Memory Systems and Memory-Centric Computing

Lecture 2: Memory-Centric Computing I

Onur Mutlu

omutlu@gmail.com

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

16 July 2024

HiPEAC ACACES Summer School 2024

SAFARI ETH zürich

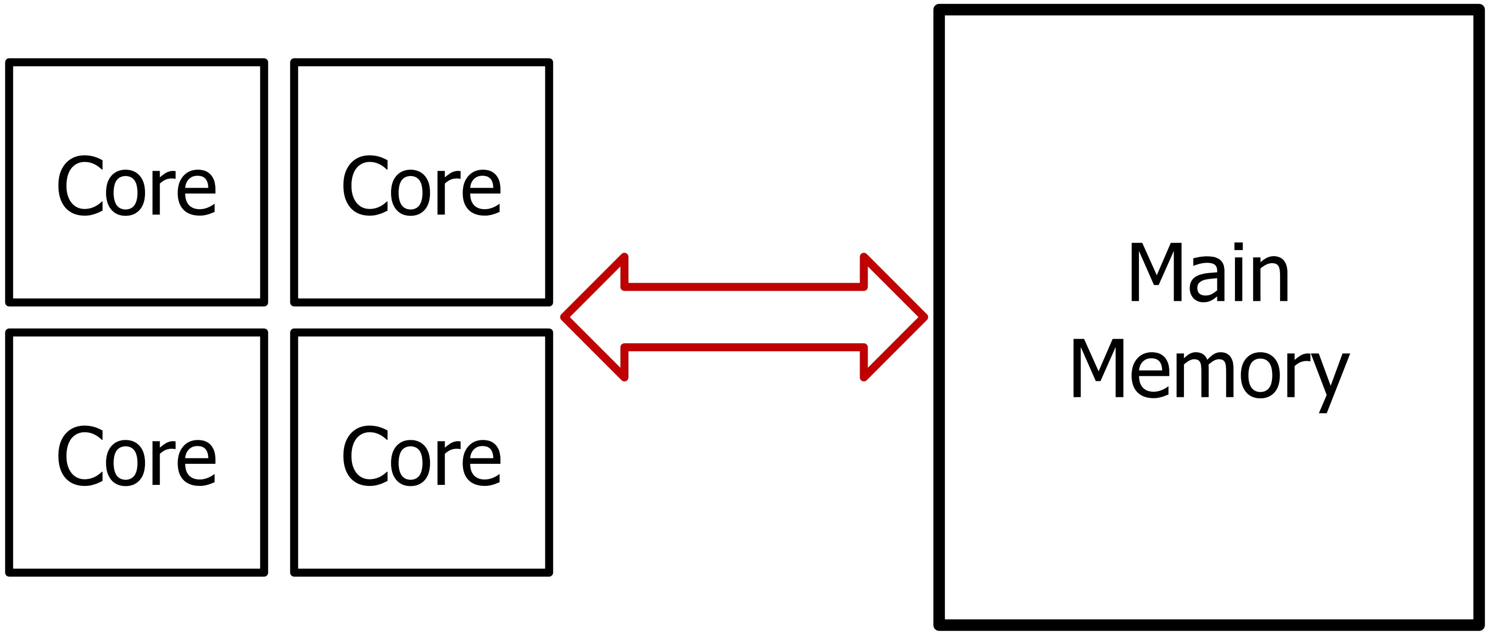
Agenda For Today

- Memory Systems and Memory-Centric Computing
  - July 15-19, 2024

- Topic 1: Memory Trends, Challenges, Opportunities, Basics
- Topic 2: Memory-Centric Computing
- Topic 3: Memory Robustness: RowHammer, RowPress & Beyond
- Topic 4: Machine Learning Driven Memory Systems
- Topic 5 (another course): Architectures for Genomics and ML
- Topic 6 (unlikely): Non-Volatile Memories and Storage
- Topic 7 (unlikely): Memory Latency, Predictability & QoS

- Major Overview Reading:
  - Mutlu et al., "A Modern Primer on Processing in Memory," Book Chapter on Emerging Computing and Devices, 2022.

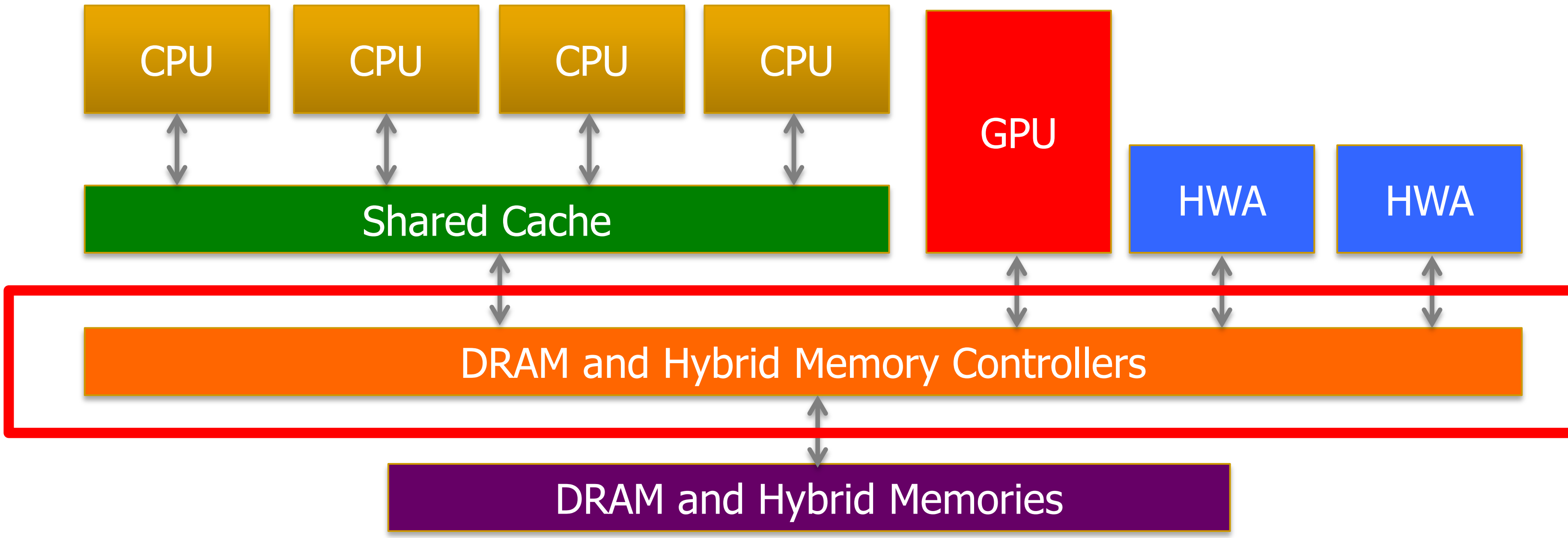
An Orthogonal Issue: Memory Interference



Cores' interfere with each other when accessing shared main memory

Uncontrolled interference leads to many problems (QoS, performance)

Goal: Predictable Performance in Complex Systems



- Heterogeneous agents: CPUs, GPUs, and HWAs
- Main memory interference between CPUs, GPUs, HWAs

How to allocate resources to heterogeneous agents to mitigate interference and provide predictable performance?

Many goals, many constraints, many metrics ...

The Future

Memory Controllers

are critical to research

They will become

even more important

Memory Control w/ Machine Learning **[ISCA'08]**

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

Self-Optimizing Memory Controllers: A Reinforcement Learning Approach

Engin İpek[1,2] **Onur Mutlu**[2] **José F. Martínez**[1] **Rich Caruana**[1]

[1]Cornell University, Ithaca, NY 14850 USA
[2] Microsoft Research, Redmond, WA 98052 USA

Solving the Memory Problem

How Do We Solve **The Memory Problem?**

- Fix it: Make memory and controllers more intelligent
  - New interfaces, functions, architectures: system-mem codesign

- Eliminate or minimize it: Replace or (more likely) augment DRAM with a different technology
  - New technologies and system-wide rethinking of memory & storage

- Embrace it: Design heterogeneous memories (none of which are perfect) and map data intelligently across them
  - New models for data management and maybe usage

- …

How Do We Solve **The Memory Problem?**

- Fix it: Make memory and controllers more intelligent
  - New interfaces, functions, architectures: system-mem codesign

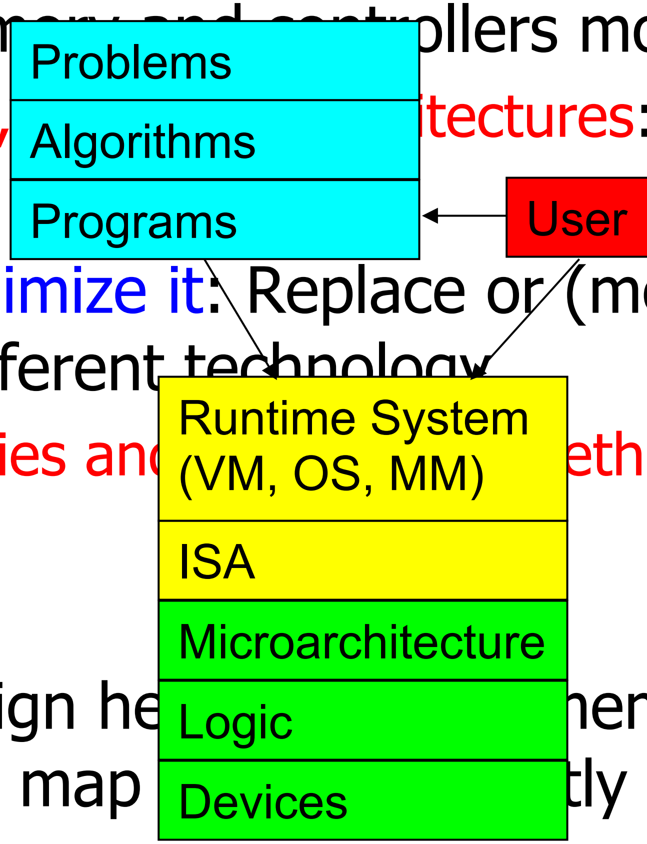
- Eliminate or minimize it: Replace or (more likely) augment DRAM with a different technology
  - New technologies and system-wide rethinking of memory & storage

- Embrace it: Design heterogeneous memories (none of which are perfect) and map data intelligently across them
  - New models for data management and maybe usage

Solutions (to memory scaling) require software/hardware/device cooperation

How Do We Solve **The Memory Problem?**

- Fix it: Make mem ollers more intelligent
  - New interfaces, itectures: system-mem codesign
- Eliminate or minimize it: Replace or (more likely) augment DRAM with a different technology
  - New technologies and ethinking of memory & storage
- Embrace it: Design he memories (none of which are perfect) and map tly across them
  - New models for data management and maybe usage



Solutions (to memory scaling) require software/hardware/device cooperation

Solution 1: New Memory Architectures

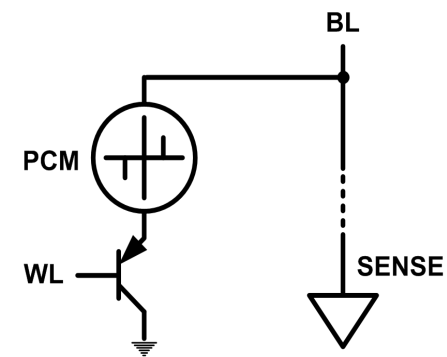
- Overcome memory shortcomings with
  - Memory-centric system design
  - Novel memory architectures, interfaces, functions
  - Better waste management (efficient utilization)

- Key issues to tackle
  - Enable reliability at low cost → high capacity
  - Reduce energy
  - Reduce latency
  - Improve bandwidth
  - Reduce waste (capacity, bandwidth, latency)
  - Enable computation close to data

Solution 2: Emerging Memory Technologies

- Some emerging **resistive** memory technologies seem more scalable than DRAM (and they are non-volatile)

- Example: Phase Change Memory
  - Data stored by changing phase of material
  - Data read by detecting material's resistance
  - Expected to scale to 9nm (2022 [ITRS 2009])
  - Prototyped at 20nm (Raoux+, IBM JRD 2008)
  - Expected to be denser than DRAM: can store multiple bits/cell



- But, emerging technologies have (many) shortcomings
  - Can they be enabled to replace/augment/surpass DRAM?

Reading: PCM as Main Memory: Idea in 2009

- Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger, **"Architecting Phase Change Memory as a Scalable DRAM Alternative"** *Proceedings of the 36th International Symposium on Computer Architecture (**ISCA**)*, pages 2-13, Austin, TX, June 2009. Slides (pdf)
  One of the 13 computer architecture papers of 2009 selected as Top Picks by IEEE Micro. Selected as a CACM Research Highlight.
  2022 Persistent Impact Prize.

Architecting Phase Change Memory as a Scalable DRAM Alternative

Benjamin C. Lee† Engin Ipek† Onur Mutlu‡ Doug Burger†

†Computer Architecture Group
Microsoft Research
Redmond, WA
{blee, ipek, dburger}@microsoft.com

‡Computer Architecture Laboratory
Carnegie Mellon University
Pittsburgh, PA
onur@cmu.edu

Reading: More on PCM As Main Memory

- Benjamin C. Lee, Ping Zhou, Jun Yang, Youtao Zhang, Bo Zhao, Engin Ipek, Onur Mutlu, and Doug Burger, **"Phase Change Technology and the Future of Main Memory"** *IEEE Micro, Special Issue: Micro's Top Picks from 2009 Computer Architecture Conferences (**MICRO TOP PICKS**)*, Vol. 30, No. 1, pages 60-70, January/February 2010.

PHASE-CHANGE TECHNOLOGY AND THE FUTURE OF MAIN MEMORY

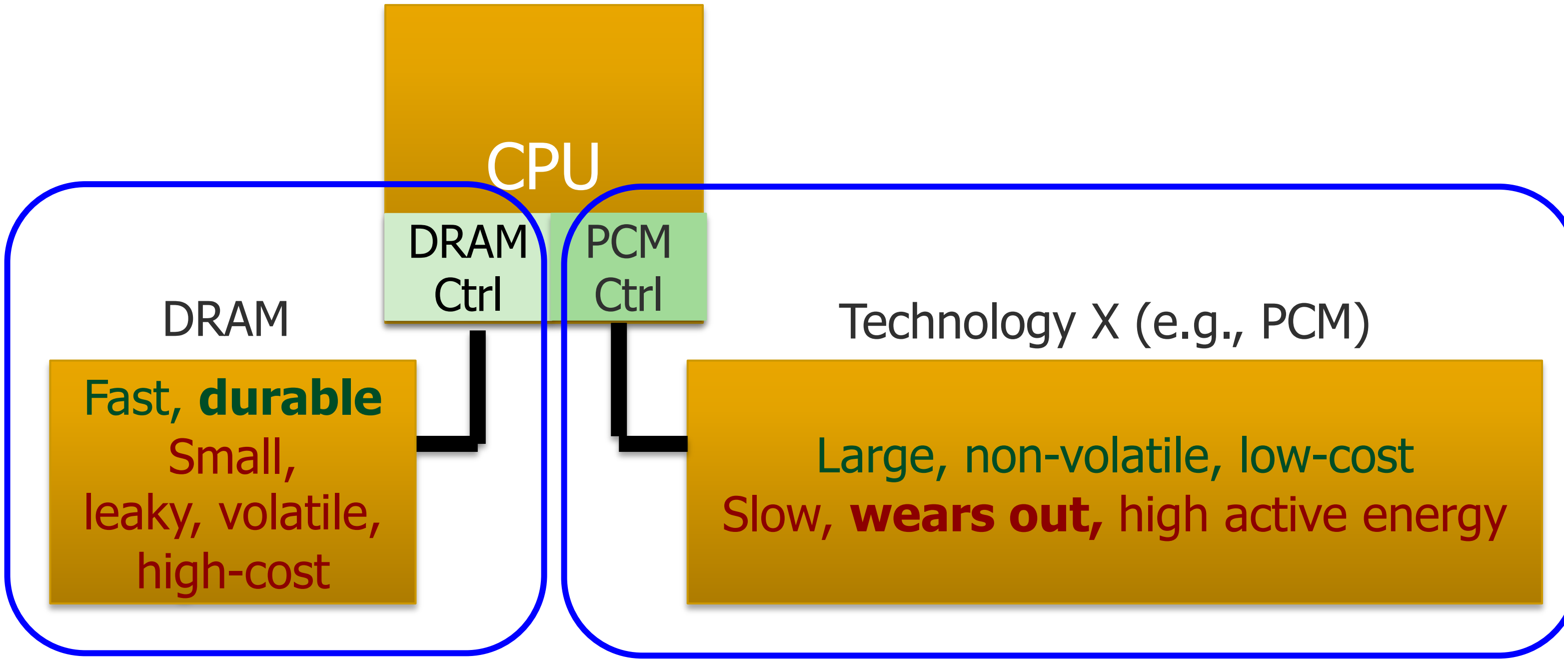
Intel Optane Persistent Memory (2019)

- Non-volatile main memory
- Based on 3D-XPoint Technology

Solution 2: Emerging Memory Technologies

- Lee+, "Architecting Phase Change Memory as a Scalable DRAM Alternative," ISCA'09, CACM'10, IEEE Micro'10.
- Meza+, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters 2012.
- Yoon, Meza+, "Row Buffer Locality Aware Caching Policies for Hybrid Memories," ICCD 2012.
- Kultursay+, "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," ISPASS 2013.
- Meza+, "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory," WEED 2013.
- Lu+, "Loose Ordering Consistency for Persistent Memory," ICCD 2014.
- Zhao+, "FIRM: Fair and High-Performance Memory Control for Persistent Memory Systems," MICRO 2014.
- Yoon, Meza+, "Efficient Data Mapping and Buffering Techniques for Multi-Level Cell Phase-Change Memories," TACO 2014.
- Ren+, "ThyNVM: Enabling Software-Transparent Crash Consistency in Persistent Memory Systems," MICRO 2015.
- Chauhan+, "NVMove: Helping Programmers Move to Byte-Based Persistence," INFLOW 2016.
- Li+, "Utility-Based Hybrid Memory Management," CLUSTER 2017.
- Yu+, "Banshee: Bandwidth-Efficient DRAM Caching via Software/Hardware Cooperation," MICRO 2017.
- Tavakkol+, "MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices," FAST 2018.
- Tavakkol+, "FLIN: Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives," ISCA 2018.
- Sadrosadati+. "LTRF: Enabling High-Capacity Register Files for GPUs via Hardware/Software Cooperative Register Prefetching," ASPLOS 2018.
- Salkhordeh+, "An Analytical Model for Performance and Lifetime Estimation of Hybrid DRAM-NVM Main Memories," TC 2019.
- Wang+, "Panthera: Holistic Memory Management for Big Data Processing over Hybrid Memories," PLDI 2019.
- Song+, "Enabling and Exploiting Partition-Level Parallelism (PALP) in Phase Change Memories," CASES 2019.
- Liu+, "Binary Star: Coordinated Reliability in Heterogeneous Memory Systems for High Performance and Scalability," MICRO'19.
- Song+, "Improving Phase Change Memory Performance with Data Content Aware Access," ISMM 2020.
- Yavits+, "WoLFRaM: Enhancing Wear-Leveling and Fault Tolerance in Resistive Memories using Programmable Address Decoders," ICCD 2020.
- Song+, "Aging-Aware Request Scheduling for Non-Volatile Main Memory," ASP-DAC 2021.

Combination: Hybrid Memory Systems

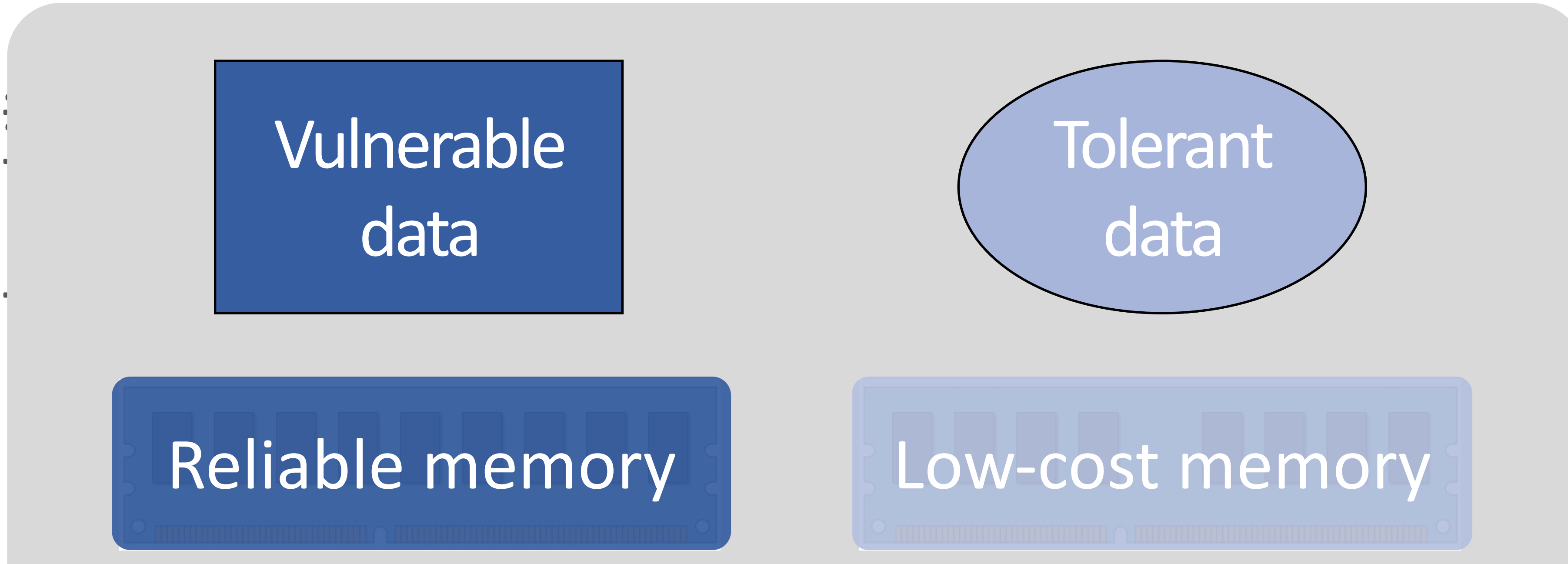


Hardware/software manage data allocation and movement
to achieve the best of multiple technologies

Meza+, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters, 2012.

Yoon, Meza et al., "Row Buffer Locality Aware Caching Policies for Hybrid Memories," ICCD 2012 Best Paper Award.

Exploiting Memory Error Tolerance with Hybrid Memory Systems



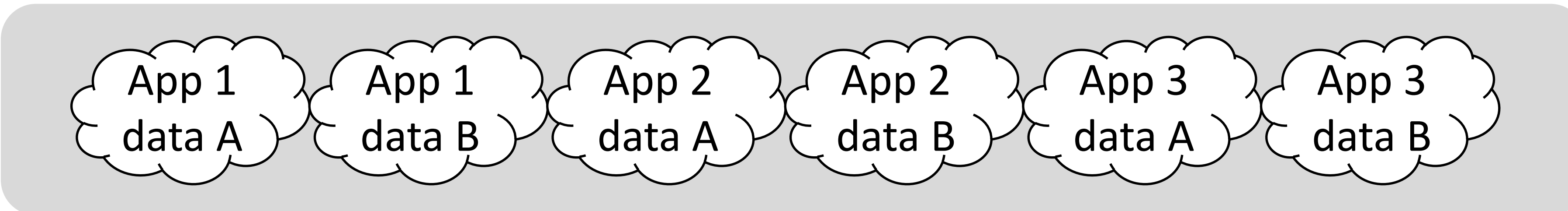
On Microsoft's Web Search workload

Reduces server hardware cost by 4.7 %

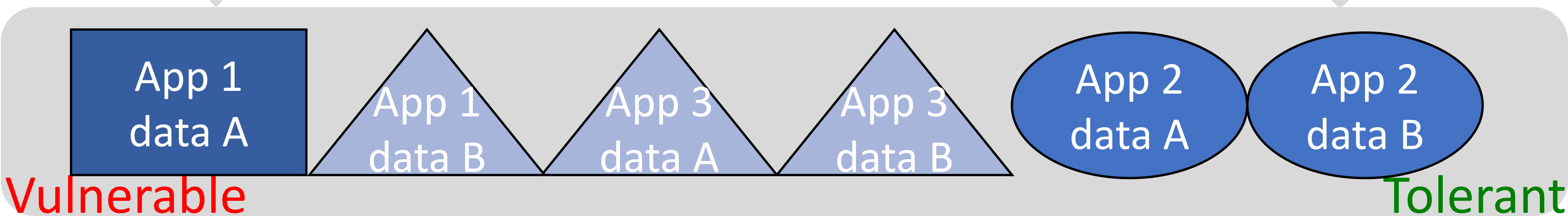
Achieves single server availability target of 99.90 %

Heterogeneous-**R**eliability **M**emory [DSN 2014]

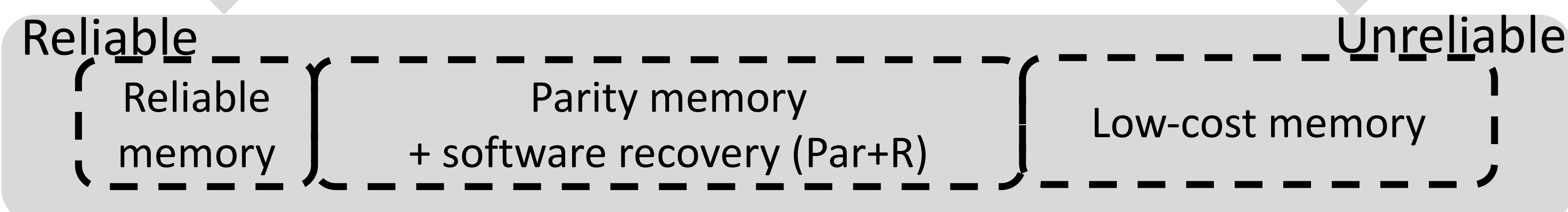
Heterogeneous-Reliability Memory



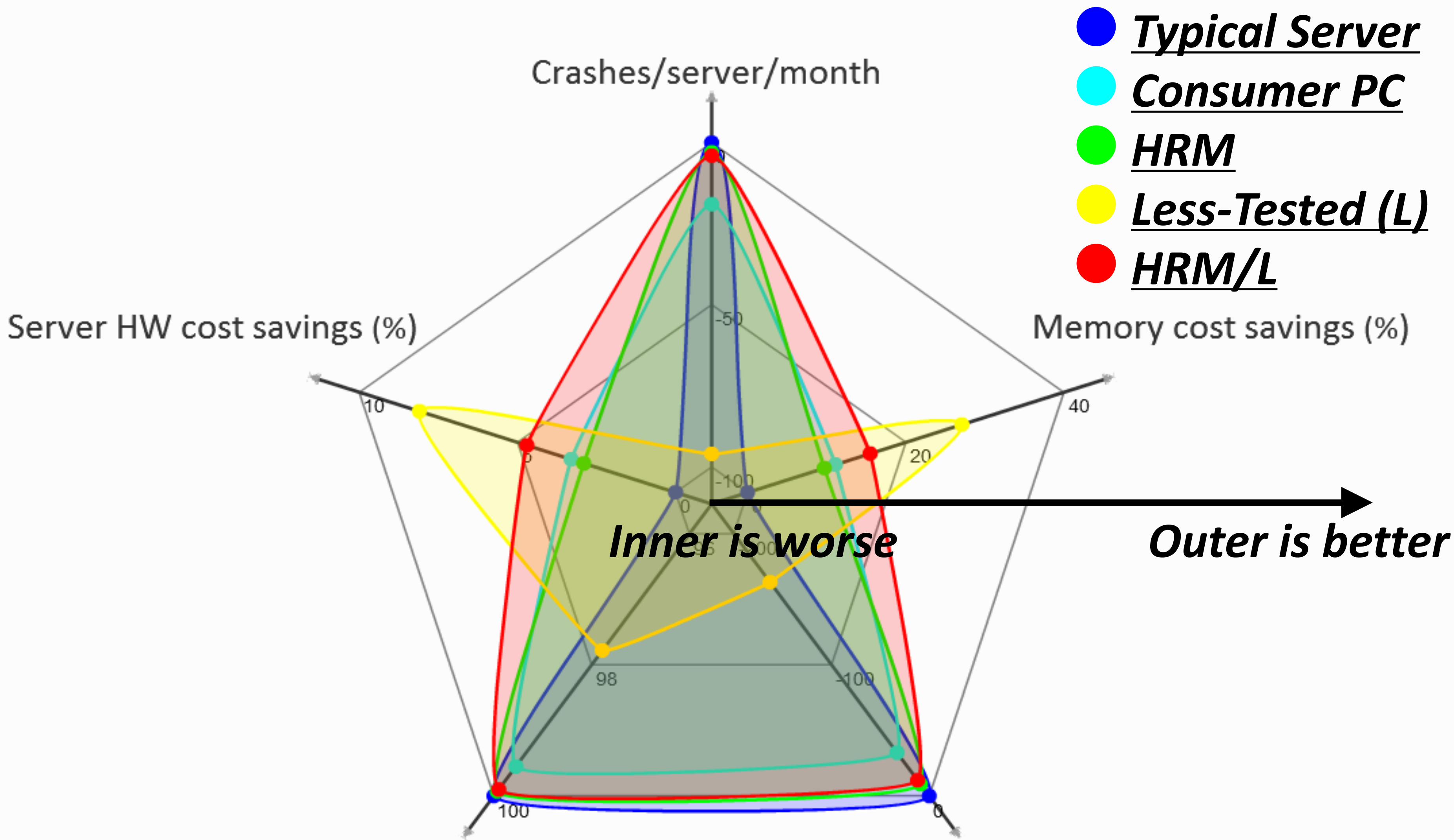
Step 1: Characterize and classify application memory error tolerance



Step 2: Map application data to the *HRM* system enabled by *SW/HW cooperative solutions*



Evaluation Results



Bigger area means better tradeoff

More on Heterogeneous Reliability Memory

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

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

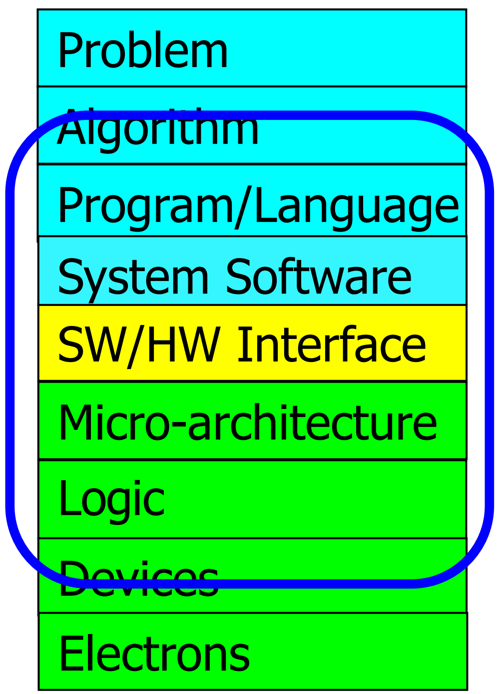
Yixin Luo Sriram Govindan* Bikash Sharma* Mark Santaniello* Justin Meza
Aman Kansal* Jie Liu* Badriddine Khessib* Kushagra Vaid* Onur Mutlu
Carnegie Mellon University, yixinluo@cs.cmu.edu, {meza, onur}@cmu.edu
*Microsoft Corporation, {srgovin, bsharma, marksan, kansal, jie.liu, bkhessib, kvaid}@microsoft.com

HRM is an Example of Our Axiom

To achieve the highest energy efficiency and performance:

we must take the expanded view

of computer architecture



Co-design across the hierarchy: Algorithms to devices

Specialize as much as possible within the design goals

Another Example: EDEN for DNNs

- Deep Neural Network evaluation is very DRAM-intensive (especially for large networks)

1. Some data and layers in DNNs are very tolerant to errors

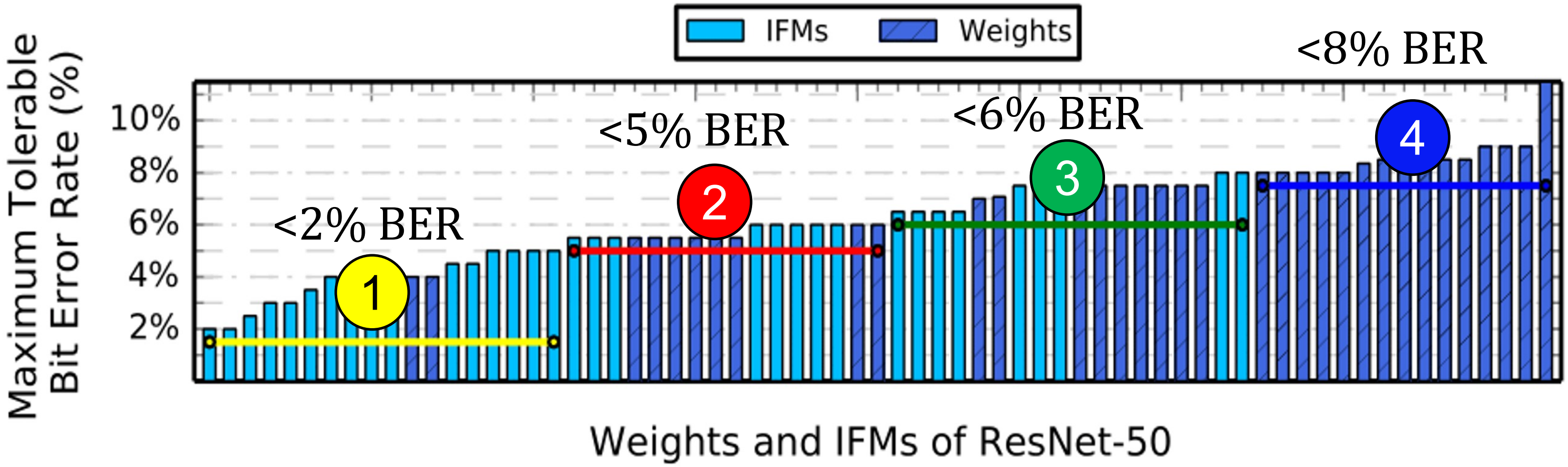
2. Map such data and layers to low-energy low-latency DRAM (which may have more errors)

3. While still achieving a user-specified DNN accuracy target by making training DRAM-error-aware

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

Example DNN Data Type to DRAM Mapping

Mapping example of ResNet-50:



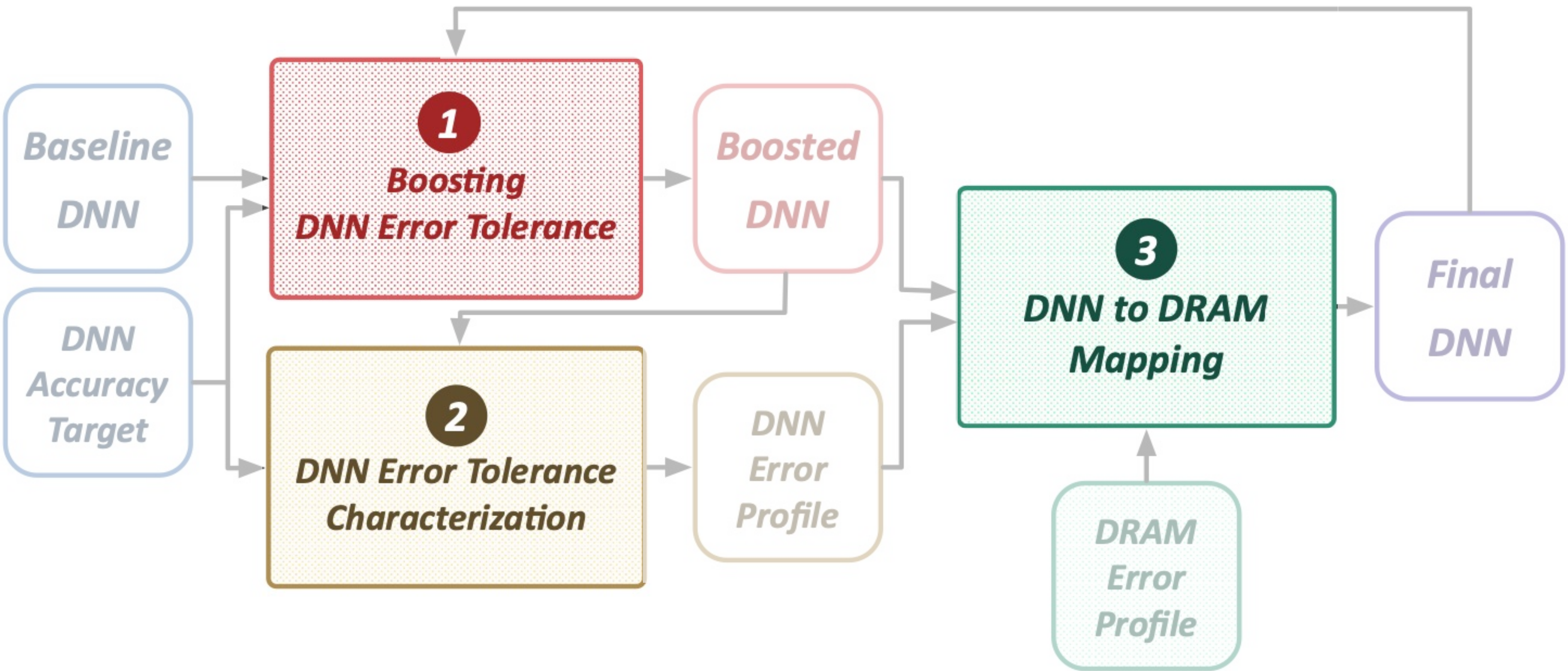
Map more error-tolerant DNN layers
to DRAM partitions **with lower voltage/latency**

4 DRAM partitions with different error rates

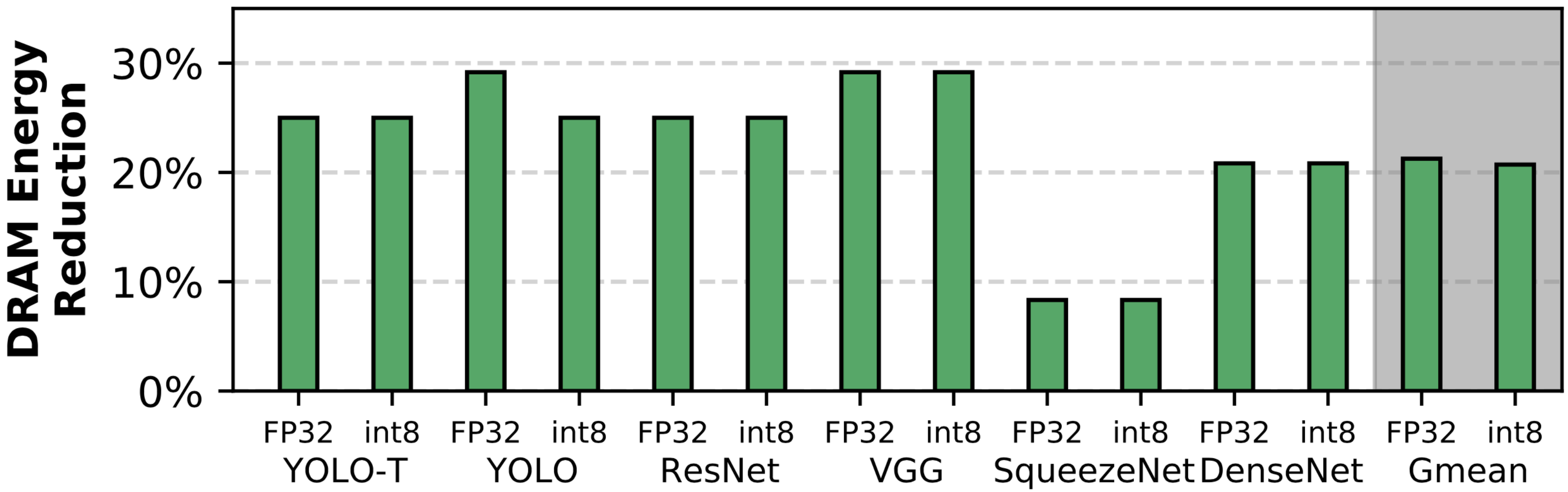
EDEN: Overview

Key idea: Enable **accurate, efficient** DNN inference using **approximate DRAM**

EDEN is an **iterative** process that has **3 key steps**



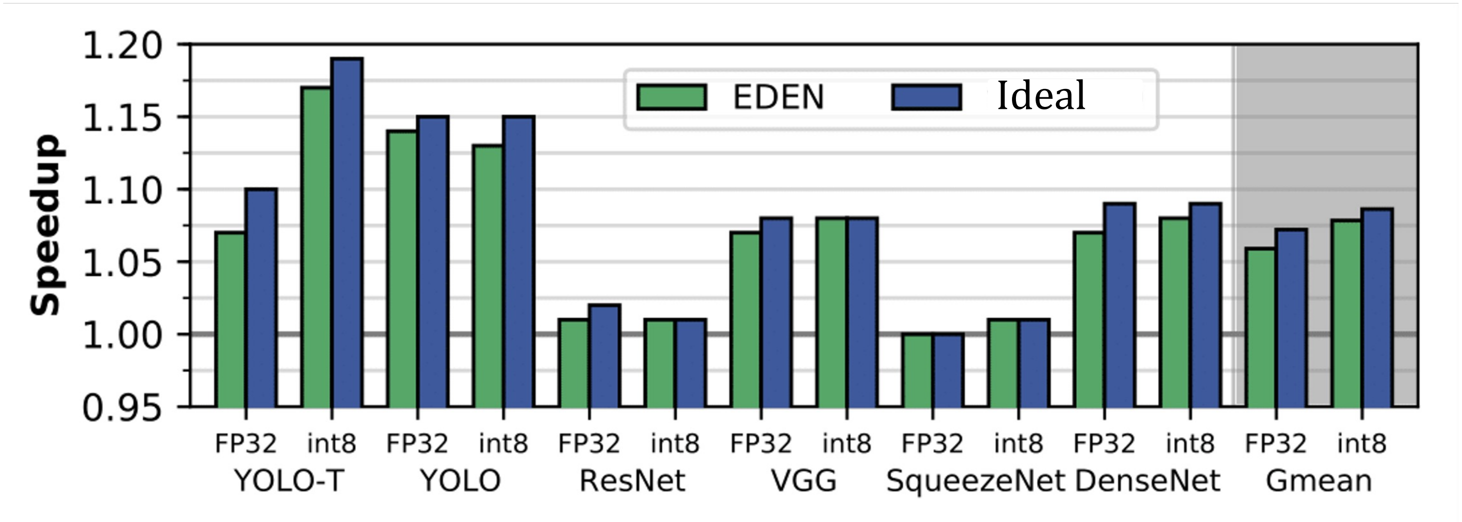
CPU: DRAM Energy Evaluation



Average 21% DRAM energy reduction

maintaining accuracy within 1% of original

CPU: Performance Evaluation



Average 8% system speedup

Some workloads achieve **17% speedup**

EDEN achieves **close to the ideal** speedup possible via row activation latency reduction

GPU, Eyeriss, and TPU: Energy Evaluation

- **GPU:** average **37% energy reduction**
- **Eyeriss**: average **31% energy reduction**
- **TPU**: average **32% energy reduction**

EDEN: Data-Aware Efficient DNN Inference

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

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

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

Data-Aware Architectures

Computing Architectures Today …

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

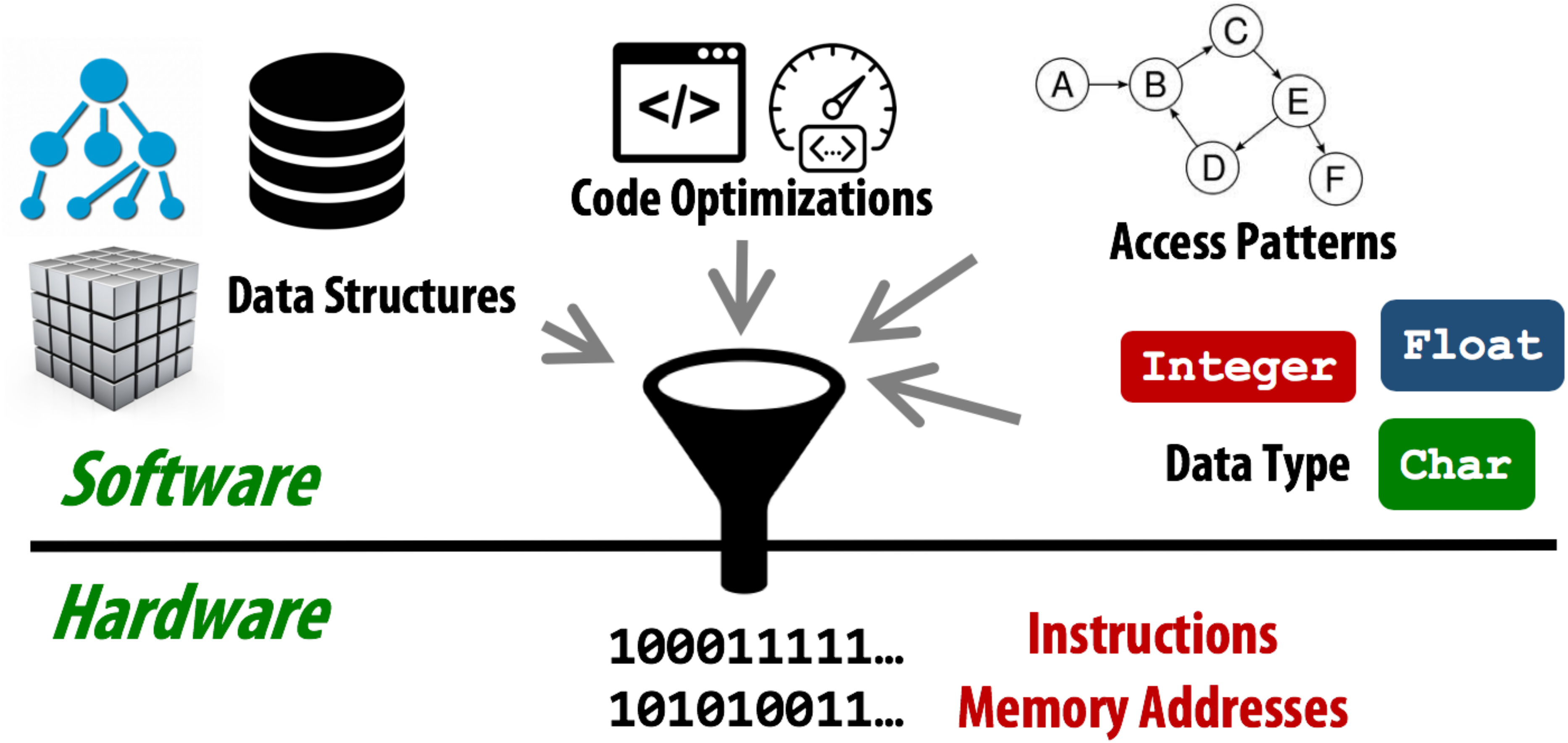
Data-Aware Architectures

- A data-aware architecture understands what it can do with and to each piece of data

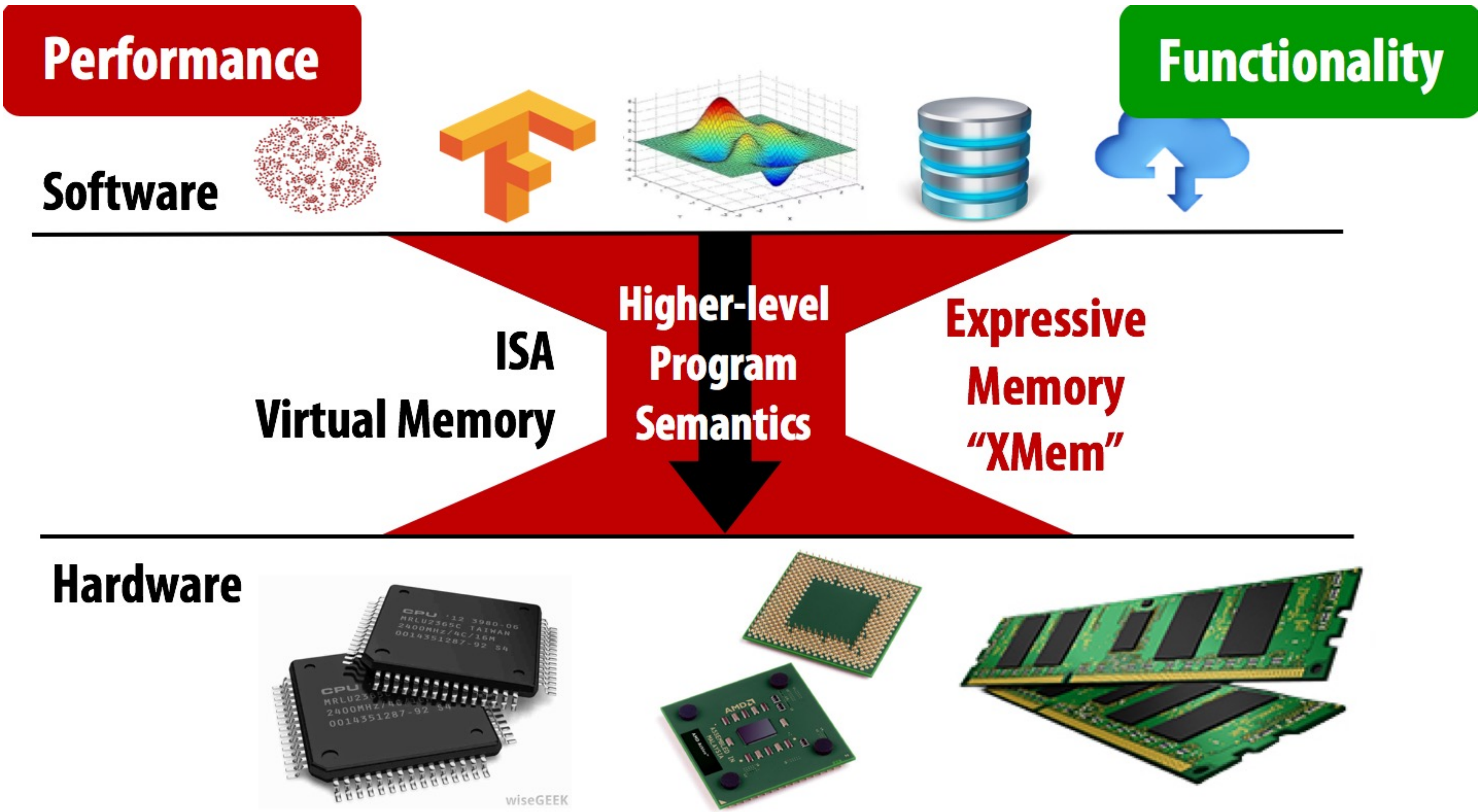
- It makes use of different properties of data to improve performance, efficiency and other metrics
  - Compressibility
  - Approximability
  - Locality
  - Sparsity
  - Criticality for Computation X
  - Access Semantics
  - Security, Privacy
  - ...

One Problem: Limited Expressiveness

Higher-level information is not visible to HW



A Solution: More Expressive Interfaces



Expressive (Memory) Interfaces

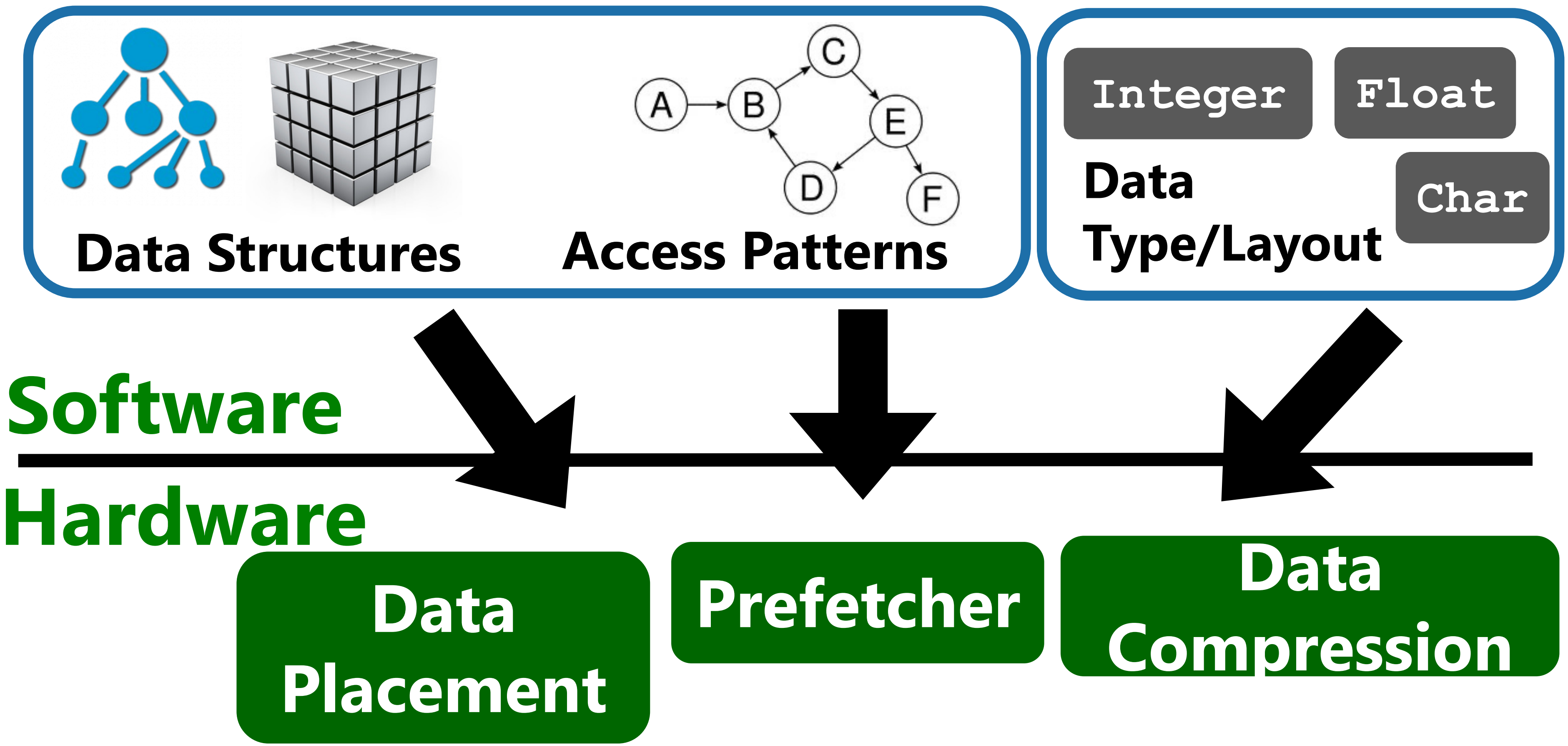
- Nandita Vijaykumar, Abhilasha Jain, Diptesh Majumdar, Kevin Hsieh, Gennady Pekhimenko, Eiman Ebrahimi, Nastaran Hajinazar, Phillip B. Gibbons and Onur Mutlu, **"A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory"**
  Proceedings of the 45th International Symposium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

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

Nandita Vijaykumar[†§] Abhilasha Jain[†] Diptesh Majumdar[†] Kevin Hsieh[†] Gennady Pekhimenko[‡]
Eiman Ebrahimi[ℵ] Nastaran Hajinazar[+] Phillip B. Gibbons[†] Onur Mutlu[§†]

[†]**Carnegie Mellon University** [‡]**University of Toronto** [ℵ]**NVIDIA**
[+]**Simon Fraser University** [§]**ETH Zürich**

SW provides key program information to HW



Broader goal: Enable many cross-layer optimizations

Express:

Data structures

Access semantics

Data types

Working set

Reuse

Access frequency

…

Optimizations:

Cache Management

Data Placement in DRAM

Data Compression

Approximation

DRAM Cache Management

NVM Management

NUCA/NUMA Optimizations

…

Benefits:

More efficient HW:

✓**Performance**

Reduced SW burden:

✓**Programmability**

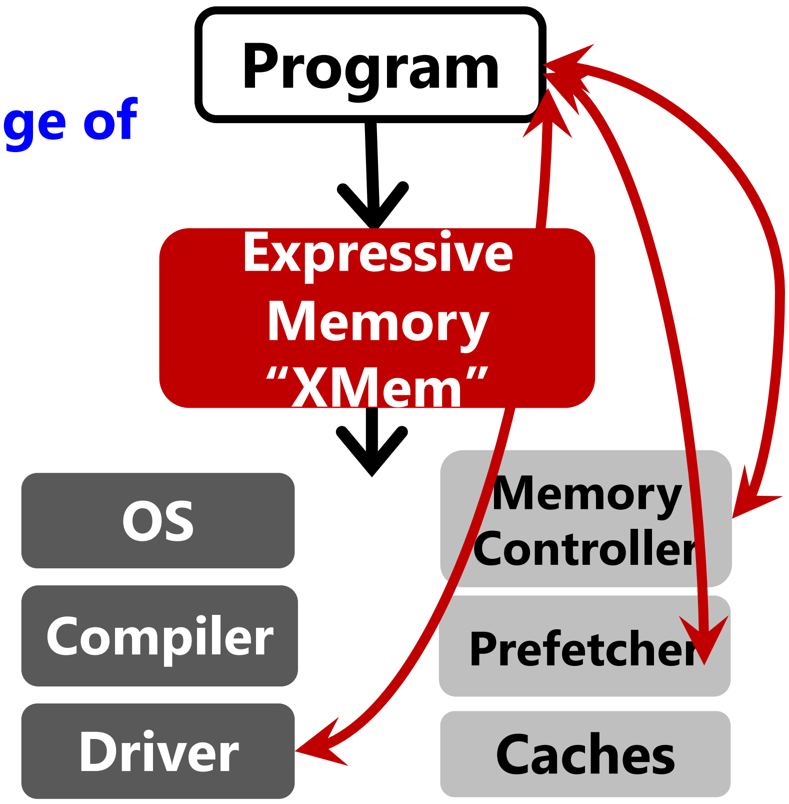
✓**Portability**

Our approach: Rich cross-layer abstractions

1. **Generality: Enable a wide range of cross-layer approaches**

2. **Minimize programmer effort**

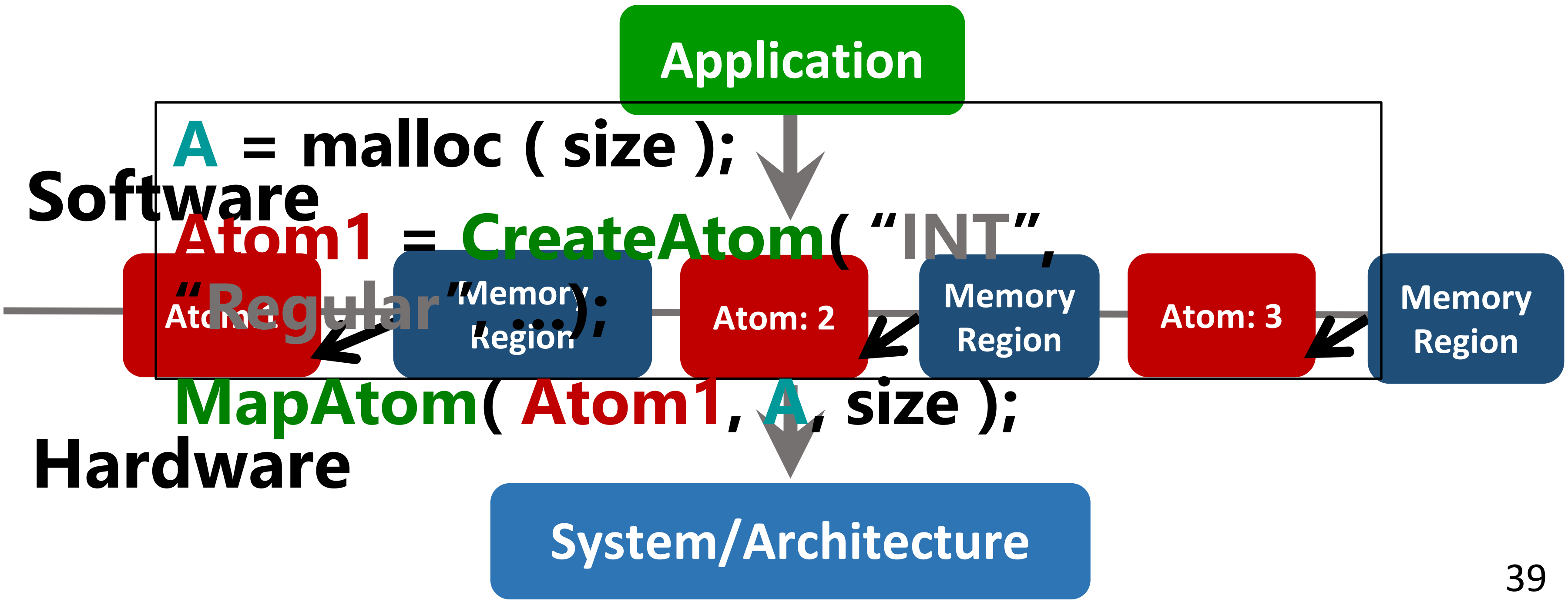
3. **Overhead**



Approach: Flexibly associate specific semantic information with any data & code

Example: XMem

- Goal: convey data semantics to the hardware enables more intelligent management of resources.
- XMem: introduces a new HW/SW abstraction, called *Atom,* for conveying data semantics



Vijaykumar+, “A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory”, ISCA 2018

XMem Aids/Enables Many Optimizations

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

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

Expressive (Memory) Interfaces

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

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

Nandita Vijaykumar[†§] Abhilasha Jain[†] Diptesh Majumdar[†] Kevin Hsieh[†] Gennady Pekhimenko[‡]
Eiman Ebrahimi[ℵ] Nastaran Hajinazar[+] Phillip B. Gibbons[†] Onur Mutlu[§†]

[†]**Carnegie Mellon University** [‡]**University of Toronto** [ℵ]**NVIDIA**
[+]**Simon Fraser University** [§]**ETH Zürich**

Expressive (Memory) Interfaces for GPUs

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

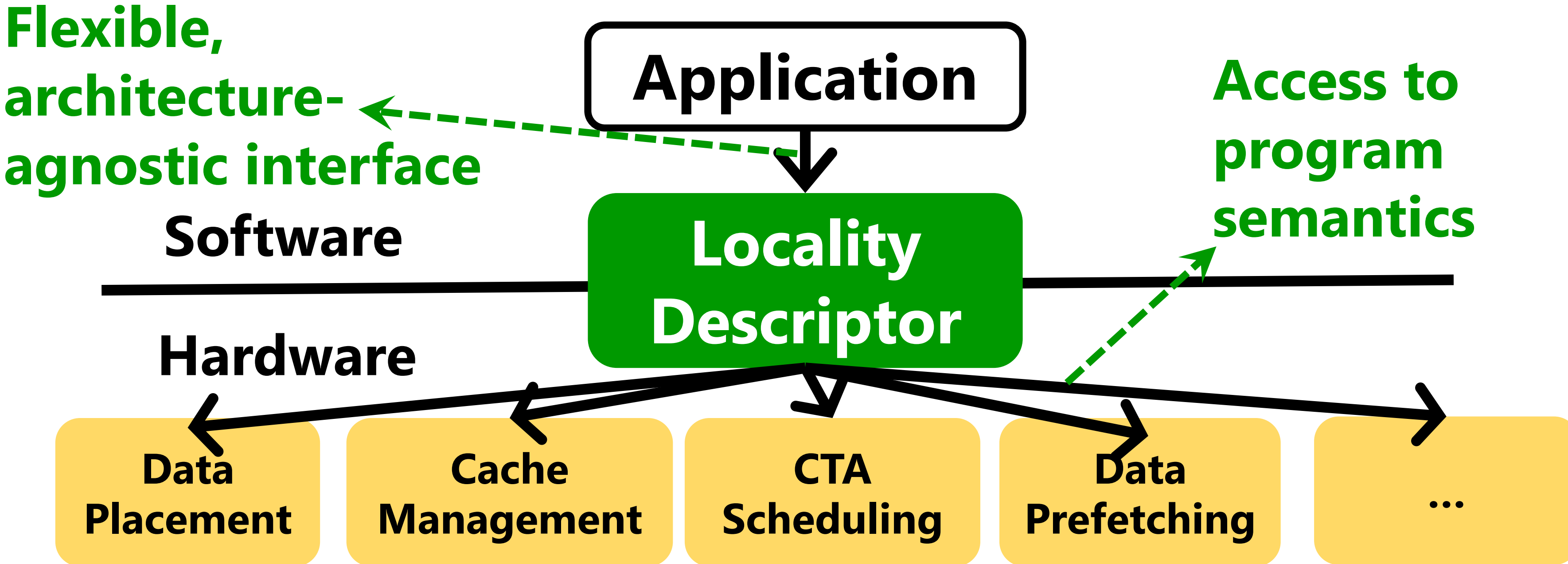
The Locality Descriptor:
A Holistic Cross-Layer Abstraction to Express Data Locality in GPUs

Nandita Vijaykumar[†§] Eiman Ebrahimi[‡] Kevin Hsieh[†]
Phillip B. Gibbons[†] Onur Mutlu[§†]

[†]**Carnegie Mellon University** [‡]**NVIDIA** [§]**ETH Zürich**

Locality Descriptor: Executive Summary

Exploiting data locality in GPUs is a challenging task



Performance Benefits:
26.6% (up to 46.6%) from cache locality
53.7% (up to 2.8x) from NUMA locality

MetaSys FPGA Prototype for Expressive Memory

- Nandita Vijaykumar, Ataberk Olgun, Konstantinos Kanellopoulos, F. Nisa Bostanci, Hasan Hassan, Mehrshad Lotfi, Phillip B. Gibbons, and Onur Mutlu,
"MetaSys: A Practical Open-source Metadata Management System to Implement and Evaluate Cross-layer Optimizations"
ACM Transactions on Architecture and Code Optimization (***TACO***), June 2022.
[arXiv version]
Presented at the 18th HiPEAC Conference, Toulouse, France, January 2023.
[Slides (pptx) (pdf)]
[Preliminary Talk Video (14 minutes)]
[SAFARI Live Seminar Video (1 hour 26 minutes)]
[MetaSys Source Code]
Best paper award at HiPEAC 2023.

MetaSys: A Practical Open-Source Metadata Management System to Implement and Evaluate Cross-Layer Optimizations

Nandita Vijaykumar[*] Ataberk Olgun[§] Konstantinos Kanellopoulos[§] Hasan Hassan[§]
Mehrshad Lotfi[§] Phillip B. Gibbons[†] Onur Mutlu[§]

[*]*University of Toronto* [§]*ETH Zürich* [†]*Carnegie Mellon University*

An Example: Heterogeneous-Reliability Memory

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

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

Yixin Luo Sriram Govindan* Bikash Sharma* Mark Santaniello* Justin Meza
Aman Kansal* Jie Liu* Badriddine Khessib* Kushagra Vaid* Onur Mutlu
Carnegie Mellon University, yixinluo@cs.cmu.edu, {meza, onur}@cmu.edu
*Microsoft Corporation, {srgovin, bsharma, marksan, kansal, jie.liu, bkhessib, kvaid}@microsoft.com

EDEN: Data-Aware Efficient DNN Inference

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

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

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

SMASH: SW/HW Indexing Acceleration

- Konstantinos Kanellopoulos, Nandita Vijaykumar, Christina Giannoula, Roknoddin Azizi, Skanda Koppula, Nika Mansouri Ghiasi, Taha Shahroodi, Juan Gomez-Luna, and Onur Mutlu,
  "SMASH: Co-designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations"
  *Proceedings of the 52nd International Symposium on Microarchitecture (**MICRO**)*, Columbus, OH, USA, October 2019.
  [Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Poster (pptx) (pdf)]
  [Lightning Talk Video (90 seconds)]
  [Full Talk Lecture (30 minutes)]

SMASH: Co-designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations

Konstantinos Kanellopoulos[1] Nandita Vijaykumar[2,1] Christina Giannoula[1,3] Roknoddin Azizi[1]
Skanda Koppula[1] Nika Mansouri Ghiasi[1] Taha Shahroodi[1] Juan Gomez Luna[1] Onur Mutlu[1,2]

[1]ETH Zürich [2]Carnegie Mellon University [3]National Technical University of Athens

Data-Aware Virtual Memory Framework

Nastaran Hajinazar, Pratyush Patel, Minesh Patel, Konstantinos Kanellopoulos, Saugata Ghose, Rachata Ausavarungnirun, Geraldo Francisco de Oliveira Jr., Jonathan Appavoo, Vivek Seshadri, and Onur Mutlu,
"The Virtual Block Interface: A Flexible Alternative to the Conventional Virtual Memory Framework"
*Proceedings of the 47th International Symposium on Computer Architecture (**ISCA**)*, Virtual, June 2020.
[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[ARM Research Summit Poster (pptx) (pdf)]
[Talk Video (26 minutes)]
[Lightning Talk Video (3 minutes)]
[Lecture Video (43 minutes)]

The Virtual Block Interface: A Flexible Alternative to the Conventional Virtual Memory Framework

Nastaran Hajinazar[★†] Pratyush Patel[⋈] Minesh Patel[★] Konstantinos Kanellopoulos[★] Saugata Ghose[‡]
Rachata Ausavarungnirun[⊙] Geraldo F. Oliveira[★] Jonathan Appavoo[◇] Vivek Seshadri[▽] Onur Mutlu[★‡]

[★]ETH Zürich [†]Simon Fraser University [⋈]University of Washington [‡]Carnegie Mellon University
[⊙]King Mongkut's University of Technology North Bangkok [◇]Boston University [▽]Microsoft Research India

SW/HW Climate Modeling Accelerator

- Gagandeep Singh, Dionysios Diamantopoulos, Christoph Hagleitner, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal,
  "NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling"
  *Proceedings of the 30th International Conference on Field-Programmable Logic and Applications (**FPL**)*, Gothenburg, Sweden, September 2020.
  [Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (23 minutes)]
  Nominated for the Stamatis Vassiliadis Memorial Award.

NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling

Gagandeep Singh[a,b,c] Dionysios Diamantopoulos[c] Christoph Hagleitner[c] Juan Gómez-Luna[b]
Sander Stuijk[a] Onur Mutlu[b] Henk Corporaal[a]
[a]Eindhoven University of Technology [b]ETH Zürich [c]IBM Research Europe, Zurich

HW/SW Time Series Analysis Accelerator

- Ivan Fernandez, Ricardo Quislant, Christina Giannoula, Mohammed Alser, Juan Gómez-Luna, Eladio Gutiérrez, Oscar Plata, and Onur Mutlu,
"NATSA: A Near-Data Processing Accelerator for Time Series Analysis"
*Proceedings of the 38th IEEE International Conference on Computer Design (**ICCD**)*, Virtual, October 2020.
[Slides (pptx) (pdf)]
[Talk Video (10 minutes)]
[Source Code]

NATSA: A Near-Data Processing Accelerator for Time Series Analysis

Ivan Fernandez[§] Ricardo Quislant[§] Christina Giannoula[†] Mohammed Alser[‡]
Juan Gómez-Luna[‡] Eladio Gutiérrez[§] Oscar Plata[§] Onur Mutlu[‡]
[§]*University of Malaga* [†]*National Technical University of Athens* [‡]*ETH Zürich*

FPGA-based Processing Near Memory

- Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu,
"FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications"
IEEE Micro (***IEEE MICRO***), 2021.

FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh[◇] Mohammed Alser[◇] Damla Senol Cali[⋈]
Dionysios Diamantopoulos[▽] Juan Gómez-Luna[◇]
Henk Corporaal[⋆] Onur Mutlu[◇⋈]

[◇]*ETH Zürich* [⋈]*Carnegie Mellon University*
[⋆]*Eindhoven University of Technology* [▽]*IBM Research Europe*

Accelerating Linked Data Structures

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

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

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

Accelerating Approximate String Matching

- Damla Senol Cali, Gurpreet S. Kalsi, Zulal Bingol, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu,
  "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"
  *Proceedings of the 53rd International Symposium on Microarchitecture (**MICRO**)*, Virtual, October 2020.
  [Lighting Talk Video (1.5 minutes)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (18 minutes)]
  [Slides (pptx) (pdf)]

GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali[†⋈] Gurpreet S. Kalsi[⋈] Zülal Bingöl[▽] Can Firtina[◇] Lavanya Subramanian[‡] Jeremie S. Kim[◇†]
Rachata Ausavarungnirun[⊙] Mohammed Alser[◇] Juan Gomez-Luna[◇] Amirali Boroumand[†] Anant Nori[⋈]
Allison Scibisz[†] Sreenivas Subramoney[⋈] Can Alkan[▽] Saugata Ghose[⋆†] Onur Mutlu[◇†▽]

[†]*Carnegie Mellon University* [⋈]*Processor Architecture Research Lab, Intel Labs* [▽]*Bilkent University* [◇]*ETH Zürich*
[‡]*Facebook* [⊙]*King Mongkut's University of Technology North Bangkok* [⋆]*University of Illinois at Urbana–Champaign*

Accelerating Genome Analysis [IEEE MICRO 2020]

- Mohammed Alser, Zulal Bingol, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  IEEE Micro (***IEEE MICRO***), Vol. 40, No. 5, pages 65-75, September/October 2020.
  [Slides (pptx)(pdf)]
  [Talk Video (1 hour 2 minutes)]

Accelerating Genome Analysis: A Primer on an Ongoing Journey

Mohammed Alser
ETH Zürich

Zülal Bingöl
Bilkent University

Damla Senol Cali
Carnegie Mellon University

Jeremie Kim
ETH Zurich and Carnegie Mellon University

Saugata Ghose
University of Illinois at Urbana–Champaign and Carnegie Mellon University

Can Alkan
Bilkent University

Onur Mutlu
ETH Zurich, Carnegie Mellon University, and Bilkent University

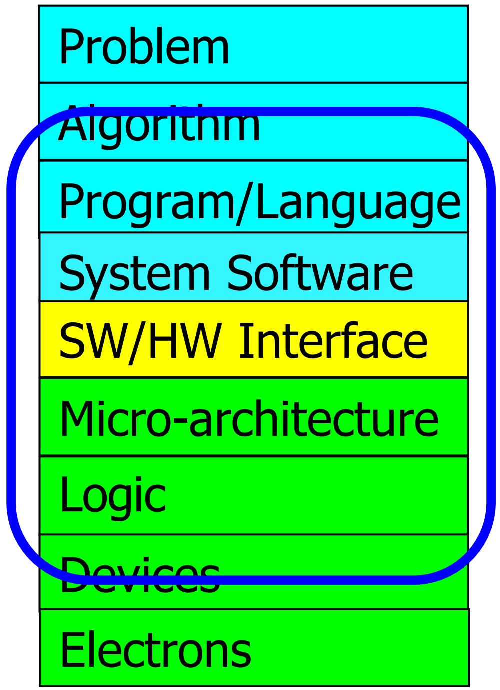
Challenge and Opportunity for Future

Data-Aware

(Expressive)

Computing Architectures

We Need to **Rethink** the Entire Stack



We can get there case by case

Simulating Memory Systems

Ramulator: A Fast and Extensible DRAM Simulator

[IEEE Comp Arch Letters'15]

Ramulator Motivation

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

Segment	*DRAM Standards & Architectures*
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLDRAM3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

Table 1. Landscape of DRAM-based memory

Ramulator

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

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

Table 3. Comparison of five simulators using two traces

Case Study: Comparison of DRAM Standards

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

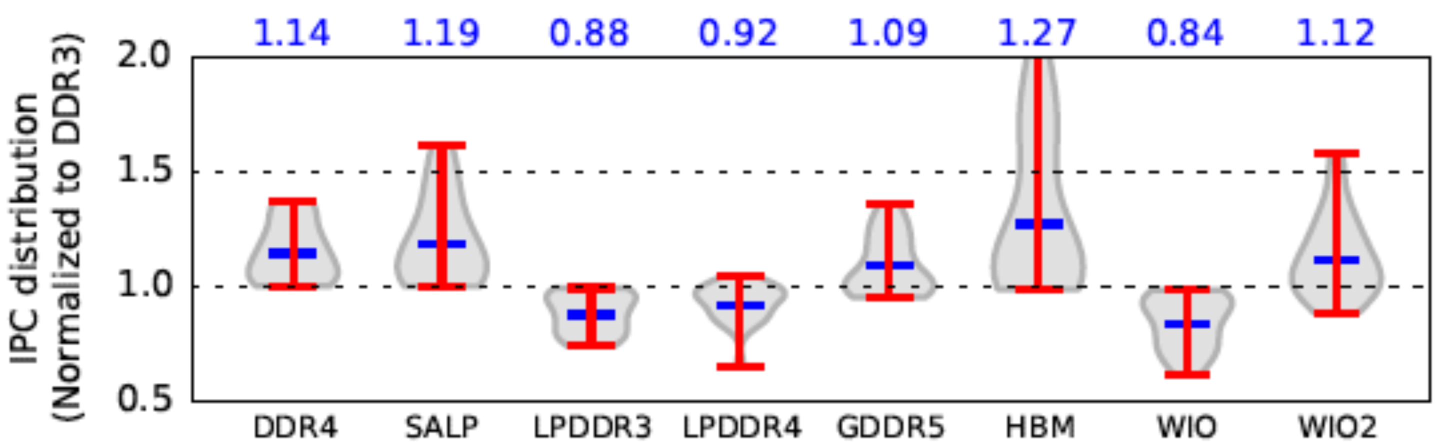


Figure 2. Performance comparison of DRAM standards

Across 22 workloads, simple CPU model

Ramulator 2.0

- Haocong Luo, Yahya Can Tugrul, F. Nisa Bostanci, Ataberk Olgun, A. Giray Yaglikci, and Onur Mutlu,
  "Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator"
  Preprint on ***arxiv***, August 2023.
  [arXiv version]
  [Ramulator 2.0 Source Code]

Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator

Haocong Luo, Yahya Can Tuğrul, F. Nisa Bostancı, Ataberk Olgun, A. Giray Yağlıkçı, and Onur Mutlu

https://arxiv.org/pdf/2308.11030.pdf

https://github.com/CMU-SAFARI/ramulator2

Optional Assignment: Ramulator 2.0

- Review the Ramulator 2.0 paper
  - Email me your review (omutlu@gmail.com)

- Download and run Ramulator 2.0
  - Compare DDR4, LPDDR5, HBM2 for benchmarks of your choice (provided in Ramulator repository)
  - Email me your report (omutlu@gmail.com)

- This can help you get into memory systems research quickly

Memory-Centric Computing

Agenda For Today

- Memory Systems and Memory-Centric Computing
  - July 15-19, 2024

- Topic 1: Memory Trends, Challenges, Opportunities, Basics
- Topic 2: Memory-Centric Computing
- Topic 3: Memory Robustness: RowHammer, RowPress & Beyond
- Topic 4: Machine Learning Driven Memory Systems
- Topic 5 (another course): Architectures for Genomics and ML
- Topic 6 (unlikely): Non-Volatile Memories and Storage
- Topic 7 (unlikely): Memory Latency, Predictability & QoS

- Major Overview Reading:
  - Mutlu et al., "A Modern Primer on Processing in Memory," Book Chapter on Emerging Computing and Devices, 2022.

