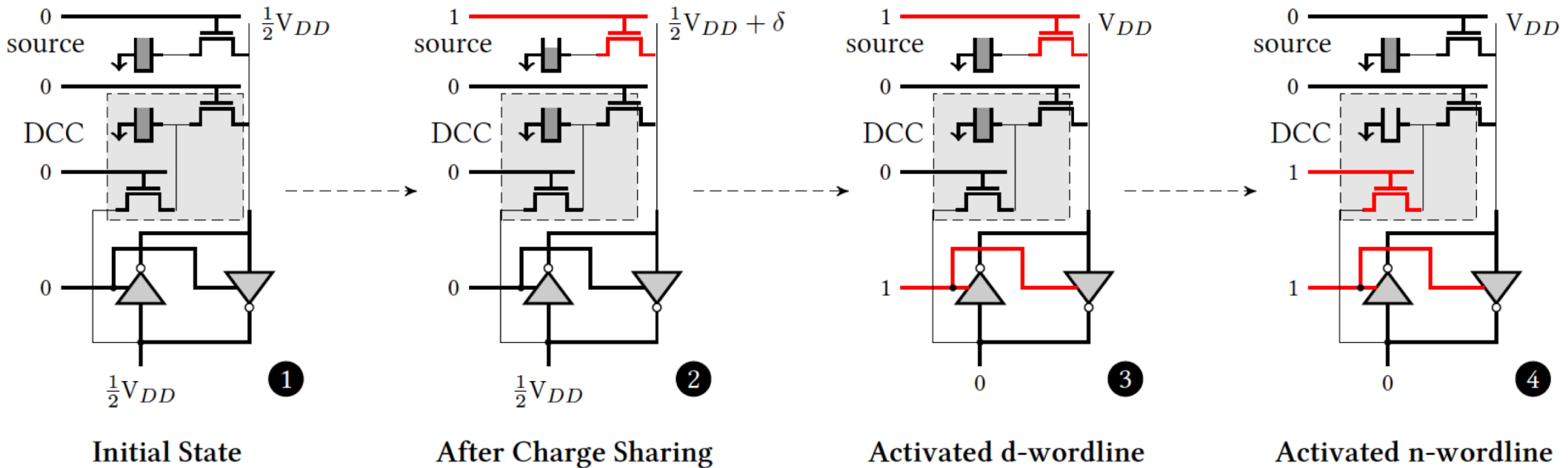


Figure 5: Bitwise NOT using a dual contact capacitor

Performance: In-DRAM Bitwise Operations

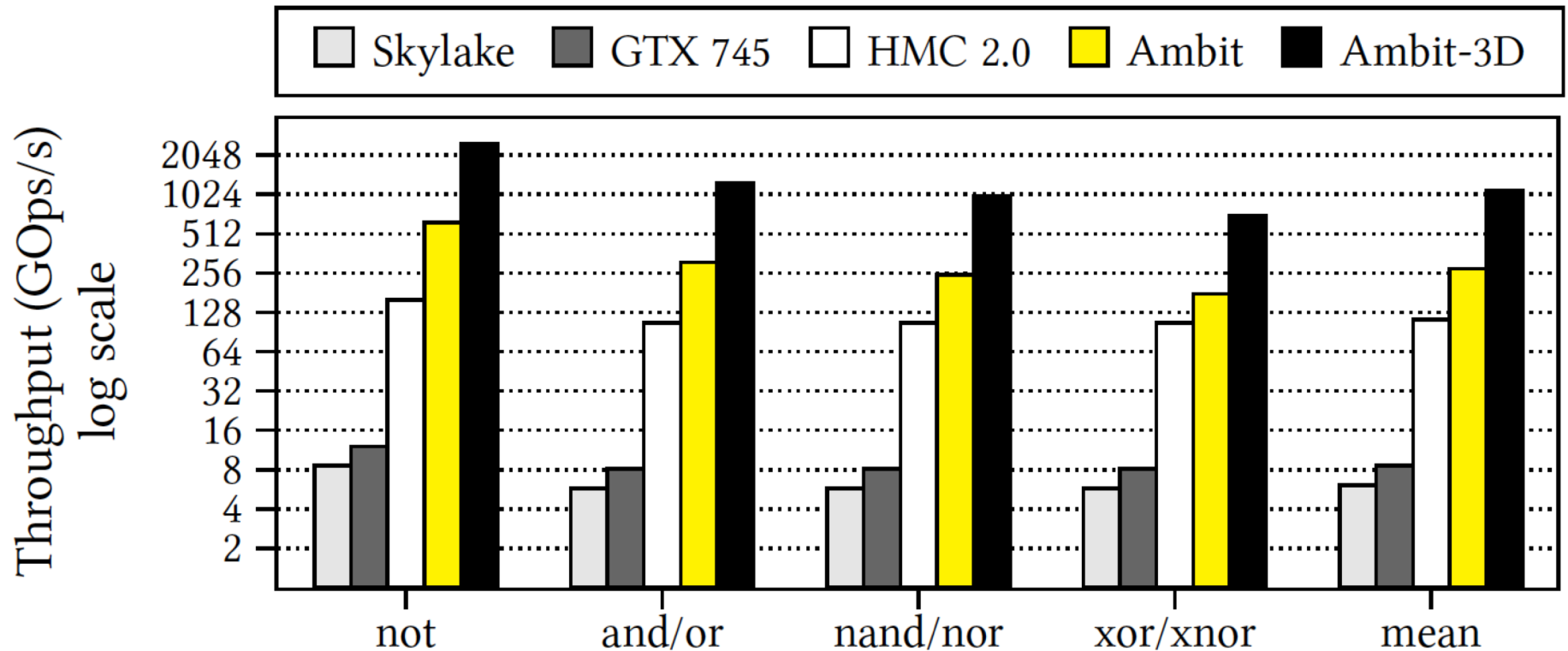


Figure 9: Throughput of bitwise operations on various systems.

