P&S Processing-in-Memory

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

Dr. Juan Gómez Luna

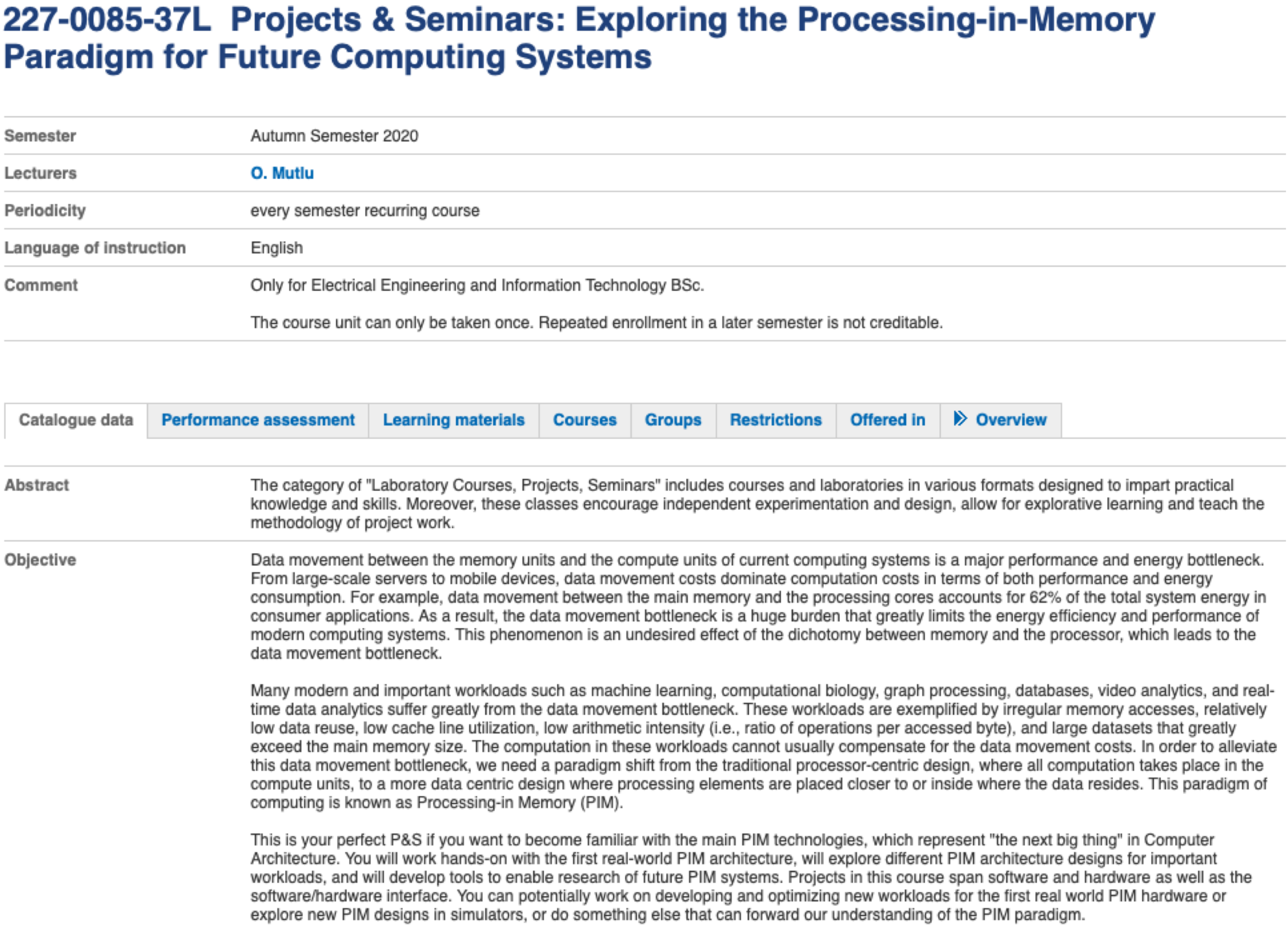
Prof. Onur Mutlu

ETH Zürich

Fall 2020

1 October 2020

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



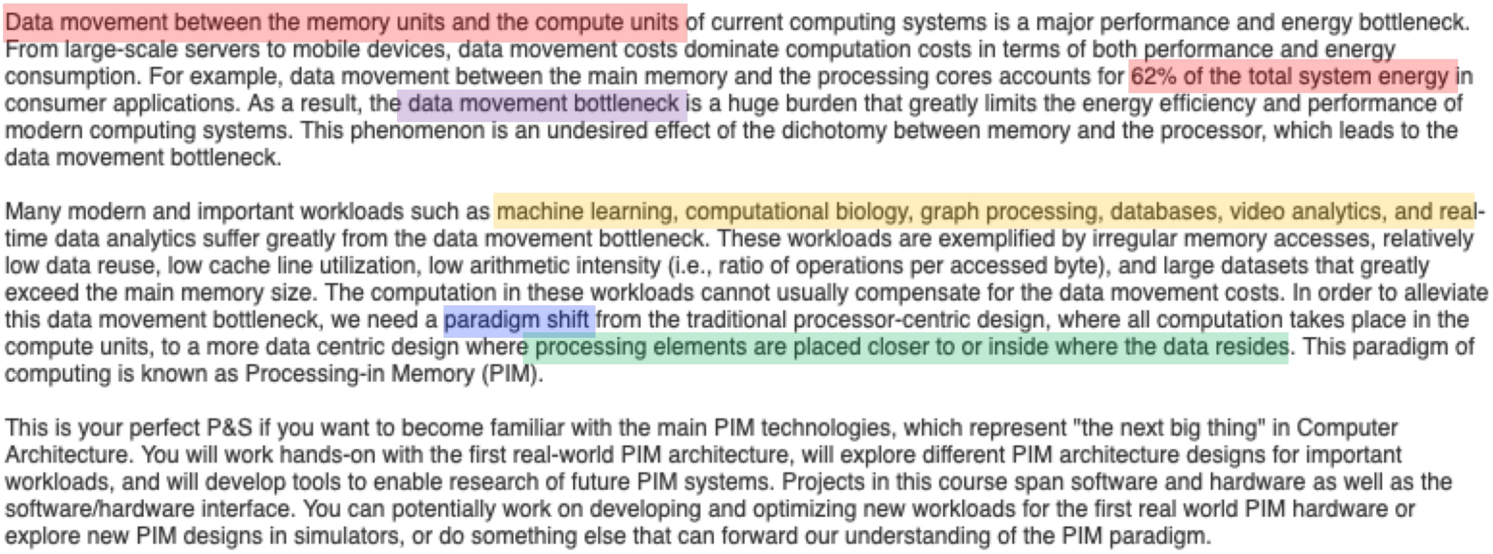
227-0085-37L Projects & Seminars: Exploring the Processing-in-Memory Paradigm for Future Computing Systems

Semester	Autumn Semester 2020
Lecturers	O. Mutlu
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.

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	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.

http://www.vvz.ethz.ch/Vorlesungsverzeichnis/lerneinheit.view?lerneinheitId=151838&semkez=2020W&ansicht=KATALOGDATEN&lang=en

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



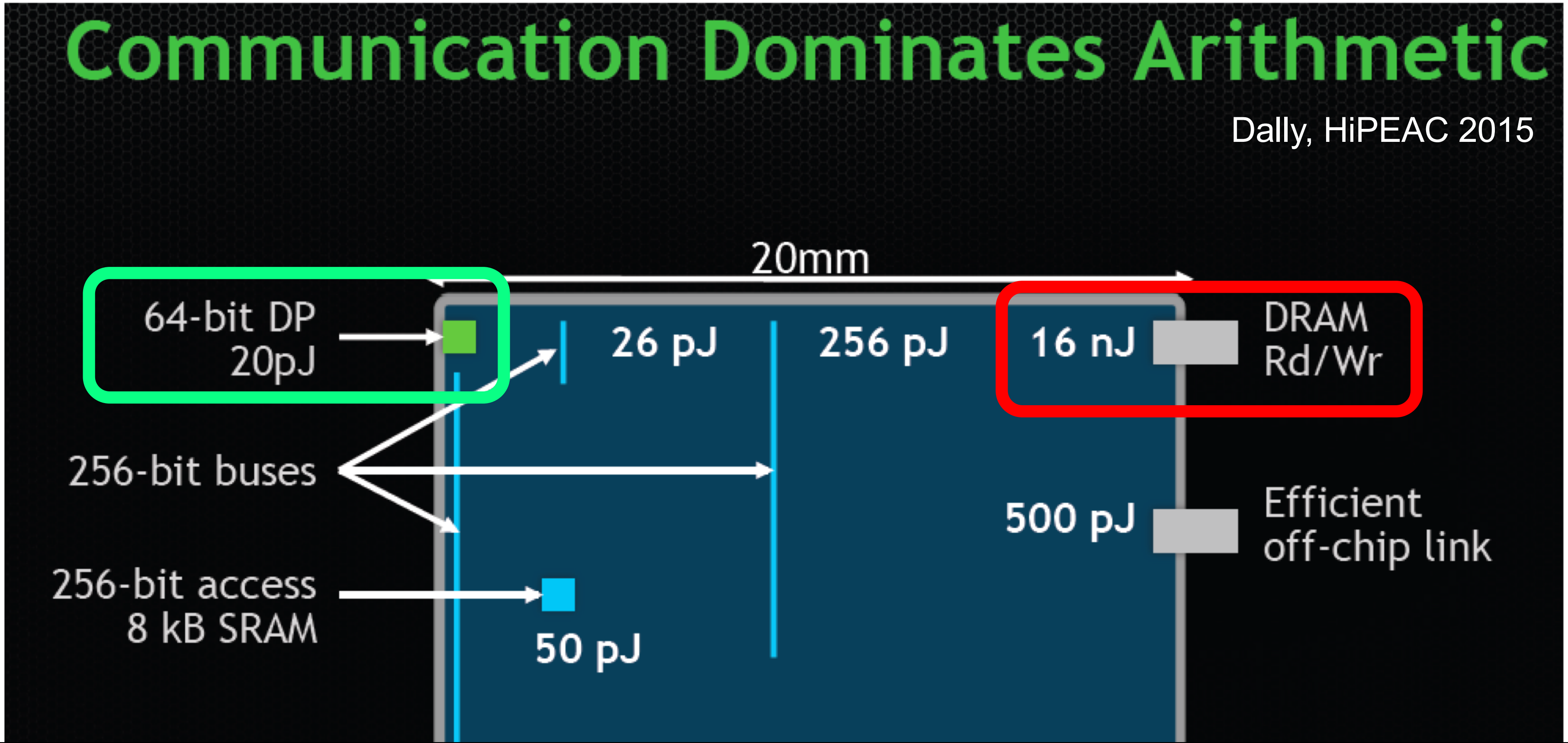
Data movement between the memory units and the compute units of current computing systems is a major performance and energy bottleneck. From large-scale servers to mobile devices, data movement costs dominate computation costs in terms of both performance and energy consumption. For example, data movement 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.

http://www.vvz.ethz.ch/Vorlesungsverzeichnis/lerneinheit.view?lerneinheitId=151838&semkez=2020W&ansicht=KATALOGDATEN&lang=en

Data Movement vs. Computation Energy



Communication Dominates Arithmetic

Dally, HiPEAC 2015

A memory access consumes ~1000X 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

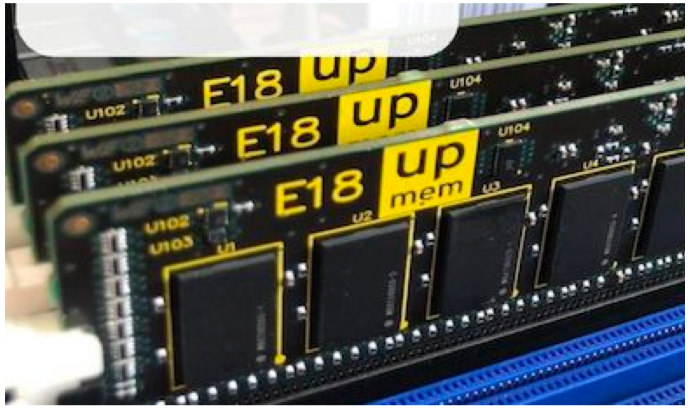
- You will learn what are key workload characteristics that make them more prone to the data movement bottleneck

- You will review traditional approaches to alleviating data movement and will get familiar with new research proposals: processing-in-memory solutions

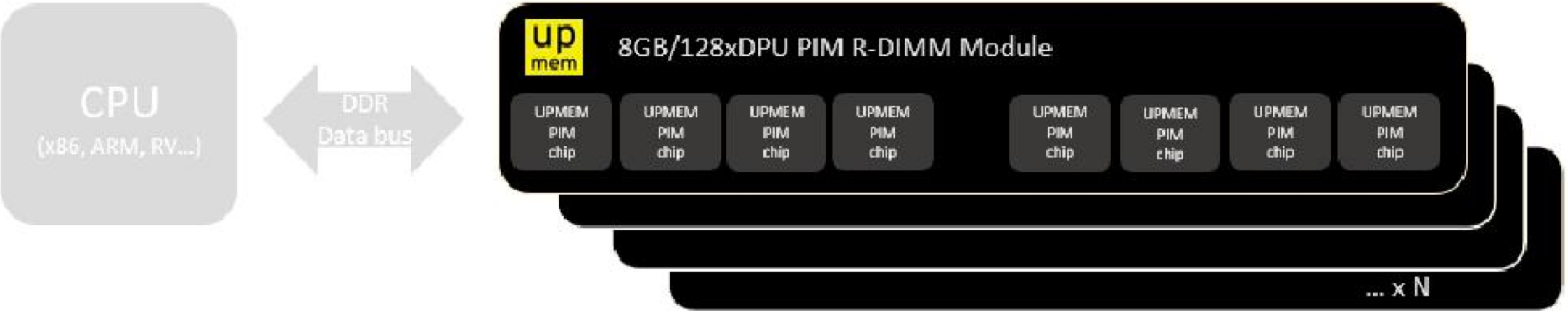
- You will work hands-on: analyzing workloads, programming PIM architectures, simulating new PIM proposals, etc.

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



https://www.anandtech.com/show/14750/hot-chips-31-analysis-inmemory-processing-by-upmem
https://www.upmem.com/video-upmem-presenting-its-true-processing-in-memory-solution-hot-chips-2019/

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)

- 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

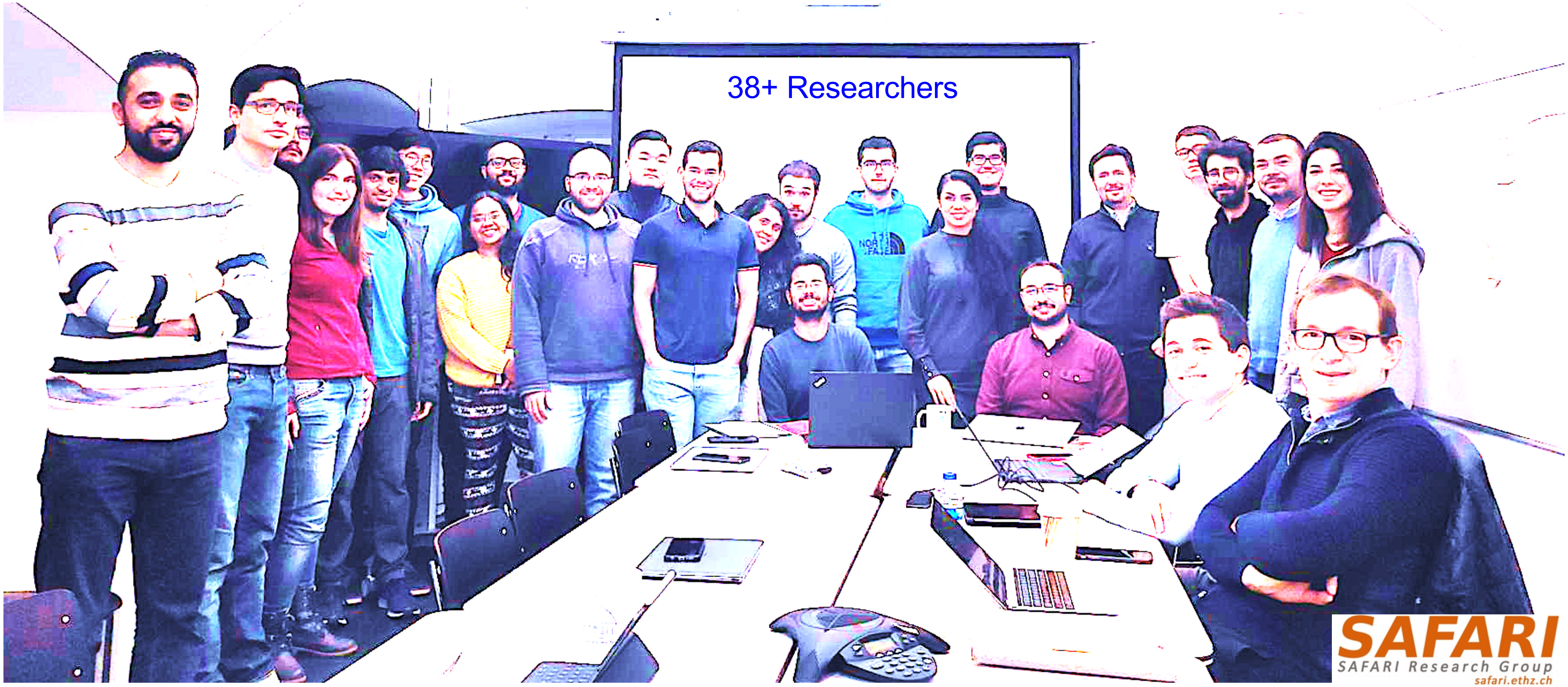
- Supervisors:
  - Dr. Haiyu Mao
  - Geraldo F. de Oliveira
  - Konstantinos Kanellopoulos
  - Nika Mansouri Ghiasi

- 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/



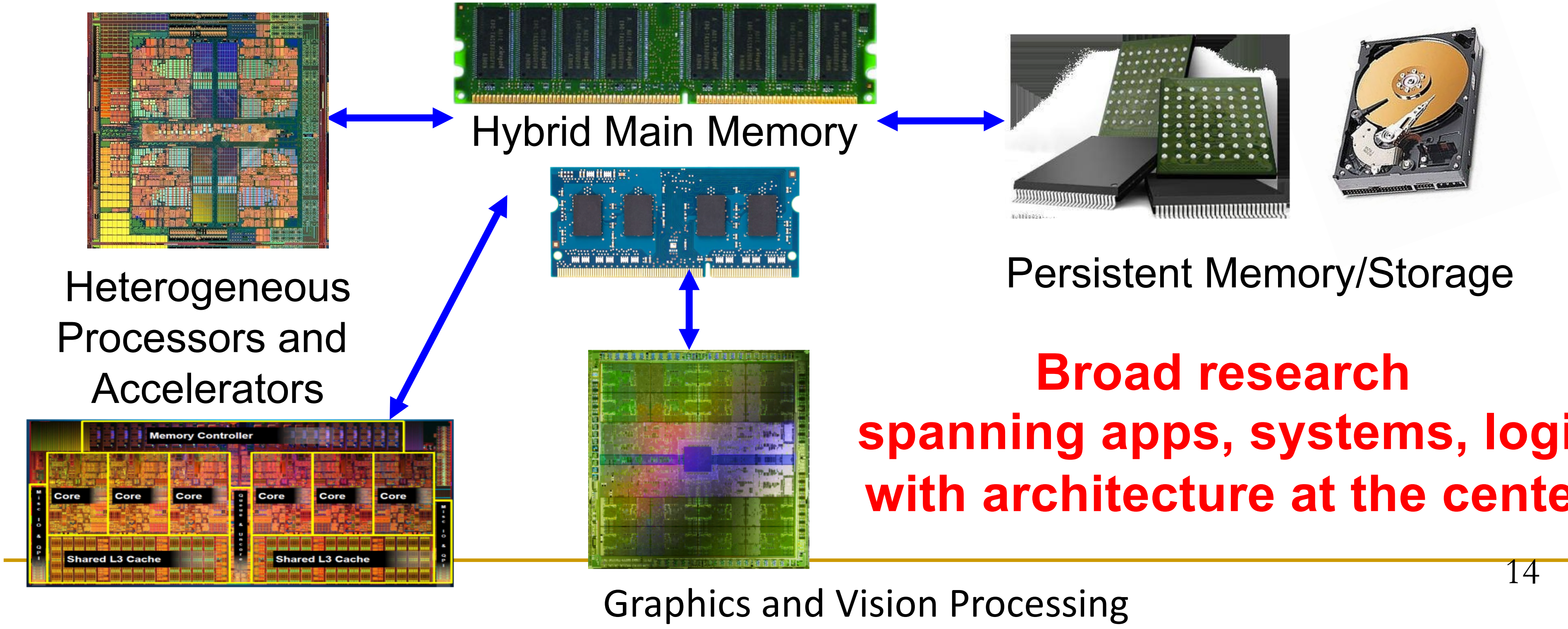
Think BIG, Aim HIGH!

https://safari.ethz.ch

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***



Hybrid Main Memory

Heterogeneous Processors and Accelerators

Persistent Memory/Storage

Graphics and Vision Processing

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

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 Piazza for Q&A

Meeting 1

- Required materials:

1. 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)]

2. Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,
"Processing Data Where It Makes Sense: Enabling In-Memory Computation"
*Invited paper in Microprocessors and Microsystems (**MICPRO**)*, June 2019.
[arXiv version]
[Slides (pptx)]
[Talk Video]

- 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, to appear in November 2019.
[Preliminary arXiv version]