The Problem

Computing

is Bottlenecked by Data

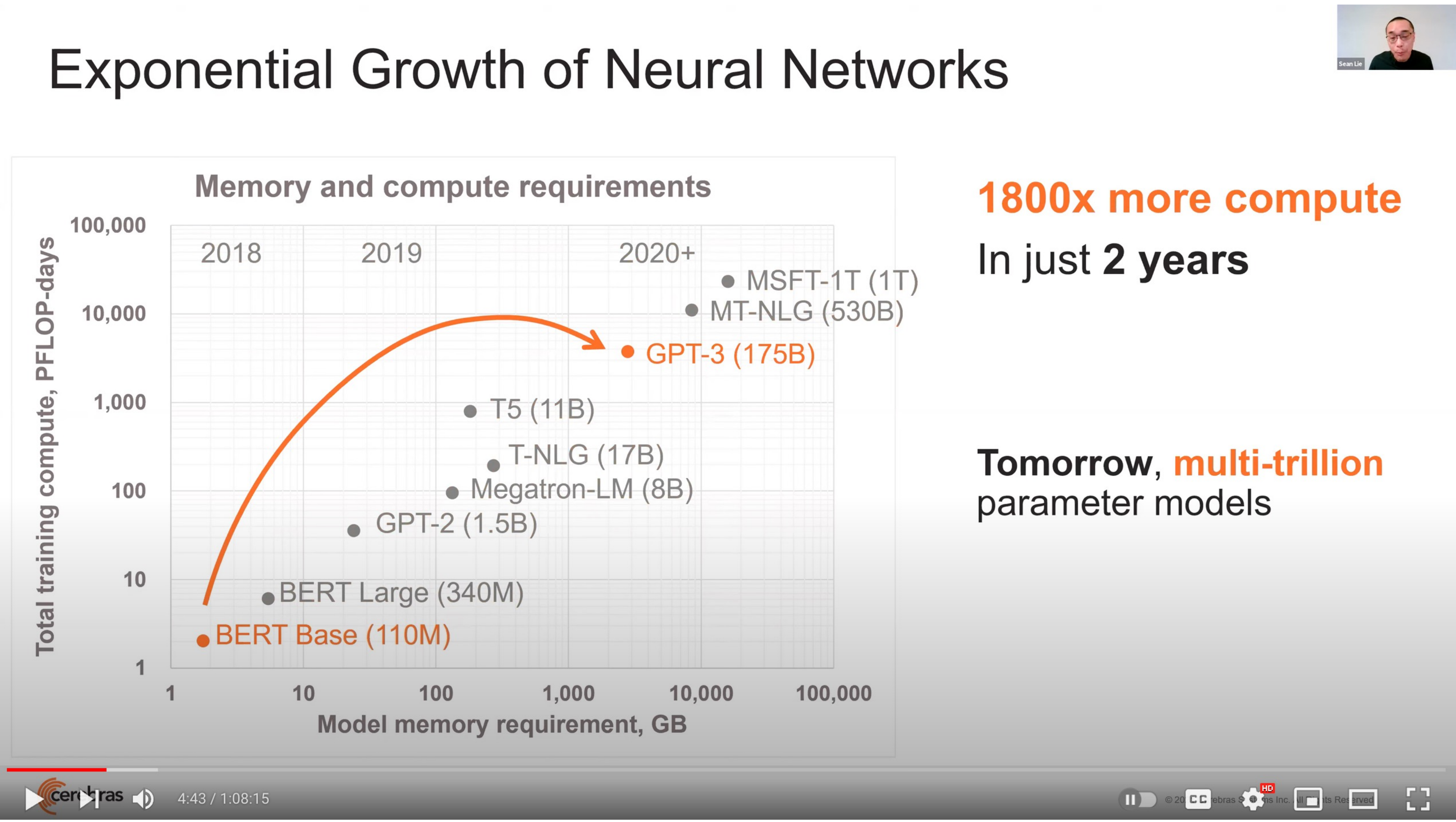
Data is Key for AI, ML, Genomics, …

- Important workloads are all data intensive

- They require rapid and efficient processing of large amounts of data

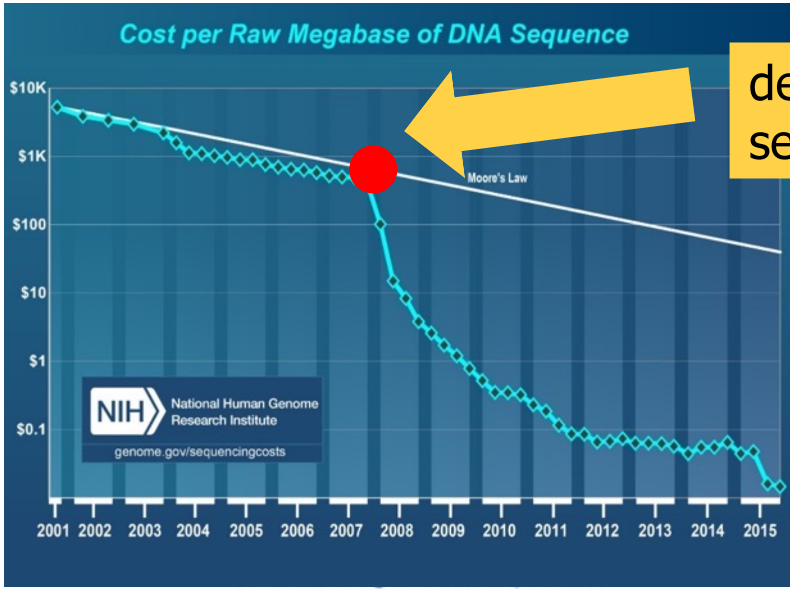
- Data is increasing
  - We can generate more than we can process
  - We need to perform more sophisticated analyses on more data

Huge Demand for Performance & Efficiency



Source: https://youtu.be/Bh13Idwcb0Q?t=283

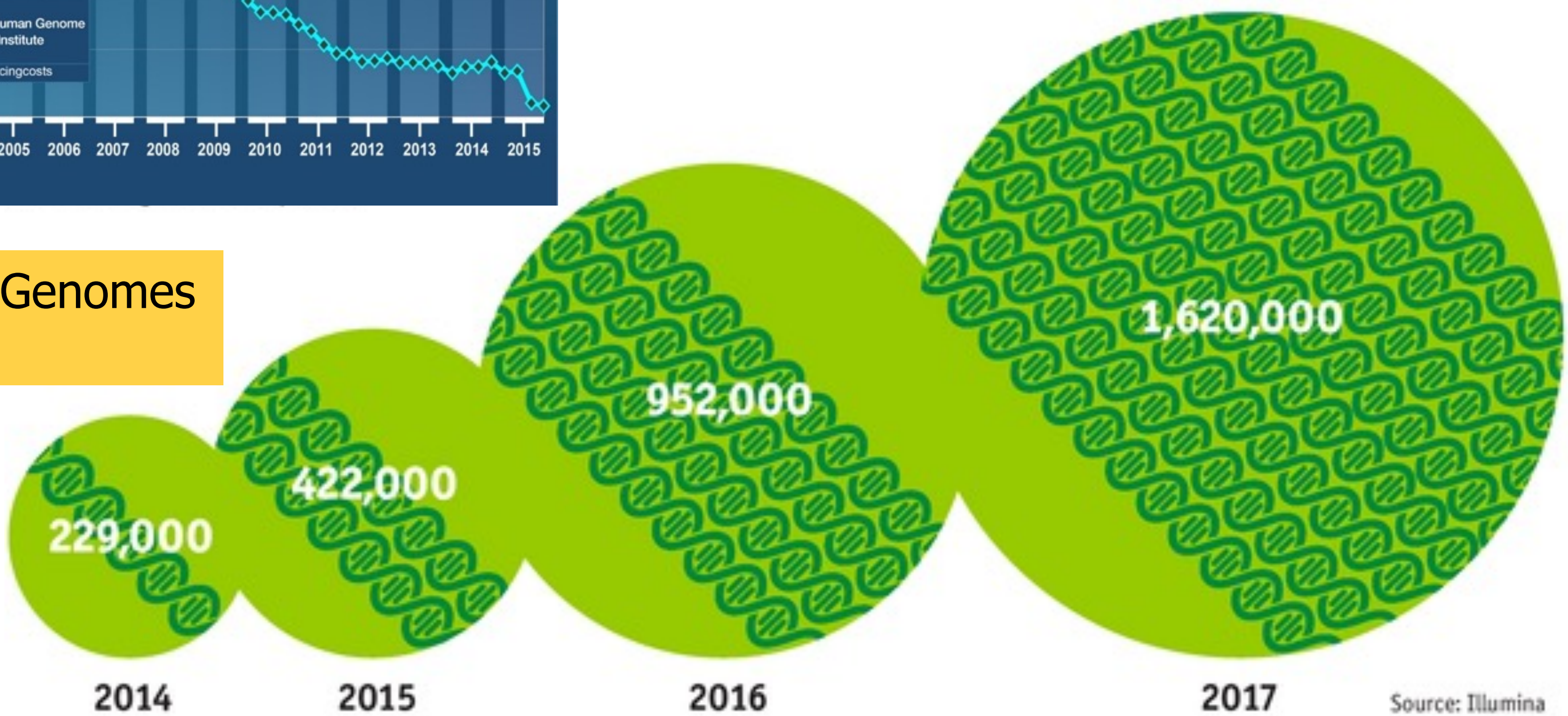
Huge Demand for Performance & Efficiency



development of new sequencing technologies

Oxford Nanopore MinION

Number of Genomes Sequenced



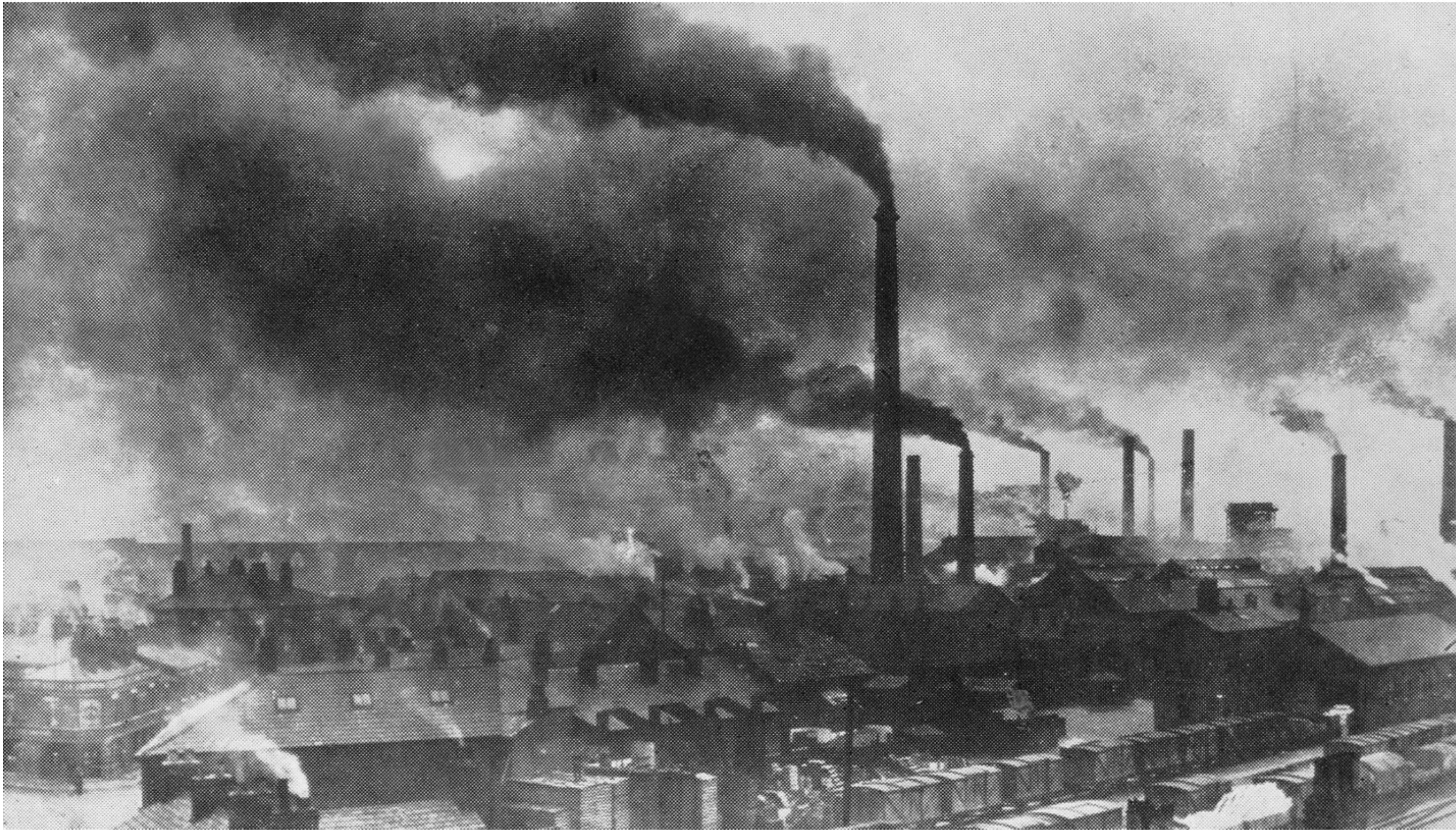
The Economist

http://www.economist.com/news/21631808-so-much-genetic-data-so-many-uses-genes-unzipped

Do We Want This?

Source: V. Milutinovic

Or This?



Source: V. Milutinovic

Challenge and Opportunity for Future

High Performance,

Energy Efficient,

Sustainable

(All at the Same Time)

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

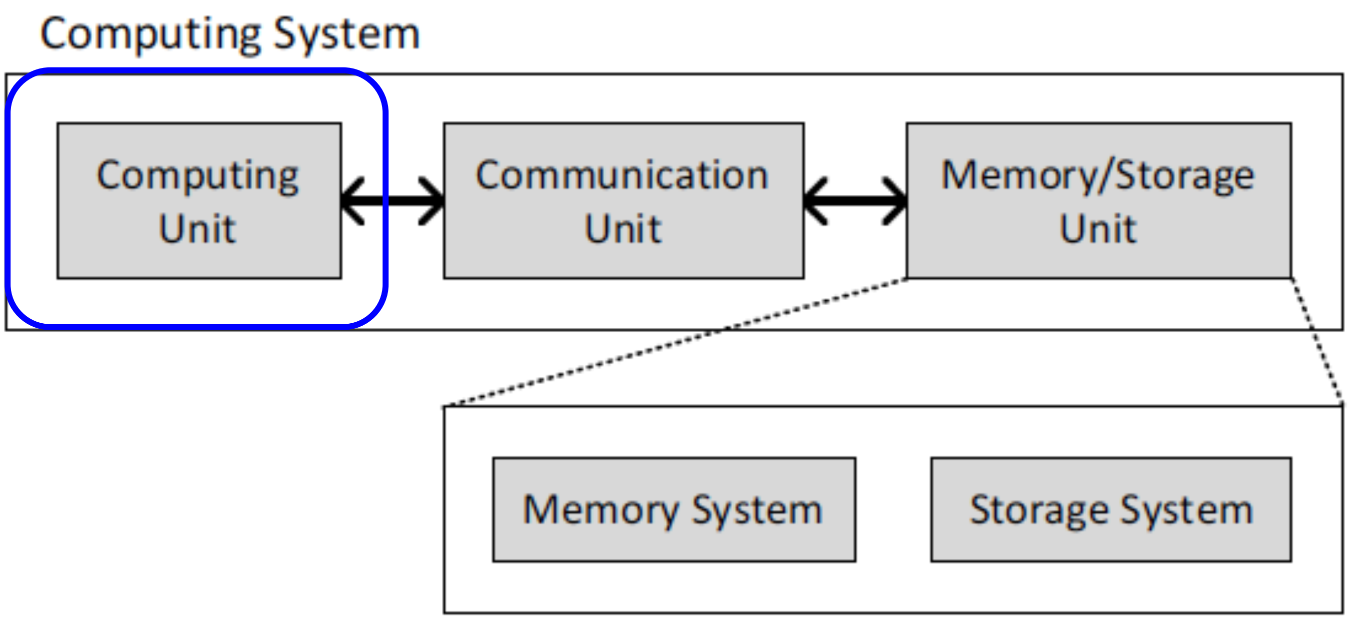
Processing of data

is performed

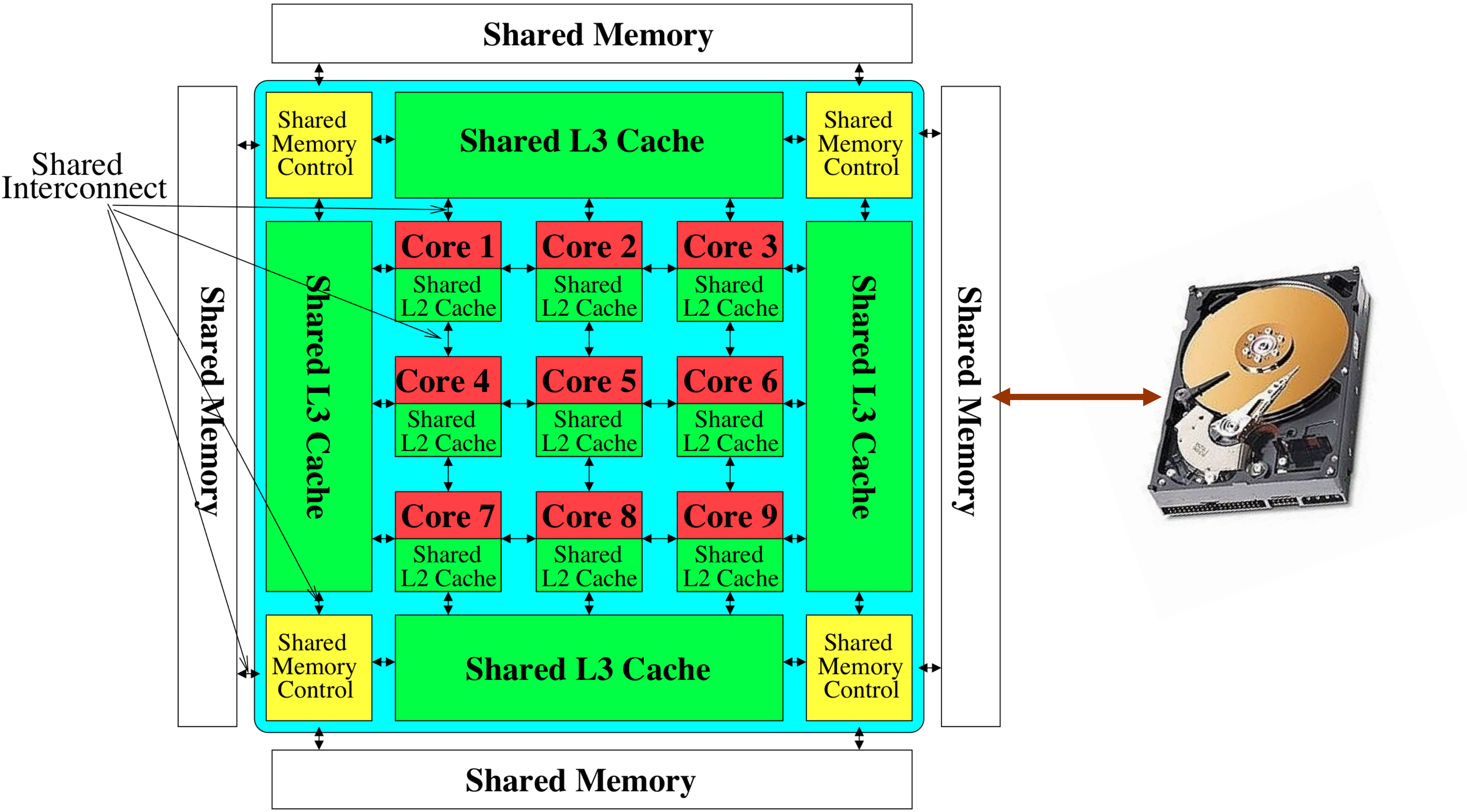
far away from the data

Today's Computing Systems

- Processor centric
- All data processed in the processor → at great system cost



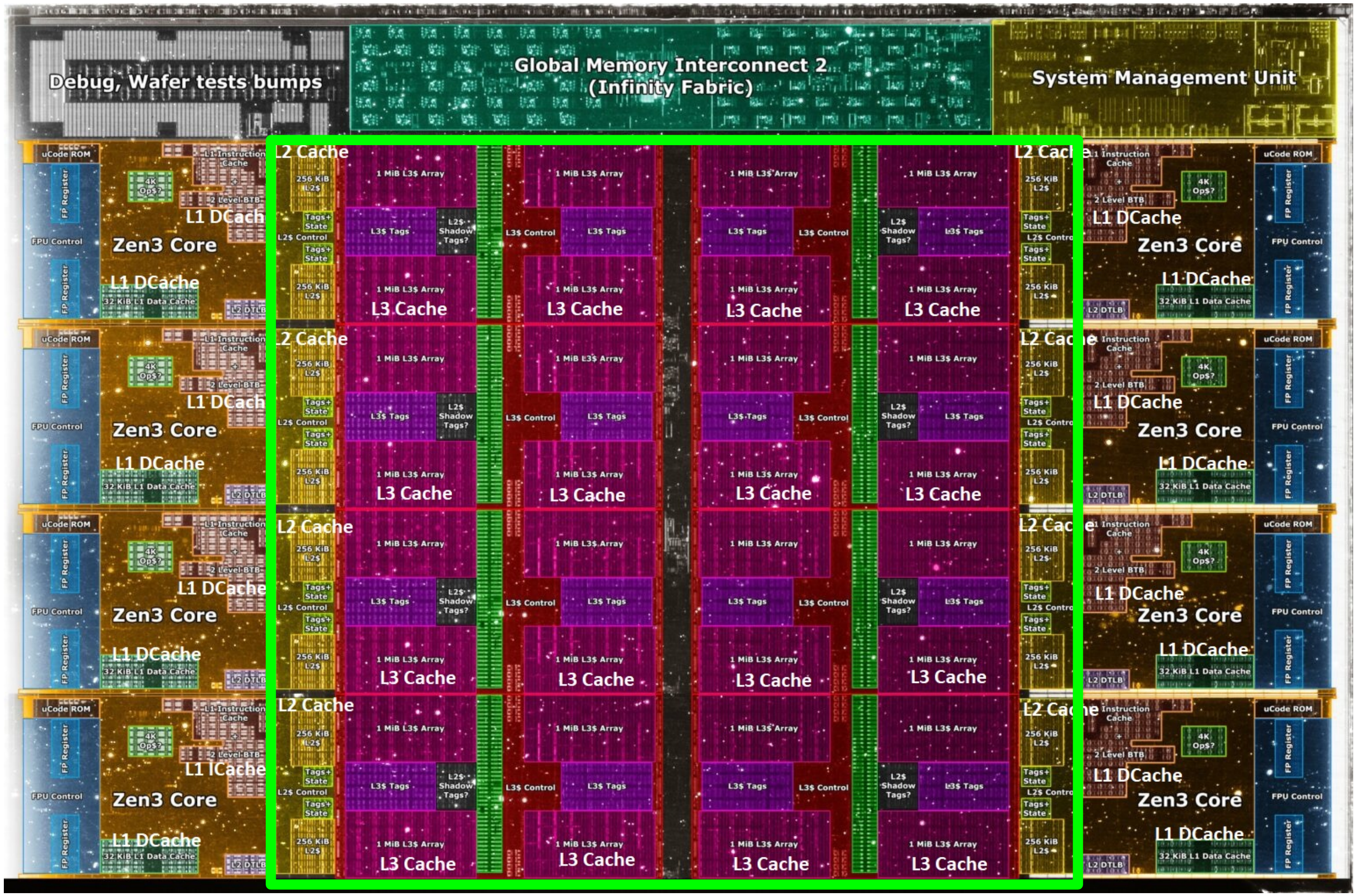
Perils of Processor-Centric Design



Most of the system is dedicated to storing and moving data

Yet, system is still bottlenecked by memory

Deeper and Larger Memory Hierarchies



AMD Ryzen 5000, 2020

Core Count:
8 cores/16 threads

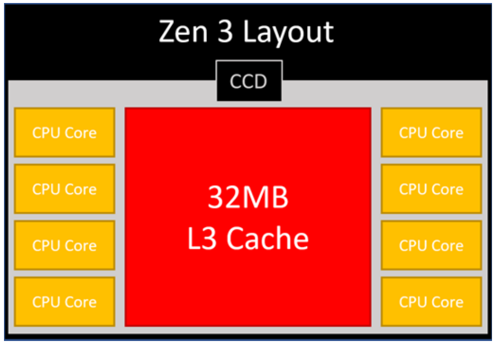
L1 Caches:
32 KB per core

L2 Caches:
512 KB per core

L3 Cache:
32 MB shared

https://wccftech.com/amd-ryzen-5000-zen-3-vermeer-undressed-high-res-die-shots-close-ups-pictured-detailed/

AMD's 3D Last Level Cache (2021)

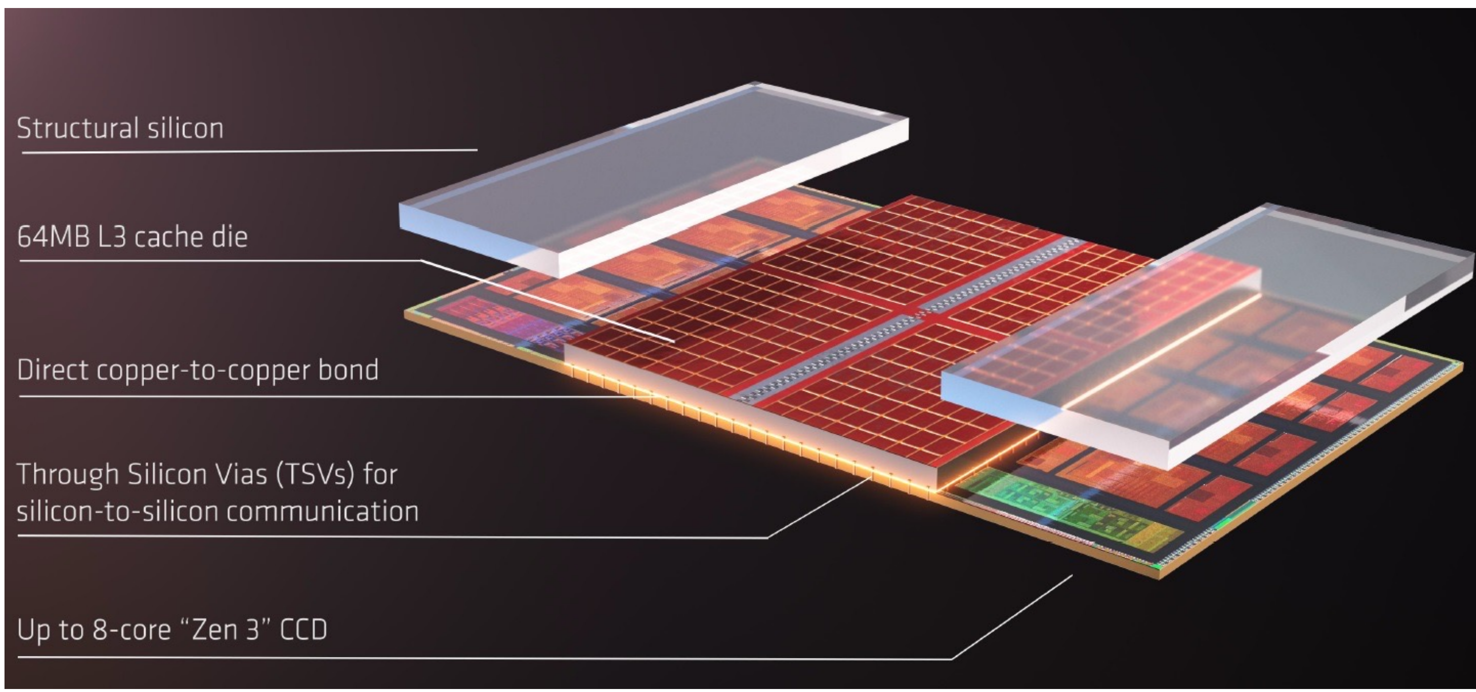


https://community.microcenter.com/discussion/5134/comparing-zen-3-to-zen-2

AMD increases the L3 size of their 8-core Zen 3 processors from 32 MB to 96 MB

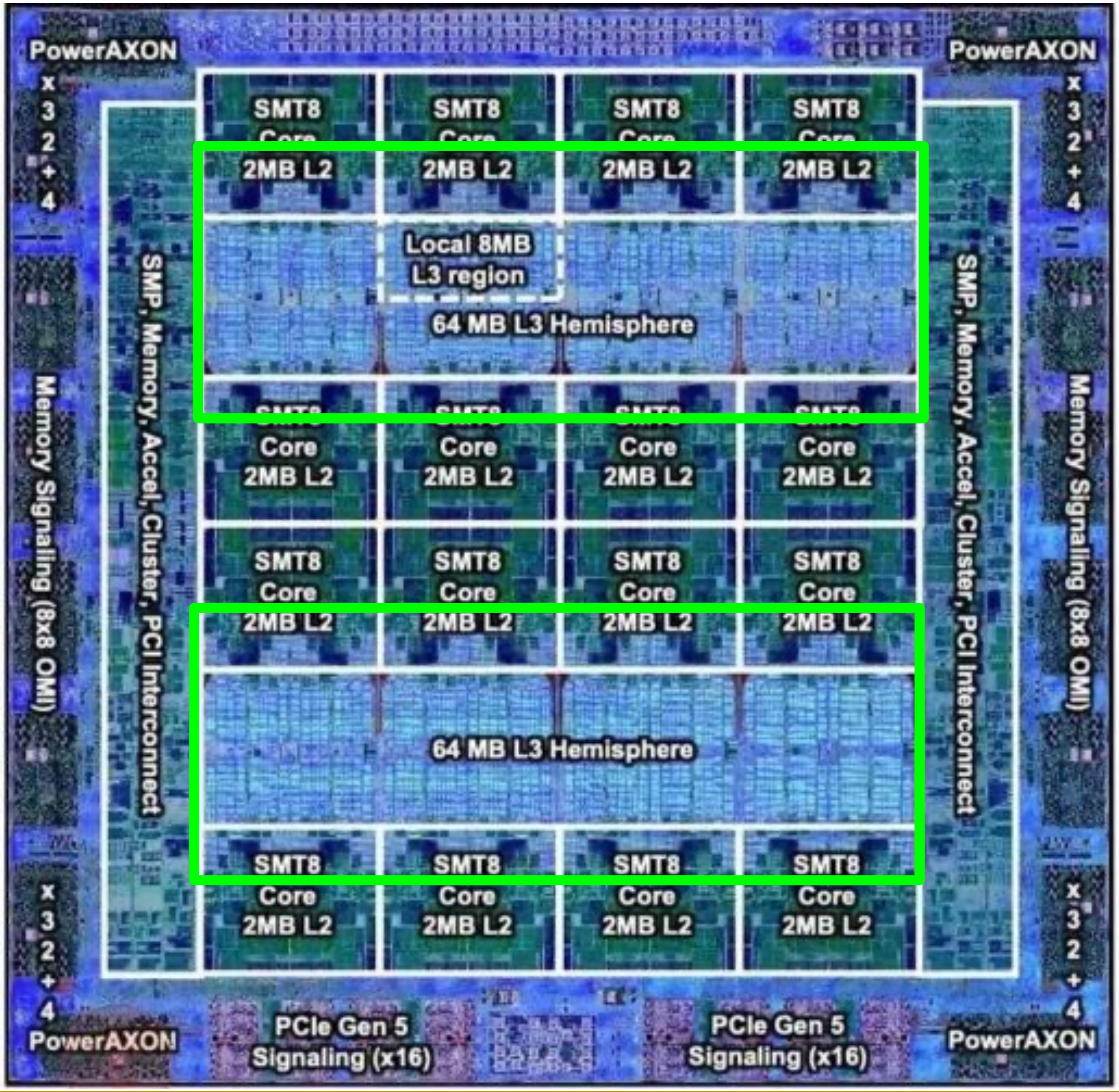
Additional 64 MB L3 cache die stacked on top of the processor die

- Connected using Through Silicon Vias (TSVs)
- Total of 96 MB L3 cache



https://youtu.be/gqAYMx34euU
https://www.tech-critter.com/amd-keynote-computex-2021/

Deeper and Larger Memory Hierarchies



IBM POWER10, 2020

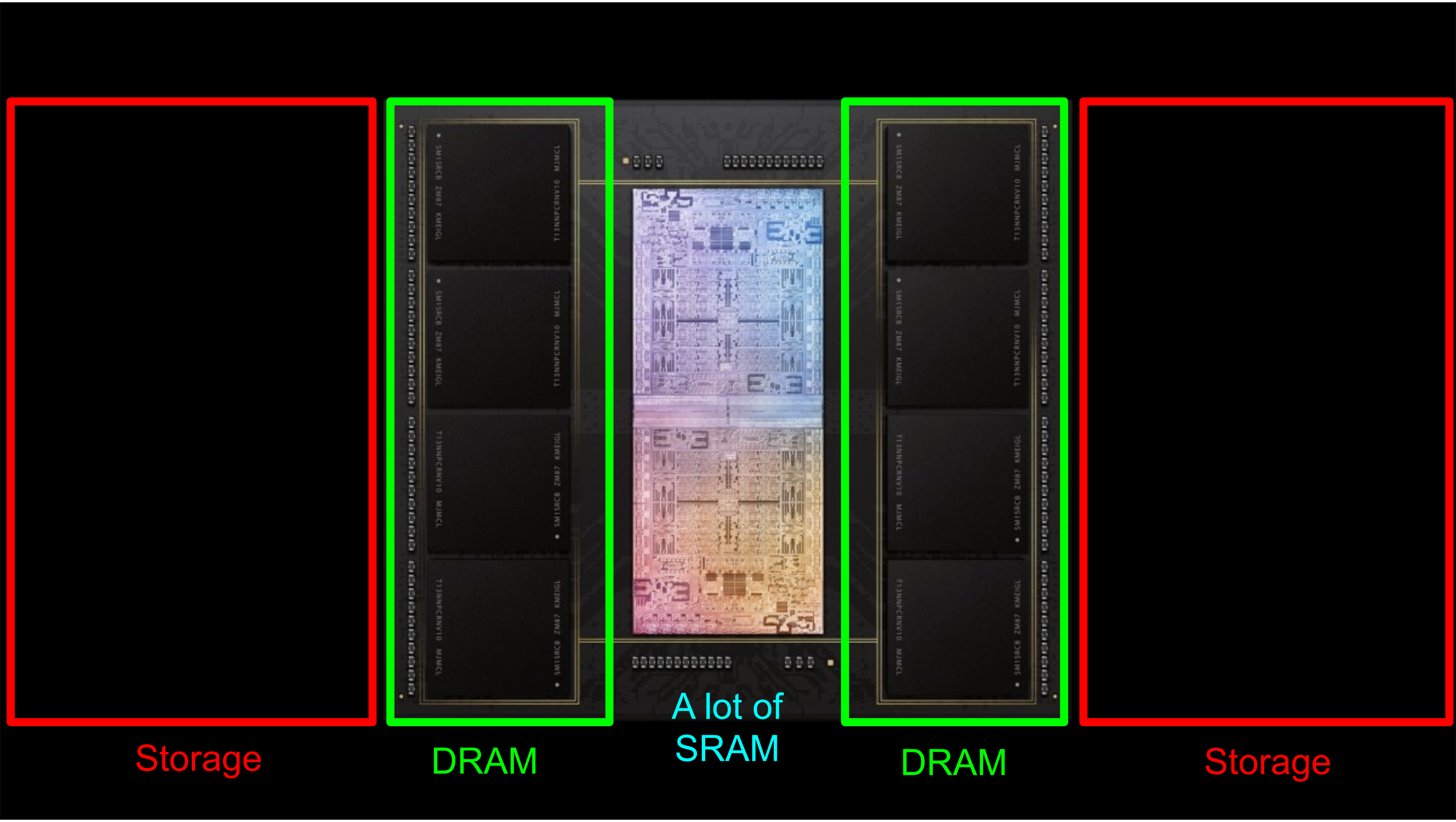
Cores:
15-16 cores,
8 threads/core

L2 Caches:
2 MB per core

L3 Cache:
120 MB shared

https://www.it-techblog.de/ibm-power10-prozessor-mehr-speicher-mehr-tempo-mehr-sicherheit/09/2020/

Deeper and Larger Memory Hierarchies



Apple M1 Ultra System (2022)

https://www.gsmarena.com/apple_announces_m1_ultra_with_20core_cpu_and_64core_gpu-news-53481.php

Data Overwhelms Modern Machines …

- Storage/memory capability

- Communication capability

- Computation capability

- Greatly impacts robustness, energy, performance, cost

It's the Memory, Stupid!

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

RICHARD SITES

It's the Memory, Stupid!

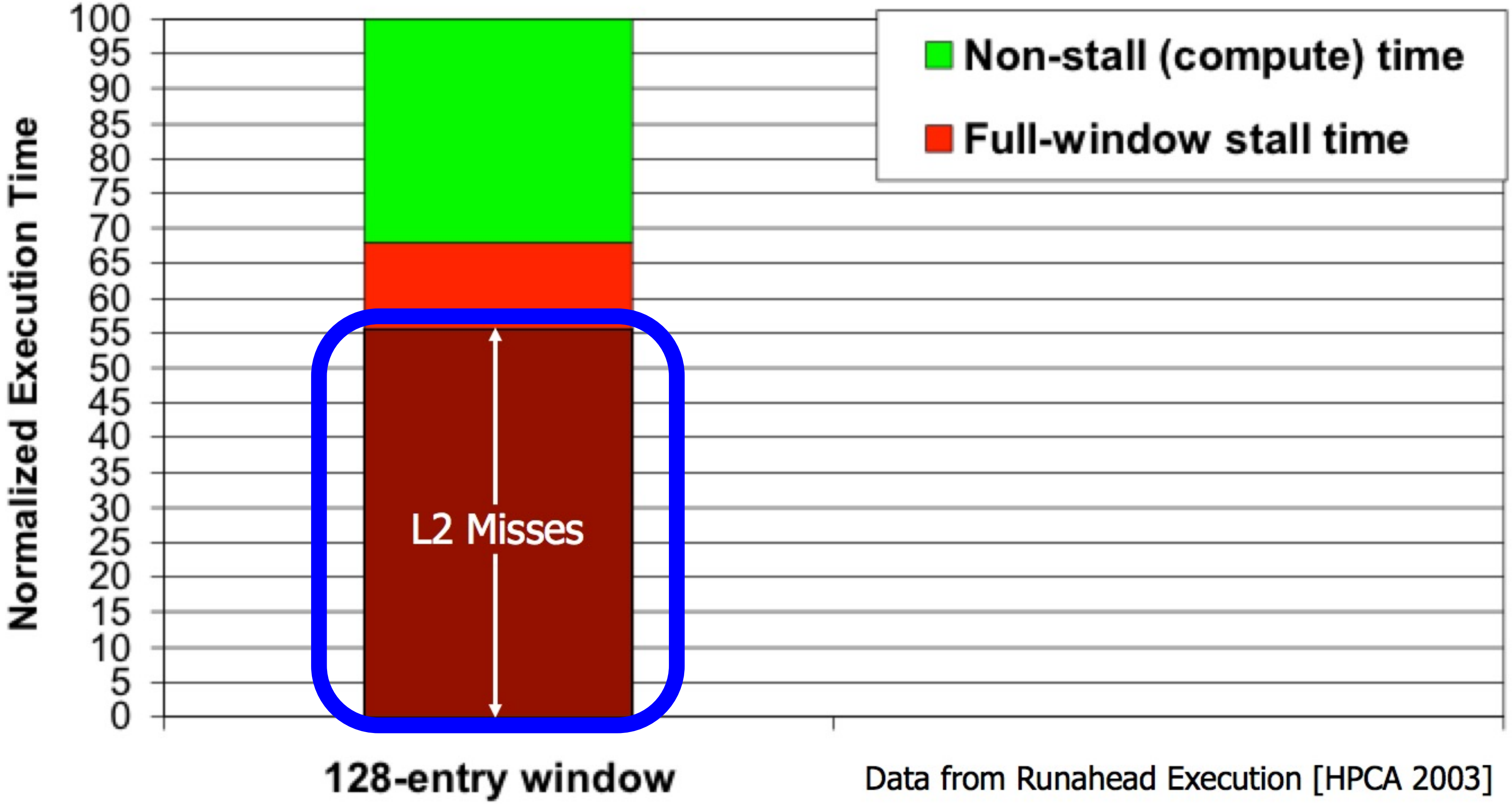
When we started the Alpha architecture design in 1988, we estimated a 25-year lifetime and a relatively modest 32% per year compounded performance improvement of implementations over that lifetime (1,000× total). We guestimated about 10× would come from CPU clock improvement, 10× from multiple instruction issue, and 10× from multiple processors.

5, 1996 MICROPROCESSOR REPORT

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

http://cva.stanford.edu/classes/cs99s/papers/architects_look_to_future.pdf

The Performance Perspective



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

- Onur Mutlu, Jared Stark, Chris Wilkerson, and Yale N. Patt, **"Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors"**
Proceedings of the 9th International Symposium on High-Performance Computer Architecture (***HPCA***), pages 129-140, Anaheim, CA, February 2003. Slides (pdf)
One of the 15 computer arch. papers of 2003 selected as Top Picks by IEEE Micro.
HPCA Test of Time Award (awarded in 2021).

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

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

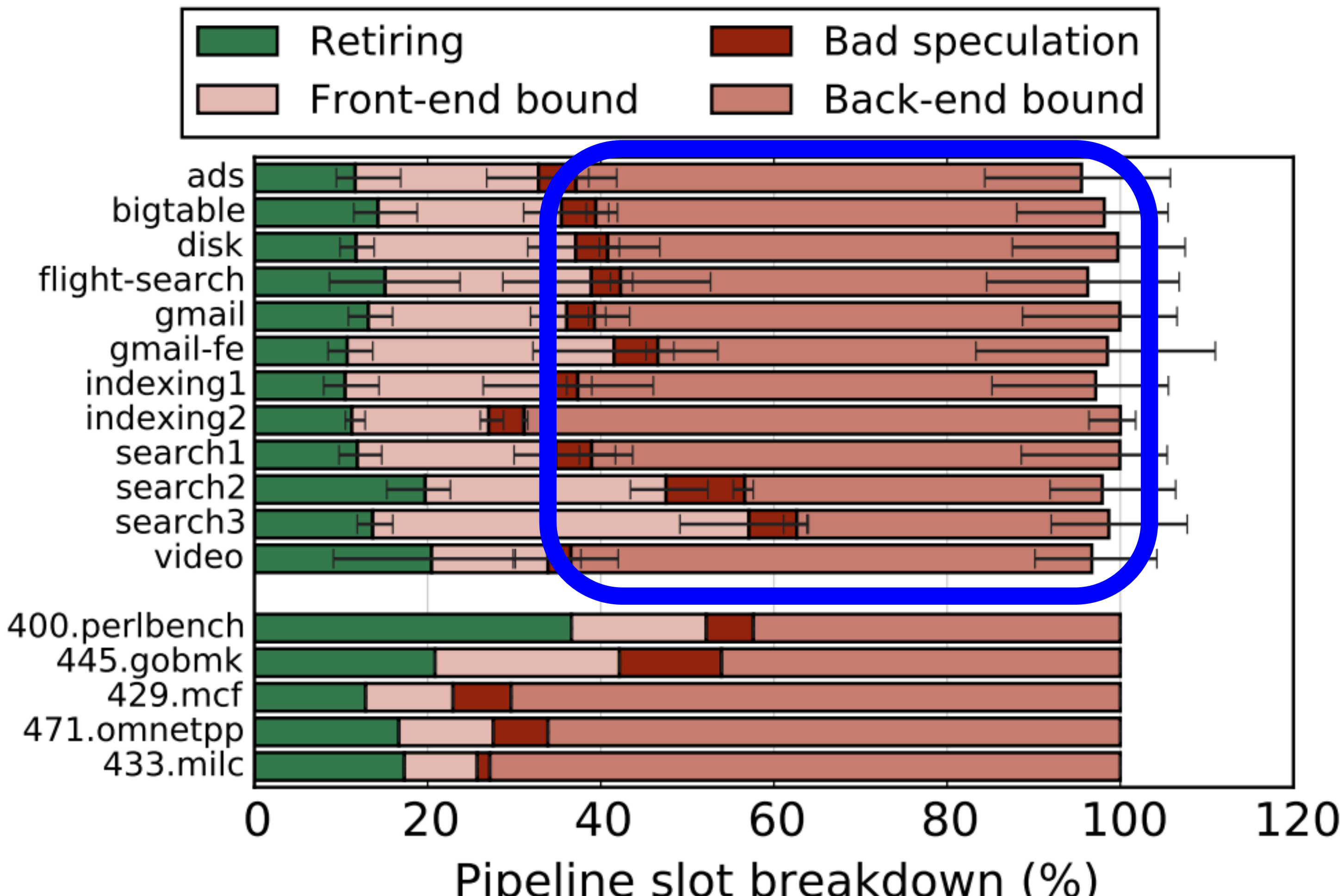
§ECE Department
The University of Texas at Austin
{onur,patt}@ece.utexas.edu

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

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

The Performance Perspective (Today)

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



Kanev+, "Profiling a Warehouse-Scale Computer," ISCA 2015.

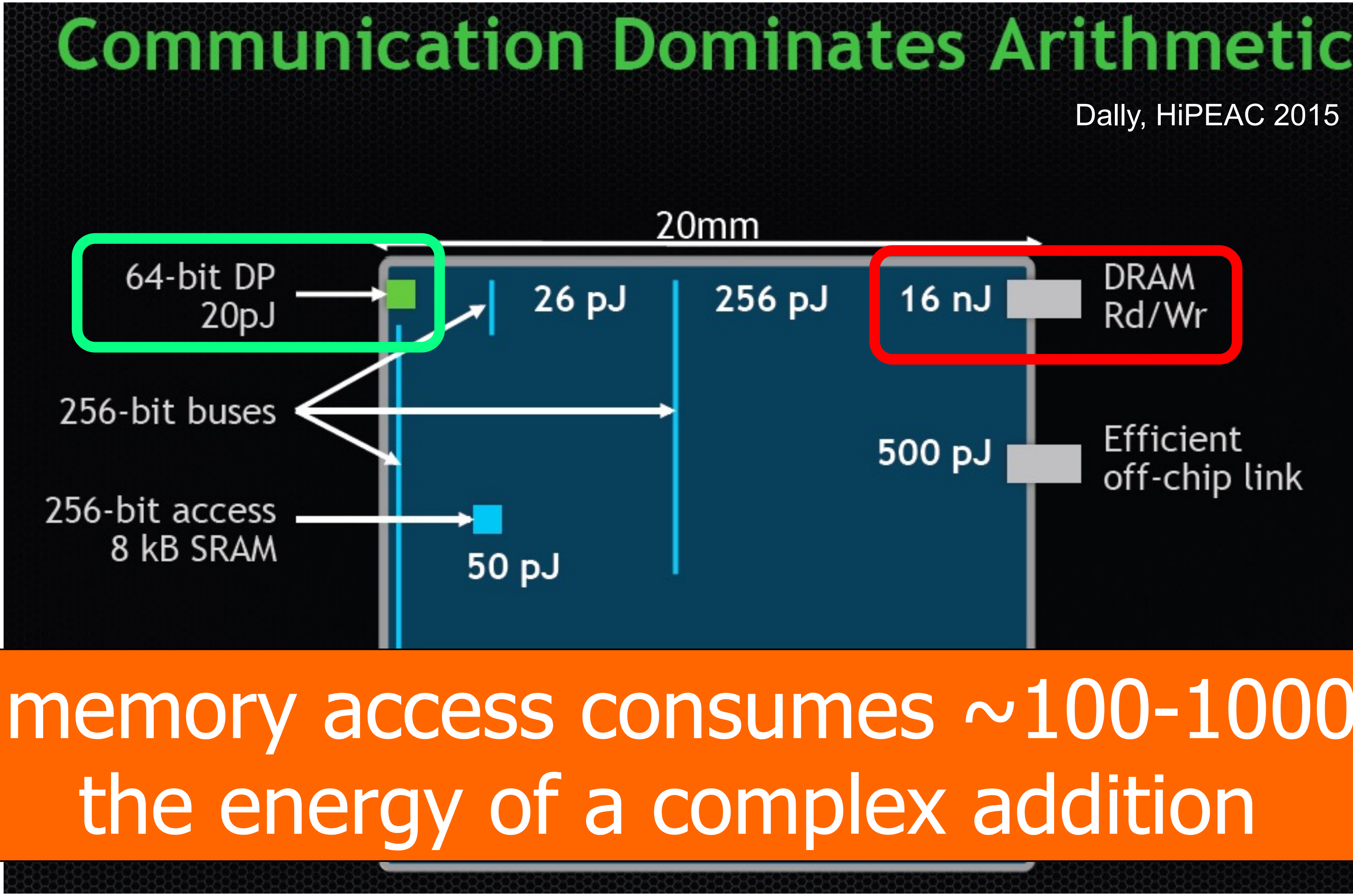
Three Key Systems Trends

1. Data access is a major bottleneck
   - Applications are increasingly data hungry

2. Energy consumption is a key limiter

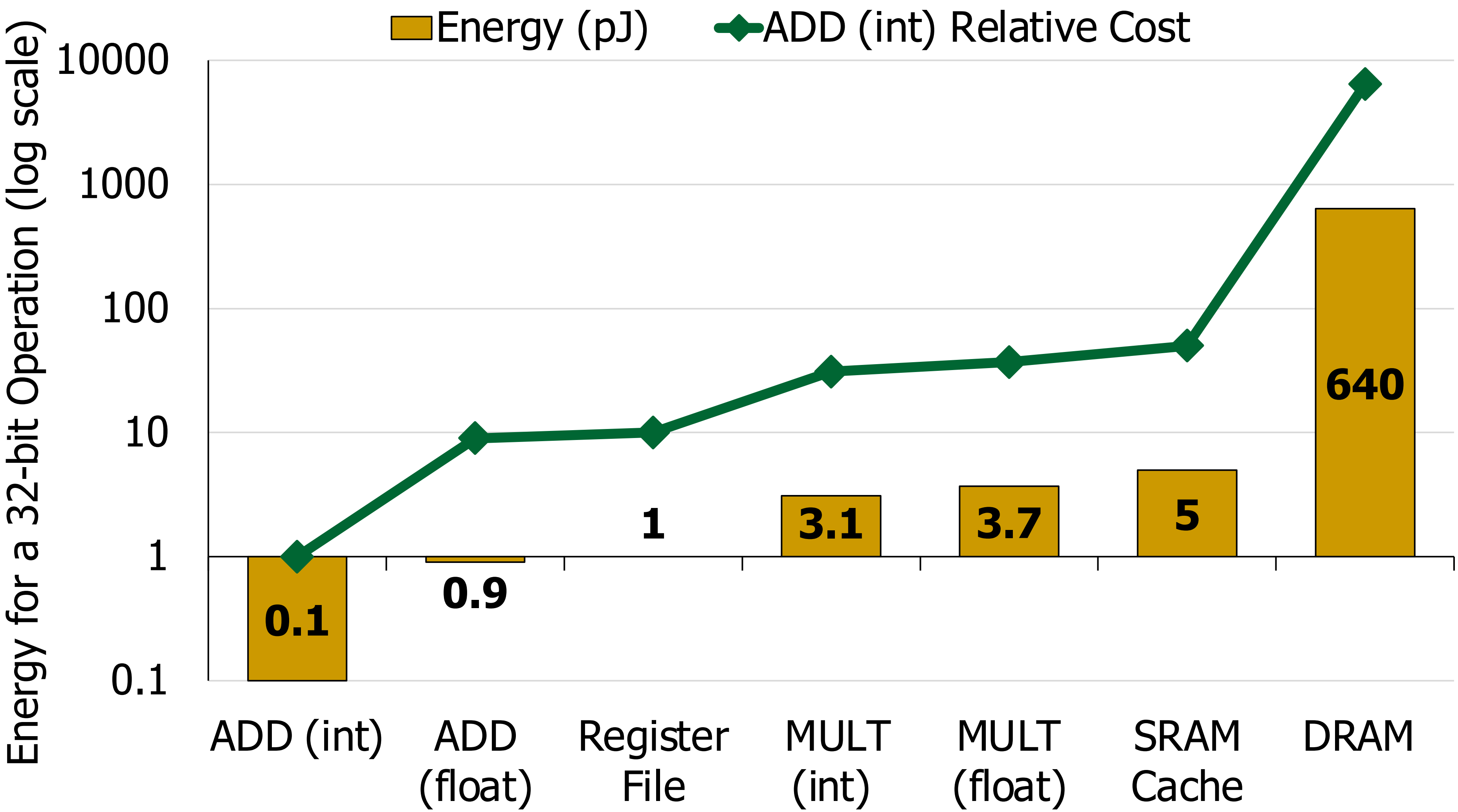
3. Data movement energy dominates compute
   - Especially true for off-chip to on-chip movement

Data Movement vs. Computation Energy



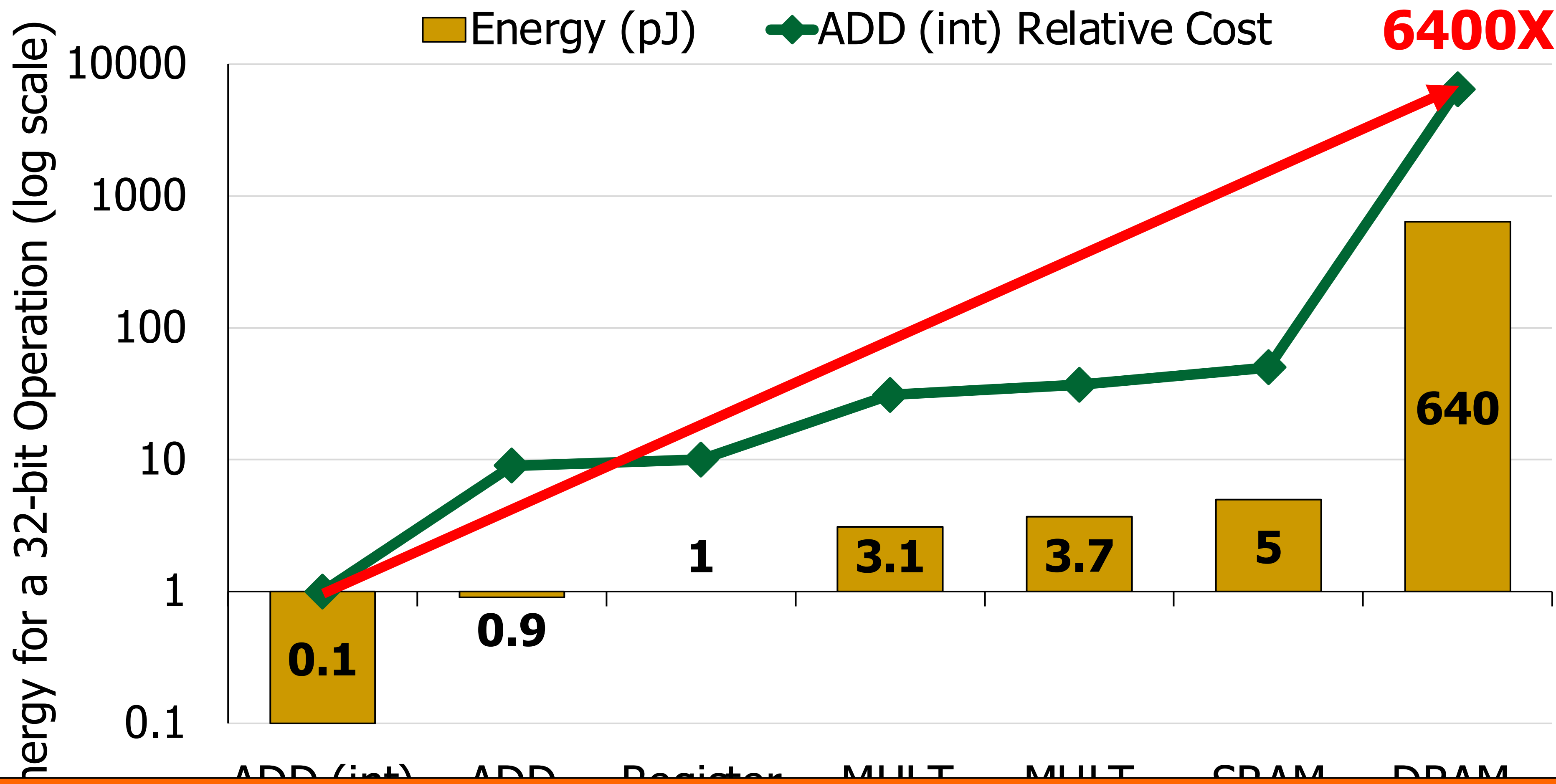
A memory access consumes ~100-1000X the energy of a complex addition

Data Movement vs. Computation Energy



Han+, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," ISCA 2016.

Data Movement vs. Computation Energy



A memory access consumes 6400X the energy of a simple integer addition

Energy Waste in Mobile Devices

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

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

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand[1] Saugata Ghose[1] Youngsok Kim[2]
Rachata Ausavarungnirun[1] Eric Shiu[3] Rahul Thakur[3] Daehyun Kim[4,3]
Aki Kuusela[3] Allan Knies[3] Parthasarathy Ranganathan[3] Onur Mutlu[5,1]

Energy Waste in Accelerators

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,
  "Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks"
  Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (***PACT***), Virtual, September 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (14 minutes)]

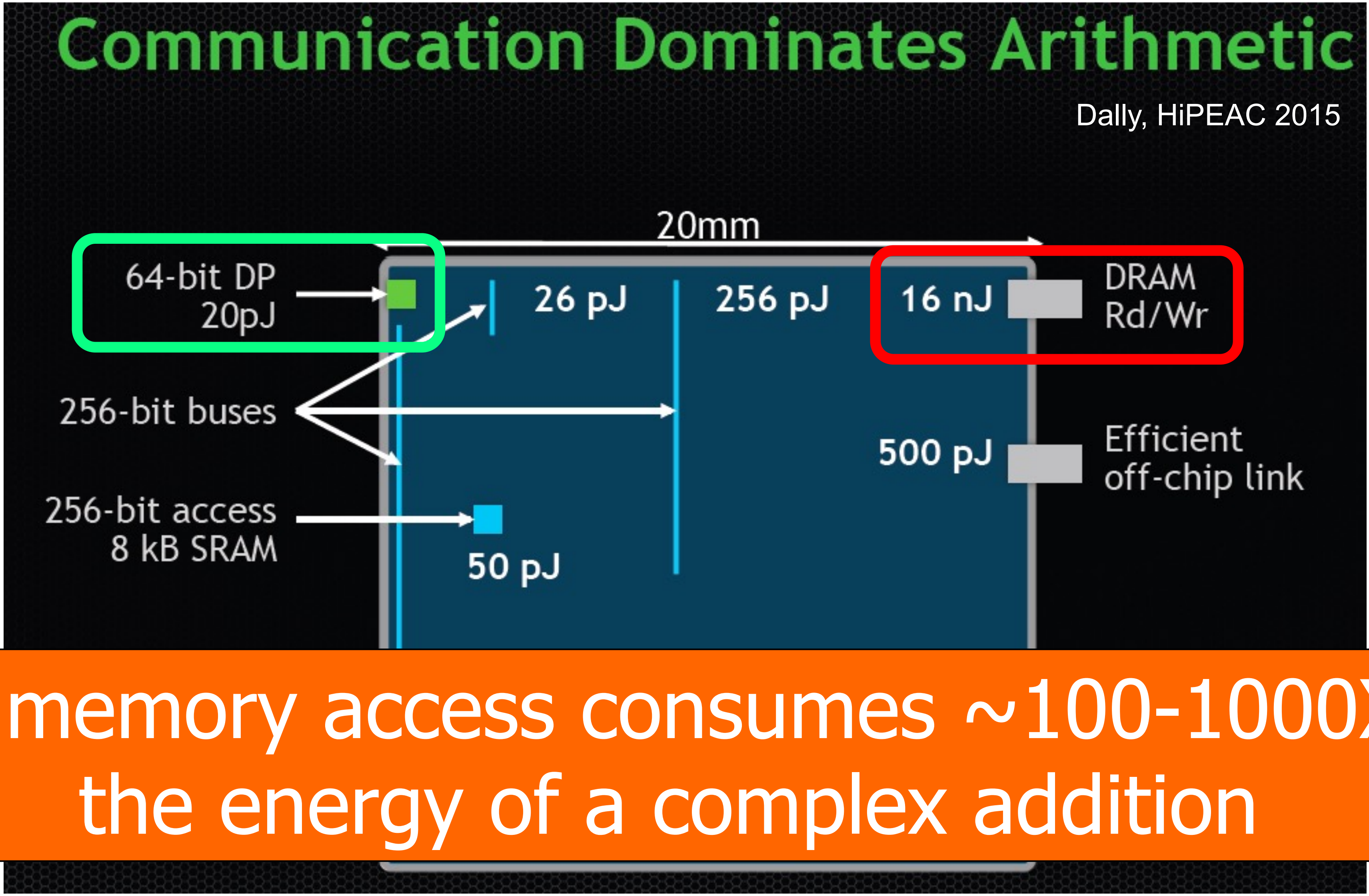
> 90% of the total system energy is spent on memory in large ML models

Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand[†◇] Saugata Ghose[‡] Berkin Akin[§] Ravi Narayanaswami[§]
Geraldo F. Oliveira[⋆] Xiaoyu Ma[§] Eric Shiu[§] Onur Mutlu[⋆†]

[†]*Carnegie Mellon Univ.* [◇]*Stanford Univ.* [‡]*Univ. of Illinois Urbana-Champaign* [§]*Google* [⋆]*ETH Zürich*

We Do Not Want to Move Data!



A memory access consumes ~100-1000X the energy of a complex addition

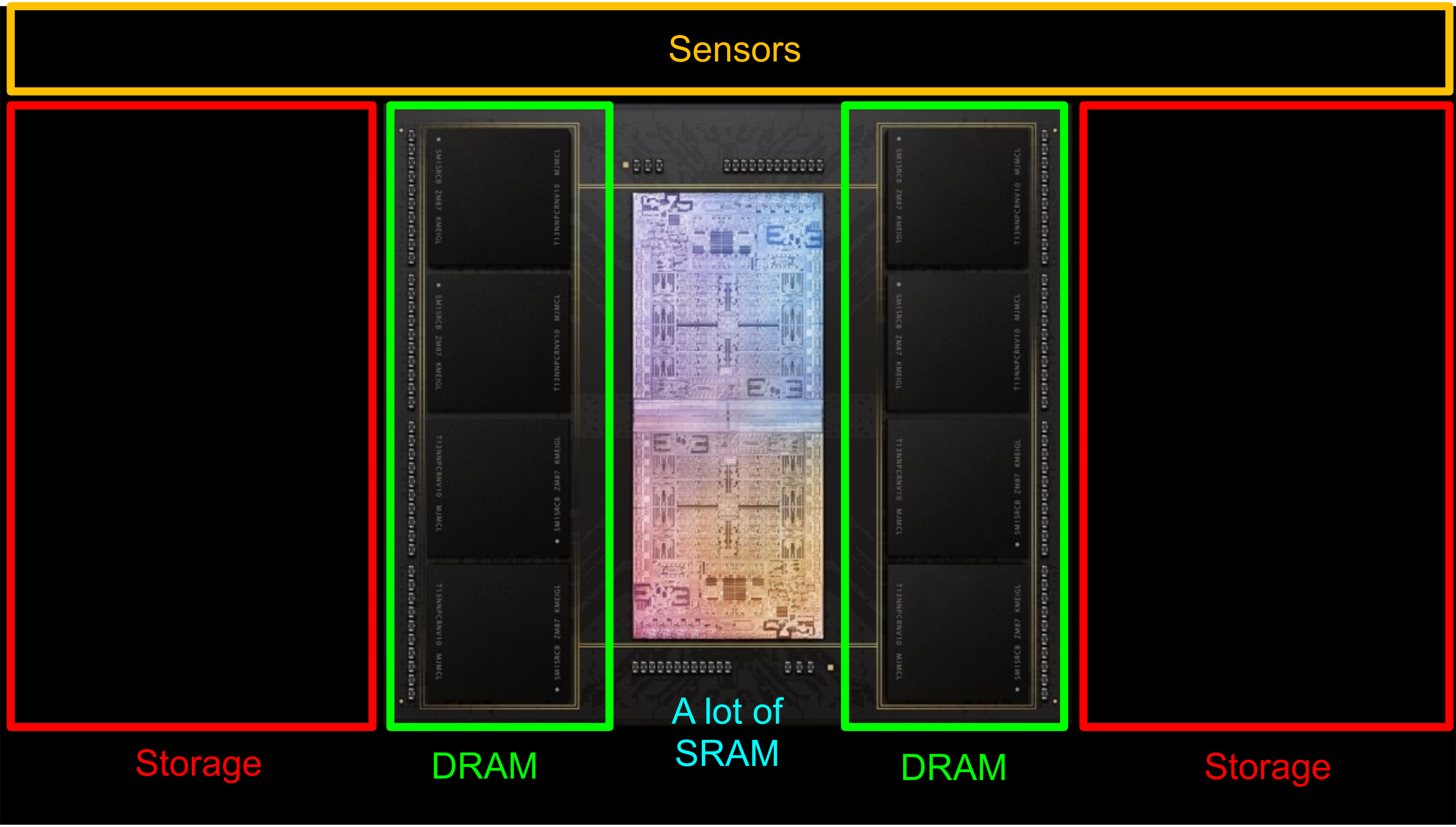
We Need A **Paradigm Shift** To …

- Enable computation with minimal data movement

- Compute where it makes sense (where data resides)

- Make computing architectures more data-centric

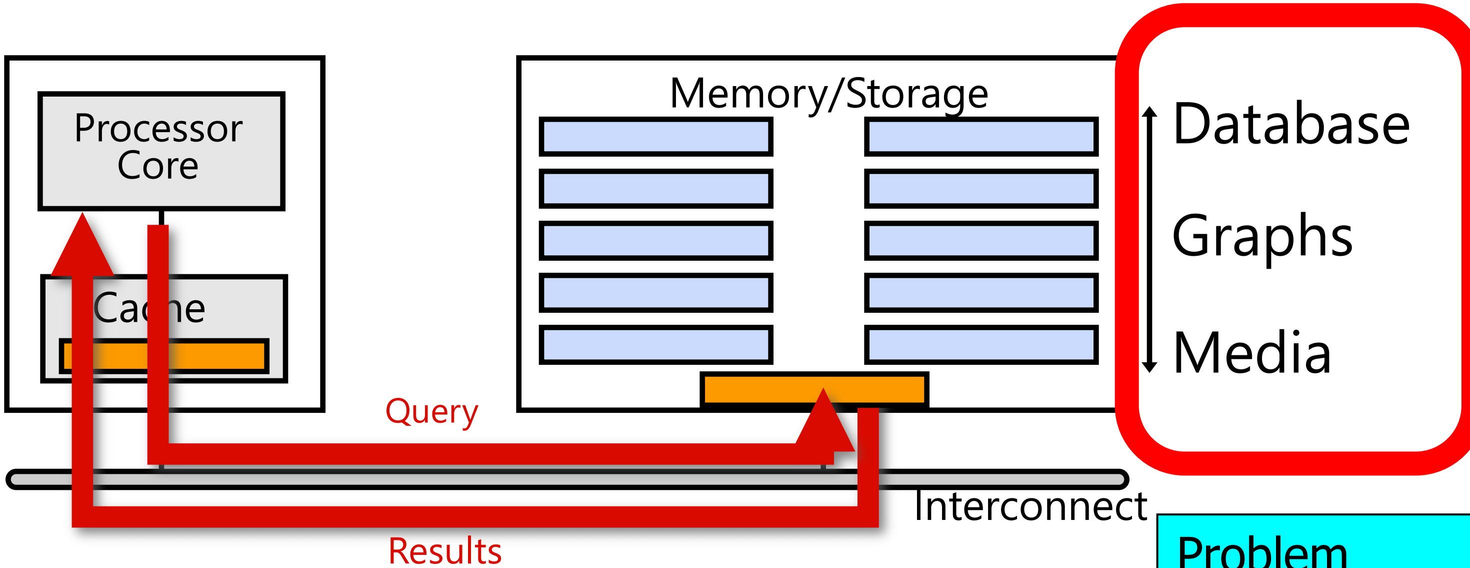
Process Data Where It Makes Sense

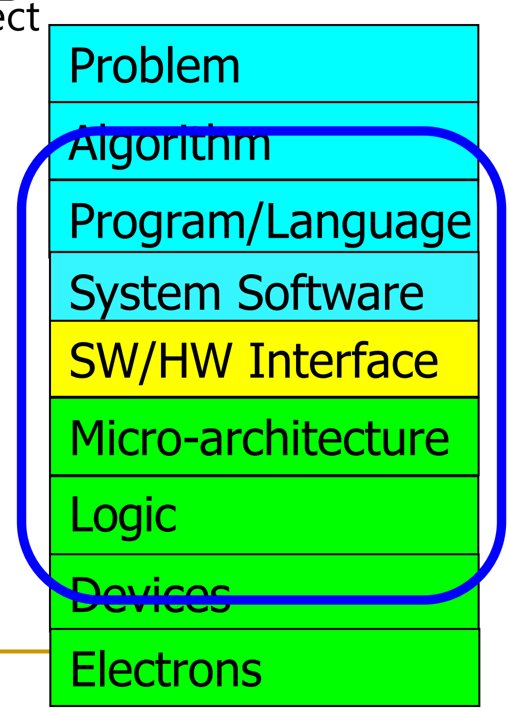


Apple M1 Ultra System (2022)

https://www.gsmarena.com/apple_announces_m1_ultra_with_20core_cpu_and_64core_gpu-news-53481.php

Goal: Processing Inside Memory/Storage





- Many questions … How do we design the:
  - compute-capable memory & controllers?
  - processors & communication units?
  - software & hardware interfaces?
  - system software, compilers, languages?
  - algorithms & theoretical foundations?

Processing in/near Memory: An Old Idea

- 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.

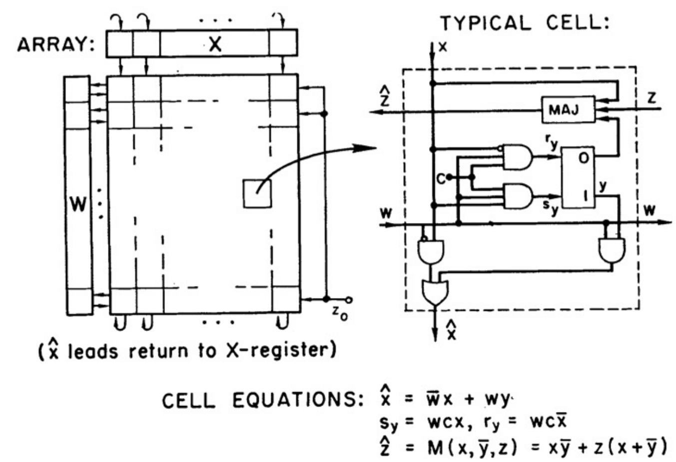


Fig. 1. Cellular sorting array I.

https://doi.org/10.1109/T-C.1969.222754

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.

Why In-Memory Computation Today?

- **Huge demand from Applications & Systems**
  - Data access bottleneck
  - Energy & power bottlenecks
  - Data movement energy dominates computation energy
  - Need all at the same time: performance, energy, sustainability
  - We can improve all metrics by minimizing data movement

- **Huge problems with Memory Technology**
  - Memory technology scaling is not going well (e.g., RowHammer)
  - Many scaling issues demand intelligence in memory

- **Designs are squeezed in the middle**

RowHammer **[Kim et al., ISCA 2014]**

One can

predictably induce errors

in most DRAM memory chips

Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.


Rowhammer

RowHammer [ISCA 2014]

- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu,
  "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"
  *Proceedings of the 41st International Symposium on Computer Architecture (**ISCA**)*, Minneapolis, MN, June 2014.
  [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Source Code and Data] [Lecture Video (1 hr 49 mins), 25 September 2020]
  One of the 7 papers of 2012-2017 selected as Top Picks in Hardware and Embedded Security for IEEE TCAD (link). Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue). Winner of the 2024 IFIP Jean-Claude Laprie Award in dependable computing (link).

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

Yoongu Kim[1] Ross Daly* Jeremie Kim[1] Chris Fallin* Ji Hye Lee[1]
Donghyuk Lee[1] Chris Wilkerson[2] Konrad Lai Onur Mutlu[1]

[1]Carnegie Mellon University [2]Intel Labs

The Push from Circuits and Devices

Main Memory Needs

Intelligent Controllers

Industry's Intelligent DRAM Controllers (I)

ISSCC 2023 / SESSION 28 / HIGH-DENSITY MEMORIES /

28.8 A 1.1V 16Gb DDR5 DRAM with Probabilistic-Aggressor Tracking, Refresh-Management Functionality, Per-Row Hammer Tracking, a Multi-Step Precharge, and Core-Bias Modulation for Security and Reliability Enhancement

Woongrae Kim, Chulmoon Jung, Seongnyuh Yoo, Duckhwa Hong, Jeongjin Hwang, Jungmin Yoon, Ohyong Jung, Joonwoo Choi, Sanga Hyun, Mankeun Kang, Sangho Lee, Dohong Kim, Sanghyun Ku, Donhyun Choi, Nogeun Joo, Sangwoo Yoon, Junseok Noh, Byeongyong Go, Cheolhoe Kim, Sunil Hwang, Mihyun Hwang, Seol-Min Yi, Hyungmin Kim, Sanghyuk Heo, Yeonsu Jang, Kyoungchul Jang, Shinho Chu, Yoonna Oh, Kwidong Kim, Junghyun Kim, Soohwan Kim, Jeongtae Hwang, Sangil Park, Junphyo Lee, Inchul Jeong, Joohwan Cho, Jonghwan Kim

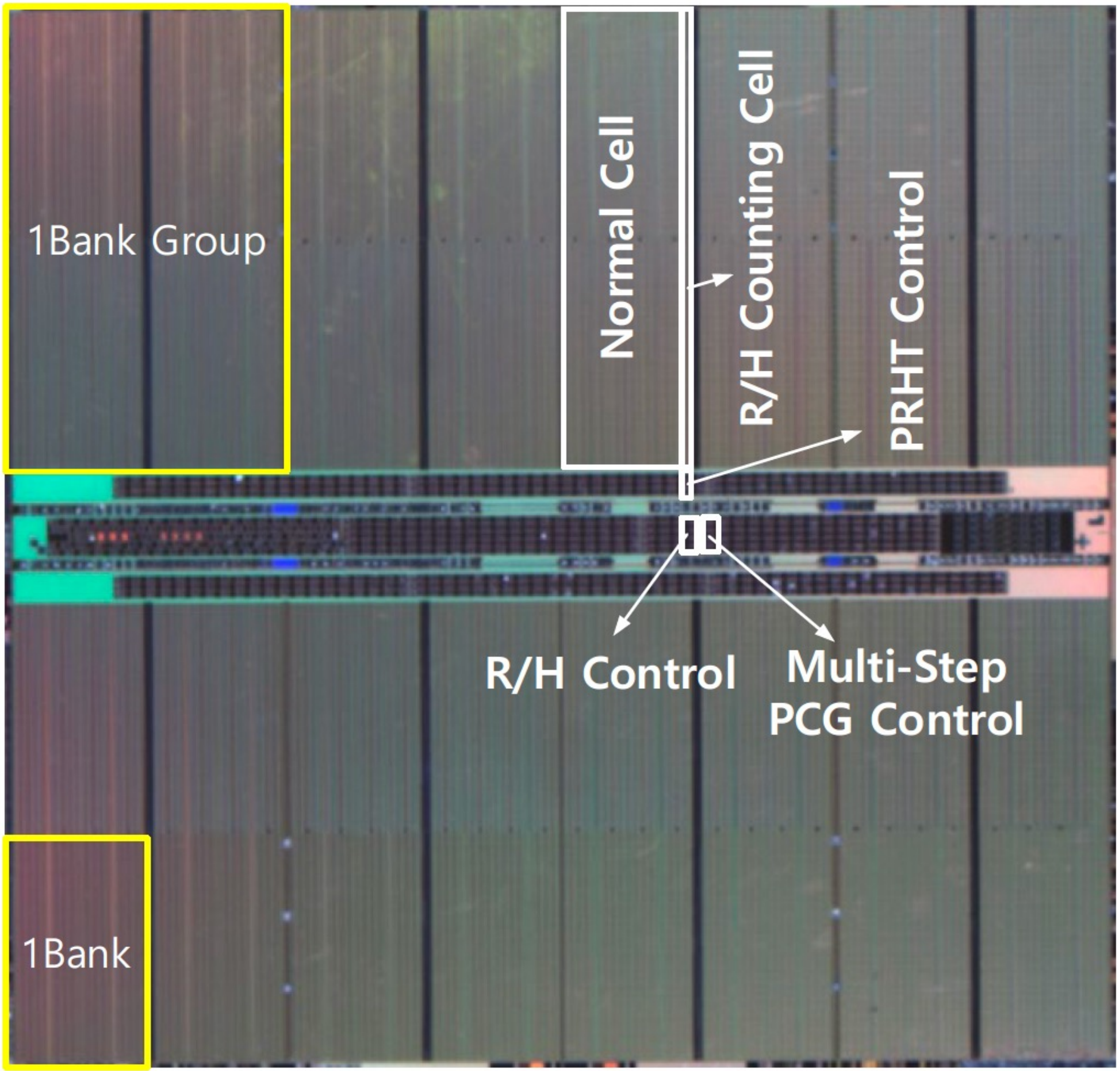
SK hynix Semiconductor, Icheon, Korea

Industry's Intelligent DRAM Controllers (II)

SK hynix Semiconductor, Icheon, Korea

DRAM products have been recently adopted in a wide range of high-performance computing applications: such as in cloud computing, in big data systems, and IoT devices. This demand creates larger memory capacity requirements, thereby requiring aggressive DRAM technology node scaling to reduce the cost per bit [1,2]. However, DRAM manufacturers are facing technology scaling challenges due to row hammer and refresh retention time beyond 1a-nm [2]. Row hammer is a failure mechanism, where repeatedly activating a DRAM row disturbs data in adjacent rows. Scaling down severely threatens reliability since a reduction of DRAM cell size leads to a reduction in the intrinsic row hammer tolerance [2,3]. To improve row hammer tolerance, there is a need to probabilistically activate adjacent rows with carefully sampled active addresses and to improve intrinsic row hammer tolerance [2]. In this paper, row-hammer-protection and refresh-management schemes are presented to guarantee DRAM security and reliability despite the aggressive scaling from 1a-nm to sub 10-nm nodes. The probabilistic-aggressor-tracking scheme with a refresh-management function (RFM) and per-row hammer tracking (PRHT) improve DRAM resilience. A multi-step precharge reinforces intrinsic row-hammer tolerance and a core-bias modulation improves retention time: even in the face of cell-transistor degradation due to technology scaling. This comprehensive scheme leads to a reduced probability of failure, due to row hammer attacks, by 93.1% and an improvement in retention time by 17%.

Industry's Intelligent DRAM Controllers (III)



ISSCC 2023 / SESSION 28 / HIGH-DENSITY MEMORIES /

28.8 A 1.1V 16Gb DDR5 DRAM with Probabilistic-Aggressor Tracking, Refresh-Management Functionality, Per-Row Hammer Tracking, a Multi-Step Precharge, and Core-Bias Modulation for Security and Reliability Enhancement

Woongrae Kim, Chulmoon Jung, Seongnyuh Yoo, Duckhwa Hong, Jeongjin Hwang, Jungmin Yoon, Ohyong Jung, Joonwoo Choi, Sanga Hyun, Mankeun Kang, Sangho Lee, Dohong Kim, Sanghyun Ku, Donhyun Choi, Nogeun Joo, Sangwoo Yoon, Junseok Noh, Byeongyong Go, Cheolhoe Kim, Sunil Hwang, Mihyun Hwang, Seol-Min Yi, Hyungmin Kim, Sanghyuk Heo, Yeonsu Jang, Kyoungchul Jang, Shinho Chu, Yoonna Oh, Kwidong Kim, Junghyun Kim, Soohwan Kim, Jeongtae Hwang, Sangil Park, Junphyo Lee, Inchul Jeong, Joohwan Cho, Jonghwan Kim

SK hynix Semiconductor, Icheon, Korea

RowHammer Solutions in JEDEC (2024)

JEDEC®

Global Standards for the Microelectronics Industry

STANDARDS & DOCUMENTS | COMMITTEES | NEWS | EVENTS & MEETINGS | JOI

DDR5 SDRAM JESD79-5C Apr 2024

Release Number: Version 1.30

Version 1.30

This standard defines the DDR5 SDRAM specification, including features, functionalities, AC and DC characteristics, packages, and ball/signal assignments. The purpose of this Standard is to define the minimum set of requirements for JEDEC compliant 8 Gb through 32 Gb for x4, x8, and x16 DDR5 SDRAM devices. This standard was created based on the DDR4 standards (JESD79-4) and some aspects of the DDR, DDR2, DDR3, and LPDDR4 standards (JESD79, JESD79-2, JESD79-3, and JESD209-4).

Committee(s): JC-42, JC-42.3

A RowHammer **Survey** Across the Stack

- Onur Mutlu and Jeremie Kim,
  "RowHammer: A Retrospective"
  *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (**TCAD**) Special Issue on Top Picks in Hardware and Embedded Security*, 2019.
  [Preliminary arXiv version]
  [Slides from COSADE 2019 (pptx)]
  [Slides from VLSI-SOC 2020 (pptx) (pdf)]
  [Talk Video (1 hr 15 minutes, with Q&A)]

RowHammer: A Retrospective

Onur Mutlu[§‡] Jeremie S. Kim[‡§]
[§]ETH Zürich [‡]Carnegie Mellon University

A RowHammer Survey: Recent Update

- Onur Mutlu, Ataberk Olgun, and A. Giray Yaglikci,
  "Fundamentally Understanding and Solving RowHammer"
  *Invited Special Session Paper at the 28th Asia and South Pacific Design Automation Conference (**ASP-DAC**)*, Tokyo, Japan, January 2023.
  [arXiv version]
  [Slides (pptx) (pdf)]
  [Talk Video (26 minutes)]

Fundamentally Understanding and Solving RowHammer

Onur Mutlu
onur.mutlu@safari.ethz.ch
ETH Zürich
Zürich, Switzerland

Ataberk Olgun
ataberk.olgun@safari.ethz.ch
ETH Zürich
Zürich, Switzerland

A. Giray Yağlıkcı
giray.yaglikci@safari.ethz.ch
ETH Zürich
Zürich, Switzerland

https://arxiv.org/pdf/2211.07613.pdf

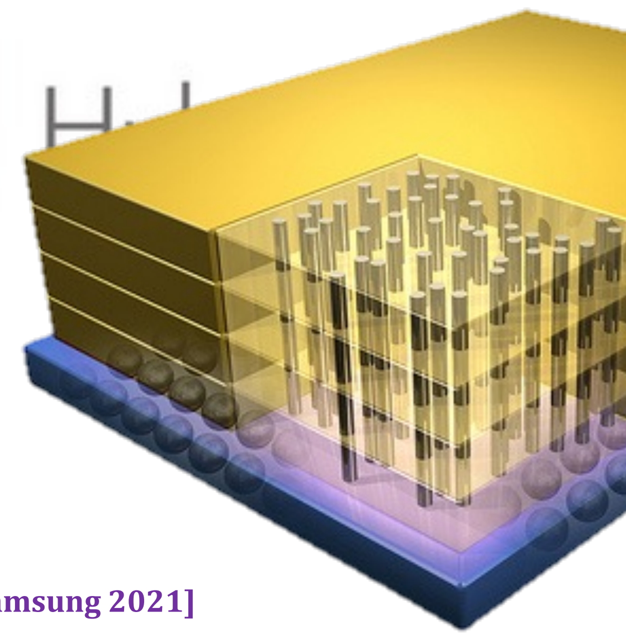
RowPress [ISCA 2023]

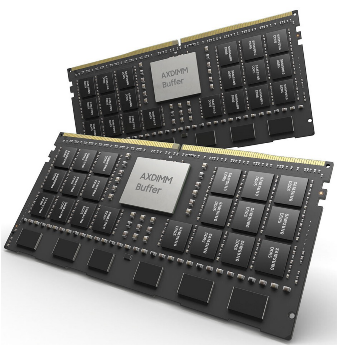
- Haocong Luo, Ataberk Olgun, Giray Yaglikci, Yahya Can Tugrul, Steve Rhyner, M. Banu Cavlak, Joel Lindegger, Mohammad Sadrosadati, and Onur Mutlu,
  "RowPress: Amplifying Read Disturbance in Modern DRAM Chips"
  *Proceedings of the 50th International Symposium on Computer Architecture (**ISCA**)*, Orlando, FL, USA, June 2023.
  [Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Lightning Talk Video (3 minutes)]
  [RowPress Source Code and Datasets (Officially Artifact Evaluated with All Badges)]
  Officially artifact evaluated as available, reusable and reproducible.
  Best artifact award at ISCA 2023. IEEE Micro Top Pick in 2024.