Energy of In-DRAM Bitwise Operations

	Design	not	and/or	nand/nor	xor/xnor
DRAM & 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.

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

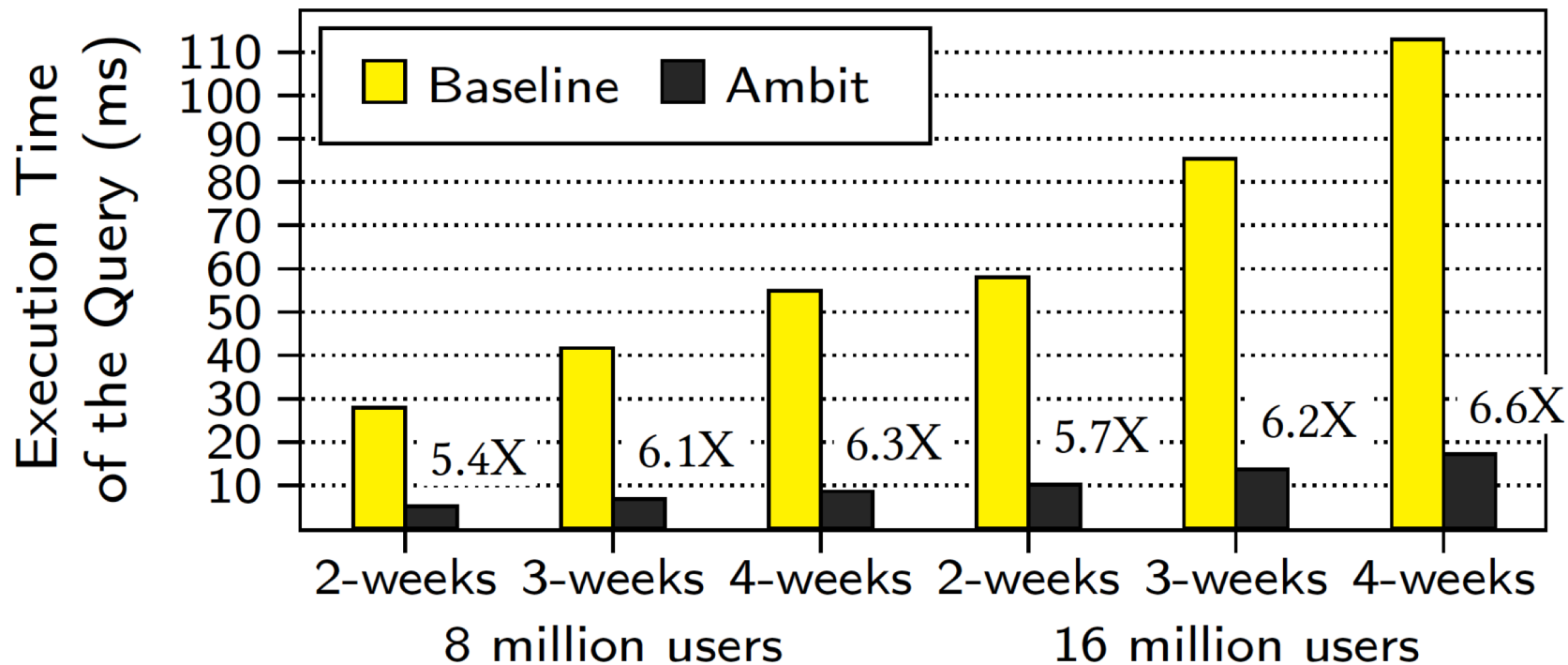


Figure 10: Bitmap index performance. The value above each bar indicates the reduction in execution time due to Ambit.

More on Ambit

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

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

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

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

SIMDRAM Framework

- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu, **"SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"** *Proceedings of the 26th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Virtual, March-April 2021.
[[2-page Extended Abstract](#)]
[[Short Talk Slides \(pptx\)](#) ([pdf](#))]
[[Talk Slides \(pptx\)](#) ([pdf](#))]
[[Short Talk Video](#) (5 mins)]
[[Full Talk Video](#) (27 mins)]

SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

*Nastaran Hajinazar^{1,2}

Nika Mansouri Ghiasi¹

*Geraldo F. Oliveira¹

Minesh Patel¹

Juan Gómez-Luna¹

Sven Gregorio¹

Mohammed Alser¹

Onur Mutlu¹

João Dinis Ferreira¹

Saugata Ghose³

¹ETH Zürich

²Simon Fraser University

³University of Illinois at Urbana–Champaign

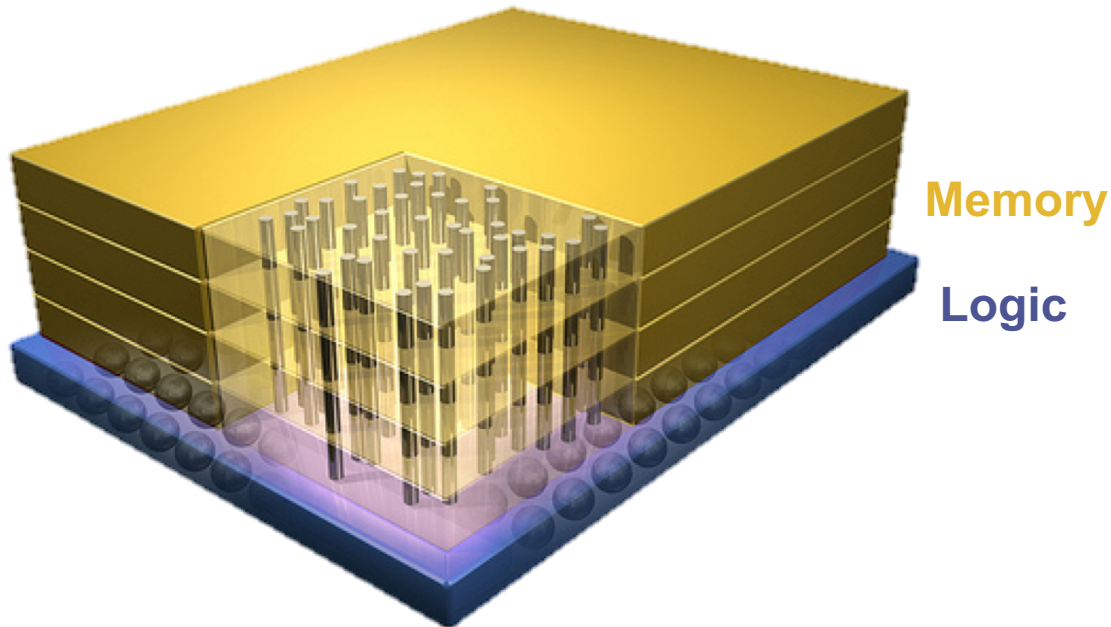
Agenda

- Major Trends Affecting Memory
- Processing in Memory: Two Directions
 - Minimally Changing Memory Chips
 - Exploiting 3D-Stacked Memory

Approach 2: 3D-Stacked Logic+Memory



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



Graph Processing

- Large graphs are everywhere (circa 2015)



36 Million
Wikipedia Pages



1.4 Billion
Facebook Users

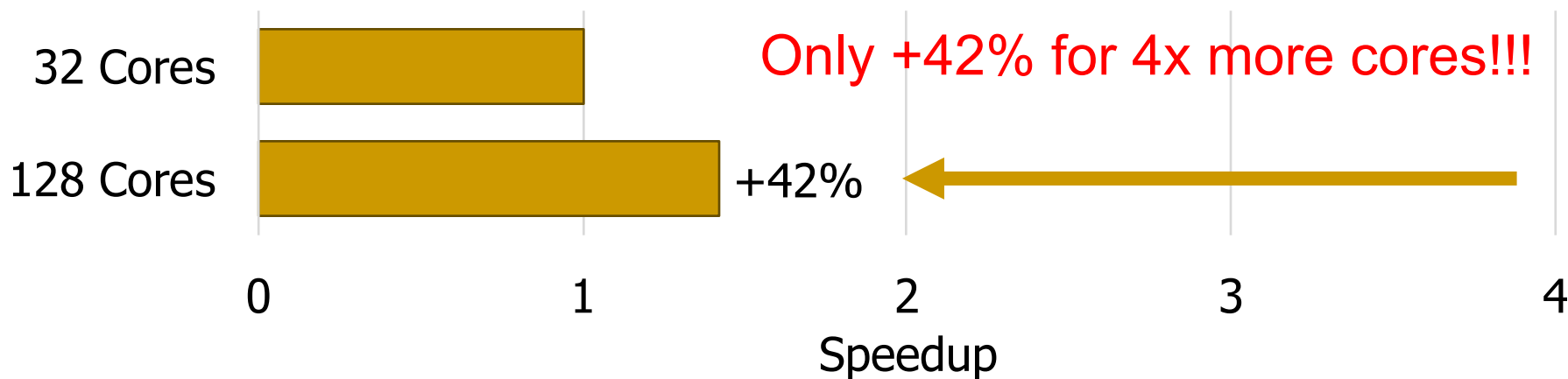


300 Million
Twitter Users



30 Billion
Instagram Photos

- Scalable large-scale graph processing is challenging

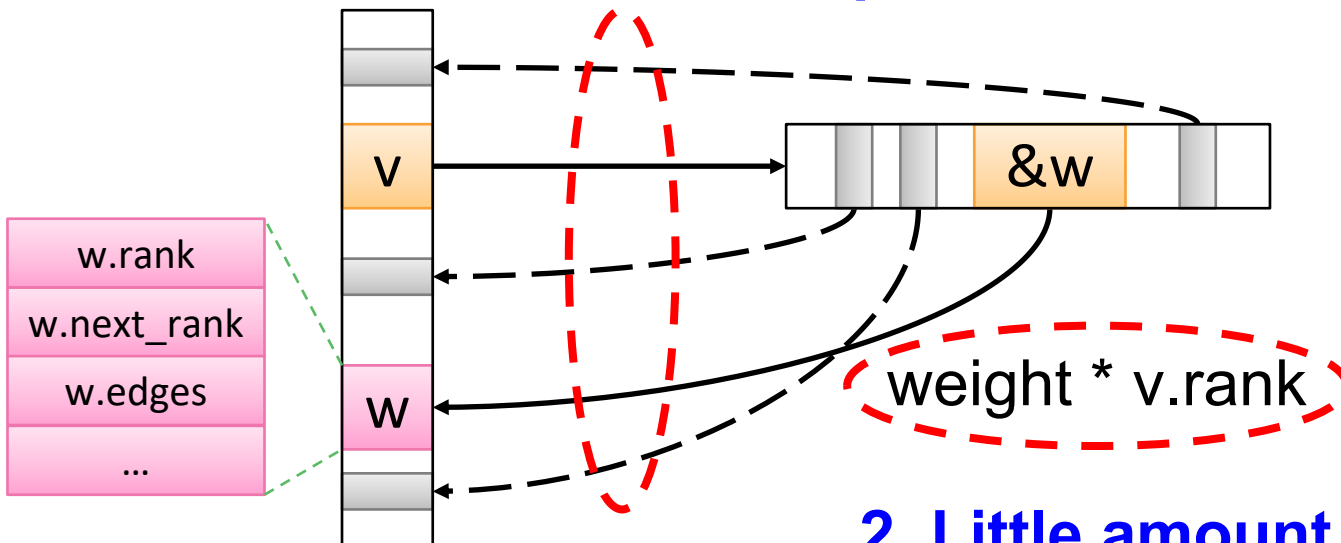


Key Bottlenecks in Graph Processing

PageRank algorithm (Page et al. 1999)