4. Computation in Memory (Professor Onur Mutlu, lecture, Fall 2019).
(PDF) (PPT)
Video
Video (ETHZ)

5. Computation in Memory II (Professor Onur Mutlu, lecture, Fall 2019).
(PDF) (PPT)
Video
Video (ETHZ)

6. Computation in Memory III (Professor Onur Mutlu, lecture, Fall 2019).
(PDF) (PPT)
Video
Video (ETHZ)

Meeting 2 (October 8th)

- 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

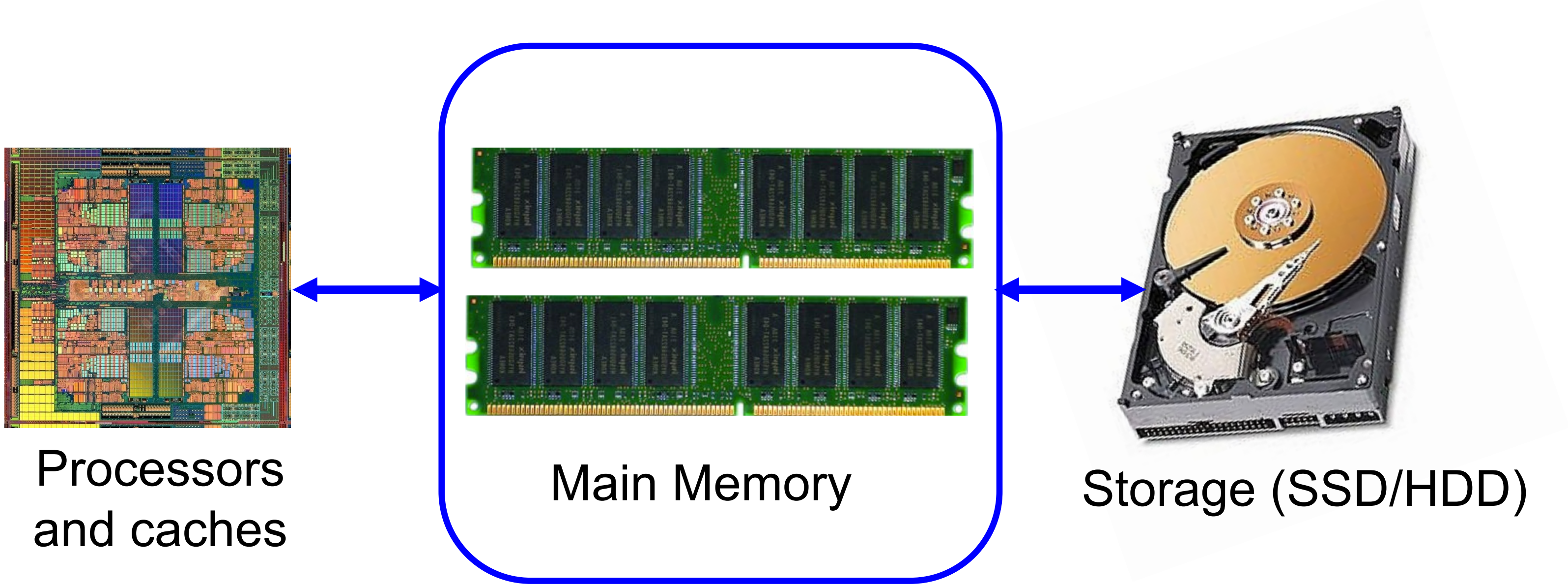
- Individual meetings with your mentor/s

- Tutorials and short talks
  - PIM programming
  - Recent research works

- Presentation of your work

An Introduction to Processing-in-Memory

The Main Memory System



Processors and caches

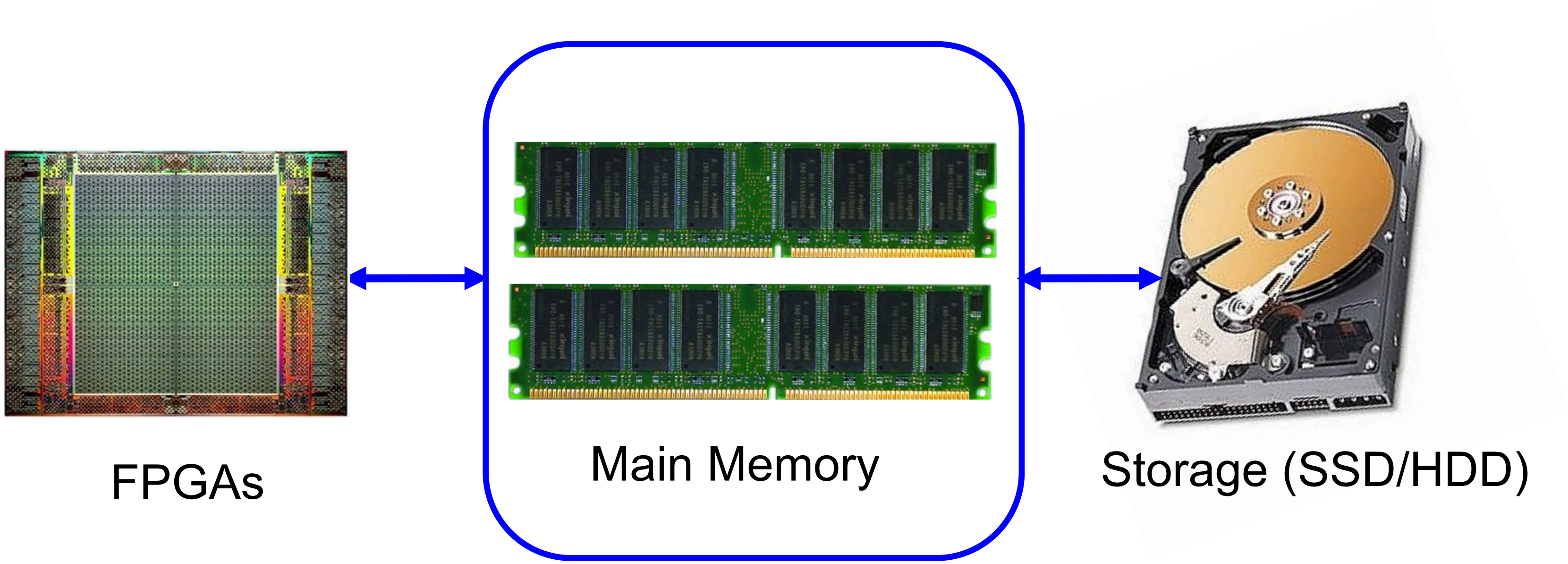
Main Memory

Storage (SSD/HDD)

- 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



FPGAs

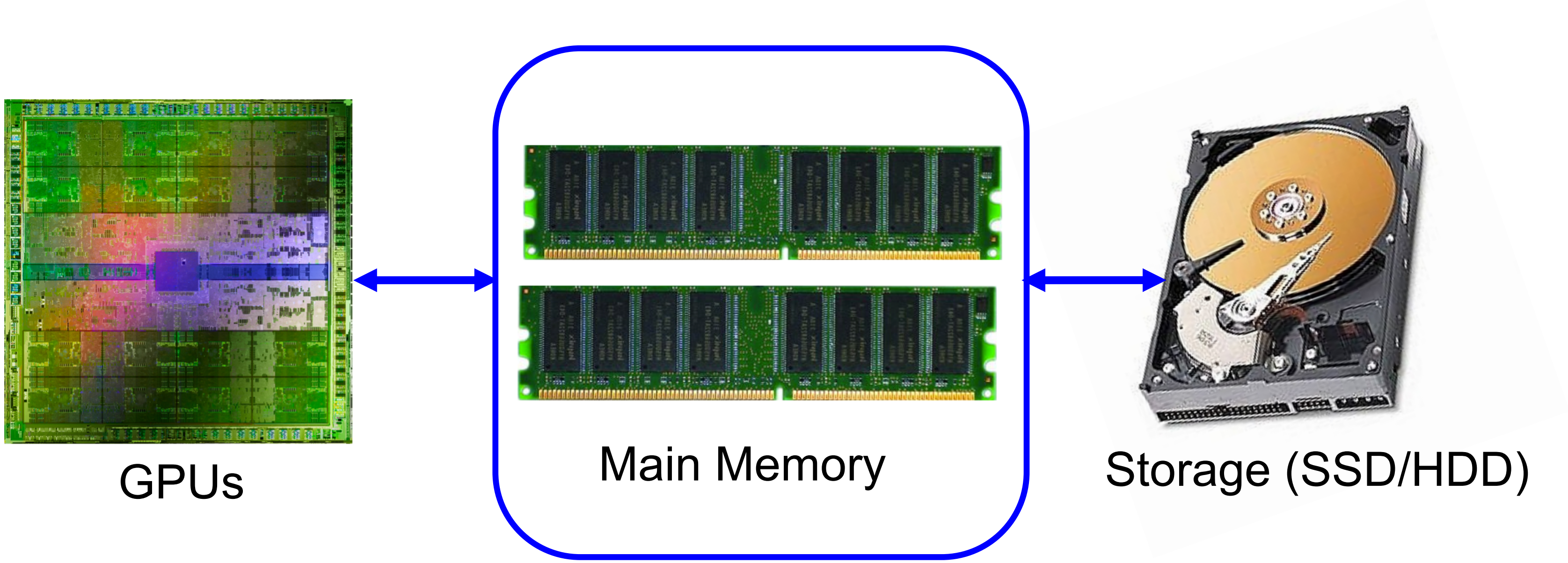
Main Memory

Storage (SSD/HDD)

- 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

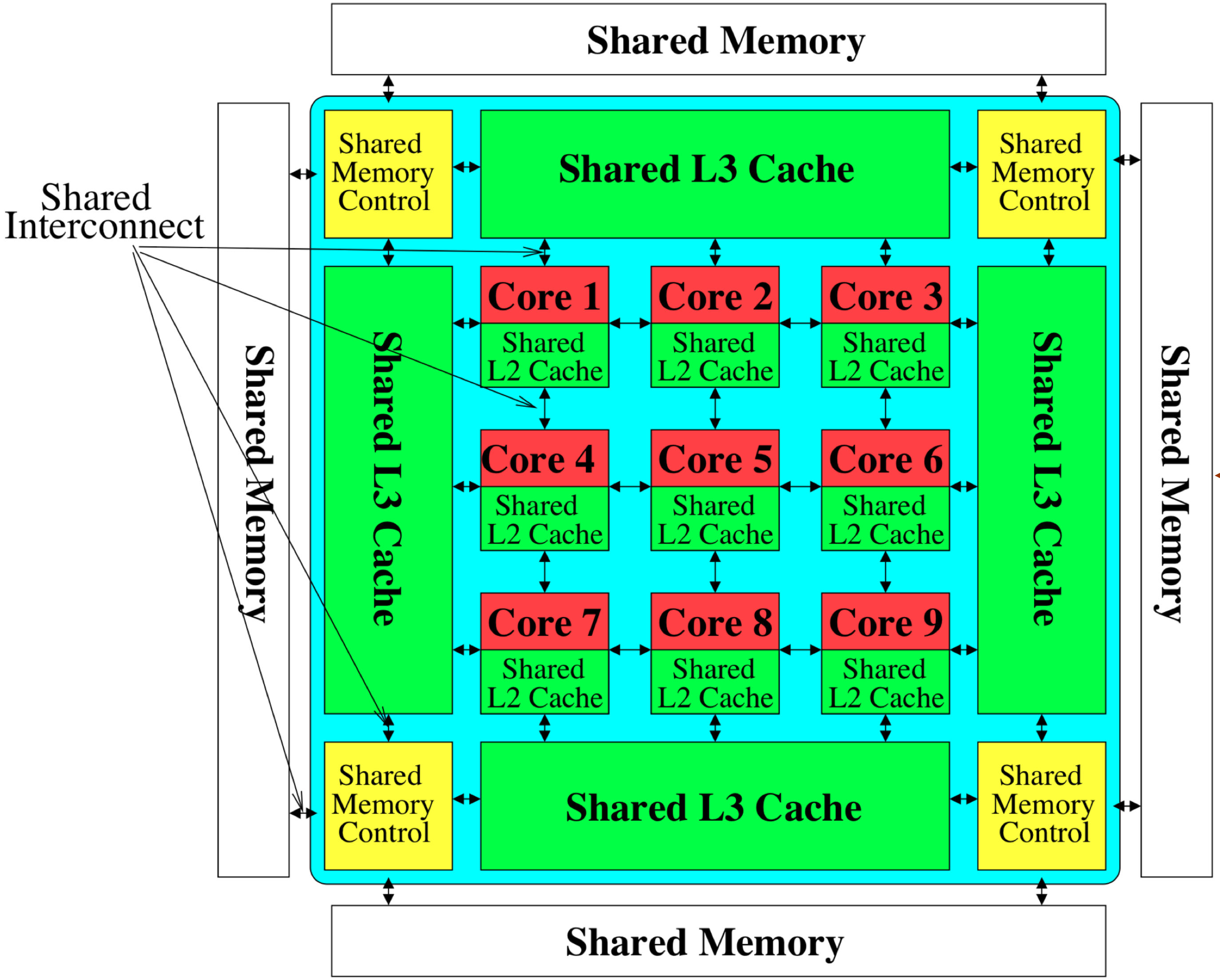
The Main Memory System

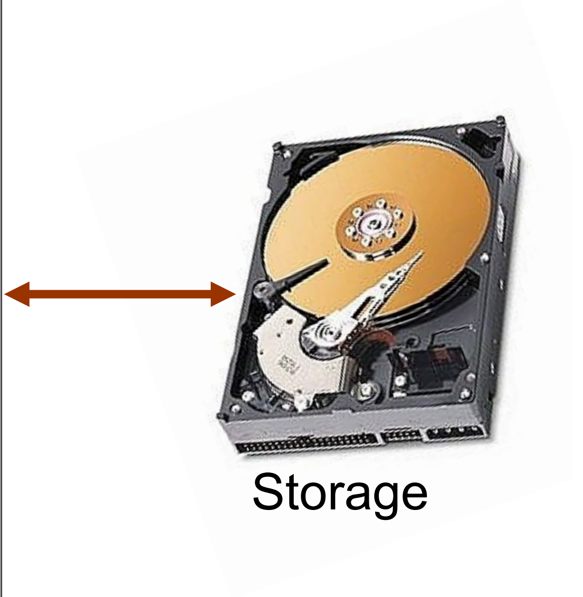


- Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor

- 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

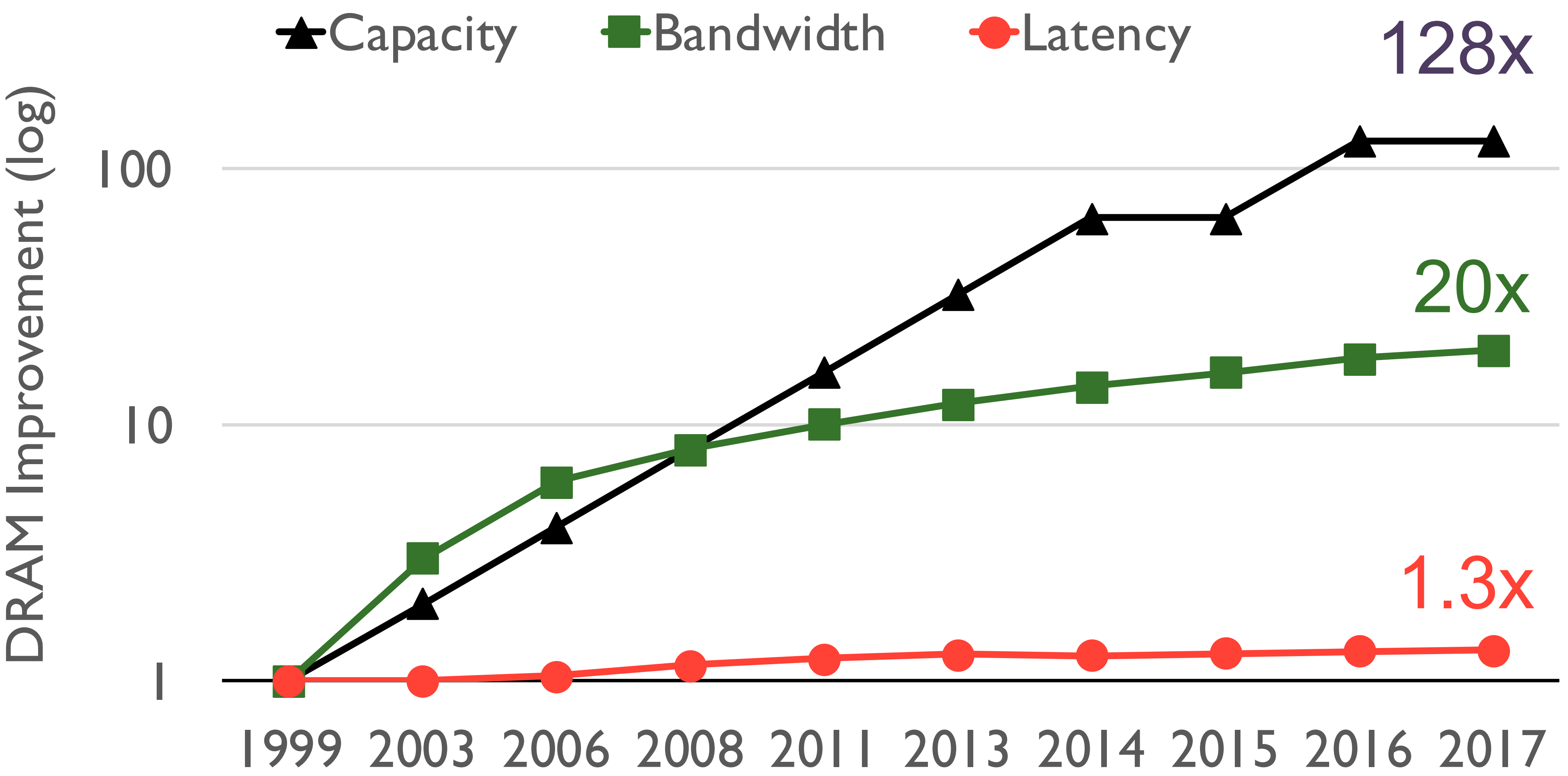
Three Key Systems Trends

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]