RowPress: Amplifying Read-Disturbance in Modern DRAM Chips

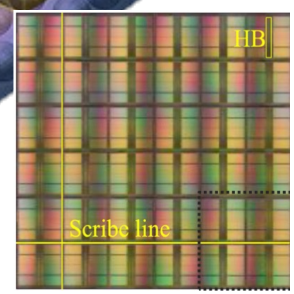
Haocong Luo Ataberk Olgun A. Giray Yağlıkçı Yahya Can Tuğrul Steve Rhyner
Meryem Banu Cavlak Joël Lindegger Mohammad Sadrosadati Onur Mutlu
ETH Zürich

Processing-in-Memory Landscape Today

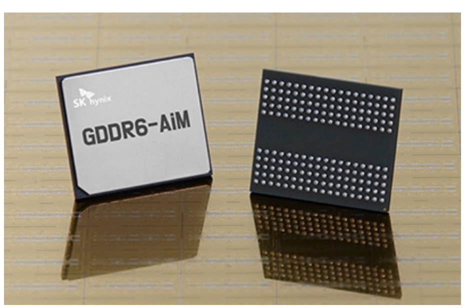




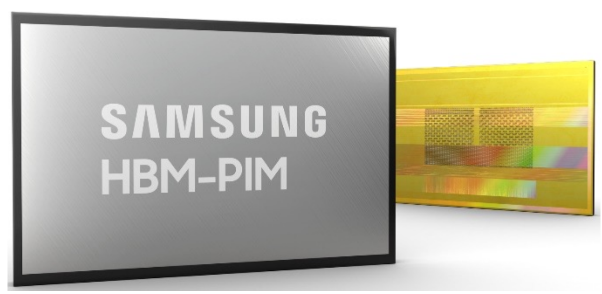
[Samsung 2021]



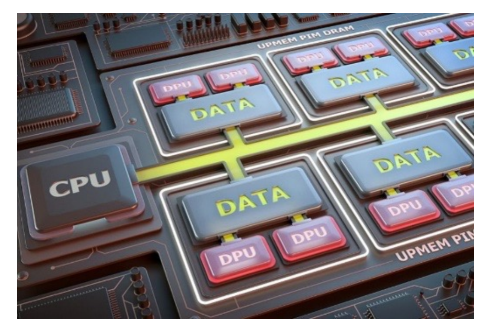
[Alibaba 2022]



[SK Hynix 2022]



[Samsung 2021]



[UPMEM 2019]

And, many other experimental chips and startups

Processing-in-Memory Landscape Today

IEEE COMPUTER ARCHITECTURE LETTERS, VOL. 22, NO. 1, JANUARY-JUNE

Computational CXL-Memory Solution for Accelerating Memory-Intensive Applications

Joonseop Sim, Soohong Ahn, Taeyoung Ahn, Seungyong Lee, Myunghyun Rhee, Jooyoung Kim, Kwangsik Shin, Donguk Moon, Euiseok Kim, and Kyoung Park

Abstract—CXL interface is the up-to-date technology that enables effective memory expansion by providing a memory-sharing protocol in configuring heterogeneous devices. However, its limited physical bandwidth can be a significant bottleneck for emerging data-intensive applications. In this work, we propose a novel CXL-based memory disaggregation architecture with a real-world prototype demonstration, which overcomes the bandwidth limitation of the CXL interface using near-data processing. The experimental results demonstrate that our design achieves up to 1.9× better performance/power efficiency than the existing CPU system.

Index Terms—Compute express link (CXL), near-data-processing (NDP)

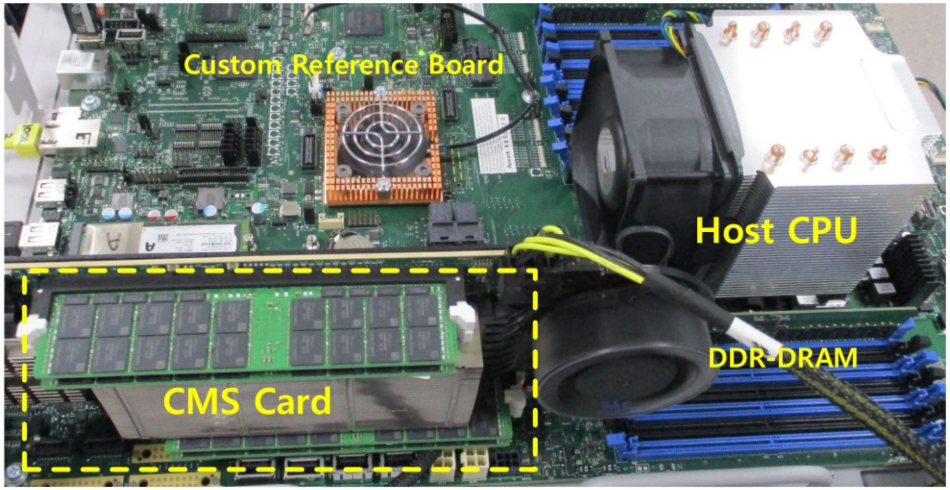


Fig. 6. FPGA prototype of proposed CMS card.

Processing-in-Memory Landscape Today

Samsung Processing in Memory Technology at Hot Chips 2023

By **Patrick Kennedy** - August 28, 2023

Samsung PIM PNM For Transformer Based AI HC35_Page_24

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

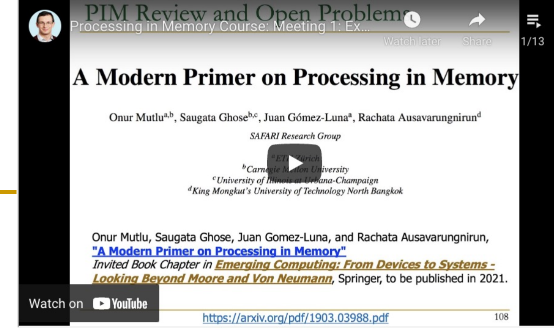
[a] *ETH Zürich*
[b] *Carnegie Mellon University*
[c] *University of Illinois at Urbana-Champaign*
[d] *King Mongkut's University of Technology North Bangkok*

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.

https://arxiv.org/pdf/2012.03112.pdf

PIM Course (Fall 2022)

- **Fall 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=processing_in_memory
- **Spring 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=processing_in_memory
- **Youtube Livestream (Fall 2022):**
  - https://www.youtube.com/watch?v=QLL0wQ9I4Dw&list=PL5Q2soXY2Zi8KzG2CQYRNQOVD0GOBrnKy
- **Youtube Livestream (Spring 2022):**
  - https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi-841fUYYUK9EsXKhQKRPyX
- Project course
  - Taken by Bachelor's/Master's students
  - Processing-in-Memory lectures
  - Hands-on research exploration
  - Many research readings



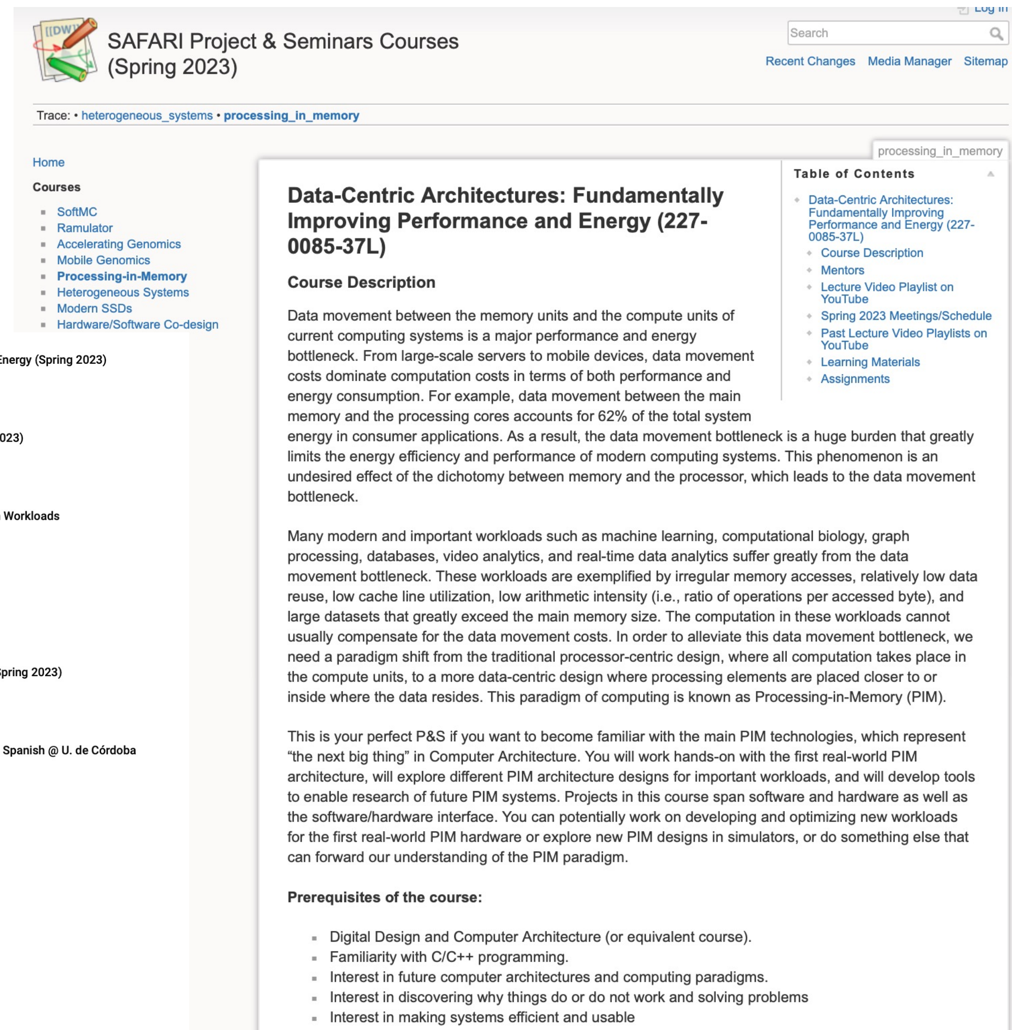
Spring 2022 Meetings/Schedule

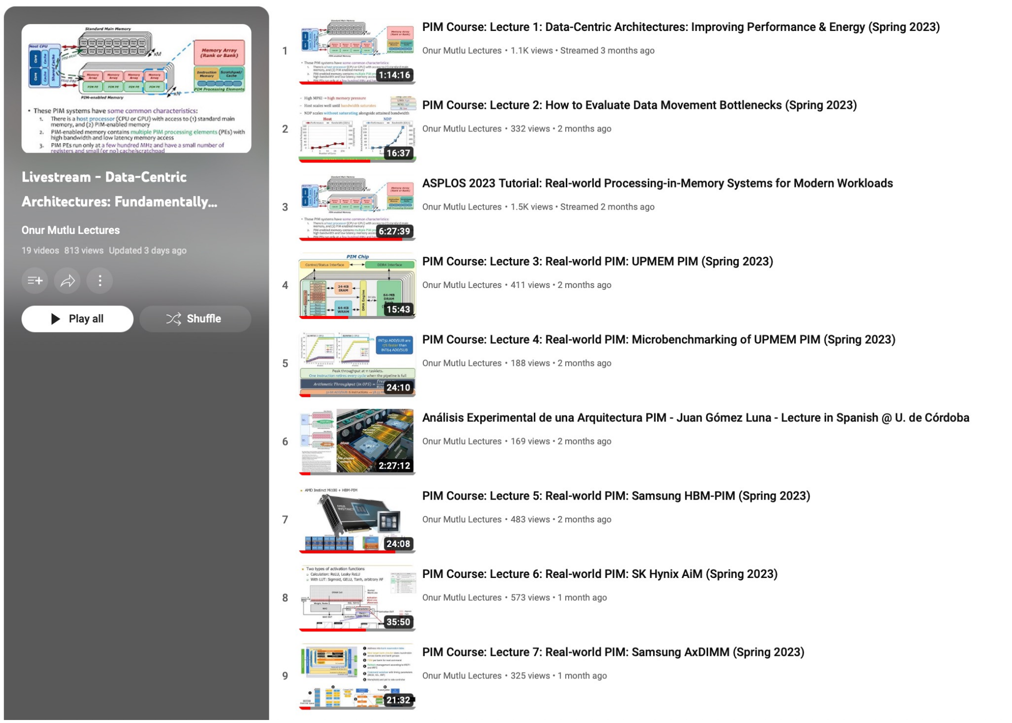
Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	10.03 Thu.	YouTube Live	M1: P&S PIM Course Presentation (PDF) (PPT)	Required Materials Recommended Materials	HW 0 Out
W2	15.03 Tue.		Hands-on Project Proposals		
	17.03 Thu.	YouTube Premiere	M2: Real-world PIM: UPMEM PIM (PDF) (PPT)		
W3	24.03 Thu.	YouTube Live	M3: Real-world PIM: Microbenchmarking of UPMEM PIM (PDF) (PPT)		
W4	31.03 Thu.	YouTube Live	M4: Real-world PIM: Samsung HBM-PIM (PDF) (PPT)		
W5	07.04 Thu.	YouTube Live	M5: How to Evaluate Data Movement Bottlenecks (PDF) (PPT)		
W6	14.04 Thu.	YouTube Live	M6: Real-world PIM: SK Hynix AiM (PDF) (PPT)		
W7	21.04 Thu.	YouTube Premiere	M7: Programming PIM Architectures (PDF) (PPT)		
W8	28.04 Thu.	YouTube Premiere	M8: Benchmarking and Workload Suitability on PIM (PDF) (PPT)		
W9	05.05 Thu.	YouTube Premiere	M9: Real-world PIM: Samsung AxDIMM (PDF) (PPT)		
W10	12.05 Thu.	YouTube Premiere	M10: Real-world PIM: Alibaba HB-PNM (PDF) (PPT)		
W11	19.05 Thu.	YouTube Live	M11: SpMV on a Real PIM Architecture (PDF) (PPT)		
W12	26.05 Thu.	YouTube Live	M12: End-to-End Framework for Processing-using-Memory (PDF) (PPT)		
W13	02.06 Thu.	YouTube Live	M13: Bit-Serial SIMD Processing using DRAM (PDF) (PPT)		
W14	09.06 Thu.	YouTube Live	M14: Analyzing and Mitigating ML Inference Bottlenecks (PDF) (PPT)		
W15	15.06 Thu.	YouTube Live	M15: In-Memory HTAP Databases with HW/SW Co-design (PDF) (PPT)		
W16	23.06 Thu.	YouTube Live	M16: In-Storage Processing for Genome Analysis (PDF) (PPT)		
W17	18.07 Mon.	YouTube Premiere	M17: How to Enable the Adoption of PIM? (PDF) (PPT)		
W18	09.08 Tue.	YouTube Premiere	SS1: ISVLSI 2022 Special Session on PIM (PDF & PPT)		

https://www.youtube.com/onurmutlulectures

Processing-in-Memory Course (Spring 2023)

- Short weekly lectures
- Hands-on projects



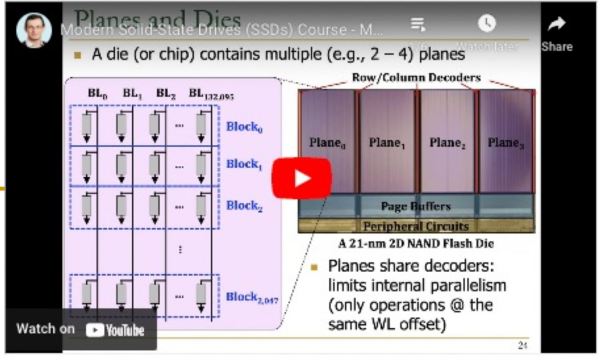


https://www.youtube.com/playlist?list=PL5Q2soXY2Zi_EObuoAZVSq_o6UySWQHvZ

https://safari.ethz.ch/projects_and_seminars/spring2023/doku.php?id=processing_in_memory

SSD Course (Spring 2023)

- **Spring 2023 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/spring2023/doku.php?id=modern_ssds
- **Fall 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=modern_ssds
- **Youtube Livestream (Spring 2023):**
  - https://www.youtube.com/watch?v=4VTwOMmsnJY&list=PL5Q2soXY2Zi_8qOM5Icpp8hB2SHtm4z57&pp=iAQB
- **Youtube Livestream (Fall 2022):**
  - https://www.youtube.com/watch?v=hqLrd-Uj0aU&list=PL5Q2soXY2Zi9BJhenUq4JI5bwhAMpAp13&pp=iAQB
- Project course
  - Taken by Bachelor's/Master's students
  - SSD Basics and Advanced Topics
  - Hands-on research exploration
  - Many research readings



Fall 2022 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	06.10		M1: P&S Course Presentation PDF PPT	Required Recommended	
W2	12.10	YouTube Live	M2: Basics of NAND Flash-Based SSDs PDF PPT	Required Recommended	
W3	19.10	YouTube Live	M3: NAND Flash Read/Write Operations PDF PPT	Required Recommended	
W4	26.10	YouTube Live	M4: Processing inside NAND Flash PDF PPT	Required Recommended	
W5	02.11	YouTube Live	M5: Advanced NAND Flash Commands & Mapping PDF PPT	Required Recommended	
W6	09.11	YouTube Live	M6: Processing inside Storage PDF PPT	Required Recommended	
W7	23.11	YouTube Live	M7: Address Mapping & Garbage Collection PDF PPT	Required Recommended	
W8	30.11	YouTube Live	M8: Introduction to MQSim PDF PPT	Required Recommended	
W9	14.12	YouTube Live	M9: Fine-Grained Mapping and Multi-Plane Operation-Aware Block Management PDF PPT	Required Recommended	
W10	04.01.2023	YouTube Premiere	M10a: NAND Flash Basics PDF PPT	Required Recommended	
			M10b: Reducing Solid-State Drive Read Latency by Optimizing Read-Retry PDF PPT Paper	Required Recommended	
			M10c: Evanesco: Architectural Support for Efficient Data Sanitization in Modern Flash-Based Storage Systems PDF PPT Paper	Required Recommended	
			M10d: DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression PDF PPT Paper	Required Recommended	
W11	11.01	YouTube Live	M11: FLIN: Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives PDF PPT	Required	
W12	25.01	YouTube Premiere	M12: Flash Memory and Solid-State Drives PDF PPT	Recommended	

https://www.youtube.com/onurmutlulectures

PIM Tutorials [MICRO'23, ISCA'23, ASPLOS'23, HPCA'23, ISCA'24]

- Lectures + Hands-on labs + Invited talks

ISCA 2023 Real-World PIM Tutorial

Search

Recent Changes Media Manager Sitemap

Trace: • **start**

start

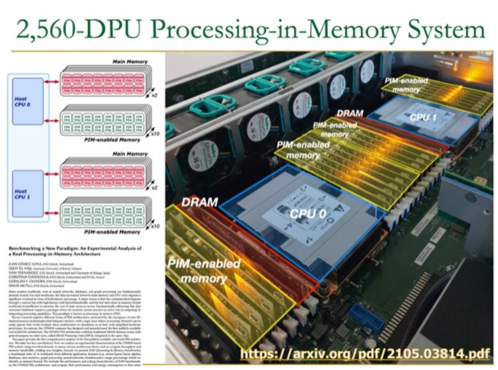
Real-world Processing-in-Memory Systems for Modern Workloads

Tutorial Description

Processing-in-Memory (PIM) is a computing paradigm that aims at overcoming the data movement bottleneck (i.e., the waste of execution cycles and energy resulting from the back-and-forth data movement between memory units and compute units) by making memory compute-capable.

Explored over several decades since the 1960s, PIM systems are becoming a reality with the advent of the first commercial products and prototypes.

A number of startups (e.g., UPMEM, Neuroblade) are already commercializing real PIM hardware, each with its own design approach and target applications. Several major vendors (e.g., Samsung, SK Hynix, Alibaba) have presented real PIM chip prototypes in the last two years. Most of these architectures have in common that they place compute units near the memory arrays. This type of PIM is called processing near memory (PNM).



PIM can provide large improvements in both performance and energy consumption for many modern applications, thereby enabling a commercially viable way of dealing with huge amounts of data that is bottlenecking our computing systems. Yet, it is critical to (1) study and understand the characteristics that make a workload suitable for a PIM architecture, (2) propose optimization strategies for PIM kernels, and (3) develop programming frameworks and tools that can lower the learning curve and ease the adoption of PIM.

This tutorial focuses on the latest advances in PIM technology, workload characterization for PIM, and programming and optimizing PIM kernels. We will (1) provide an introduction to PIM and taxonomy of PIM systems, (2) give an overview and a rigorous analysis of existing real-world PIM hardware, (3) conduct hand-on labs about important workloads (machine learning, sparse linear algebra, bioinformatics, etc.) using real PIM systems, and (4) shed light on how to improve future PIM systems for such workloads.

Table of Contents

https://www.youtube.com/live/GIb5EgSrWk0

https://events.safari.ethz.ch/isca-pim-tutorial/

PIM Tutorial at ISCA 2024

ISCA 2024 Memory-Centric Computing Systems Tutorial

Saturday, June 29, Buenos Aires, Argentina

Organizers: Geraldo F. Oliveira, Dr. Mohammad Sadrosadati, Ataberk Olgun, Professor Onur Mutlu

Program: https://events.safari.ethz.ch/isca24-memorycentric-tutorial/

Overview of PIM | PIM taxonomy

PIM in memory & storage

Real-world PNM systems

PUM for bulk bitwise operations

Programming techniques & tools

Infrastructures for PIM Research

Research challenges & opportunities

https://www.youtube.com/watch?v=KV2MXvcBgb0

https://events.safari.ethz.ch/isca24-memorycentric-tutorial

We Need to Think Differently from the Past Approaches

Processing in Memory: Two Approaches

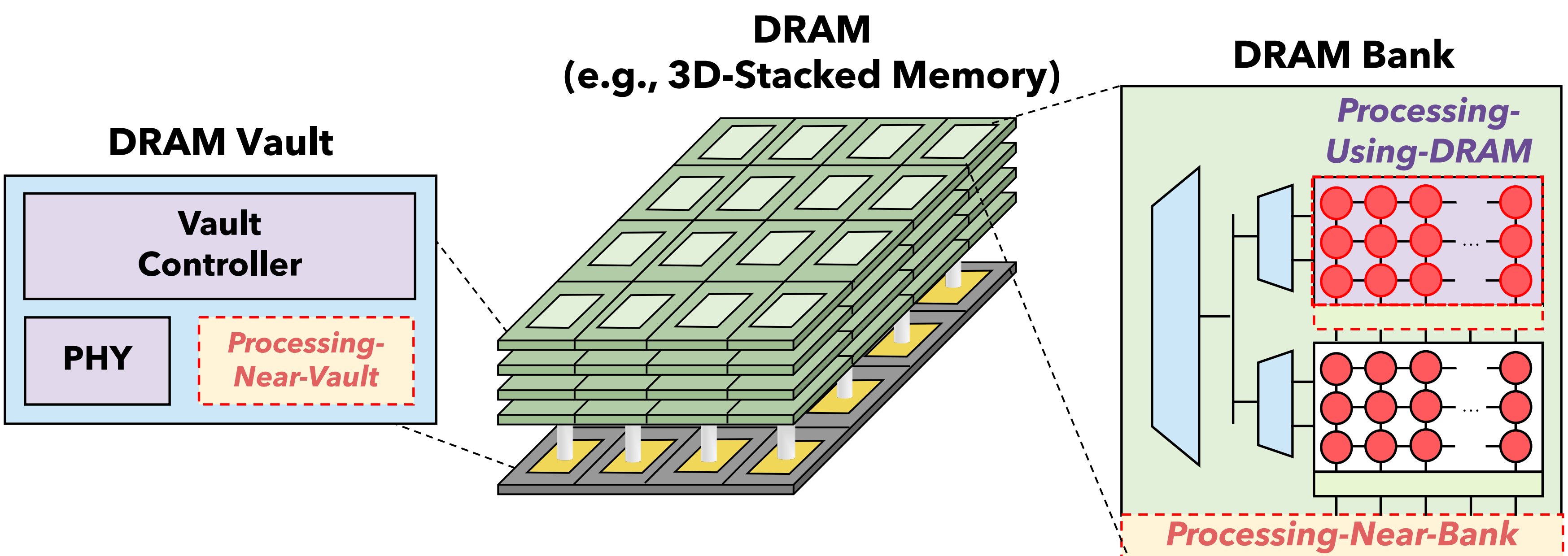
1. Processing **using** Memory
2. Processing **near** Memory

Processing-in-Memory:
Nature of Computation

Two main approaches for Processing-in-Memory:

1 **Processing-Near-Memory**: Design compute logic and memory separately (as today) and integrate logic closer to memory

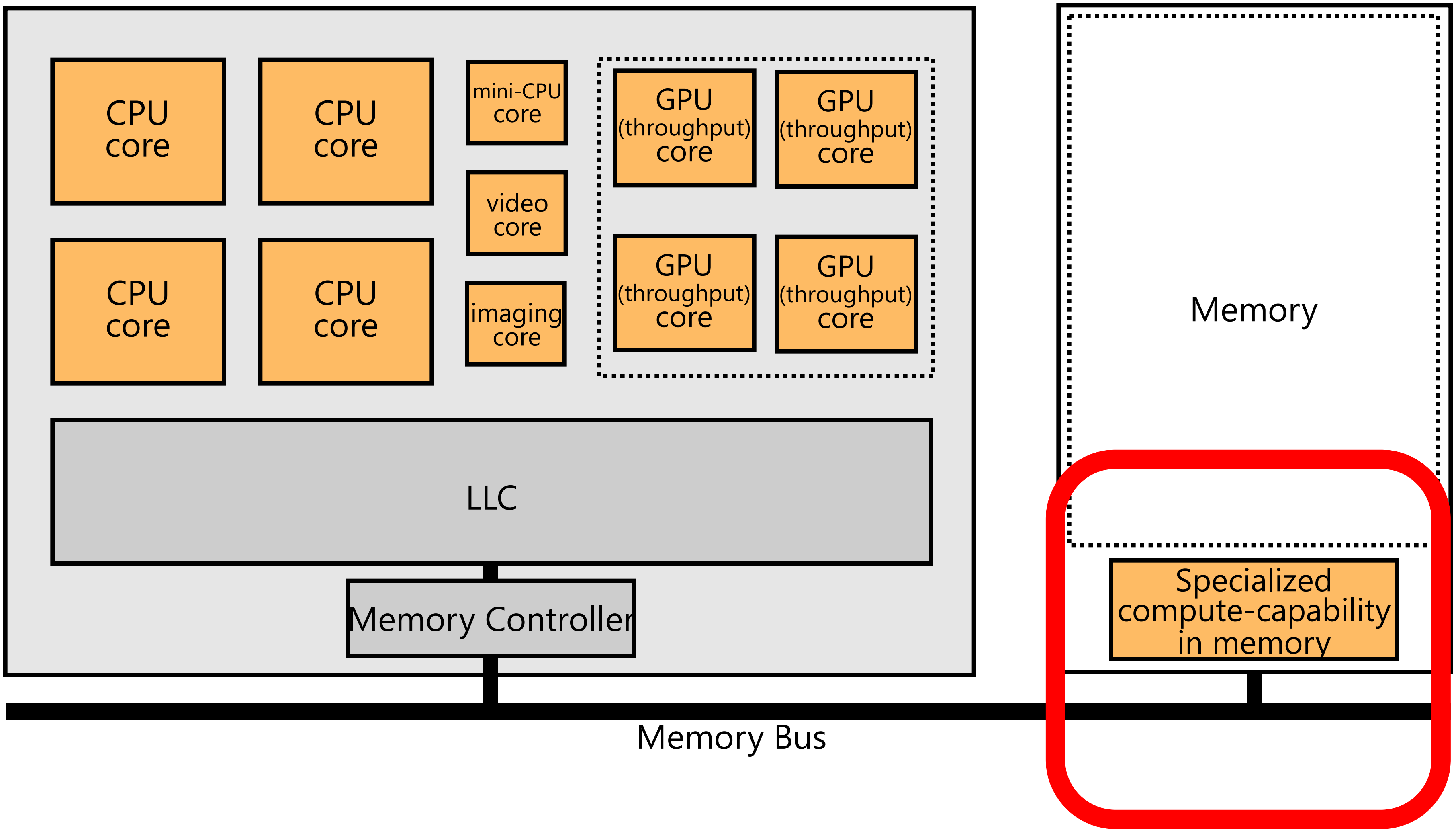
2 **Processing-Using-Memory**: Use analog operational principles of memory circuitry to perform computation (no compute logic)



A PIM Taxonomy

- **Nature** (of computation)
  - Using: Use operational properties of memory structures
  - Near: Add logic close to memory structures
- **Technology**
  - Flash, DRAM, SRAM, RRAM, MRAM, FeRAM, PCM, 3D, …
- **Location**
  - Sensor, Cold Storage, Hard Disk, SSD, Main Memory, Cache, Register File, Memory Controller, Interconnect, …
- A tuple of the three determines "PIM type"
- One can combine multiple "PIM types" in a system

Mindset: Memory as an Accelerator



Memory similar to a "conventional" accelerator

Example PIM Type: Processing using DRAM

- Nature: Using
- Technology: DRAM
- Location: Main Memory

- Processing using DRAM in Main Memory

- Seshadri+, "Fast Bulk Bitwise AND and OR in DRAM", IEEE CAL 2015.
- Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," MICRO 2017.
- Hajinazar+, "SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM," ASPLOS 2021.
- Oliveira+, "MIMDRAM: An End-to-End Processing-Using-DRAM System for High-Throughput, Energy-Efficient and Programmer-Transparent Multiple-Instruction Multiple-Data Processing," HPCA 2024.

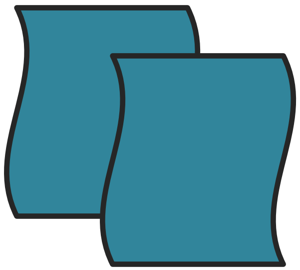
Processing using DRAM

- We can support
  - Bulk bitwise AND, OR, NOT, MAJ
  - Bulk bitwise COPY and INIT/ZERO
  - True Random Number Generation; Physical Unclonable Functions
  - Lookup Table based more complex computation
- At low cost
- Using analog computation capability of DRAM
  - Idea: activating (multiple) rows performs computation
    - Even in commodity off-the-shelf DRAM chips!

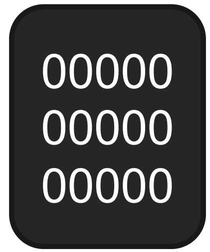
- 30-77X performance and energy improvement
  - Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," MICRO 2017.
  - Seshadri+"RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data," MICRO 2013.

Starting Simple: Data Copy and Initialization

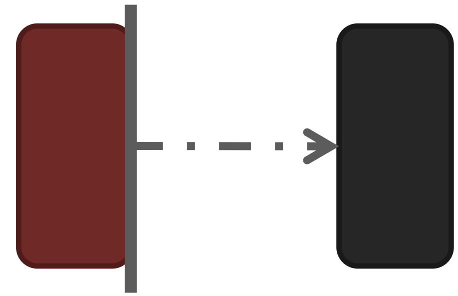
memmove & *memcpy*: 5% cycles in Google's datacenter [Kanev+ ISCA'15]



Forking

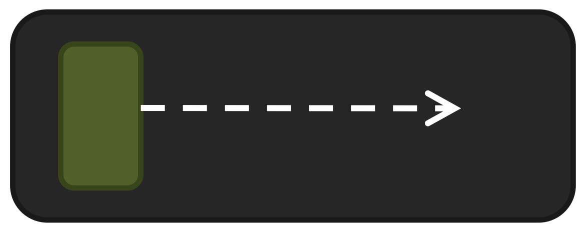


Zero initialization
(e.g., security)



Checkpointing

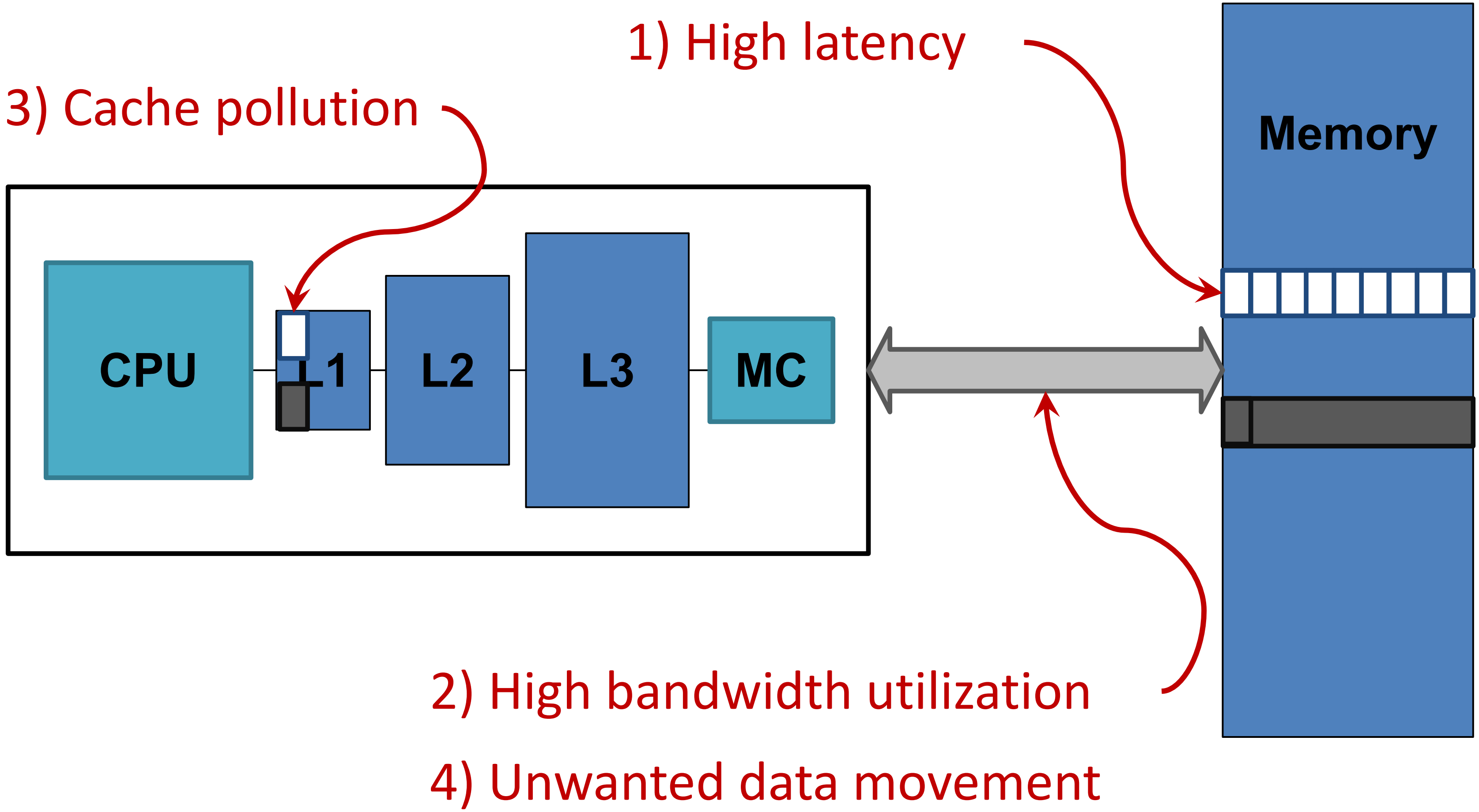
VM Cloning
Deduplication



Page Migration

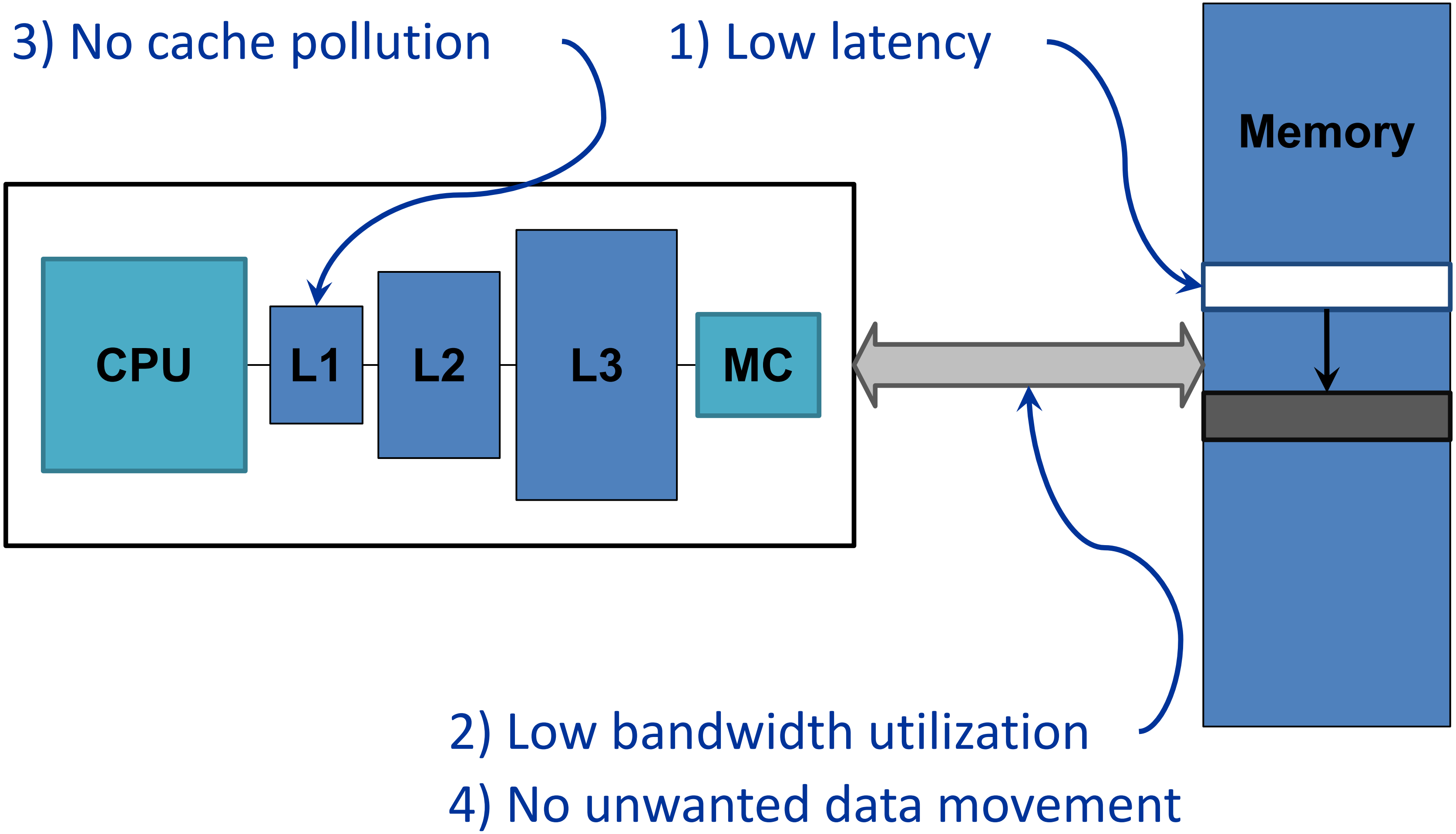
...
Many more

Today's Systems: Bulk Data Copy



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

Future Systems: In-Memory Copy

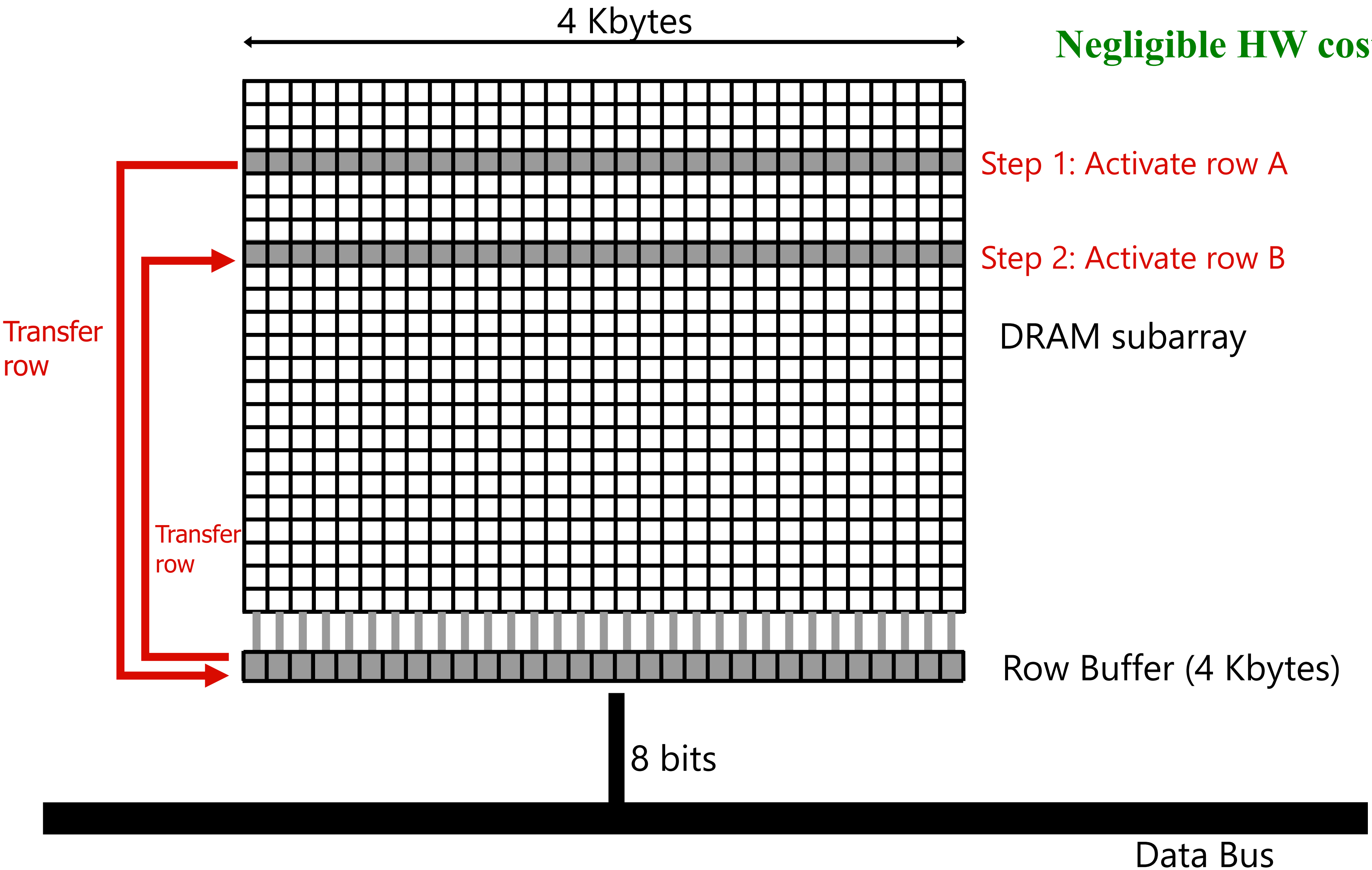


1046ns, 3.6uJ → 90ns, 0.04uJ

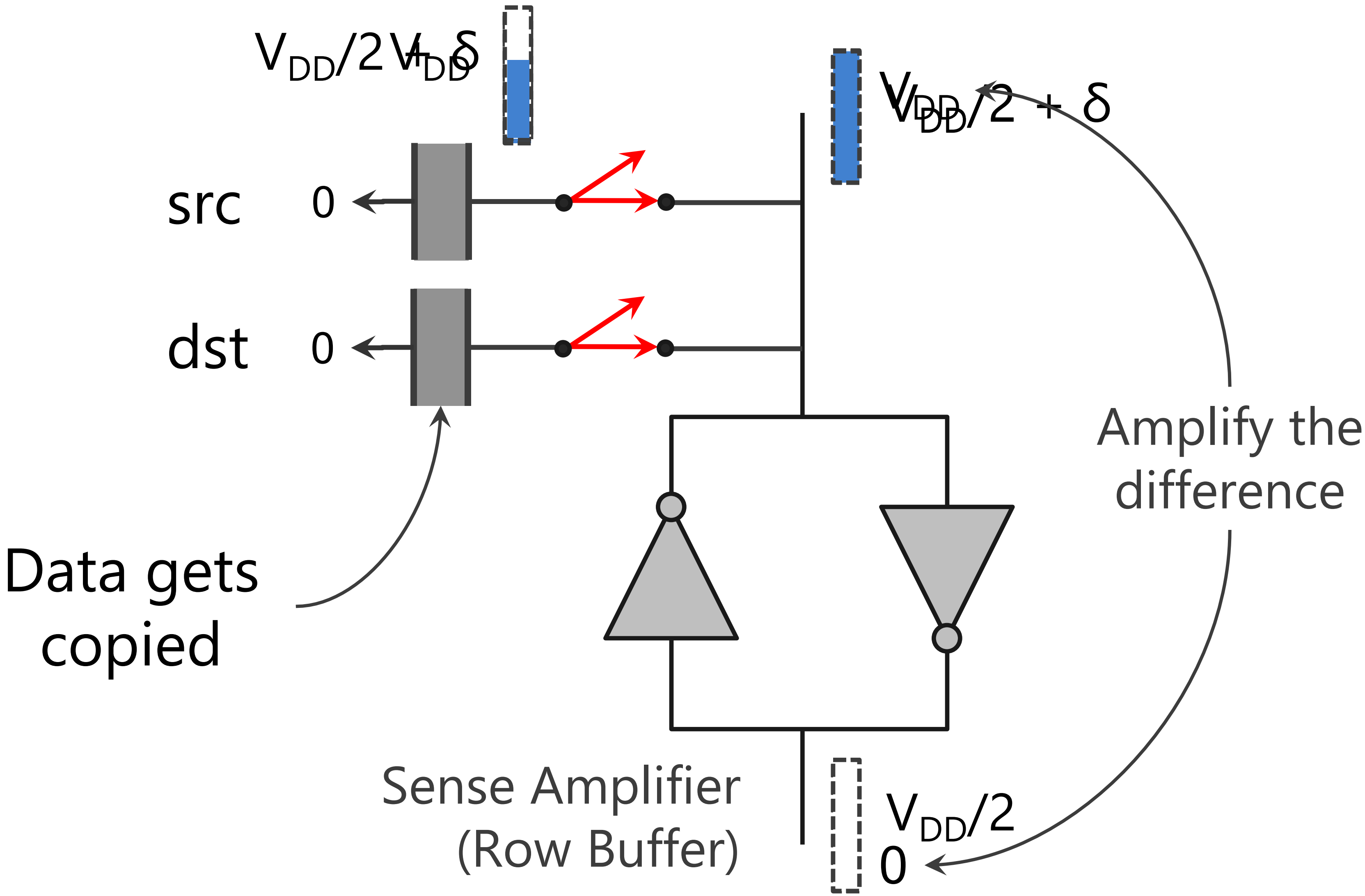
RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates

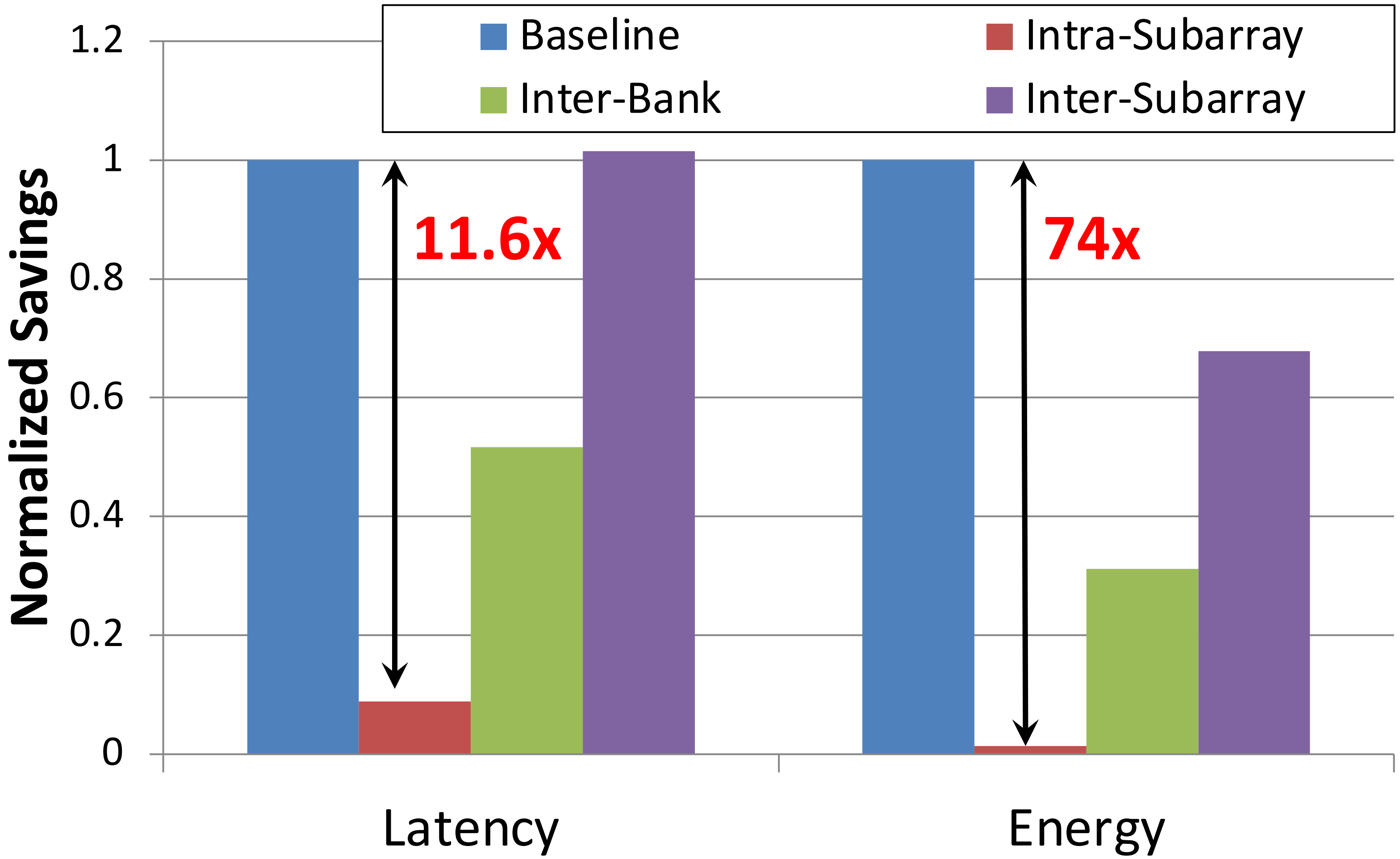
Negligible HW cost



RowClone: Intra-Subarray



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 in Off-the-Shelf DRAM Chips

- Idea: Violate DRAM timing parameters to mimic RowClone

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

https://parallel.princeton.edu/papers/micro19-gao.pdf

Real Processing Using Memory Prototype

- End-to-end RowClone & TRNG using off-the-shelf DRAM chips
- Idea: Violate DRAM timing parameters to mimic RowClone

PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM

Ataberk Olgun[§,†] Juan Gómez Luna[§] Konstantinos Kanellopoulos[§] Behzad Salami[§,*]
Hasan Hassan[§] Oğuz Ergin[†] Onur Mutlu[§]

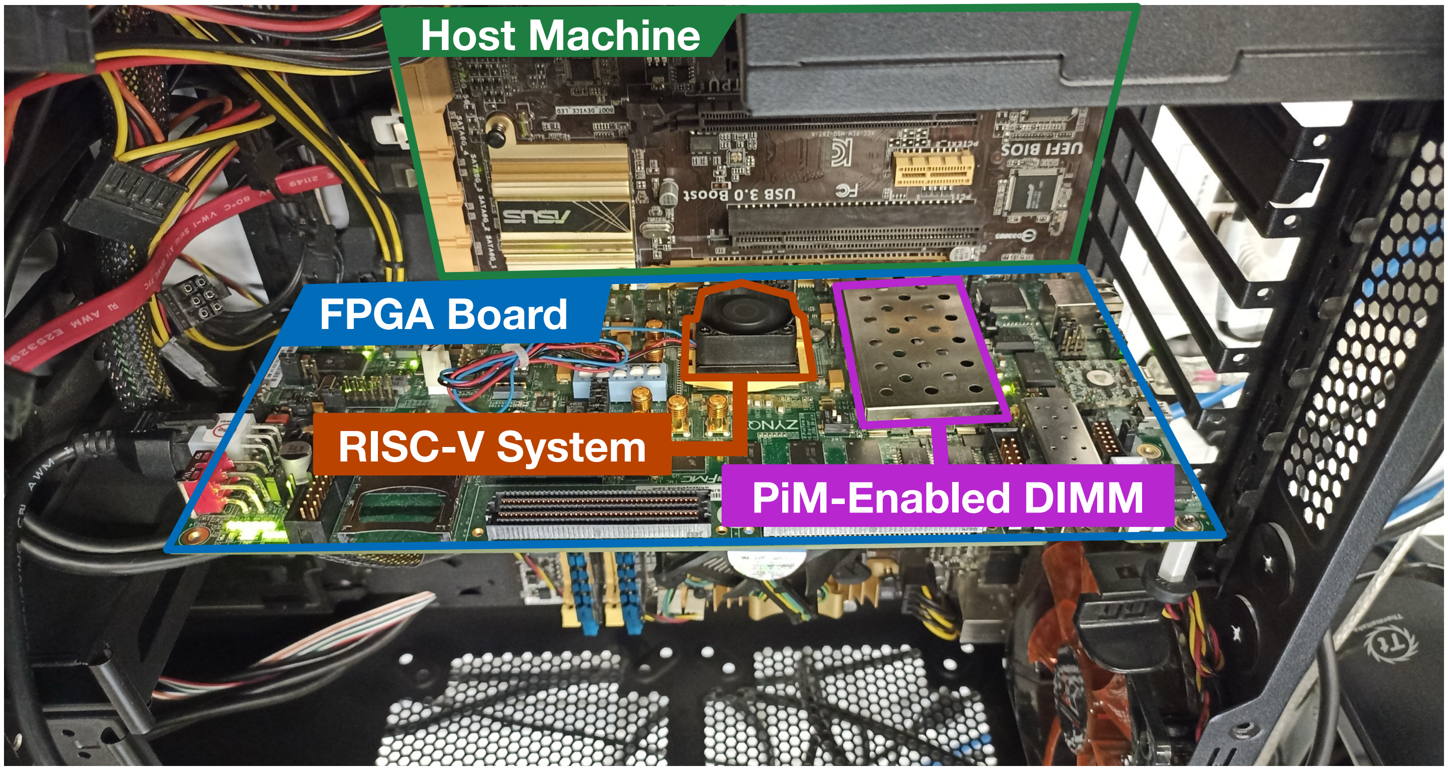
[§]ETH Zürich [†]TOBB ETÜ [*]BSC

https://arxiv.org/pdf/2111.00082.pdf

https://github.com/cmu-safari/pidram

https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Real Processing-using-Memory Prototype



https://arxiv.org/pdf/2111.00082.pdf

https://github.com/cmu-safari/pidram

https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Real Processing-using-Memory Prototype

README.md

Building a PiDRAM Prototype

To build PiDRAM's prototype on Xilinx ZC706 boards, developers need to use the two sub-projects in this directory. `fpga-zynq` is a repository branched off of UCB-BAR's fpga-zynq repository. We use `fpga-zynq` to generate rocket chip designs that support end-to-end DRAM PuM execution. `controller-hardware` is where we keep the main Vivado project and Verilog sources for PiDRAM's memory controller and the top level system design.

Rebuilding Steps

1. Navigate into `fpga-zynq` and read the README file to understand the overall workflow of the repository
   - Follow the readme in `fpga-zynq/rocket-chip/riscv-tools` to install dependencies
2. Create the Verilog source of the rocket chip design using the `ZynqCopyFPGAConfig`
   - Navigate into zc706, then run `make rocket CONFIG=ZynqCopyFPGAConfig -j<number of cores>`
3. Copy the generated Verilog file (should be under zc706/src) and overwrite the same file in `controller-hardware/source/hdl/impl/rocket-chip`
4. Open the Vivado project in `controller-hardware/Vivado_Project` using Vivado 2016.2
5. Generate a bitstream
6. Copy the bitstream (system_top.bit) to `fpga-zynq/zc706`
7. Use the `./build_script.sh` to generate the new `boot.bin` under `fpga-images-zc706`, you can use this file to program the FPGA using the SD-Card
   - For details, follow the relevant instructions in `fpga-zynq/README.md`

You can run programs compiled with the RISC-V Toolchain supplied within the `fpga-zynq` repository. To install the toolchain, follow the instructions under `fpga-zynq/rocket-chip/riscv-tools`.

Generating DDR3 Controller IP sources

We cannot provide the sources for the Xilinx PHY IP we use in PiDRAM's memory controller due to licensing issues. We describe here how to regenerate them using Vivado 2016.2. First, you need to generate the IP RTL files:

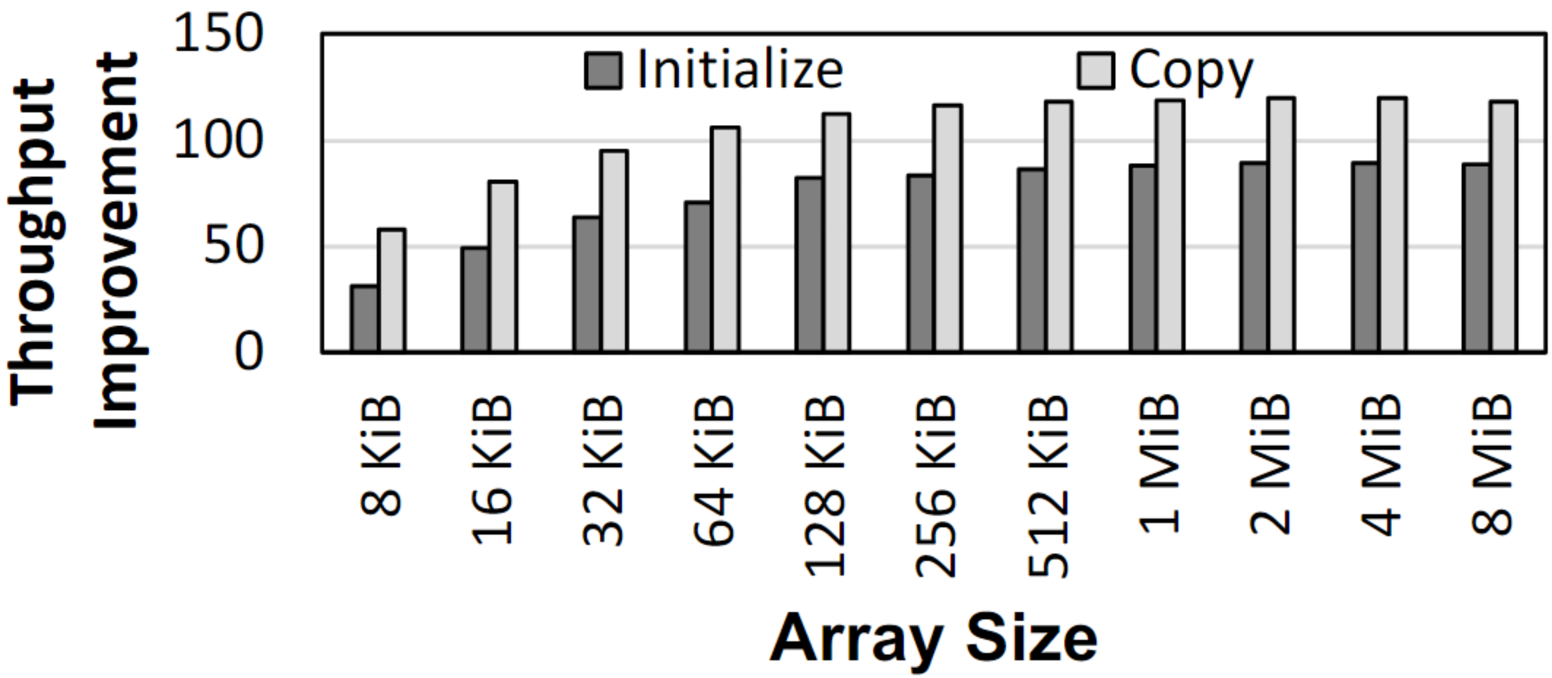
1- Open IP Catalog
2- Find "Memory Interface Generator (MIG 7 Series)" IP and double click

https://arxiv.org/pdf/2111.00082.pdf

https://github.com/cmu-safari/pidram

https://www.youtube.com/watch?v=qeukNs5XI3g&t=4192s

Microbenchmark Copy/Initialization Throughput



In-DRAM Copy and Initialization improve throughput by 119x and 89x

More on PiDRAM

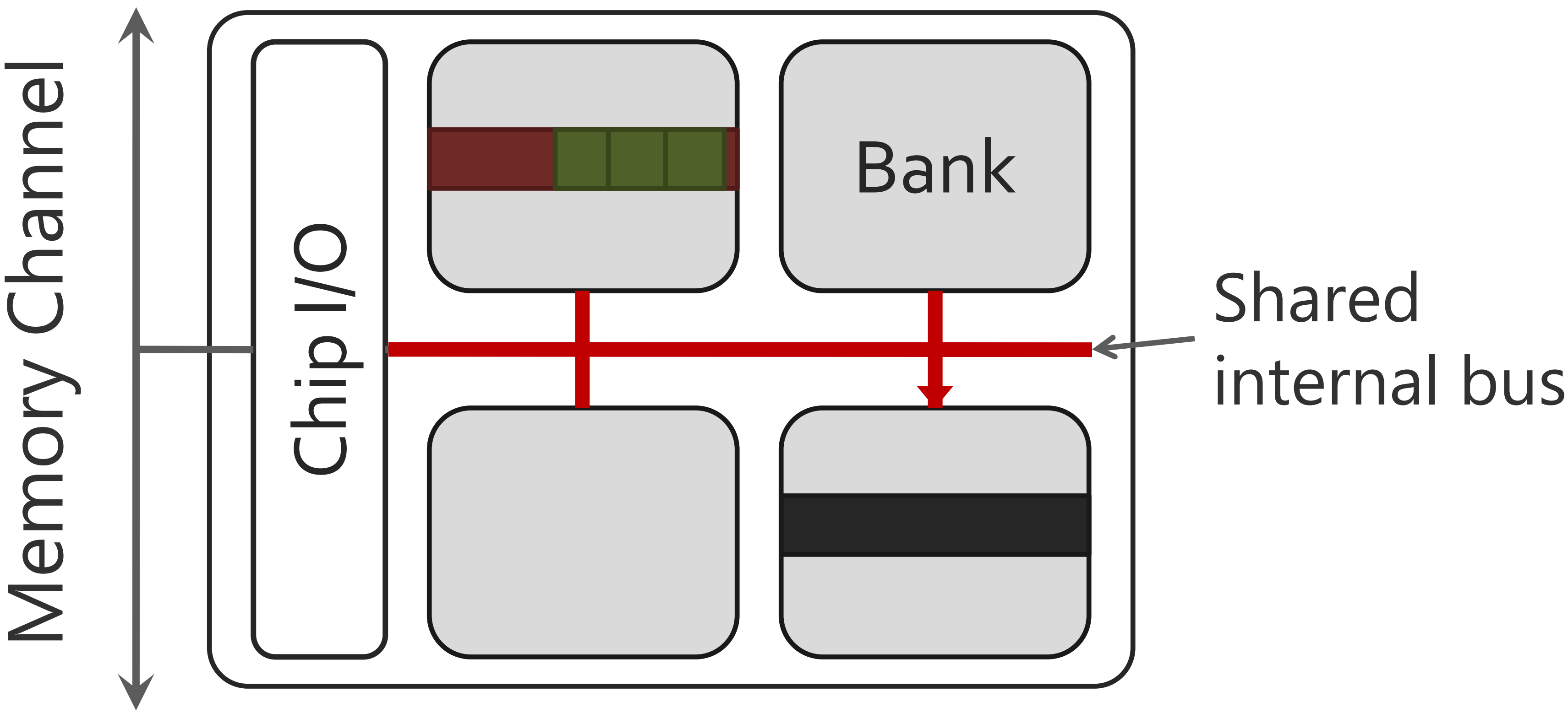
- Ataberk Olgun, Juan Gomez Luna, Konstantinos Kanellopoulos, Behzad Salami, Hasan Hassan, Oguz Ergin, and Onur Mutlu,
  "PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM"
  ACM Transactions on Architecture and Code Optimization (***TACO***), March 2023.
  [arXiv version]
  Presented at the 18th HiPEAC Conference, Toulouse, France, January 2023.
  [Slides (pptx) (pdf)]
  [Longer Lecture Slides (pptx) (pdf)]
  [Lecture Video (40 minutes)]
  [PiDRAM Source Code]

PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM

Ataberk Olgun[§] Juan Gómez Luna[§] Konstantinos Kanellopoulos[§] Behzad Salami[§]
Hasan Hassan[§] Oğuz Ergin[†] Onur Mutlu[§]

[§]*ETH Zürich* [†]*TOBB University of Economics and Technology*

RowClone: Inter-Bank



Overlap the latency of the read and the write
1.9X latency reduction, 3.2X energy reduction

RowClone Extensions and Follow-Up Work

- Can this be improved to do faster inter-subarray copy?
  - Yes, see LISA [Chang et al., HPCA 2016]
- Can we enable data movement at smaller granularities within a bank?
  - Yes, see FIGARO [Wang et al., MICRO 2020]
- Can this be improved to do better inter-bank copy?
  - Yes, see Network-on-Memory [CAL 2020]
- Can similar ideas and DRAM properties be used to perform computation on data?
  - Yes, see Ambit [Seshadri et al., CAL 2015, MICRO 2017]

LISA: Increasing Connectivity in DRAM

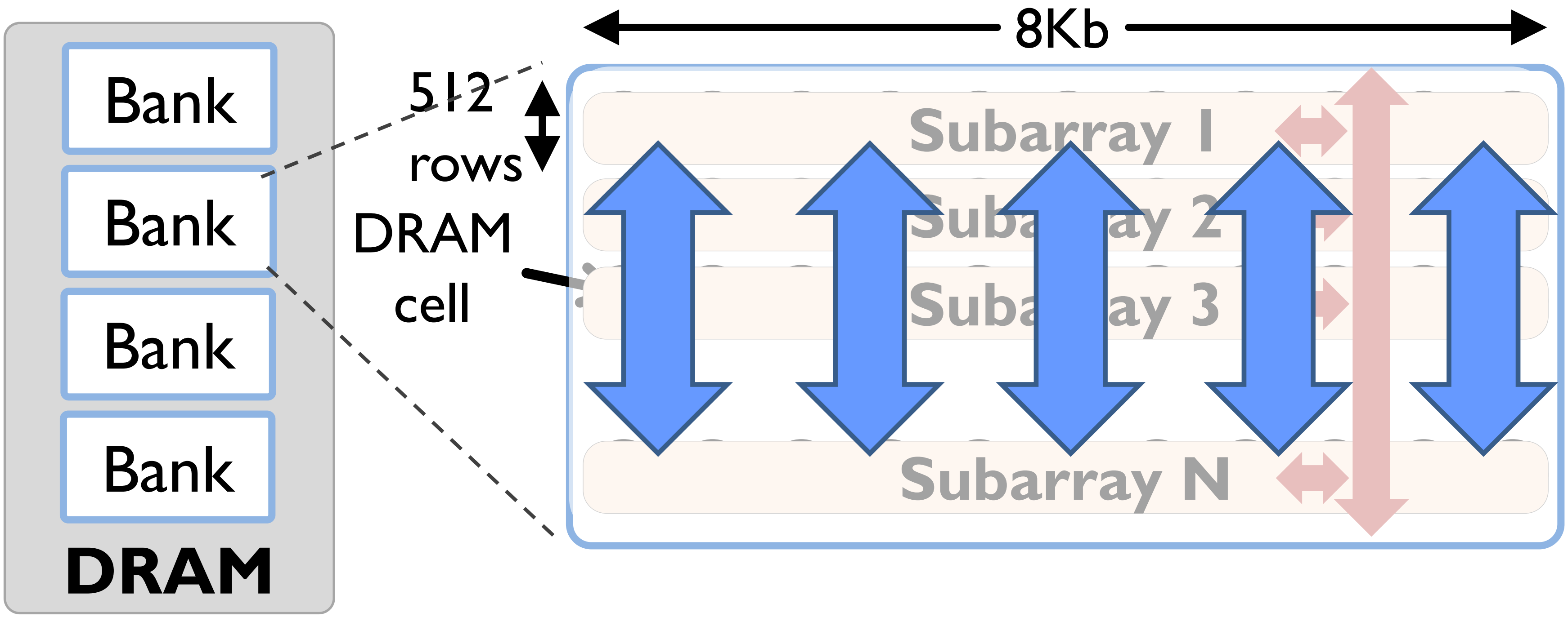
- Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,
  "Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM"
  *Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (**HPCA**)*,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

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

Kevin K. Chang[†], Prashant J. Nair[*], Donghyuk Lee[†], Saugata Ghose[†], Moinuddin K. Qureshi[*], and Onur Mutlu[†]

[†]*Carnegie Mellon University* [*]*Georgia Institute of Technology*

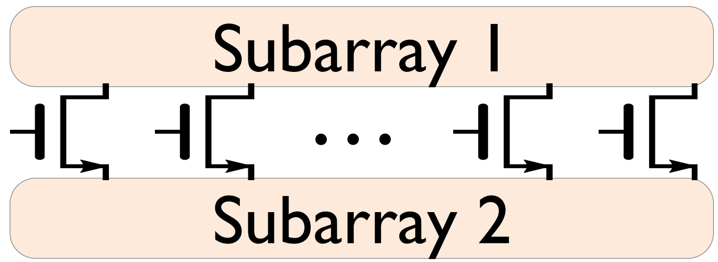
Moving Data Inside DRAM?



Goal: Provide a new substrate to enable wide connectivity between subarrays

Key Idea and Applications

- **Low-cost Inter-linked subarrays (LISA)**
  - Fast bulk data movement between subarrays
  - Wide datapath via isolation transistors: 0.8% DRAM chip area



- LISA is a **versatile substrate** → new applications

Fast bulk data copy: Copy latency 1.363ms→0.148ms (9.2x)
→ 66% speedup, -55% DRAM energy

In-DRAM caching: Hot data access latency 48.7ns→21.5ns (2.2x)
→ 5% speedup

Fast precharge: Precharge latency 13.1ns→5.0ns (2.6x)
→ 8% speedup

More on LISA

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

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

Kevin K. Chang†, Prashant J. Nair*, Donghyuk Lee†, Saugata Ghose†, Moinuddin K. Qureshi*, and Onur Mutlu†
†*Carnegie Mellon University* **Georgia Institute of Technology*

FIGARO: Fine-Grained In-DRAM Copy

- Yaohua Wang, Lois Orosa, Xiangjun Peng, Yang Guo, Saugata Ghose, Minesh Patel, Jeremie S. Kim, Juan Gómez Luna, Mohammad Sadrosadati, Nika Mansouri Ghiasi, and Onur Mutlu,
"FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching"
*Proceedings of the 53rd International Symposium on Microarchitecture (**MICRO**)*, Virtual, October 2020.

FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching

Yaohua Wang[⋆] Lois Orosa[†] Xiangjun Peng[⊙⋆] Yang Guo[⋆] Saugata Ghose[◇‡] Minesh Patel[†]
Jeremie S. Kim[†] Juan Gómez Luna[†] Mohammad Sadrosadati[§] Nika Mansouri Ghiasi[†] Onur Mutlu[†‡]

[⋆]*National University of Defense Technology* [†]*ETH Zürich* [⊙]*Chinese University of Hong Kong*
[◇]*University of Illinois at Urbana–Champaign* [‡]*Carnegie Mellon University* [§]*Institute of Research in Fundamental Sciences*

Network-On-Memory: Fast Inter-Bank Copy

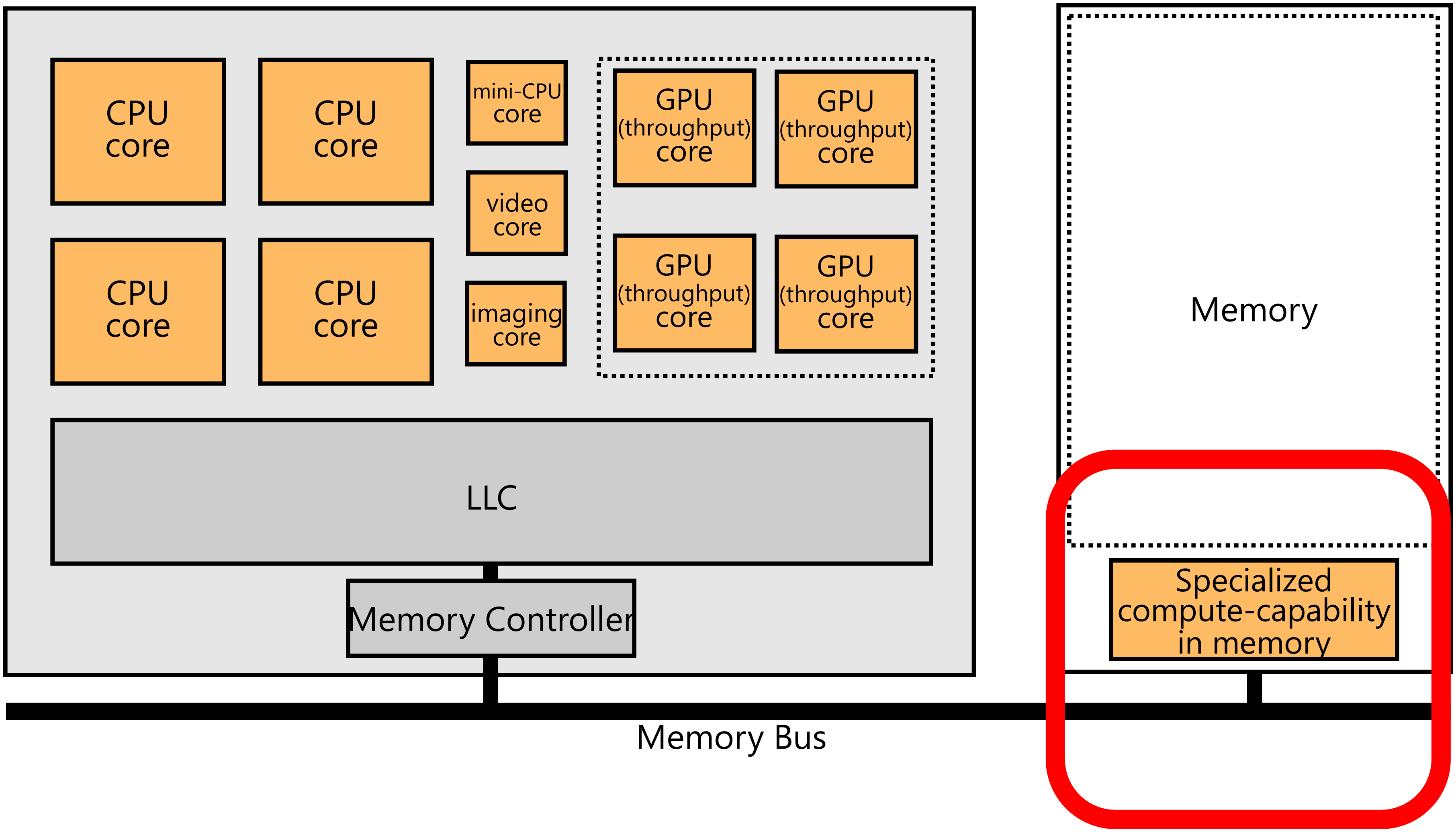
- Seyyed Hossein SeyyedAghaei Rezaei, Mehdi Modarressi, Rachata Ausavarungnirun, Mohammad Sadrosadati, Onur Mutlu, and Masoud Daneshtalab,
  "NoM: Network-on-Memory for Inter-Bank Data Transfer in Highly-Banked Memories"
  IEEE Computer Architecture Letters (***CAL***), to appear in 2020.

NoM: Network-on-Memory for Inter-bank Data Transfer in Highly-banked Memories

Seyyed Hossein SeyyedAghaei Rezaei[1] Mehdi Modarressi[1,3] Rachata Ausavarungnirun[2]
Mohammad Sadrosadati[3] Onur Mutlu[4] Masoud Daneshtalab[5]

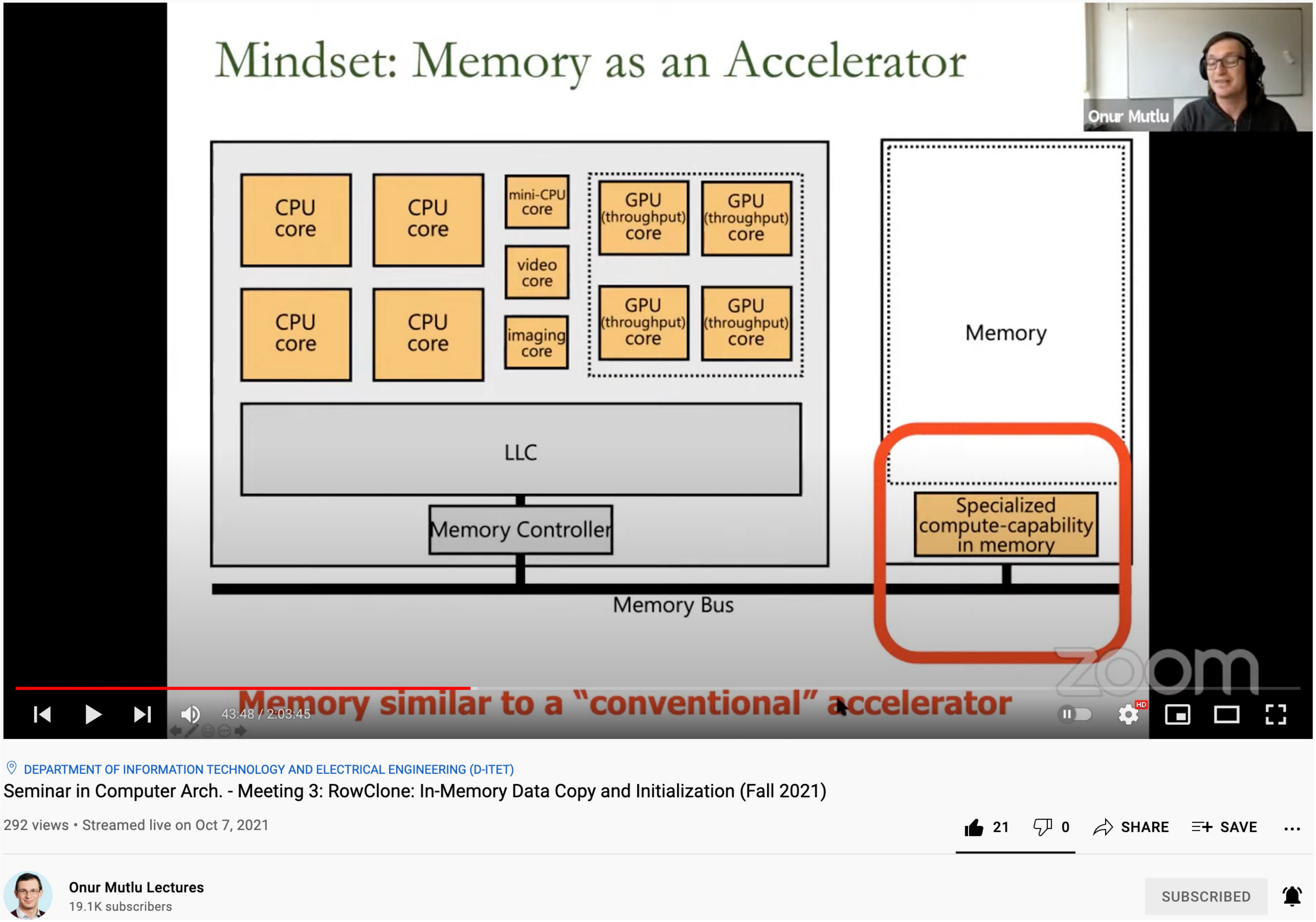
[1]University of Tehran [2]King Mongkut's University of Technology North Bangkok [3]Institute for Research in Fundamental Sciences
[4]ETH Zürich [5]Mälardalens University

Mindset: Memory as an Accelerator



Memory similar to a "conventional" accelerator

Lecture on RowClone & Processing using DRAM



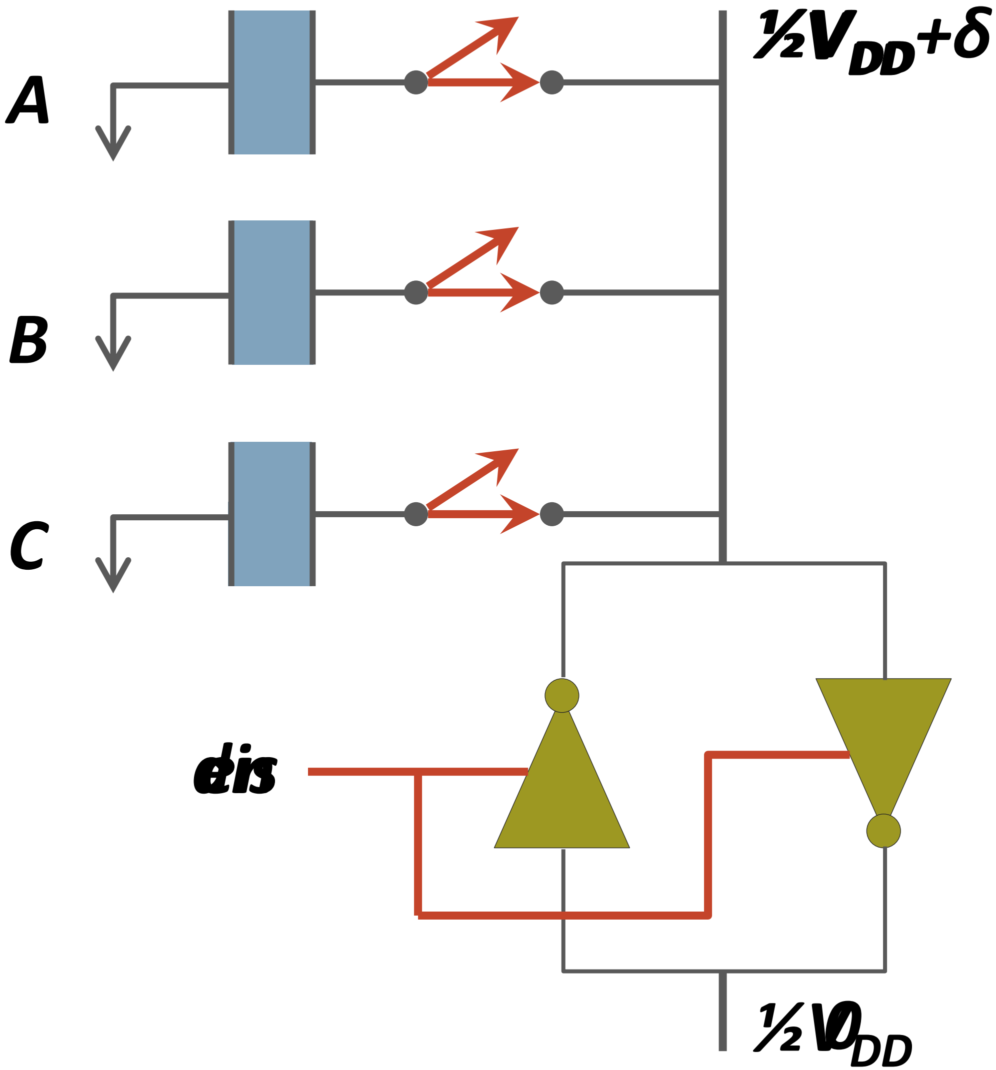
https://www.youtube.com/watch?v=n6Pwg1qax_E&list=PL5Q2soXY2Zi_7UBNmC9B8Yr5JSwTG9yH4&index=4

(Truly) In-Memory Computation

- We can support in-DRAM AND, OR, NOT, MAJ
- At low cost
- Using analog computation capability of DRAM
  - Idea: activating multiple rows performs computation
- 30-60X performance and energy improvement
  - Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," MICRO 2017.