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

1. Frequent random memory accesses



2. Little amount of computation

Two Key Questions in 3D-Stacked PIM

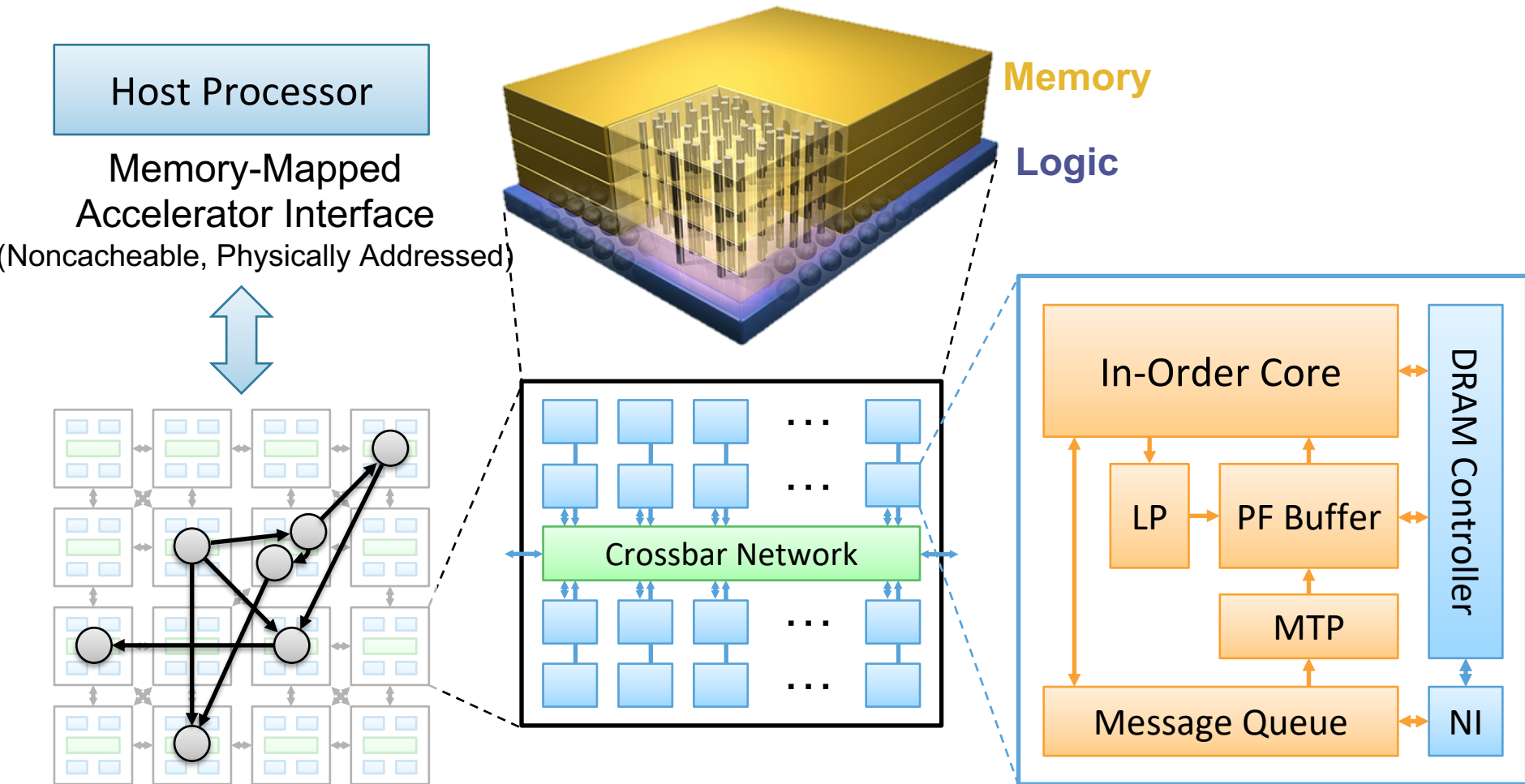
- How can we accelerate important applications if we use 3D-stacked memory as a coarse-grained accelerator?
 - what is the architecture and programming model?
 - what are the mechanisms for acceleration?