Graph/Tree Processing

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

In-Memory Data Analytics

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

Datacenter Workloads

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

DRAM Latency Is Critical for Performance

In-memory Databases

Graph/Tree Processing

Long memory latency → performance bottleneck

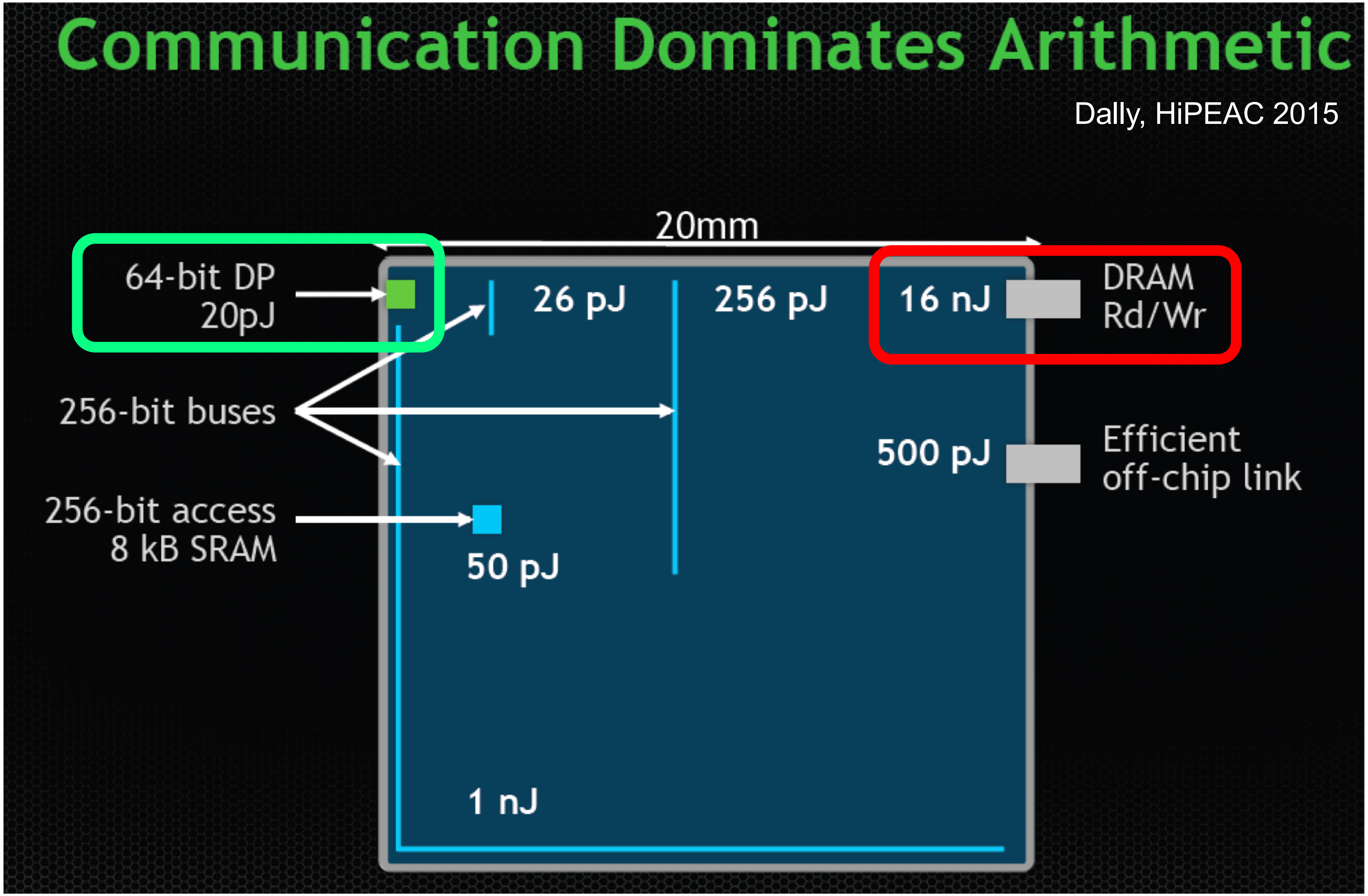
In-Memory Data Analytics

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

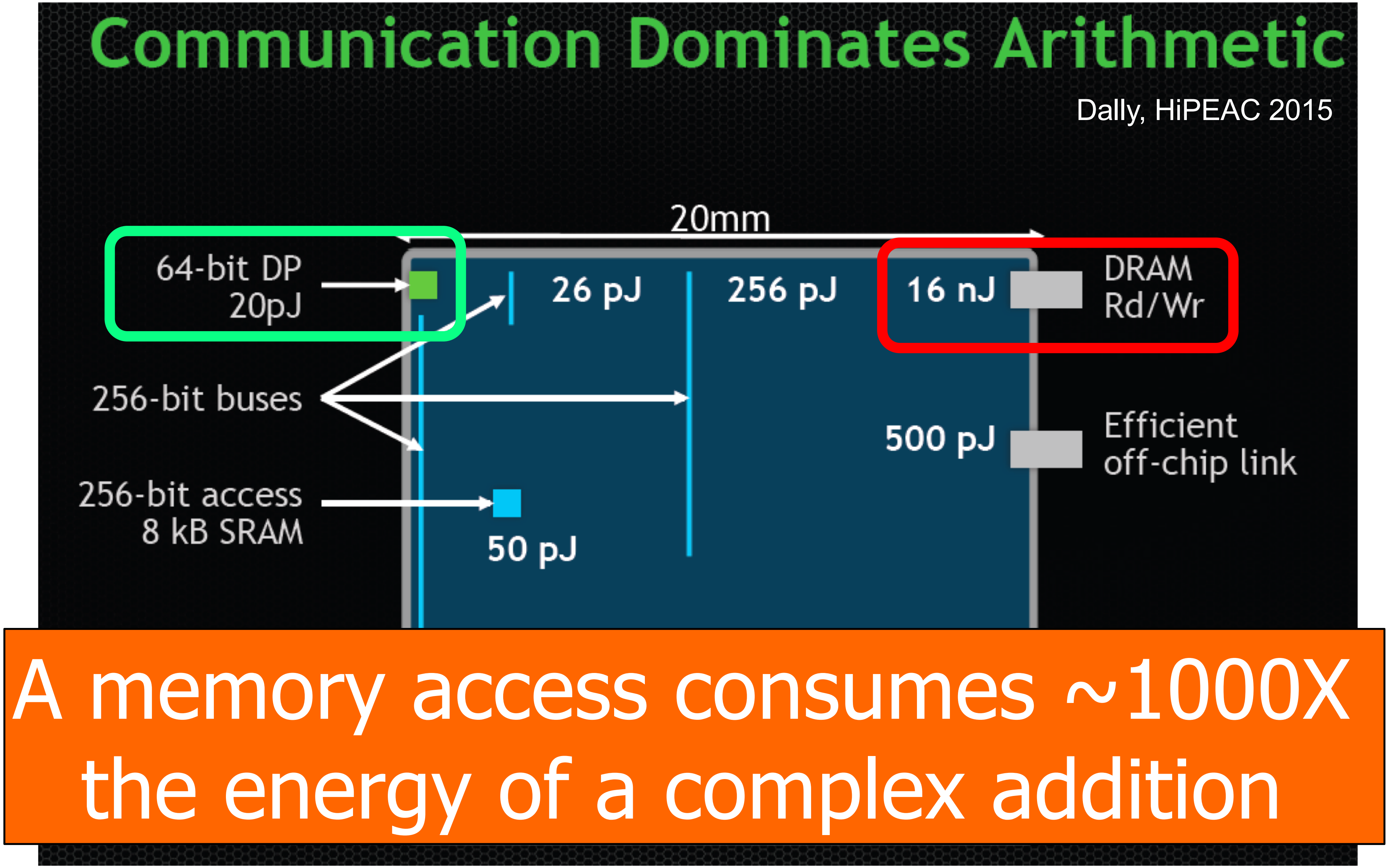
Datacenter Workloads

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

The Energy Perspective



Data Movement vs. Computation Energy



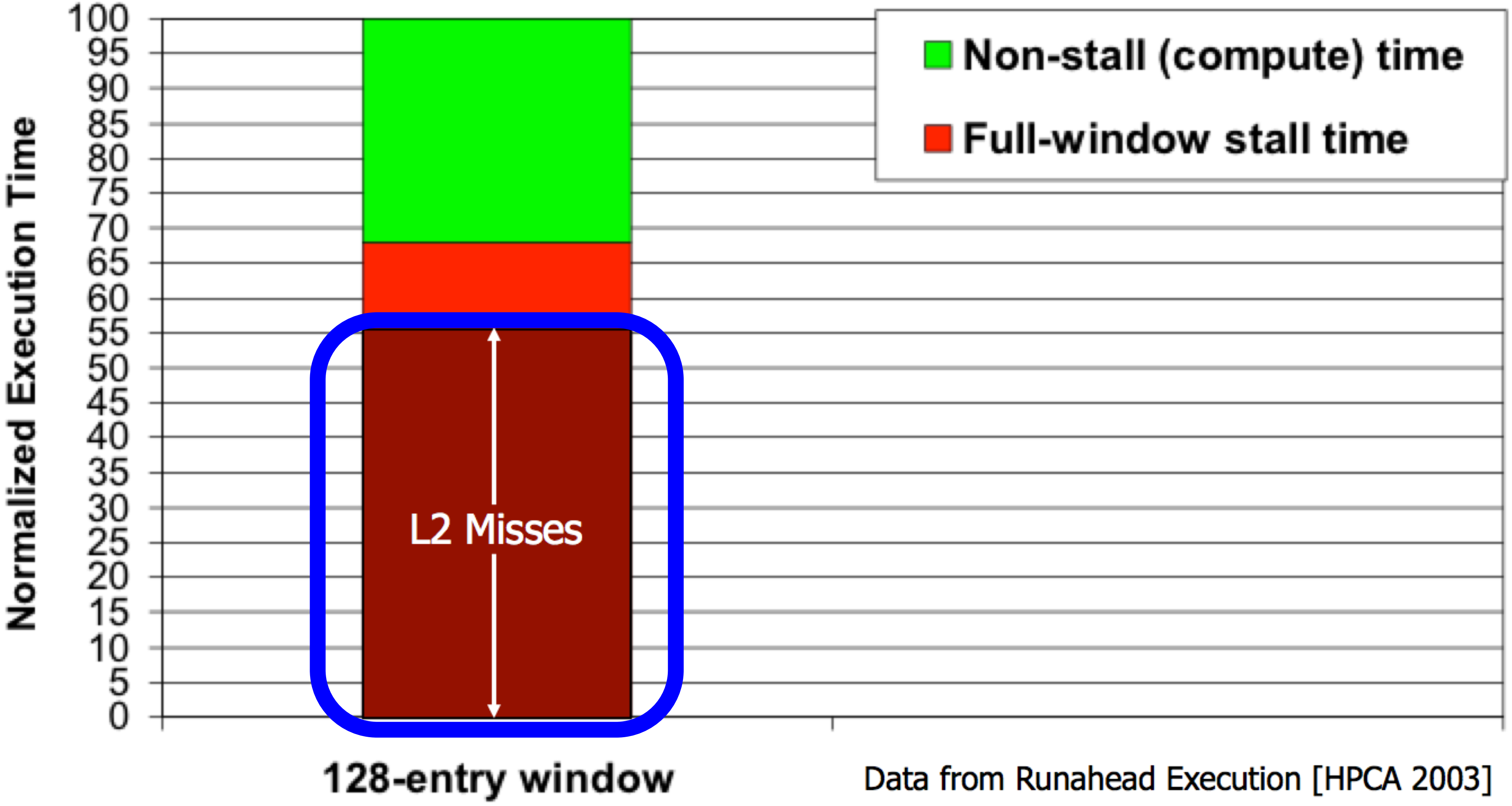
Communication Dominates Arithmetic

Dally, HiPEAC 2015

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

The Performance Perspective (1996-2005)

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

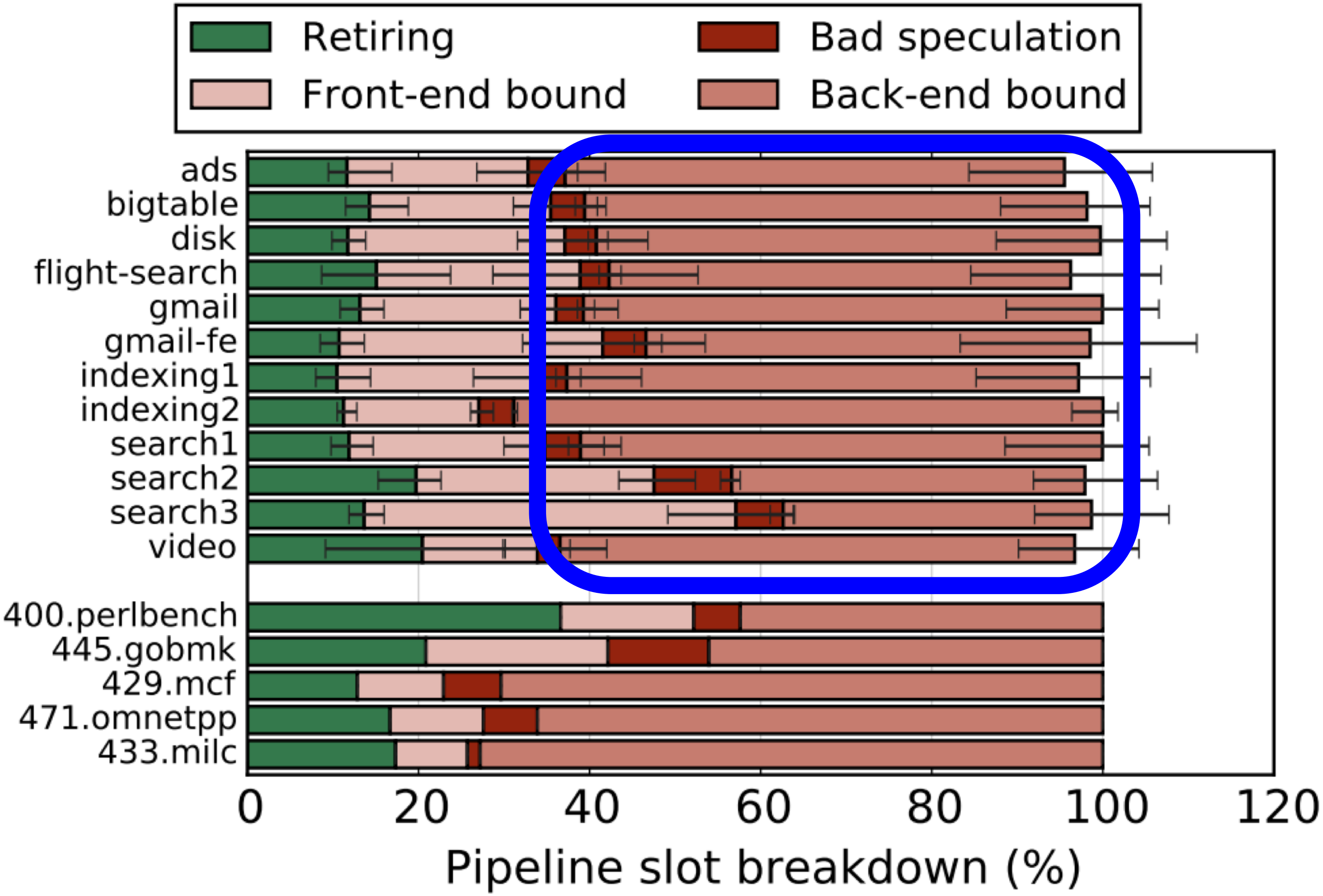


Data from Runahead Execution [HPCA 2003]

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

The Performance Perspective (Today)

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



Kanev+, "Profiling a Warehouse-Scale Computer," ISCA 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

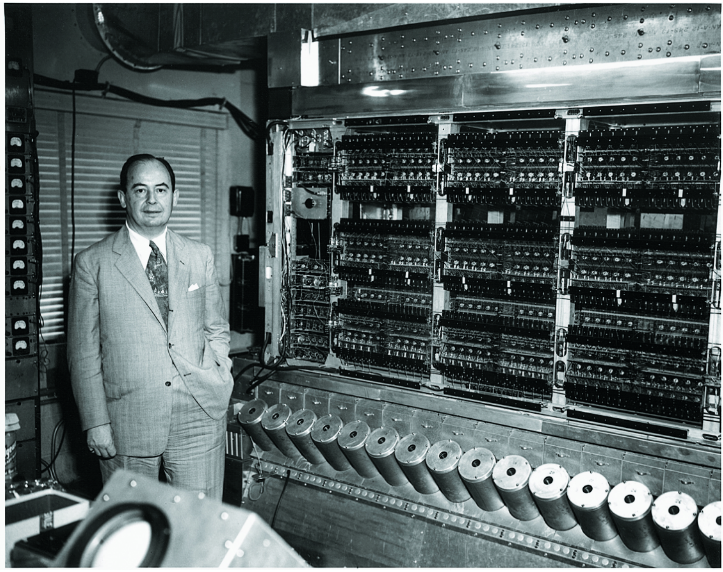
Processing of data

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.

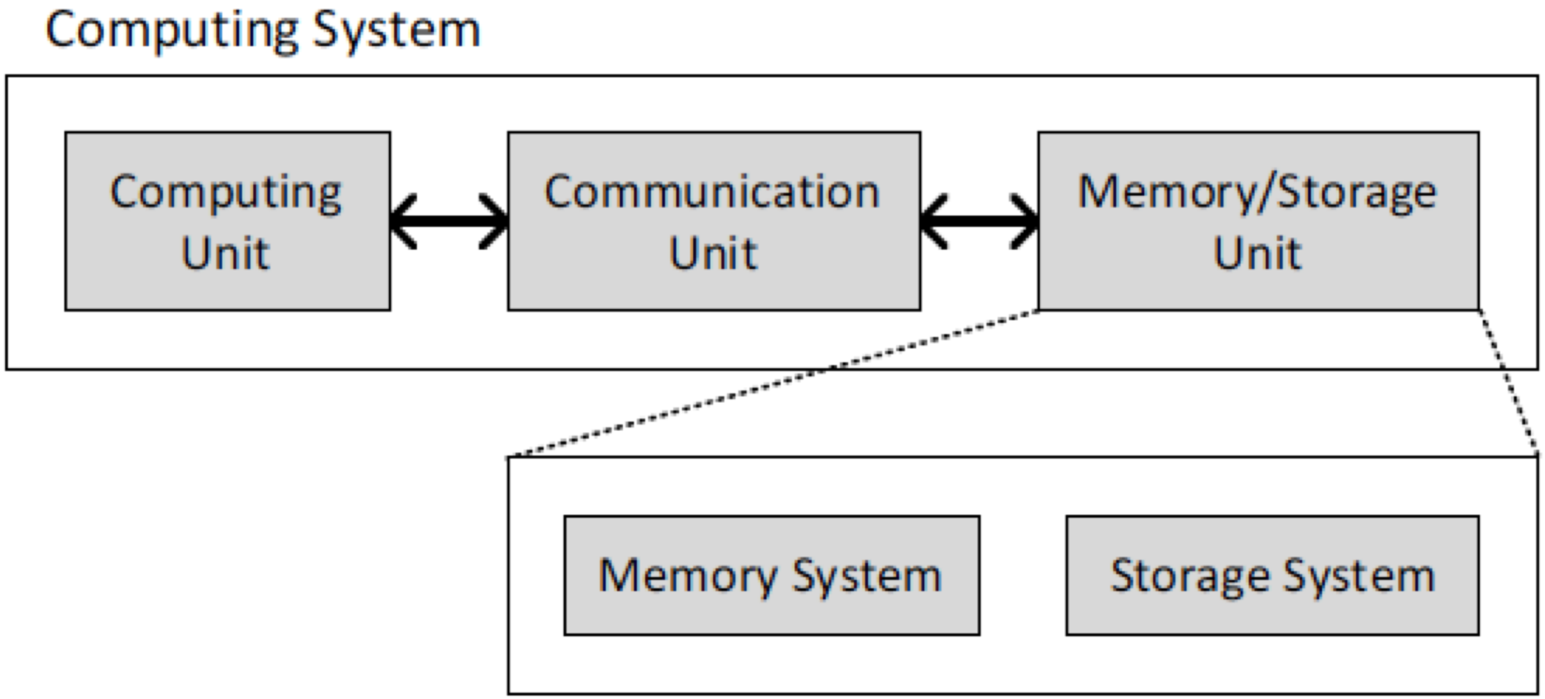
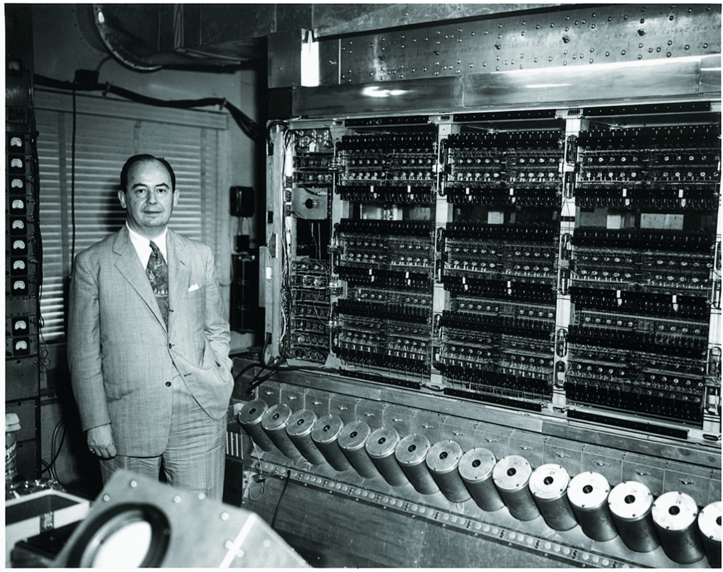


Image source: https://lbsitbytes2010.wordpress.com/2013/03/29/john-von-neumann-roll-no-15/

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.