- New memory technologies enable even more opportunities
  - Memristors, resistive RAM, phase change mem, STT-MRAM, ...
  - Can operate on data with minimal movement

In-DRAM AND/OR: Triple Row Activation



Final State

AB + BC + AC

C(A + B) + ~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

In-DRAM NOT: Dual Contact Cell

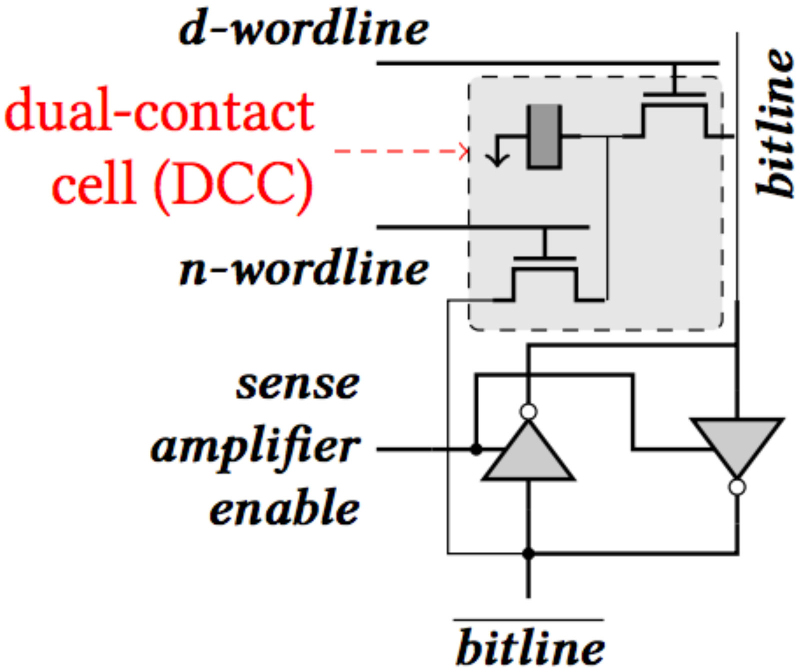


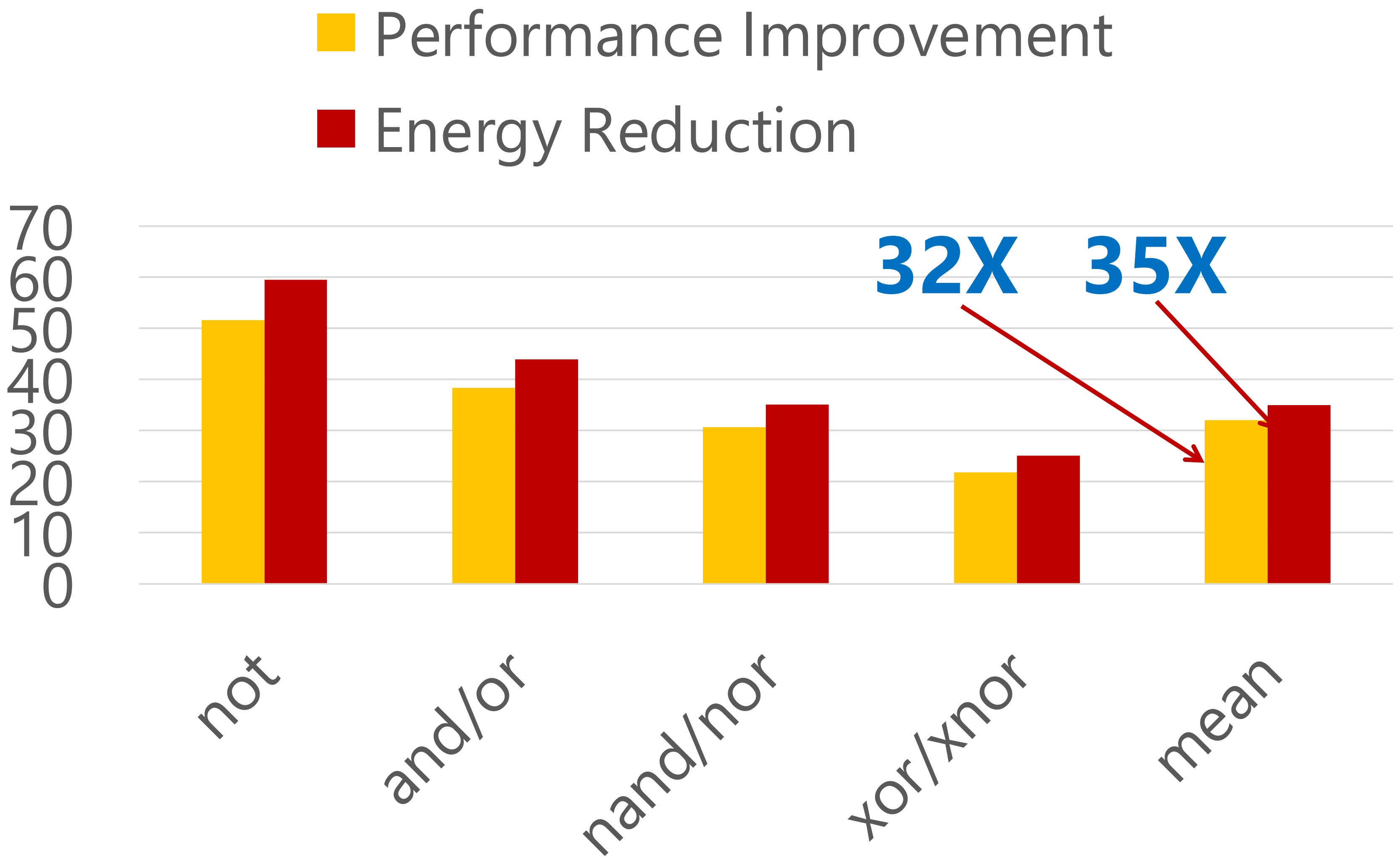
Figure 5: A dual-contact cell connected to both ends of a sense amplifier

Idea:

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

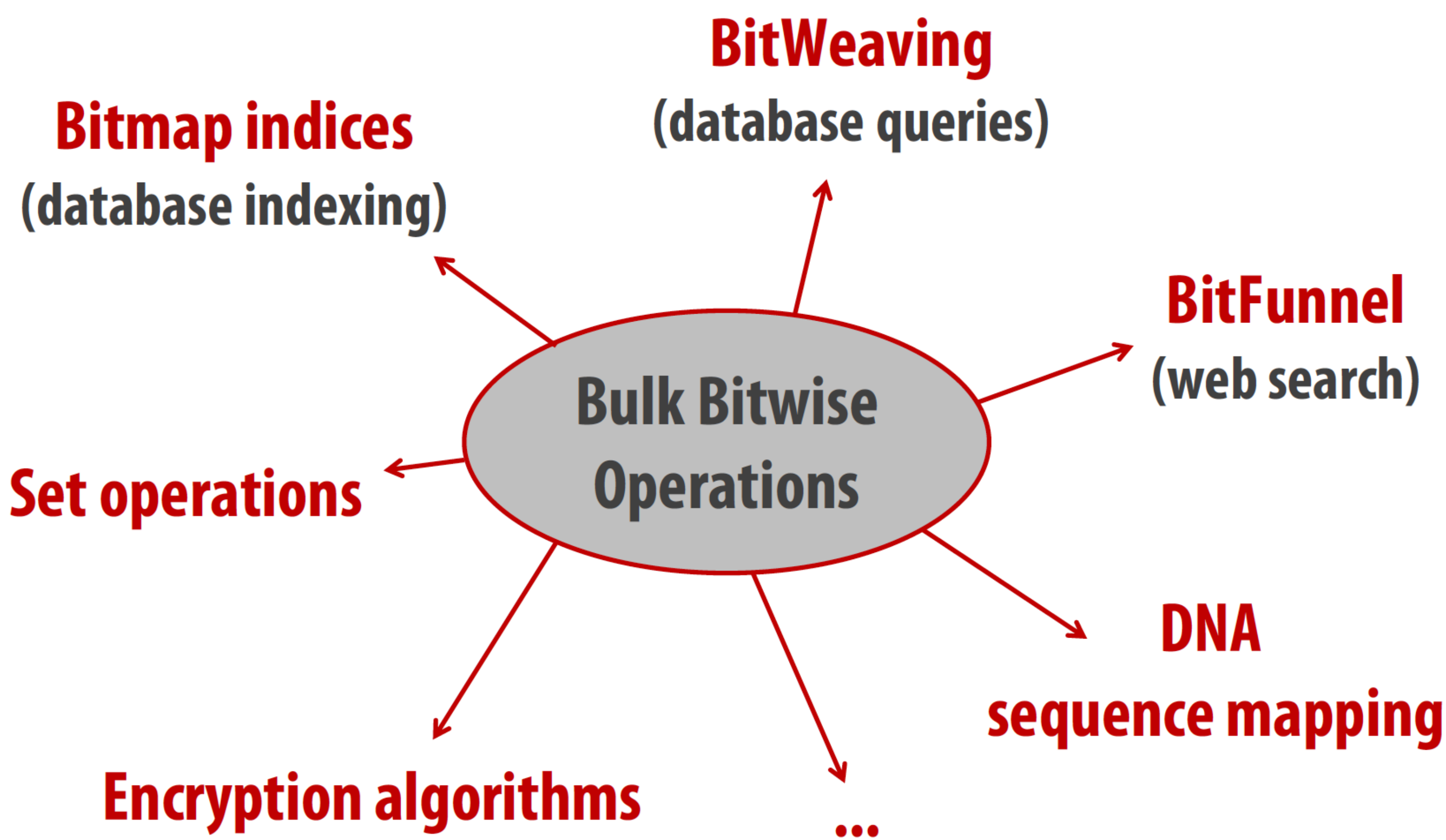
Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

Ambit vs. DDR3: Performance and Energy



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

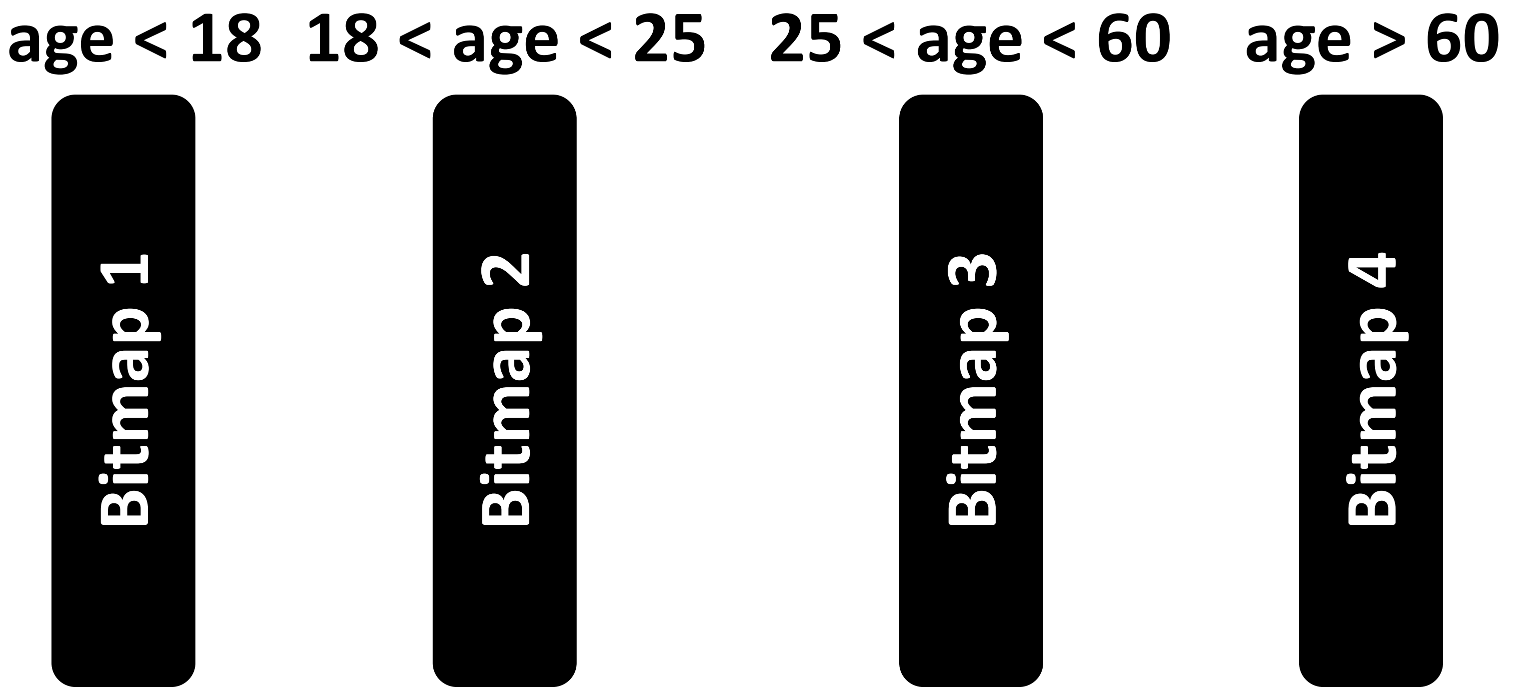
Bulk Bitwise Operations in Workloads



[1] Li and Patel, BitWeaving, SIGMOD 2013
[2] Goodwin+, BitFunnel, SIGIR 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**



Performance: Bitmap Index on Ambit

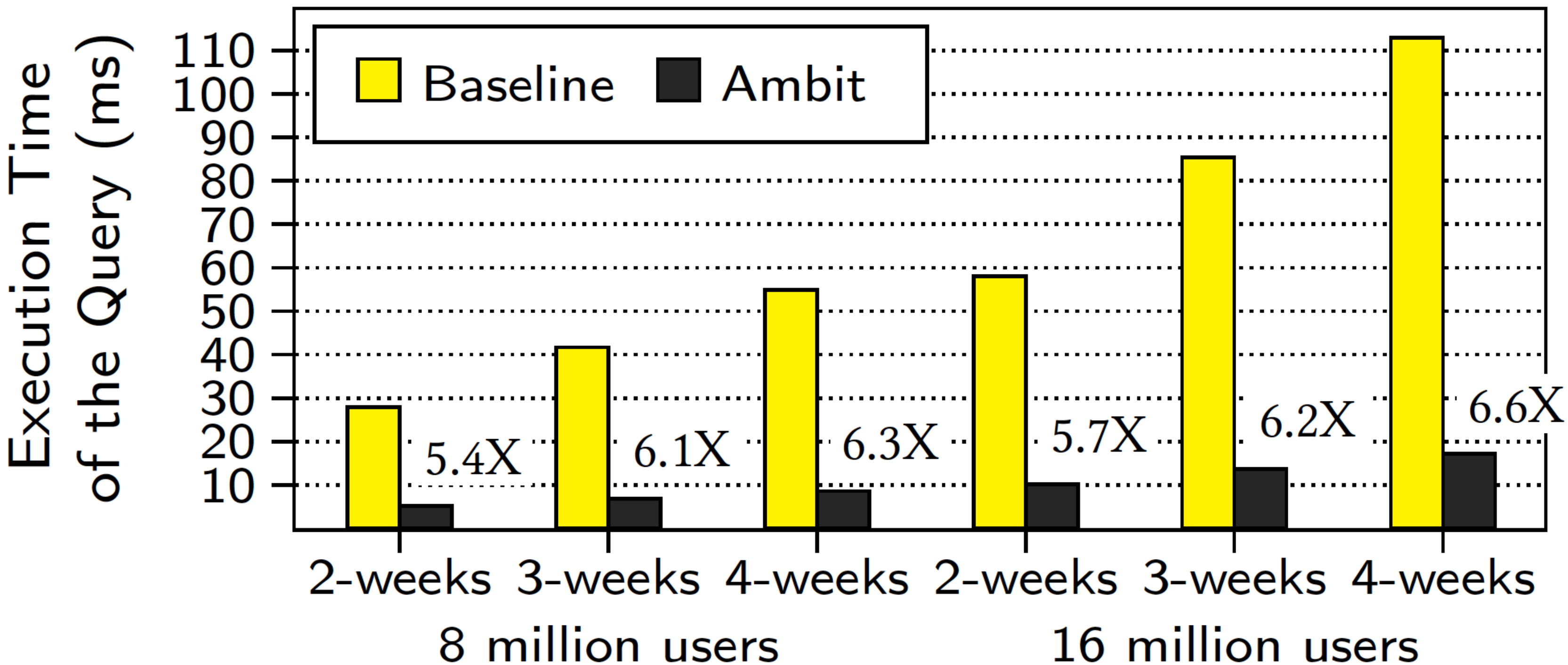


Figure 10: Bitmap index performance. The value above each bar indicates the reduction in execution time due to Ambit.

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

In-DRAM Acceleration of Database Queries

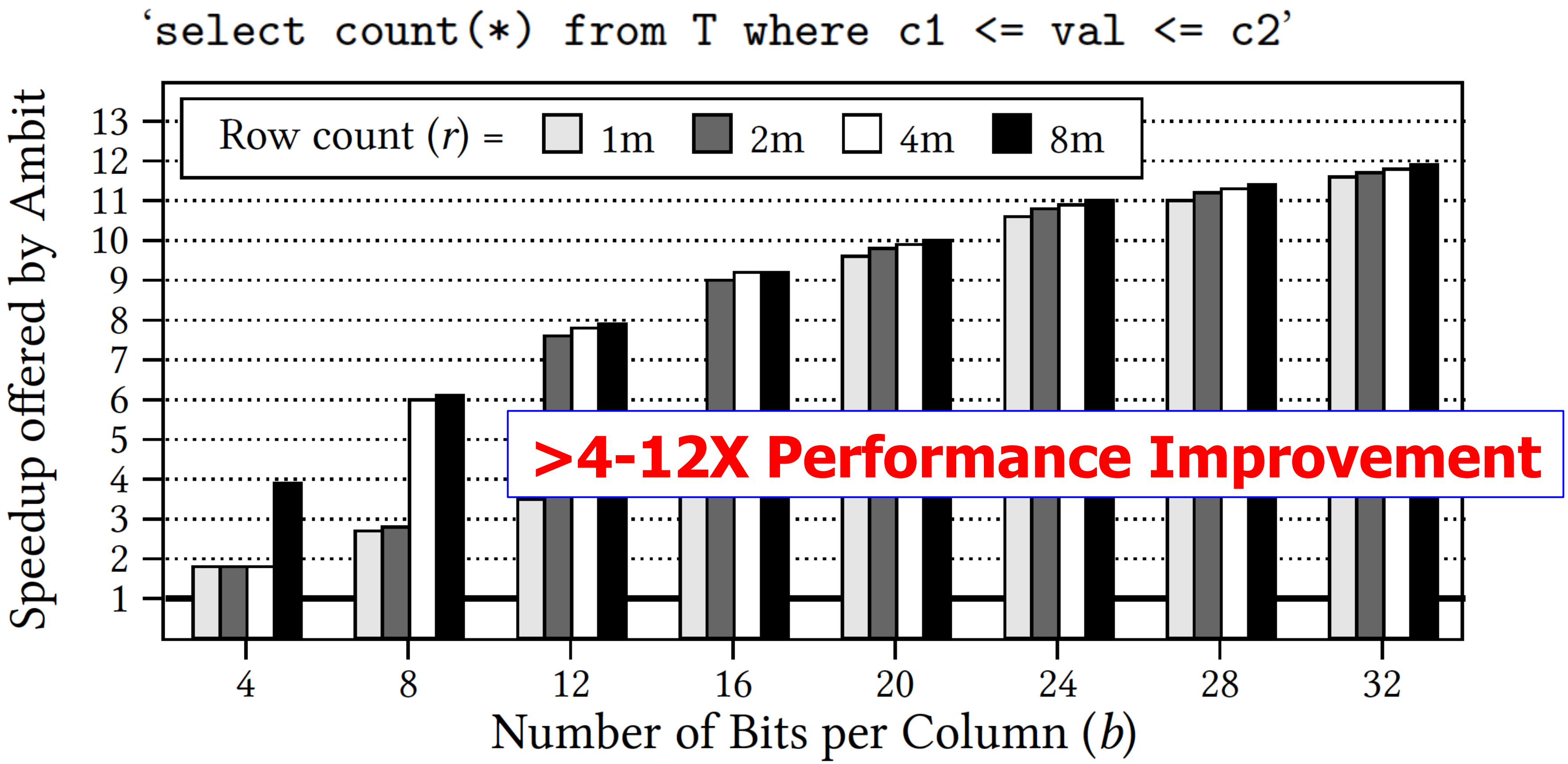


Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

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

More on In-DRAM Bulk AND/OR

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

Fast Bulk Bitwise AND and OR in DRAM

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

*Carnegie Mellon University †Intel Pittsburgh

More on Ambit

- Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
  "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology"
  *Proceedings of the 50th International Symposium on Microarchitecture (**MICRO**)*, Boston, MA, USA, October 2017.
  [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]

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

In-DRAM Bulk Bitwise Execution

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

In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
visesha@microsoft.com

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

SIMDRAM Framework

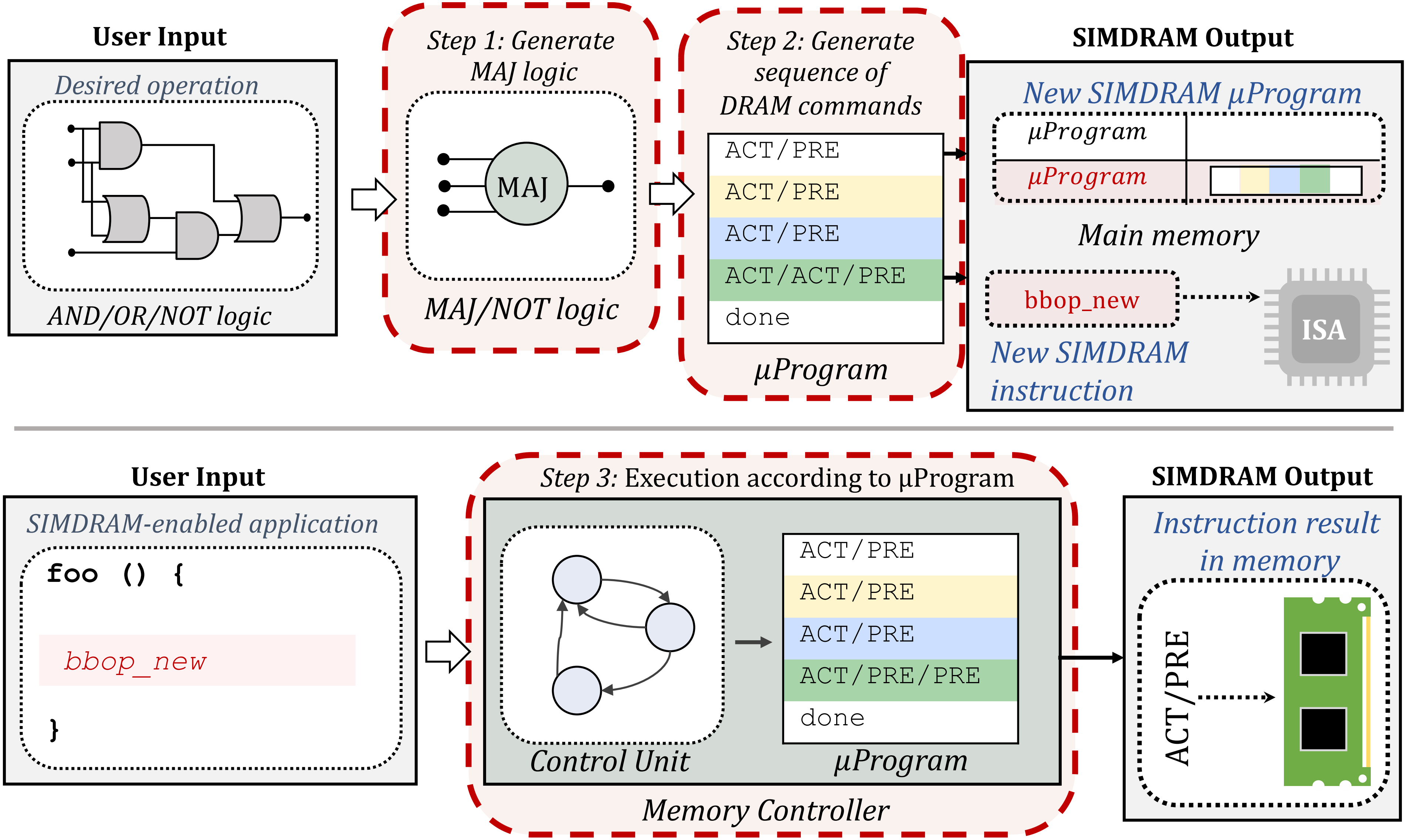
- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu,
  "SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"
  *Proceedings of the 26th International Conference 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] *Geraldo F. Oliveira[1] Sven Gregorio[1] João Dinis Ferreira[1]
Nika Mansouri Ghiasi[1] Minesh Patel[1] Mohammed Alser[1] Saugata Ghose[3]
Juan Gómez-Luna[1] Onur Mutlu[1]

[1]ETH Zürich [2]Simon Fraser University [3]University of Illinois at Urbana–Champaign

SIMDRAM Framework: Overview



SIMDRAM Key Results

Evaluated on:

- 16 complex in-DRAM operations
- 7 commonly-used real-world applications

SIMDRAM provides:

- **88×** and **5.8×** the **throughput** of a **CPU** and a **high-end GPU**, respectively, over **16 operations**
- **257×** and **31×** the **energy efficiency** of a **CPU** and a **high-end GPU**, respectively, over **16 operations**
- **21×** and **2.1×** the **performance** of a **CPU** an a **high-end GPU**, over **seven real-world applications**

More on SIMDRAM

- 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] *Geraldo F. Oliveira[1] Sven Gregorio[1] João Dinis Ferreira[1]
Nika Mansouri Ghiasi[1] Minesh Patel[1] Mohammed Alser[1] Saugata Ghose[3]
Juan Gómez-Luna[1] Onur Mutlu[1]

[1]ETH Zürich [2]Simon Fraser University [3]University of Illinois at Urbana–Champaign

We Covered Until Here in Lecture 2

Memory Systems and Memory-Centric Computing

Lecture 2: Memory-Centric Computing I

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu

16 July 2024

HiPEAC ACACES Summer School 2024

SAFARI

ETH zürich

Backup Slides
(To Be Covered in Lecture 3)

MIMDRAM: More Flexible Processing using DRAM

- **Appears at HPCA 2024** https://arxiv.org/pdf/2402.19080.pdf

MIMDRAM: An End-to-End Processing-Using-DRAM System for High-Throughput, Energy-Efficient and Programmer-Transparent Multiple-Instruction Multiple-Data Computing

Geraldo F. Oliveira[†] Ataberk Olgun[†] Abdullah Giray Yağlıkçı[†] F. Nisa Bostancı[†]
Juan Gómez-Luna[†] Saugata Ghose[‡] Onur Mutlu[†]

[†] *ETH Zürich* [‡] *Univ. of Illinois Urbana-Champaign*

*Our **goal** is to design a flexible PUD system that overcomes the limitations caused by the large and rigid granularity of PUD. To this end, we propose MIMDRAM, a hardware/software co-designed PUD system that introduces new mechanisms to allocate and control only the necessary resources for a given PUD operation. The key idea of MIMDRAM is to leverage fine-grained DRAM (i.e., the ability to independently access smaller segments of a large DRAM row) for PUD computation. MIMDRAM exploits this key idea to enable a multiple-instruction multiple-data (MIMD) execution model in each DRAM subarray (and SIMD execution within each DRAM row segment).*

MIMDRAM: Executive Summary

Problem: Processing-Using-DRAM (PUD) suffers from three issues caused by DRAM's large and rigid access granularity
- Underutilization due to data parallelism variation in (and across) applications
- Limited computation support due to a lack of interconnects
- Challenging programming model due to a lack of compilers

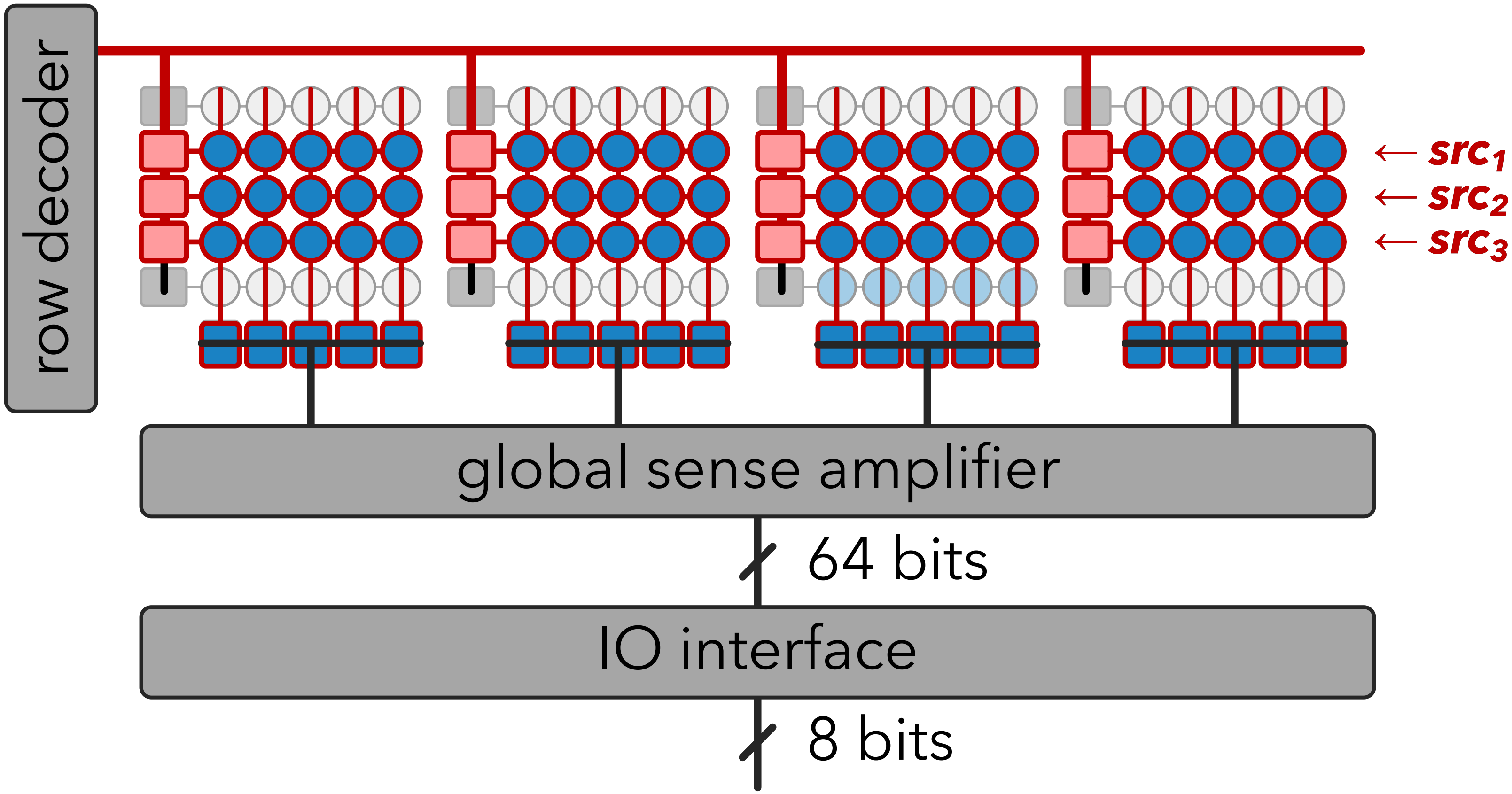
Goal: Design a flexible PUD system that overcomes the three limitations caused by DRAM's large and rigid access granularity

Key Mechanism: MIMDRAM, a hardware/software co-design PUD system
- **Key idea**: leverage fine-grained DRAM for PUD operation
- **HW**: - simple changes to the DRAM array, enabling concurrent PUD operations
  - low-cost interconnects at the DRAM peripherals for data reduction
- **SW**: - compiler and OS support to generate and map PUD instructions

Key Results: MIMDRAM achieves
- **14.3x**, **30.6x**, and **6.8x** the energy efficiency of state-of-the-art PUD systems, a high-end CPU and GPU, respectively
- Small area cost to a DRAM chip (**1.11%**) and CPU die (**0.6%**)

Background:
In-DRAM Copy/Init, Majority & NOT Operations

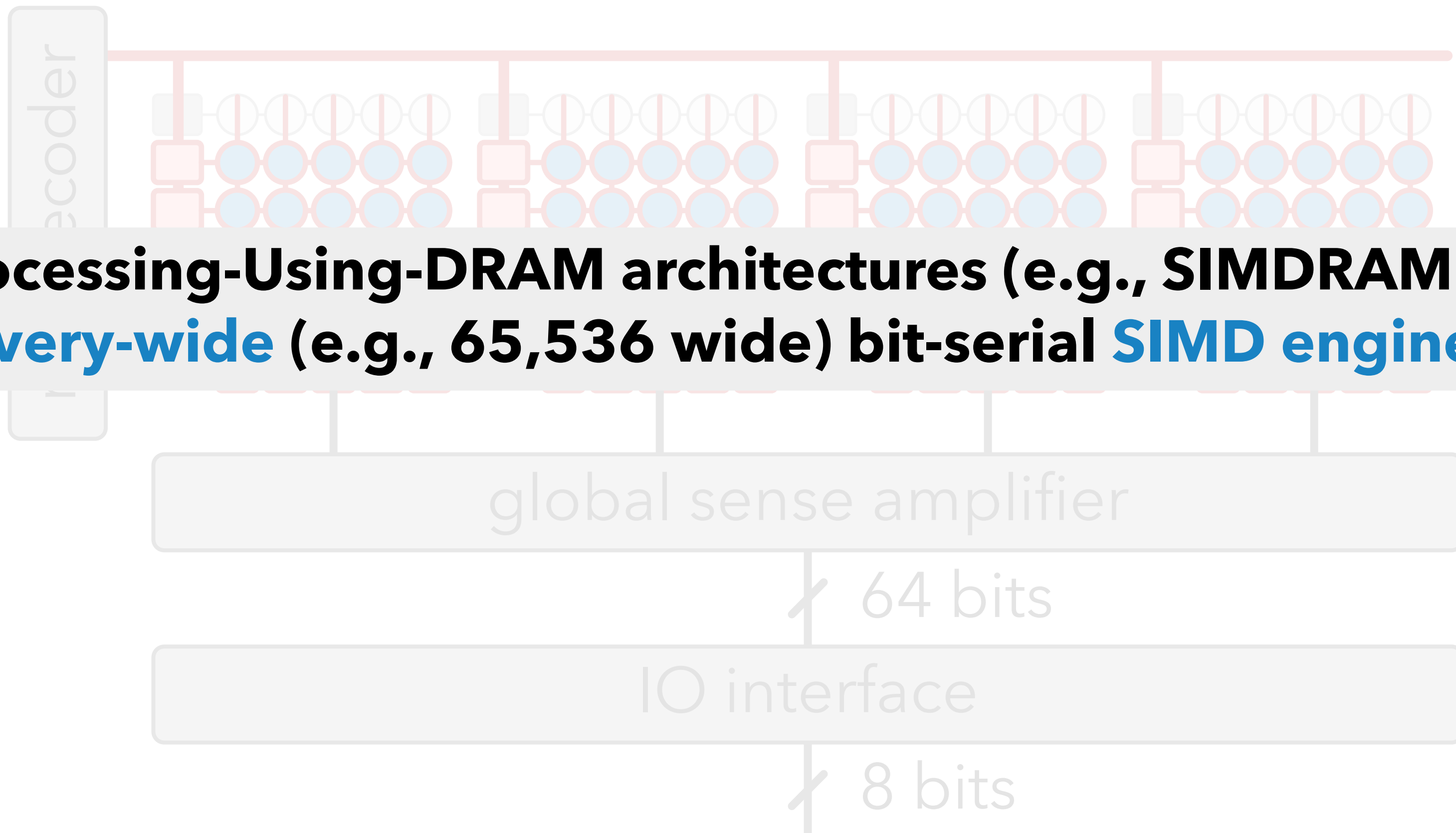
In-DRAM majority is performed by simultaneously activating three DRAM rows



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

Background:
In-DRAM Majority Operations

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



Processing-Using-DRAM architectures (e.g., SIMDRAM) are very-wide (e.g., 65,536 wide) bit-serial SIMD engines

Oliveira, Geraldo F., et al. " **SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM**," in ASPLOS, 2021

Limitations of PUD Systems: Overview

PUD systems suffer from three sources of inefficiency due to the large and rigid DRAM access granularity

1 **SIMD Underutilization**
- due to data parallelism variation within and across applications
- leads to throughput and energy waste

2 **Limited Computation Support**
- due to a lack of low-cost interconnects across columns
- limits PUD operations to only parallel map constructs

3 **Challenging Programming Model**
- due to a lack of compiler support for PUD systems
- creates a burden on programmers, limiting PUD adoption

Limitations of PUD Systems:
Challenging Programming Model

Programmer's Tasks:

Goal:

Map & align data structures

Just write my kernel

High-level code for

C[i] = (A[i] > pred[i])? A[i] + B[i] : A[i] - B[i]

```
for (int i = 0; i < size ; ++ i){
    bool cond = A[i] > pred[i];
    if (cond) C[i] = A[i] + B[i];
    else  C[i] = A[i] - B[i];
}
```

Limitations of PUD Systems:
Challenging Programming Model

Programmer's Tasks:

Map & align data structures

Identify array boundaries

Goal:

Just write my kernel

High-level code for

C[i] = (A[i] > pred[i])? A[i] + B[i] : A[i] - B[i]

```
for (int i = 0; i < size ; ++ i){
   bool cond = A[i] > pred[i];
   if (cond) C[i] = A[i] + B[i];
   else  C[i] = A[i] - B[i];
}
```

Limitations of PUD Systems: Challenging Programming Model

Programmer's Tasks:

Map & align data structures	Identify array boundaries	Manually unroll loop	Map C to PUD instructions

Goal:

Just write my kernel

High-level code for

C[i] = (A[i] > pred[i])? A[i] + B[i] : A[i] - B[i]

```
for (int i = 0; i < size ; ++ i){
    bool cond = A[i] > pred[i];
    if (cond) C[i] = A[i] + B[i];
    else  C[i] = A[i] - B[i];
}
```

Limitations of PUD Systems:
Challenging Programming Model

Programmer's Tasks:

Goal:

Map & align data structures	Identify array boundaries	Manually unroll loop	Map C to PUD instructions	Orchestrate data movement	Just write my kernel

High-level code for

C[i] = (A[i] > pred[i])? A[i] + B[i] : A[i] - B[i]

```
for (int i = 0; i < size ; ++ i){
   bool cond = A[i] > pred[i];
   if (cond) C[i] = A[i] + B[i];
   else  C[i] = A[i] - B[i];
}
```

Limitations of PUD Systems:
Challenging Programming Model

Programmer's Tasks: | **Goal:**

Map & align data structures	Identify array boundaries	Manually unroll loop	Map C to PUD instructions	Orchestrate data movement	Just write my kernel

PUD's assembly-like code for

C[i] = (A[i] > pred[i])? A[i] + B[i] : A[i] – B[i]

```
bbop_trsp_init(A , size , elm_size);
bbop_trsp_init(B , size , elm_size);
bbop_trsp_init(C , size , elm_size);

bbop_add(D , A , B , size , elm_size);
bbop_sub(E , A , B , size , elm_size);
bbop_greater(F , A , pred , size , elm_size);
bbop_if_else(C , D , E , F , size , elm_size);
```

Problem & Goal

Problem

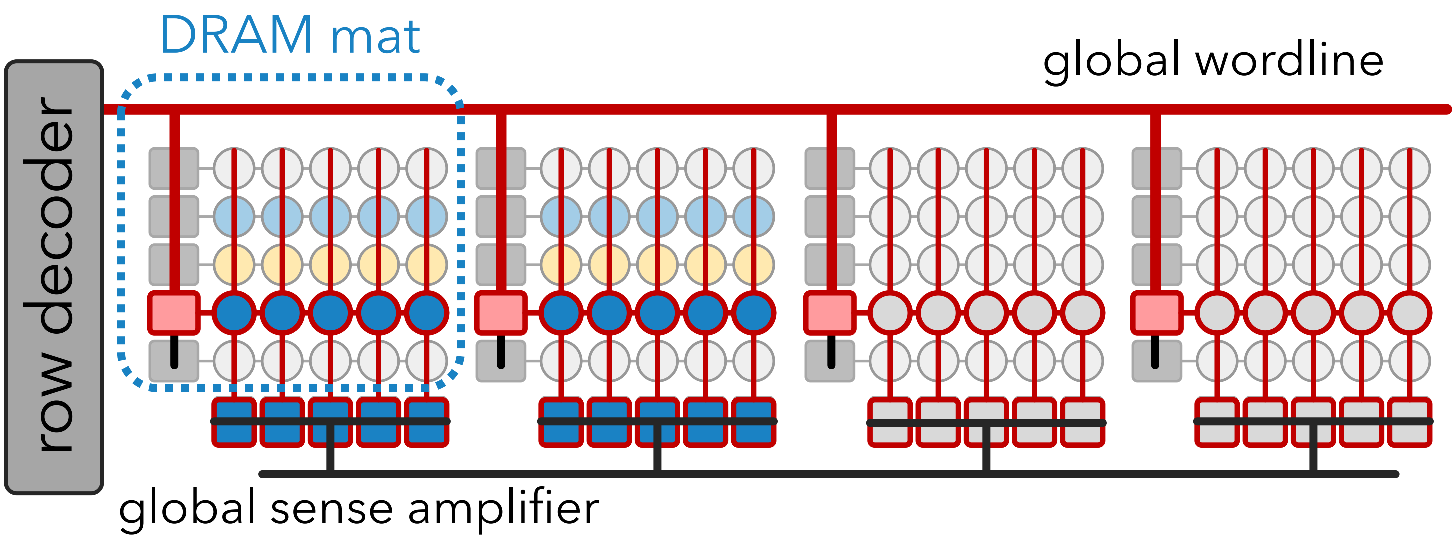
Processing-Using-DRAM's large and rigid granularity limits its applicability and efficiency for different applications

Goal

Design a flexible PUD system that overcomes the three limitations caused by large and rigid DRAM access granularity

MIMDRAM:
Key Idea (I)

DRAM's hierarchical organization can enable <u>fine-grained access</u>

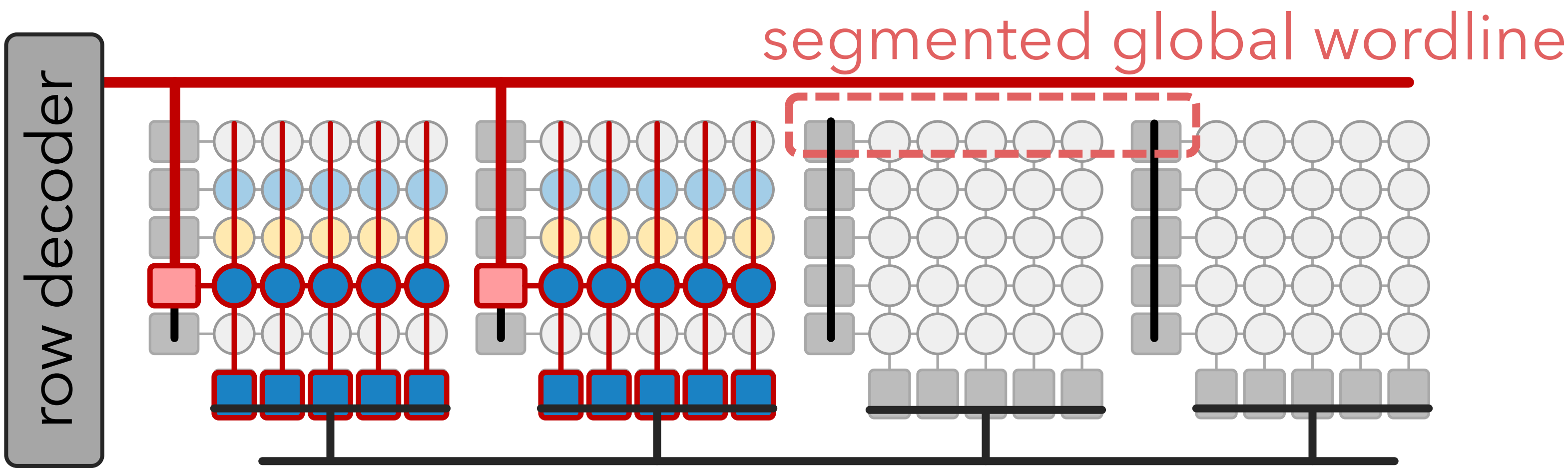


Key Issue:
on a DRAM access, the global wordline propagates across all DRAM mats

Fine-Grained DRAM:
segments the global wordline to access individual DRAM mats

MIMDRAM:
Key Idea (II)

Fine-Grained DRAM:
segments the global wordline to access individual DRAM mats



Fine-grained DRAM for energy-efficient DRAM access:

[Cooper-Balis+, 2010]: Fine-Grained Activation for Power Reduction in DRAM
[Udipi+, 2010]: Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores
[Zhang+, 2014]: Half-DRAM
[Ha+, 2016]: Improving Energy Efficiency of DRAM by Exploiting Half Page Row Access
[O'Connor+, 2017]: Fine-Grained DRAM
[Olgun+, 2024]: Sectored DRAM

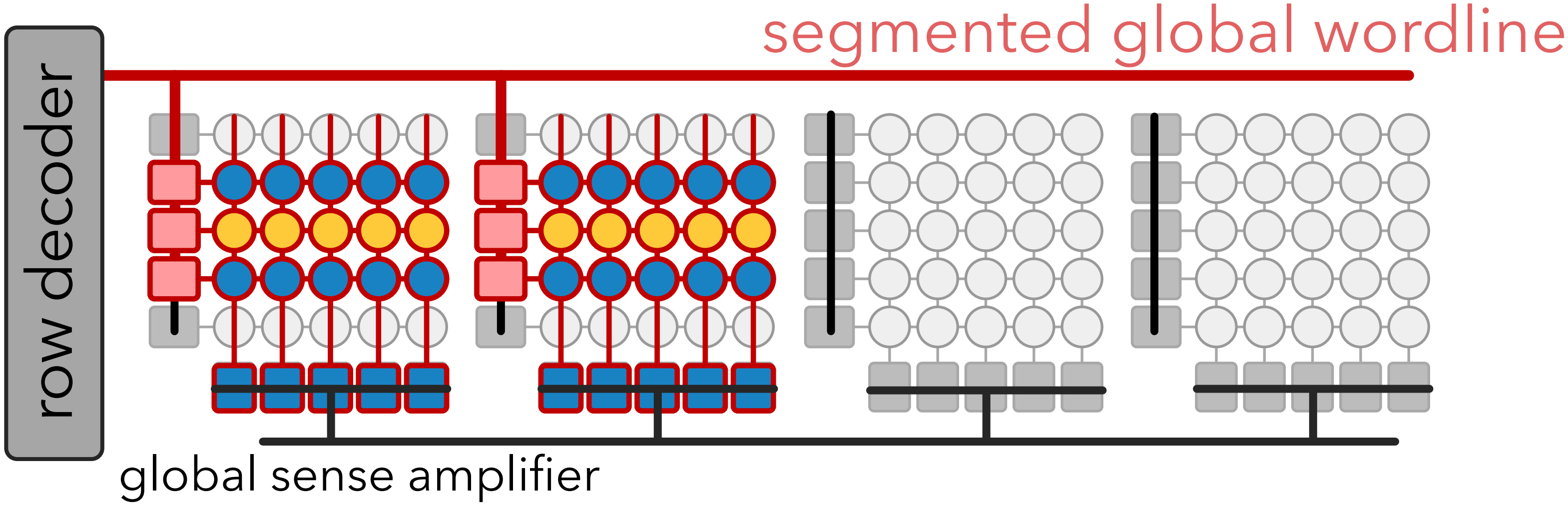
Sectored DRAM

- Ataberk Olgun, F. Nisa Bostanci, Geraldo F. Oliveira, Yahya Can Tugrul, Rahul Bera, A. Giray Yaglikci, Hasan Hassan, Oguz Ergin, and Onur Mutlu,
"Sectored DRAM: A Practical Energy-Efficient and High-Performance Fine-Grained DRAM Architecture"
ACM Transactions on Architecture and Code Optimization (***TACO***),
[online] June 2024.
[arXiv version]
[ACM Digital Library version]

Sectored DRAM: A Practical Energy-Efficient and High-Performance Fine-Grained DRAM Architecture

Ataberk Olgun[§] F. Nisa Bostancı[§†] Geraldo F. Oliveira[§] Yahya Can Tuğrul[§†] Rahul Bera[§]
A. Giray Yağlıkcı[§] Hasan Hassan[§] Oğuz Ergin[†] Onur Mutlu[§]

MIMDRAM:
Key Idea (III)

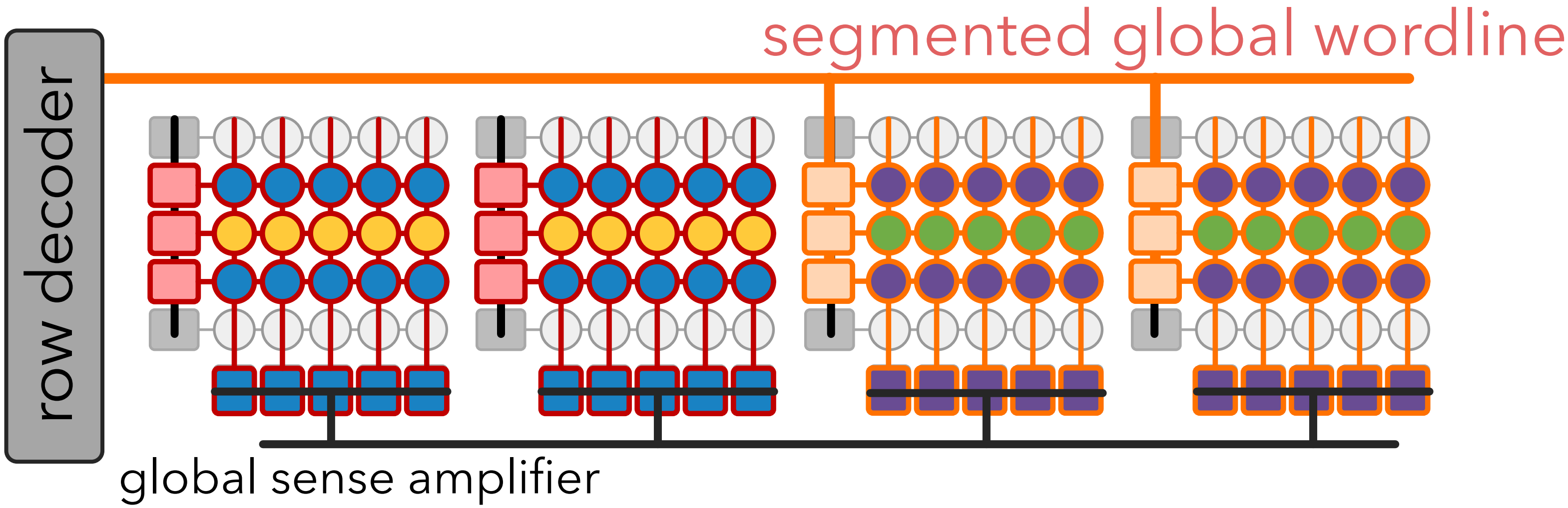


Fine-grained DRAM for processing-using-DRAM:

1 **Improves SIMD utilization**

- for a single PUD operation, only access the DRAM mats with target data

MIMDRAM:
Key Idea (III)

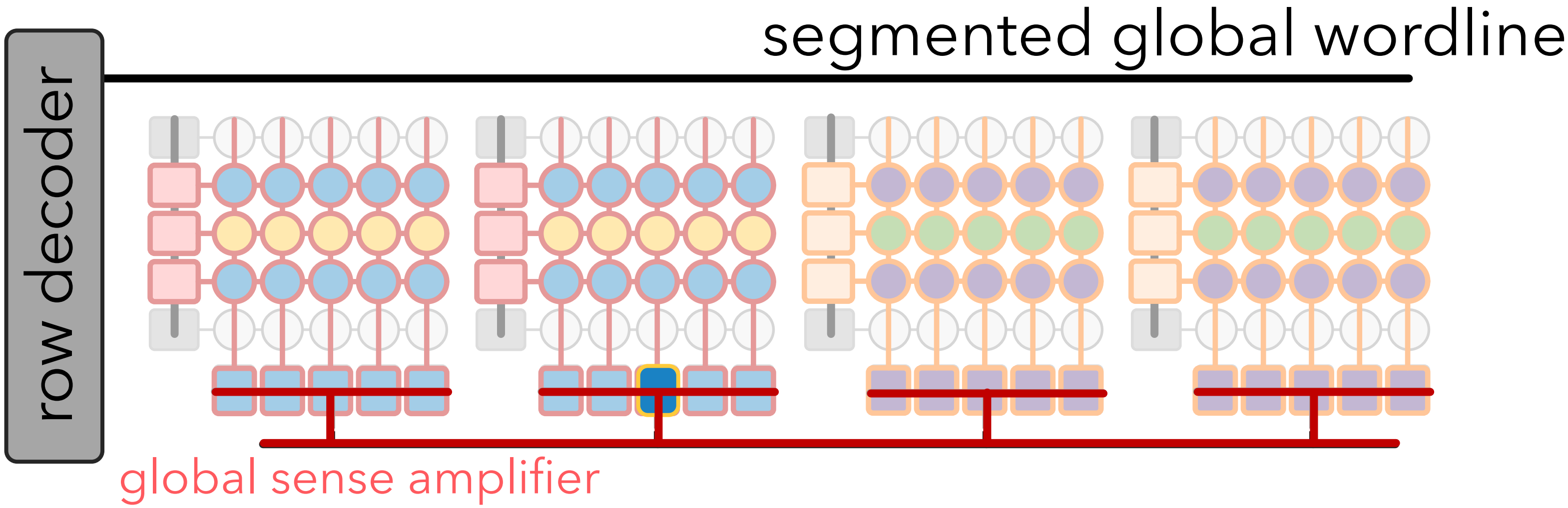


Fine-grained DRAM for processing-using-DRAM:

1 Improves SIMD utilization

- for a single PUD operation, only access the DRAM mats with target data
- for multiple PUD operations, execute independent operations concurrently
  → **multiple instruction, multiple data (MIMD) execution model**

MIMDRAM:
Key Idea (III)

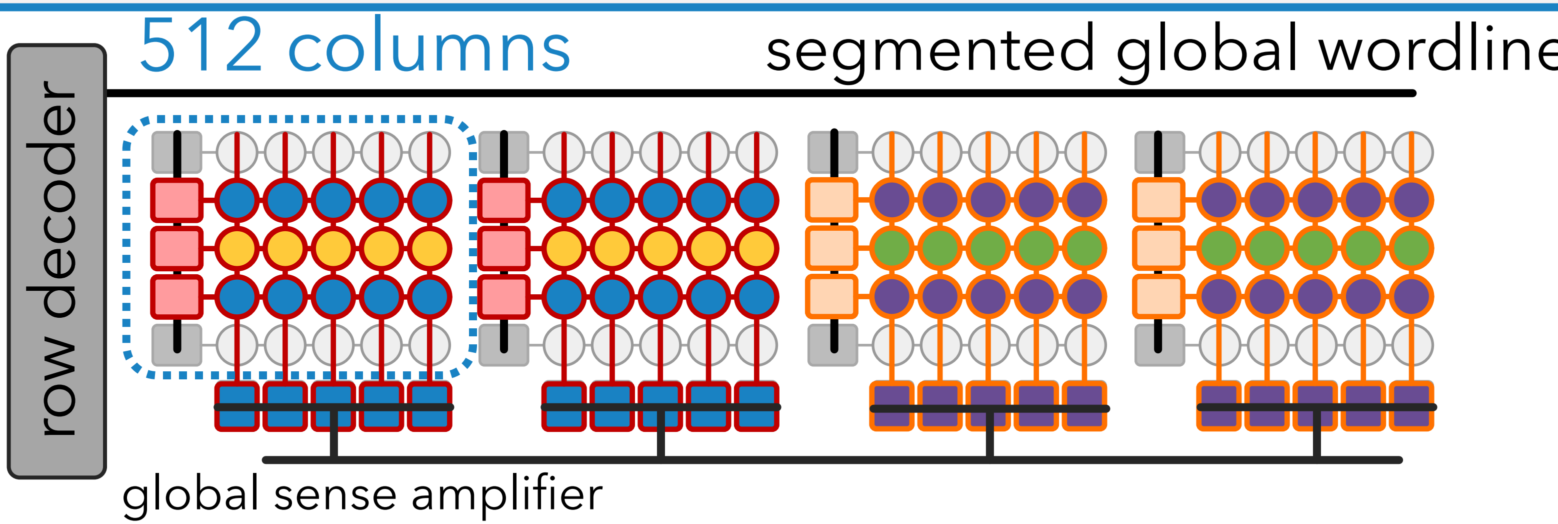


Fine-grained DRAM for processing-using-DRAM:

1 **Improves SIMD utilization**
- for a single PUD operation, only access the DRAM mats with target data
- for multiple PUD operations, execute independent operations concurrently
→ **multiple instruction, multiple data (MIMD) execution model**

2 **Enables low-cost interconnects for vector reduction**
- global and local data buses can be used for inter-/intra-mat communication

MIMDRAM:
Key Idea (III)



Fine-grained DRAM for processing-using-DRAM:

1 Improves SIMD utilization

- for a single PUD operation, only access the DRAM mats with target data
- for multiple PUD operations, execute independent operations concurrently
  → **multiple instruction, multiple data (MIMD) execution model**

2 Enables low-cost interconnects for vector reduction

- global and local data buses can be used for inter-/intra-mat communication

3 Eases programmability

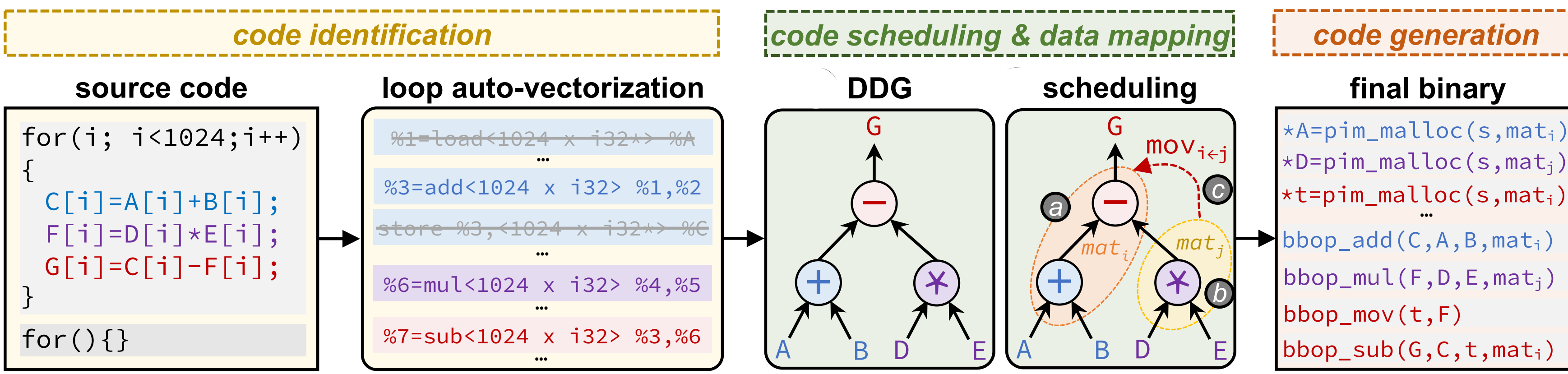
- SIMD parallelism in a DRAM mat is on par with vector ISAs' SIMD width

MIMDRAM:
Compiler Support (I)

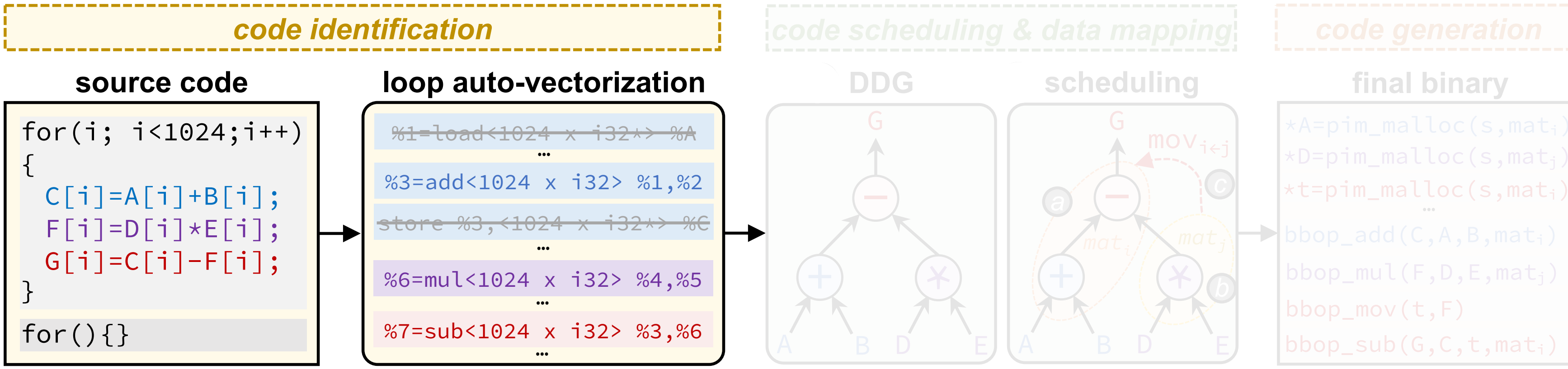
Goal

Transparently:
extract SIMD parallelism from an application, and
schedule PUD instructions while maximizing utilization

Three new LLVM-based passes targeting PUD execution



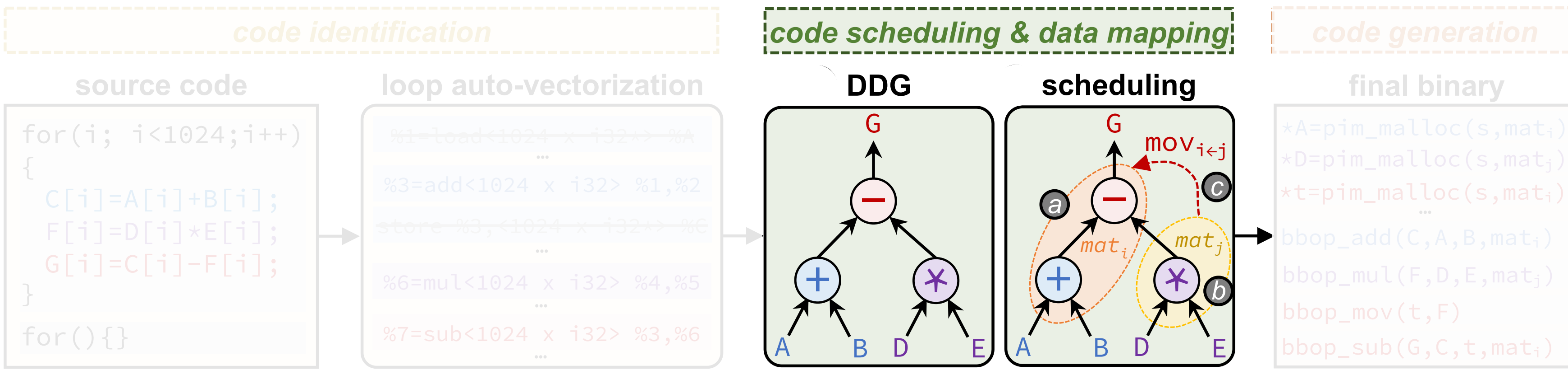
MIMDRAM:
Compiler Support (II)



Goal

Identify SIMD parallelism, generate PUD instructions, and set the appropriate vectorization factor

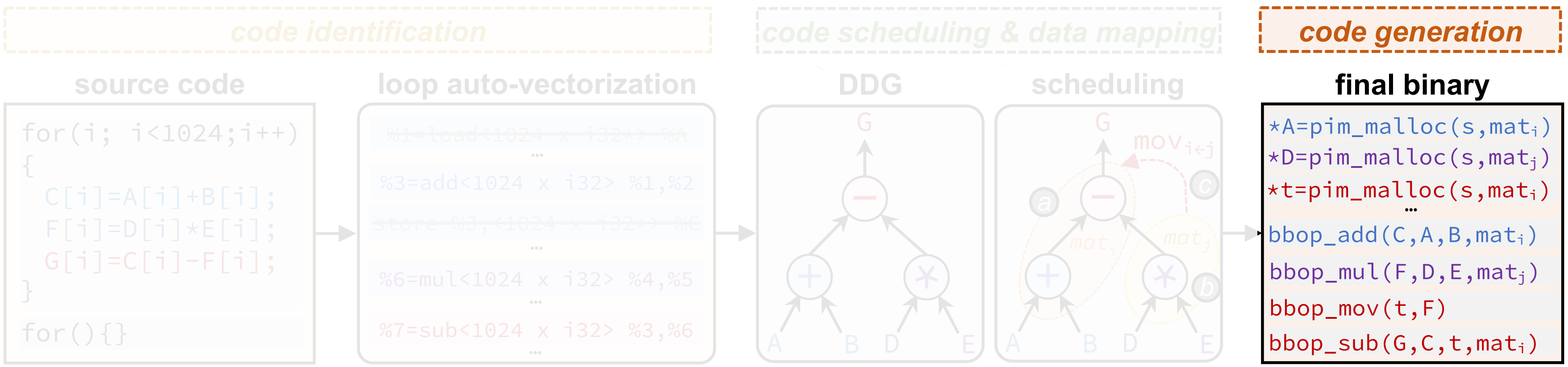
MIMDRAM:
Compiler Support (II)



Goal: Identify SIMD parallelism, generate PUD instructions, and set the appropriate vectorization factor

Goal: **Improve SIMD utilization by allowing the distribution of independent PUD instructions across DRAM mats**

MIMDRAM:
Compiler Support (III)



Goal: Identify SIMD parallelism, generate PUD instructions, and set the appropriate vectorization factor

Goal: Improve SIMD utilization by allowing the distribution of independent PUD instructions across DRAM mats

Goal: **Generate the appropriate binary for data allocation and PUD instructions**

MIMDRAM:
System Support

- Instruction set architecture
- Execution & data transposition
- Data coherence
- Address translation
- Data allocation & alignment
- Mat label translation

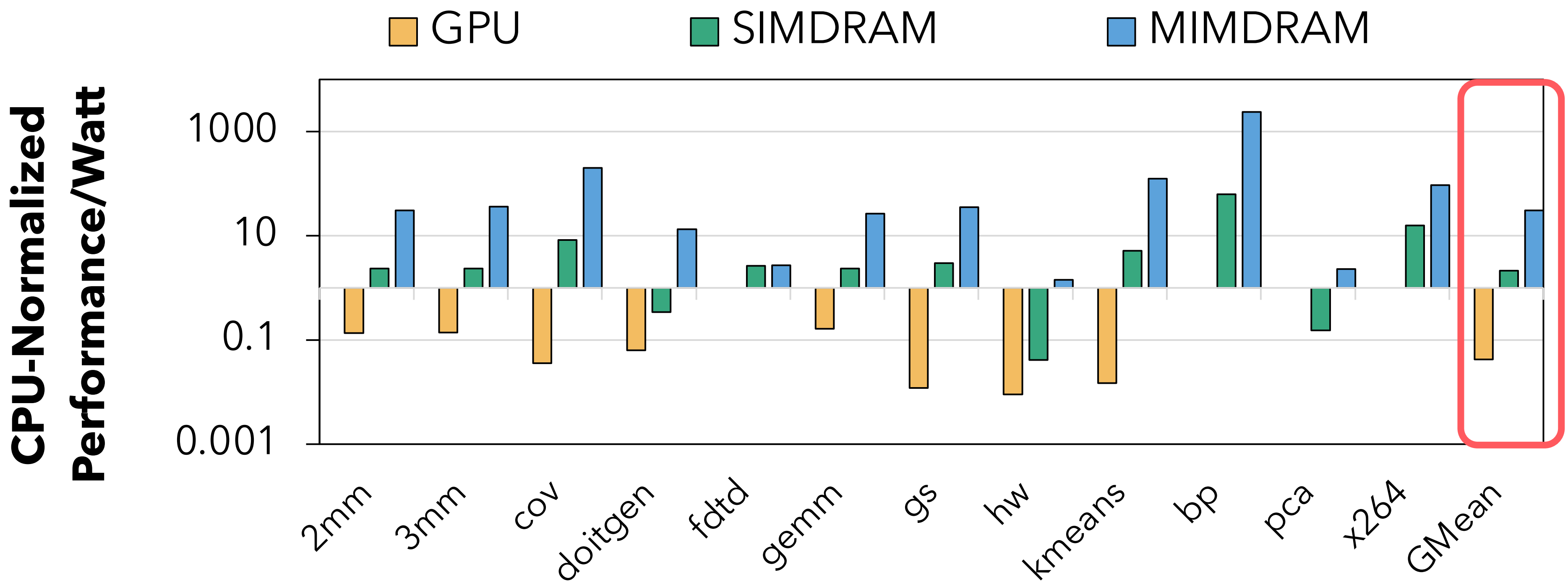
Evaluation: Methodology Overview

- **Evaluation Setup**
  - **CPU:** Intel Skylake CPU
  - **GPU:** NVIDIA A100 GPU
  - **PUD:** SIMDRAM [Oliveira+, 2021] and DRISA [Li+, 2017]
  - **PND:** Fulcrum [Lenjani+, 2020]
  - **https://github.com/CMU-SAFARI/MIMDRAM**

- **Workloads:**
  - 12 workloads from Polybench, Rodinia, Phoenix, and SPEC2017
  - 495 multi-programmed application mixes

- **Two-Level Analysis**
  - **Single application** → leverages intra-application data parallelism
  - **Multi-programmed workload** → leverages inter-application data parallelism

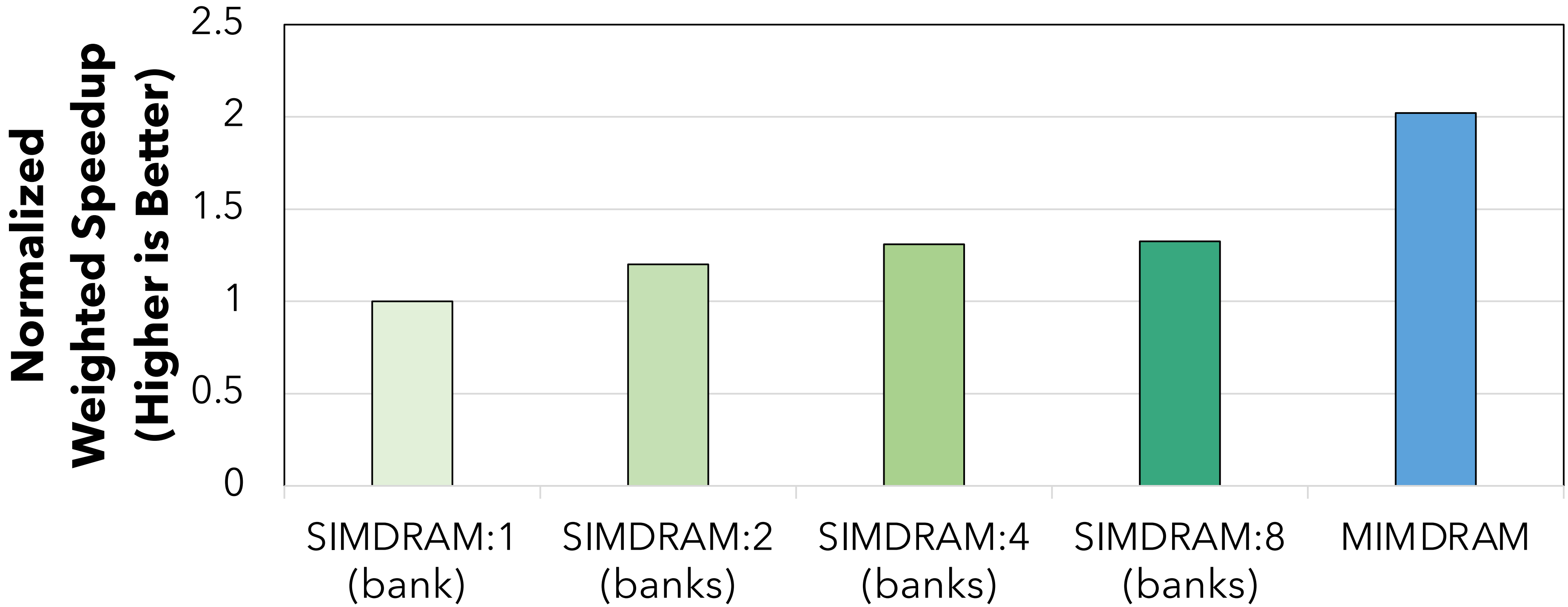
Evaluation:
Single Application Analysis – Energy Efficiency



Takeaway

MIMDRAM significantly improves energy efficiency compared to CPU (30.6x), GPU (6.8x), and SIMDRAM (14.3x)

Evaluation:
Multi-Programmed Workload Analysis



Takeaway

MIMDRAM significantly improves system throughput (1.68x) compared to SIMDRAM

Evaluation:
More in the Paper

- **MIMDRAM with subarray and bank-level parallelism**
  - MIMDRAM provides significant performance gains compared to the baseline CPU (**13.2x**) and GPU (**2x**)

- **Comparison to DRISA and Fulcrum for multi-programmed workloads**
  - MIMDRAM achieves system throughput on par with DRISA and Fulcrum

- **MIMDRAM's SIMD utilization versus SIMDRAM**
  - MIMDRAM provides **15.6x** the utilization of SIMDRAM

- **Area analysis**
  - MIMDRAM adds small area cost to a DRAM chip (**1.11%**) and CPU die (**0.6%**)

MIMDRAM: Summary

We introduced MIMDRAM,
a hardware/software co-designed processing-using-DRAM system

- **Key idea**: leverage fine-grained DRAM for processing-using-DRAM operation
- **HW**: - simple changes to DRAM, enabling concurrent instruction execution
  - low-cost interconnects at the DRAM peripherals for data reduction
- **SW**: - compiler and OS support to generate and map instructions

Our evaluation demonstrates that MIMDRAM

- **significantly** improves **performance**, **energy efficiency**, and **throughput** compared to processor-centric (CPU and GPU) and memory-centric (SIMDRAM, DRISA, and Fulcrum) architectures
- incurs **small area cost** to a DRAM chip and CPU die

https://github.com/CMU-SAFARI/MIMDRAM

Two Other Works on PIM Programmability

Adoption: How to Ease **Programmability**? (I)

- Geraldo F. Oliveira, Alain Kohli, David Novo, Juan Gómez-Luna, Onur Mutlu, **"DaPPA: A Data-Parallel Framework for Processing-in-Memory Architectures,"** in *PACT SRC Student Competition,* Vienna, Austria, October 2023.

DaPPA: A Data-Parallel Framework for Processing-in-Memory Architectures

Geraldo F. Oliveira* Alain Kohli* David Novo‡ Juan Gómez-Luna* Onur Mutlu*

**ETH Zürich ‡LIRMM, Univ. Montpellier, CNRS*

Adoption: How to Ease **Programmability**? (II)

- Jinfan Chen, Juan Gómez-Luna, Izzat El Hajj, YuXin Guo, and Onur Mutlu,
"SimplePIM: A Software Framework for Productive and Efficient Processing in Memory"
*Proceedings of the 32nd International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, Vienna, Austria, October 2023.