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

Tesseract: An In-Memory Accelerator for Graph Processing

Tesseract System for Graph Processing

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

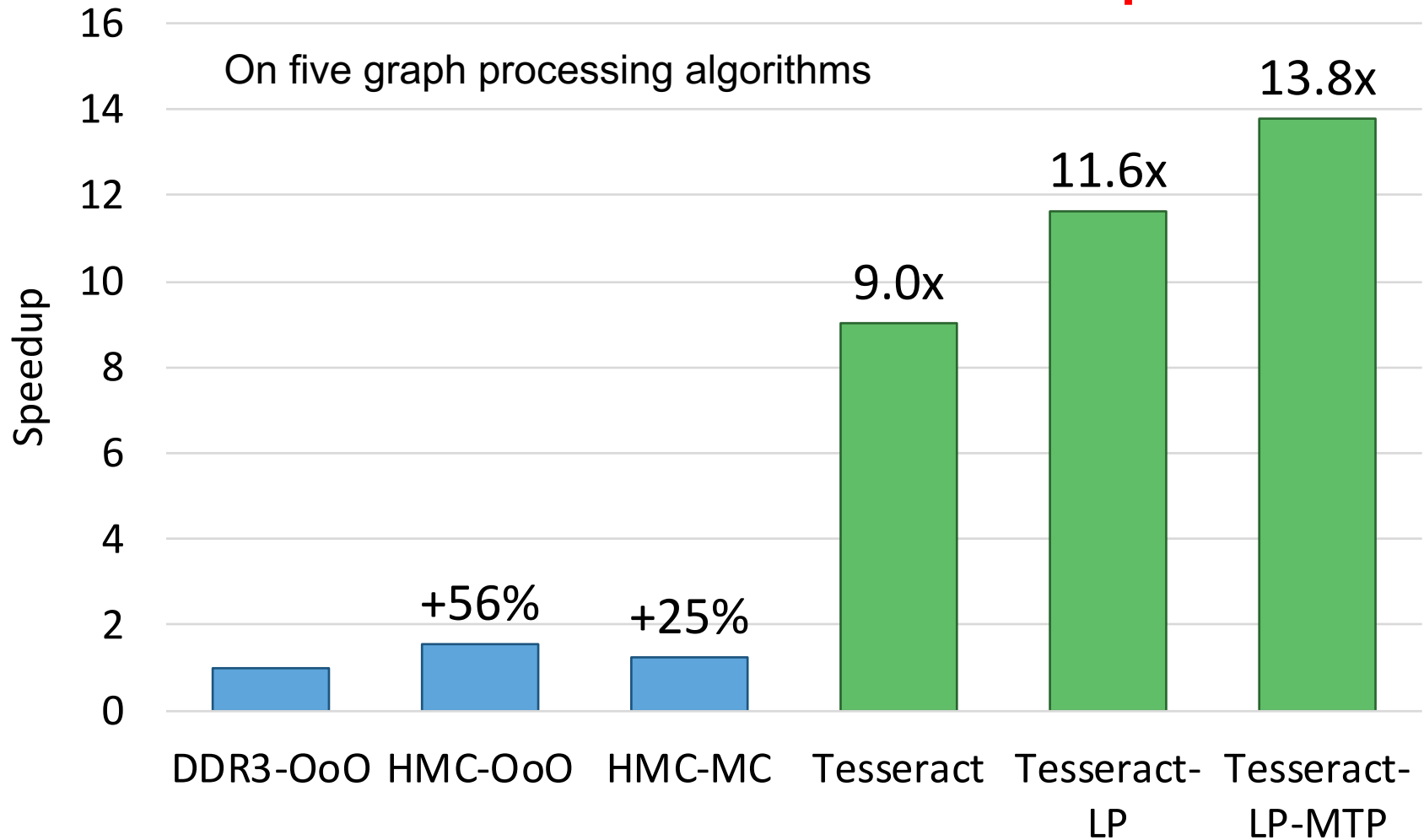


Tesseract System for Graph Processing

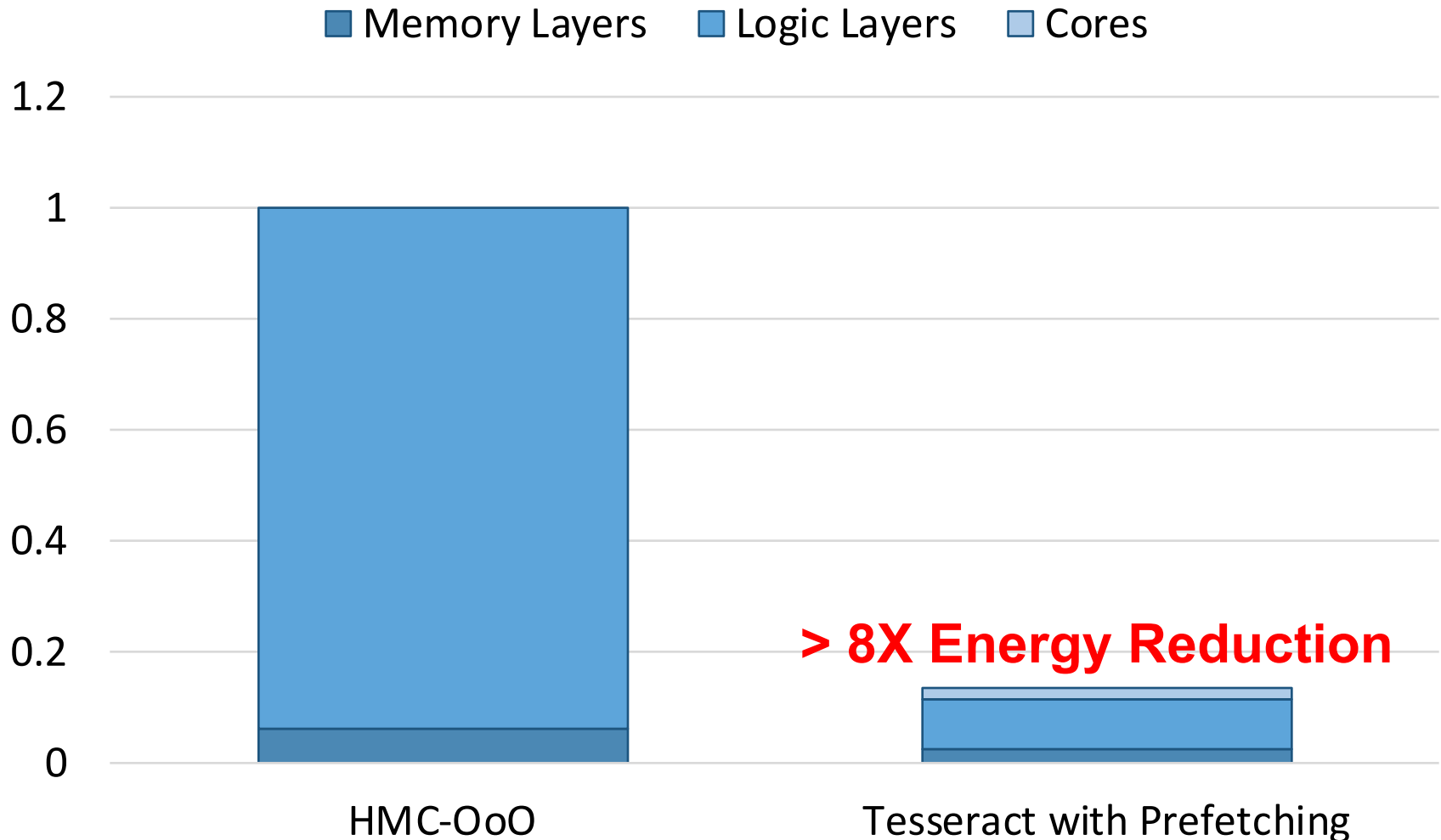
- Evaluation on
 - DDR3 DRAM, computation on Out-of-Order (OoO) core
 - Hybrid Memory Cube (HMC) DRAM, computation on Out-of-Order (OoO) core
 - HMC DRAM, computation on the Memory Controller (MC)
 - Tesseract
 - With or without List Prefetching (LP)
 - With or without Message Triggered Prefetching (MTP), specified by the programmer