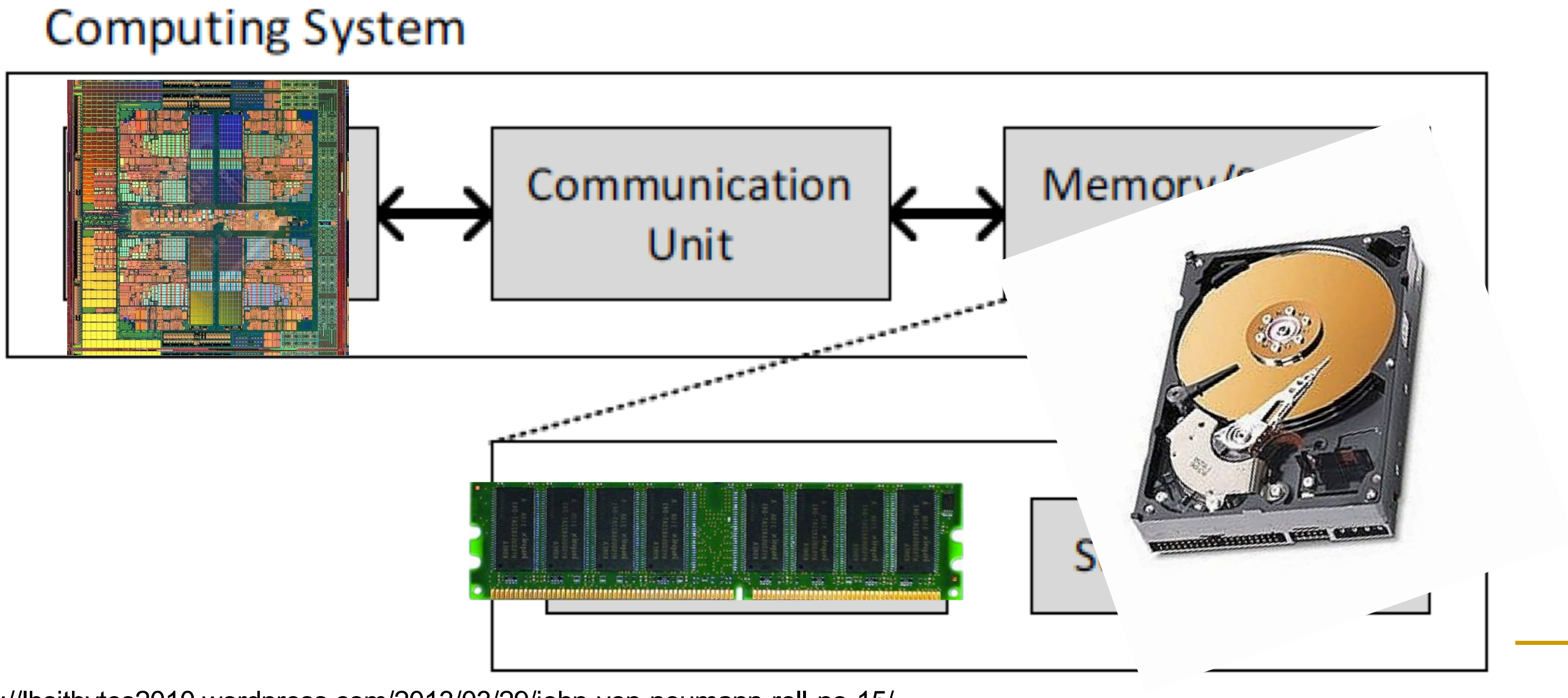
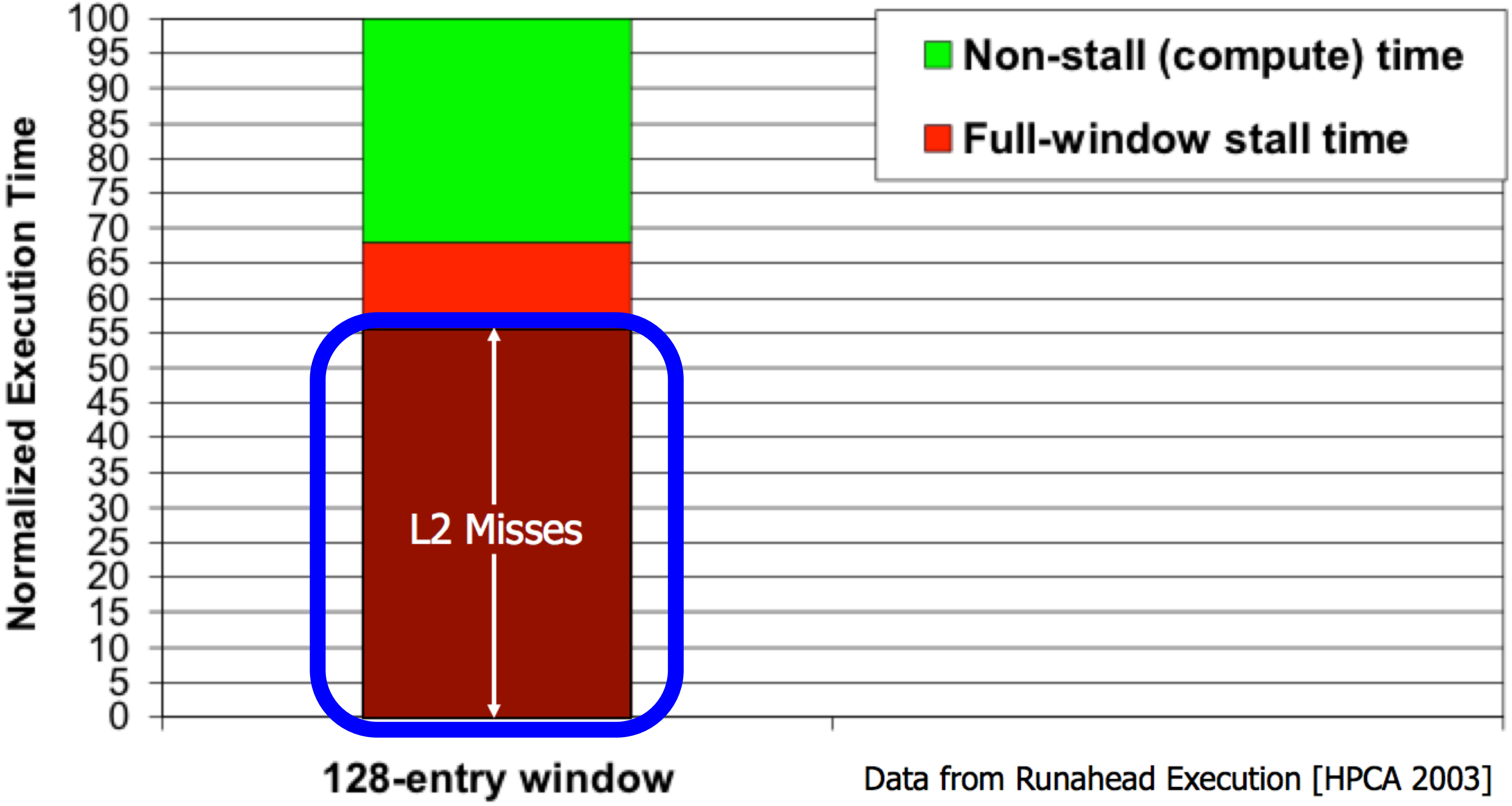


Image source: https://lbsitbytes2010.wordpress.com/2013/03/29/john-von-neumann-roll-no-15/

Yet …

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



Data from Runahead Execution [HPCA 2003]

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

Perils of Processor-Centric Design

- Grossly-imbalanced systems
  - ❑ Processing done only in **one place**
  - ❑ Everything else just stores and moves data: **data moves a lot**

  → Energy inefficient

  → Low performance

  → Complex

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

  → Energy inefficient

  → Low performance

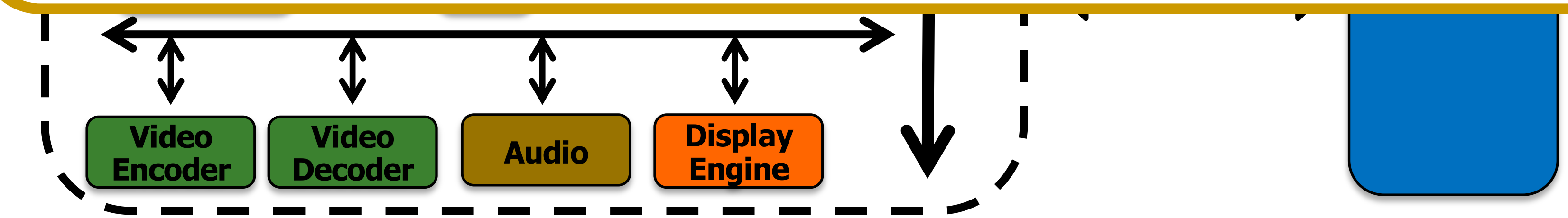
  → Complex

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 …

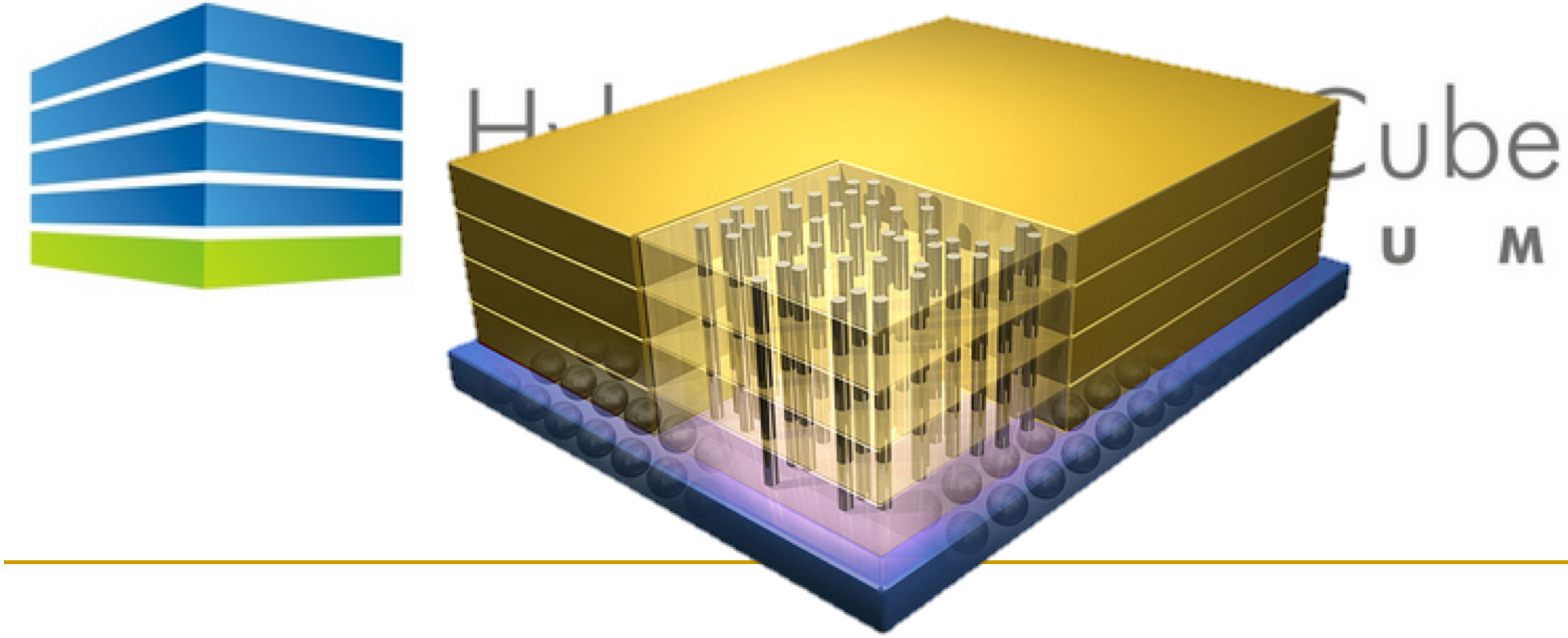
- Enable computation with minimal data movement

- 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



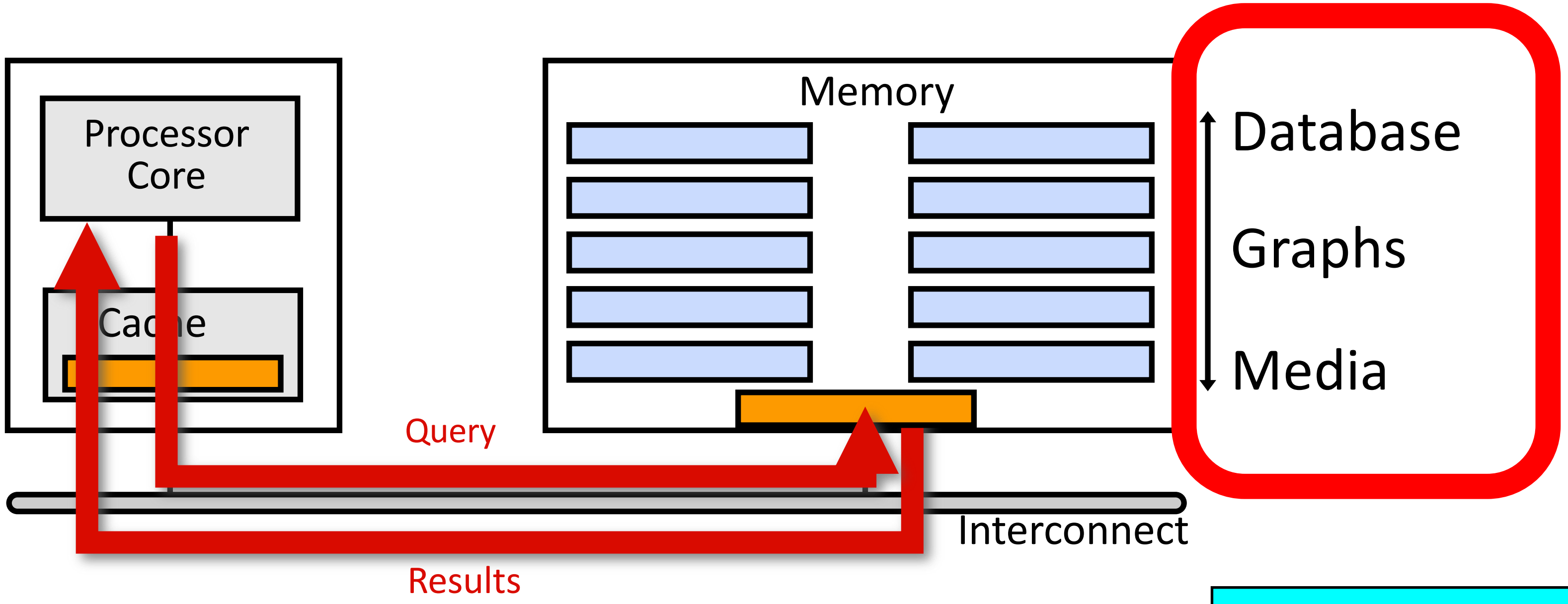
Challenge and Opportunity for Future

High Performance

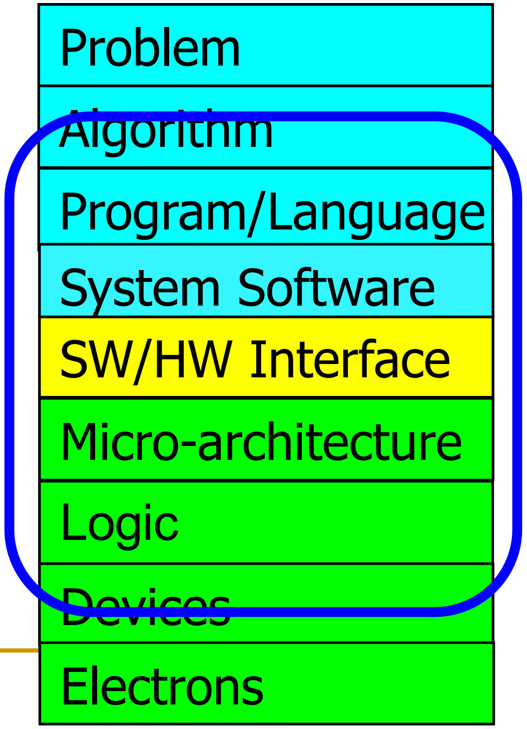
and

Energy Efficiency

Goal: Processing Inside Memory



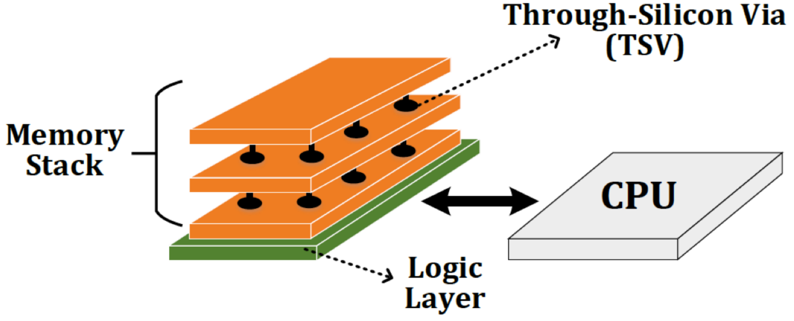
- Many questions… How do we design the:
  - compute-capable memory & controllers?
  - processor chip?
  - software and hardware interfaces?
  - system software and languages?
  - algorithms?



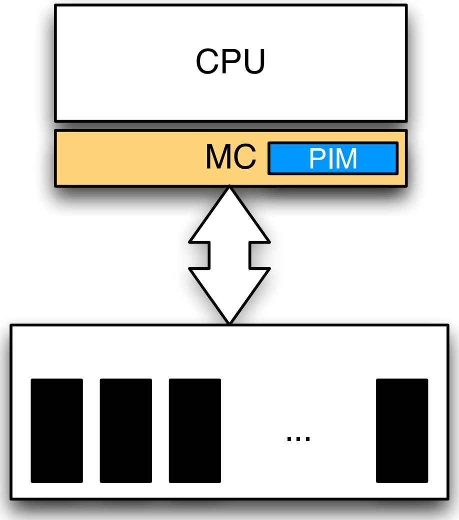
Processing In-Memory (PIM)

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

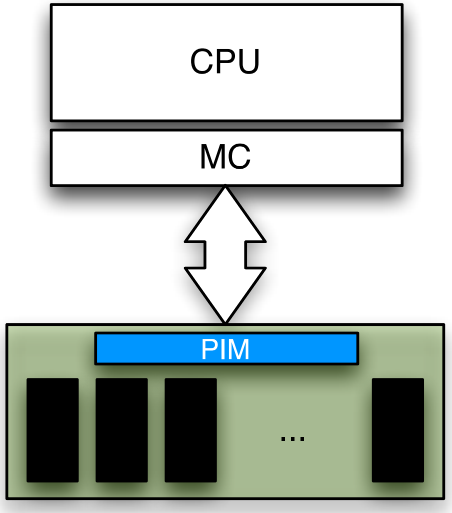
Logic layer
3D stacked DRAM



Memory controller

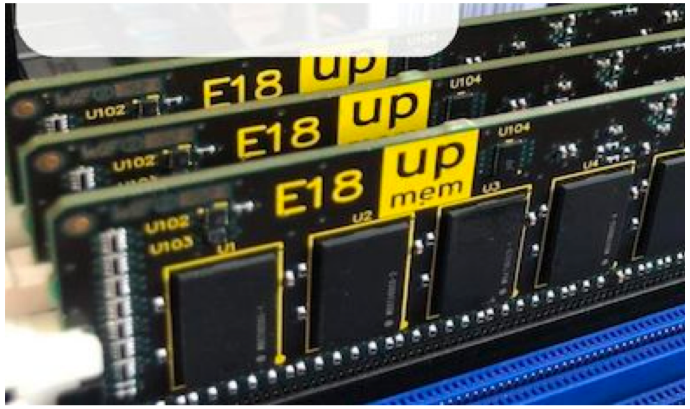


Memory module
(DIMM)

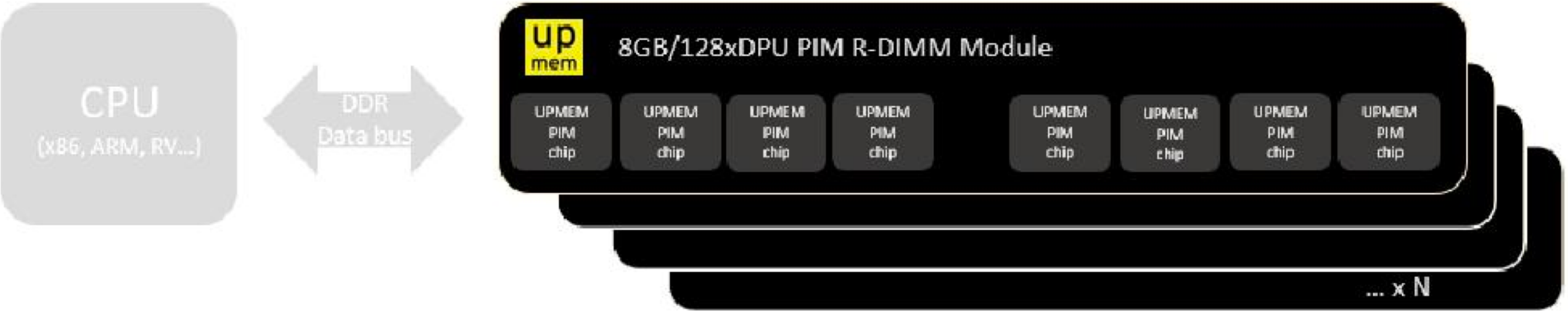


UPMEM Processing-in-DRAM Engine (2019)

- Processing in DRAM Engine
- Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.



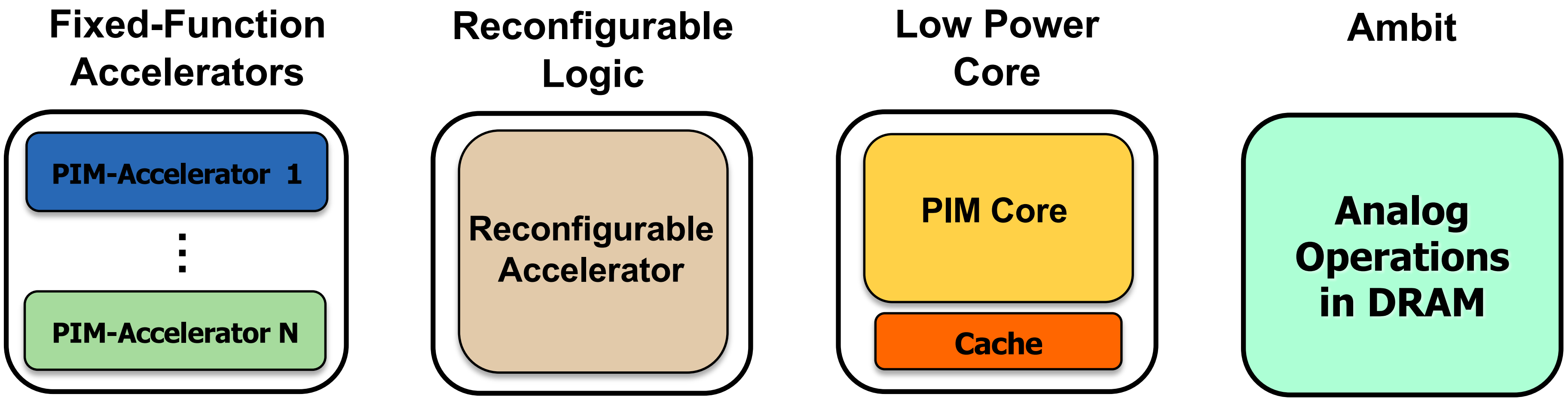
- Replaces **standard** DIMMs
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth



https://www.anandtech.com/show/14750/hot-chips-31-analysis-inmemory-processing-by-upmem
https://www.upmem.com/video-upmem-presenting-its-true-processing-in-memory-solution-hot-chips-2019/

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
- Ambit: In-DRAM Bulk Bitwise Operations (Seshadri+, MICRO'17)



Agenda

- Major Trends Affecting Memory

- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips

  - Exploiting 3D-Stacked Memory

Approach 1: Minimally Changing DRAM

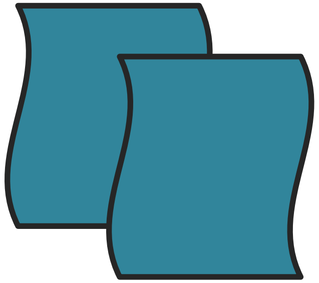
- DRAM has great capability to perform bulk data movement and computation internally with small changes
  - Can exploit internal bandwidth to move data
  - Can exploit analog computation capability
  - ...