SimplePIM: A Software Framework for Productive and Efficient Processing-in-Memory

Jinfan Chen[1] Juan Gómez-Luna[1] Izzat El Hajj[2] Yuxin Guo[1] Onur Mutlu[1]
[1]ETH Zürich [2]American University of Beirut

Real DRAM Chips Are Already Quite Capable: FC-DRAM & SiMRA

Recall: DRAM Testing Infrastructure

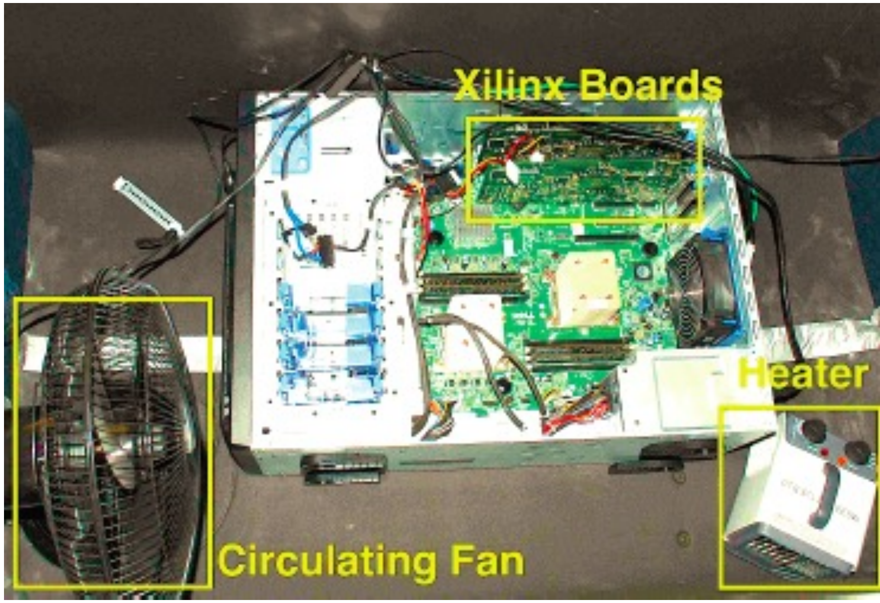
An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

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

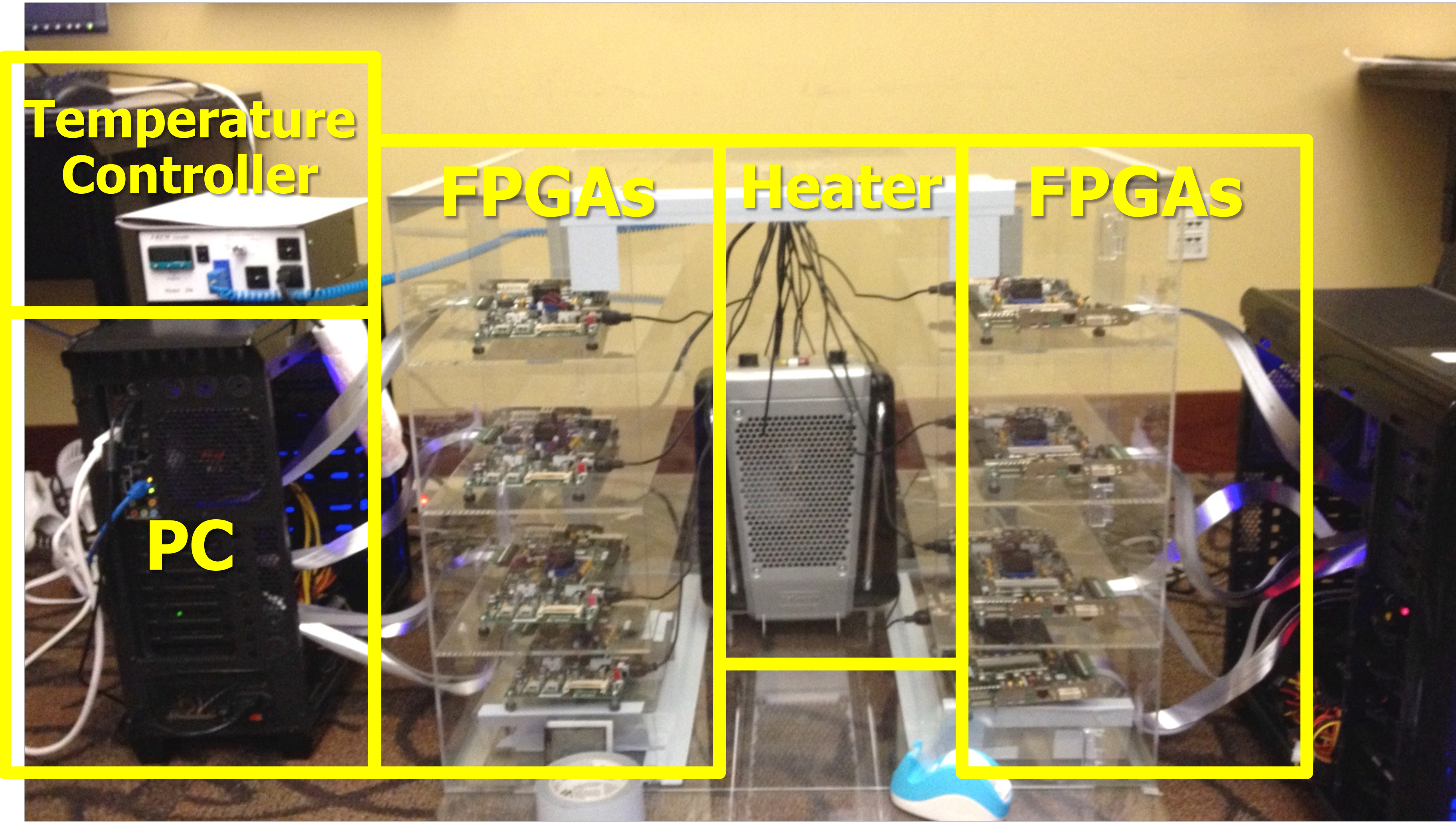
Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

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

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



Recall: DRAM Testing Infrastructure

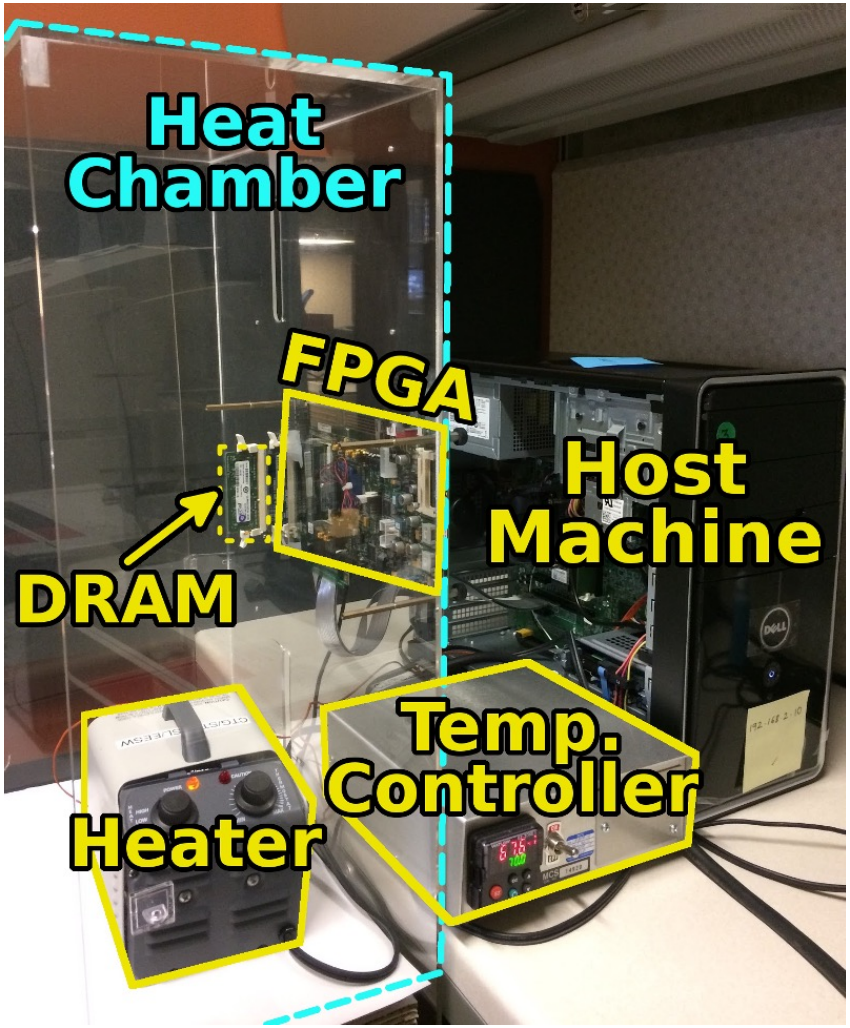


SoftMC: Open Source DRAM Infrastructure

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

- **Flexible**
- **Easy to Use (C++ API)**
- **Open-source**

  github.com/CMU-SAFARI/SoftMC



SoftMC: Open Source DRAM Infrastructure

- Hasan Hassan, Nandita Vijaykumar, Samira Khan, Saugata Ghose, Kevin Chang, Gennady Pekhimenko, Donghyuk Lee, Oguz Ergin, and Onur Mutlu,
"SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies"
Proceedings of the 23rd International Symposium on High-Performance Computer Architecture (***HPCA***), Austin, TX, USA, February 2017.
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)]
[Full Talk Lecture (39 minutes)]
[Source Code]

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

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

[1]*ETH Zürich* [2]*TOBB University of Economics & Technology* [3]*Carnegie Mellon University* [4]*University of Virginia* [5]*Microsoft Research* [6]*NVIDIA Research*

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

DRAM Bender

- Ataberk Olgun, Hasan Hassan, A Giray Yağlıkçı, Yahya Can Tuğrul, Lois Orosa, Haocong Luo, Minesh Patel, Oğuz Ergin, and Onur Mutlu,
"DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips"
*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (**TCAD**)*, 2023.
[Extended arXiv version]
[DRAM Bender Source Code]
[DRAM Bender Tutorial Video (43 minutes)]

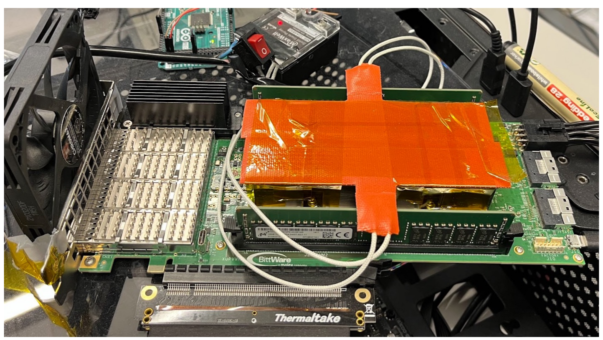
DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips

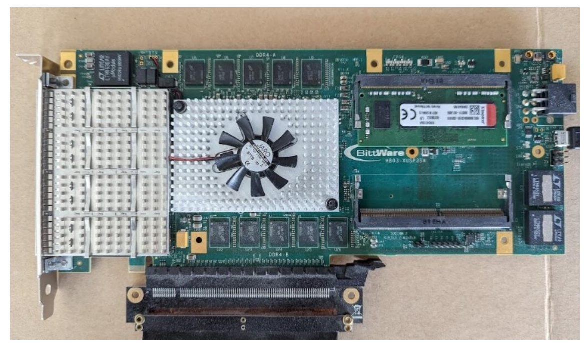
Ataberk Olgun[§] Hasan Hassan[§] A. Giray Yağlıkçı[§] Yahya Can Tuğrul[§†]
Lois Orosa[§⊙] Haocong Luo[§] Minesh Patel[§] Oğuz Ergin[†] Onur Mutlu[§]
[§]*ETH Zürich* [†]*TOBB ETÜ* [⊙]*Galician Supercomputing Center*

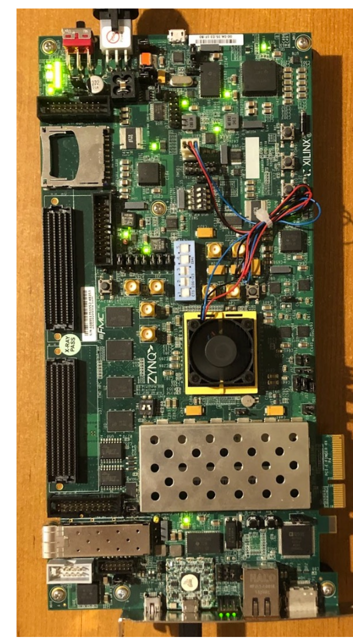
DRAM Bender: Prototypes

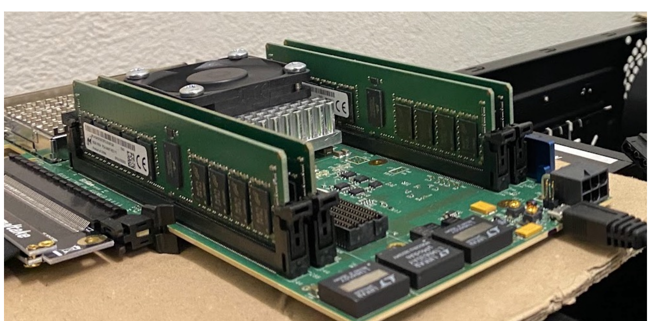
Testing Infrastructure	Protocol Support	FPGA Support
SoftMC [134]	DDR3	One Prototype
LiteX RowHammer Tester (LRT) [17]	DDR3/4, LPDDR4	Two Prototypes
DRAM Bender (this work)	**DDR3/DDR4**	**Five Prototypes**

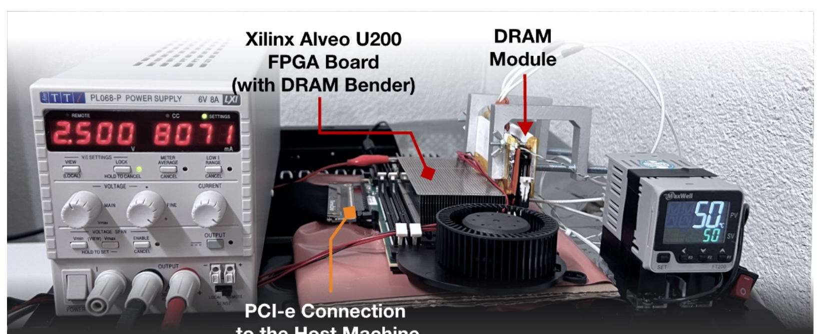
Five out of the box FPGA-based prototypes











https://github.com/CMU-SAFARI/DRAM-Bender

DRAM Chips Are Already (Quite) Capable!

- **Appears at HPCA 2024** https://arxiv.org/pdf/2402.18736.pdf

Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel Yahya Can Tuğrul Ataberk Olgun F. Nisa Bostancı A. Giray Yağlıkçı
Geraldo F. Oliveira Haocong Luo Juan Gómez-Luna Mohammad Sadrosadati Onur Mutlu

ETH Zürich

We experimentally demonstrate that COTS DRAM chips are capable of performing 1) functionally-complete *Boolean operations: NOT, NAND, and NOR and 2) many-input (i.e., more than two-input) AND and OR operations. We present an extensive characterization of* new *bulk bitwise operations in 256 off-the-shelf modern DDR4 DRAM chips. We evaluate the reliability of these operations using a metric called success rate: the fraction of correctly performed bitwise operations. Among our 19 new observations, we highlight four major results. First, we can perform the NOT operation on COTS DRAM chips with 98.37% success rate on average. Second, we can perform up to 16-input NAND, NOR, AND, and OR operations on COTS DRAM chips with high reliability (e.g., 16-input NAND, NOR, AND, and OR with average success rate of 94.94%, 95.87%, 94.94%, and 95.85%, respectively). Third, data pattern only slightly*

DRAM Chips Are Already (Quite) Capable!

- https://arxiv.org/pdf/2312.02880.pdf

PULSAR: Simultaneous Many-Row Activation for Reliable and High-Performance Computing in Off-the-Shelf DRAM Chips

Ismail Emir Yuksel Yahya Can Tugrul F. Nisa Bostanci Abdullah Giray Yaglikci Ataberk Olgun
Geraldo F. Oliveira Melina Soysal Haocong Luo Juan Gomez Luna Mohammad Sadrosadati
Onur Mutlu

ETH Zurich

We propose PULSAR, a new technique to enable high-success-rate and high-performance PuM operations in off-the-shelf DRAM chips. PULSAR leverages our new observation that a carefully-crafted sequence of DRAM commands simultaneously activates up to 32 DRAM rows. PULSAR overcomes the limitations of existing techniques by 1) replicating the input data to improve the success rate and 2) enabling new bulk bitwise operations (e.g., many-input majority, *Multi-RowInit*, and *Bulk-Write*) to improve the performance.

DRAM Chips Are Already (Quite) Capable!

- **Appears at DSN 2024**

Code Reproducible

Dataset Reproducible

Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel[1] Yahya Can Tuğrul[1,2] F. Nisa Bostancı[1] Geraldo F. Oliveira[1]
A. Giray Yağlıkçı[1] Ataberk Olgun[1] Melina Soysal[1] Haocong Luo[1]
Juan Gómez-Luna[1] Mohammad Sadrosadati[1] Onur Mutlu[1]
[1]*ETH Zürich* [2]*TOBB University of Economics and Technology*

https://arxiv.org/pdf/2405.06081

The Capability of COTS DRAM Chips

We **demonstrate** that **COTS DRAM chips:**

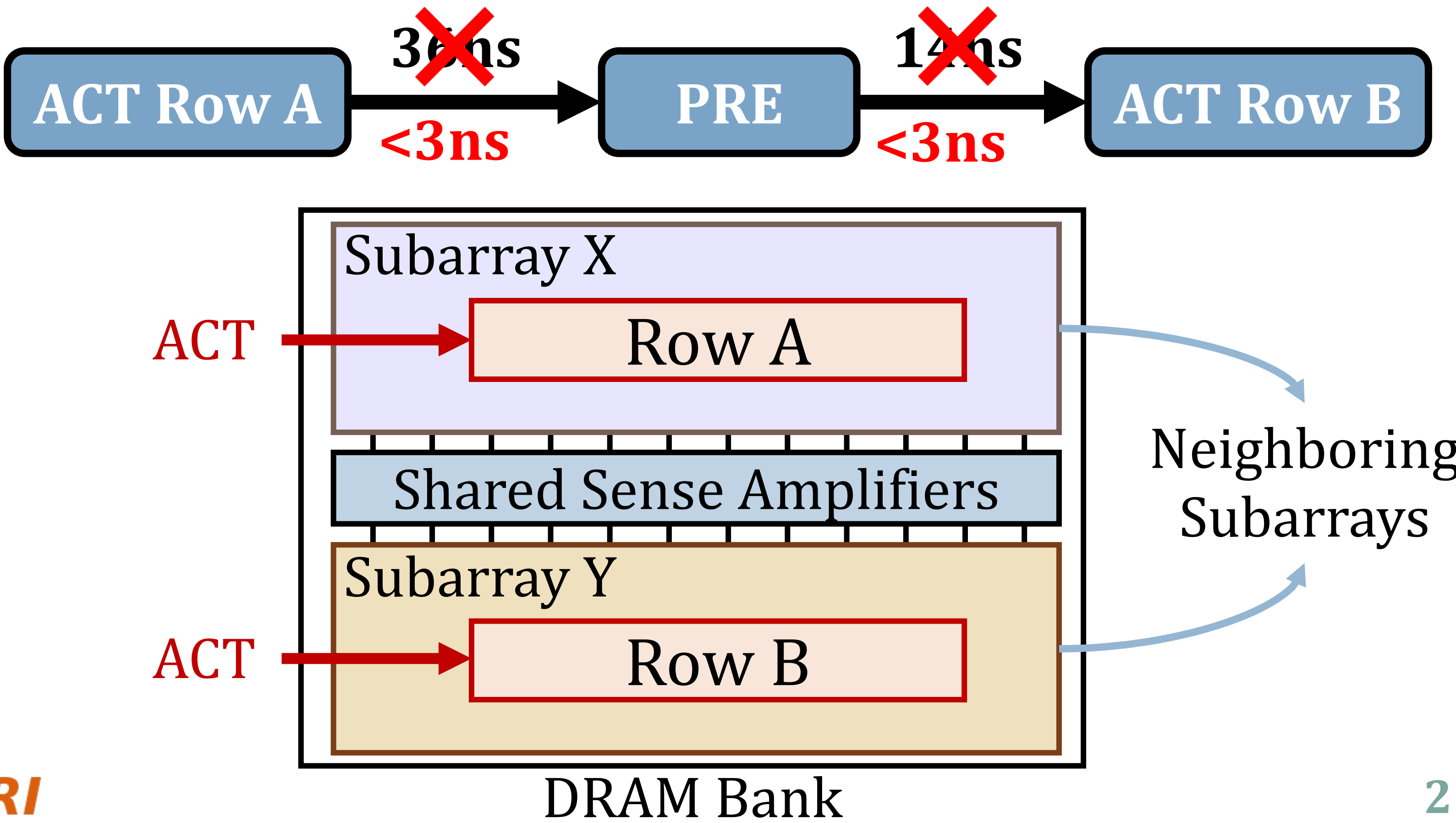
1 Can **simultaneously activate** up to **48 rows** in **two neighboring subarrays**

2 Can perform **NOT operation** with up to **32 output operands**

3 Can perform up to **16-input AND, NAND, OR, and NOR** operations

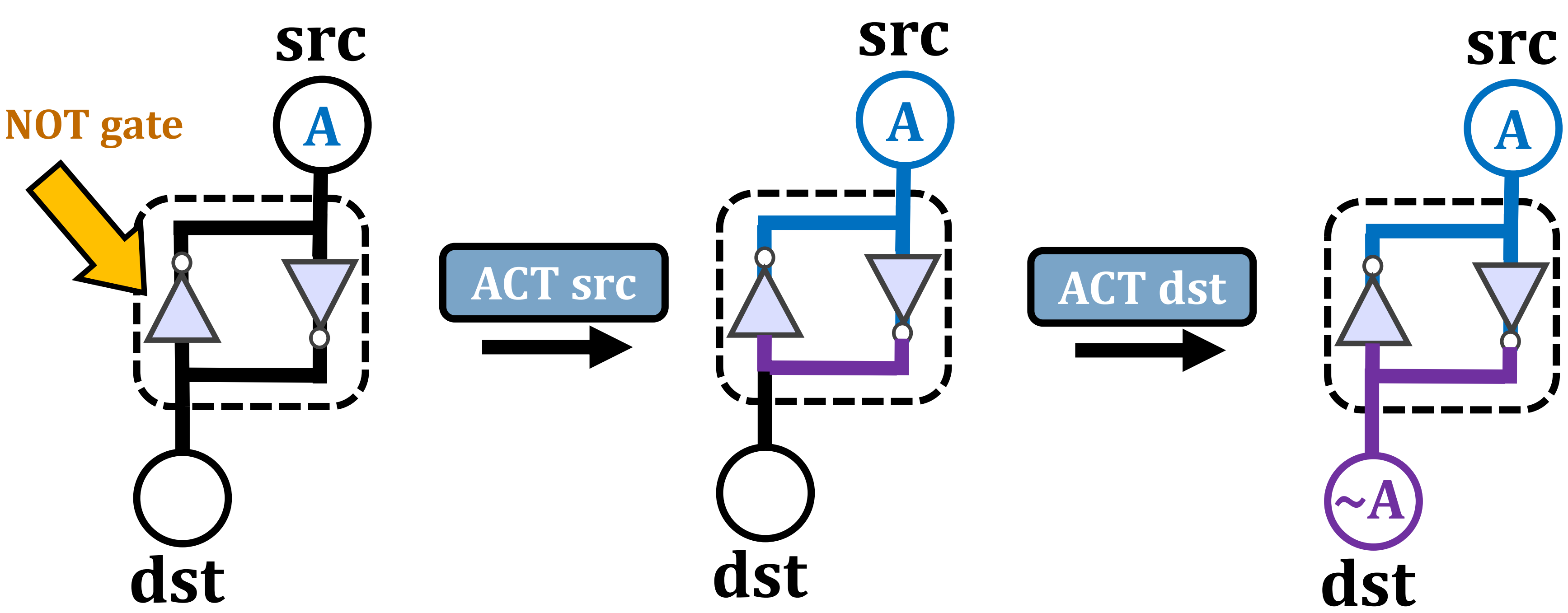
Finding: SiMRA Across Subarrays

Activating two rows in **quick succession**
can **simultaneously** activate
multiple rows in neighboring subarrays



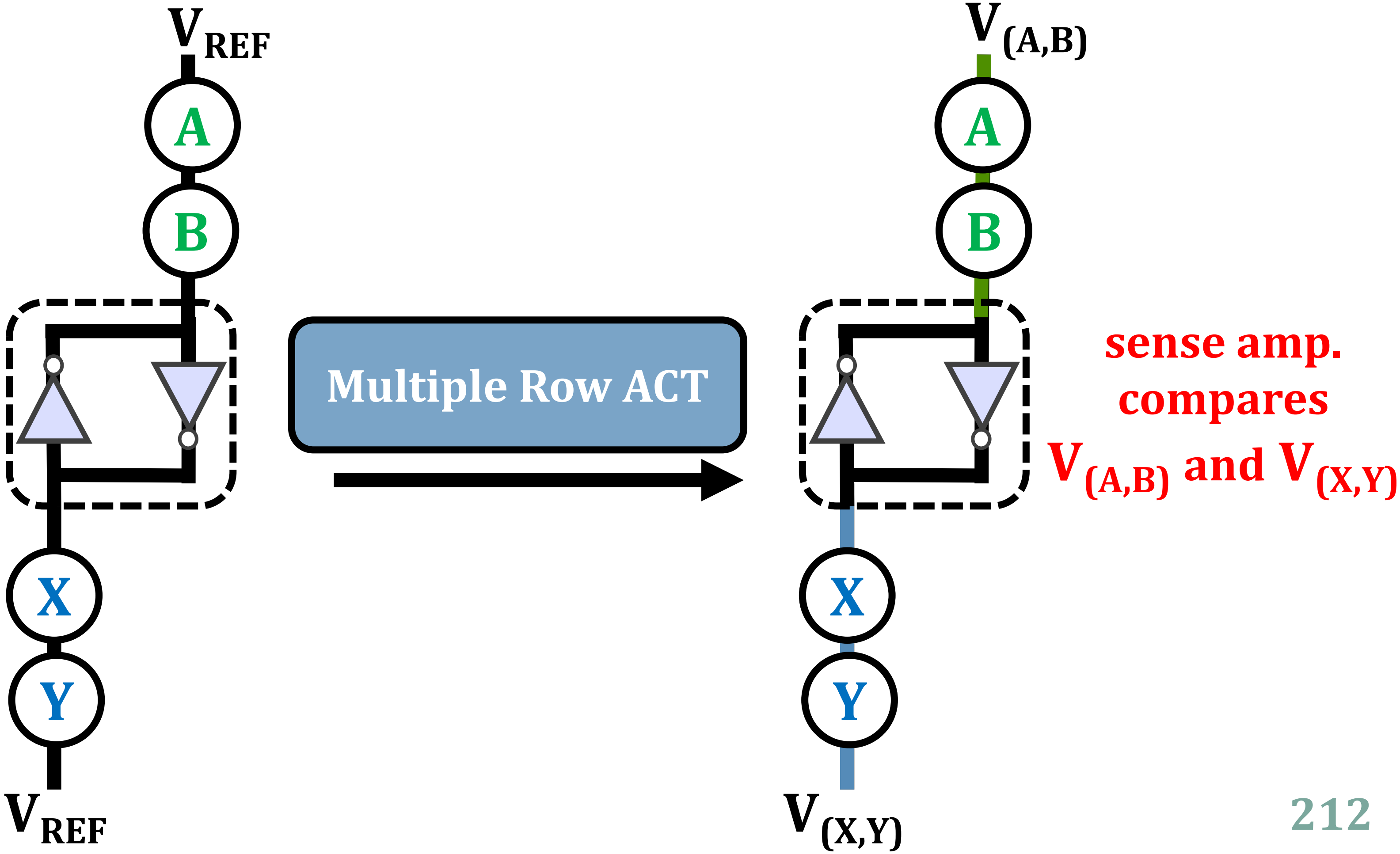
Key Idea: NOT Operation

Connect rows in neighboring subarrays
through **a NOT gate** by simultaneously activating rows

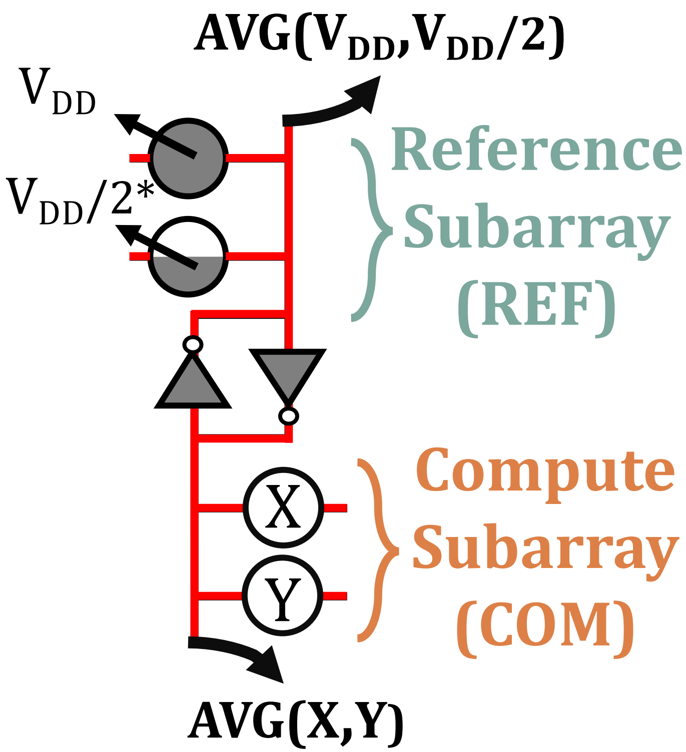


Key Idea: NAND, NOR, AND, OR

Manipulate the bitline voltage to express **a wide variety of functions** using multiple-row activation in neighboring subarrays

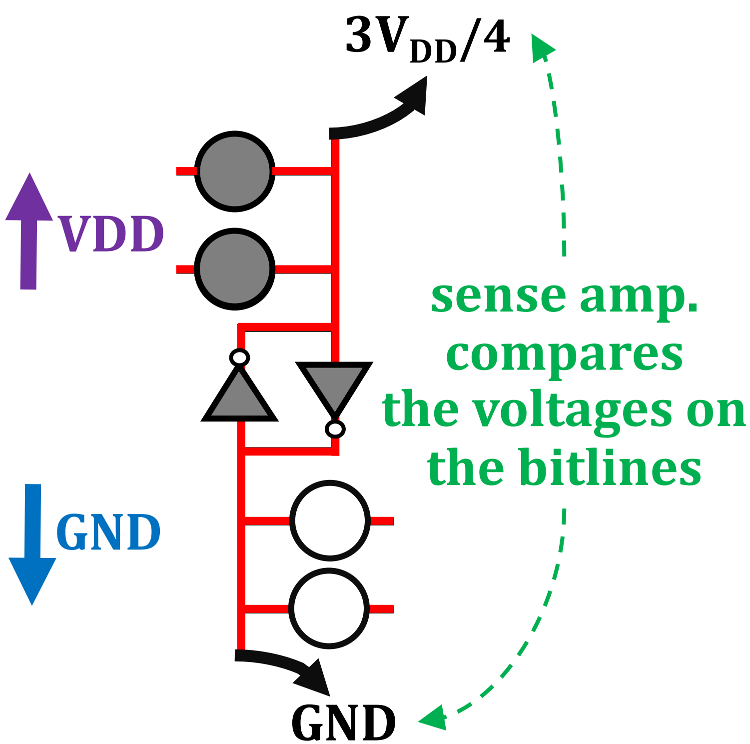


Two-Input AND and NAND Operations



*Gao et al., "FracDRAM: Fractional Values in Off-the-Shelf DRAM," in MICRO, 2022.

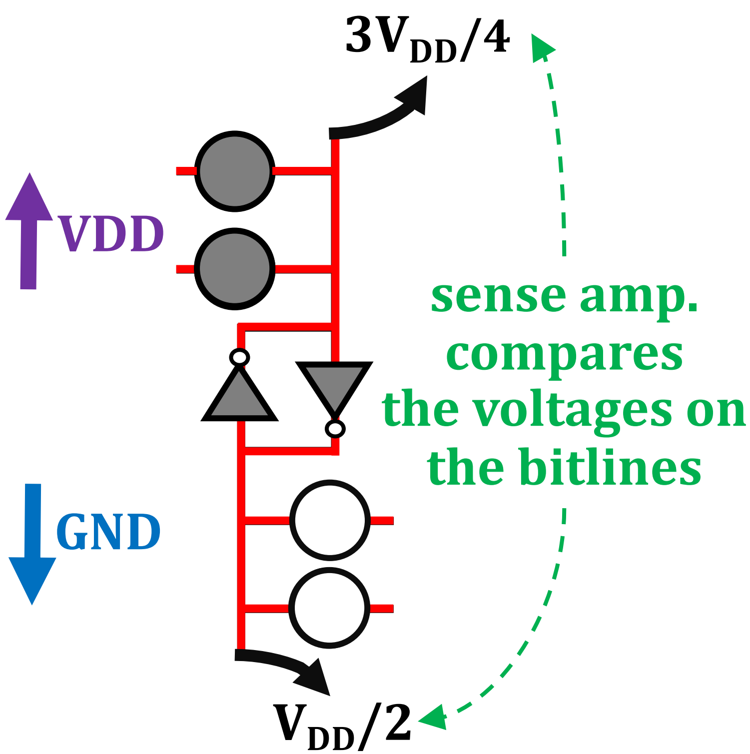
Two-Input AND and NAND Operations



$V_{DD}=1$ & $GND = 0$

X	Y	COM	REF
0	0	0	1

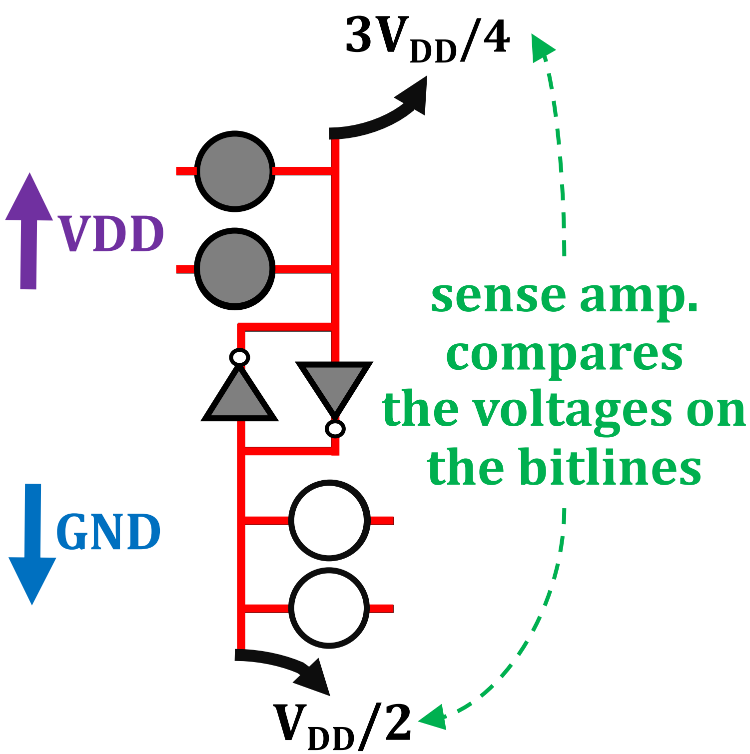
Two-Input AND and NAND Operations



V_{DD}=1 & GND = 0

X	Y	COM	REF
0	0	0	1
0	1	0	1

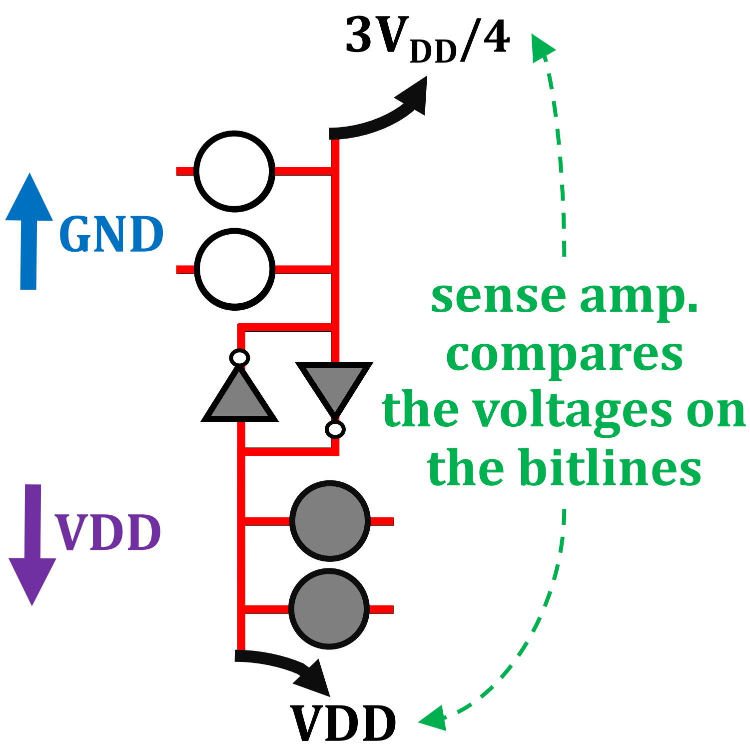
Two-Input AND and NAND Operations



V_{DD}=1 & GND = 0

X	Y	COM	REF
0	0	0	1
0	1	0	1
1	0	0	1

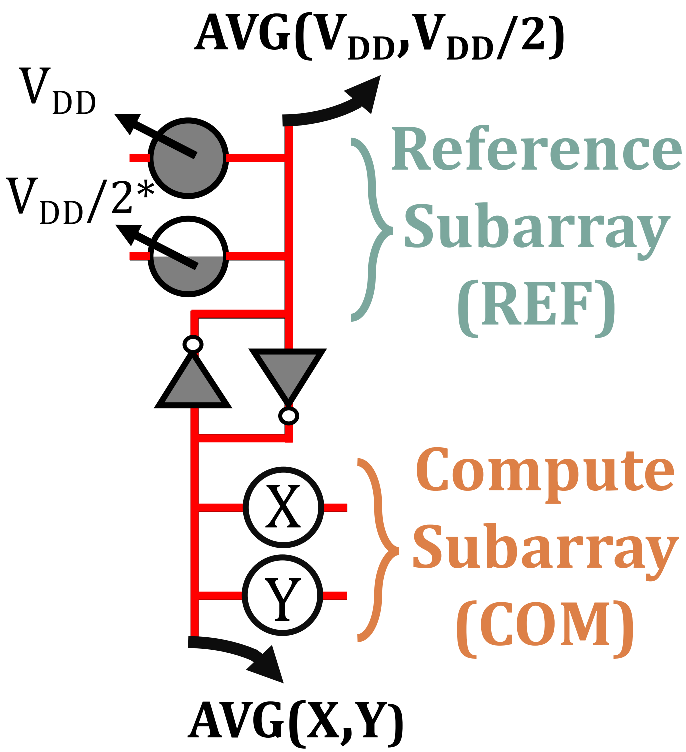
Two-Input AND and NAND Operations



V_{DD}=1 & GND = 0

X	Y	COM	REF
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Two-Input AND and NAND Operations



V_{DD}=1 & GND = 0

X	Y	COM	REF
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0
		AND	NAND

*Gao et al., "FracDRAM: Fractional Values in Off-the-Shelf DRAM," in MICRO, 2022.

Many-Input AND, NAND, OR, and NOR Operations

We can express AND, NAND, OR, and NOR operations by carefully manipulating the **reference voltage**

Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel Yahya Can Tuğrul Ataberk Olgun F. Nisa Bostancı A. Giray Yağlıkçı
Geraldo F. Oliveira Haocong Luo Juan Gómez-Luna Mohammad Sadrosadati Onur Mutlu

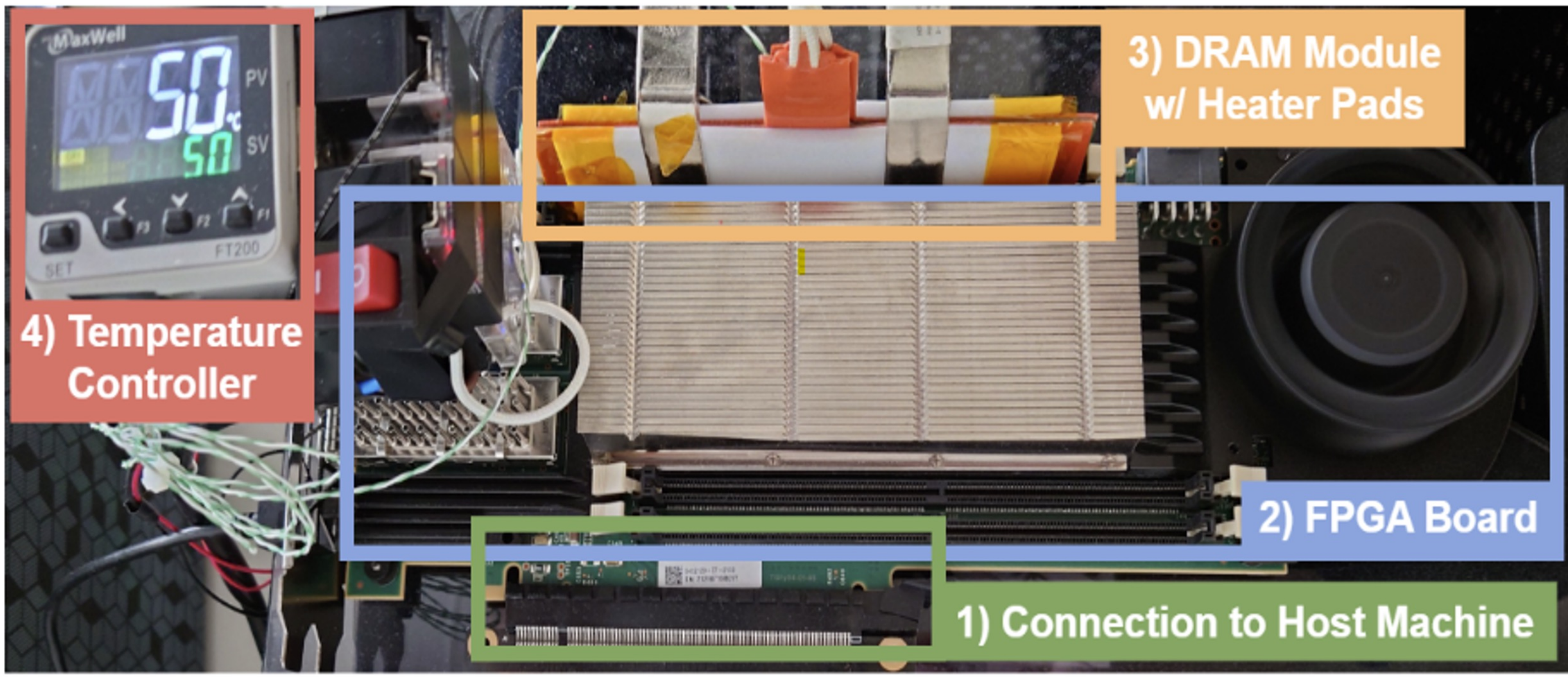
ETH Zürich

(More details in the paper)

https://arxiv.org/pdf/2402.18736.pdf

DRAM Testing Infrastructure

- Developed from DRAM Bender [Olgun+, TCAD'23]*
- Fine-grained control over DRAM commands, timings, and temperature



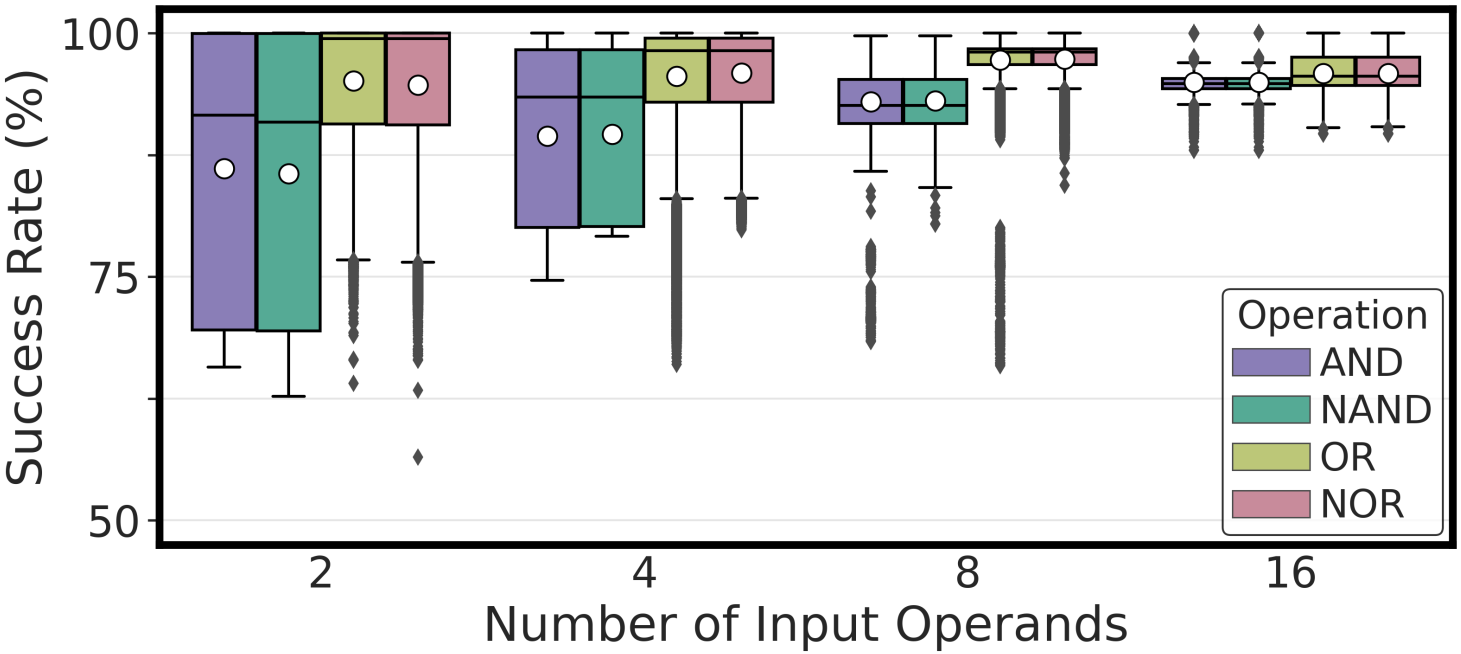
*Olgun et al., "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," TCAD, 2023.

DRAM Chips Tested

- 256 DDR4 chips from two major DRAM manufacturers
- Covers different die revisions and chip densities

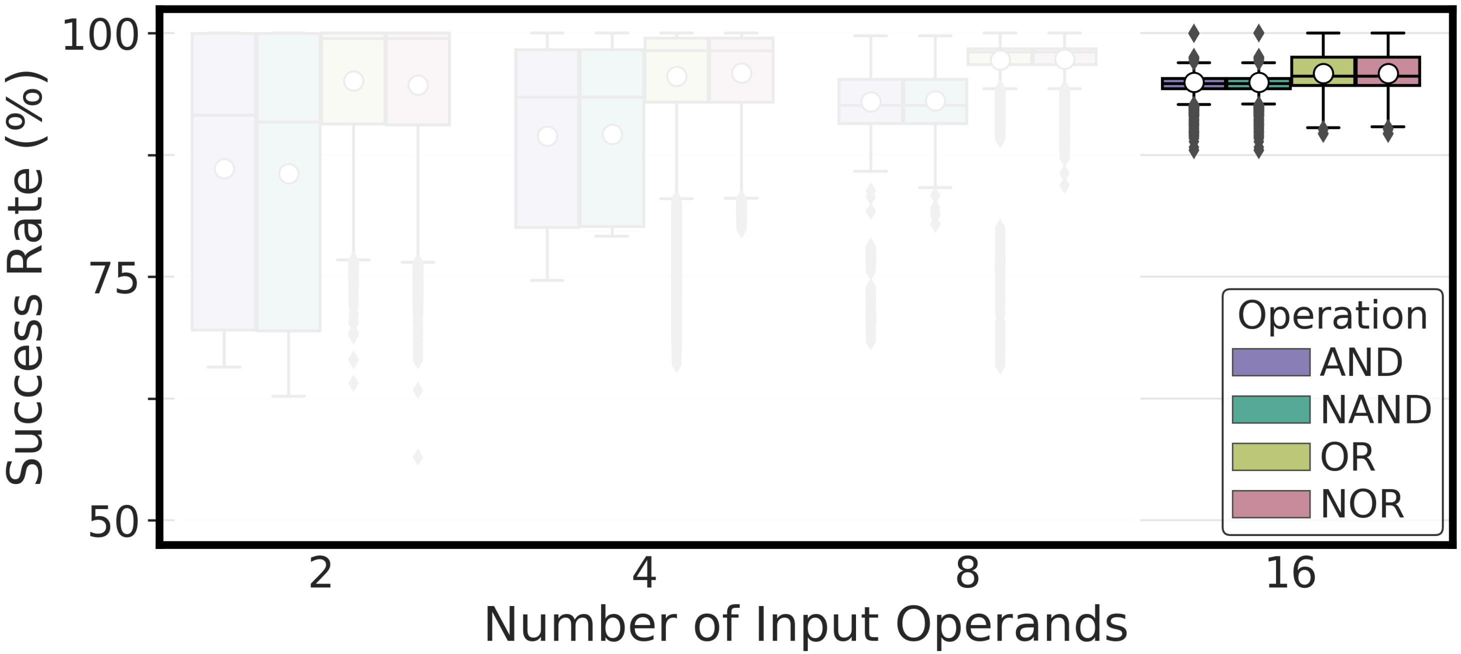
Chip Mfr.	#Modules (#Chips)	Die Rev.	Mfr. Date[a]	Chip Density	Chip Org.	Speed Rate
SK Hynix	9 (72)	M	N/A	4Gb	x8	2666MT/s
	5 (40)	A	N/A	4Gb	x8	2133MT/s
	1 (16)	A	N/A	8Gb	x8	2666MT/s
	1 (32)	A	18-14	4Gb	x4	2400MT/s
	1 (32)	A	16-49	8Gb	x4	2400MT/s
	1 (32)	M	16-22	8Gb	x4	2666MT/s
Samsung	1 (8)	F	21-02	4Gb	x8	2666MT/s
	2 (16)	D	21-10	8Gb	x8	2133MT/s
	1 (8)	A	22-12	8Gb	x8	3200MT/s

Performing AND, NAND, OR, and NOR



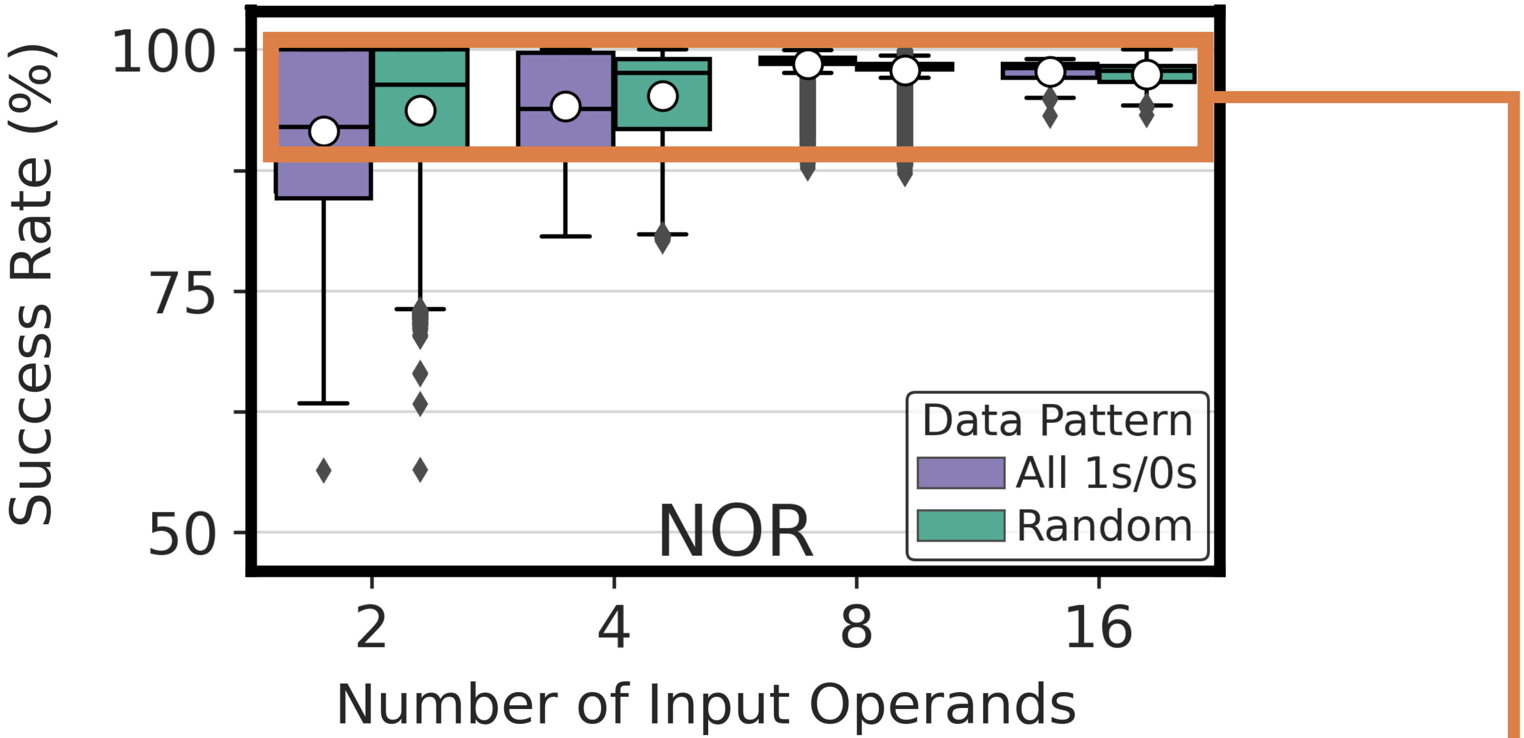
COTS DRAM chips can perform {2, 4, 8, 16}-input AND, NAND, OR, and NOR operations

Performing AND, NAND, OR, and NOR



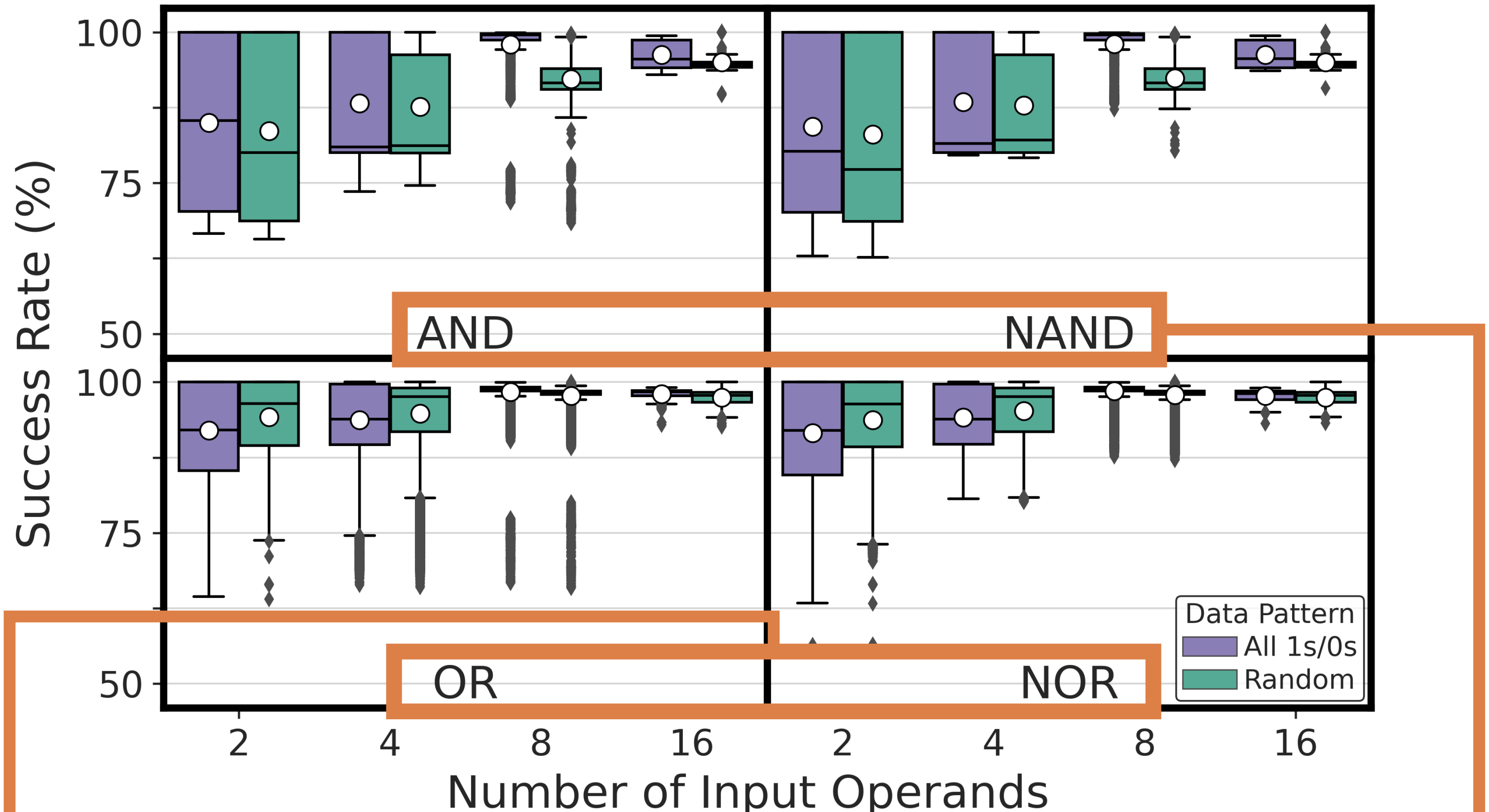
COTS DRAM chips can perform 16-input AND, NAND, OR, and NOR operations with very high success rate (>94%)

Impact of Data Pattern



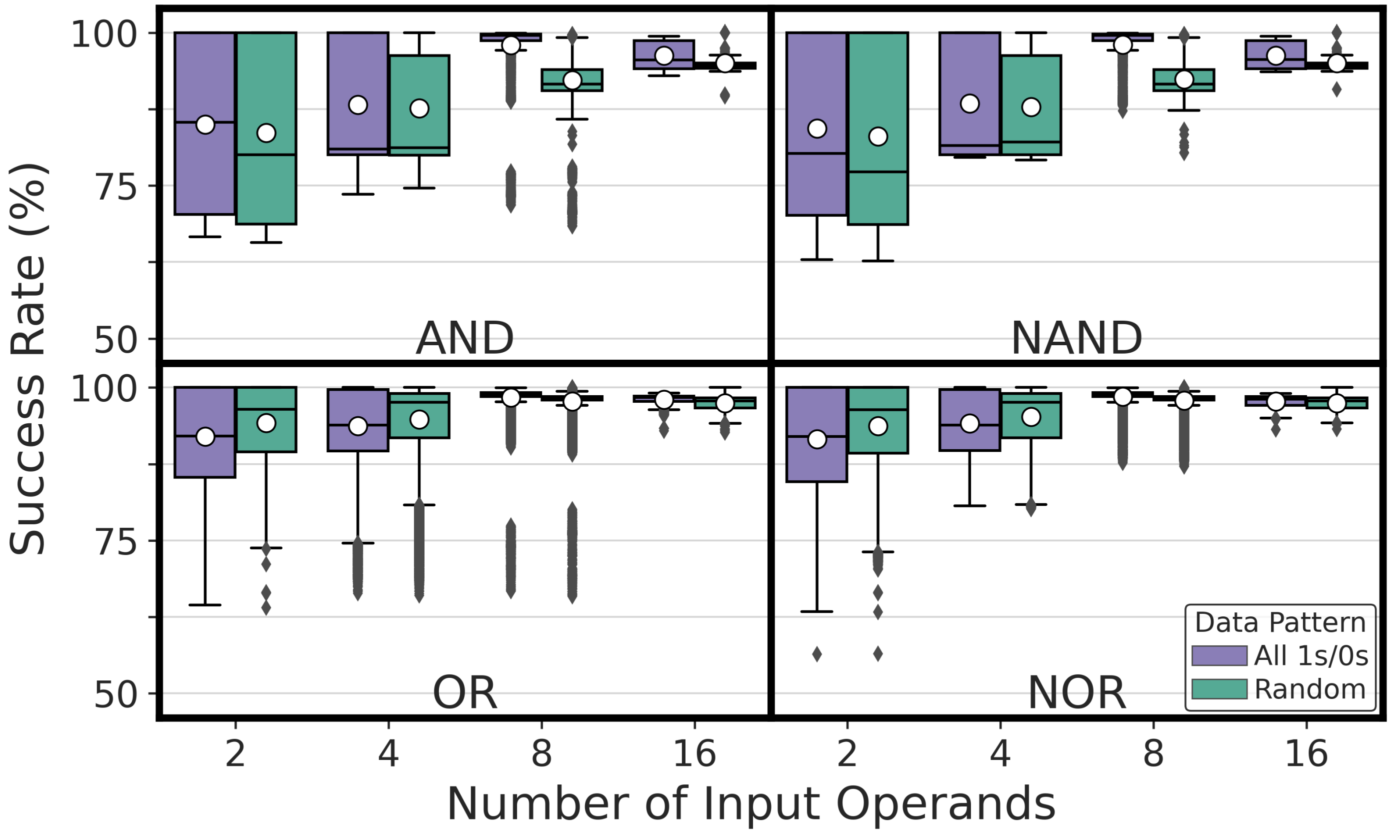
1.98% variation in **average success rate** across **all number of input operands**

Impact of Data Pattern



Impact of data pattern is **consistent** across all tested operations

Impact of Data Pattern



Data pattern slightly affects the reliability of AND, NAND, OR, and NOR operations

More in the Paper

- Detailed hypotheses & key ideas to perform
  - NOT operation
  - Many-input AND, NAND, OR, and NOR operations

- How the reliability of bitwise operations are affected by
  - The location of activated rows
  - Temperature (for AND, NAND, OR, and NOR)
  - DRAM speed rate
  - Chip density and die revision

- Discussion on the limitations of COTS DRAM chips

Available on arXiv

Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel Yahya Can Tuğrul Ataberk Olgun F. Nisa Bostancı A. Giray Yağlıkçı
Geraldo F. Oliveira Haocong Luo Juan Gómez-Luna Mohammad Sadrosadati Onur Mutlu

ETH Zürich

Processing-using-DRAM (PuD) is an emerging paradigm that leverages the analog operational properties of DRAM circuitry to enable massively parallel in-DRAM computation. PuD has the potential to significantly reduce or eliminate costly data movement between processing elements and main memory. A common approach for PuD architectures is to make use of bulk bitwise computation (e.g., AND, OR, NOT). Prior works experimentally demonstrate three-input MAJ (i.e., MAJ3) and two-input AND and OR operations in commercial off-the-shelf (COTS) DRAM chips. Yet, demonstrations on COTS DRAM chips do not *provide a functionally complete set of operations (e.g., NAND or AND and NOT).*

We experimentally demonstrate that COTS DRAM chips are capable of performing 1) functionally-complete *Boolean operations: NOT, NAND, and NOR and 2) many-input (i.e., more than two-input) AND and OR operations. We present an extensive*

systems and applications [12, 13]. Processing-using-DRAM (PuD) [29–32] is a promising paradigm that can alleviate the data movement bottleneck. PuD uses the analog operational properties of the DRAM circuitry to enable massively parallel in-DRAM computation. Many prior works [29–53] demonstrate that PuD can greatly reduce or eliminate data movement.

A widely used approach for PuD is to perform bulk bitwise operations, i.e., bitwise operations on large bit vectors. To perform bulk bitwise operations using DRAM, prior works propose modifications to the DRAM circuitry [29–31, 33, 35, 36, 43, 44, 46, 48–58]. Recent works [38, 41, 42, 45] experimentally demonstrate the feasibility of executing data copy & initialization [42, 45], i.e., the RowClone operation [49], and a subset of bitwise operations, i.e., three-input bitwise majority (MAJ3) and two-input AND and OR operations in unmodified commercial off-the-shelf (COTS) DRAM chips by operating beyond

https://arxiv.org/pdf/2402.18736.pdf

Summary

- We experimentally demonstrate that **commercial off-the-shelf (COTS)** DRAM chips can perform:
  - **Functionally-complete** Boolean operations: NOT, NAND, and NOR
  - **Up to 16-input** AND, NAND, OR, and NOR operations
- We characterize **the success rate** of these operations on **256 COTS DDR4 chips** from **two major manufacturers**
- We highlight **two key results:**
  - We can perform **NOT** and **{2, 4, 8, 16}-input AND, NAND, OR, and NOR** operations on COTS DRAM chips with **very high success rates (>94%)**
  - **Data pattern** and **temperature** only slightly affect the reliability of these operations

We believe these empirical results demonstrate the promising potential of using DRAM as a computation substrate

Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips

Experimental Characterization and Analysis

Code Reproducible

Dataset Reproducible

İsmail Emir Yüksel

Yahya C. Tuğrul F. Nisa Bostancı Geraldo F. Oliveira

A. Giray Yağlıkçı Ataberk Olgun Melina Soysal Haocong Luo

Juan Gómez–Luna Mohammad Sadr Onur Mutlu

SAFARI

ETH zürich

Executive Summary

Motivation:

- **Processing-Using-DRAM (PUD)** alleviates **data movement bottlenecks**
- Commercial off-the-shelf (COTS) DRAM chips can perform **three-input majority (MAJ3)** and **in-DRAM copy** operations

Goal: To experimentally analyze and understand

- The **computational capability** of COTS DRAM chips beyond that of prior works
- The **robustness** of such capability under various **operating conditions**

Experimental Study: 120 DDR4 chips from **two major manufacturers**

- COTS DRAM chips can perform **MAJ5, MAJ7, and MAJ9** operations and **copy** one DRAM row to **up to 31 different rows** at once
- Storing **multiple redundant copies** of MAJ's input operands (i.e., input replication) drastically increases **robustness** (>30% higher success rate)
- **Operating conditions** (temperature, voltage, and data pattern) **affect** the robustness of in-DRAM operations (by up to 11.52% success rate)

Leveraging Simultaneous Many-Row Activation

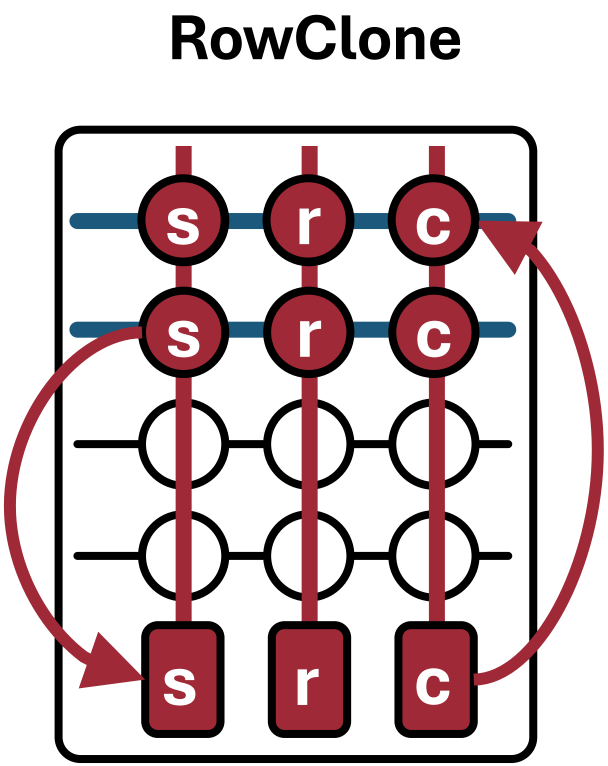
1 Perform **MAJX (where X>3)** operations

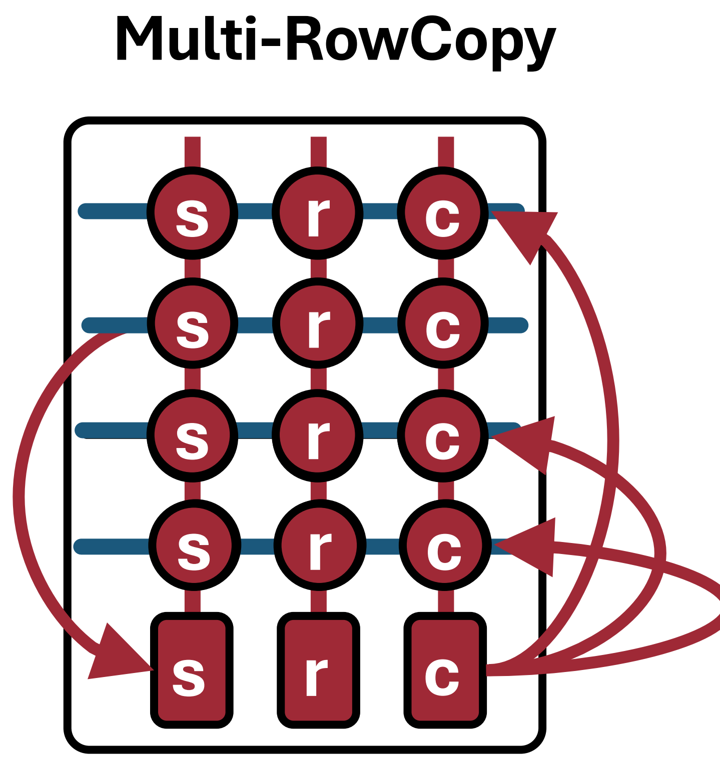
2 **Increase the robustness** of MAJX operations

3 Copy **one row's content** to **multiple rows**

In-DRAM Multiple Row Copy (Multi-RowCopy)

Simultaneously activate many rows to copy **one row's content** to **multiple destination rows**





[Seshadri+ MICRO'13]

Key Takeaways from Multi-RowCopy

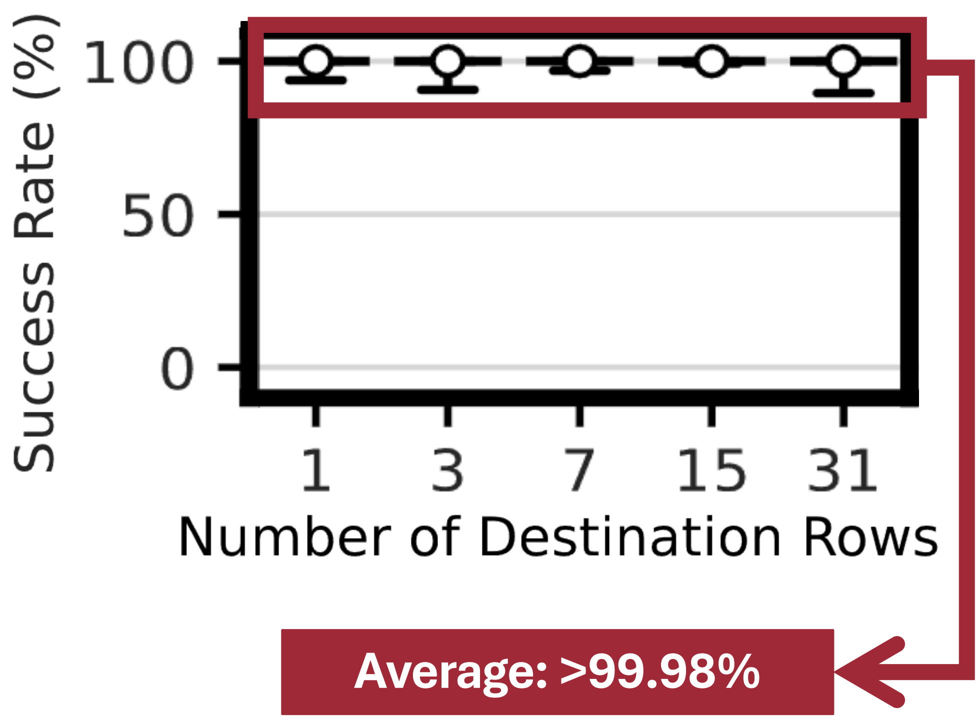
Key Takeaway 1

COTS DRAM chips are capable of copying one row's data to 1, 3, 7, 15, and 31 other rows at very high success rates

Key Takeaway 2

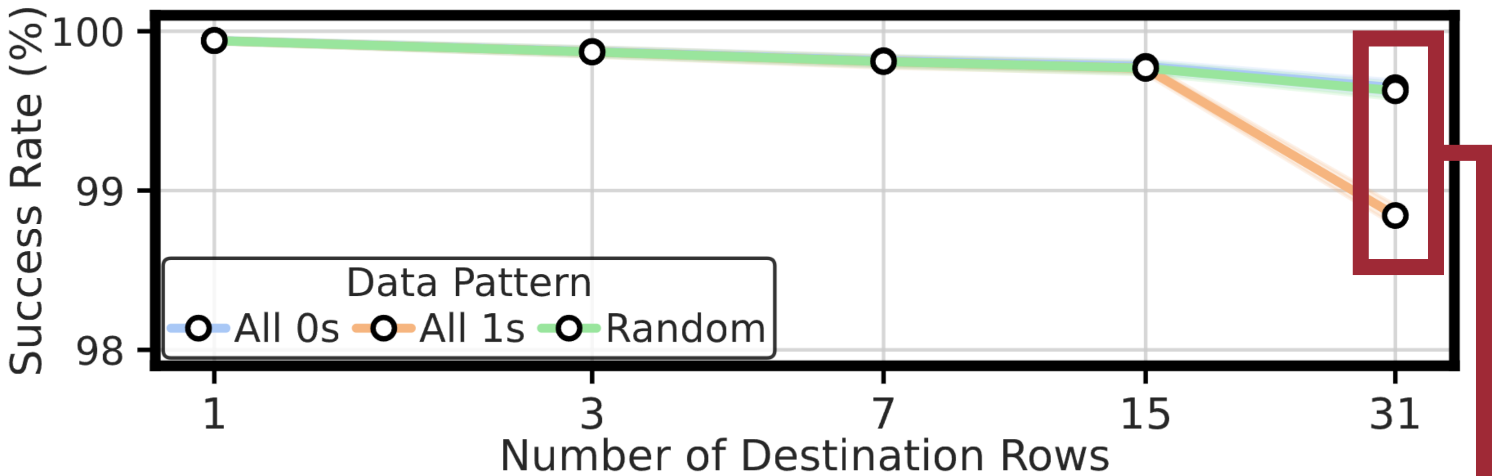
Multi-RowCopy in COTS DRAM chips is highly resilient to changes in data pattern, temperature, and wordline voltage

Robustness of Multi-RowCopy



COTS DRAM chips can copy one row's content to up to 31 rows with a very high success rate

Impact of Data Pattern



At most 0.79% decrease in average success rate

Data pattern has a small effect on the success rate of the Multi-RowCopy operation

Also in the Paper: Impact of Temperature & Voltage

Temperature

50 °C → 90 °C

Increasing temperature up to 90°C has a very small effect on the success rate of the Multi-RowCopy operation

Wordline Voltage

2.5V → 2.1V

Reducing the wordline voltage only slightly affects the success rate of the Multi-RowCopy operation

More in the Paper

- Detailed hypotheses and key ideas on
  - Hypothetical row decoder circuitry
  - Input Replication
- More characterization results
  - Power consumption of simultaneous many-row activation
  - Effect of timing delays between ACT-PRE and PRE-ACT commands
  - Effect of temperature and wordline voltage
- Circuit-level (SPICE) experiments for input replication
- Potential performance benefits of enabling new in-DRAM operations
  - Majority-based computation
  - Content destruction-based cold-boot attack prevention
- Discussions on the limitations of tested COTS DRAM chips

Available on arXiv

Code Reproducible

Dataset Reproducible

Simultaneous Many-Row Activation in Off-the-Shelf DRAM Chips: Experimental Characterization and Analysis

İsmail Emir Yüksel[1] Yahya Can Tuğrul[1,2] F. Nisa Bostancı[1] Geraldo F. Oliveira[1]
A. Giray Yağlıkçı[1] Ataberk Olgun[1] Melina Soysal[1] Haocong Luo[1]
Juan Gómez-Luna[1] Mohammad Sadrosadati[1] Onur Mutlu[1]
[1]*ETH Zürich* [2]*TOBB University of Economics and Technology*

We experimentally analyze the computational capability of commercial off-the-shelf (COTS) DRAM chips and the robustness of these capabilities under various timing delays between DRAM commands, data patterns, temperature, and voltage levels. We extensively characterize 120 COTS DDR4 chips from two major manufacturers. We highlight four key results of our study. First, COTS DRAM chips are capable of 1) simultaneously activating up to 32 rows (i.e., simultaneous many-row activation), 2) executing a majority of X (`MAJX`) operation where X>3 (i.e., `MAJ5`, `MAJ7`, and `MAJ9` operations), and 3) copying a DRAM row (concurrently) to up to 31 other DRAM rows, which we call `Multi-RowCopy`. Second, storing multiple copies of `MAJX`'s input operands on all simultaneously activated rows drastically increases the success rate (i.e., the percentage of DRAM cells that correctly *perform the computation) of the `MAJX` operation. For example, `MAJ3` with 32-row activation (i.e.,*

A subset of PIM proposals devise mechanisms that enable PUM using DRAM cells for computation, including data copy and initialization [67, 72, 77, 78, 89, 104, 127], Boolean logic [56, 64–66, 68, 70, 72, 76, 79, 122, 127–129], majority-based arithmetic [64, 66, 69, 72, 91, 127, 130, 131], and lookup table based operations [82, 106, 107, 132]. We refer to DRAM-based PUM as *Processing-Using-DRAM (PUD)* and the computation performed using DRAM cells as PUD operations.

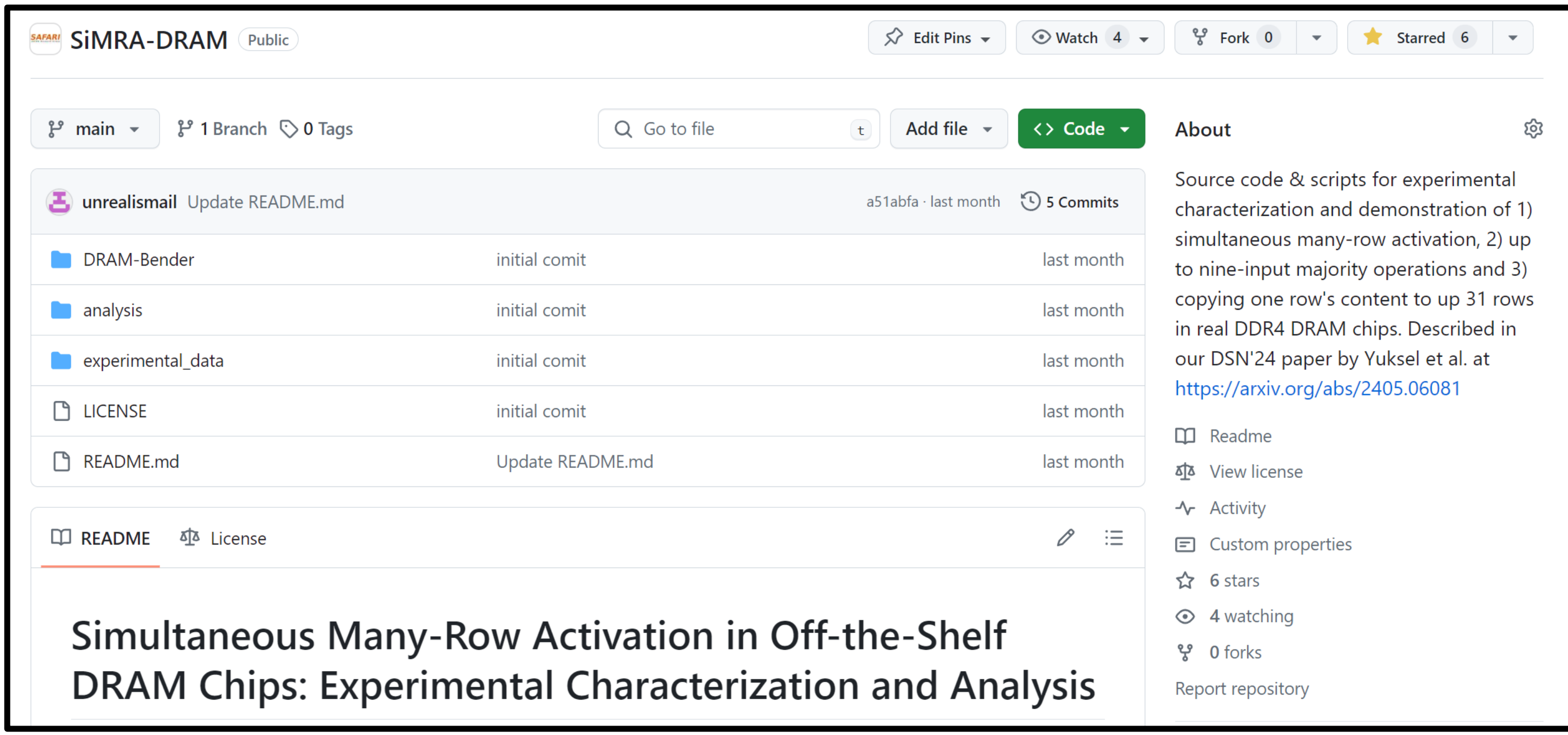
PUD benefits from the bulk data parallelism in DRAM devices to perform bulk bitwise PUD operations. Prior works show that bulk bitwise operations are used in a wide variety of important applications, including databases and web search [64, 67, 79, 130, 133–140], data analytics [64, 141–144], graph processing [56, 80, 94, 130, 145], genome analysis [60, 99, 146–149], cryptography [150, 151], set operations [56, 64], and hyperdimensional computing [152–154].

https://arxiv.org/pdf/2405.06081

Our Work is Open Source and Artifact Evaluated

Code Reproducible

Dataset Reproducible



https://github.com/CMU-SAFARI/SiMRA-DRAM

What Else Can We Do Using DRAM?

In-DRAM True Random Number Generation

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, Lois Orosa, and Onur Mutlu, **"D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput"**
  *Proceedings of the 25th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Washington, DC, USA, February 2019.
  [Slides (pptx) (pdf)]
  [Full Talk Video (21 minutes)]
  [Full Talk Lecture Video (27 minutes)]
  Top Picks Honorable Mention by IEEE Micro.

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

Jeremie S. Kim[‡§] Minesh Patel[§] Hasan Hassan[§] Lois Orosa[§] Onur Mutlu[§‡]
[‡]Carnegie Mellon University [§]ETH Zürich

In-DRAM True Random Number Generation

- Ataberk Olgun, Minesh Patel, A. Giray Yaglikci, Haocong Luo, Jeremie S. Kim, F. Nisa Bostanci, Nandita Vijaykumar, Oguz Ergin, and Onur Mutlu,
  "QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips"
  Proceedings of the 48th International Symposium on Computer Architecture (***ISCA***), Virtual, June 2021.
  [Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]
  [Talk Video (25 minutes)]
  [SAFARI Live Seminar Video (1 hr 26 mins)]

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

Ataberk Olgun[§†] Minesh Patel[§] A. Giray Yağlıkçı[§] Haocong Luo[§]
Jeremie S. Kim[§] F. Nisa Bostancı[§†] Nandita Vijaykumar[§⊙] Oğuz Ergin[†] Onur Mutlu[§]
[§]*ETH Zürich* [†]*TOBB University of Economics and Technology* [⊙]*University of Toronto*

In-DRAM True Random Number Generation

- F. Nisa Bostanci, Ataberk Olgun, Lois Orosa, A. Giray Yaglikci, Jeremie S. Kim, Hasan Hassan, Oguz Ergin, and Onur Mutlu,
  "DR-STRaNGe: End-to-End System Design for DRAM-based True Random Number Generators"
  Proceedings of the 28th International Symposium on High-Performance Computer Architecture (***HPCA***), Virtual, April 2022.
  [Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]

DR-STRaNGe: End-to-End System Design for DRAM-based True Random Number Generators

F. Nisa Bostancı[†§] Ataberk Olgun[†§] Lois Orosa[§] A. Giray Yağlıkçı[§]
Jeremie S. Kim[§] Hasan Hassan[§] Oğuz Ergin[†] Onur Mutlu[§]

[†] *TOBB University of Economics and Technology* [§] *ETH Zürich*

https://arxiv.org/pdf/2201.01385.pdf

In-DRAM Physical Unclonable Functions

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu,
  "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern DRAM Devices"
  *Proceedings of the 24th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Vienna, Austria, February 2018.
  [Lightning Talk Video]
  [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)]
  [Full Talk Lecture Video (28 minutes)]

The DRAM Latency PUF:

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

Jeremie S. Kim[†§] Minesh Patel[§] Hasan Hassan[§] Onur Mutlu[§†]

[†]Carnegie Mellon University [§]ETH Zürich

In-DRAM Lookup-Table Based Execution

João Dinis Ferreira, Gabriel Falcao, Juan Gómez-Luna, Mohammed Alser, Lois Orosa, Mohammad Sadrosadati, Jeremie S. Kim, Geraldo F. Oliveira, Taha Shahroodi, Anant Nori, and Onur Mutlu,
"pLUTo: Enabling Massively Parallel Computation in DRAM via Lookup Tables"
*Proceedings of the 55th International Symposium on Microarchitecture (**MICRO**)*, Chicago, IL, USA, October 2022.
[Slides (pptx) (pdf)]
[Longer Lecture Slides (pptx) (pdf)]
[Lecture Video (26 minutes)]
[arXiv version]
[Source Code (Officially Artifact Evaluated with All Badges)]
Officially artifact evaluated as available, reusable and reproducible.

pLUTo: Enabling Massively Parallel Computation in DRAM via Lookup Tables

João Dinis Ferreira[§] Gabriel Falcao[†] Juan Gómez-Luna[§] Mohammed Alser[§]
Lois Orosa[§▽] Mohammad Sadrosadati[§] Jeremie S. Kim[§] Geraldo F. Oliveira[§]
Taha Shahroodi[‡] Anant Nori[⋆] Onur Mutlu[§]
[§]ETH Zürich [†]IT, University of Coimbra [▽]Galicia Supercomputing Center [‡]TU Delft [⋆]Intel

https://arxiv.org/pdf/2104.07699.pdf

What About Other Types of Memories?