Tesseract Graph Processing Performance

>13X Performance Improvement



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

Seoul National University [§]Oracle Labs [†]Carnegie Mellon University

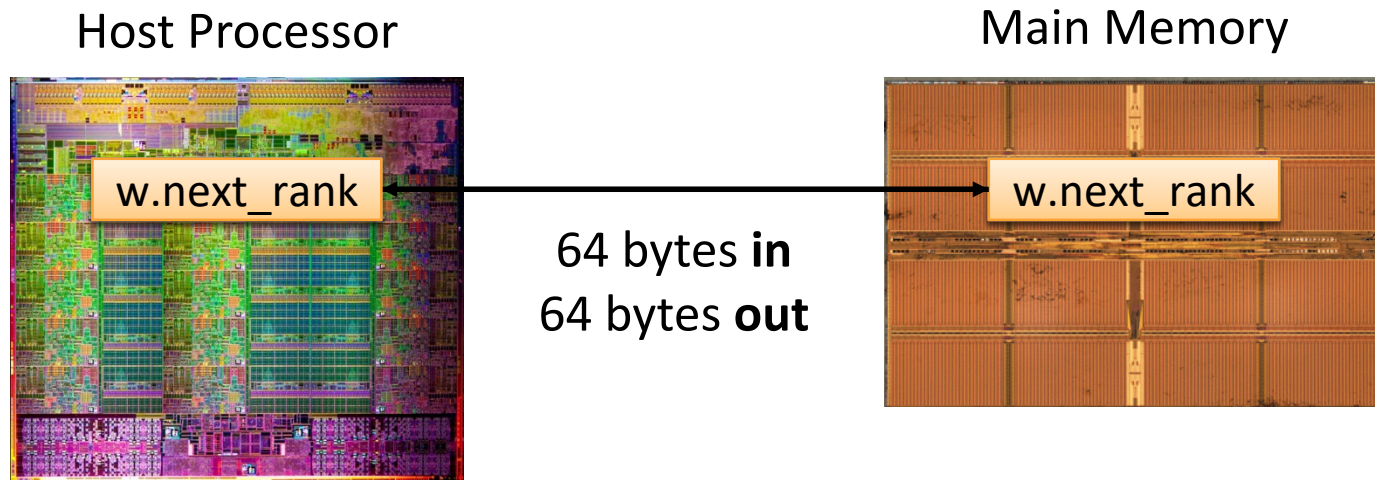
Two Key Questions in 3D-Stacked PIM

- 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 for Graph Processing

Simple PIM Operations as ISA Extensions (I)

```
for (v: graph.vertices) { PageRank algorithm (Page et al. 1999)
    value = weight * v.rank;
    for (w: v.successors) {
        w.next_rank += value;
    }
}
```

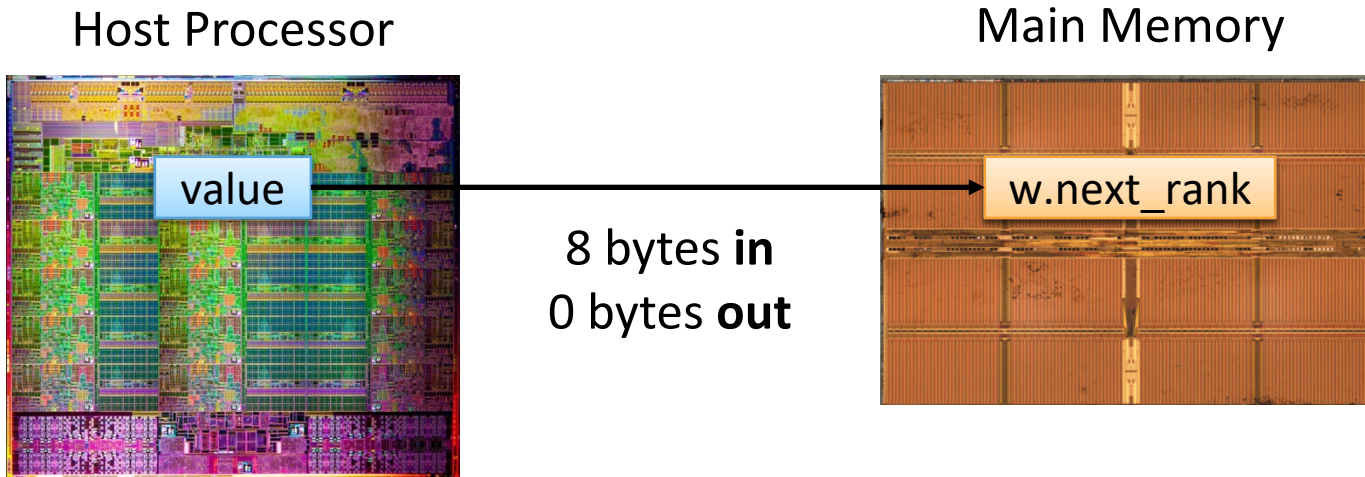


Conventional Architecture

Simple PIM Operations as ISA Extensions (II)

```
for (v: graph.vertices) { PageRank algorithm (Page et al. 1999)
  value = weight * v.rank;
  for (w: v.successors) {
    __pim_add(&w.next_rank, value);
  }
}
```