- 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)

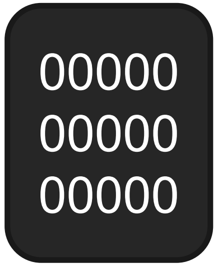
RowClone:
In-Memory Copy and Initialization

Starting Simple: Data Copy and Initialization

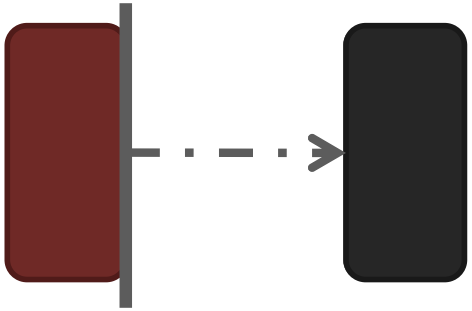
memmove & *memcpy*: 5% cycles in Google's datacenter [Kanev+ ISCA'15]



Forking

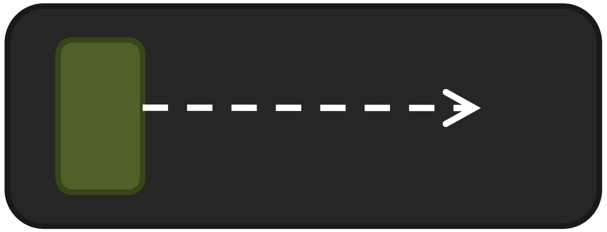


Zero initialization
(e.g., security)



Checkpointing

VM Cloning
Deduplication

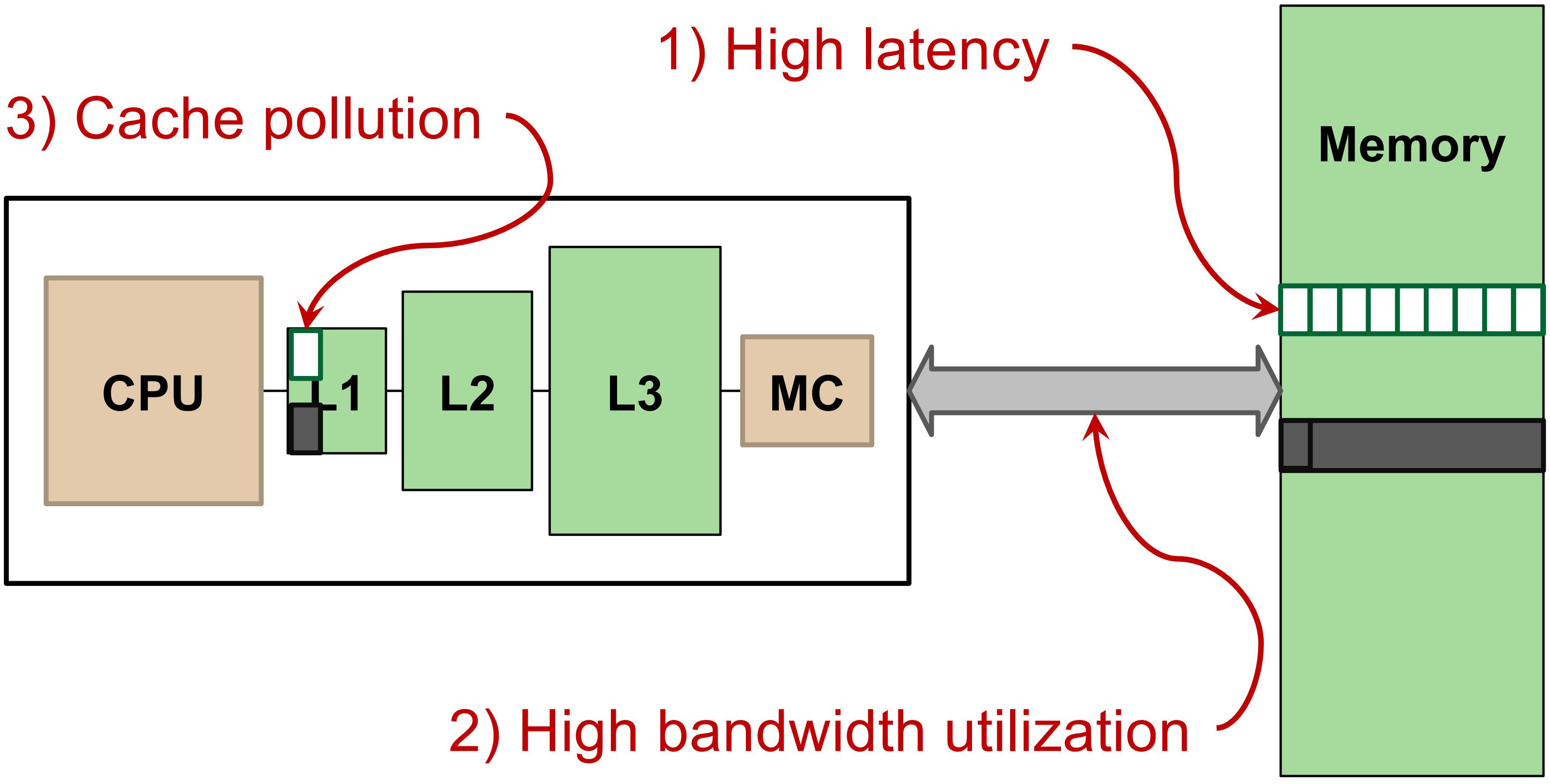


Page Migration

• • •

Many more

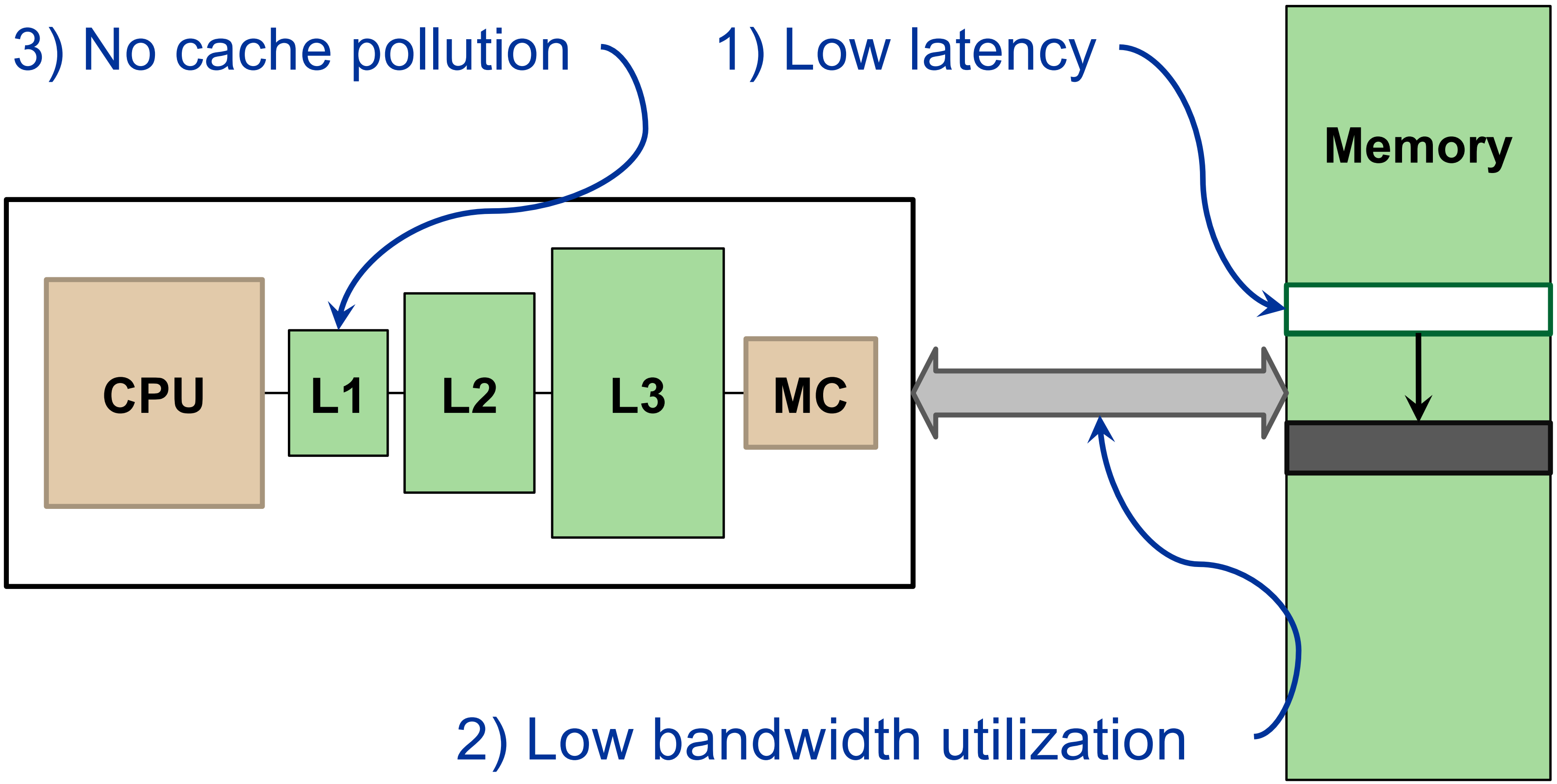
Today's Systems: Bulk Data Copy



4) Unwanted data movement

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

Future Systems: In-Memory Copy



3) No cache pollution

1) Low latency

2) Low bandwidth utilization

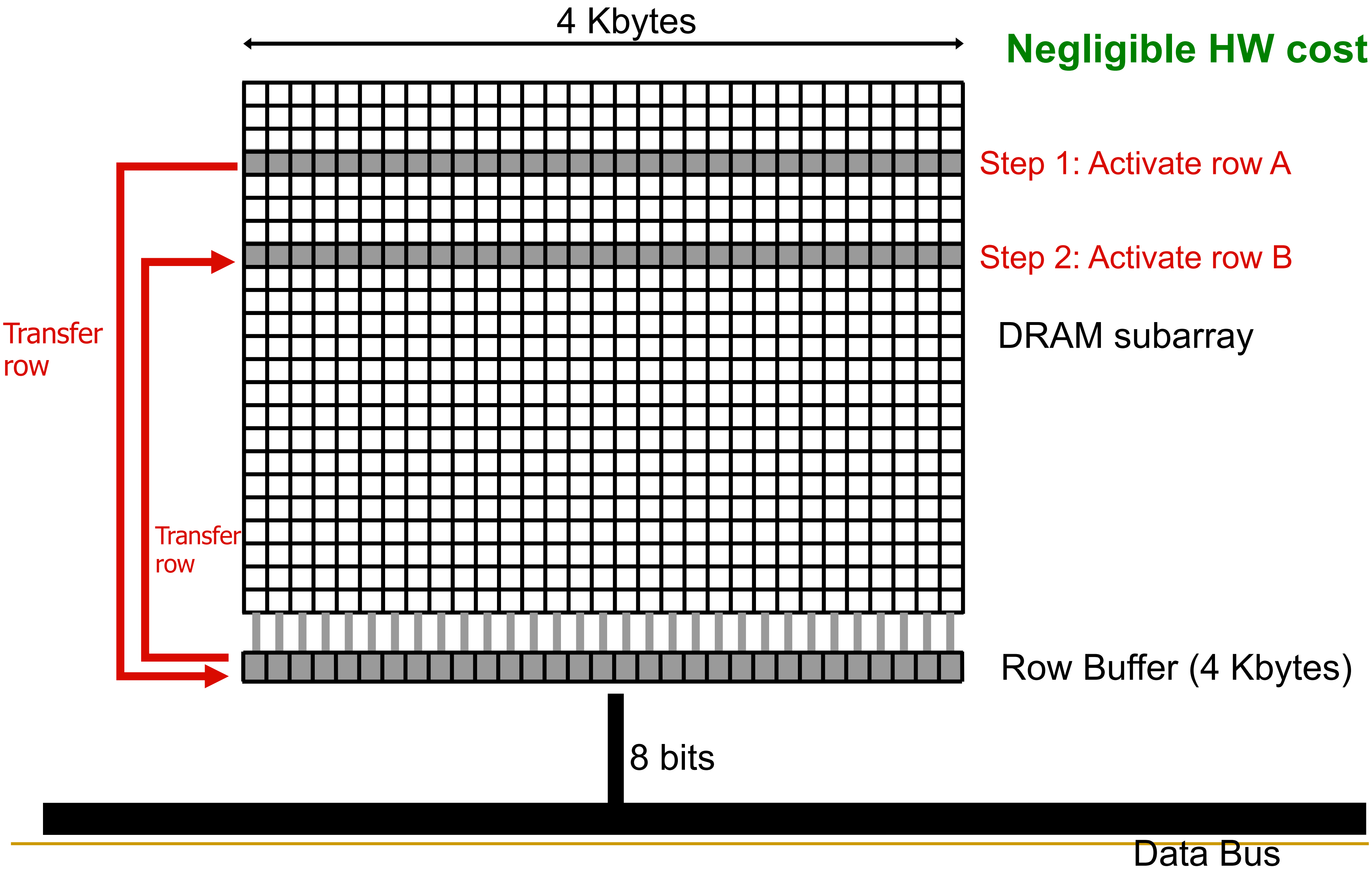
4) No unwanted data movement

1046ns, 3.6uJ → 90ns, 0.04uJ

RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates

Negligible HW cost



4 Kbytes

Step 1: Activate row A

Step 2: Activate row B

DRAM subarray

Transfer row

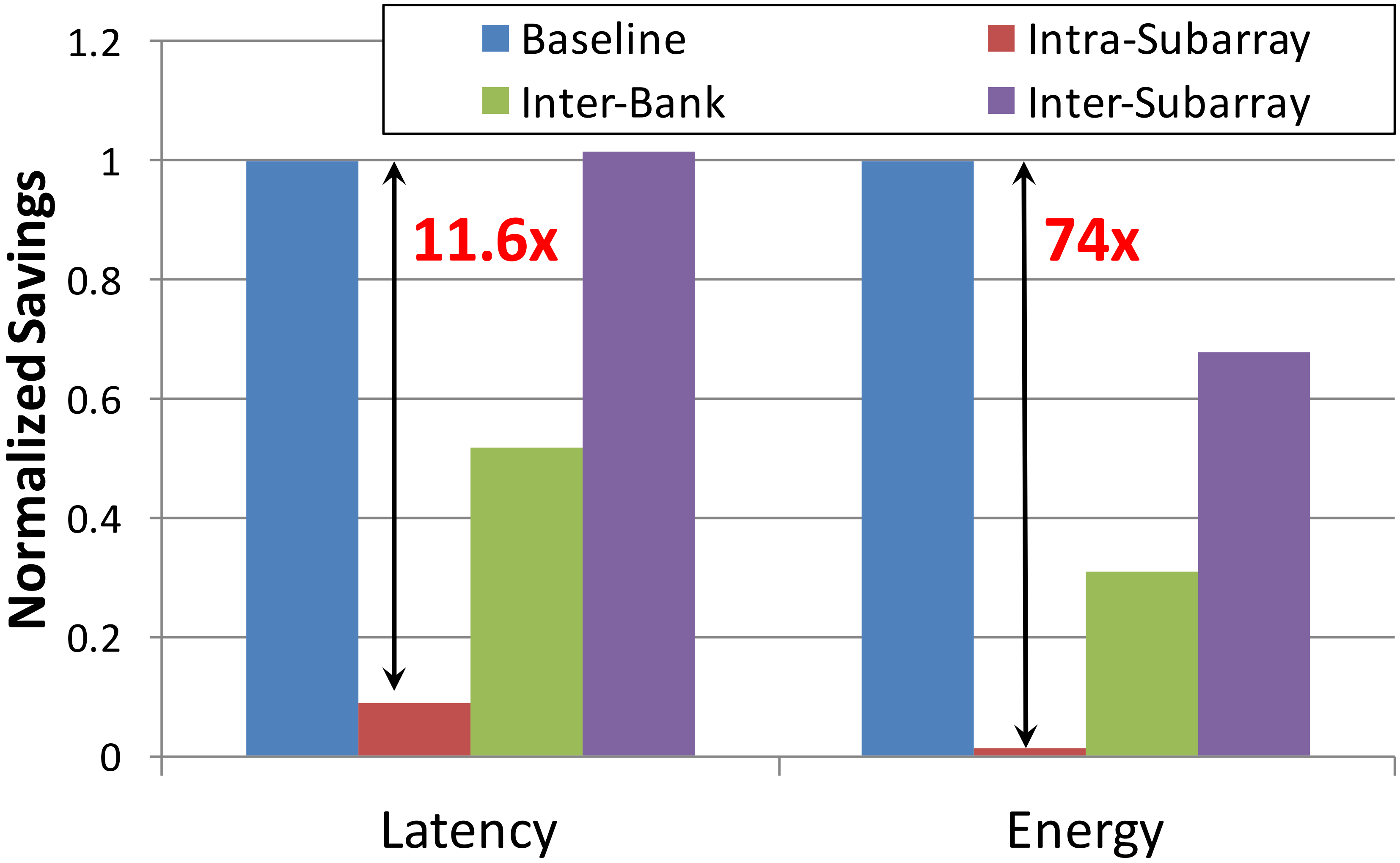
Transfer row

Row Buffer (4 Kbytes)

8 bits

Data Bus

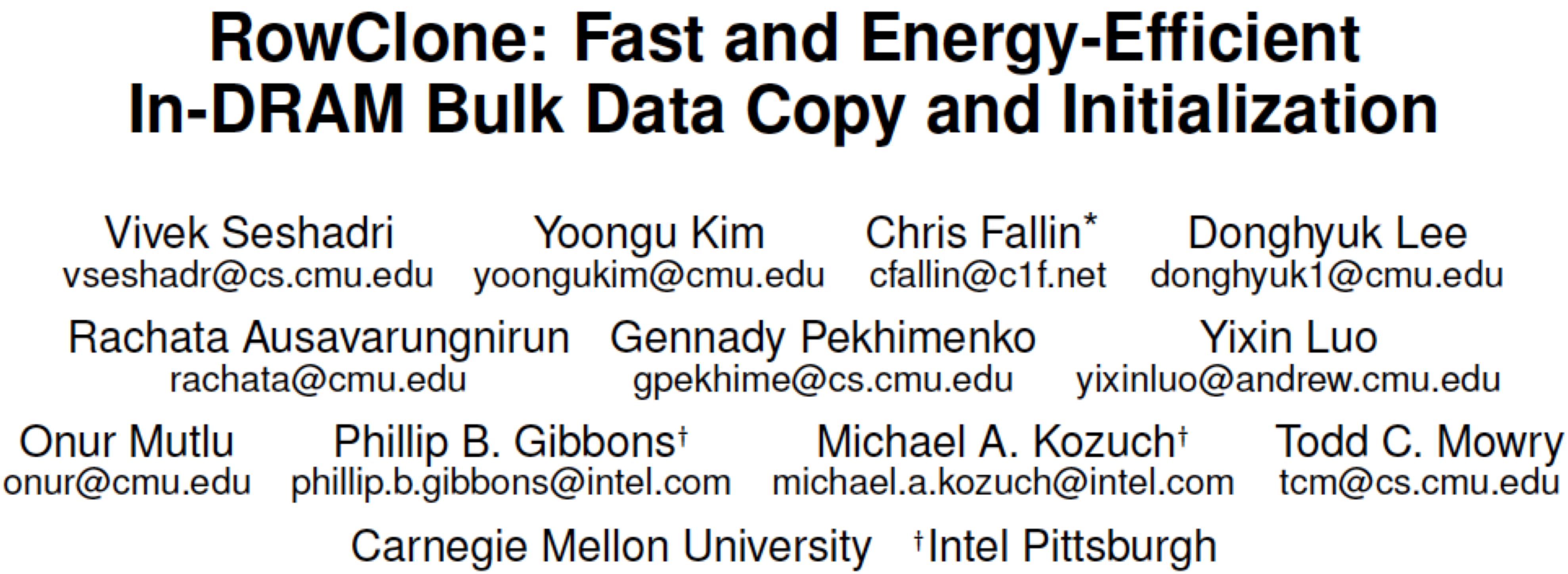
RowClone: Latency and Energy Savings



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

Ambit:
In-Memory Bulk Bitwise Operations

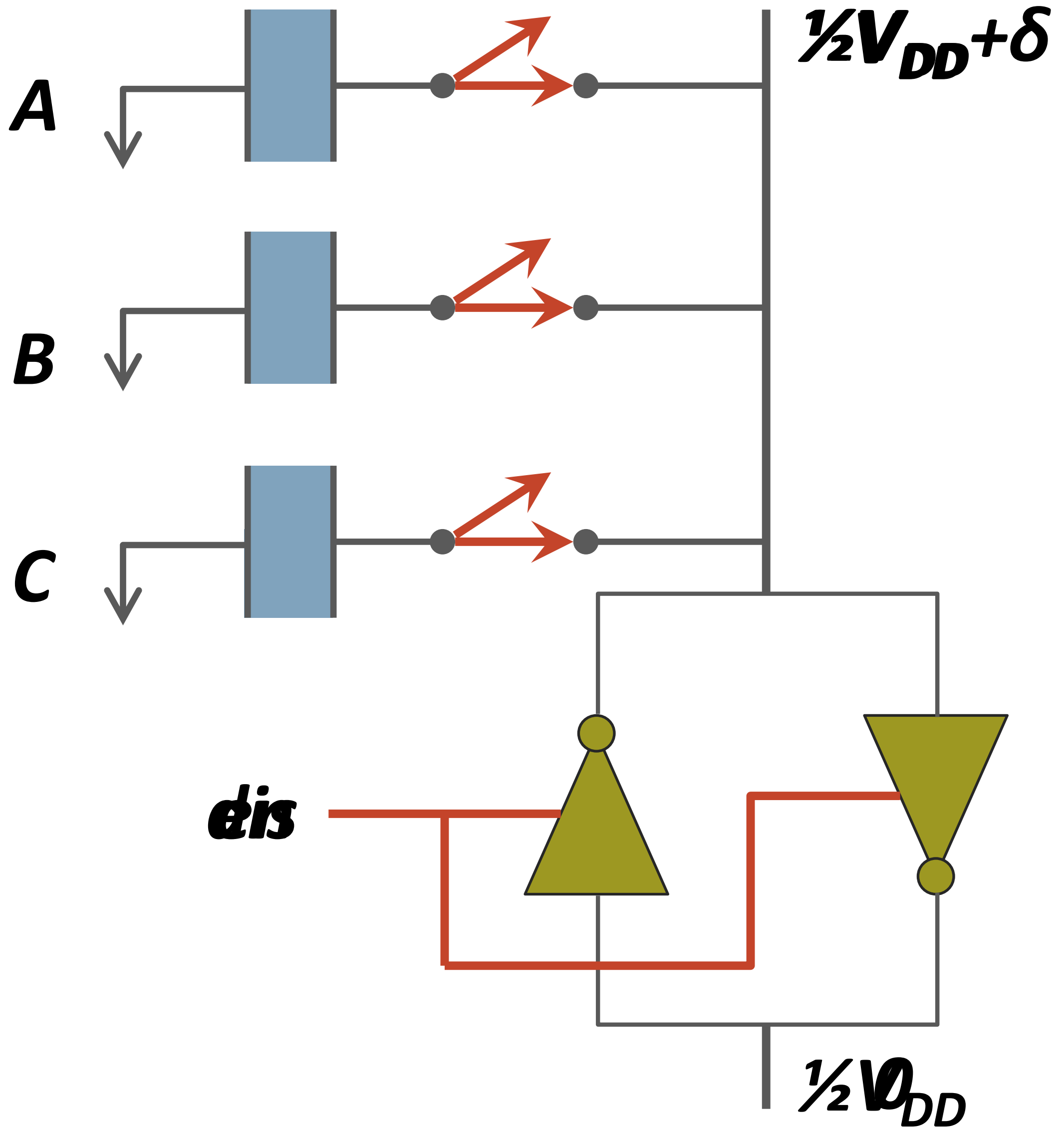
In-Memory Bulk Bitwise Operations

- 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) +
~C(AB)