In-Flash Bulk Bitwise Execution

- Jisung Park, Roknoddin Azizi, Geraldo F. Oliveira, Mohammad Sadrosadati, Rakesh Nadig, David Novo, Juan Gómez-Luna, Myungsuk Kim, and Onur Mutlu,
"Flash-Cosmos: In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory"
Proceedings of the 55th International Symposium on Microarchitecture (***MICRO***), Chicago, IL, USA, October 2022.
[Slides (pptx) (pdf)]
[Longer Lecture Slides (pptx) (pdf)]
[Lecture Video (44 minutes)]
[arXiv version]

Flash-Cosmos: In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory

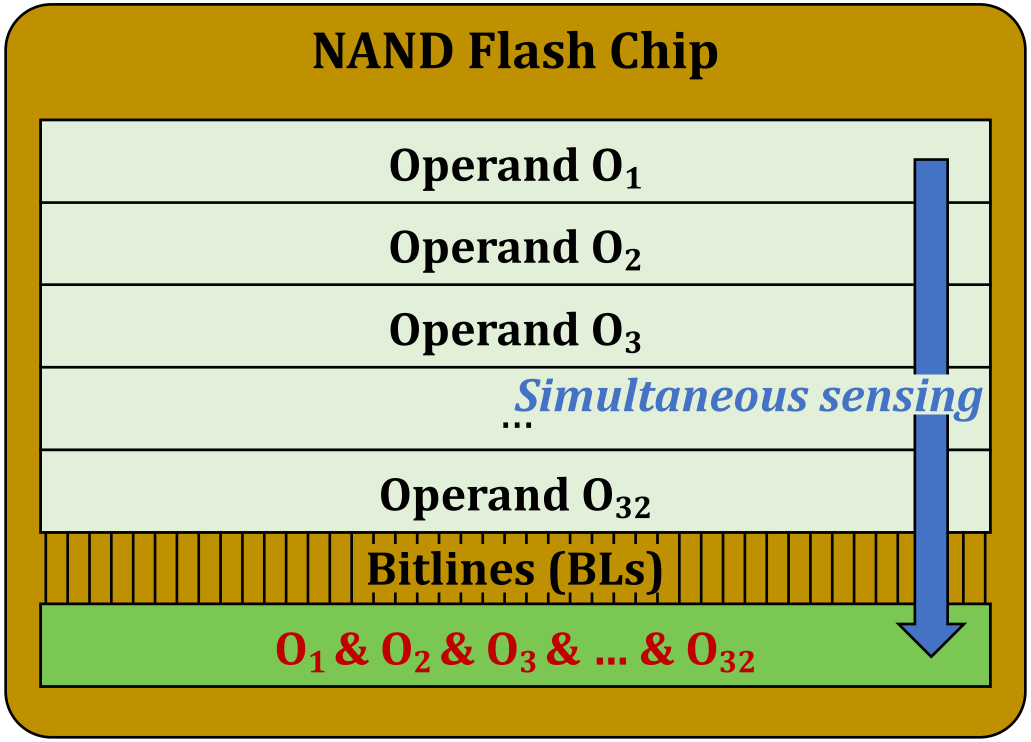
Jisung Park[§∇] Roknoddin Azizi[§] Geraldo F. Oliveira[§] Mohammad Sadrosadati[§]
Rakesh Nadig[§] David Novo[†] Juan Gómez-Luna[§] Myungsuk Kim[‡] Onur Mutlu[§]

[§]*ETH Zürich* [∇]*POSTECH* [†]*LIRMM, Univ. Montpellier, CNRS* [‡]*Kyungpook National University*

https://arxiv.org/pdf/2209.05566.pdf

Flash-Cosmos: Basic Ideas

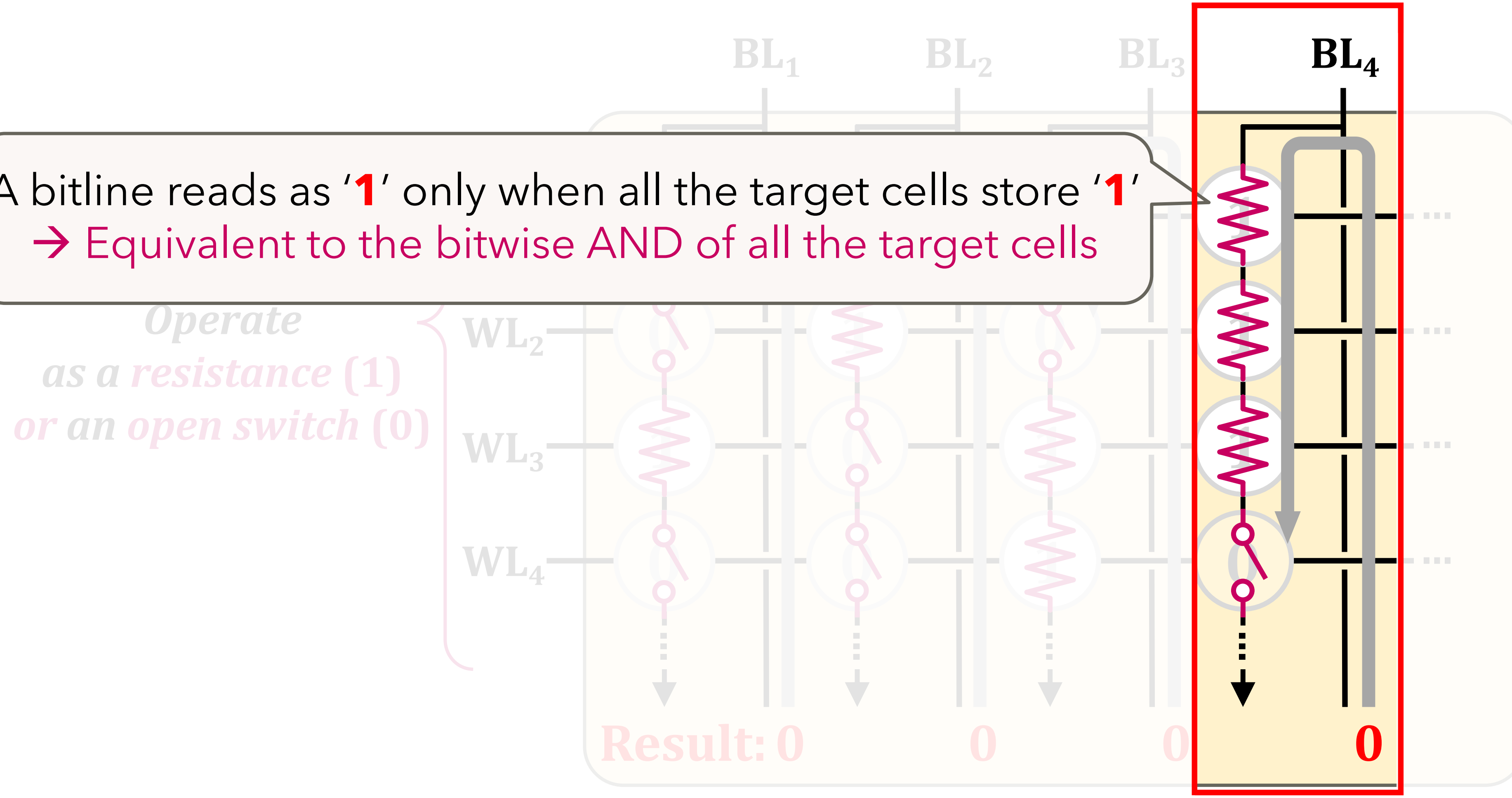
- Flash-Cosmos enables
  - Computation on multiple operands with a single sensing operation
  - Accurate computation results by eliminating raw bit errors in stored data



Multi-Wordline Sensing (MWS): Bitwise AND

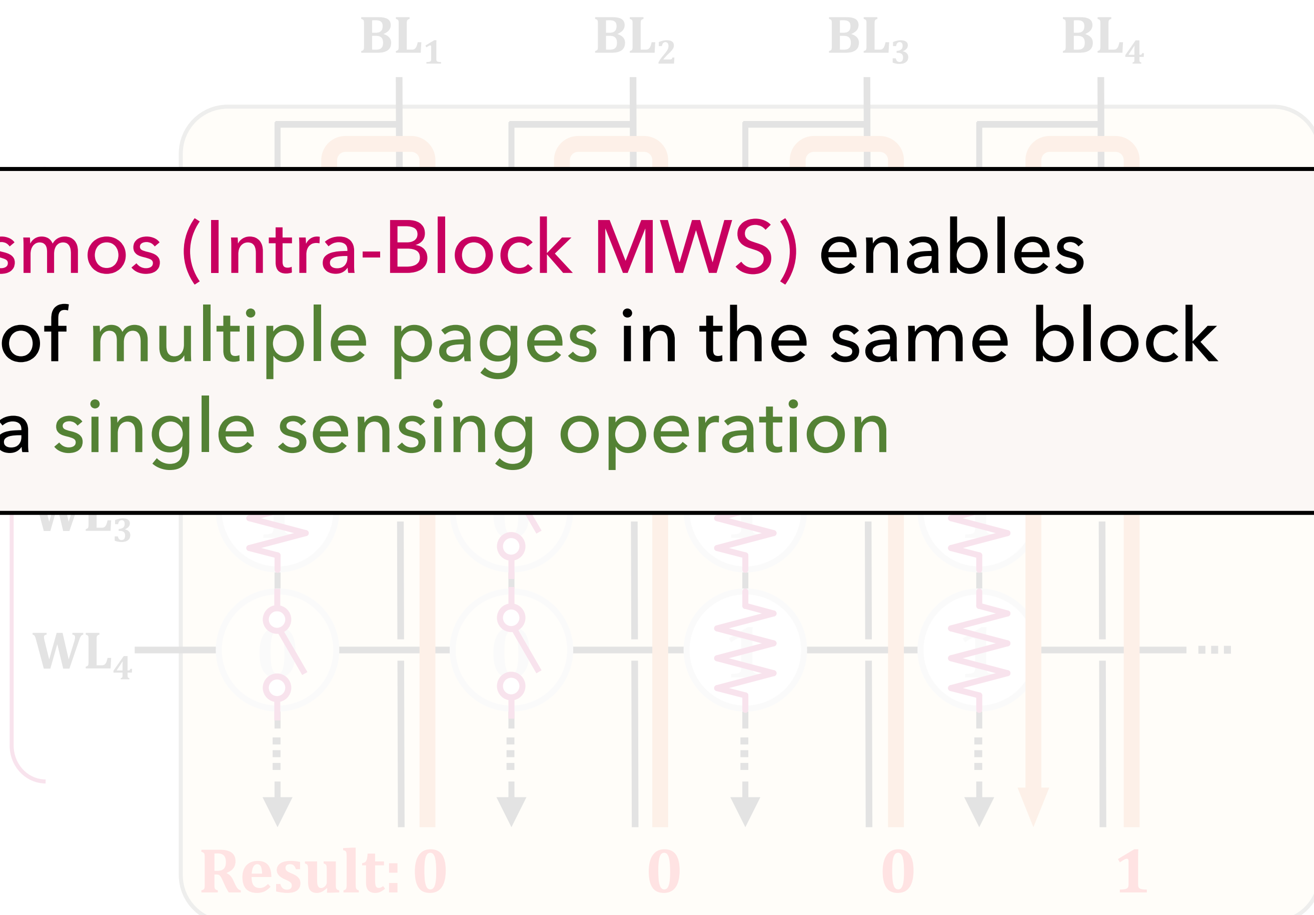
- Intra-Block MWS:
  Simultaneously activates multiple WLs in the same block
  → Bitwise AND of the stored data in the WLs

A bitline reads as '**1**' only when all the target cells store '**1**'
→ Equivalent to the bitwise AND of all the target cells



Multi-Wordline Sensing (MWS): Bitwise AND

- Intra-Block MWS:
  Simultaneously activates multiple WLs in the same block
  → Bitwise AND of the stored data in the WLs

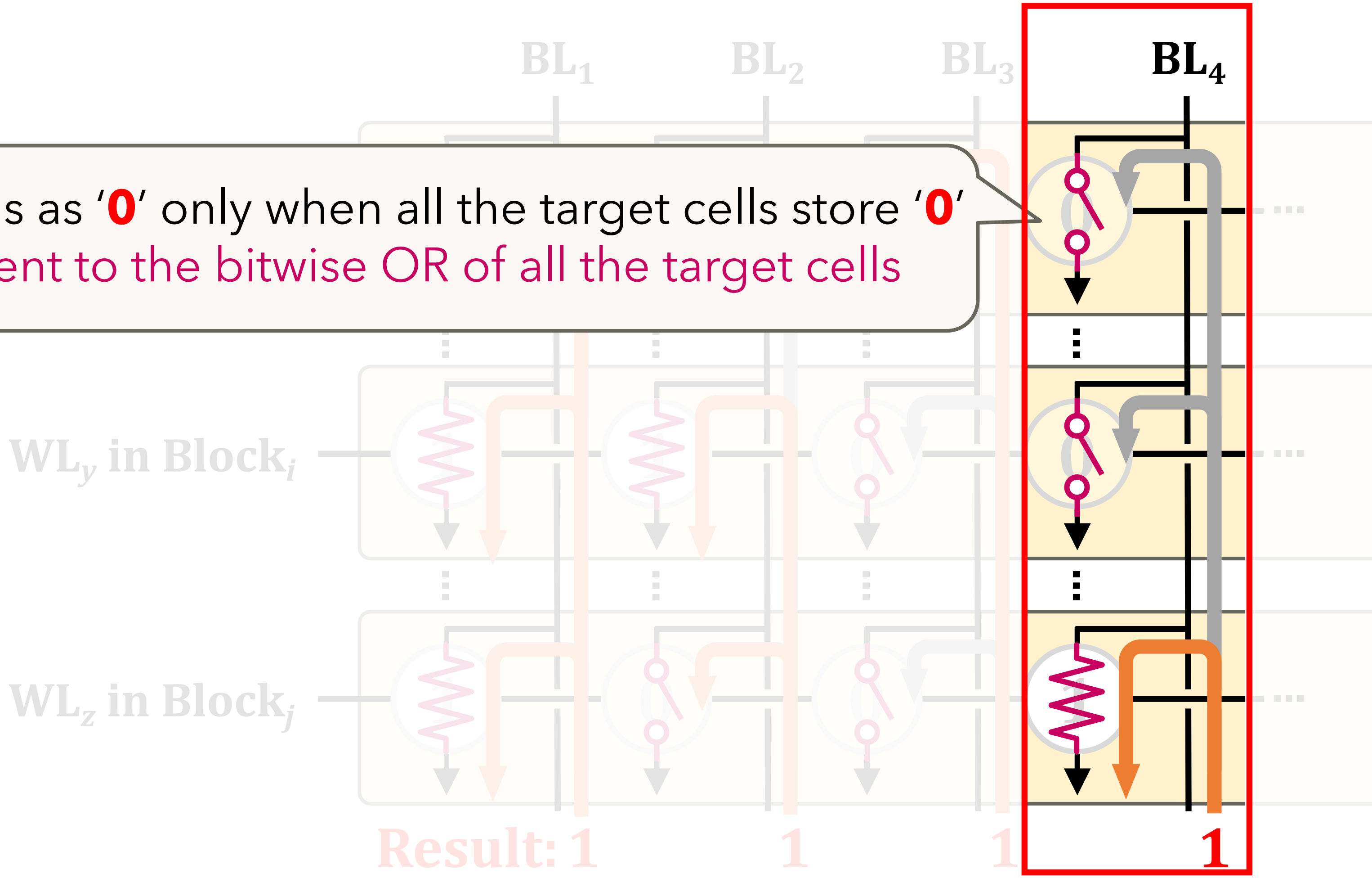


Flash-Cosmos (Intra-Block MWS) enables bitwise AND of multiple pages in the same block via a single sensing operation

Multi-Wordline Sensing (MWS): Bitwise OR

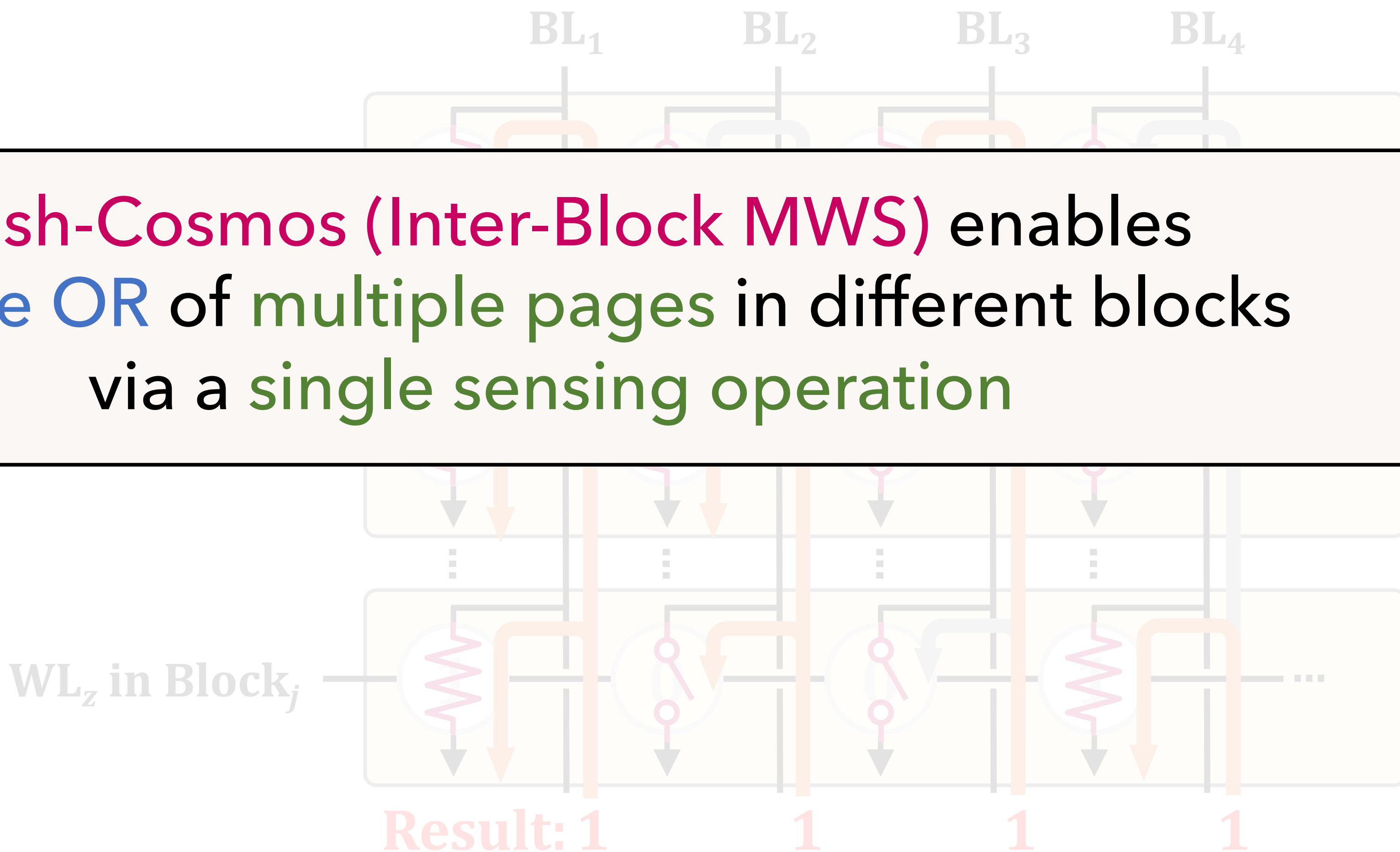
- **Inter-Block MWS**:
  Simultaneously activates multiple WLs in different blocks
  → Bitwise OR of the stored data in the WLs

A bitline reads as '**0**' only when all the target cells store '**0**'
→ Equivalent to the bitwise OR of all the target cells



Multi-Wordline Sensing (MWS): Bitwise OR

- Inter-Block MWS:
  Simultaneously activates multiple WLs in different blocks
  → Bitwise OR of the stored data in the WLs



Flash-Cosmos (Inter-Block MWS) enables bitwise OR of multiple pages in different blocks via a single sensing operation

Other Types of Bitwise Operations

Flash-Cosmos also enables
other types of bitwise operations
(NOT/NAND/NOR/XOR/XNOR)
leveraging existing features of NAND flash memory

Flash-Cosmos: In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory

Jisung Park[§∇] Roknoddin Azizi[§] Geraldo F. Oliveira[§] Mohammad Sadrosadati[§]
Rakesh Nadig[§] David Novo[†] Juan Gómez-Luna[§] Myungsuk Kim[‡] Onur Mutlu[§]

[§]*ETH Zürich* [∇]*POSTECH* [†]*LIRMM, Univ. Montpellier, CNRS* [‡]*Kyungpook National University*

https://arxiv.org/abs/2209.05566.pdf

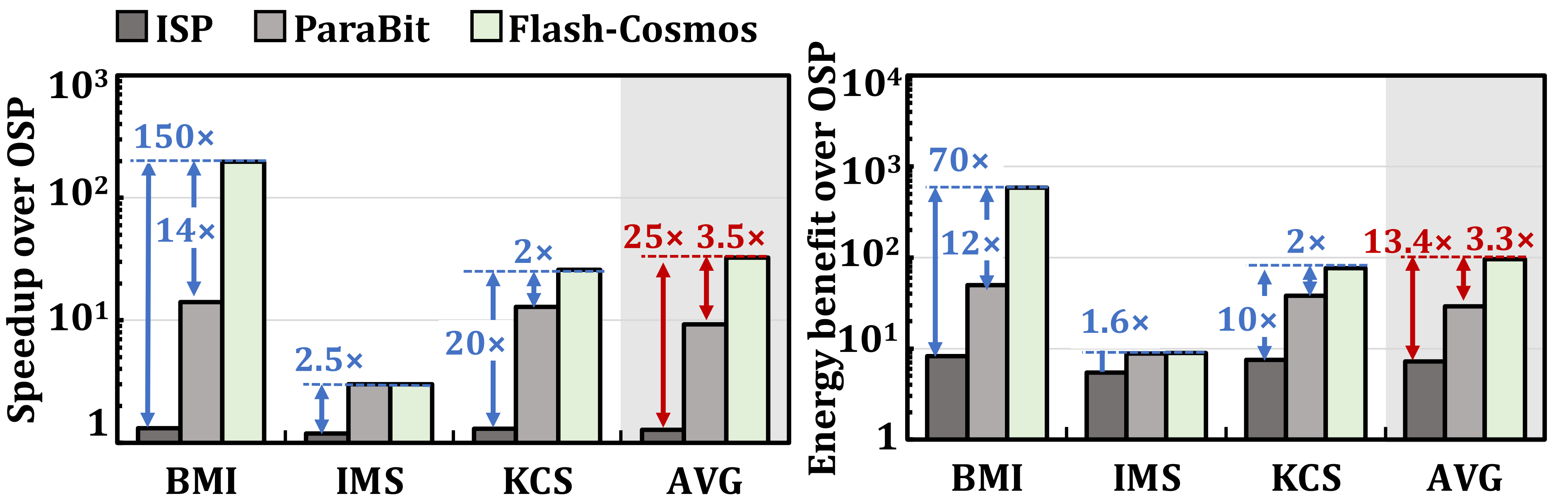
Results: Real-Device Characterization

No changes to the cell array
of commodity NAND flash chips

Can have many operands
(AND: up to 48, OR: up to 4)
with small increase in sensing latency (< 10%)

ESP significantly improves
the reliability of computation results
(no observed bit error in the tested flash cells)

Results: Performance & Energy



Flash-Cosmos provides significant performance & energy benefits over all the baselines

The larger the number of operands, the higher the performance & energy benefits

Pinatubo: RowClone and Bitwise Ops in PCM

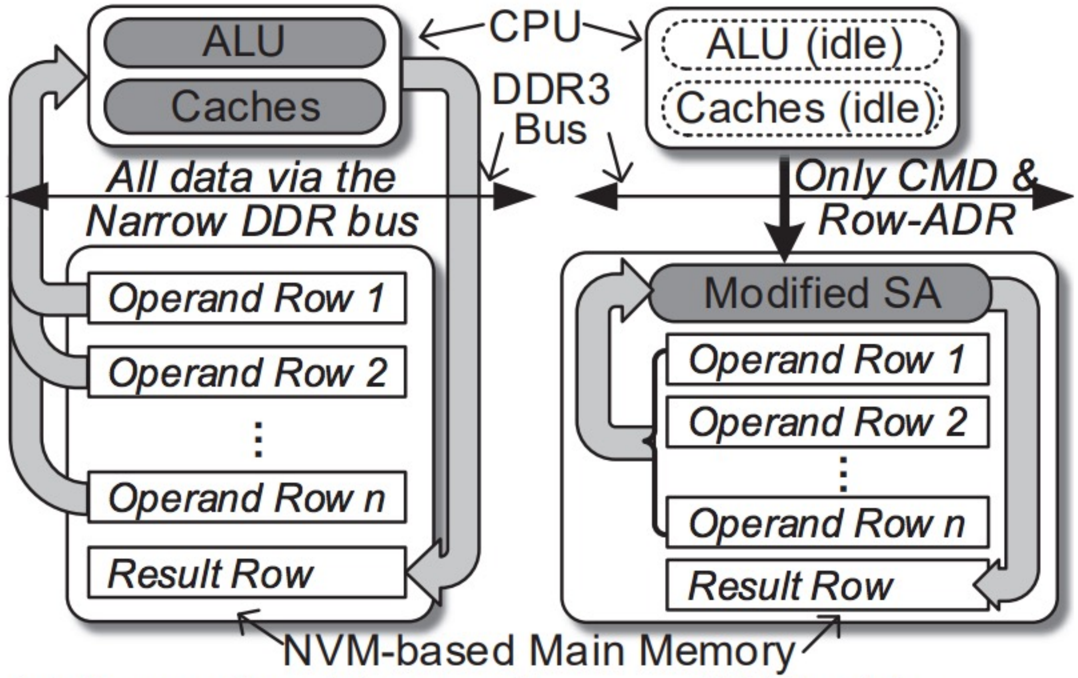
Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories

Shuangchen Li[1,*], Cong Xu[2], Qiaosha Zou[1,5], Jishen Zhao[3], Yu Lu[4], and Yuan Xie[1]

University of California, Santa Barbara[1], Hewlett Packard Labs[2]
University of California, Santa Cruz[3], Qualcomm Inc.[4], Huawei Technologies Inc.[5]
{shuangchenli, yuanxie}ece.ucsb.edu[1]

https://cseweb.ucsd.edu/~jzhao/files/Pinatubo-dac2016.pdf

Pinatubo: RowClone and Bitwise Ops in PCM



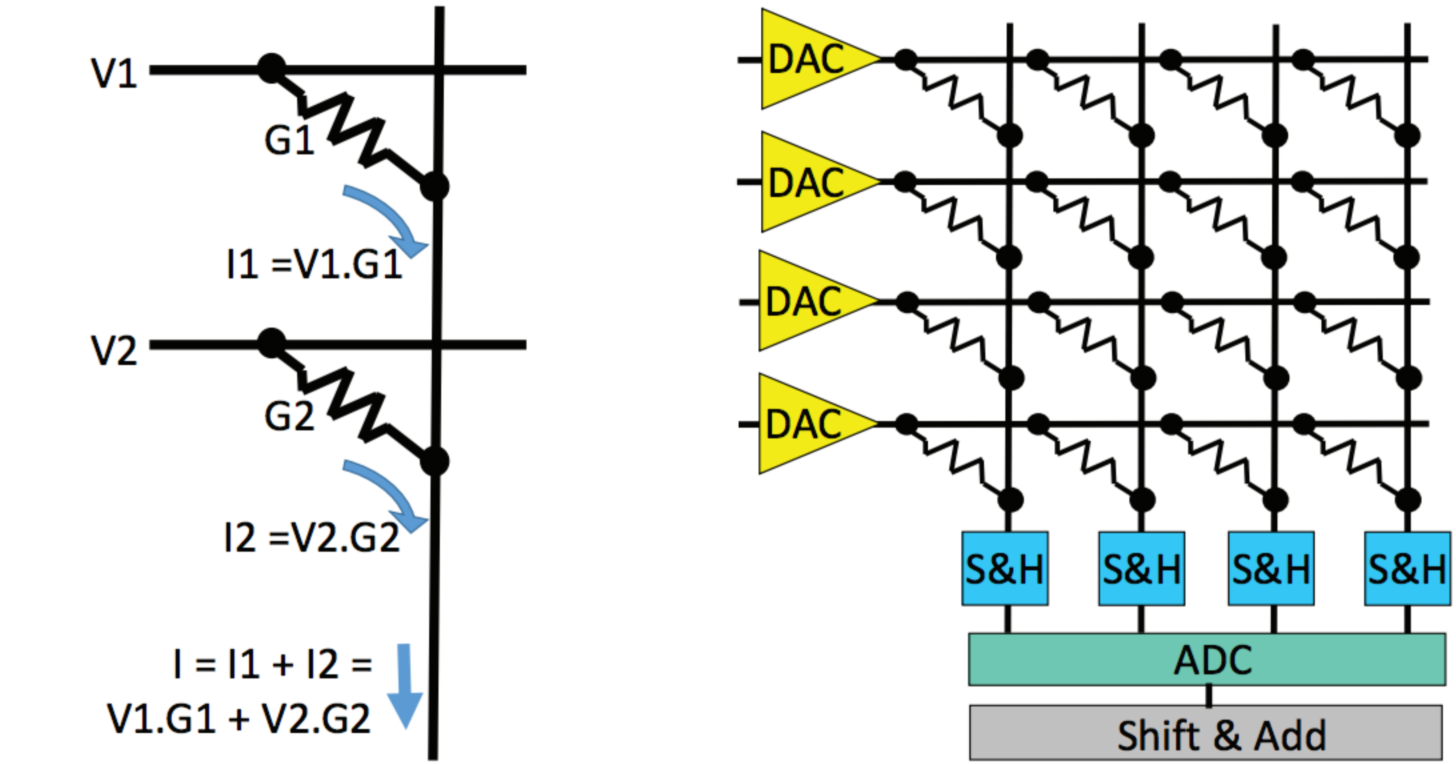
(a) Conventional Approach (b) Pinatubo

Figure 2: Overview: (a) Computing-centric approach, moving tons of data to CPU and write back. (b) The proposed Pinatubo architecture, performs n-row bitwise operations inside NVM in one step.

In-Memory Crossbar Array Operations

- Some emerging NVM technologies have crossbar array structure
  - Memristors, resistive RAM, phase change mem, STT-MRAM, …
- Crossbar arrays can be used to perform dot product operations using "analog computation capability"
  - Can operate on multiple pieces of data using Kirchoff's laws
    - Bitline current is a sum of products of wordline V x (1 / cell R)
  - Computation is in analog domain inside the crossbar array
- Need peripheral circuitry for D→A and A→D conversion of inputs and outputs

In-Memory Crossbar Computation

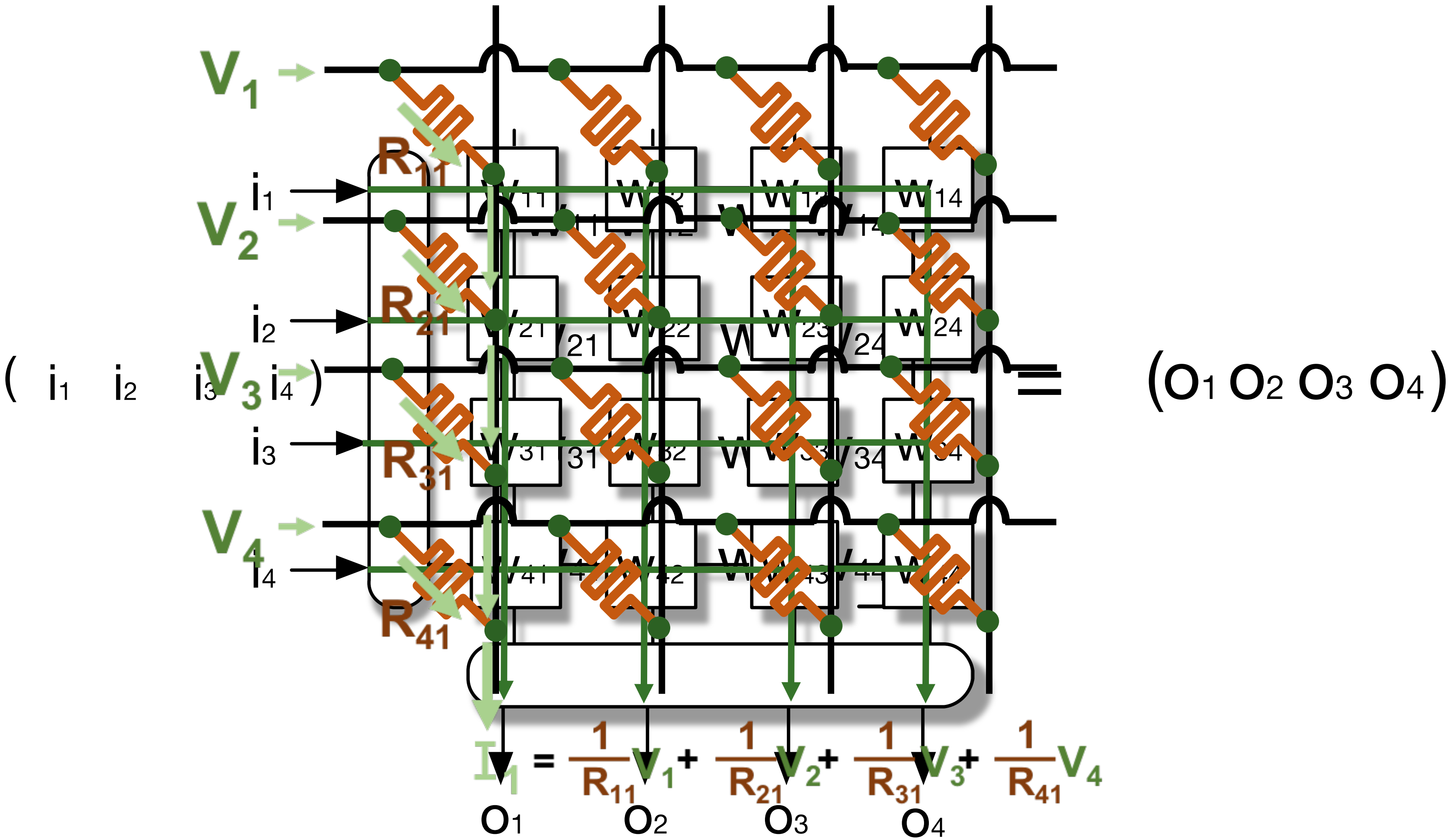


(a) Multiply-Accumulate operation

(b) Vector-Matrix Multiplier

Fig. 1. (a) Using a bitline to perform an analog sum of products operation. (b) A memristor crossbar used as a vector-matrix multiplier.

In-Memory Crossbar Computation



Other Readings on Processing using NVM

- Shafiee+, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars", ISCA 2016.

- Chi+, "PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory", ISCA 2016.

- Prezioso+, "Training and Operation of an Integrated Neuromorphic Network based on Metal-Oxide Memristors", Nature 2015

- Ambrogio+, "Equivalent-accuracy accelerated neural-network training using analogue memory", Nature 2018.

Processing in Memory: Two Approaches

1. Processing using Memory

2. Processing near Memory

A Modern Primer on Processing in Memory

Onur Mutlu[a,b], Saugata Ghose[b,c], Juan Gómez-Luna[a], Rachata Ausavarungnirun[d]

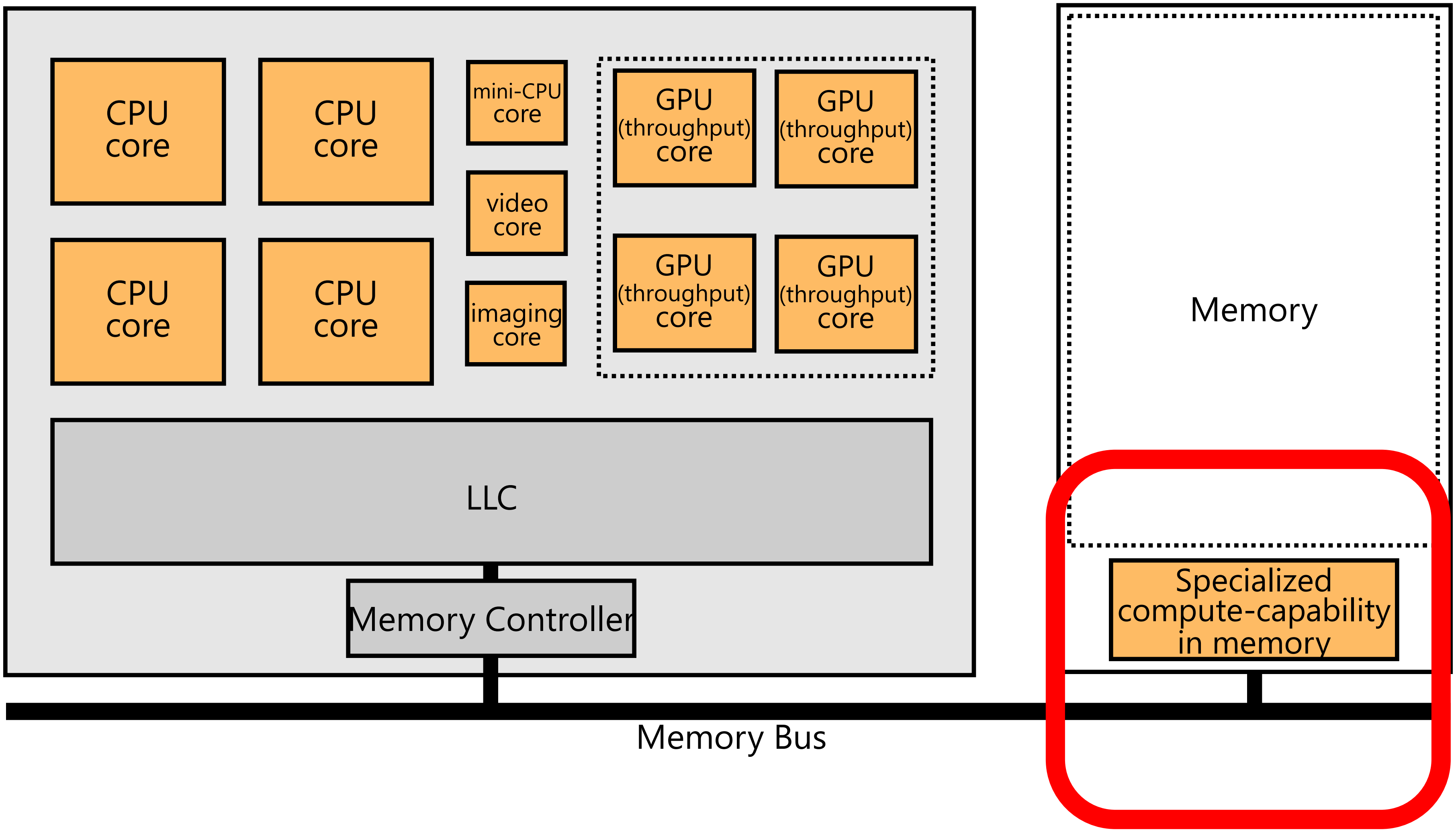
SAFARI Research Group

[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.

https://arxiv.org/pdf/2012.03112.pdf

Mindset: Memory as an Accelerator



Memory similar to a "conventional" accelerator

Accelerating In-Memory Graph Analytics

- Large graphs are everywhere (circa 2015)

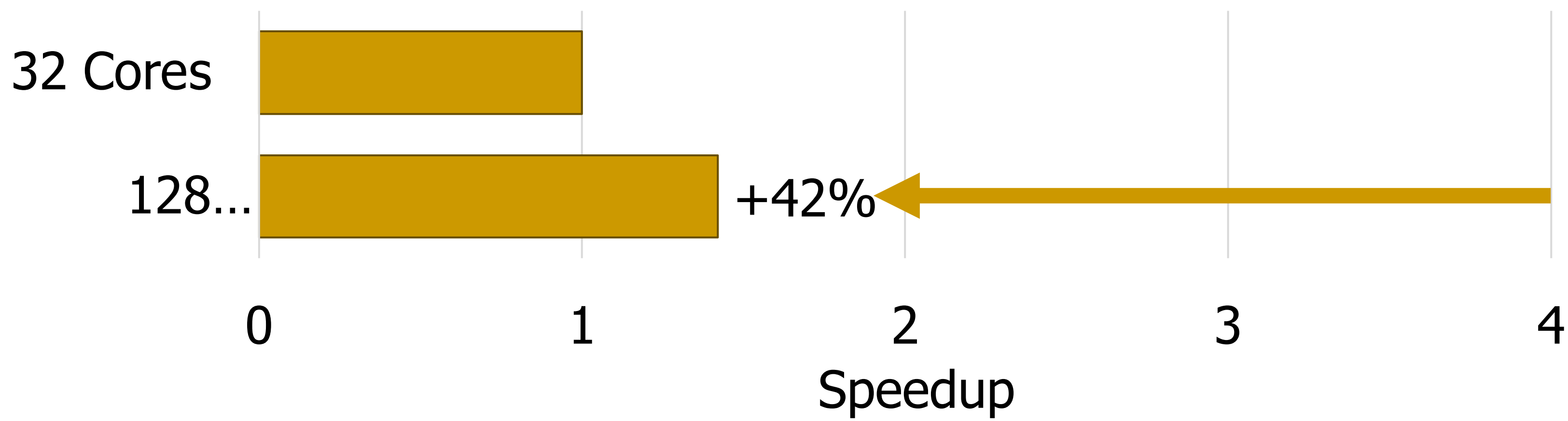
36 Million
Wikipedia Pages

1.4 Billion
Facebook Users

300 Million
Twitter Users

30 Billion
Instagram Photos

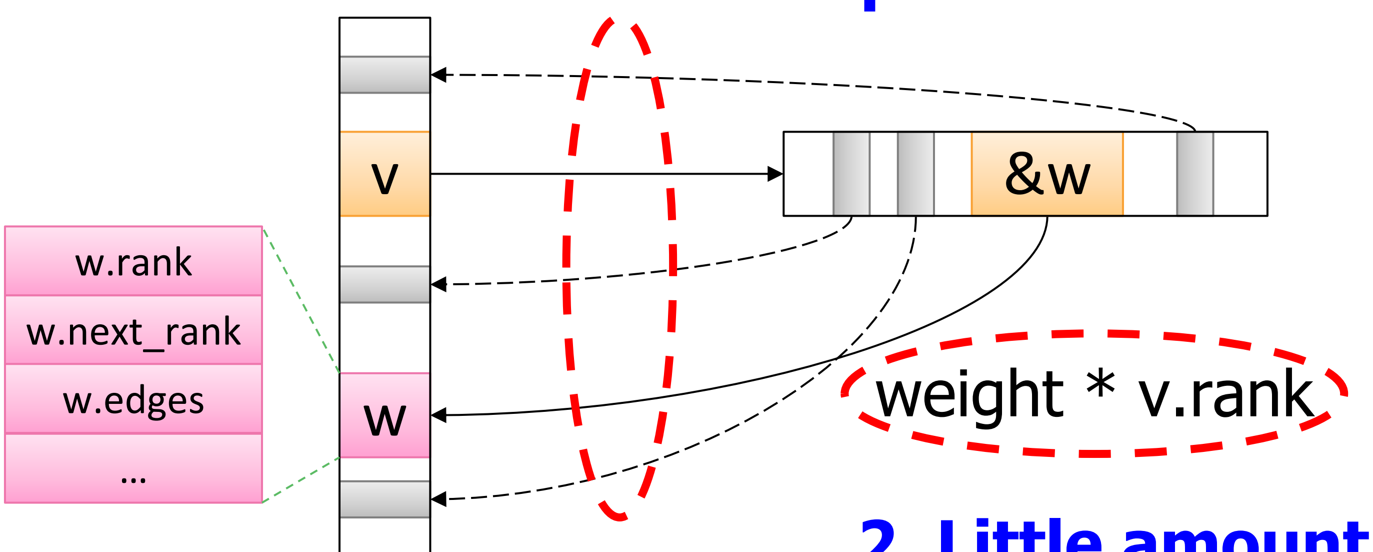
- Scalable large-scale graph processing is challenging



Key Bottlenecks in Graph Processing

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

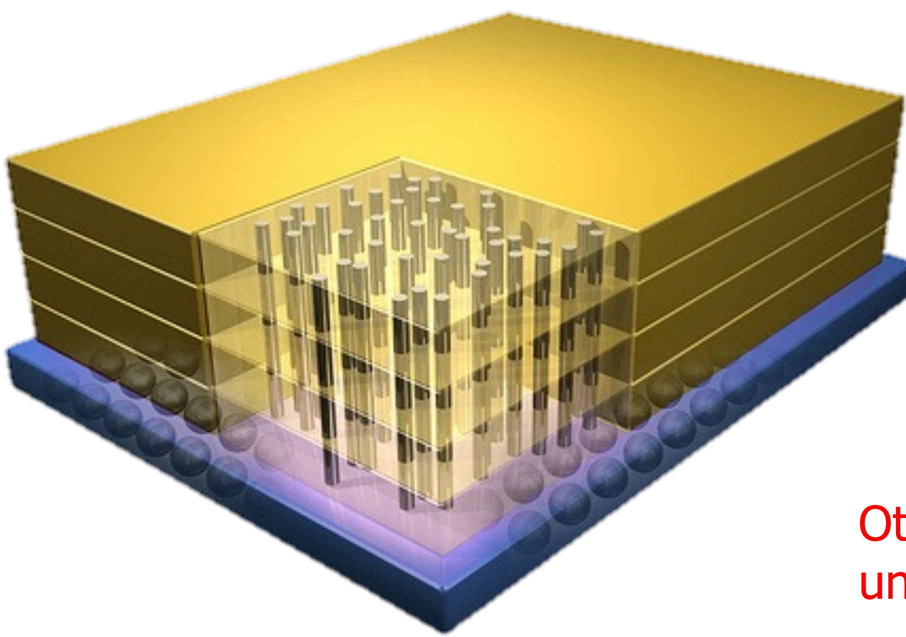
1. Frequent random memory accesses



2. Little amount of computation

Opportunity: 3D-Stacked Logic+Memory

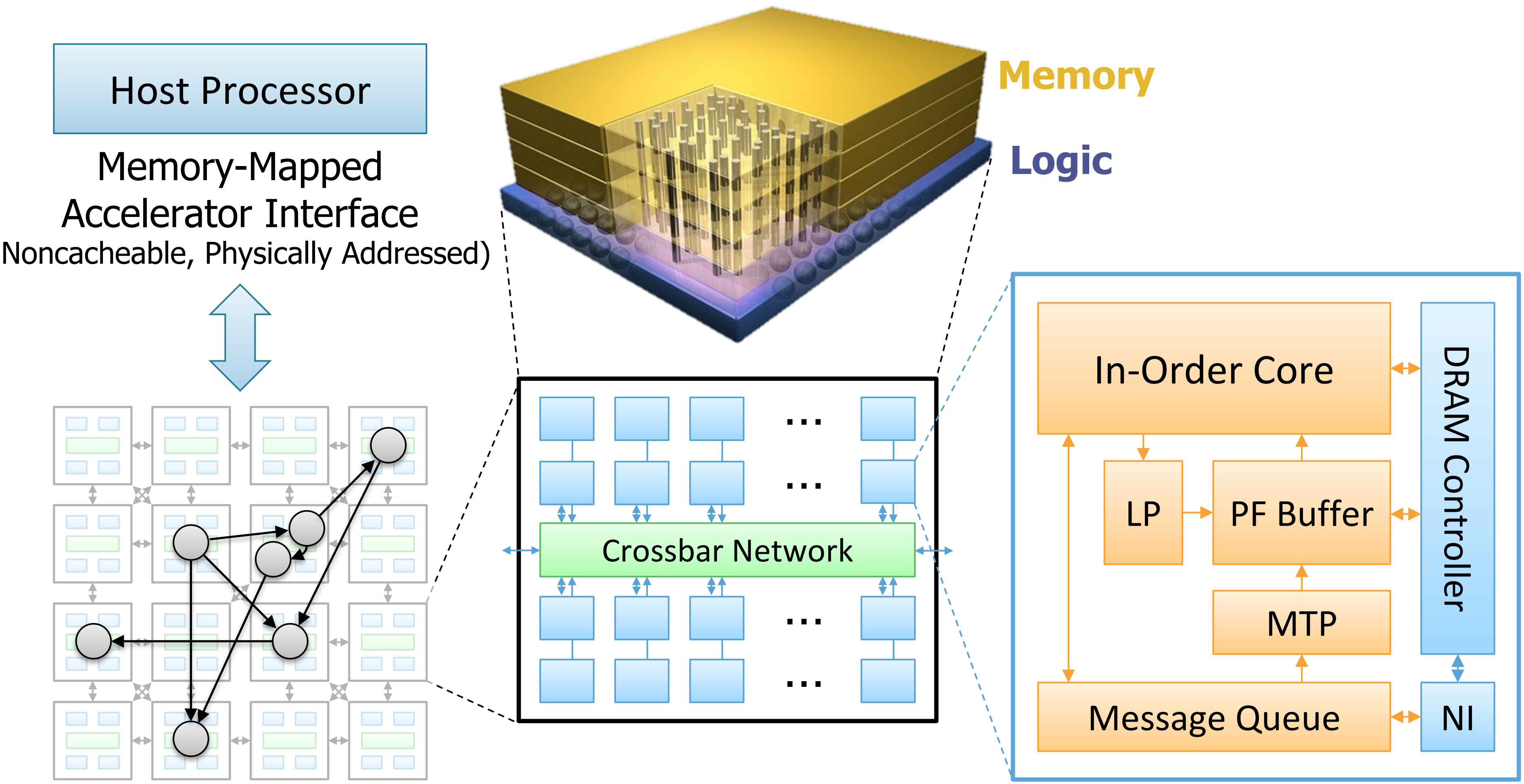
Hybrid Memory Cube
CONSORTIUM



Other "True 3D" technologies under development

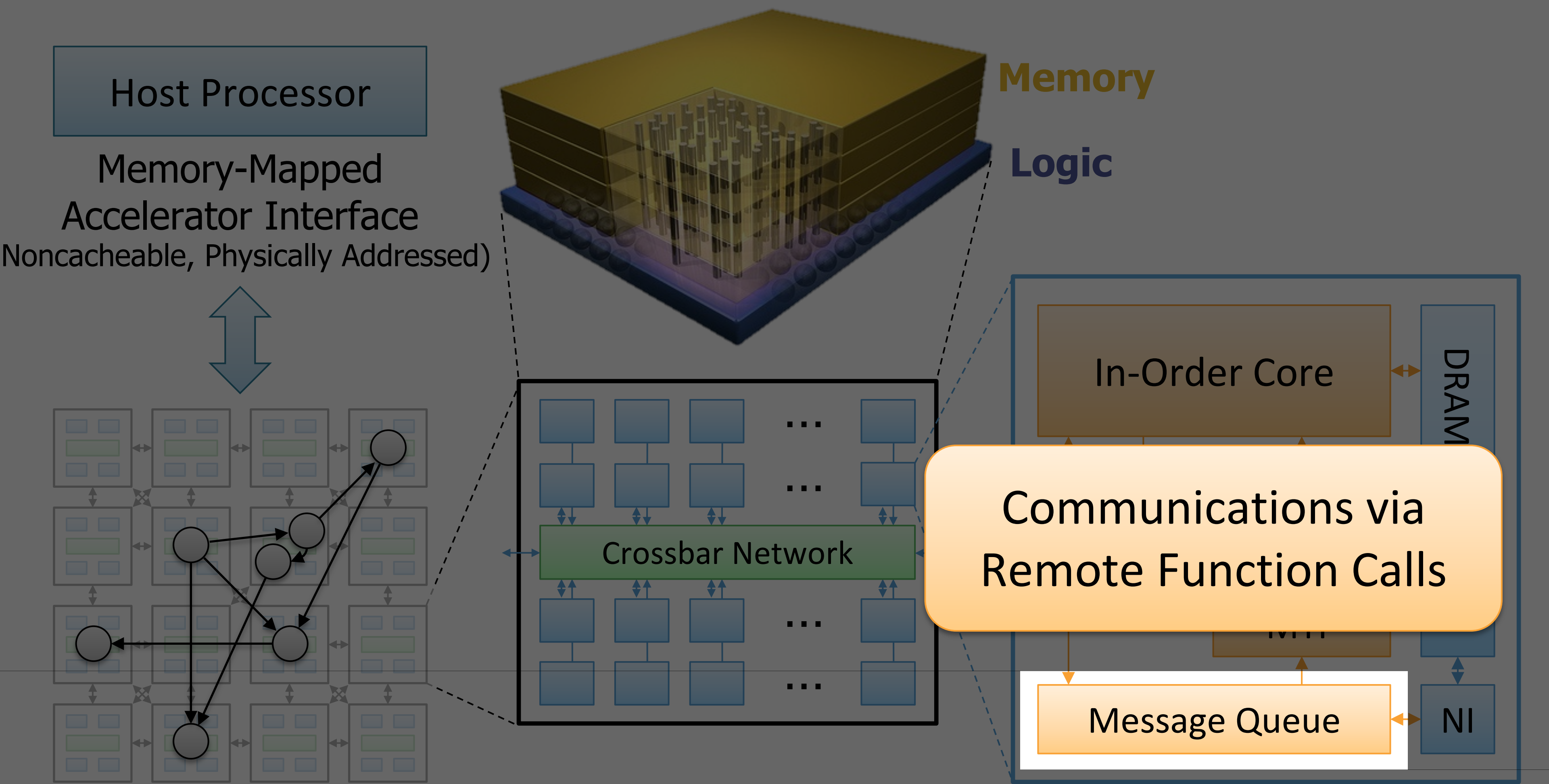
Tesseract System for Graph Processing

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



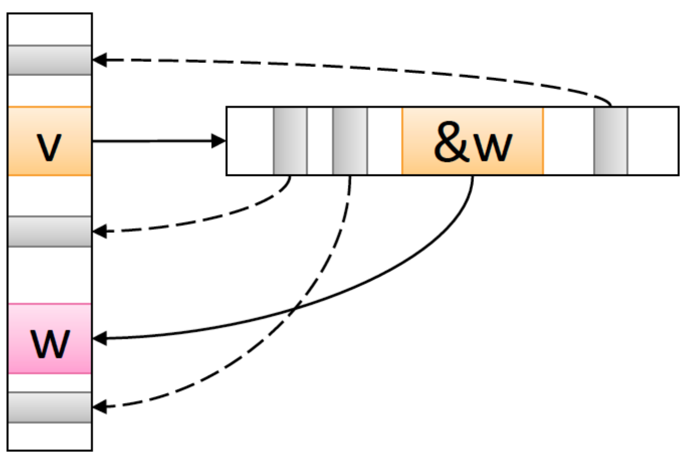
 Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

Tesseract System for Graph Processing



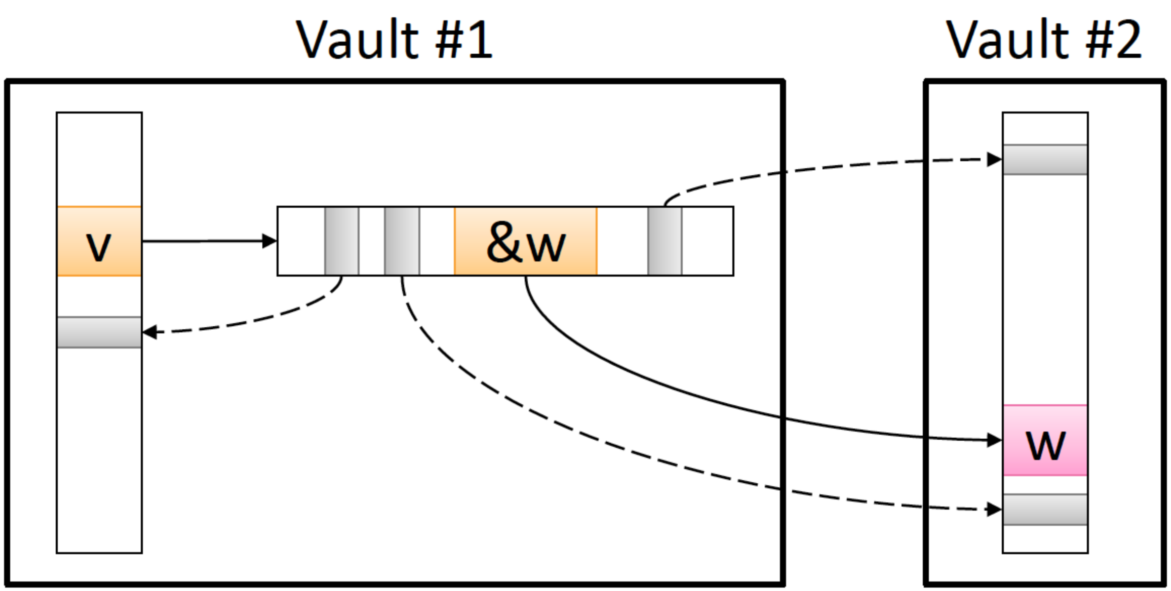
Communications In Tesseract (I)

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



Communications In Tesseract (II)

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

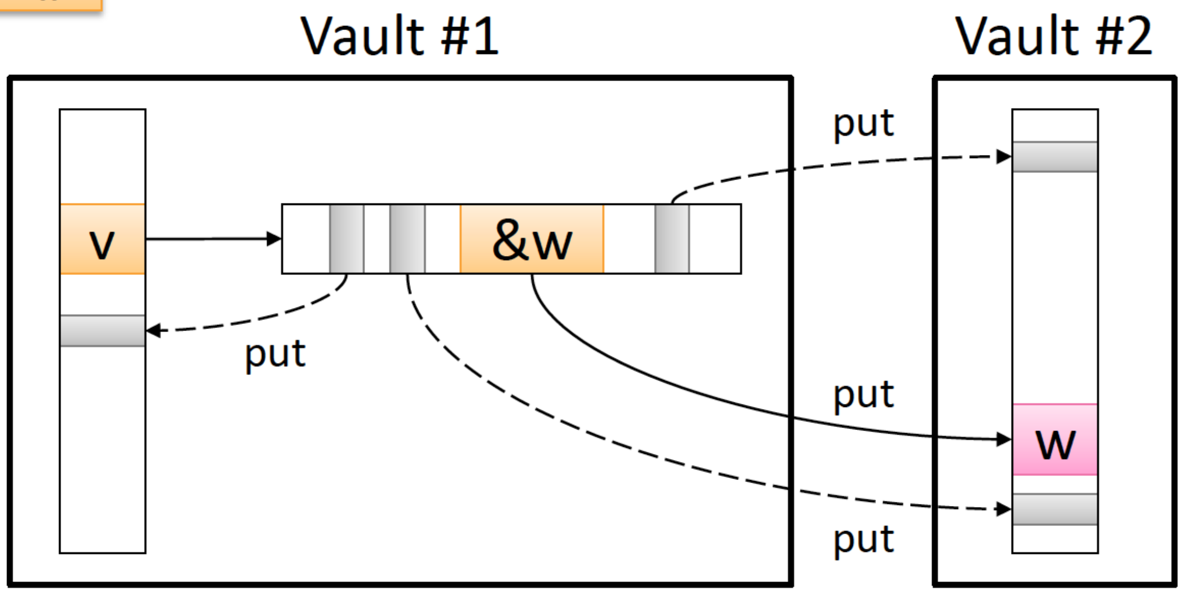


Communications In Tesseract (III)

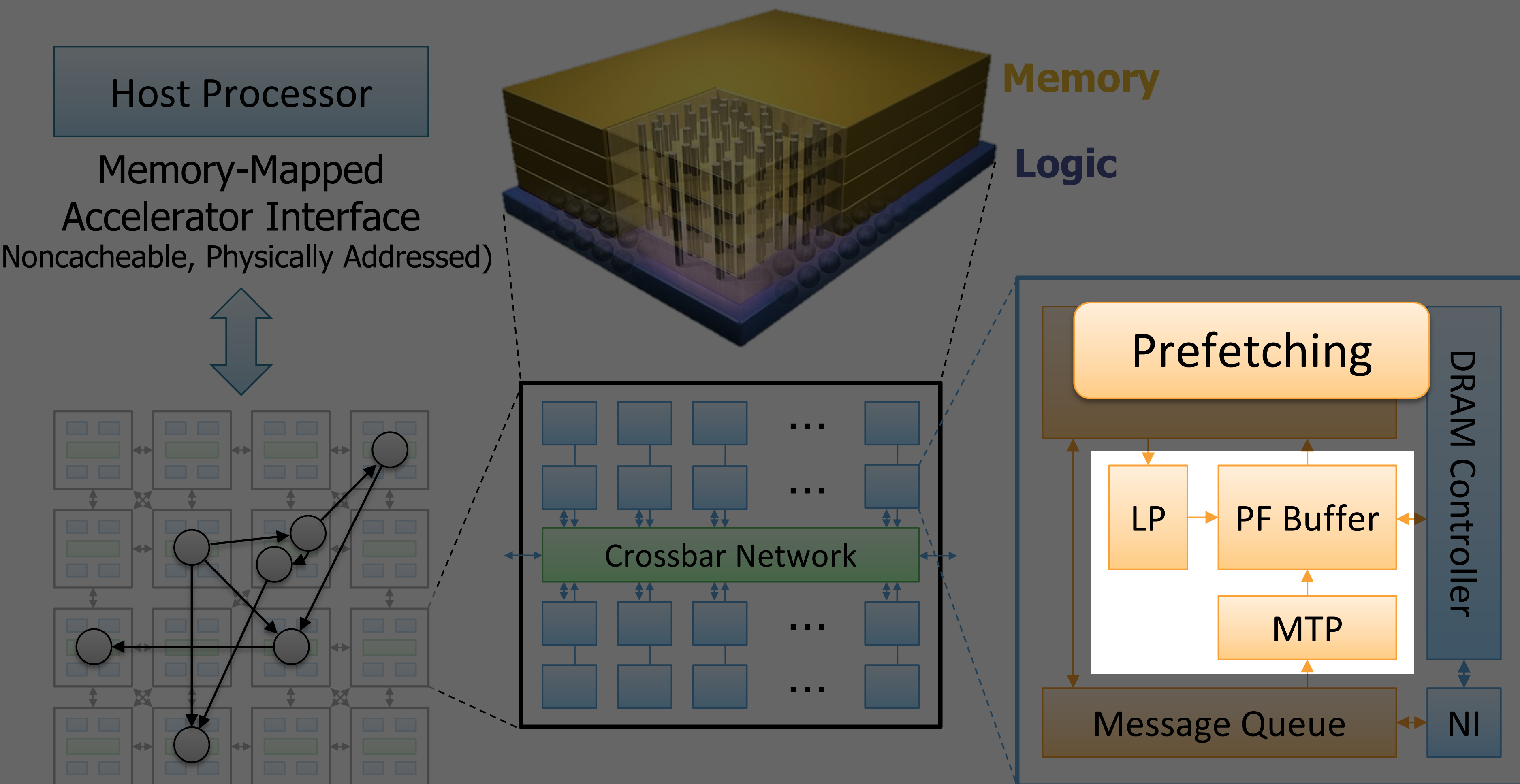
```
for (v: graph.vertices) {
  for (w: v.successors) {
    put(w.id, function() { w.next_rank += weight * v.rank; });
  }
}
barrier();
```

Non-blocking Remote Function Call

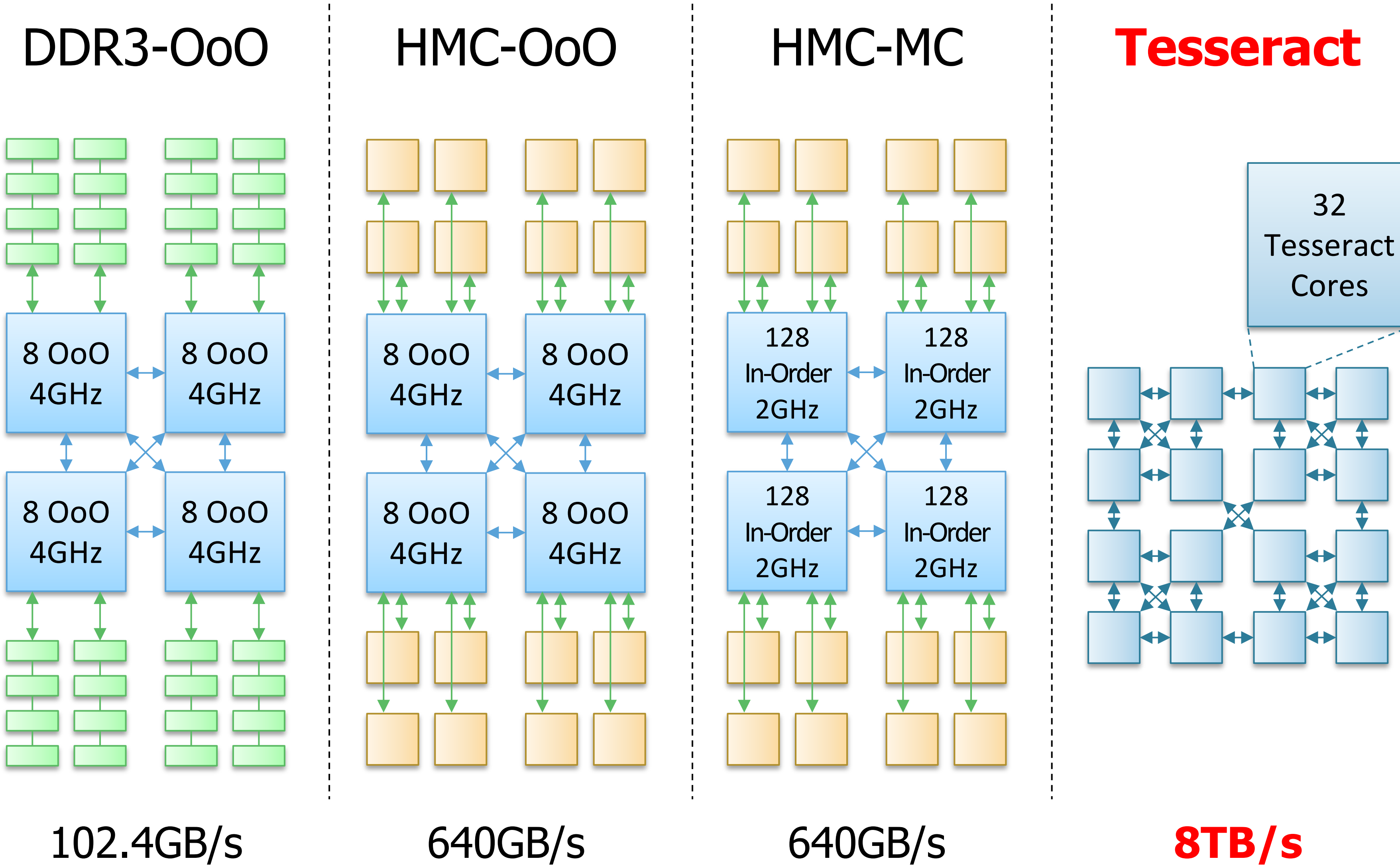
Can be **delayed** until the nearest barrier



Tesseract System for Graph Processing

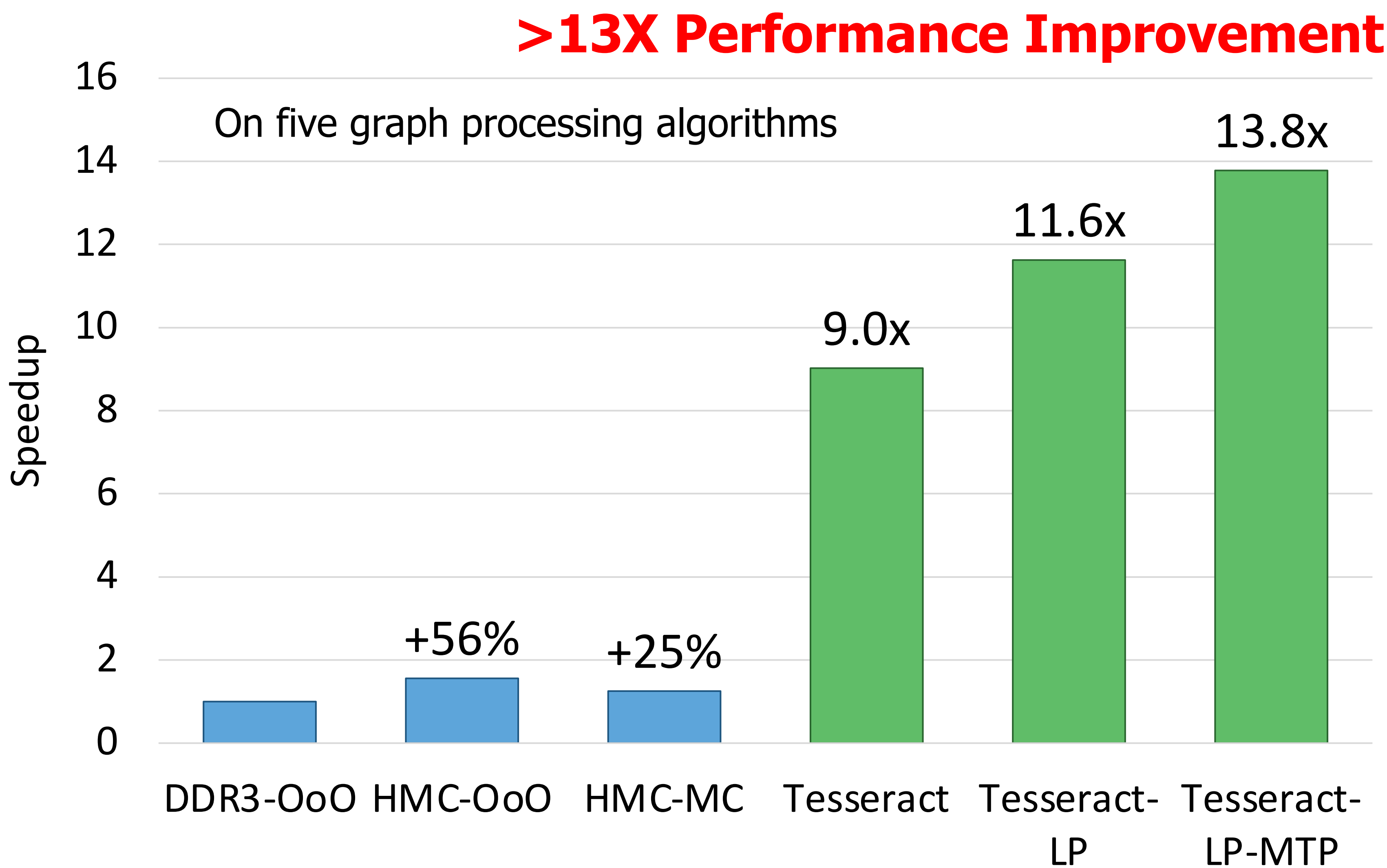


Evaluated Systems



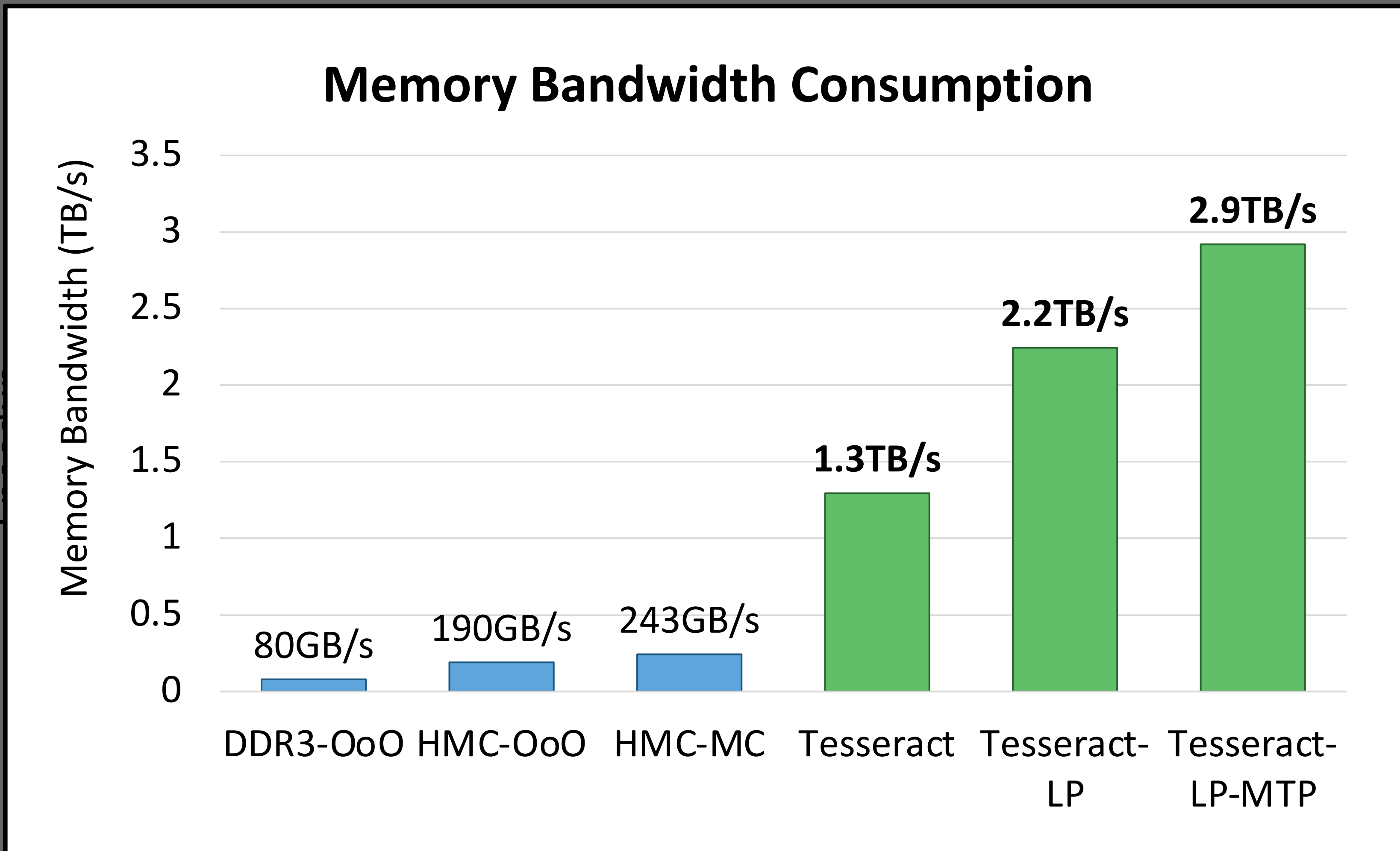
SAFARI Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

Tesseract Graph Processing Performance



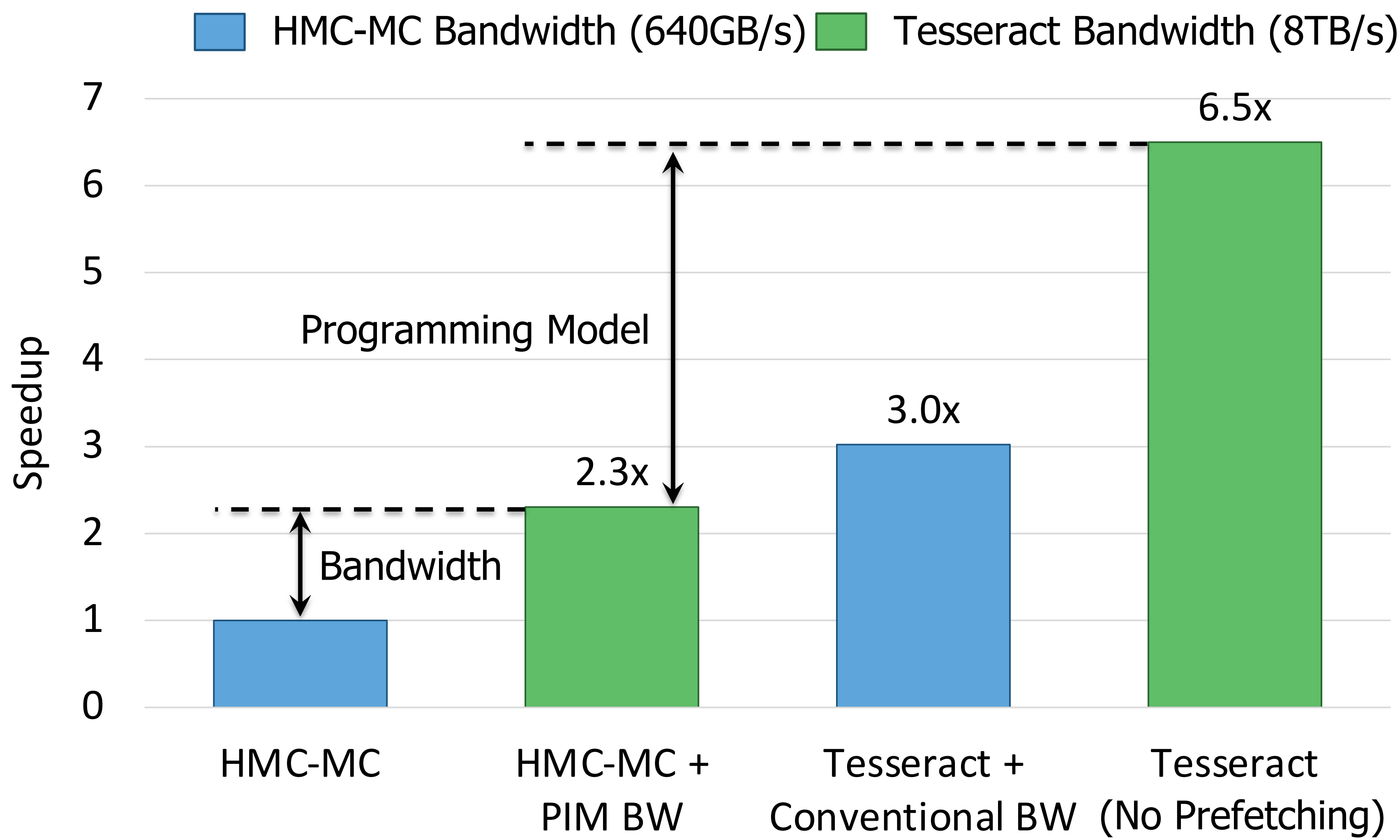
SAFARI Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

Tesseract Graph Processing Performance

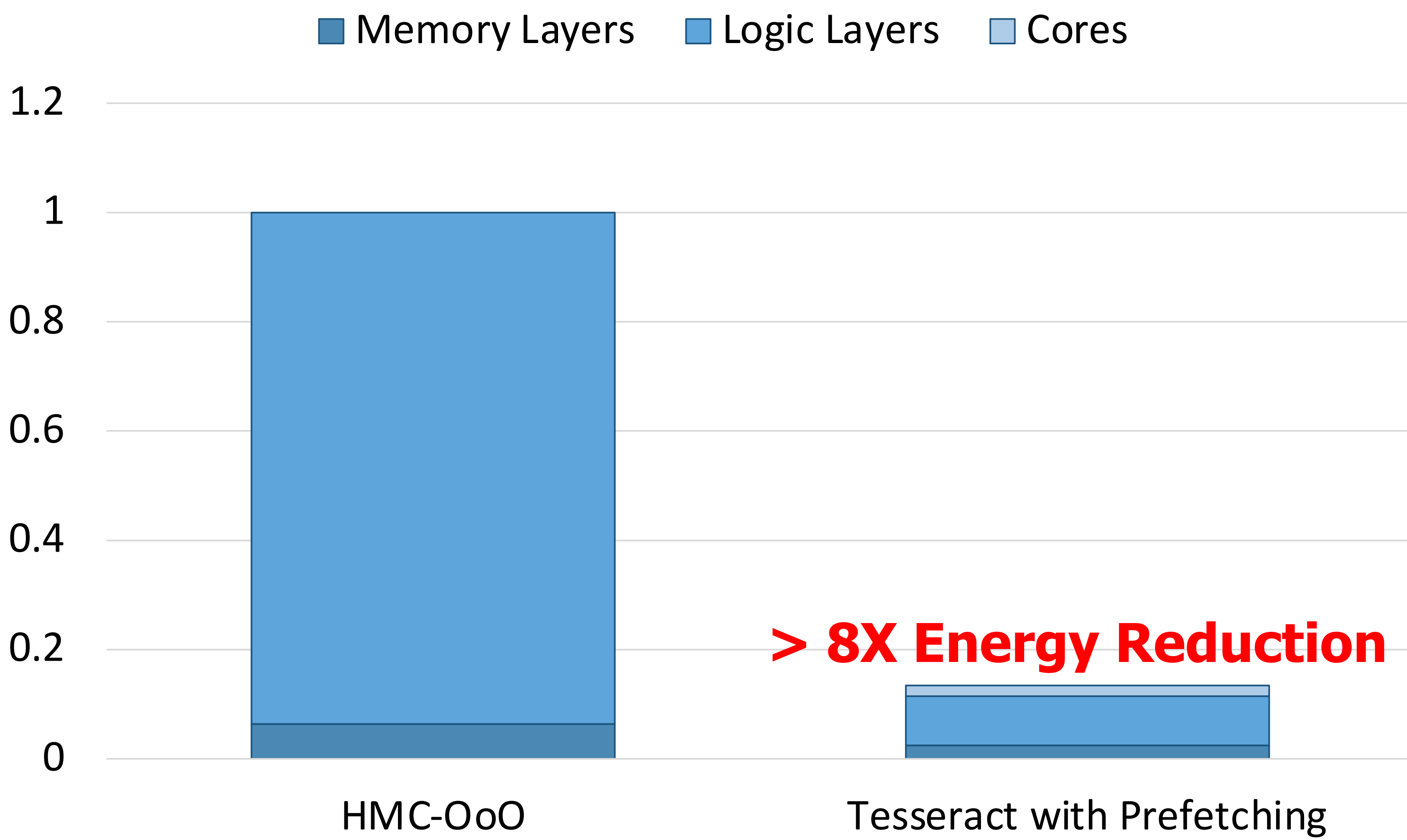


LP
LP-MTP

Effect of Bandwidth & Programming Model



Tesseract Graph Processing System Energy



 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 (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)]
  Top Picks Honorable Mention by IEEE Micro.
  Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).