pim.add r1, (r2)



In-Memory Addition

PEI: Benchmarks

- Graph processing
 - Average Teenage Follower (AT)
 - Breadth-First Search (BFS)
 - PageRank (PR)
 - Single-Source Shortest Path (SP)
 - Weakly Connected Components (WCC)
- Other benchmarks that can benefit from PEI
 - Data analytics
 - Hash Join (HJ)
 - Histogram (HG)
 - Radix Partitioning (RP)
 - Machine learning and data mining
 - Streamcluster (SC)
 - Support Vector Machine (SVM)

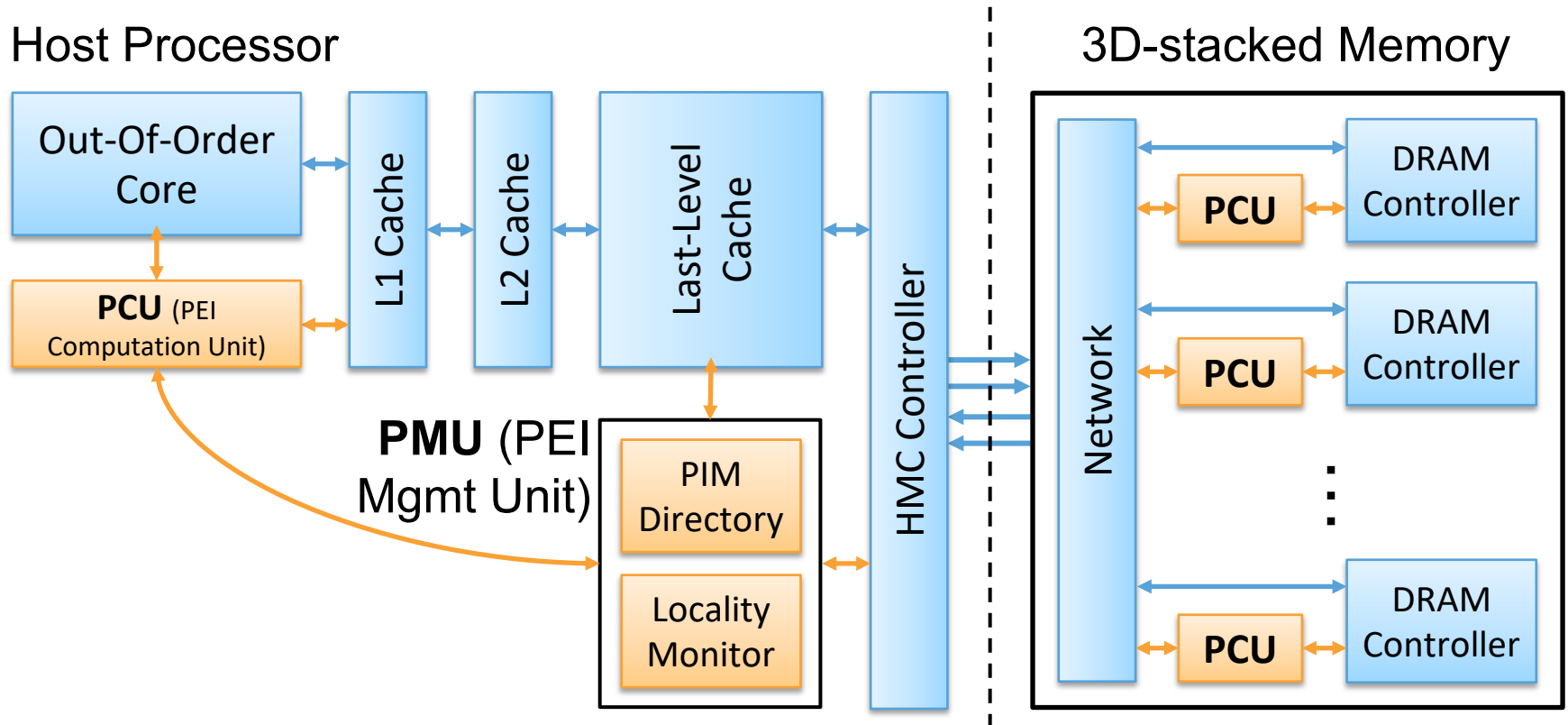
PEI: PIM-Enabled Instructions: Examples

Table 1: Summary of Supported PIM Operations

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

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

Example PEI Microarchitecture



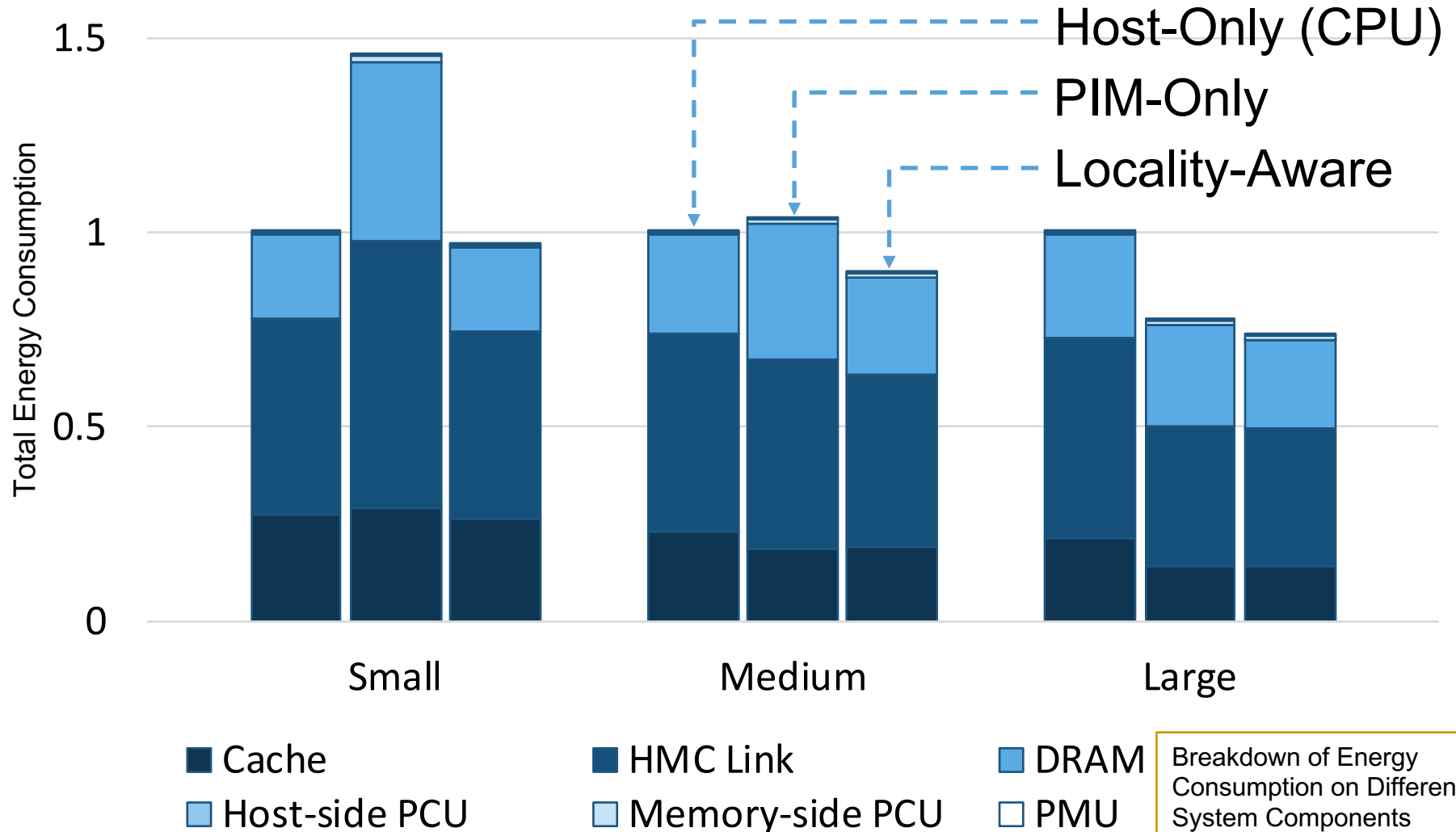
Example PEI uArchitecture

PEI Performance Delta: Large Data Sets

(Large Inputs, Baseline: CPU-Only)



PEI Energy Consumption



More on 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[†] Kiyoungh Choi

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