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

In-DRAM Bulk Bitwise AND/OR Operation

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

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

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

More on In-DRAM Bulk AND/OR

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

Fast Bulk Bitwise AND and OR in DRAM

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

*Carnegie Mellon University †Intel Pittsburgh

In-DRAM NOT: Dual Contact Cell

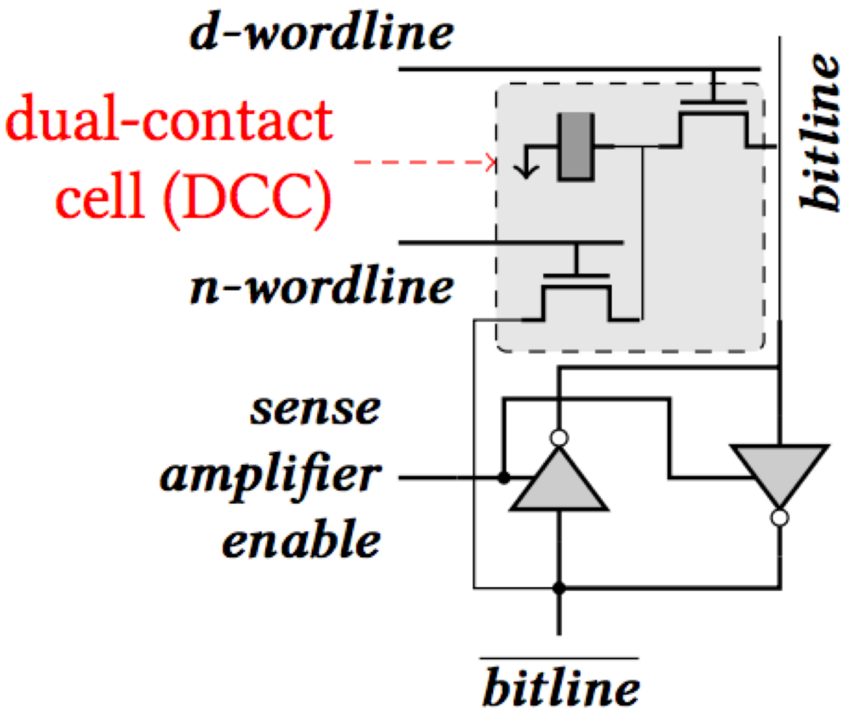


Figure 5: A dual-contact cell connected to both ends of a sense amplifier

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

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

In-DRAM NOT Operation

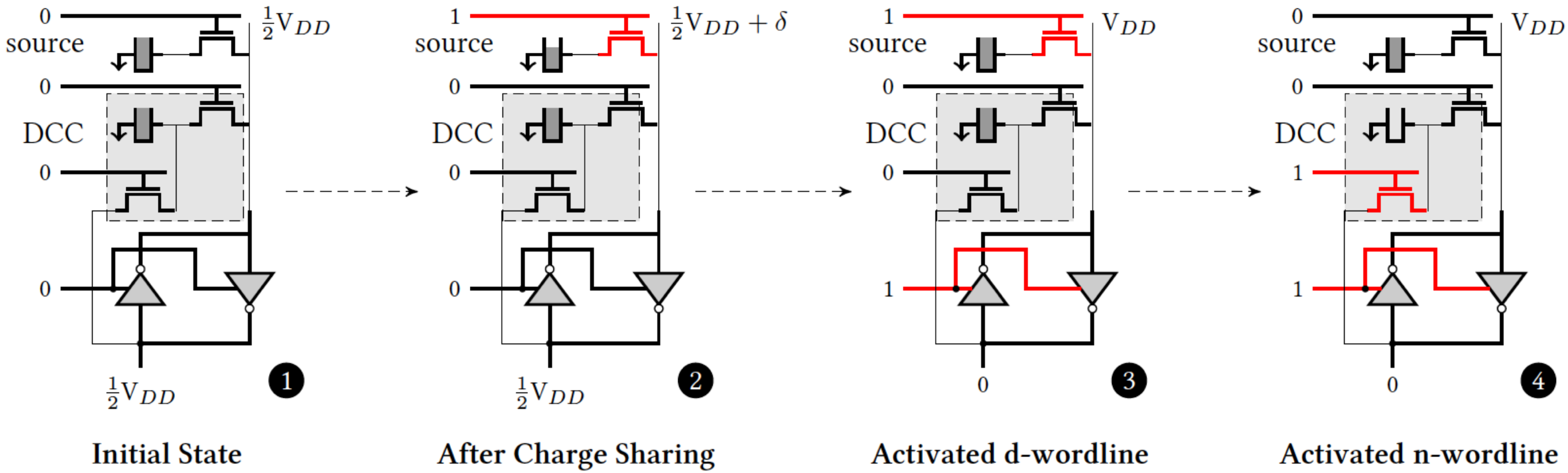


Figure 5: Bitwise NOT using a dual contact capacitor

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

Performance: In-DRAM Bitwise Operations

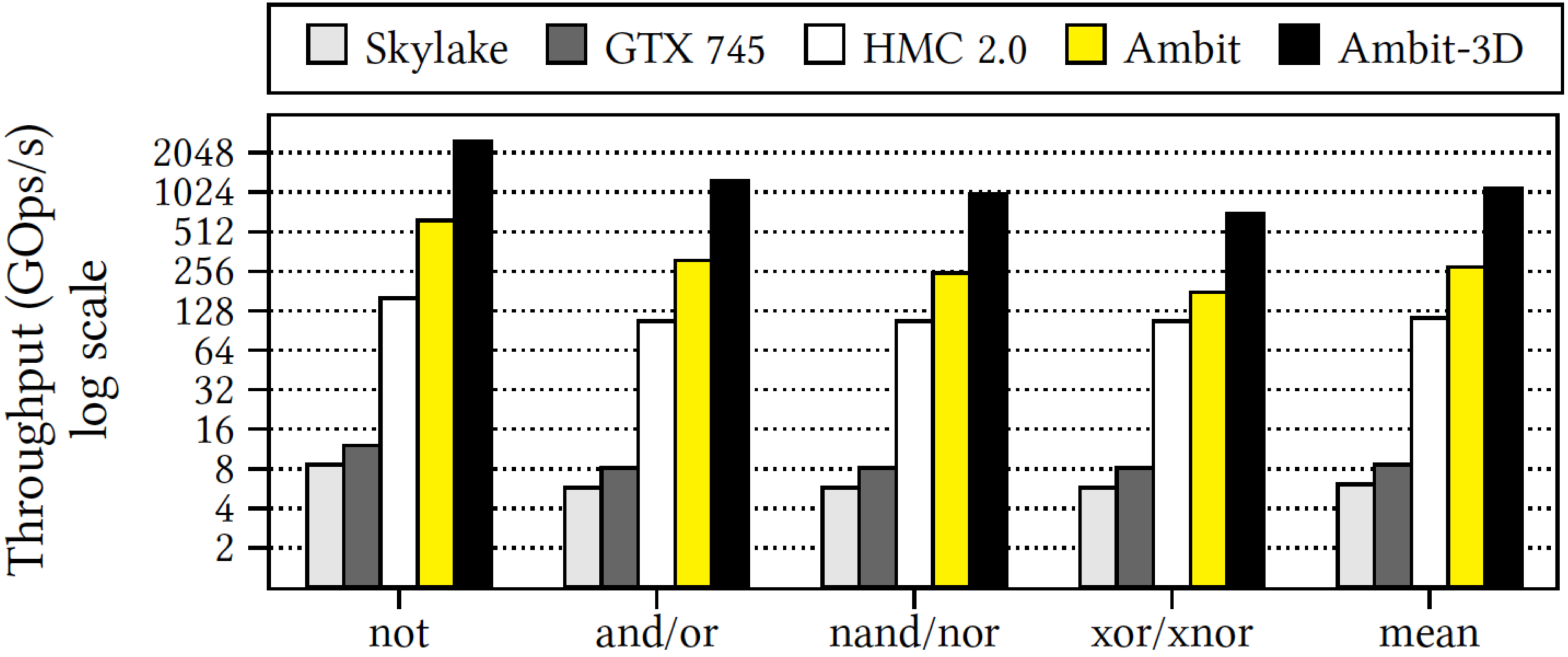
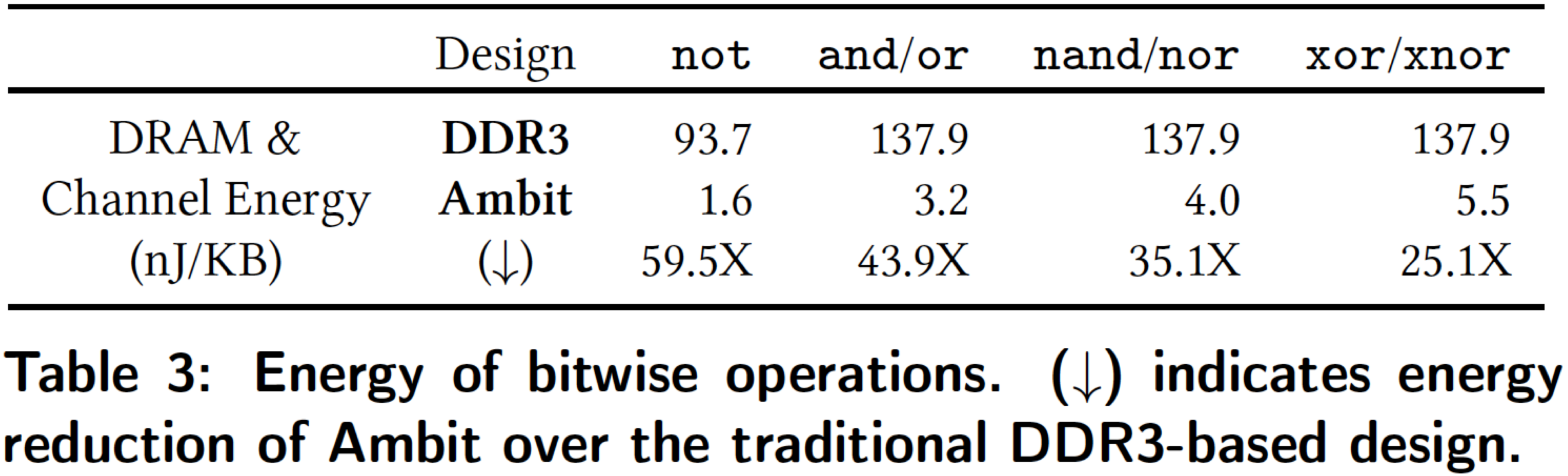


Figure 9: Throughput of bitwise operations on various systems.

Energy of In-DRAM Bitwise Operations



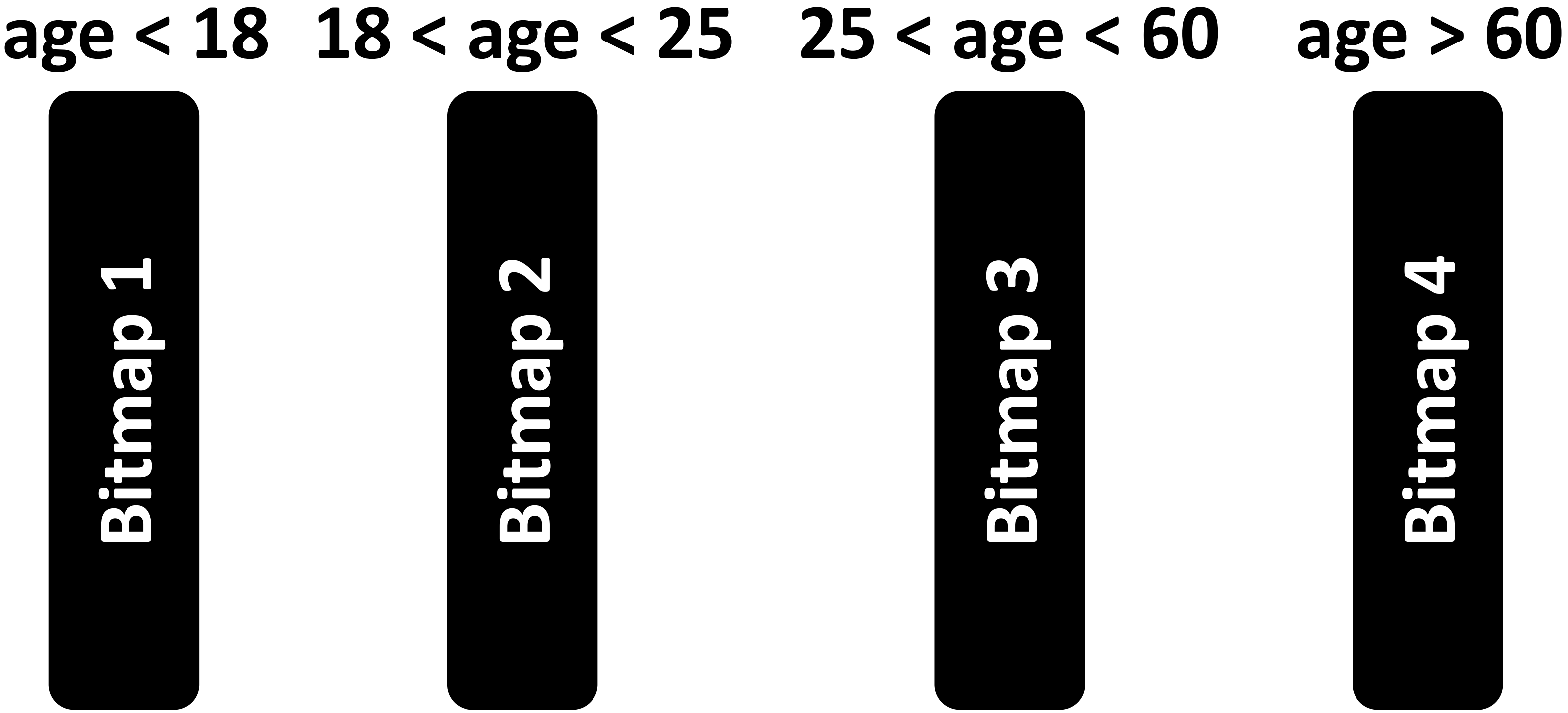
	Design	`not`	`and/or`	`nand/nor`	`xor/xnor`
DRAM & Channel Energy (nJ/KB)	**DDR3**	93.7	137.9	137.9	137.9
	Ambit	1.6	3.2	4.0	5.5
	(↓)	59.5X	43.9X	35.1X	25.1X

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

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

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

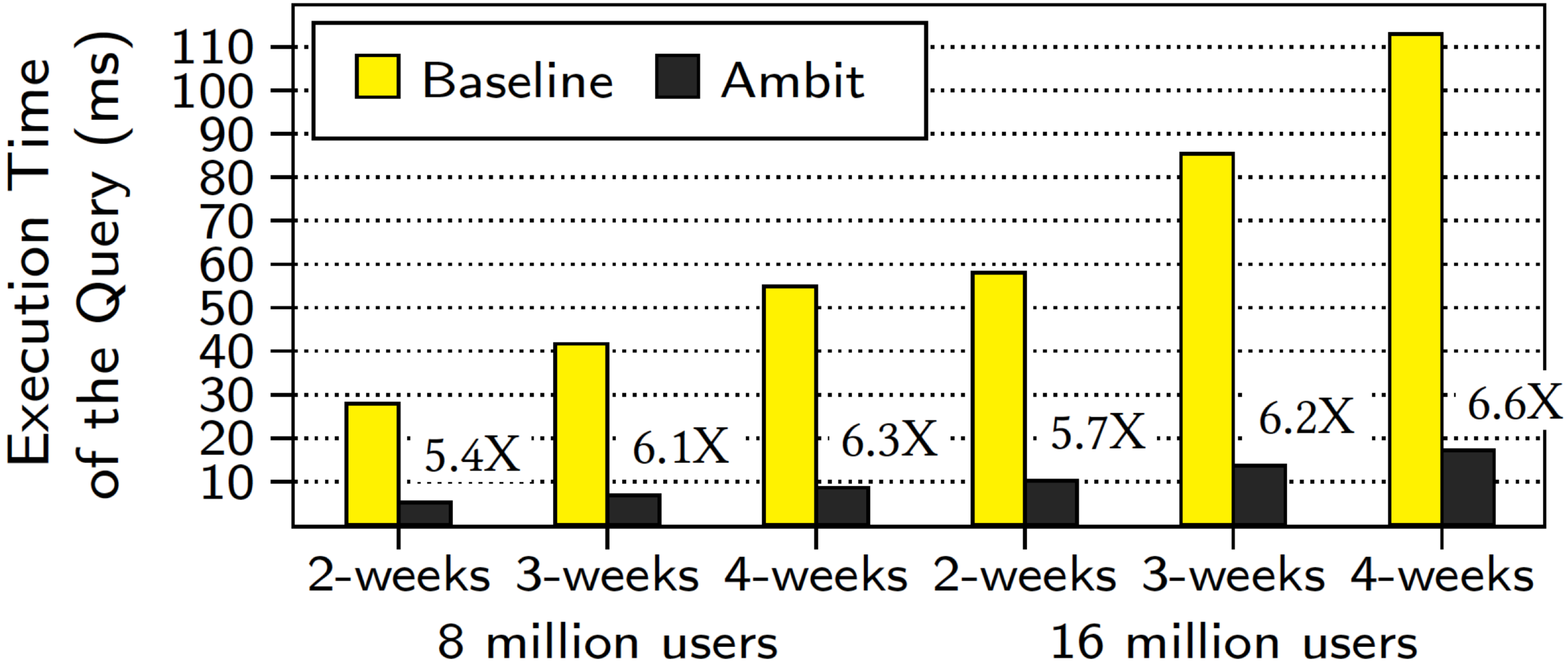
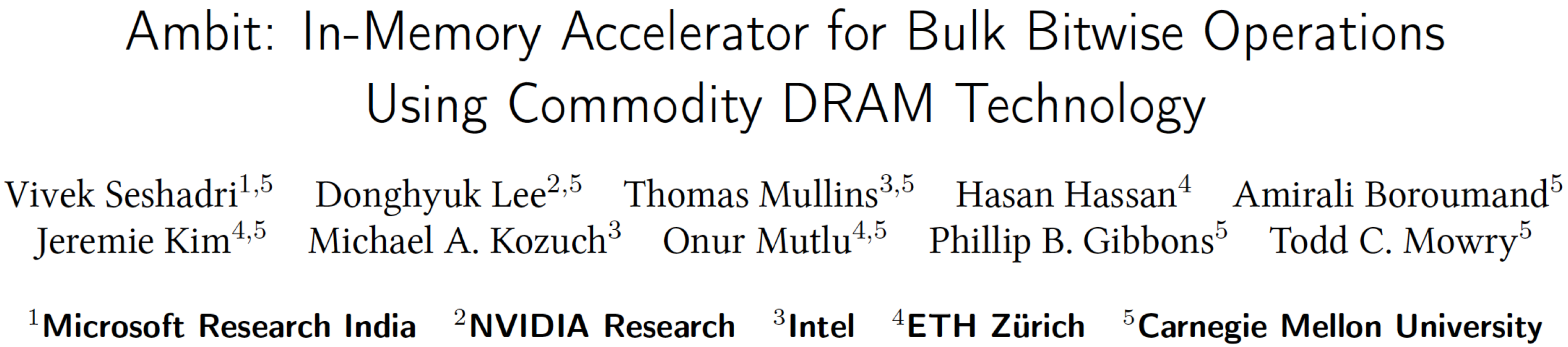


Figure 10: Bitmap index performance. The value above each bar indicates the reduction in execution time due to Ambit.