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

Accelerating Graph Pattern Mining

- Maciej Besta, Raghavendra Kanakagiri, Grzegorz Kwasniewski, Rachata Ausavarungnirun, Jakub Beránek, Konstantinos Kanellopoulos, Kacper Janda, Zur Vonarburg-Shmaria, Lukas Gianinazzi, Ioana Stefan, Juan Gómez-Luna, Marcin Copik, Lukas Kapp-Schwoerer, Salvatore Di Girolamo, Nils Blach, Marek Konieczny, Onur Mutlu, and Torsten Hoefler,
  "SISA: Set-Centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems"
  *Proceedings of the 54th International Symposium on Microarchitecture (**MICRO**)*, Virtual, October 2021.
  [Slides (pdf)]
  [Talk Video (22 minutes)]
  [Lightning Talk Video (1.5 minutes)]
  [Full arXiv version]

SISA: Set-Centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems

Maciej Besta[1], Raghavendra Kanakagiri[2], Grzegorz Kwasniewski[1], Rachata Ausavarungnirun[3], Jakub Beránek[4], Konstantinos Kanellopoulos[1], Kacper Janda[5], Zur Vonarburg-Shmaria[1], Lukas Gianinazzi[1], Ioana Stefan[1], Juan Gómez-Luna[1], Marcin Copik[1], Lukas Kapp-Schwoerer[1], Salvatore Di Girolamo[1], Nils Blach[1], Marek Konieczny[5], Onur Mutlu[1], Torsten Hoefler[1]

[1]ETH Zurich, Switzerland [2]IIT Tirupati, India [3]King Mongkut's University of Technology North Bangkok, Thailand [4]Technical University of Ostrava, Czech Republic [5]AGH-UST, Poland

PIM for Mobile Devices

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu,
  "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"
  *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Williamsburg, VA, USA, March 2018.
  [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
  [Lightning Talk Video (2 minutes)]
  [Full Talk Video (21 minutes)]

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand[1] Saugata Ghose[1] Youngsok Kim[2]
Rachata Ausavarungnirun[1] Eric Shiu[3] Rahul Thakur[3] Daehyun Kim[4,3]
Aki Kuusela[3] Allan Knies[3] Parthasarathy Ranganathan[3] Onur Mutlu[5,1]

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand

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

Consumer Devices

Consumer devices are everywhere!

Energy consumption is a first-class concern in consumer devices

Popular Consumer Workloads

Chrome

Google's web browser

TensorFlow Mobile

Google's machine learning framework

Video Playback

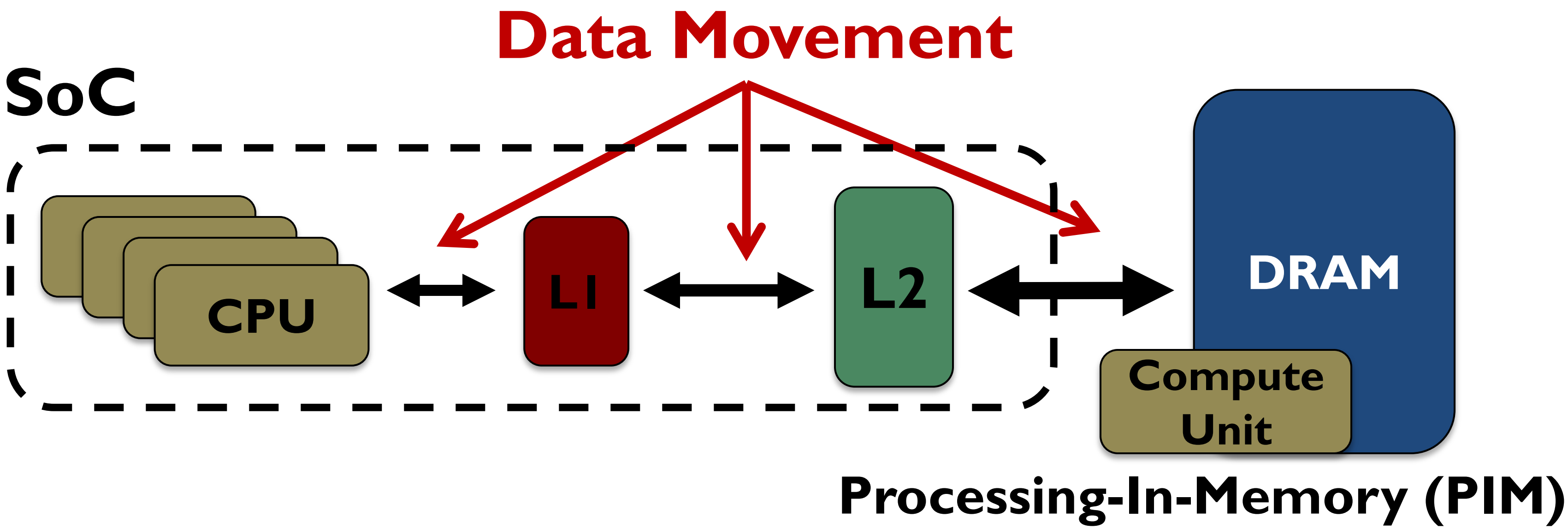
Google's video codec

Video Capture

Google's video codec

Energy Cost of Data Movement

1st key observation: 62.7% of the total system energy is spent on data movement



Potential solution: move computation close to data

Challenge: limited area and energy budget

Using PIM to Reduce Data Movement

2nd key observation: a significant fraction of the data movement often comes from simple functions

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

Small embedded low-power core

PIM Core

Small fixed-function accelerators

PIM Accelerator

Offloading to PIM logic reduces energy and improves performance, on average, by 2.3X and 2.2X

Workload Analysis

Chrome

Google's web browser

TensorFlow Mobile

Google's machine learning framework

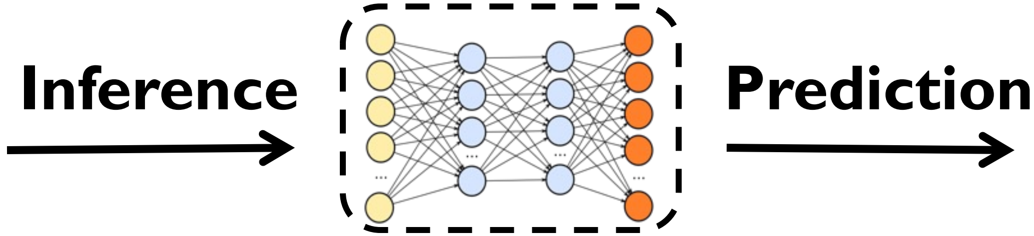
Video Playback

Google's video codec

Video Capture

Google's video codec

TensorFlow Mobile

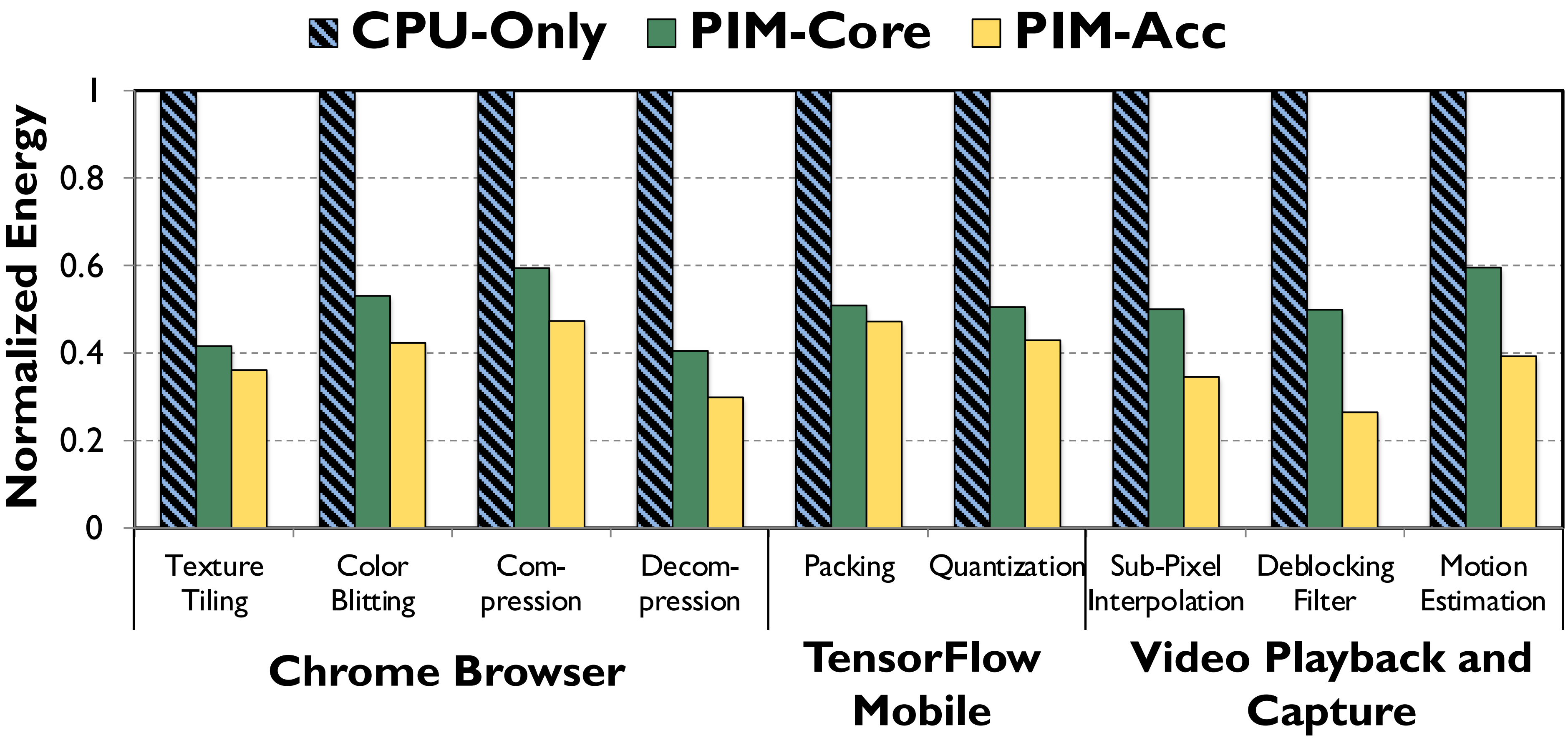


57.3% of the inference energy is spent on
data movement

↓

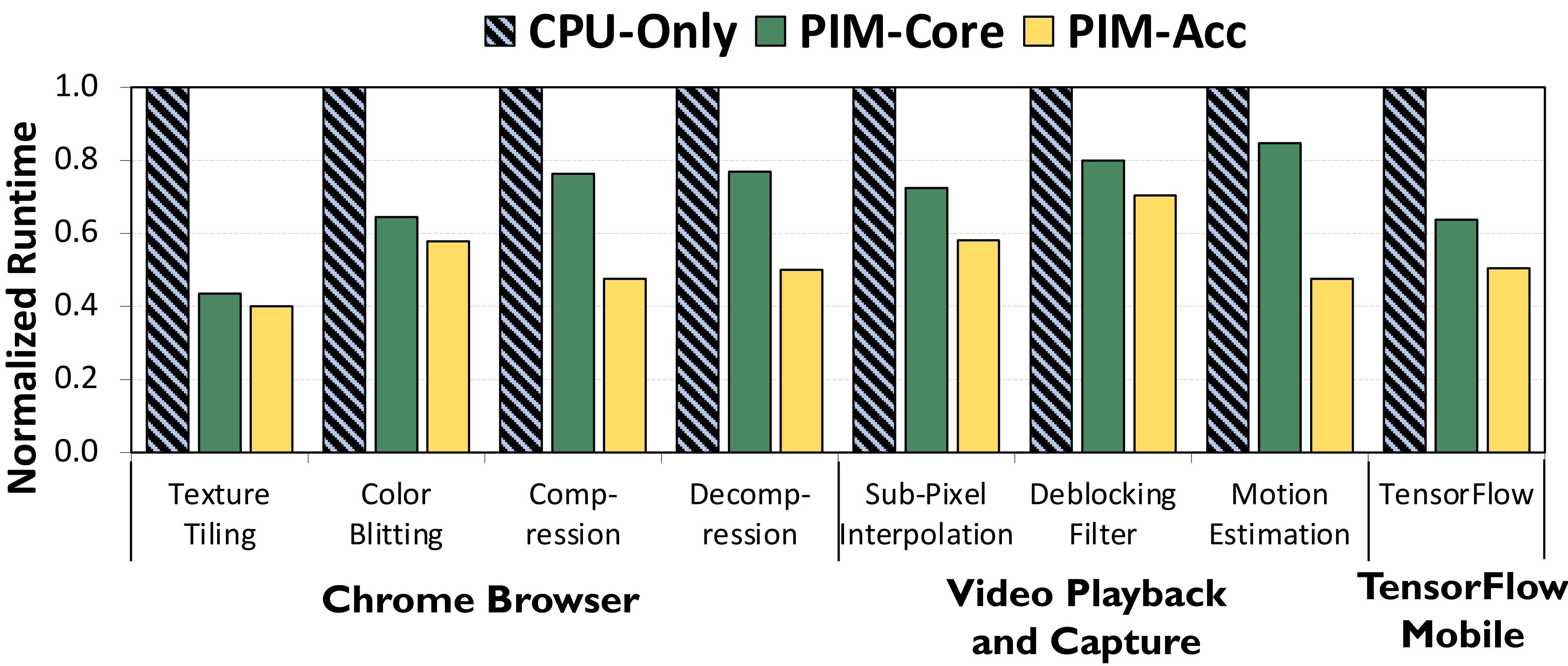
54.4% of the **data movement** energy comes from
packing/unpacking and **quantization**

Normalized Energy



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

Normalized Runtime



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

More on PIM for Mobile Devices

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu,
  "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"
  *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Williamsburg, VA, USA, March 2018.
  [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
  [Lightning Talk Video (2 minutes)]
  [Full Talk Video (21 minutes)]

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand[1] Saugata Ghose[1] Youngsok Kim[2]
Rachata Ausavarungnirun[1] Eric Shiu[3] Rahul Thakur[3] Daehyun Kim[4,3]
Aki Kuusela[3] Allan Knies[3] Parthasarathy Ranganathan[3] Onur Mutlu[5,1]

Accelerating Neural Network Inference

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,
"Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks"
*Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, Virtual, September 2021.
[Slides (pptx) (pdf)]
[Talk Video (14 minutes)]

Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand[†◇] Saugata Ghose[‡] Berkin Akin[§] Ravi Narayanaswami[§]
Geraldo F. Oliveira[⋆] Xiaoyu Ma[§] Eric Shiu[§] Onur Mutlu[⋆†]

[†]*Carnegie Mellon Univ.* [◇]*Stanford Univ.* [‡]*Univ. of Illinois Urbana-Champaign* [§]*Google* [⋆]*ETH Zürich*

Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand Saugata Ghose Berkin Akin
Ravi Narayanaswami Geraldo F. Oliveira Xiaoyu Ma
Eric Shiu Onur Mutlu

PACT 2021

SAFARI

Carnegie Mellon | University of Illinois Urbana-Champaign | Google | ETH Zürich

Executive Summary

Context: We extensively analyze **a state-of-the-art edge ML accelerator (Google Edge TPU)** using **24 Google edge models**

- Wide range of models (CNNs, LSTMs, Transducers, RCNNs)

Problem: The Edge TPU accelerator suffers from **three challenges:**

- It operates **significantly below** its peak throughput
- It operates **significantly below** its theoretical energy efficiency
- It **inefficiently** handles memory accesses

Key Insight: These shortcomings arise from **the monolithic design** of the Edge TPU accelerator

- The Edge TPU accelerator design does not account for **layer heterogeneity**

Key Mechanism: A new framework called **Mensa**

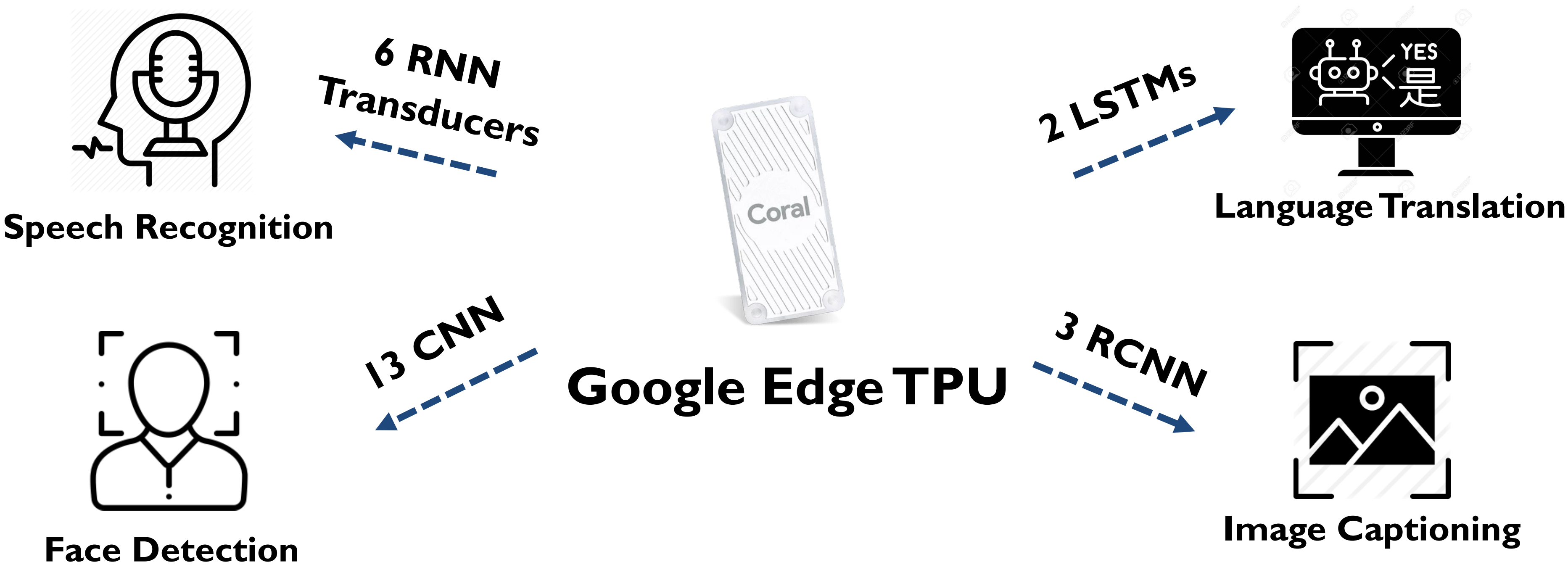
- Mensa consists of heterogeneous accelerators whose dataflow and hardware are specialized for specific families of layers

Key Results: We design a version of Mensa for Google edge ML models

- Mensa improves performance and energy by **3.0X** and **3.1X**
- Mensa reduces cost and improves area efficiency

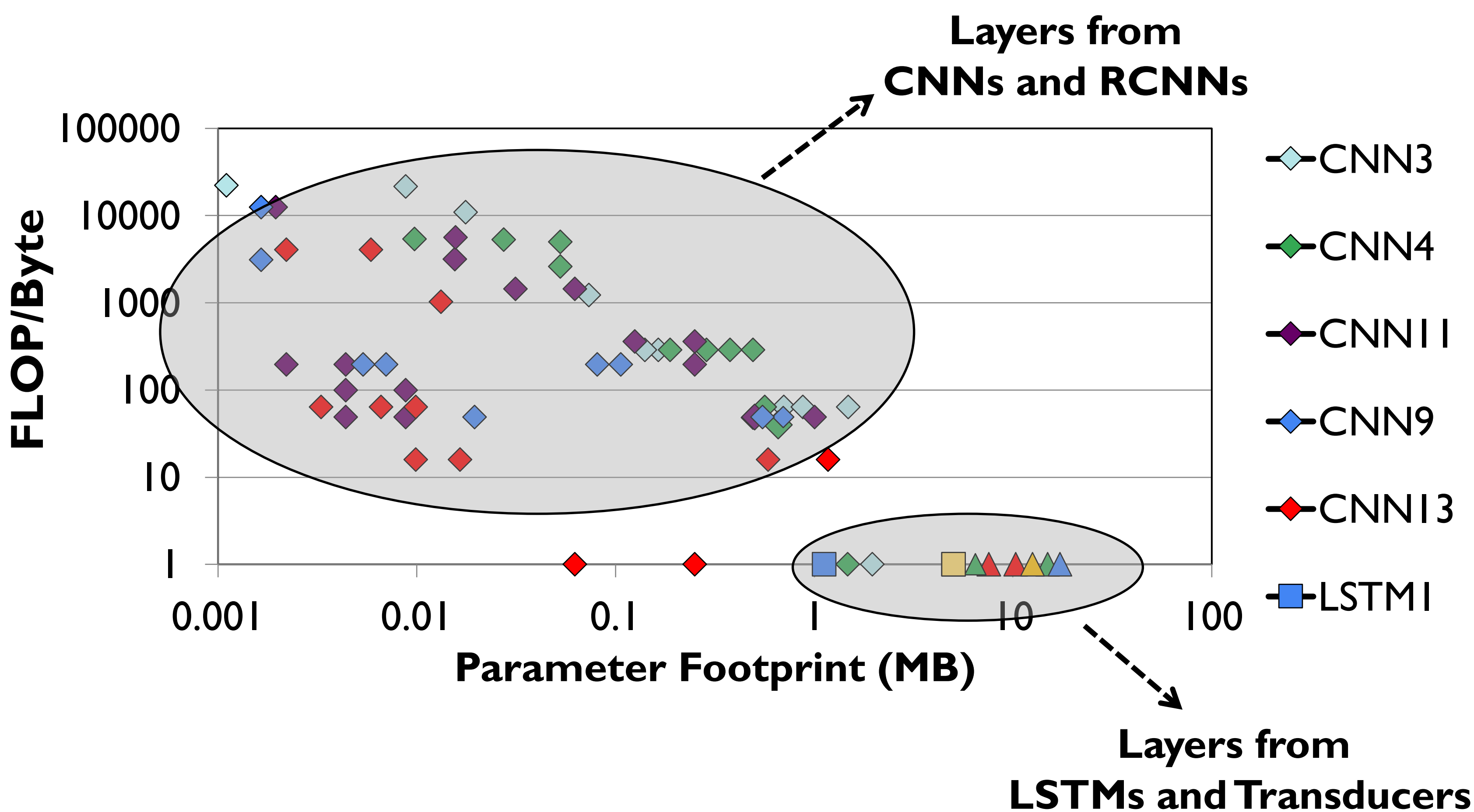
Google Edge Neural Network Models

We analyze inference execution using 24 edge NN models



Diversity Across the Models

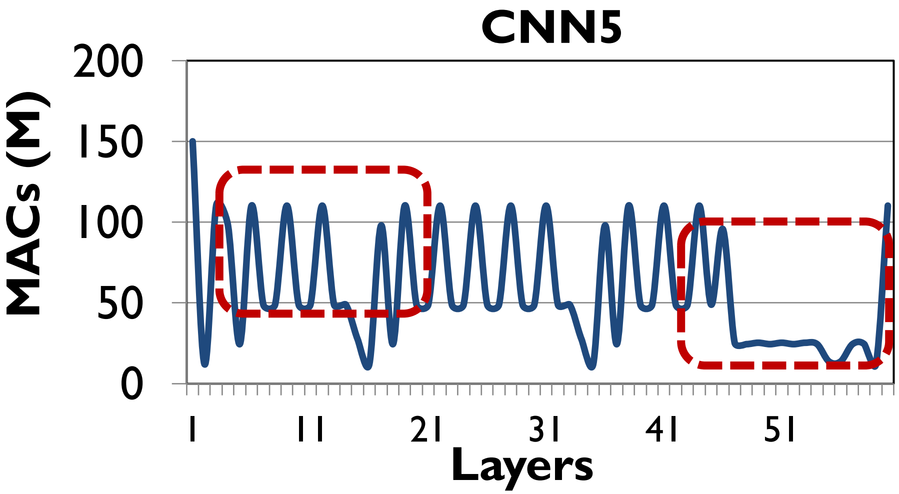
Insight 1: there is significant variation in terms of <u>layer characteristics</u> across the models

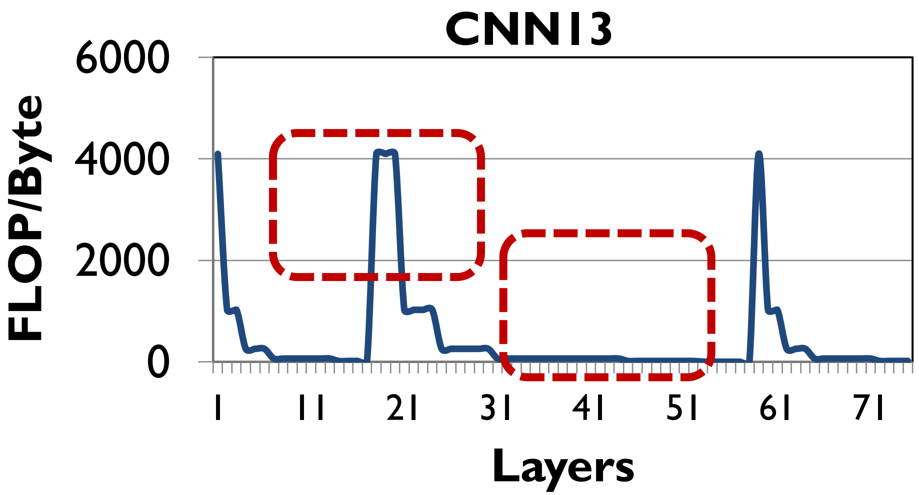


Diversity Within the Models

Insight 2: even within each model, layers exhibit significant variation in terms of layer characteristics

For example, our analysis of edge CNN models shows:

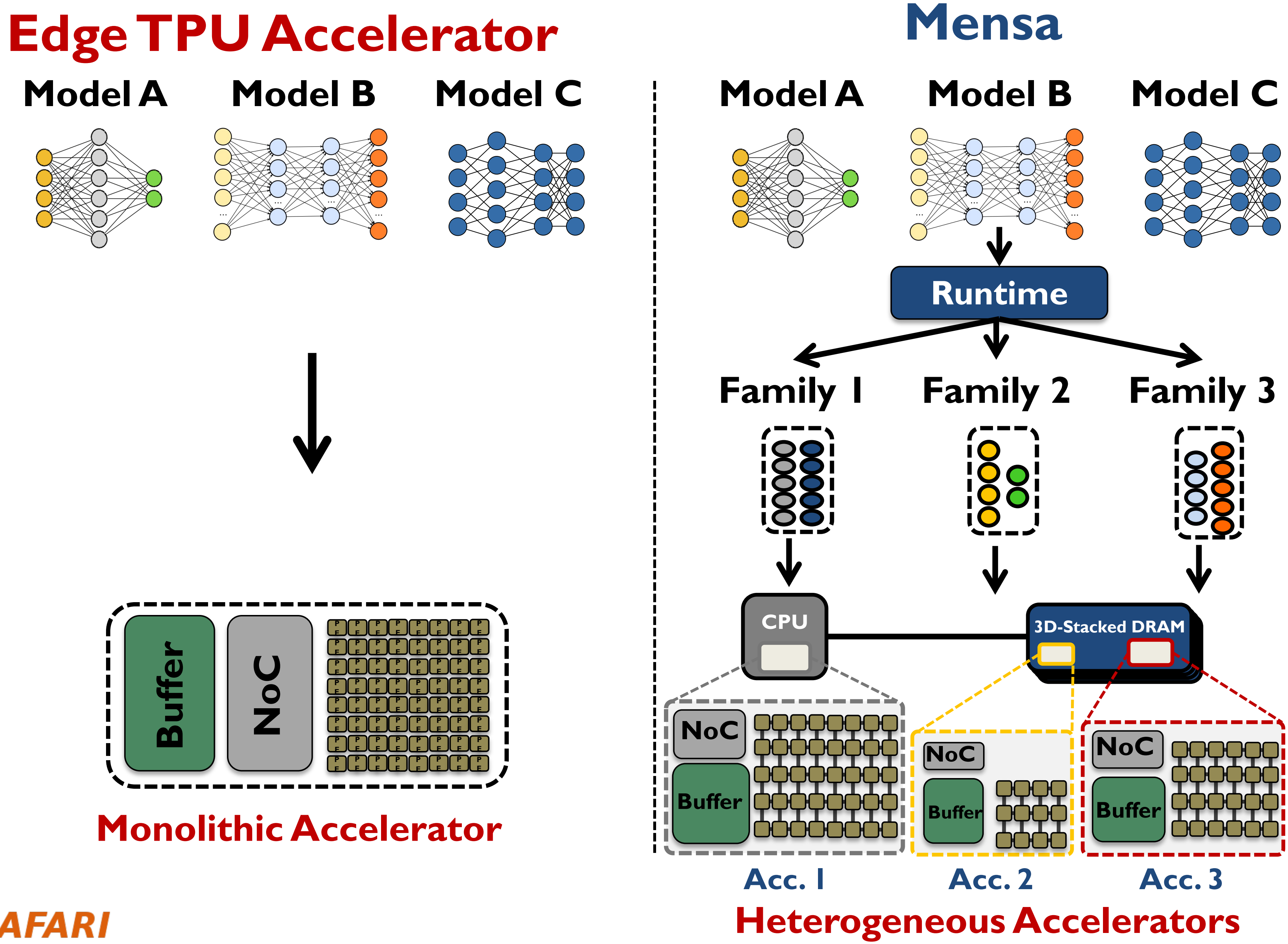




Variation in MAC intensity: up to 200x across layers

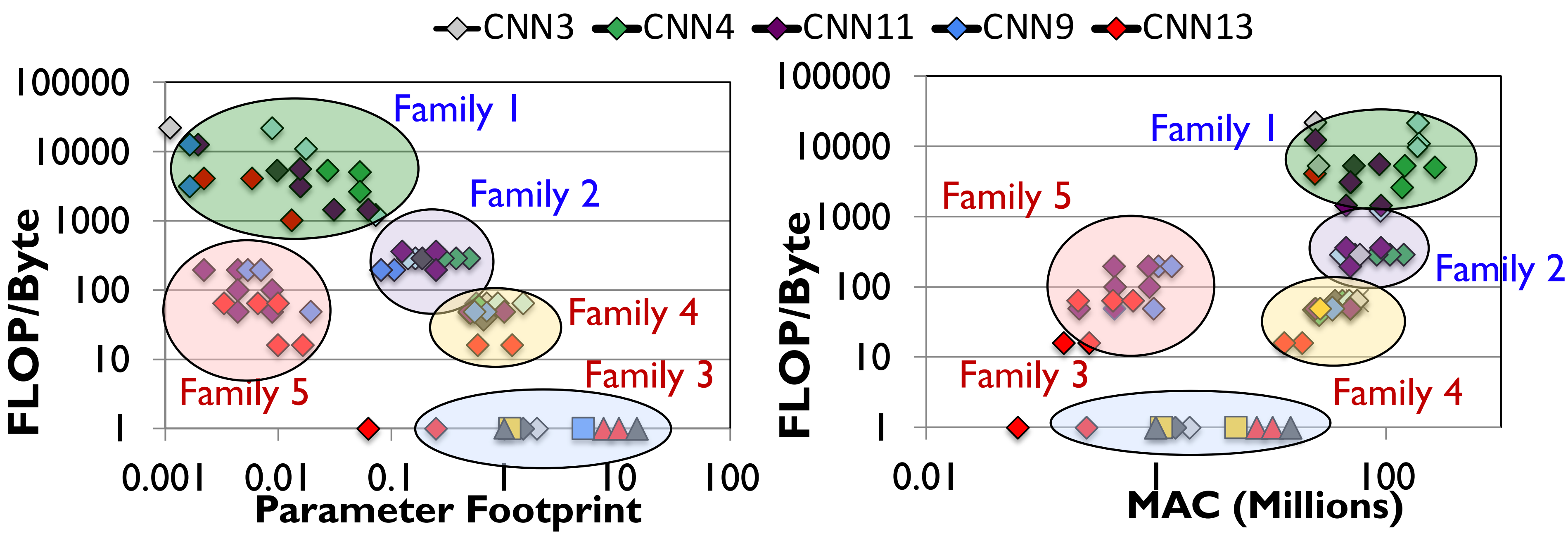
Variation in FLOP/Byte: up to 244x across layers

Mensa High-Level Overview



Identifying Layer Families

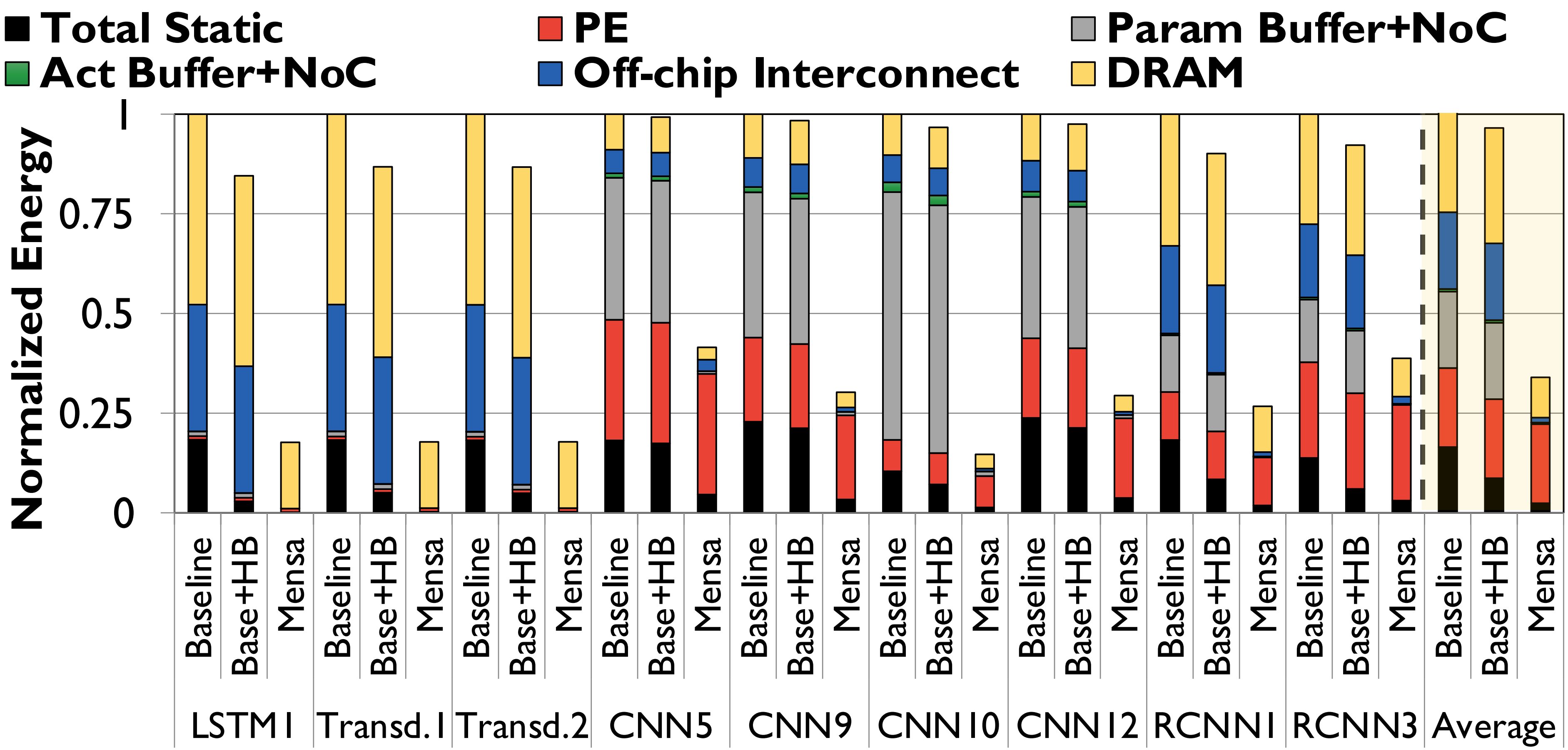
Key observation: the majority of layers group into a small number of layer families



Families 1 & 2: low parameter footprint, high data reuse and MAC intensity → compute-centric layers

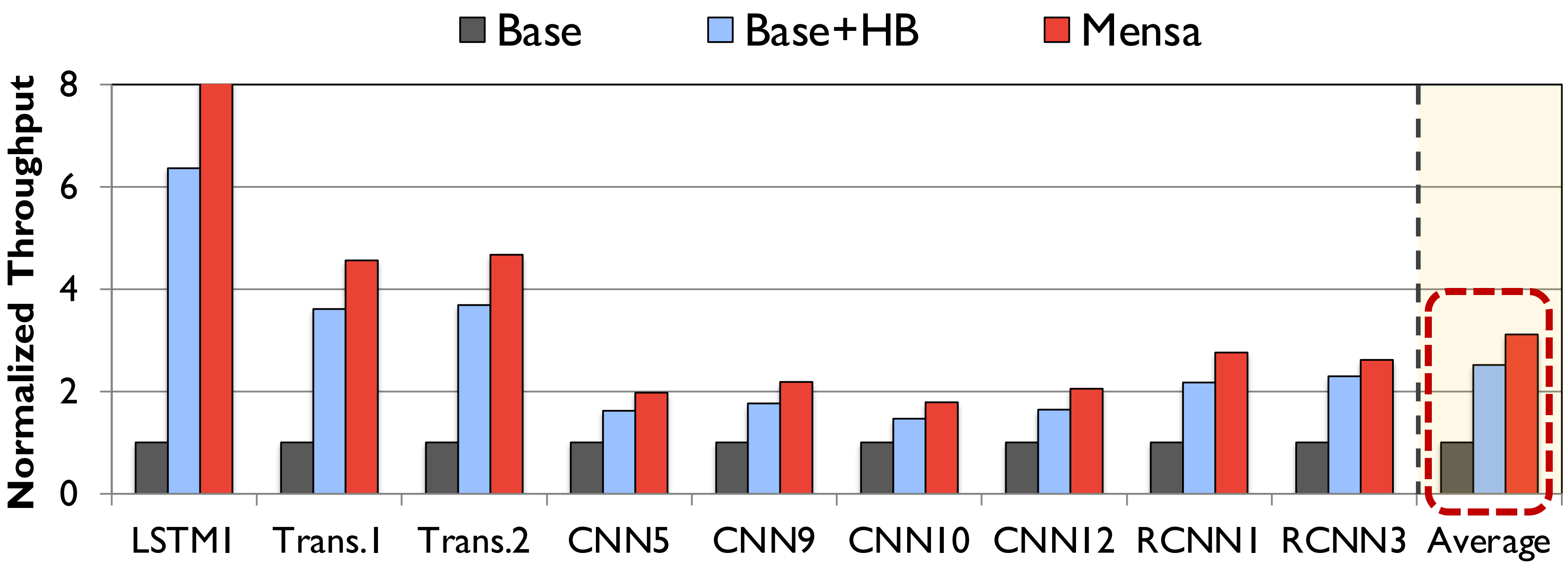
Families 3, 4 & 5: high parameter footprint, low data reuse and MAC intensity → data-centric layers

Mensa: Energy Reduction



Mensa-G reduces energy consumption by 3.0X compared to the baseline Edge TPU

Mensa: Throughput Improvement



Mensa-G improves inference throughput by 3.1X compared to the baseline Edge TPU

Mensa: Highly-Efficient ML Inference

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,
"Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks"
*Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, Virtual, September 2021.
[Slides (pptx) (pdf)]
[Talk Video (14 minutes)]

Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand[†◇] Saugata Ghose[‡] Berkin Akin[§] Ravi Narayanaswami[§]
Geraldo F. Oliveira[⋆] Xiaoyu Ma[§] Eric Shiu[§] Onur Mutlu[⋆†]

[†] *Carnegie Mellon Univ.* [◇] *Stanford Univ.* [‡] *Univ. of Illinois Urbana-Champaign* [§] *Google* [⋆] *ETH Zürich*

In-Storage Genomic Data Filtering [ASPLOS 2022]

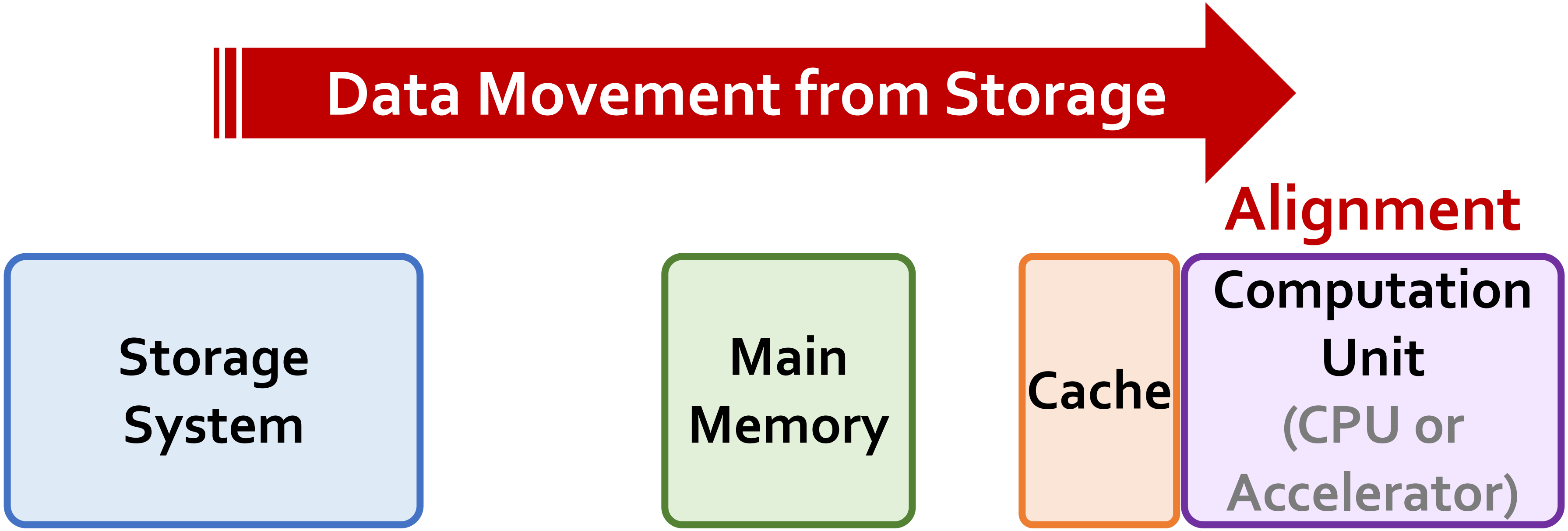
- Nika Mansouri Ghiasi, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alserr, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu,
  "GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis"
  *Proceedings of the 27th International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Virtual, February-March 2022.
  [Lightning Talk Slides (pptx) (pdf)]
  [Lightning Talk Video (90 seconds)]

GenStore: A High-Performance In-Storage Processing System for Genome Sequence Analysis

Nika Mansouri Ghiasi[1] Jisung Park[1] Harun Mustafa[1] Jeremie Kim[1] Ataberk Olgun[1]
Arvid Gollwitzer[1] Damla Senol Cali[2] Can Firtina[1] Haiyu Mao[1] Nour Almadhoun Alserr[1]
Rachata Ausavarungnirun[3] Nandita Vijaykumar[4] Mohammed Alser[1] Onur Mutlu[1]

[1]ETH Zürich [2]Bionano Genomics [3]KMUTNB [4]University of Toronto

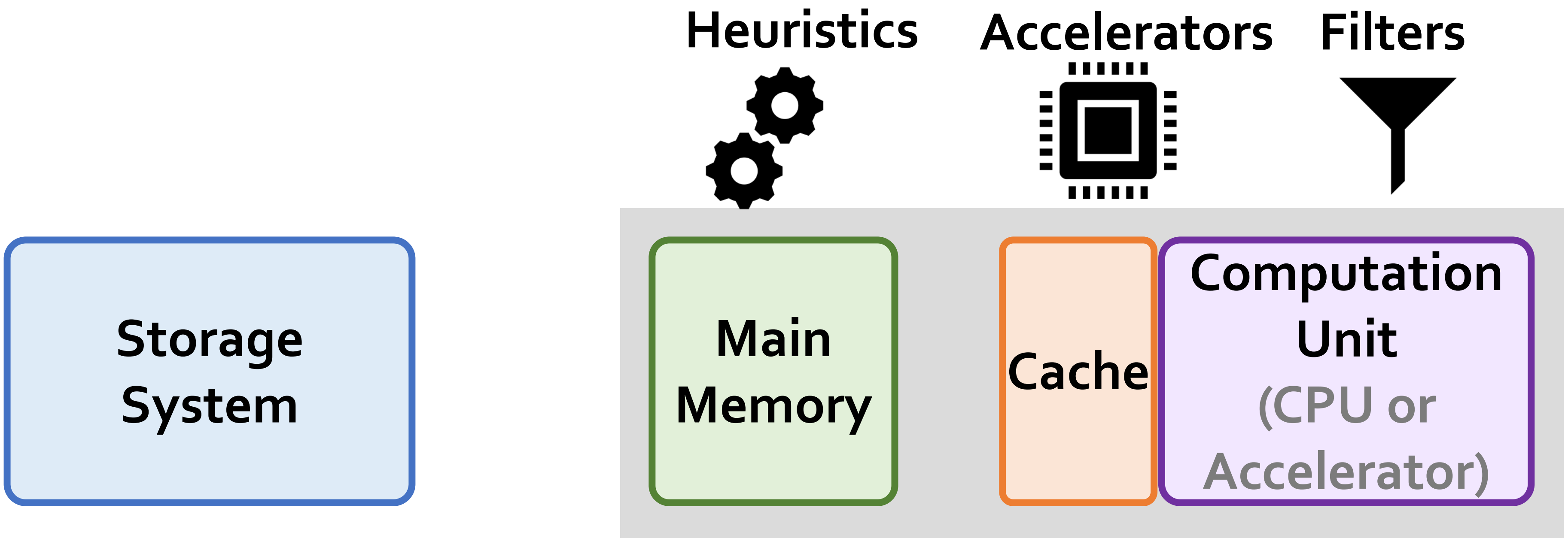
Genome Sequence Analysis



✗ Computation overhead

✗ Data movement overhead

Compute-Centric Accelerators

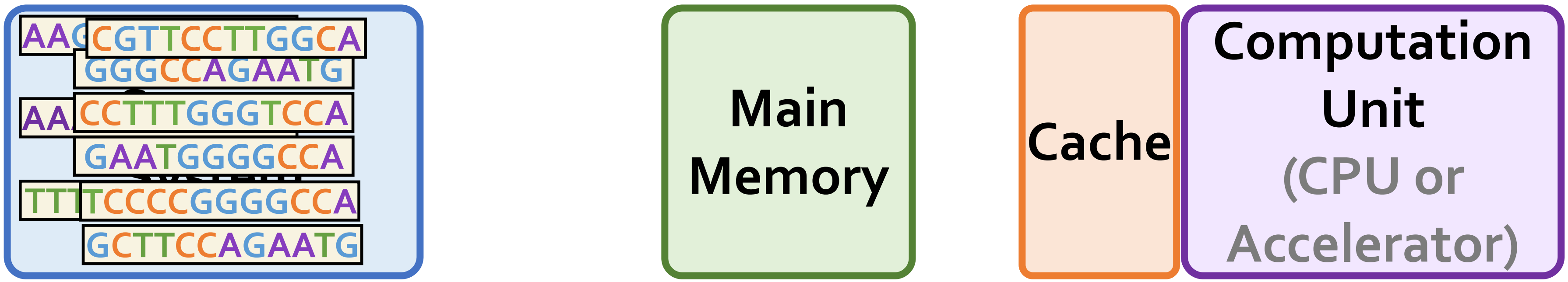


✓ Computation overhead

✗ Data movement overhead

Key Idea: In-Storage Filtering

Filter reads that do *not* require alignment
inside the storage system



Filtered Reads

Exactly-matching reads
Do not need expensive approximate string matching during alignment

Non-matching reads
Do not have potential matching locations and can skip alignment

GenStore

Filter reads that do *not* require alignment *inside the storage system*

GenStore-Enabled Storage System

Main Memory

Cache

Computation Unit (CPU or Accelerator)

✓ Computation overhead

✓ Data movement overhead

GenStore provides significant speedup (1.4x - 33.6x) and energy reduction (3.9x – 29.2x) at low cost

In-Storage Genomic Data Filtering [ASPLOS 2022]

- Nika Mansouri Ghiasi, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alserr, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu,
  "GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis"
  *Proceedings of the 27th International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Virtual, February-March 2022.
  [Lightning Talk Slides (pptx) (pdf)]
  [Lightning Talk Video (90 seconds)]

GenStore: A High-Performance In-Storage Processing System for Genome Sequence Analysis

Nika Mansouri Ghiasi[1] Jisung Park[1] Harun Mustafa[1] Jeremie Kim[1] Ataberk Olgun[1]
Arvid Gollwitzer[1] Damla Senol Cali[2] Can Firtina[1] Haiyu Mao[1] Nour Almadhoun Alserr[1]
Rachata Ausavarungnirun[3] Nandita Vijaykumar[4] Mohammed Alser[1] Onur Mutlu[1]

[1]ETH Zürich [2]Bionano Genomics [3]KMUTNB [4]University of Toronto

DAMOV Analysis Methodology & Workloads

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

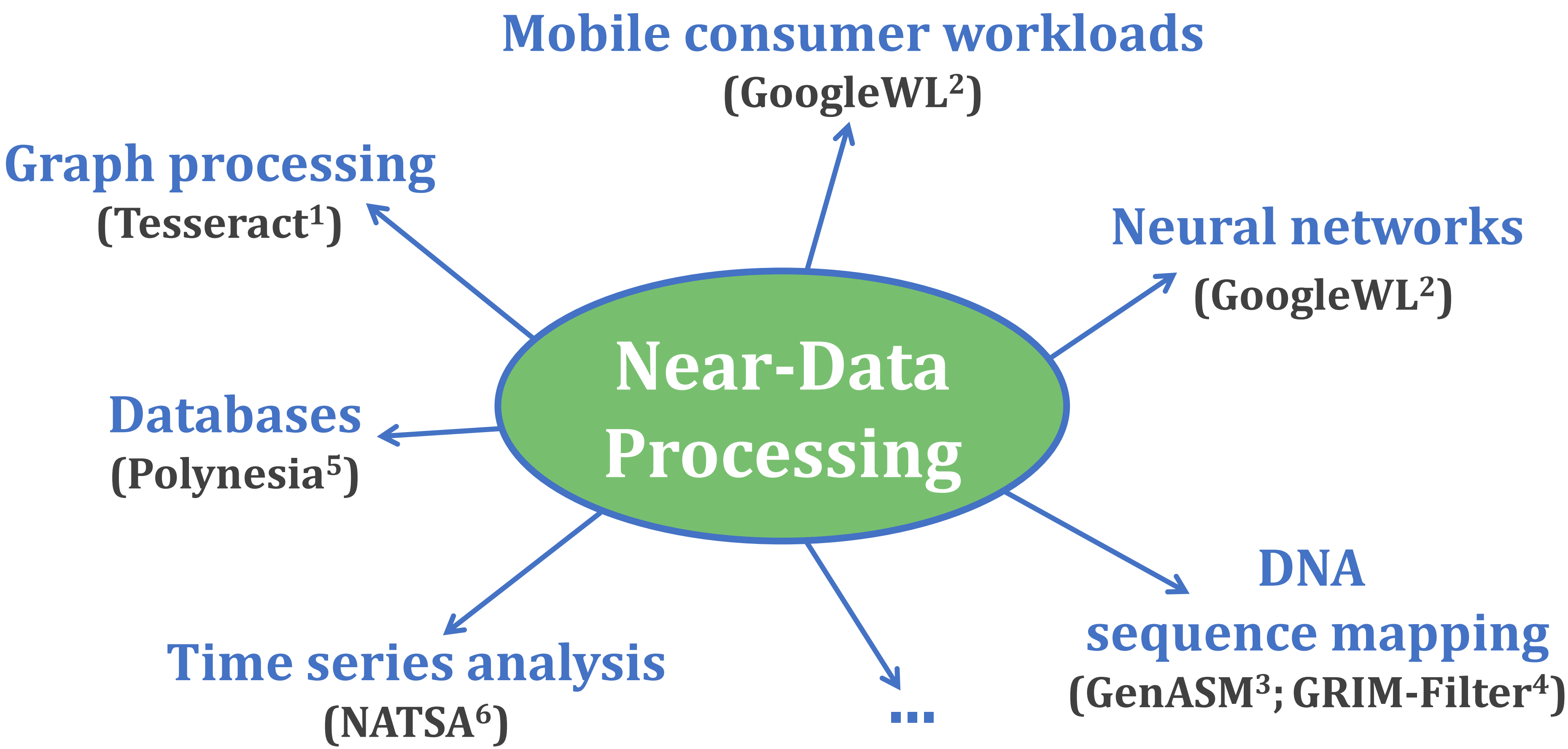
GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
LOIS OROSA, ETH Zürich, Switzerland
SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA
NANDITA VIJAYKUMAR, University of Toronto, Canada
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland
MOHAMMAD SADROSADATI, Institute for Research in Fundamental Sciences (IPM), Iran & ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

Data movement between the CPU and main memory is a first-order obstacle against improving performance, scalability, and energy efficiency in modern systems. Computer systems employ a range of techniques to reduce overheads tied to data movement, spanning from traditional mechanisms (e.g., deep multi-level cache hierarchies, aggressive hardware prefetchers) to emerging techniques such as Near-Data Processing (NDP), where some computation is moved close to memory. Prior NDP works investigate the root causes of data movement bottlenecks using different profiling methodologies and tools. However, there is still a lack of understanding about the key metrics that can identify different data movement bottlenecks and their relation to traditional and emerging data movement mitigation mechanisms. Our goal is to methodically identify potential sources of data movement over a broad set of applications and to comprehensively compare traditional compute-centric data movement mitigation techniques (e.g., caching and prefetching) to more memory-centric techniques (e.g., NDP), thereby developing a rigorous understanding of the best techniques to mitigate each source of data movement.

With this goal in mind, we perform the first large-scale characterization of a wide variety of applications, across a wide range of application domains, to identify fundamental program properties that lead to data movement to/from main memory. We develop the first systematic methodology to classify applications based on the sources contributing to data movement bottlenecks. From our large-scale characterization of 77K functions across 345 applications, we select 144 functions to form the first open-source benchmark suite (DAMOV) for main memory data movement studies. We select a diverse range of functions that (1) represent different types of data movement bottlenecks, and (2) come from a wide range of application domains. Using NDP as a case study, we identify new insights about the different data movement bottlenecks and use these insights to determine the most suitable data movement mitigation mechanism for a particular application. We open-source DAMOV and the complete source code for our new characterization methodology at https://github.com/CMU-SAFARI/DAMOV.

https://arxiv.org/pdf/2105.03725.pdf

When to Employ Near-Data Processing?



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

[2] Boroumand+, "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," ASPLOS, 2018

[3] Cali+, "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis," MICRO, 2020

[4] Kim+, "GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies," BMC Genomics, 2018

[5] Boroumand+, "Polynesia: Enabling Effective Hybrid Transactional/Analytical Databases with Specialized Hardware/Software Co-Design," arXiv:2103.00798 [cs.AR], 2021

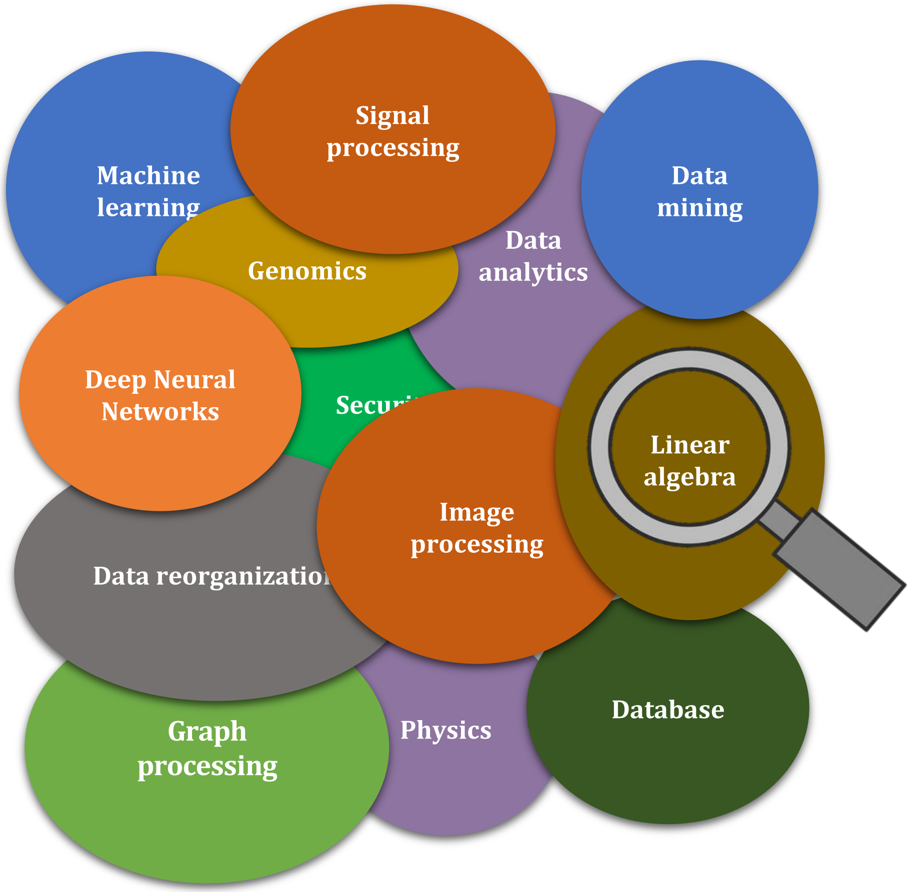
[6] Fernandez+, "NATSA: A Near-Data Processing Accelerator for Time Series Analysis," ICCD, 2020

Step 1: Application Profiling

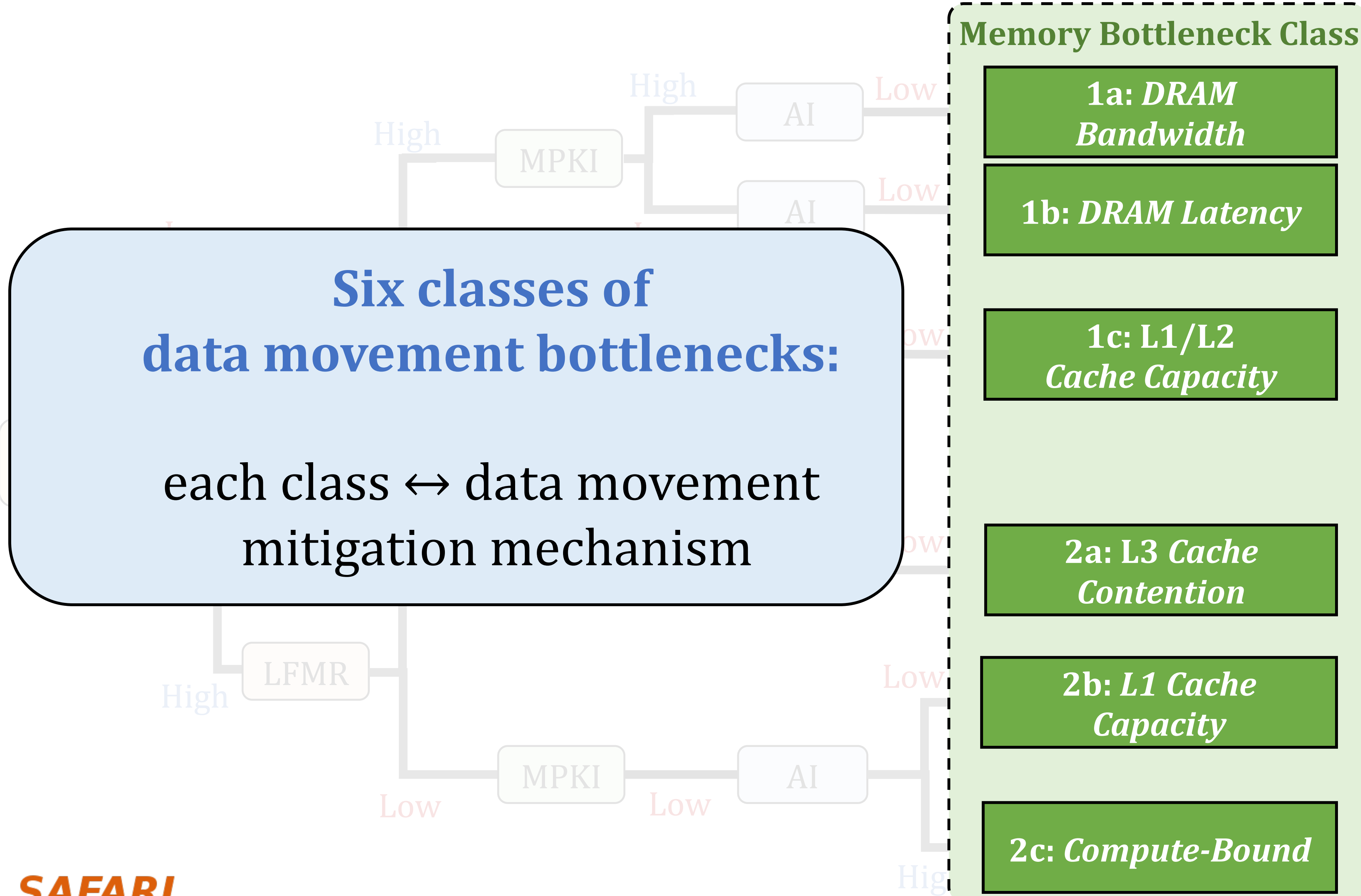
- We analyze 345 applications from distinct domains:

- Graph Processing
- Deep Neural Networks
- Physics
- High-Performance Computing
- Genomics
- Machine Learning
- Databases
- Data Reorganization
- Image Processing
- Map-Reduce
- Benchmarking
- Linear Algebra

...



Step 3: Memory Bottleneck Analysis

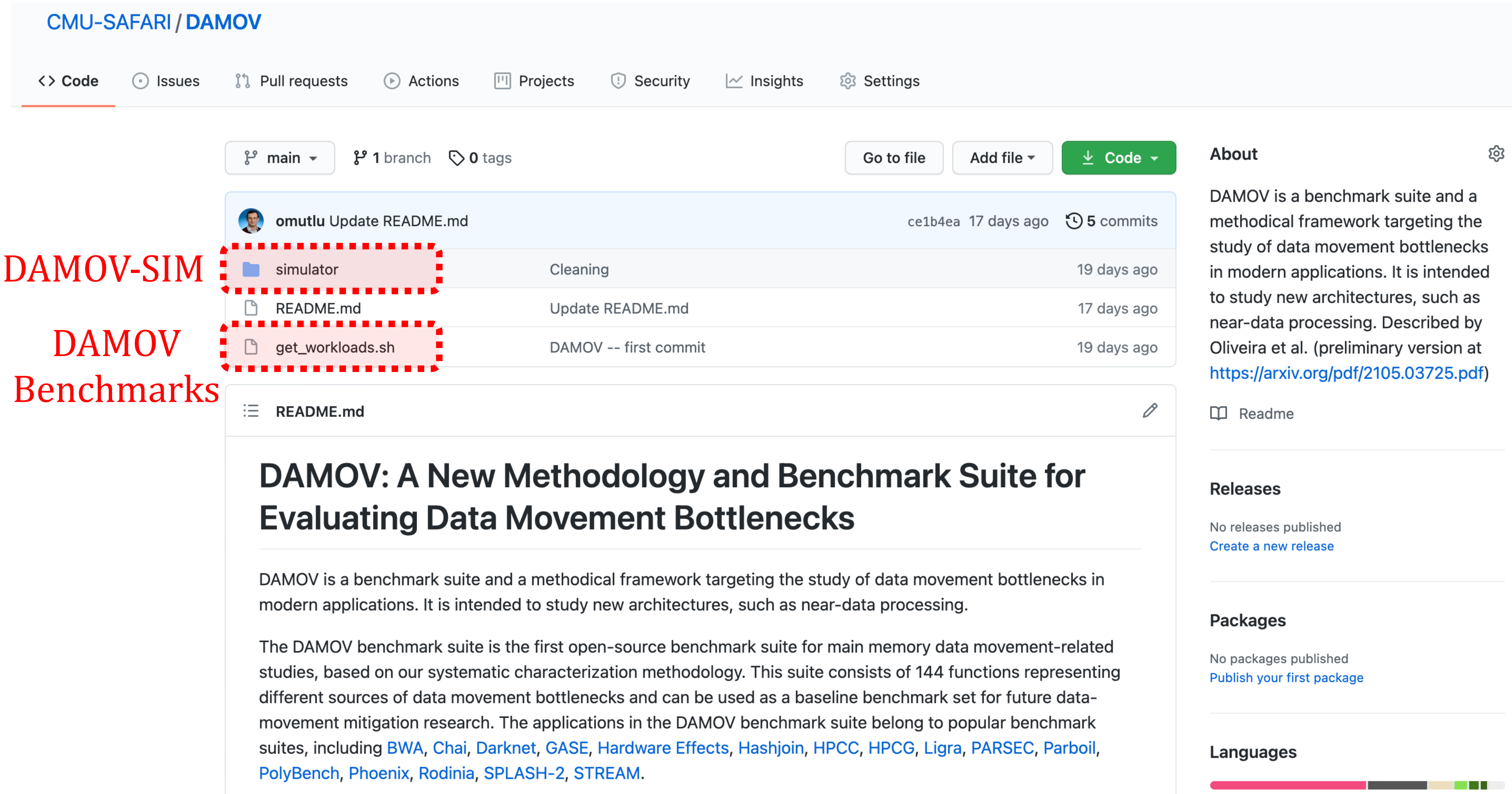


Six classes of data movement bottlenecks:

each class ↔ data movement mitigation mechanism

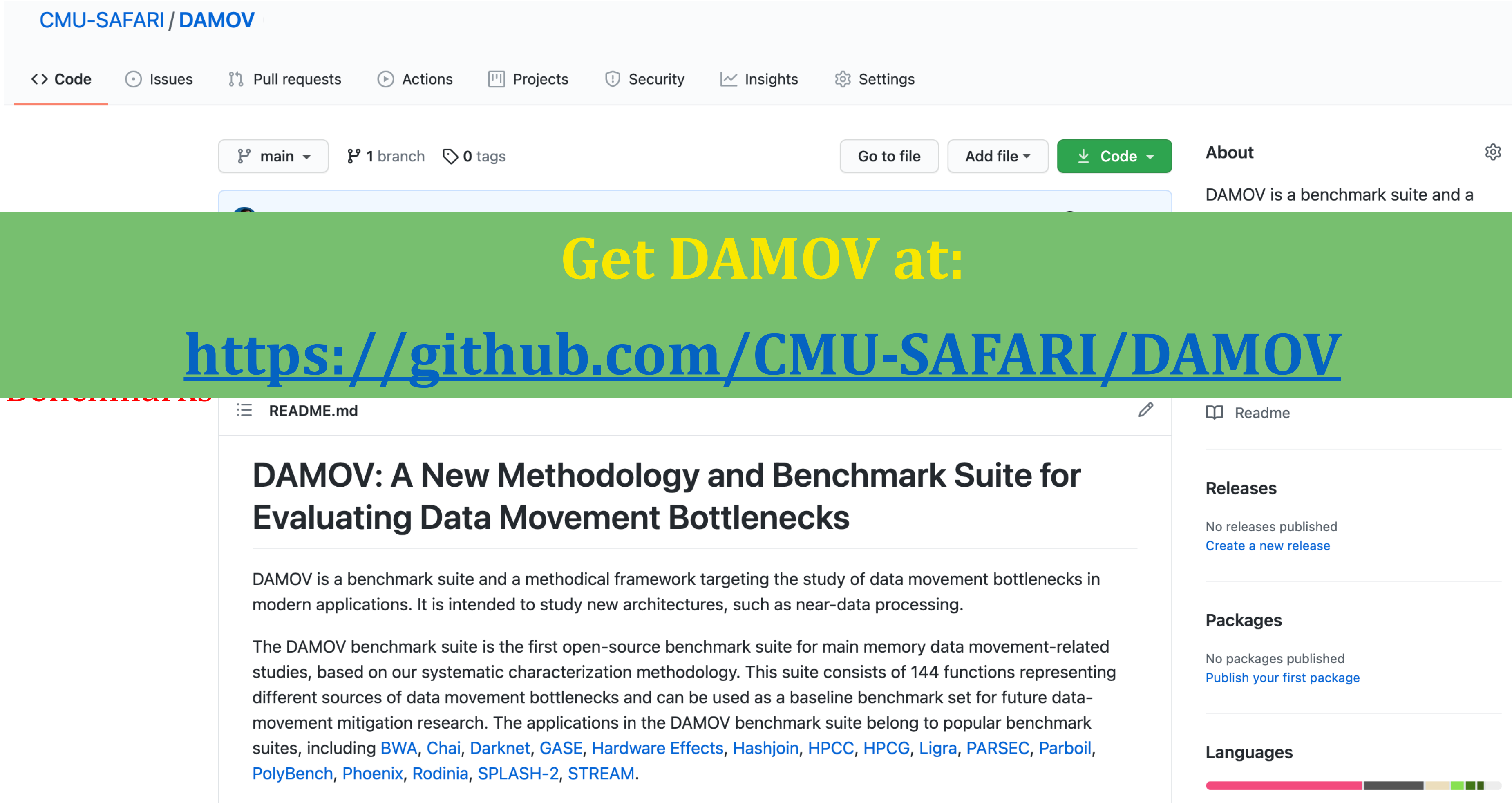
DAMOV is Open Source

- We open-source our benchmark suite and our toolchain



DAMOV is Open Source

- We open-source our benchmark suite and our toolchain

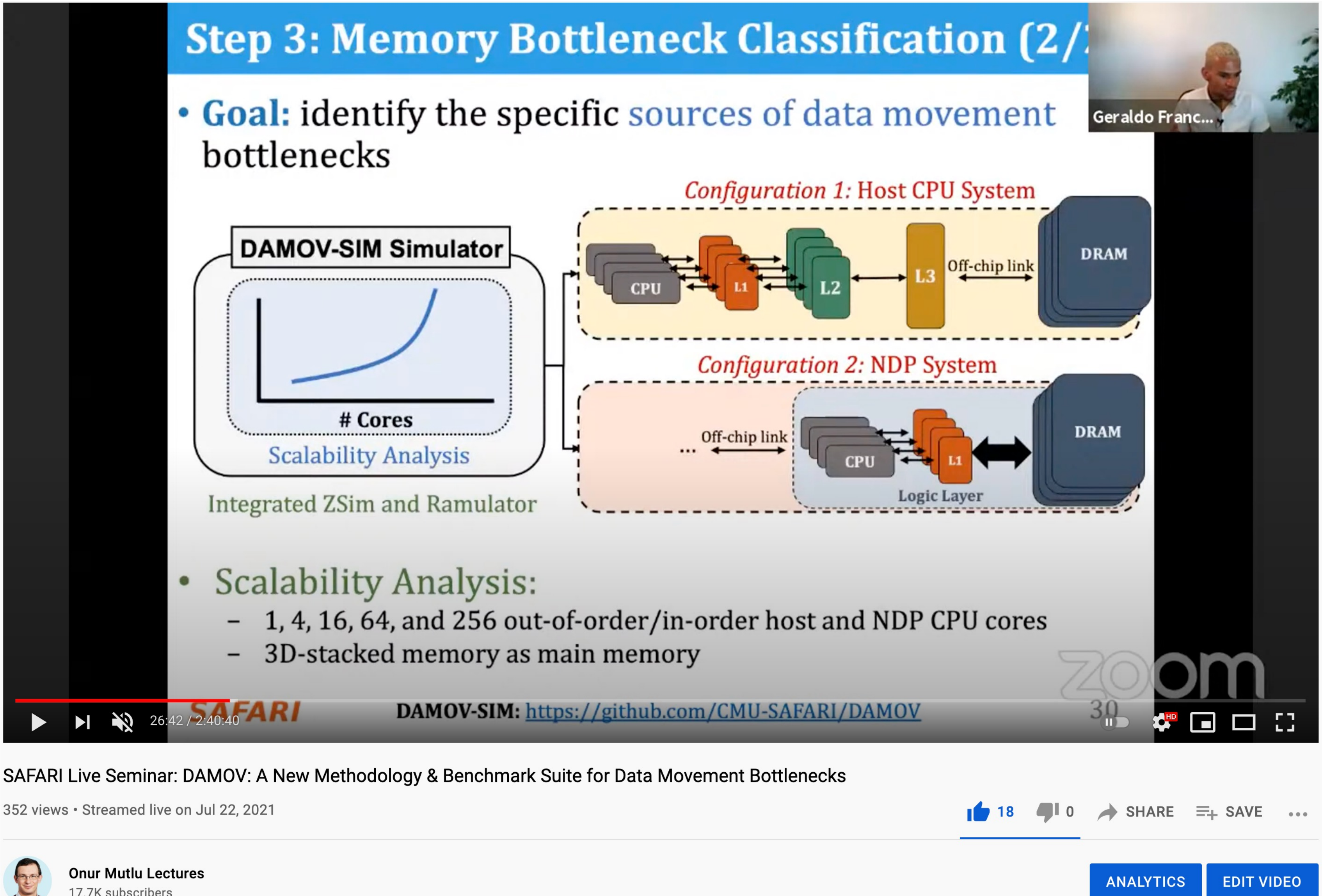


Get DAMOV at:

https://github.com/CMU-SAFARI/DAMOV

SAFARI

More on DAMOV Analysis Methodology & Workloads



https://www.youtube.com/watch?v=GWideVyo0nM&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9&index=3

More on DAMOV Methods & Benchmarks

- Geraldo F. Oliveira, Juan Gomez-Luna, Lois Orosa, Saugata Ghose, Nandita Vijaykumar, Ivan fernandez, Mohammad Sadrosadati, and Onur Mutlu,
  "DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks"
  IEEE Access, 8 September 2021.
  *Preprint in **arXiv***, 8 May 2021.
  [arXiv preprint]
  [IEEE Access version]
  [DAMOV Suite and Simulator Source Code]
  [SAFARI Live Seminar Video (2 hrs 40 mins)]
  [Short Talk Video (21 minutes)]

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

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
LOIS OROSA, ETH Zürich, Switzerland
SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA
NANDITA VIJAYKUMAR, University of Toronto, Canada
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland
MOHAMMAD SADROSADATI, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

Ramulator 2.0 for PIM Systems

- Haocong Luo, Yahya Can Tugrul, F. Nisa Bostanci, Ataberk Olgun, A. Giray Yaglikci, and Onur Mutlu,
  "Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator"
  *Preprint on **arxiv***, August 2023.
  [arXiv version]
  [Ramulator 2.0 Source Code]

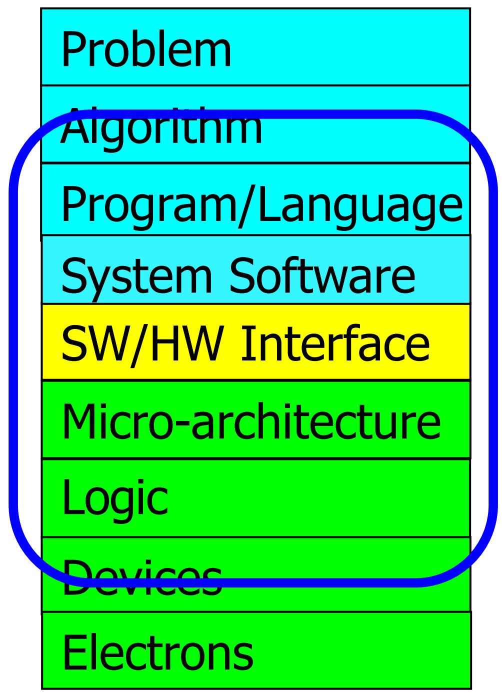
Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator

Haocong Luo, Yahya Can Tuğrul, F. Nisa Bostancı, Ataberk Olgun, A. Giray Yağlıkçı, and Onur Mutlu

https://arxiv.org/pdf/2308.11030.pdf

We Need to Revisit the Entire Stack

- With a **memory-centric mindset**



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.

https://arxiv.org/pdf/2012.03112.pdf

PIM Review and Open Problems (II)

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

Saugata Ghose[†] Amirali Boroumand[†] Jeremie S. Kim[†§] Juan Gómez-Luna[§] Onur Mutlu[§†]

[†]*Carnegie Mellon University* [§]*ETH Zürich*

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu, **"Processing-in-Memory: A Workload-Driven Perspective"**
Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence, to appear in November 2019.
[Preliminary arXiv version]

https://arxiv.org/pdf/1907.12947.pdf

Processing in Memory: Adoption Challenges

1. Processing **using** Memory
2. Processing **near** Memory

How to Enable Adoption of Processing in Memory

Potential Barriers to Adoption of PIM

1. **Applications** & **software** for PIM

2. Ease of **programming** (interfaces and compiler/HW support)

3. **System** and **security** support: coherence, synchronization, virtual memory, isolation, communication interfaces, …

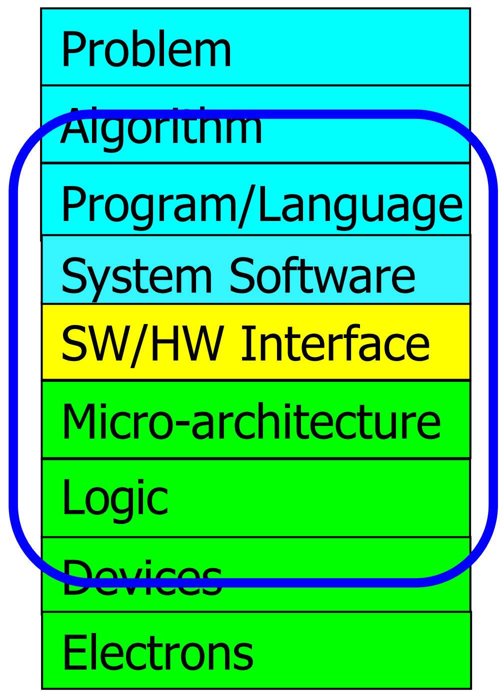
4. **Runtime** and **compilation** systems for adaptive scheduling, data mapping, access/sharing control, …

5. **Infrastructures** to assess benefits and feasibility

All can be solved with change of mindset

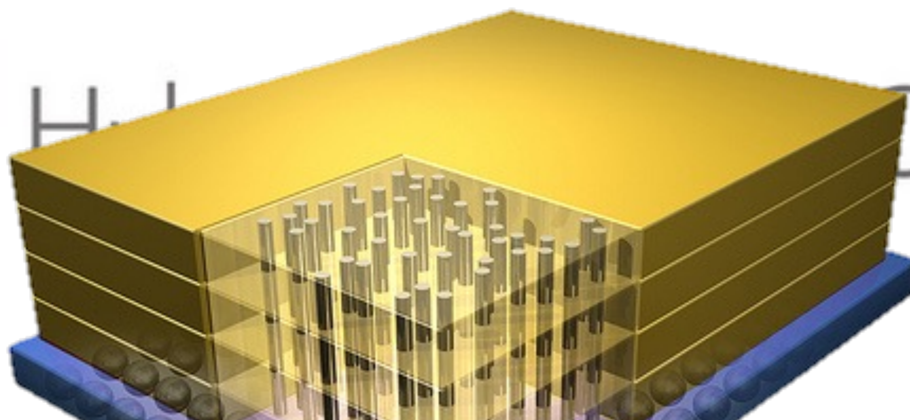
We Need to Revisit the Entire Stack

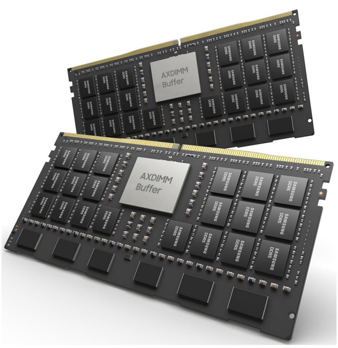
- With a **memory-centric mindset**



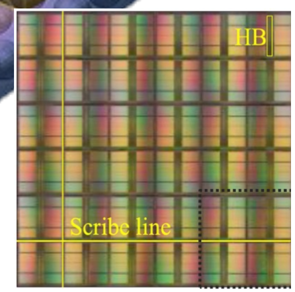
We can get there step by step

Processing-in-Memory Landscape Today

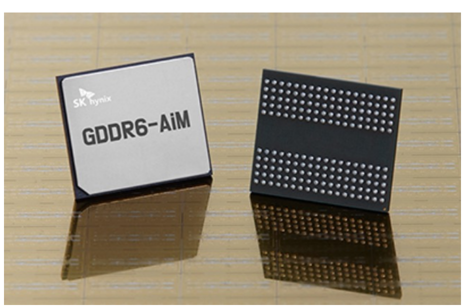




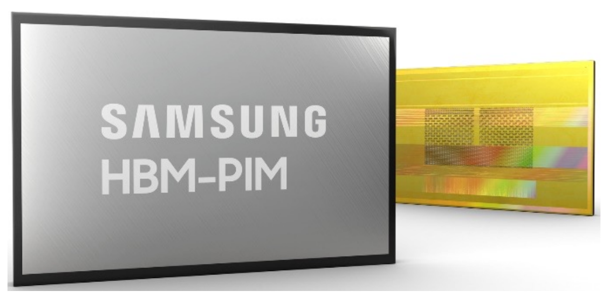
[Samsung 2021]



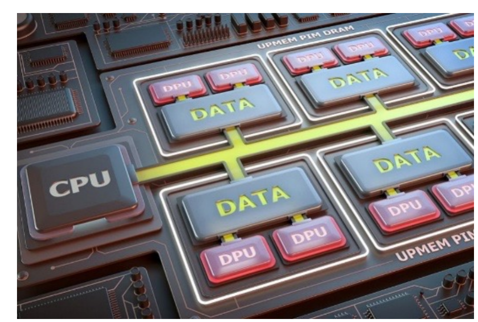
[Alibaba 2022]



[SK Hynix 2022]



[Samsung 2021]



[UPMEM 2019]

And, many other experimental chips and startups

Adoption: How to Keep It **Simple**?