Seoul National University

[†]Carnegie Mellon University

Agenda

- Major Trends Affecting Memory
- Processing in Memory: Two Directions
 - Minimally Changing Memory Chips
 - Exploiting 3D-Stacked Memory

Eliminating the Adoption Barriers

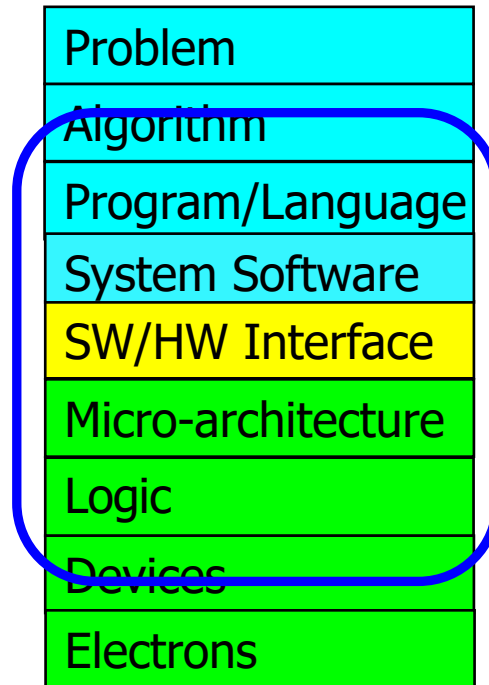
How to Enable Adoption
of Processing in Memory

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

A Modern Primer on Processing in Memory

Onur Mutlu^{a,b}, Saugata Ghose^{b,c}, Juan Gómez-Luna^a, Rachata Ausavarungnirun^d

SAFARI Research Group

^a*ETH Zürich*

^b*Carnegie Mellon University*

^c*University of Illinois at Urbana-Champaign*

^d*King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,

"A Modern Primer on Processing in Memory"

*Invited Book Chapter in **Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann**, Springer, to be published in 2021.*

P&S Processing-in-Memory

Exploring the Processing-in-Memory Paradigm
for Future Computing Systems

Dr. Juan Gómez Luna

Prof. Onur Mutlu

ETH Zürich

Fall 2021

5 October 2021