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

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 Hassan[4] Amirali Boroumand[5]
Jeremie Kim[4,5] Michael A. Kozuch[3] Onur Mutlu[4,5] Phillip B. Gibbons[5] Todd C. Mowry[5]

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

Agenda

- Major Trends Affecting Memory

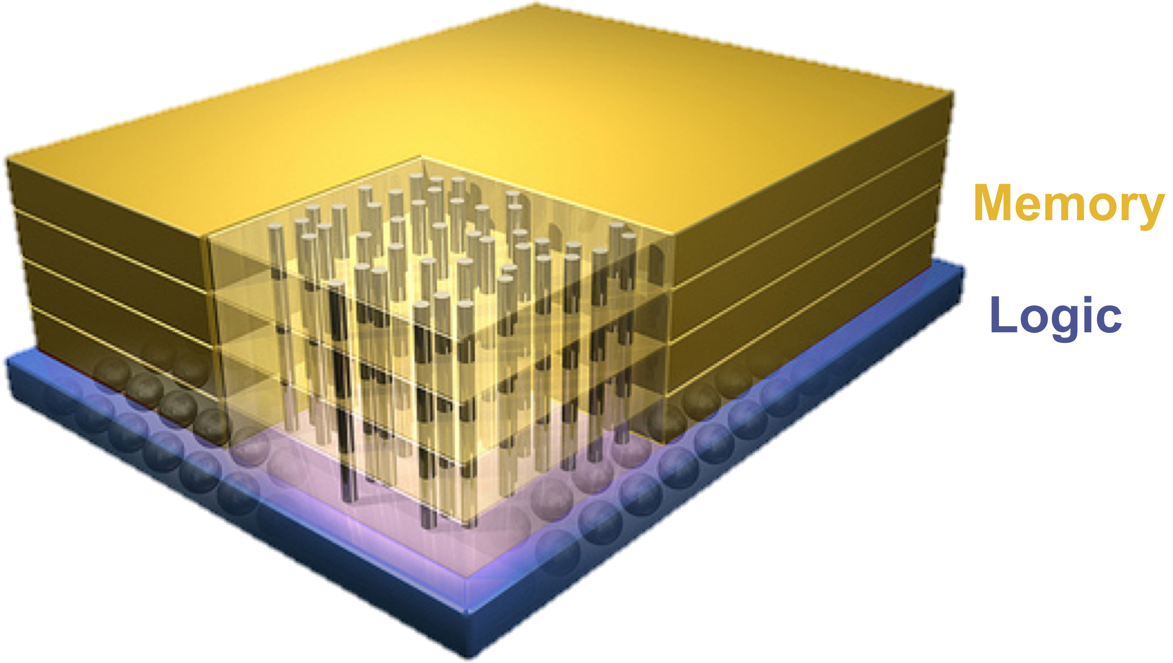
- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips

  - Exploiting 3D-Stacked Memory

Approach 2: 3D-Stacked Logic+Memory

Hybrid Memory Cube

CONSORTIUM



Graph Processing

- Large graphs are everywhere (circa 2015)

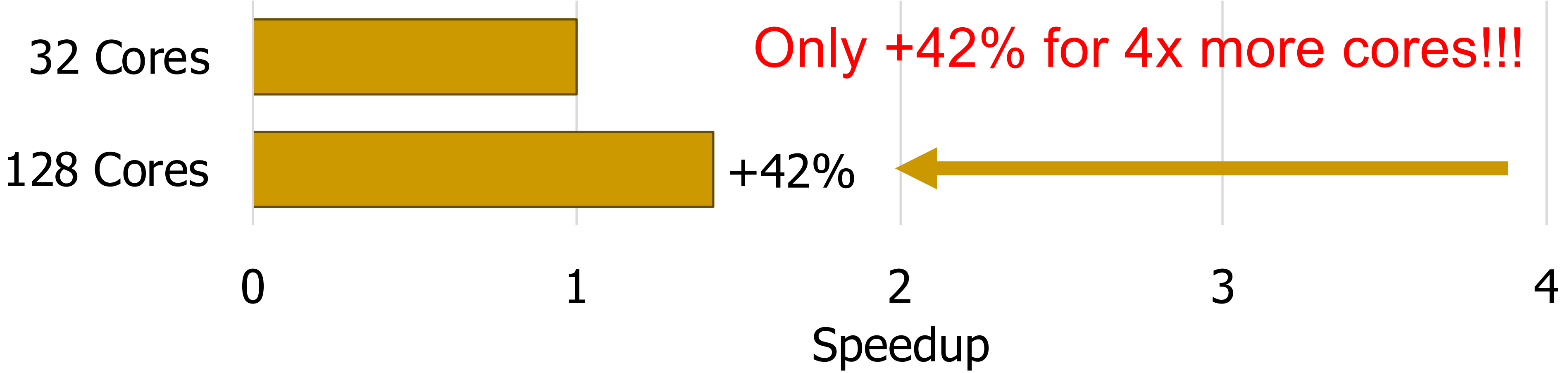
36 Million
Wikipedia Pages

1.4 Billion
Facebook Users

300 Million
Twitter Users

30 Billion
Instagram Photos

- Scalable large-scale graph processing is challenging



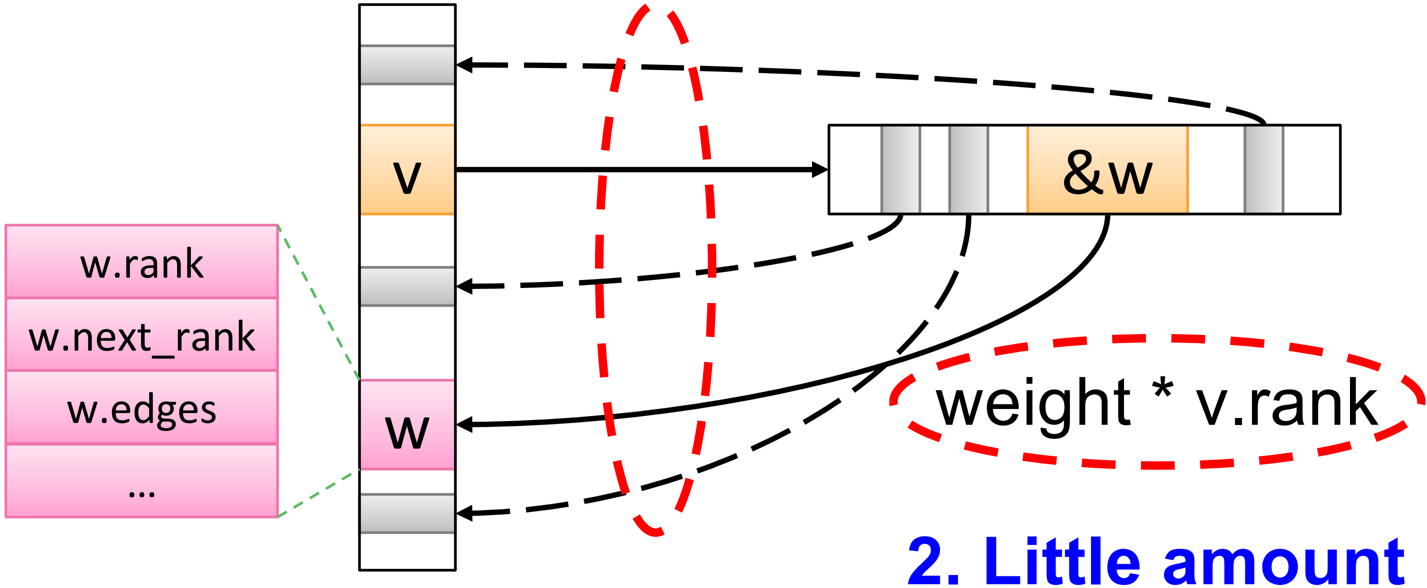
Only +42% for 4x more cores!!!

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

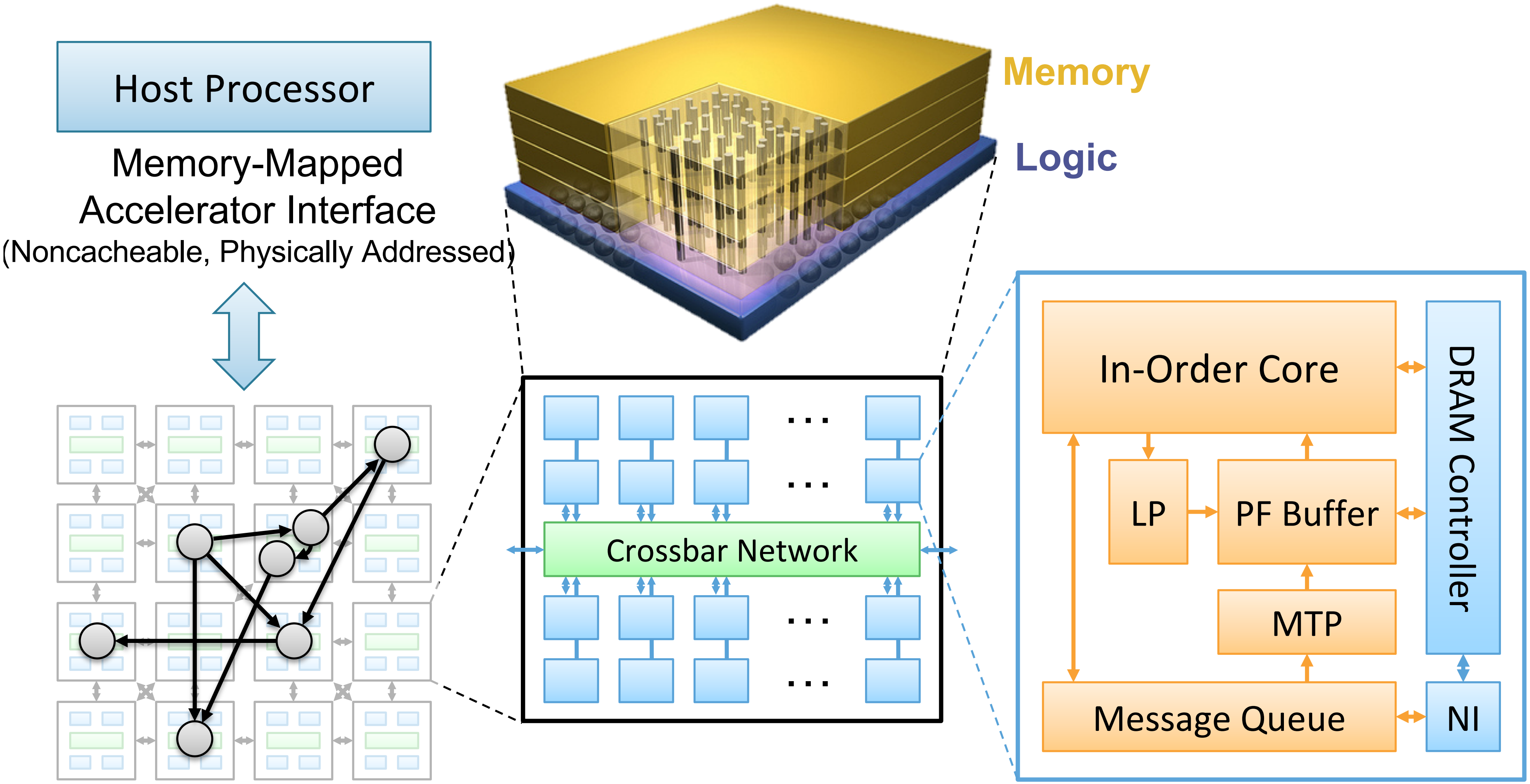
- How can we accelerate important applications if we use 3D-stacked memory as a coarse-grained accelerator?
  - what is the architecture and programming model?
  - what are the mechanisms for acceleration?

- 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



Host Processor

Memory-Mapped Accelerator Interface

(Noncacheable, Physically Addressed)

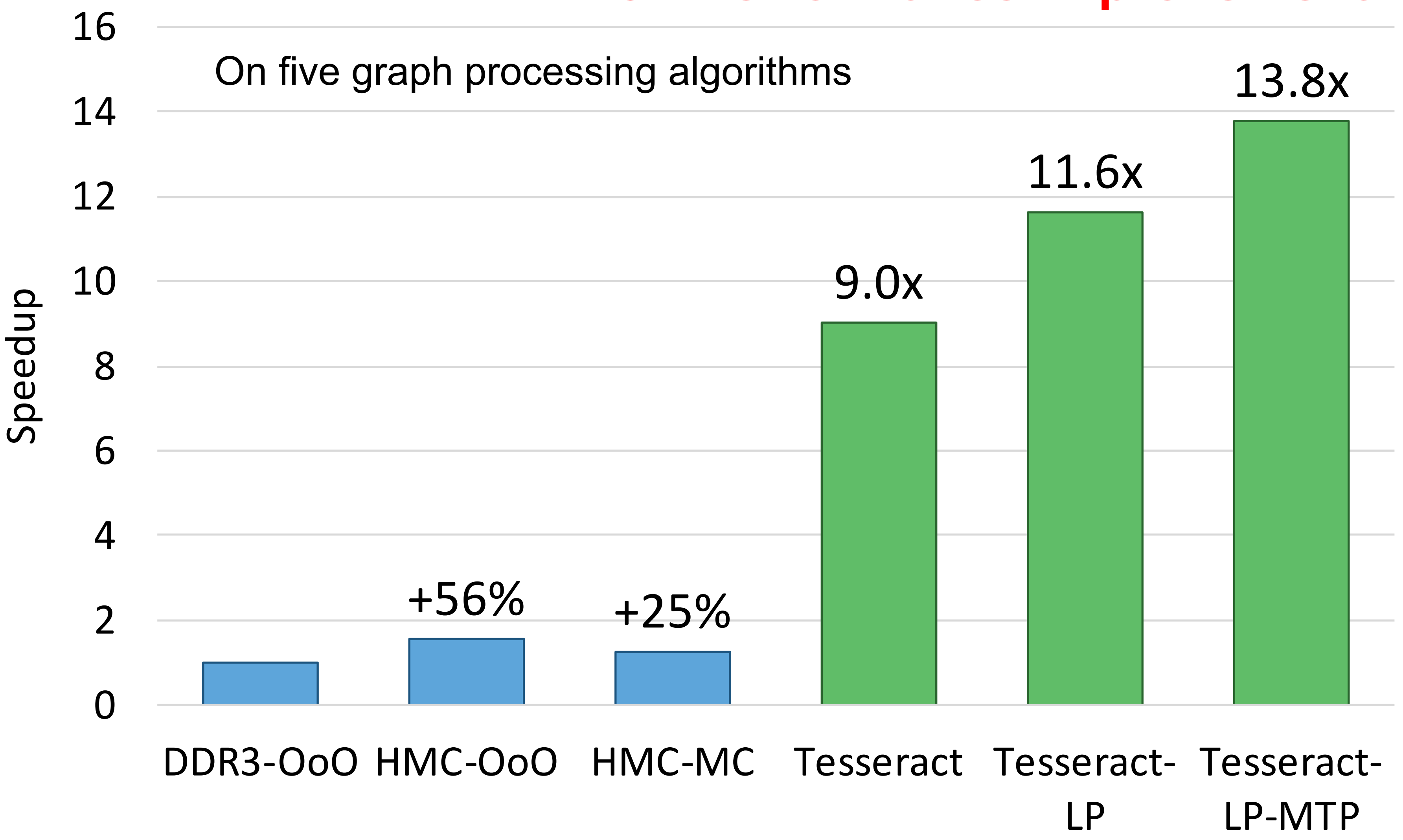
Ahn+, “A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing” ISCA 2015.

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

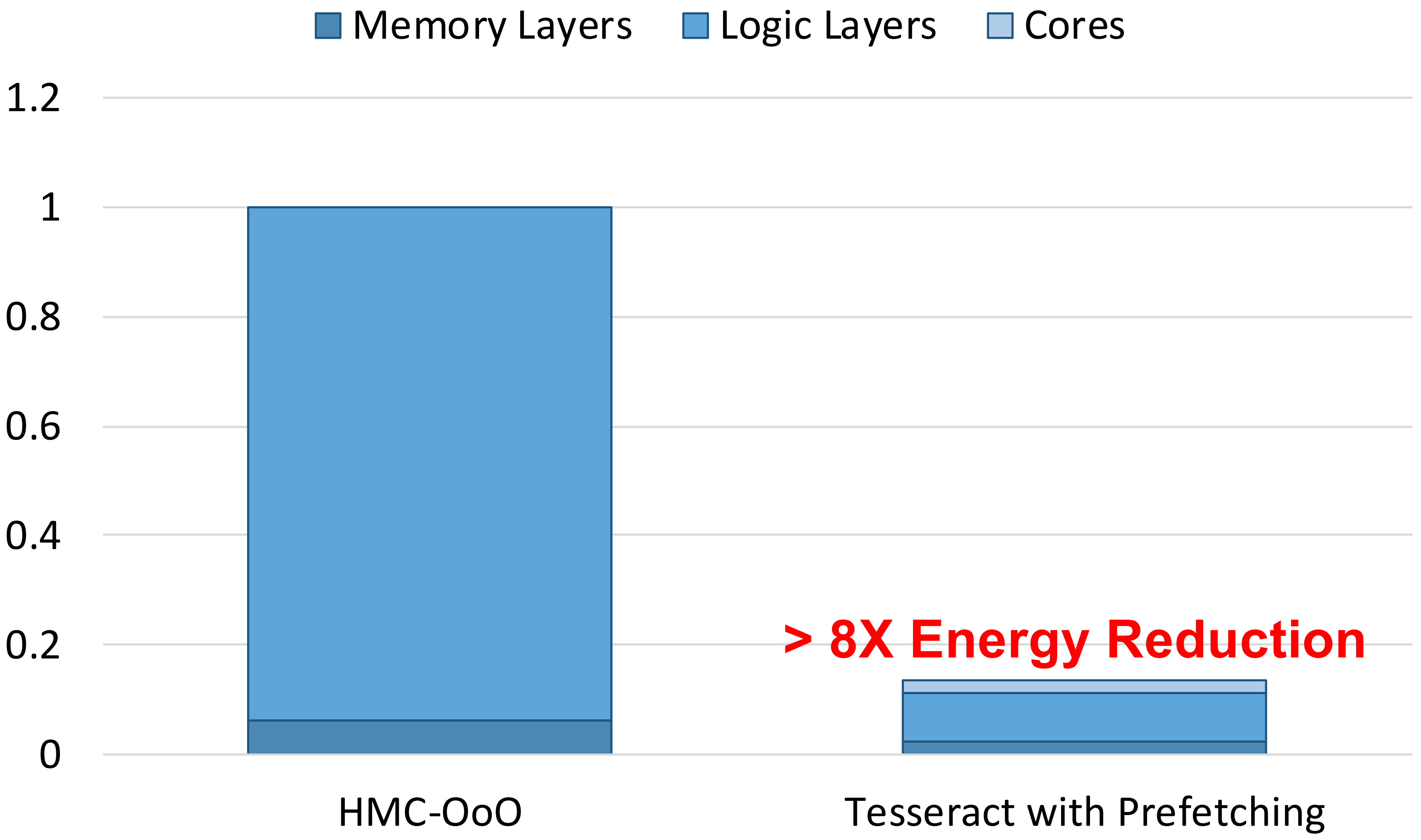
Tesseract Graph Processing Performance

>13X Performance Improvement



Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

Tesseract Graph Processing System Energy

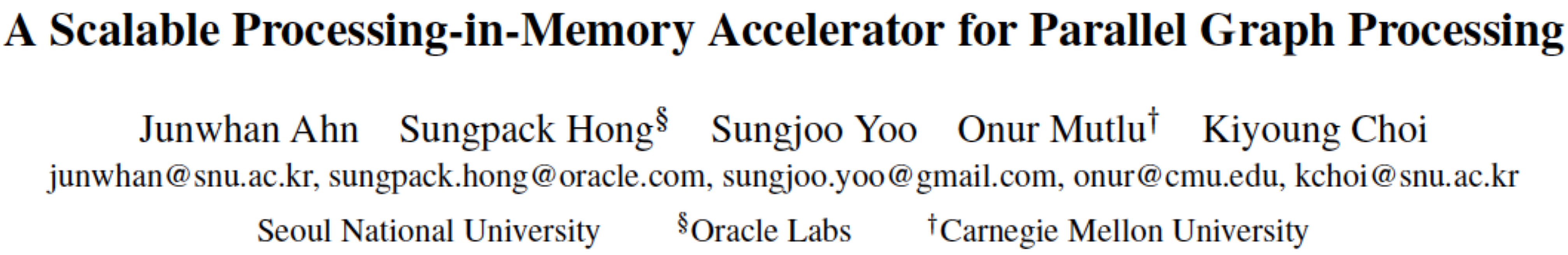


> 8X Energy Reduction

Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

More on Tesseract

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi,
"A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing"
*Proceedings of the 42nd International Symposium on Computer Architecture (**ISCA**)*, Portland, OR, June 2015.
[Slides (pdf)] [Lightning Session Slides (pdf)]



A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Junwhan Ahn Sungpack Hong[§] Sungjoo Yoo Onur Mutlu[†] Kiyoung Choi
junwhan@snu.ac.kr, sungpack.hong@oracle.com, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr
Seoul National University [§]Oracle Labs [†]Carnegie Mellon University

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?

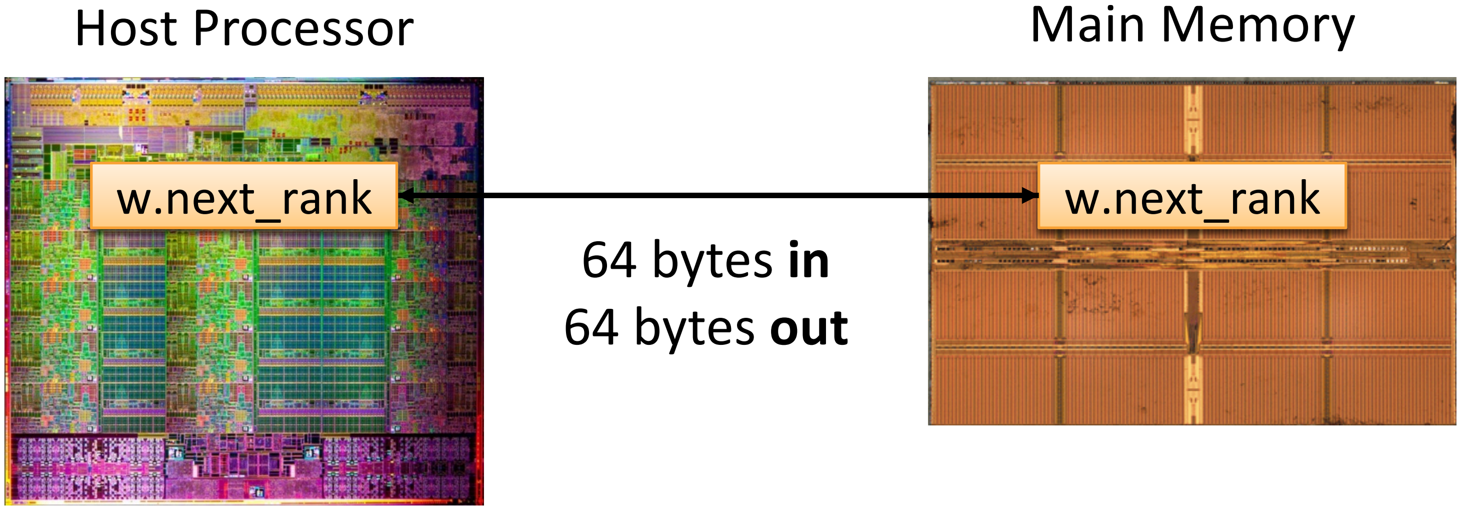
- What is the minimal processing-in-memory support we can provide?
  - without changing the system significantly
  - while achieving significant benefits

PIM-Enabled Instructions for Graph Processing

Simple PIM Operations as ISA Extensions (I)

PageRank algorithm (Page et al. 1999)

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



Host Processor

Main Memory

w.next_rank

w.next_rank

64 bytes **in**
64 bytes **out**

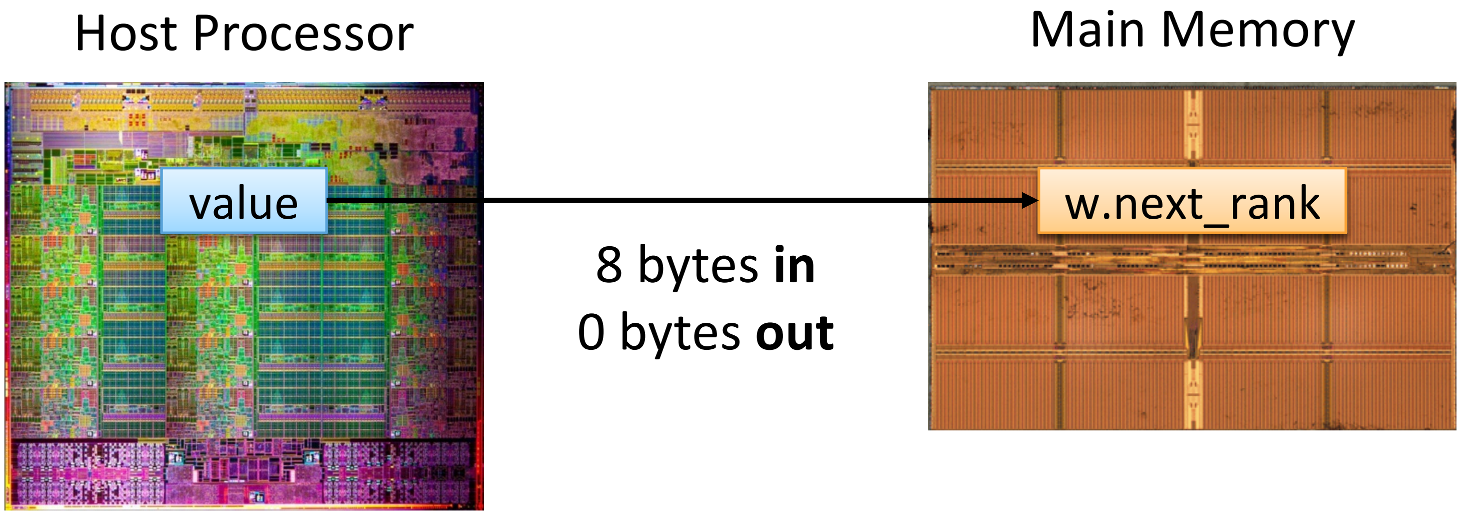
Conventional Architecture

Simple PIM Operations as ISA Extensions (II)

PageRank algorithm (Page et al. 1999)

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

pim.add r1, (r2)



Host Processor

Main Memory

value

w.next_rank

8 bytes **in**
0 bytes **out**

In-Memory Addition

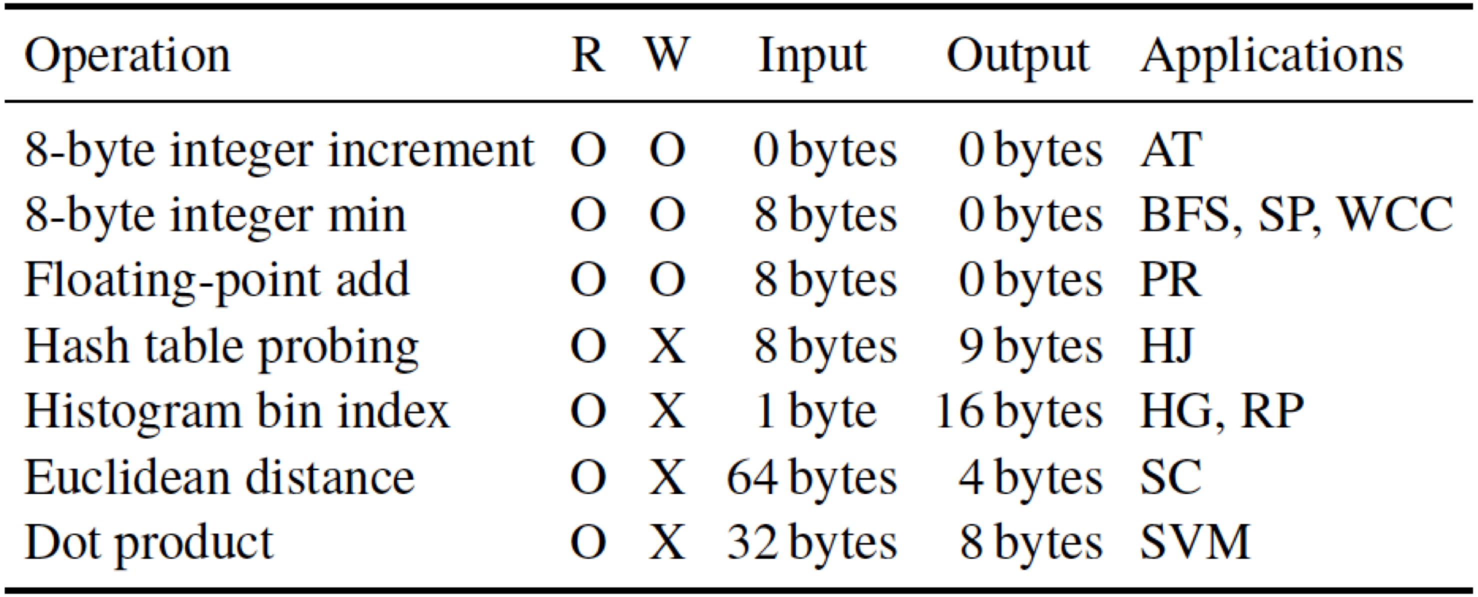
PEI: Benchmarks

- 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

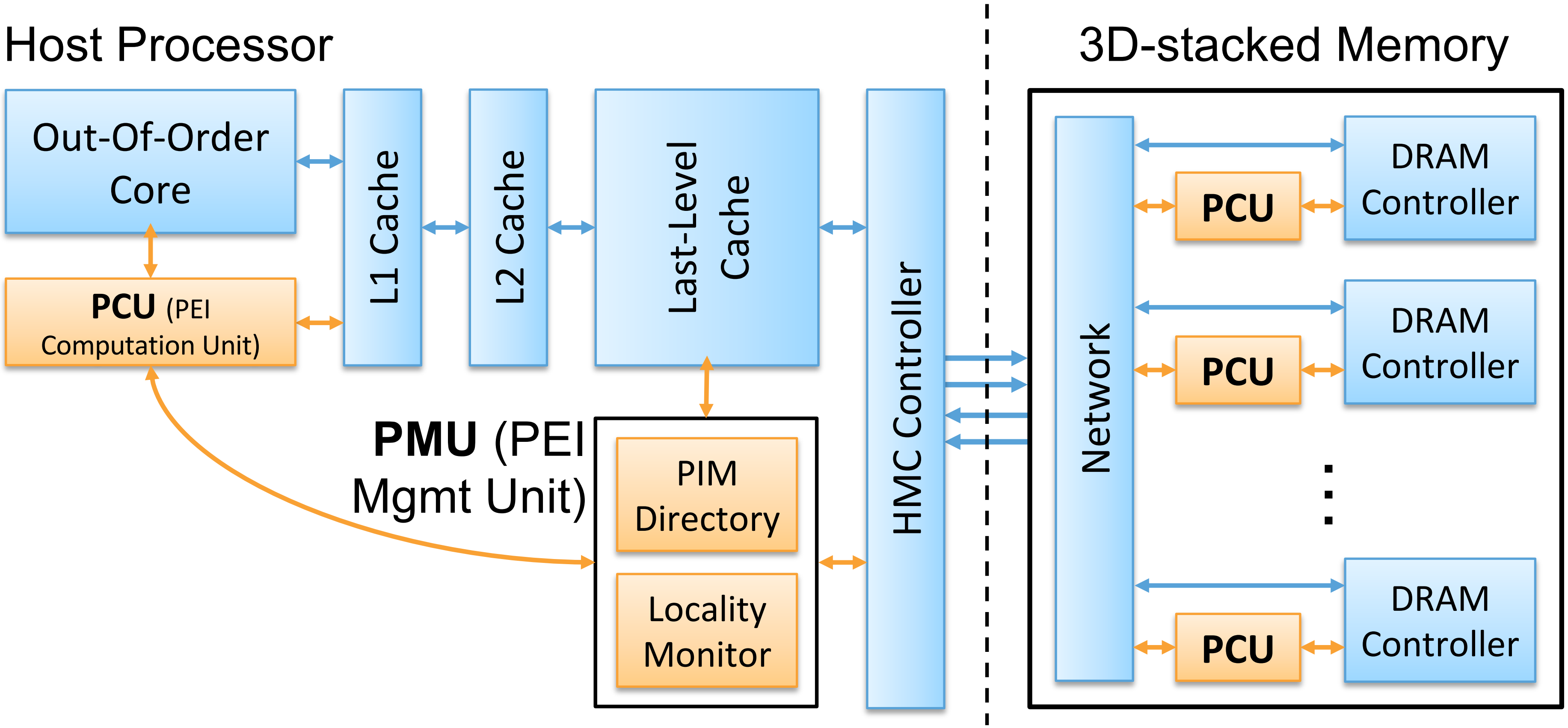
Table 1: Summary of Supported PIM Operations



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

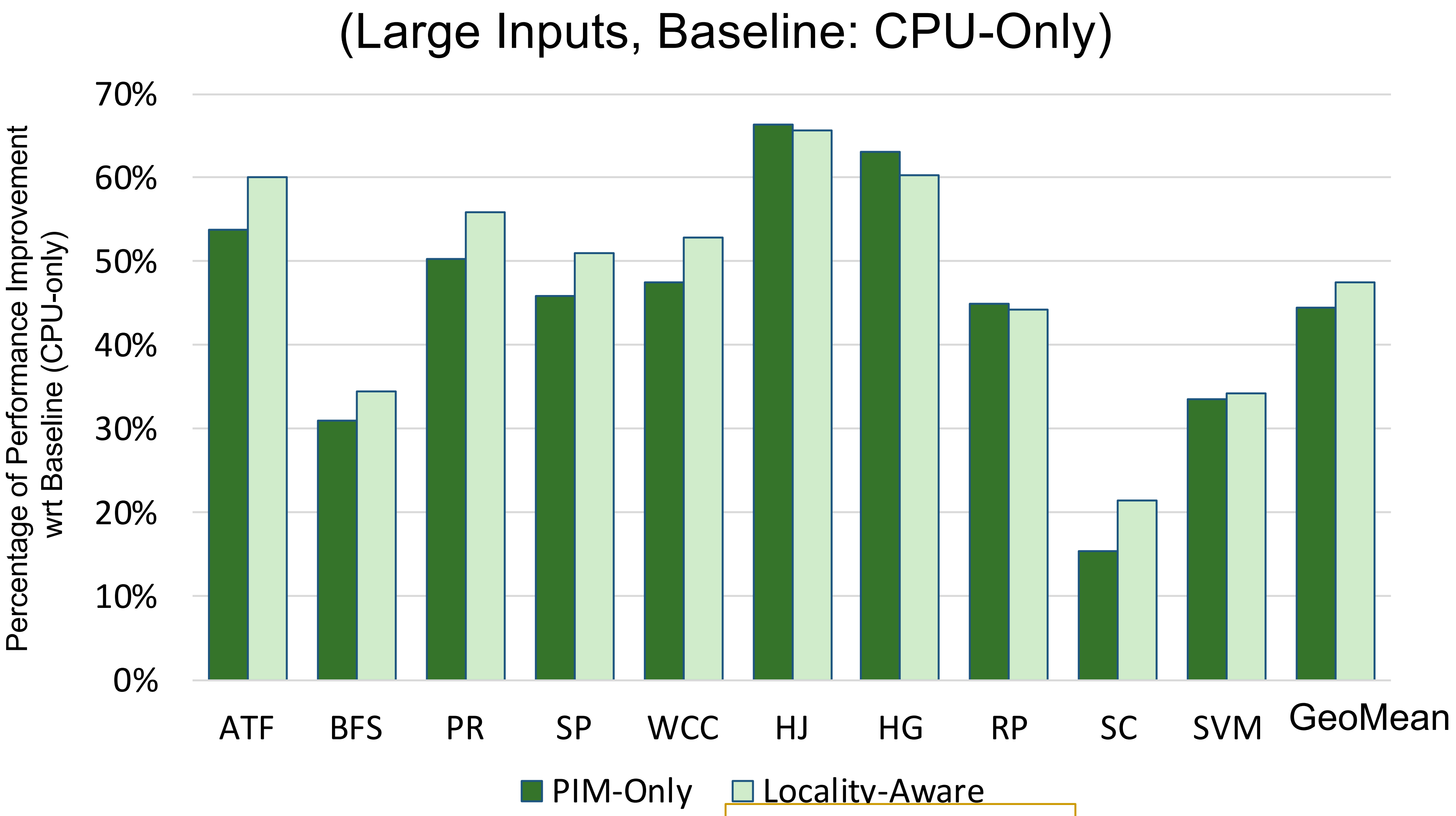
- Executed either in memory or in the processor: dynamic decision
  - Low-cost locality monitoring for a single instruction
- Cache-coherent, virtually-addressed, single cache block only
- Atomic between different PEIs
- *Not* atomic with normal instructions (use *pfence* for ordering)

Example PEI Microarchitecture



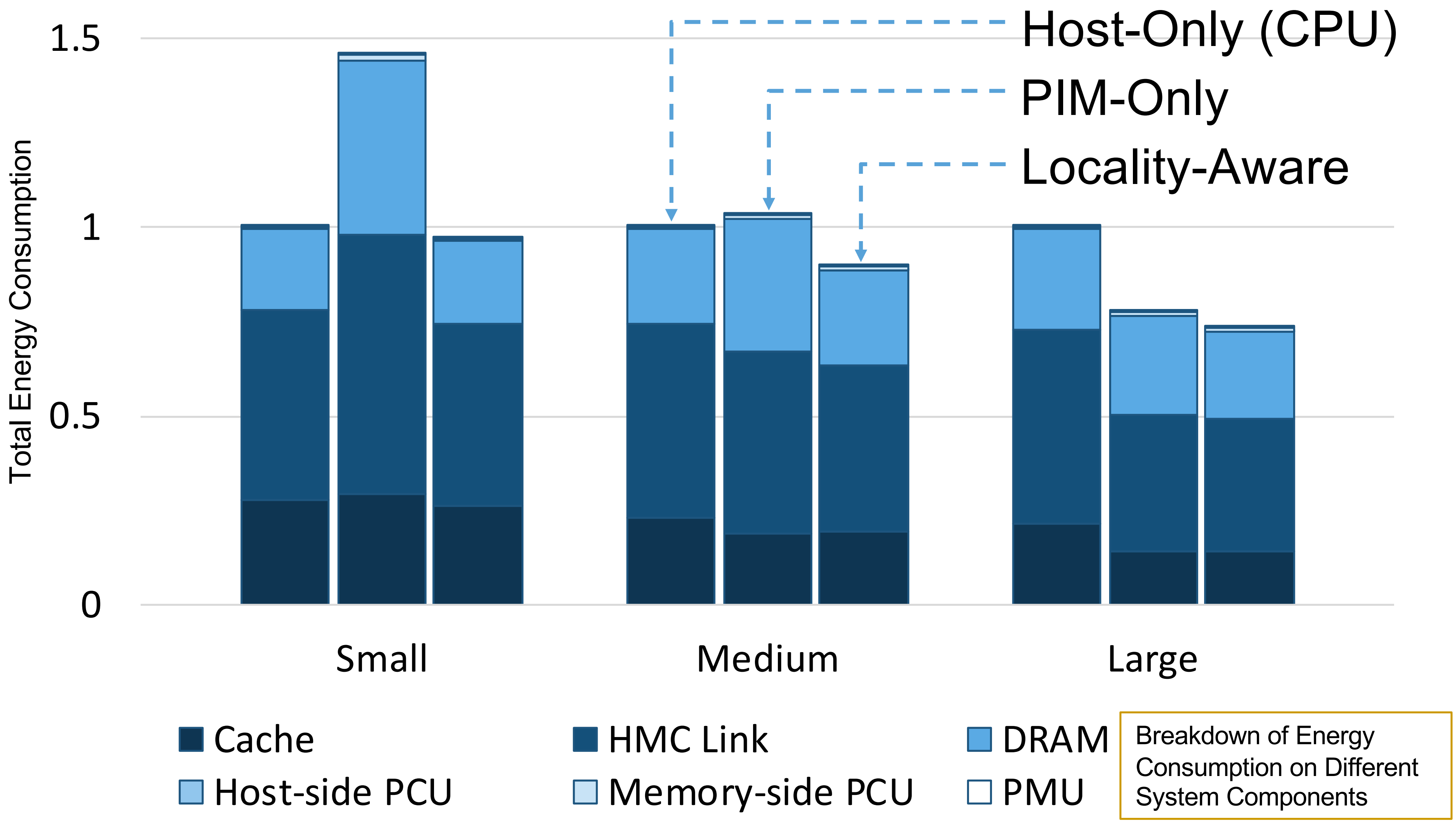
Example PEI uArchitecture

PEI Performance Delta: Large Data Sets



Locality-Aware = PIM or CPU depending on data location

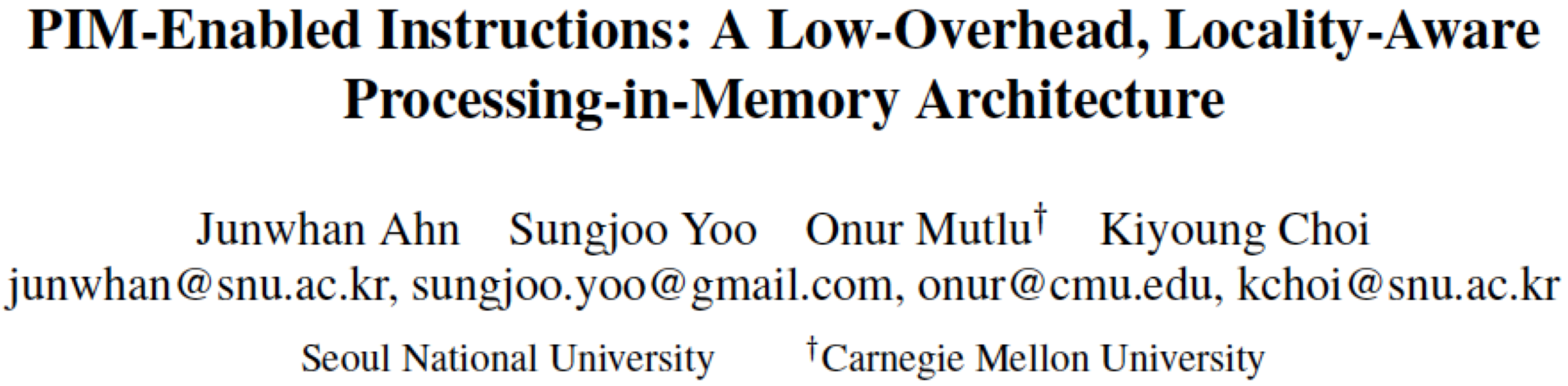
PEI Energy Consumption



Breakdown of Energy Consumption on Different System Components

More on PIM-Enabled Instructions

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi, **"PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture"** *Proceedings of the 42nd International Symposium on Computer Architecture (**ISCA**)*, Portland, OR, June 2015. [Slides (pdf)] [Lightning Session Slides (pdf)]



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

Junwhan Ahn Sungjoo Yoo Onur Mutlu[†] Kiyoung Choi
junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr
Seoul National University [†]Carnegie Mellon University

Agenda

- Major Trends Affecting Memory

- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips

  - Exploiting 3D-Stacked Memory

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 2020

1 October 2020