- 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

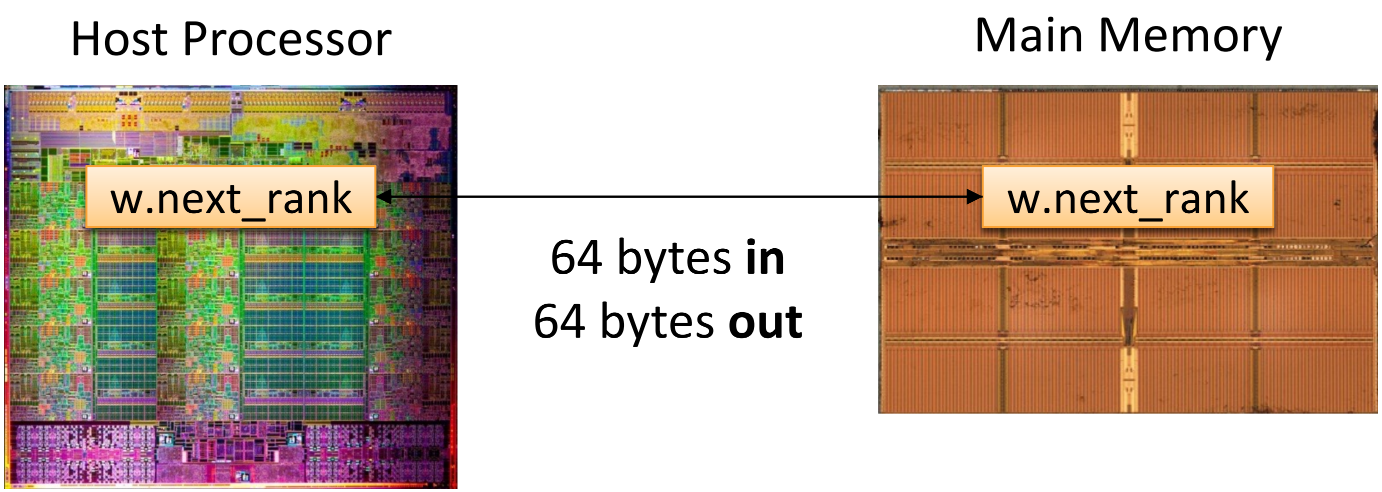
Junwhan Ahn Sungjoo Yoo Onur Mutlu[†] Kiyoung Choi
junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr
Seoul National University [†]Carnegie Mellon University

PEI: PIM-Enabled Instructions (Ideas)

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

Simple PIM Operations as ISA Extensions (II)

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



Conventional Architecture

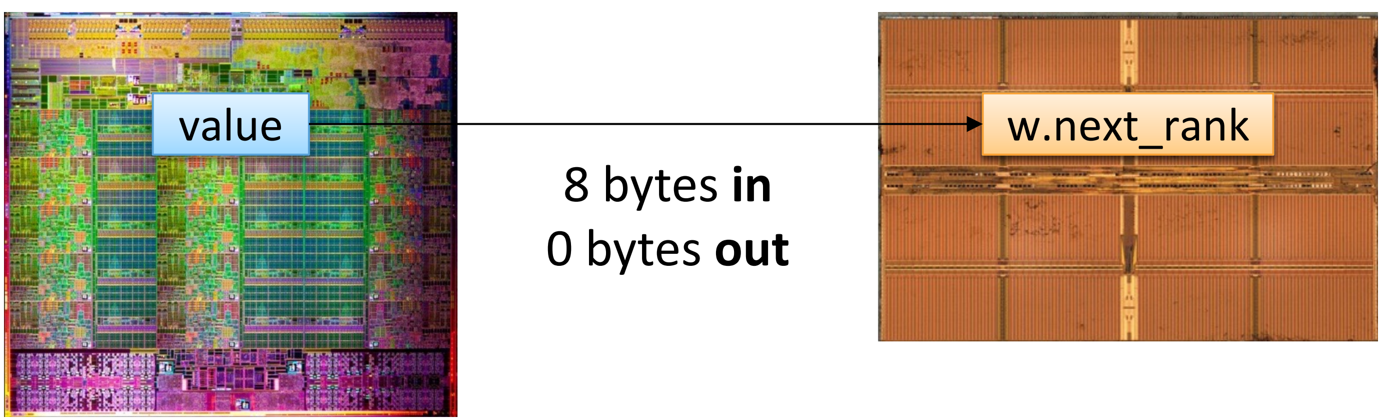
Simple PIM Operations as ISA Extensions (III)

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

pim.add r1, (r2)

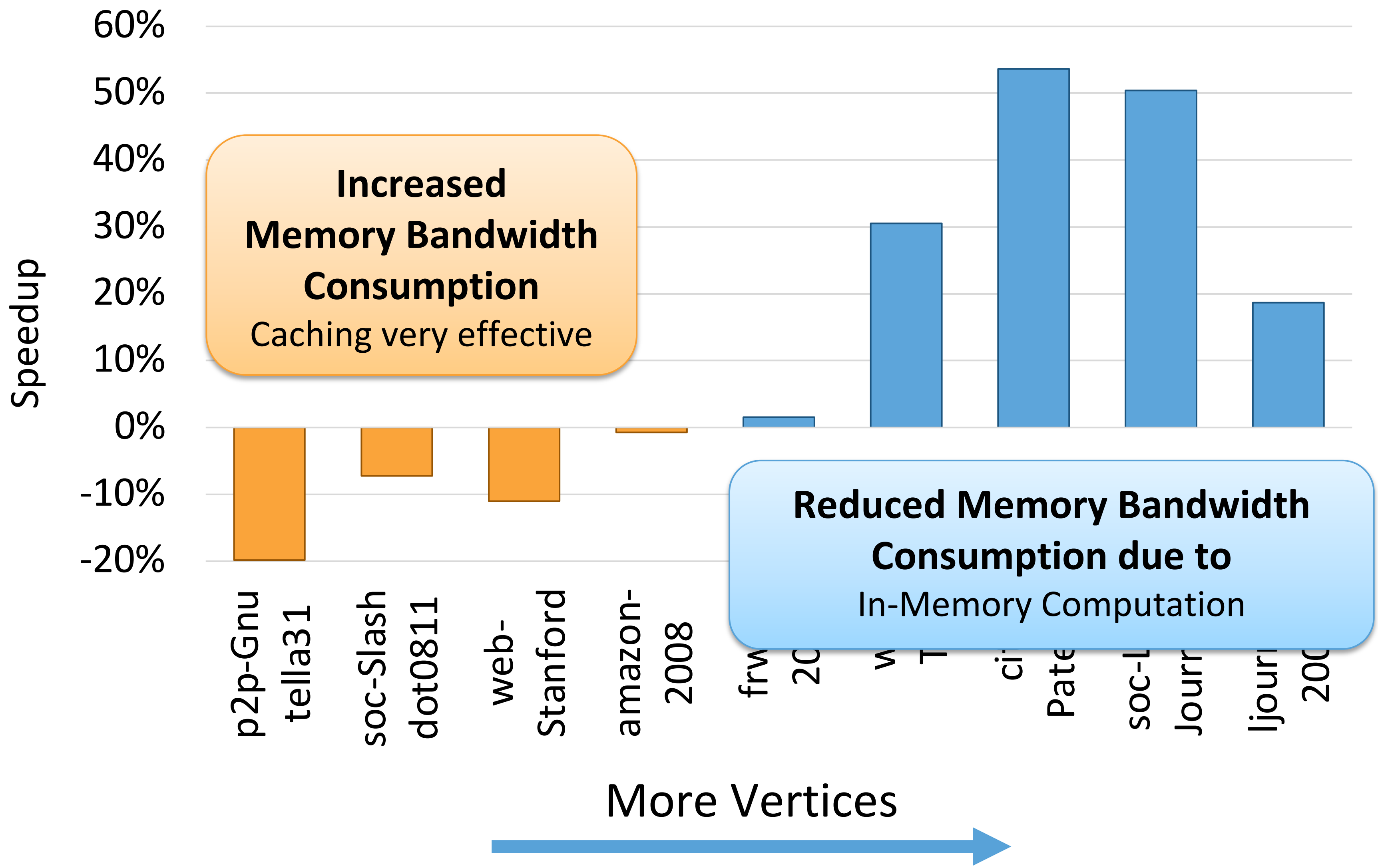
Host Processor

Main Memory



In-Memory Addition

Always Executing in Memory? Not A Good Idea



PEI: PIM-Enabled Instructions (Example)

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

pim.add r1, (r2)

pfence

Table 1: Summary of Supported PIM Operations

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

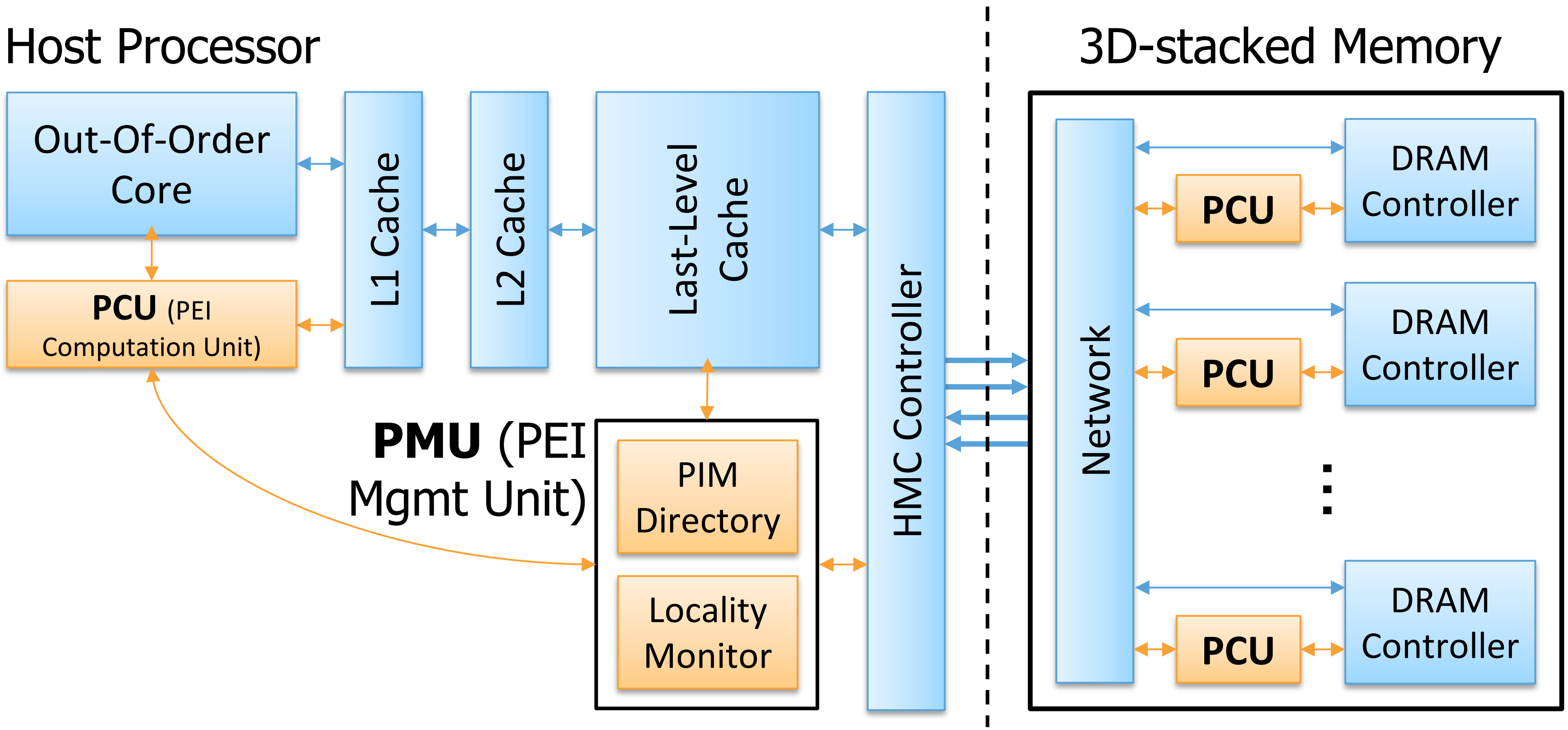
- Executed either in memory or in the processor: dynamic decision
  - Low-cost locality monitoring for a single instruction
- Cache-coherent, virtually-addressed, single cache block only
- Atomic between different PEIs
- *Not* atomic with normal instructions (use *pfence* for ordering)

PIM-Enabled Instructions

- Key to practicality: single-cache-block restriction
  - Each PEI can access *at most one last-level cache block*
  - Similar restrictions exist in atomic instructions

- Benefits
  - **Localization**: each PEI is bounded to one memory module
  - **Interoperability**: easier support for cache coherence and virtual memory
  - **Simplified locality monitoring**: data locality of PEIs can be identified simply by the cache control logic

Example (Abstract) PEI uArchitecture



Example PEI uArchitecture

PEI: Initial Evaluation Results

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

Table 2: Baseline Simulation Configuration

Component	Configuration
Core	16 out-of-order cores, 4 GHz, 4-issue
L1 I/D-Cache	Private, 32 KB, 4/8-way, 64 B blocks, 16 MSHRs
L2 Cache	Private, 256 KB, 8-way, 64 B blocks, 16 MSHRs
L3 Cache	Shared, 16 MB, 16-way, 64 B blocks, 64 MSHRs
On-Chip Network	Crossbar, 2 GHz, 144-bit links
Main Memory	32 GB, 8 HMCs, daisy-chain (80 GB/s full-duplex)
HMC	4 GB, 16 vaults, 256 DRAM banks [20]
– DRAM	FR-FCFS, tCL = tRCD = tRP = 13.75 ns [27]
– Vertical Links	64 TSVs per vault with 2 Gb/s signaling rate [23]

- Pin-based cycle-level x86-64 simulation

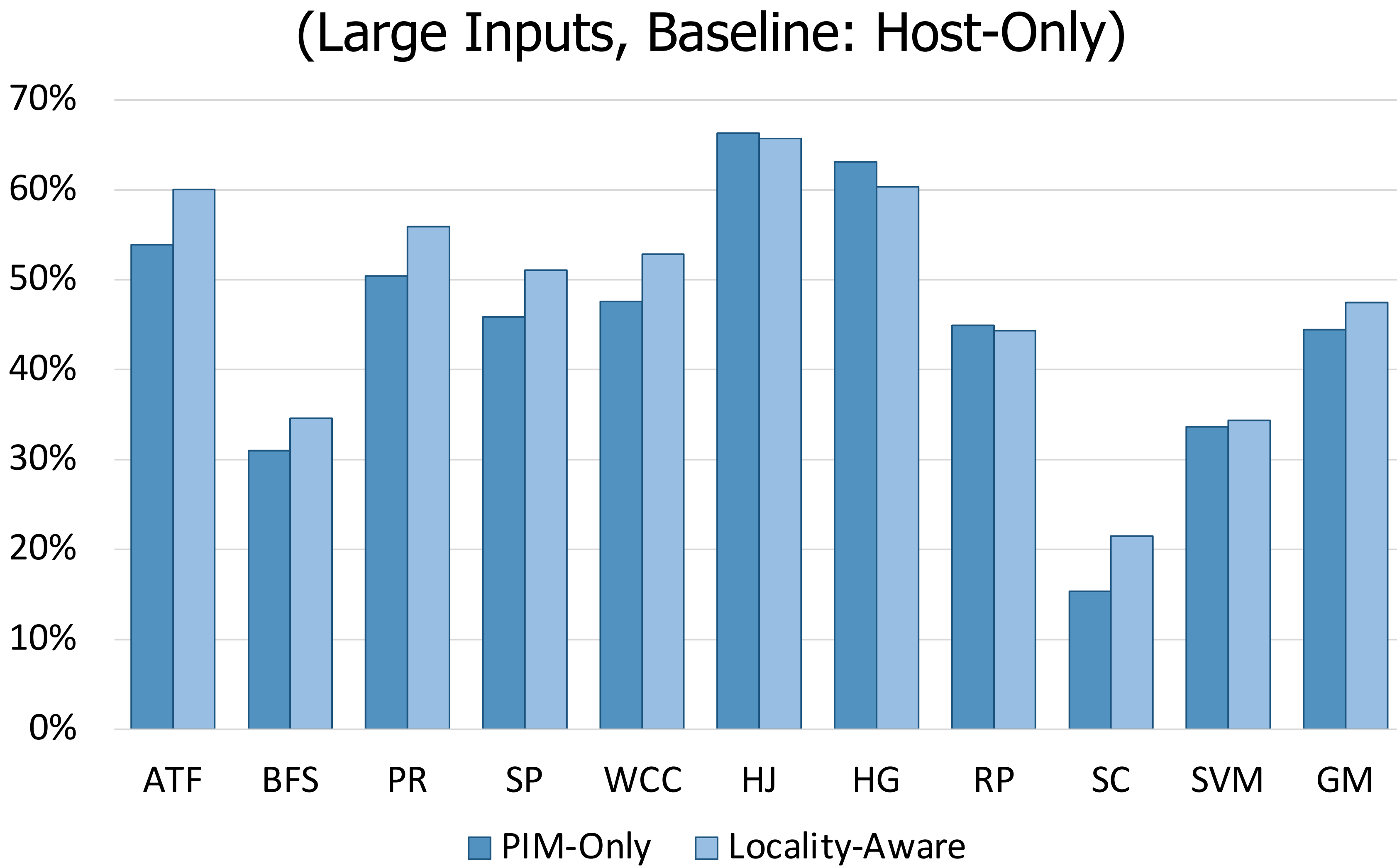
- **Performance Improvement and Energy Reduction:**
  - 47% average speedup with large input data sets
  - 32% speedup with small input data sets
  - 25% avg. energy reduction in a single node with large input data sets

Evaluated Data-Intensive Applications

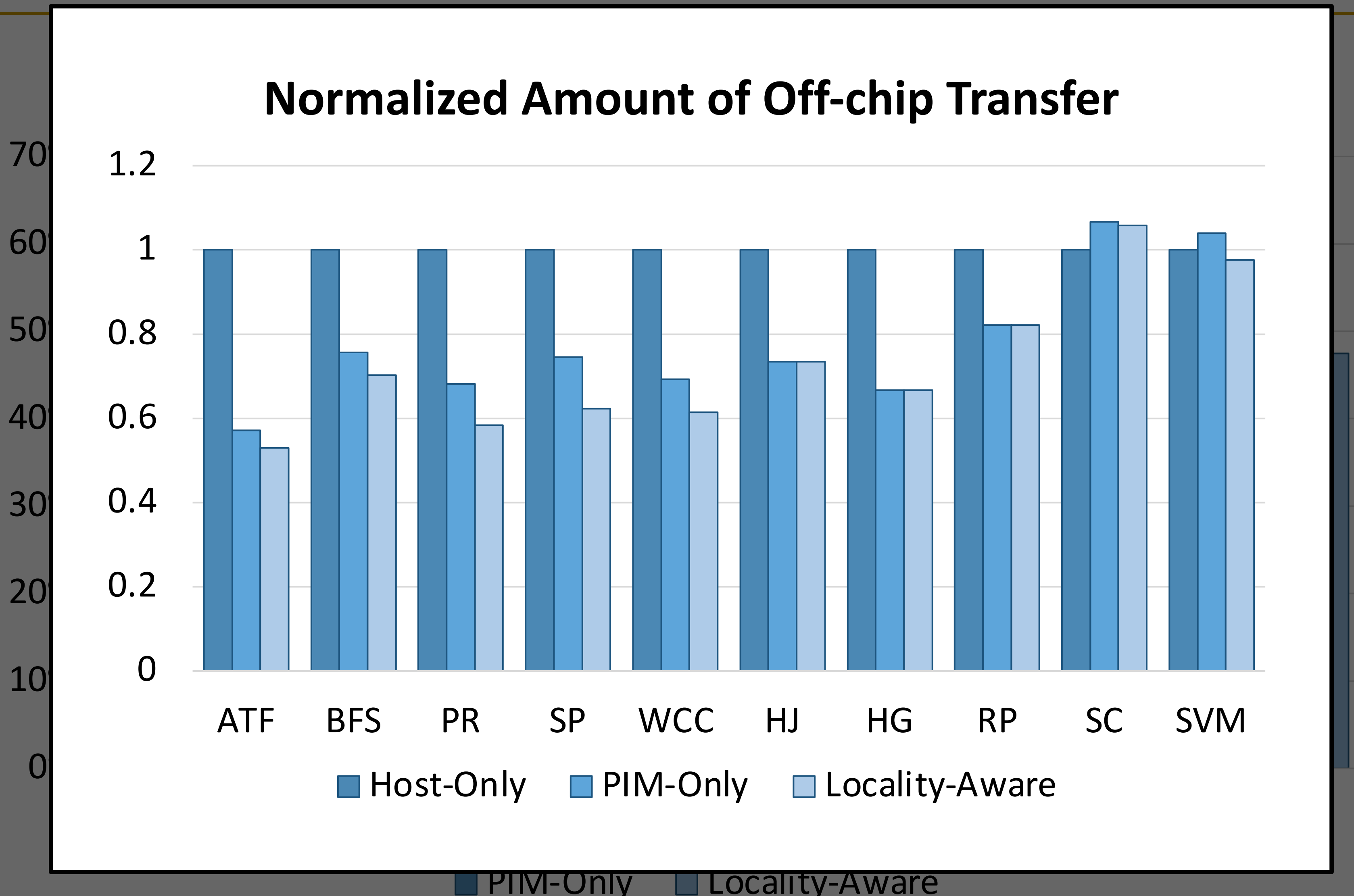
- Ten emerging data-intensive workloads
  - Large-scale graph processing
    - Average teenage follower, BFS, PageRank, single-source shortest path, weakly connected components
  - In-memory data analytics
    - Hash join, histogram, radix partitioning
  - Machine learning and data mining
    - Streamcluster, SVM-RFE

- Three input sets (small, medium, large) for each workload to show the impact of data locality

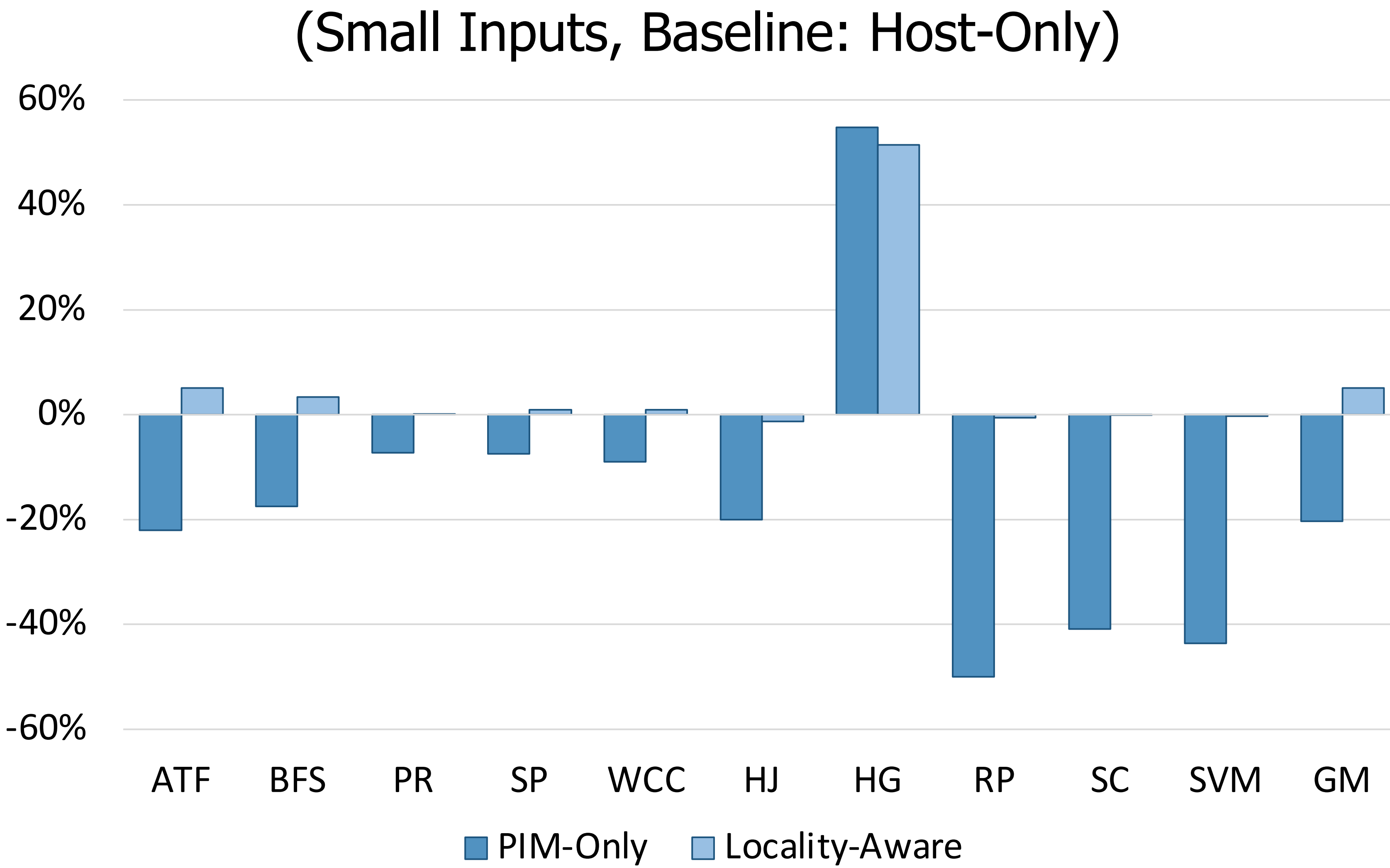
PEI Performance Delta: Large Data Sets



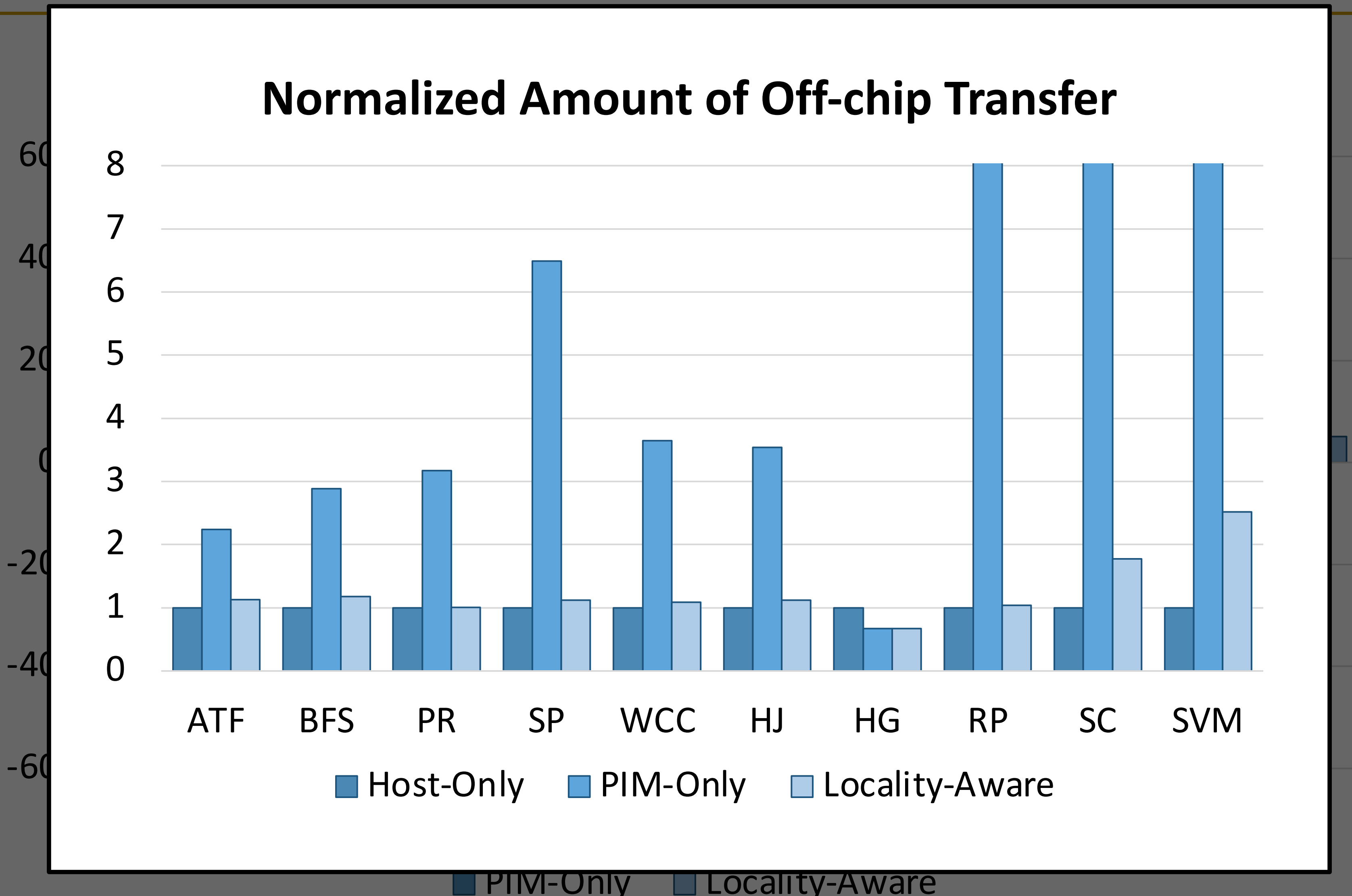
PEI Performance: Large Data Sets



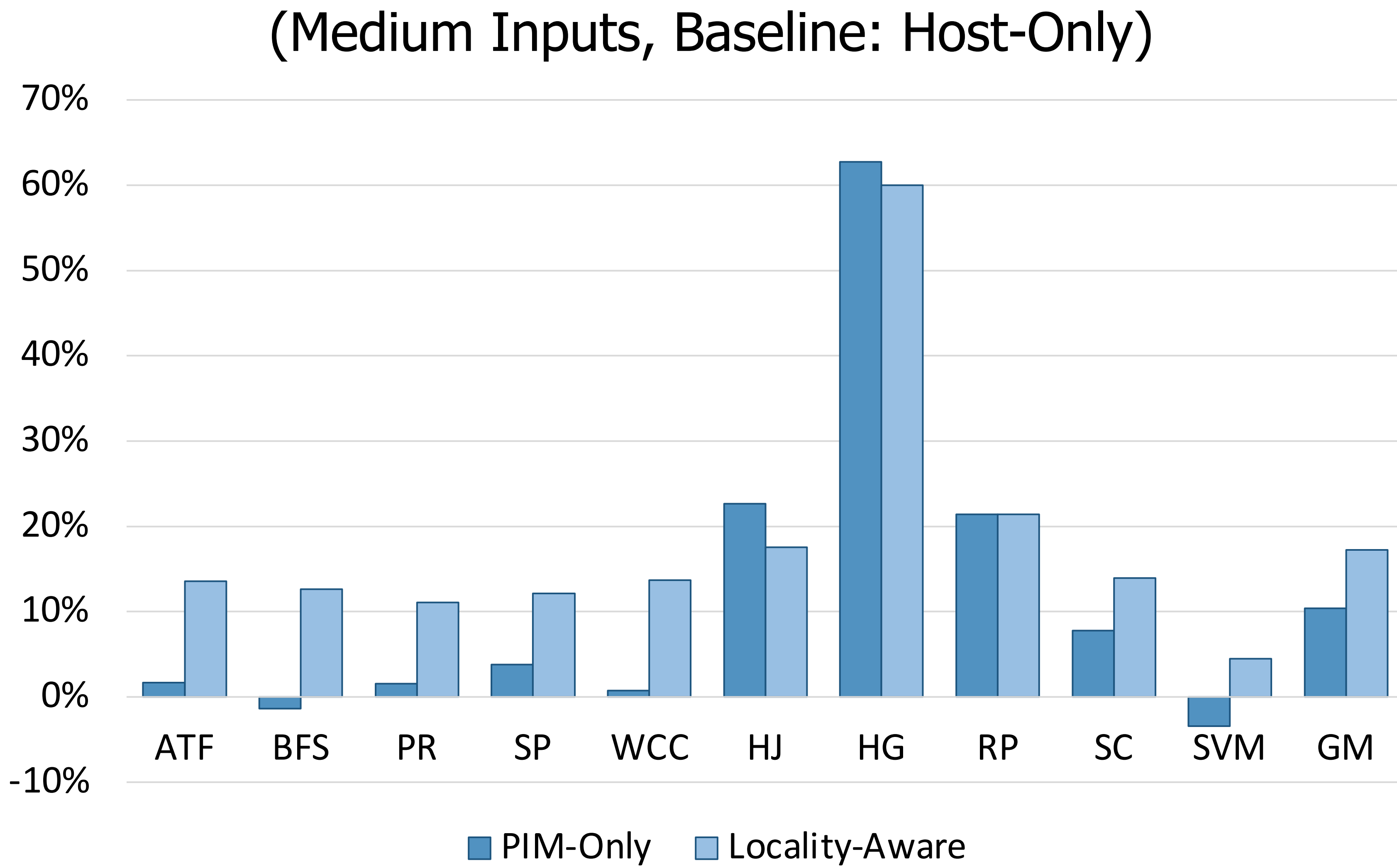
PEI Performance Delta: Small Data Sets



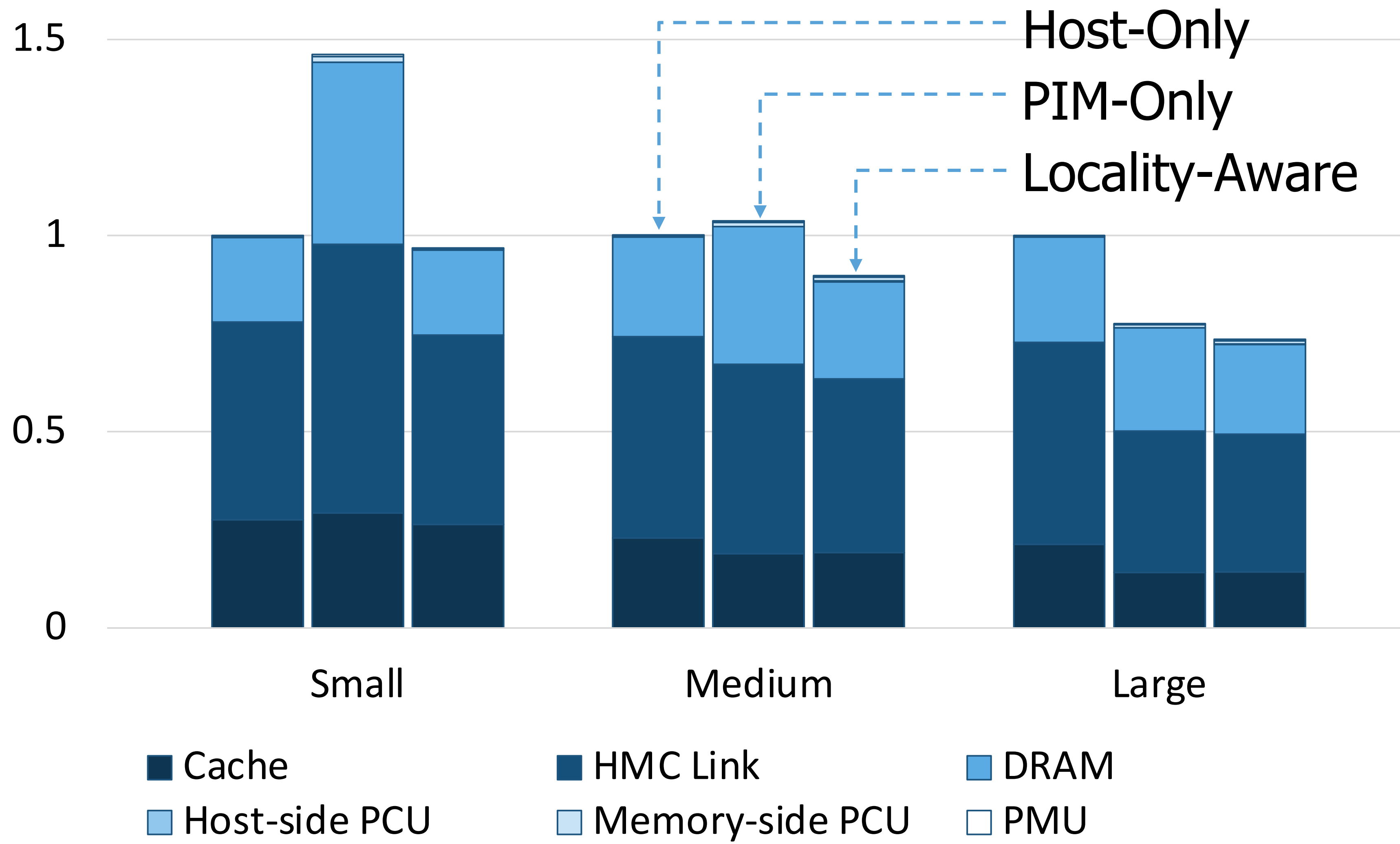
PEI Performance: Small Data Sets



PEI Performance Delta: Medium Data Sets



PEI Energy Consumption



PEI: Advantages & Disadvantages

- Advantages

  + Simple and low cost approach to PIM

  + No changes to programming model, virtual memory

  + Dynamically decides where to execute an instruction

- Disadvantages

  - Does not take full advantage of PIM potential

    - Single cache block restriction is limiting

Adoption: How to Keep It **Simple**?

- 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

Adoption: How to Ease **Programmability**? (I)

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

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

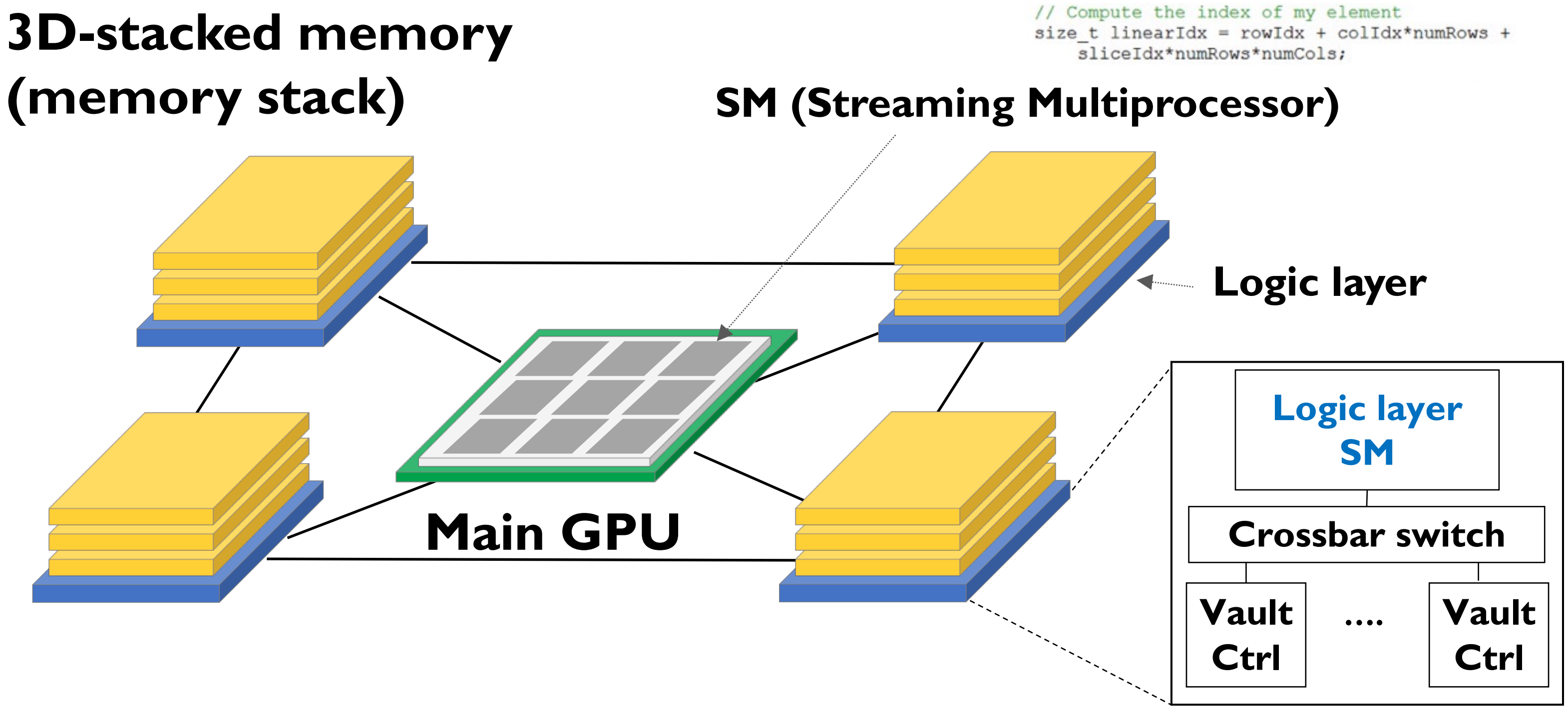
Kevin Hsieh[‡] Eiman Ebrahimi[†] Gwangsun Kim[*] Niladrish Chatterjee[†] Mike O'Connor[†]
Nandita Vijaykumar[‡] Onur Mutlu[§‡] Stephen W. Keckler[†]
[‡]Carnegie Mellon University [†]NVIDIA [*]KAIST [§]ETH Zürich

Truly Distributed GPU Processing with PIM

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

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

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



Adoption: How to Ease **Programmability**? (II)

- Geraldo F. Oliveira, Alain Kohli, David Novo, Juan Gómez-Luna, Onur Mutlu, **"DaPPA: A Data-Parallel Framework for Processing-in-Memory Architectures,"** in *PACT SRC Student Competition,* Vienna, Austria, October 2023.

DaPPA: A Data-Parallel Framework for Processing-in-Memory Architectures

Geraldo F. Oliveira[*] Alain Kohli[*] David Novo[‡] Juan Gómez-Luna[*] Onur Mutlu[*]

[*] *ETH Zürich* [‡] *LIRMM, Univ. Montpellier, CNRS*

Adoption: How to Ease **Programmability**? (III)

- Jinfan Chen, Juan Gómez-Luna, Izzat El Hajj, YuXin Guo, and Onur Mutlu,
"SimplePIM: A Software Framework for Productive and Efficient Processing in Memory"
*Proceedings of the 32nd International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, Vienna, Austria, October 2023.

SimplePIM: A Software Framework for Productive and Efficient Processing-in-Memory

Jinfan Chen[1] Juan Gómez-Luna[1] Izzat El Hajj[2] Yuxin Guo[1] Onur Mutlu[1]
[1]ETH Zürich [2]American University of Beirut

Adoption: How to Ease **Programmability**? (IV)

- Geraldo F. Oliveira, Juan Gomez-Luna, Lois Orosa, Saugata Ghose, Nandita Vijaykumar, Ivan fernandez, Mohammad Sadrosadati, and Onur Mutlu,
"DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks"
IEEE Access, 8 September 2021.
Preprint in ***arXiv***, 8 May 2021.
[arXiv preprint]
[IEEE Access version]
[DAMOV Suite and Simulator Source Code]
[SAFARI Live Seminar Video (2 hrs 40 mins)]
[Short Talk Video (21 minutes)]

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

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
LOIS OROSA, ETH Zürich, Switzerland
SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA
NANDITA VIJAYKUMAR, University of Toronto, Canada
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland
MOHAMMAD SADROSADATI, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

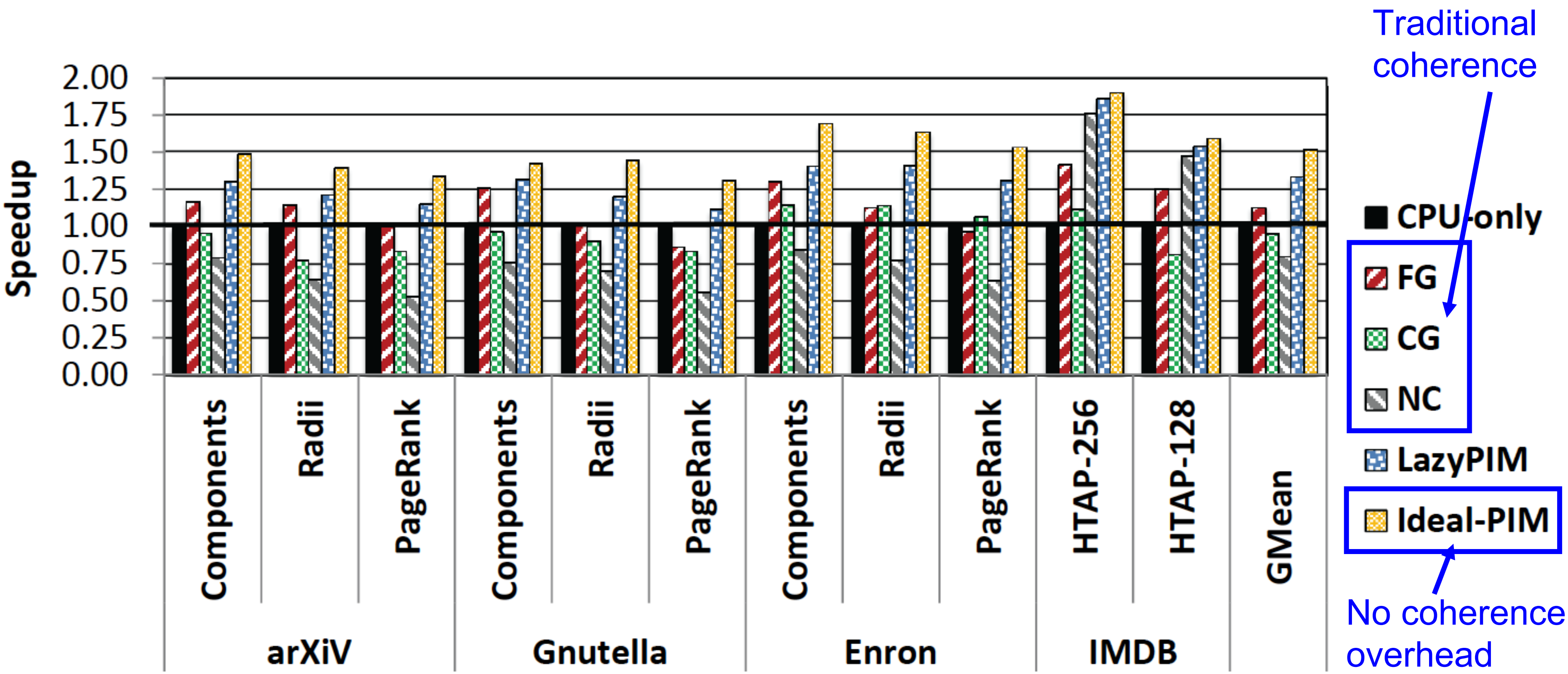
Adoption: How to Maintain **Coherence**? (I)

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

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

Amirali Boroumand[†], Saugata Ghose[†], Minesh Patel[†], Hasan Hassan[†§], Brandon Lucia[†],
Kevin Hsieh[†], Krishna T. Malladi[⋆], Hongzhong Zheng[⋆], and Onur Mutlu[‡†]
[†] *Carnegie Mellon University* [⋆] *Samsung Semiconductor, Inc.* [§] *TOBB ETÜ* [‡] *ETH Zürich*

Challenge: Coherence for Hybrid CPU-PIM Apps



Adoption: How to Maintain **Coherence?** (II)

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

CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators

Amirali Boroumand[†] Saugata Ghose[†] Minesh Patel[★] Hasan Hassan[★]
Brandon Lucia[†] Rachata Ausavarungnirun[†‡] Kevin Hsieh[†]
Nastaran Hajinazar[◇†] Krishna T. Malladi[§] Hongzhong Zheng[§] Onur Mutlu[★†]

[†]Carnegie Mellon University [★]ETH Zürich [‡]KMUTNB
[◇]Simon Fraser University [§]Samsung Semiconductor, Inc.

Adoption: How to Support **Synchronization?**

- Christina Giannoula, Nandita Vijaykumar, Nikela Papadopoulou, Vasileios Karakostas, Ivan Fernandez, Juan Gómez-Luna, Lois Orosa, Nectarios Koziris, Georgios Goumas, Onur Mutlu,
"SynCron: Efficient Synchronization Support for Near-Data-Processing Architectures"
*Proceedings of the 27th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Virtual, February-March 2021.
[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Talk Video (21 minutes)]
[Short Talk Video (7 minutes)]

SynCron: Efficient Synchronization Support for Near-Data-Processing Architectures

Christina Giannoula[†‡] Nandita Vijaykumar[*‡] Nikela Papadopoulou[†] Vasileios Karakostas[†] Ivan Fernandez[§‡]
Juan Gómez-Luna[‡] Lois Orosa[‡] Nectarios Koziris[†] Georgios Goumas[†] Onur Mutlu[‡]
[†]*National Technical University of Athens* [‡]*ETH Zürich* [*]*University of Toronto* [§]*University of Malaga*

Adoption: How to Support **Virtual Memory?**

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

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

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

Adoption: Evaluation Infrastructures

- Haocong Luo, Yahya Can Tugrul, F. Nisa Bostanci, Ataberk Olgun, A. Giray Yaglikci, and Onur Mutlu,
  "Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator"
  Preprint on ***arxiv***, August 2023.
  [arXiv version]
  [Ramulator 2.0 Source Code]

Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator

Haocong Luo, Yahya Can Tuğrul, F. Nisa Bostancı, Ataberk Olgun, A. Giray Yağlıkçı, and Onur Mutlu

https://arxiv.org/pdf/2308.11030.pdf

Methodologies, Workloads, and Tools for Processing-in-Memory: Enabling the Adoption of Data-Centric Architectures

Geraldo F. Oliveira and Onur Mutlu

geraldofojunior@gmail.com

https://geraldofojunior.github.io/

SAFARI

ETH Zürich

Processing-in-Memory:
Challenges

To fully support PIM systems, we need to develop:

1. **Workload characterization** methodologies and **benchmark suites** targeting PIM architectures
2. **Frameworks** that can facilitate the implementation of **complex operations and algorithms** using PIM primitives
3. **Compiler support** and **compiler optimizations** targeting PIM architectures
4. **Operating system support** for PIM-aware virtual memory, memory management, data allocation and mapping
5. **End-to-End System-on-Chip** Design Beyond DRAM

The lack of tools and system support for PIM architectures limit the adoption of PIM systems

An Example: SimplePIM Framework

- Jinfan Chen, Juan Gómez-Luna, Izzat El Hajj, YuXin Guo, and Onur Mutlu,
  "SimplePIM: A Software Framework for Productive and Efficient Processing in Memory"
  *Proceedings of the 32nd International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, Vienna, Austria, October 2023.

SimplePIM: A Software Framework for Productive and Efficient Processing-in-Memory

Jinfan Chen[1] Juan Gómez-Luna[1] Izzat El Hajj[2] Yuxin Guo[1] Onur Mutlu[1]
[1]ETH Zürich [2]American University of Beirut

Programming a PIM System

- PIM programming is challenging
  - Manage data movement between host DRAM and PIM DRAM
    - Parallel, serial, broadcast, and gather/scatter transfers
  - Manage data movement between PIM DRAM bank and scratchpad
    - 8-byte aligned and maximum of 2,048 bytes
  - Multithreaded programming model
  - Inter-thread synchronization
    - Barriers, handshakes, mutexes, and semaphores

Our Goal

Design a high-level programming framework that abstracts these hardware-specific complexities and provides a clean yet powerful interface for ease of use and high program performance

The SimplePIM Programming Framework

- SimplePIM provides standard abstractions to build and deploy applications on PIM systems
  - Management interface
    - Metadata for PIM-resident arrays
  - Communication interface
    - Abstractions for host-PIM and PIM-PIM communication
  - Processing interface
    - Iterators (`map`, `reduce`, `zip`) to implement workloads

Productivity Improvement (I)

- Example: Hand-optimized histogram with UPMEM SDK

```
... // Initialize global variables and functions for histogram
int main_kernel() {
  if (tasklet_id == 0)
    mem_reset(); // Reset the heap
  ... // Initialize variables and the histogram
  T *input_buff_A = (T*)mem_alloc(2048); // Allocate buffer in scratchpad memory

  for (unsigned int byte_index = base_tasklet; byte_index < input_size; byte_index += stride) {
    // Boundary checking
    uint32_t l_size_bytes = (byte_index + 2048 >= input_size) ? (input_size - byte_index) : 2048;
    // Load scratchpad with a DRAM block
    mram_read((const __mram_ptr void*)(mram_base_addr_A + byte_index), input_buff_A, l_size_bytes);
    // Histogram calculation
    histogram(hist, bins, input_buff_A, l_size_bytes/sizeof(uint32_t));
  }
  ...
  barrier_wait(&my_barrier); // Barrier to synchronize PIM threads
  ... // Merging histograms from different tasklets into one histo_dpu
  // Write result from scratchpad to DRAM
  if (tasklet_id == 0)
    if (bins * sizeof(uint32_t) <= 2048)
      mram_write(histo_dpu, (__mram_ptr void*)mram_base_addr_histo, bins * sizeof(uint32_t));
    else
      for (unsigned int offset = 0; offset < ((bins * sizeof(uint32_t)) >> 11); offset++) {
        mram_write(histo_dpu + (offset << 9), (__mram_ptr void*)(mram_base_addr_histo +
                   (offset << 11)), 2048);
      }
  return 0;
}
```

Productivity Improvement (II)

- Example: SimplePIM histogram

```
// Programmer-defined functions in the file "histo_filepath"
void init_func (uint32_t size, void* ptr) {
  char* casted_value_ptr = (char*) ptr;
  for (int i = 0; i < size; i++)
    casted_value_ptr[i] = 0;
}

void acc_func (void* dest, void* src) {
  *(uint32_t*)dest += *(uint32_t*)src;
}

void map_to_val_func (void* input, void* output, uint32_t* key) {
  uint32_t d = *((uint32_t*)input);
  *(uint32_t*)output = 1;
  *key = d * bins >> 12;
}

// Host side handle creation and iterator call
handle_t* handle = simple_pim_create_handle("histo_filepath", REDUCE, NULL, 0);

// Transfer (scatter) data to PIM, register as "t1"
simple_pim_array_scatter("t1", src, bins, sizeof(T), management);

// Run histogram on "t1" and produce "t2"
simple_pim_array_red("t1", "t2", sizeof(T), bins, handle, management);
```

Productivity Improvement (III)

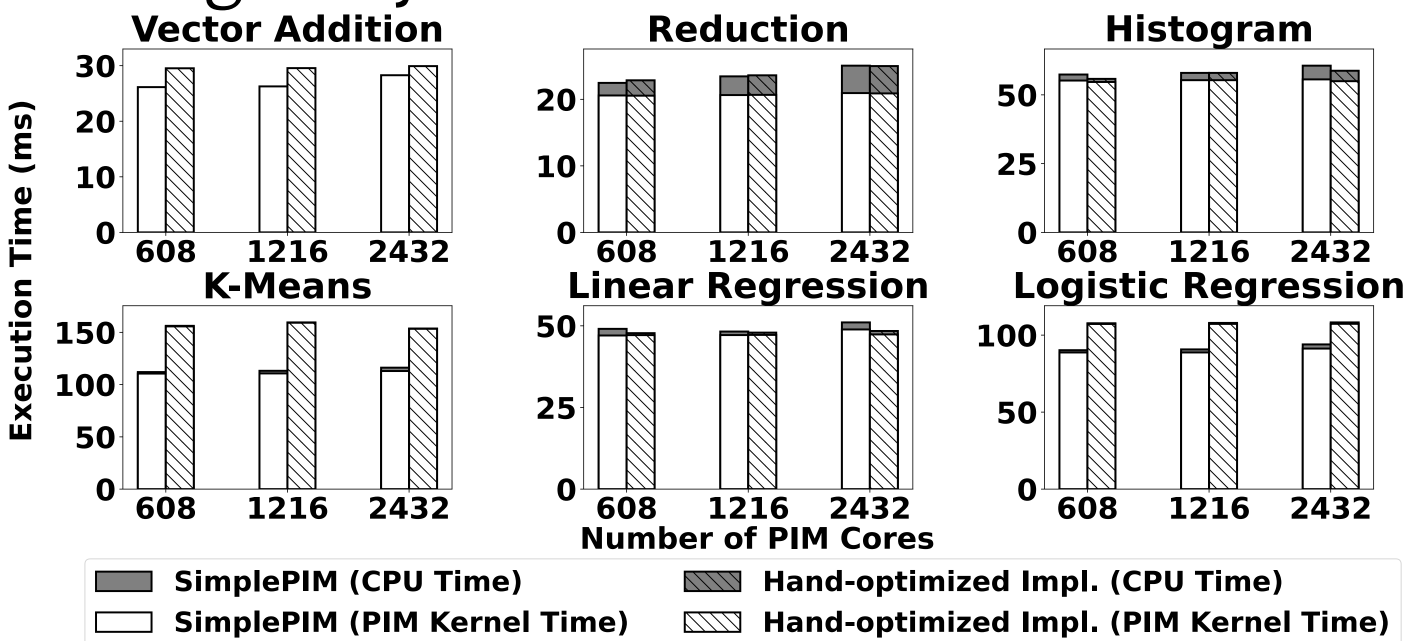
- Lines of code (LoC) reduction

	SimplePIM	Hand-optimized	LoC Reduction
Reduction	14	83	5.93×
Vector Addition	14	82	5.86×
Histogram	21	114	5.43×
Linear Regression	48	157	3.27×
Logistic Regression	59	176	2.98×
K-Means	68	206	3.03×

SimplePIM reduces the number of lines of effective code by a factor of 2.98× to 5.93×

Performance Evaluation

- Weak scaling analysis



SimplePIM achieves comparable performance for `reduction`, `histogram`, and `linear regression`

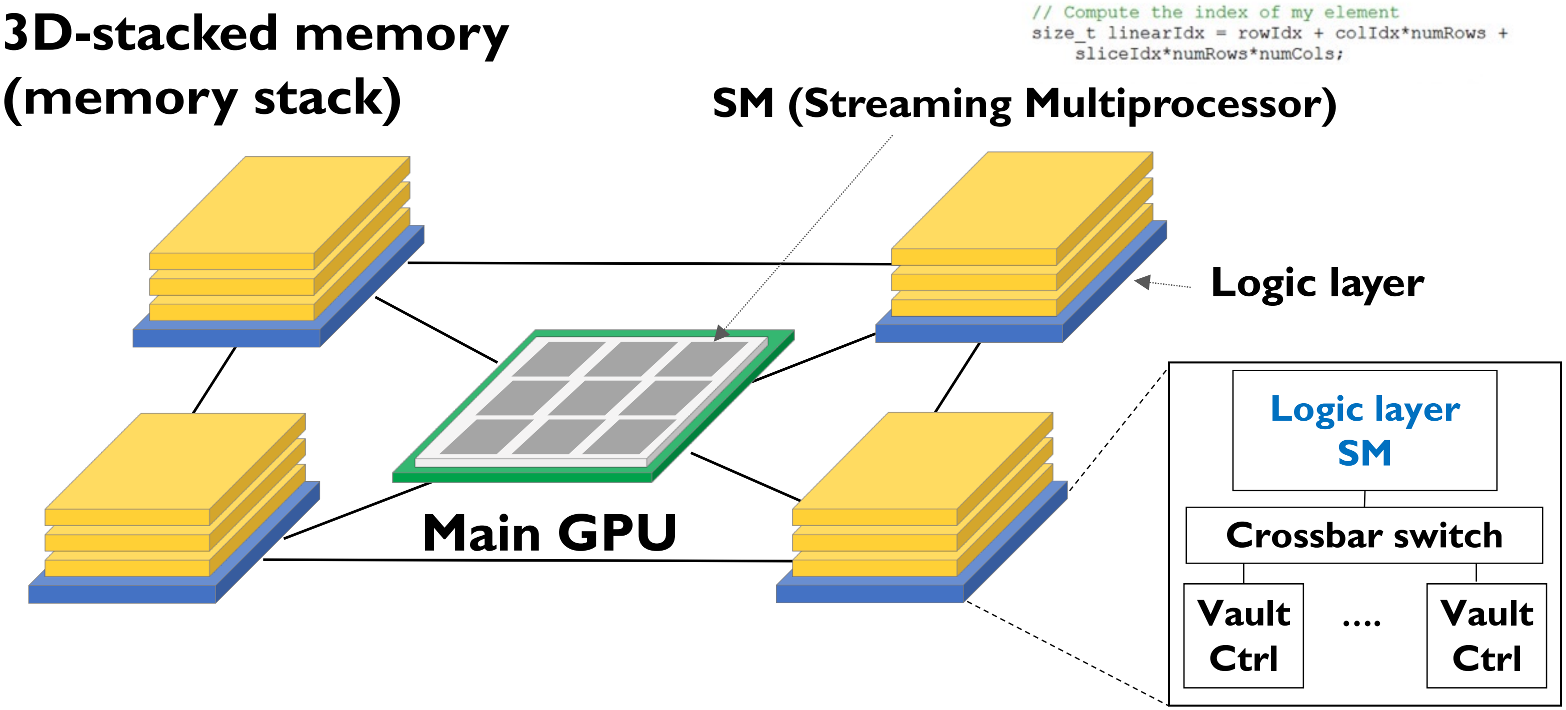
SimplePIM outperforms hand-optimized implementations for `vector addition`, `logistic regression`, and `k-means` by 10%-37%

Truly Distributed GPU Processing with PIM

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

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

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



Accelerating GPU Execution with PIM (I)

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

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

Kevin Hsieh[‡] Eiman Ebrahimi[†] Gwangsun Kim[*] Niladrish Chatterjee[†] Mike O'Connor[†]
Nandita Vijaykumar[‡] Onur Mutlu[§‡] Stephen W. Keckler[†]
[‡]**Carnegie Mellon University** [†]**NVIDIA** [*]**KAIST** [§]**ETH Zürich**

Accelerating GPU Execution with PIM (II)

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

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

Ashutosh Pattnaik[1] Xulong Tang[1] Adwait Jog[2] Onur Kayıran[3]
Asit K. Mishra[4] Mahmut T. Kandemir[1] Onur Mutlu[5,6] Chita R. Das[1]

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

Accelerating Linked Data Structures

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

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

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

Accelerating Dependent Cache Misses

- Milad Hashemi, Khubaib, Eiman Ebrahimi, Onur Mutlu, and Yale N. Patt, **"Accelerating Dependent Cache Misses with an Enhanced Memory Controller"**
Proceedings of the 43rd International Symposium on Computer Architecture (***ISCA***), Seoul, South Korea, June 2016.
[Slides (pptx) (pdf)]
[Lightning Session Slides (pptx) (pdf)]

Accelerating Dependent Cache Misses with an Enhanced Memory Controller

Milad Hashemi*, Khubaib†, Eiman Ebrahimi‡, Onur Mutlu§, Yale N. Patt*

**The University of Texas at Austin   †Apple   ‡NVIDIA   §ETH Zürich & Carnegie Mellon University*

Accelerating Runahead Execution

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

Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads

Milad Hashemi*, Onur Mutlu§, Yale N. Patt*

**The University of Texas at Austin* *§ETH Zürich*

Accelerating Climate Modeling

- Gagandeep Singh, Dionysios Diamantopoulos, Christoph Hagleitner, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal,
  "NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling"
  *Proceedings of the 30th International Conference on Field-Programmable Logic and Applications (**FPL**)*, Gothenburg, Sweden, September 2020.
  [Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (23 minutes)]
  Nominated for the Stamatis Vassiliadis Memorial Award.

NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling

Gagandeep Singh[a,b,c] Dionysios Diamantopoulos[c] Christoph Hagleitner[c] Juan Gómez-Luna[b]
Sander Stuijk[a] Onur Mutlu[b] Henk Corporaal[a]
[a]Eindhoven University of Technology [b]ETH Zürich [c]IBM Research Europe, Zurich

Accelerating Approximate String Matching

- Damla Senol Cali, Gurpreet S. Kalsi, Zulal Bingol, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu,
  "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"
  Proceedings of the 53rd International Symposium on Microarchitecture (***MICRO***), Virtual, October 2020.
  [Lighting Talk Video (1.5 minutes)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (18 minutes)]
  [Slides (pptx) (pdf)]

GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali[†⋈] Gurpreet S. Kalsi[⋈] Zülal Bingöl[▽] Can Firtina[◇] Lavanya Subramanian[‡] Jeremie S. Kim[◇†]
Rachata Ausavarungnirun[⊙] Mohammed Alser[◇] Juan Gomez-Luna[◇] Amirali Boroumand[†] Anant Nori[⋈]
Allison Scibisz[†] Sreenivas Subramoney[⋈] Can Alkan[▽] Saugata Ghose[⋆†] Onur Mutlu[◇†▽]

[†]*Carnegie Mellon University* [⋈]*Processor Architecture Research Lab, Intel Labs* [▽]*Bilkent University* [◇]*ETH Zürich*
[‡]*Facebook* [⊙]*King Mongkut's University of Technology North Bangkok* [⋆]*University of Illinois at Urbana–Champaign*

Accelerating Sequence-to-Graph Mapping

- Damla Senol Cali, Konstantinos Kanellopoulos, Joel Lindegger, Zulal Bingol, Gurpreet S. Kalsi, Ziyi Zuo, Can Firtina, Meryem Banu Cavlak, Jeremie Kim, Nika MansouriGhiasi, Gagandeep Singh, Juan Gomez-Luna, Nour Almadhoun Alserr, Mohammed Alser, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu,
  "SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping"
  *Proceedings of the 49th International Symposium on Computer Architecture (**ISCA**)*, New York, June 2022.
  [arXiv version]

SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping

Damla Senol Cali[1] Konstantinos Kanellopoulos[2] Joël Lindegger[2] Zülal Bingöl[3]
Gurpreet S. Kalsi[4] Ziyi Zuo[5] Can Firtina[2] Meryem Banu Cavlak[2] Jeremie Kim[2]
Nika Mansouri Ghiasi[2] Gagandeep Singh[2] Juan Gómez-Luna[2] Nour Almadhoun Alserr[2]
Mohammed Alser[2] Sreenivas Subramoney[4] Can Alkan[3] Saugata Ghose[6] Onur Mutlu[2]

[1]Bionano Genomics [2]ETH Zürich [3]Bilkent University [4]Intel Labs
[5]Carnegie Mellon University [6]University of Illinois Urbana-Champaign

 https://arxiv.org/pdf/2205.05883.pdf

Accelerating Basecalling + Read Mapping

- Haiyu Mao, Mohammed Alser, Mohammad Sadrosadati, Can Firtina, Akanksha Baranwal, Damla Senol Cali, Aditya Manglik, Nour Almadhoun Alserr, and Onur Mutlu,
  "GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping"
  *Proceedings of the 55th International Symposium on Microarchitecture (**MICRO**)*,
  Chicago, IL, USA, October 2022.
  [Slides (pptx) (pdf)]
  [Longer Lecture Slides (pptx) (pdf)]
  [Lecture Video (25 minutes)]
  [arXiv version]

GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping

Haiyu Mao[1] Mohammed Alser[1] Mohammad Sadrosadati[1] Can Firtina[1] Akanksha Baranwal[1]
Damla Senol Cali[2] Aditya Manglik[1] Nour Almadhoun Alserr[1] Onur Mutlu[1]
[1]*ETH Zürich* [2]*Bionano Genomics*

Accelerating Time Series Analysis

- Ivan Fernandez, Ricardo Quislant, Christina Giannoula, Mohammed Alser, Juan Gómez-Luna, Eladio Gutiérrez, Oscar Plata, and Onur Mutlu,
  "NATSA: A Near-Data Processing Accelerator for Time Series Analysis"
  *Proceedings of the 38th IEEE International Conference on Computer Design (**ICCD**)*, Virtual, October 2020.
  [Slides (pptx) (pdf)]
  [Talk Video (10 minutes)]
  [Source Code]

NATSA: A Near-Data Processing Accelerator for Time Series Analysis

Ivan Fernandez[§] Ricardo Quislant[§] Christina Giannoula[†] Mohammed Alser[‡]
Juan Gómez-Luna[‡] Eladio Gutiérrez[§] Oscar Plata[§] Onur Mutlu[‡]

[§]*University of Malaga* [†]*National Technical University of Athens* [‡]*ETH Zürich*

Accelerating Graph Pattern Mining

- Maciej Besta, Raghavendra Kanakagiri, Grzegorz Kwasniewski, Rachata Ausavarungnirun, Jakub Beránek, Konstantinos Kanellopoulos, Kacper Janda, Zur Vonarburg-Shmaria, Lukas Gianinazzi, Ioana Stefan, Juan Gómez-Luna, Marcin Copik, Lukas Kapp-Schwoerer, Salvatore Di Girolamo, Nils Blach, Marek Konieczny, Onur Mutlu, and Torsten Hoefler,
  "SISA: Set-Centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems"
  *Proceedings of the 54th International Symposium on Microarchitecture (**MICRO**)*, Virtual, October 2021.
  [Slides (pdf)]
  [Talk Video (22 minutes)]
  [Lightning Talk Video (1.5 minutes)]
  [Full arXiv version]

SISA: Set-Centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems

Maciej Besta[1], Raghavendra Kanakagiri[2], Grzegorz Kwasniewski[1], Rachata Ausavarungnirun[3], Jakub Beránek[4], Konstantinos Kanellopoulos[1], Kacper Janda[5], Zur Vonarburg-Shmaria[1], Lukas Gianinazzi[1], Ioana Stefan[1], Juan Gómez-Luna[1], Marcin Copik[1], Lukas Kapp-Schwoerer[1], Salvatore Di Girolamo[1], Nils Blach[1], Marek Konieczny[5], Onur Mutlu[1], Torsten Hoefler[1]

[1]ETH Zurich, Switzerland [2]IIT Tirupati, India [3]King Mongkut's University of Technology North Bangkok, Thailand [4]Technical University of Ostrava, Czech Republic [5]AGH-UST, Poland

Accelerating HTAP Database Systems

- Amirali Boroumand, Saugata Ghose, Geraldo F. Oliveira, and Onur Mutlu,
  "Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design"
  Proceedings of the 38th International Conference on Data Engineering (***ICDE***), Virtual, May 2022.
  [arXiv version]
  [Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]

Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design

Amirali Boroumand[†] Saugata Ghose[◇] Geraldo F. Oliveira[‡] Onur Mutlu[‡]
[†]*Google* [◇]*Univ. of Illinois Urbana-Champaign* [‡]*ETH Zürich*

Mensa: Highly-Efficient ML Inference

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,
"Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks"
*Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, Virtual, September 2021.
[Slides (pptx) (pdf)]
[Talk Video (14 minutes)]

Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand[†◇] Saugata Ghose[‡] Berkin Akin[§] Ravi Narayanaswami[§]
Geraldo F. Oliveira[⋆] Xiaoyu Ma[§] Eric Shiu[§] Onur Mutlu[⋆†]

[†]*Carnegie Mellon Univ.* [◇]*Stanford Univ.* [‡]*Univ. of Illinois Urbana-Champaign* [§]*Google* [⋆]*ETH Zürich*

Accelerating Data-Intensive Workloads

- 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

FPGA-based Processing Near Memory

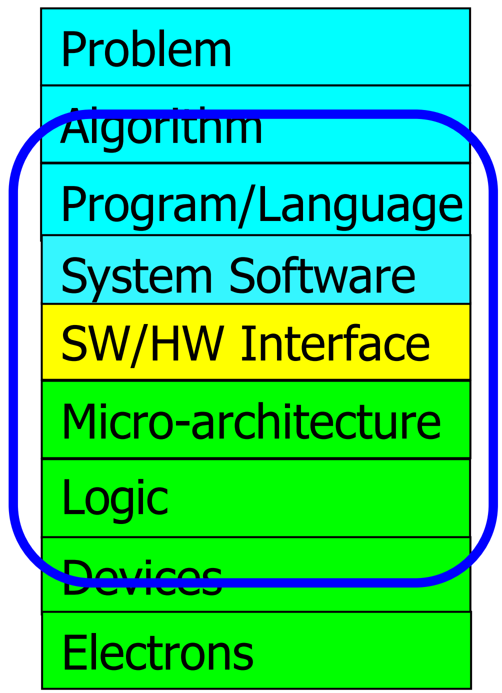
- Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu,
"FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications"
IEEE Micro (***IEEE MICRO***), 2021.

FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh[◇] **Mohammed Alser**[◇] **Damla Senol Cali**[⋈]
Dionysios Diamantopoulos[▽] **Juan Gómez-Luna**[◇]
Henk Corporaal[⋆] **Onur Mutlu**[◇⋈]

[◇]*ETH Zürich* [⋈]*Carnegie Mellon University*
[⋆]*Eindhoven University of Technology* [▽]*IBM Research Europe*

We Need to Revisit the Entire Stack



We can get there step by step

Processing-in-Memory
in the Real World

PIM Tutorial at ISCA 2024

ISCA 2024 Memory-Centric Computing Systems Tutorial

Saturday, June 29, Buenos Aires, Argentina

Organizers: Geraldo F. Oliveira, Dr. Mohammad Sadrosadati, Ataberk Olgun, Professor Onur Mutlu

Program: https://events.safari.ethz.ch/isca24-memorycentric-tutorial/

- Overview of PIM | PIM taxonomy
- PIM in memory & storage
- Real-world PNM systems
- PUM for bulk bitwise operations
- Programming techniques & tools
- Infrastructures for PIM Research
- Research challenges & opportunities

https://www.youtube.com/watch?v=KV2MXvcBgb0

https://events.safari.ethz.ch/isca24-memorycentric-tutorial

PIM Tutorials [MICRO'23, ISCA'23, ASPLOS'23, HPCA'23, ISCA'24]

- Lectures + Hands-on labs + Invited talks

ISCA 2023 Real-World PIM Tutorial

Search

Recent Changes Media Manager Sitemap

Trace: • start

start

Real-world Processing-in-Memory Systems for Modern Workloads

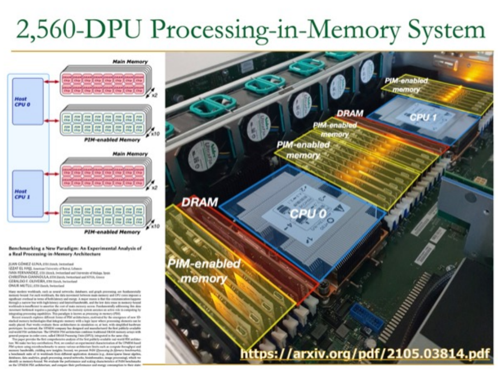
Tutorial Description

Table of Contents

Processing-in-Memory (PIM) is a computing paradigm that aims at overcoming the data movement bottleneck (i.e., the waste of execution cycles and energy resulting from the back-and-forth data movement between memory units and compute units) by making memory compute-capable.

Explored over several decades since the 1960s, PIM systems are becoming a reality with the advent of the first commercial products and prototypes.

A number of startups (e.g., UPMEM, Neuroblade) are already commercializing real PIM hardware, each with its own design approach and target applications. Several major vendors (e.g., Samsung, SK Hynix, Alibaba) have presented real PIM chip prototypes in the last two years. Most of these architectures have in common that they place compute units near the memory arrays. This type of PIM is called processing near memory (PNM).



PIM can provide large improvements in both performance and energy consumption for many modern applications, thereby enabling a commercially viable way of dealing with huge amounts of data that is bottlenecking our computing systems. Yet, it is critical to (1) study and understand the characteristics that make a workload suitable for a PIM architecture, (2) propose optimization strategies for PIM kernels, and (3) develop programming frameworks and tools that can lower the learning curve and ease the adoption of PIM.

This tutorial focuses on the latest advances in PIM technology, workload characterization for PIM, and programming and optimizing PIM kernels. We will (1) provide an introduction to PIM and taxonomy of PIM systems, (2) give an overview and a rigorous analysis of existing real-world PIM hardware, (3) conduct hand-on labs about important workloads (machine learning, sparse linear algebra, bioinformatics, etc.) using real PIM systems, and (4) shed light on how to improve future PIM systems for such workloads.

https://www.youtube.com/live/GIb5EgSrWk0

https://events.safari.ethz.ch/isca-pim-tutorial/

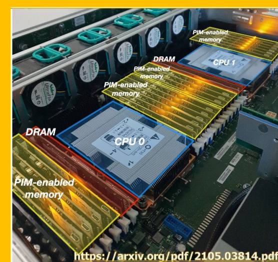
Real PIM Tutorial [ISCA 2023]

- June 18: Lectures + Hands-on labs + Invited talks

ISCA 2023 Real-World PIM Tutorial
Sunday, June 18, Orlando, Florida

Organizers: Juan Gómez Luna, Onur Mutlu, Ataberk Olgun
Program: https://events.safari.ethz.ch/isca-pim-tutorial/

Overview PIM | PNM | UPMEM PIM | PNM for neural networks | PNM for recommender systems | PNM for ML workloads | How to enable PIM? | PUM prototypes
Hands-on Labs: Benchmarking | Accelerating real-world workloads



Tutorial Materials

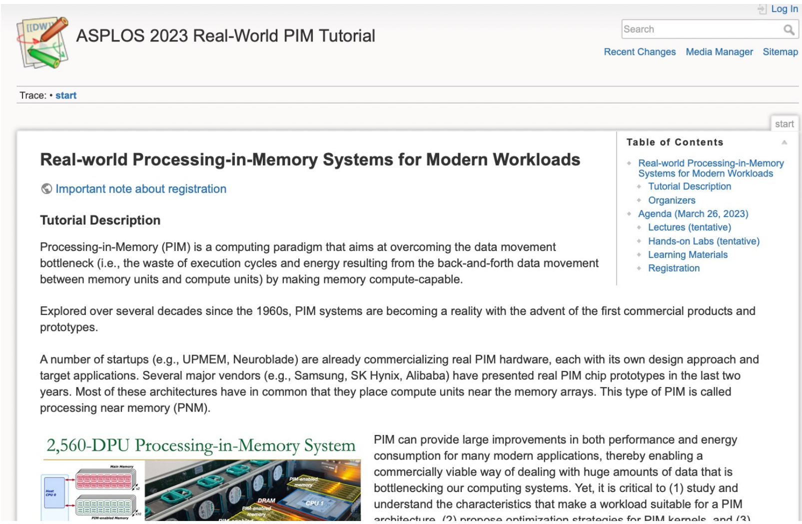
Time	Speaker	Title	Materials
8:55am-9:00am	Dr. Juan Gómez Luna	Welcome & Agenda	(PDF) (PPT)
9:00am-10:20am	Prof. Onur Mutlu	Memory-Centric Computing	(PDF) (PPT)
10:20am-11:00am	Dr. Juan Gómez Luna	Processing-Near-Memory: Real PNM Architectures / Programming General-purpose PIM	(PDF) (PPT)
11:20am-11:50am	Prof. Izzat El Hajj	High-throughput Sequence Alignment using Real Processing-in-Memory Systems	(PDF) (PPT)
11:50am-12:30pm	Dr. Christina Giannoula	SparseP: Towards Efficient Sparse Matrix Vector Multiplication for Real Processing-In-Memory Systems	(PDF) (PPT)
2:00pm-2:45pm	Dr. Sukhan Lee	Introducing Real-world HBM-PIM Powered System for Memory-bound Applications	(PDF) (PPT)
2:45pm-3:30pm	Dr. Juan Gómez Luna / Ataberk Olgun	Processing-Using-Memory: Exploiting the Analog Operational Properties of Memory Components / PUM Prototypes: PiDRAM	(PDF) (PPT) (PDF) (PPT)
4:00pm-4:40pm	Dr. Juan Gómez Luna	Accelerating Modern Workloads on a General-purpose PIM System	(PDF) (PPT)
4:40pm-5:20pm	Dr. Juan Gómez Luna	Adoption Issues: How to Enable PIM?	(PDF) (PPT)
5:20pm-5:30pm	Dr. Juan Gómez Luna	Hands-on Lab: Programming and Understanding a Real Processing-in-Memory Architecture	(Handout) (PDF) (PPT)

https://www.youtube.com/live/GIb5EgSrWk0

https://events.safari.ethz.ch/isca-pim-tutorial/

Real PIM Tutorial [ASPLOS 2023]

- March 26: Lectures + Hands-on labs + Invited talks



Log In

ASPLOS 2023 Real-World PIM Tutorial

Search

Recent Changes Media Manager Sitemap

Trace: • start

start

Real-world Processing-in-Memory Systems for Modern Workloads

Important note about registration

Tutorial Description

Processing-in-Memory (PIM) is a computing paradigm that aims at overcoming the data movement bottleneck (i.e., the waste of execution cycles and energy resulting from the back-and-forth data movement between memory units and compute units) by making memory compute-capable.

Explored over several decades since the 1960s, PIM systems are becoming a reality with the advent of the first commercial products and prototypes.

A number of startups (e.g., UPMEM, Neuroblade) are already commercializing real PIM hardware, each with its own design approach and target applications. Several major vendors (e.g., Samsung, SK Hynix, Alibaba) have presented real PIM chip prototypes in the last two years. Most of these architectures have in common that they place compute units near the memory arrays. This type of PIM is called processing near memory (PNM).

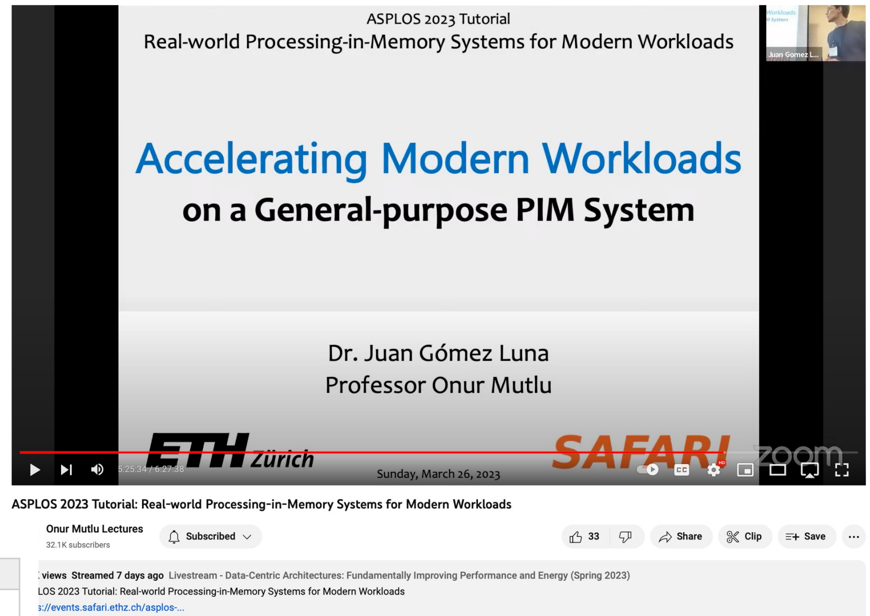
2,560-DPU Processing-in-Memory System

PIM can provide large improvements in both performance and energy consumption for many modern applications, thereby enabling a commercially viable way of dealing with huge amounts of data that is bottlenecking our computing systems. Yet, it is critical to (1) study and understand the characteristics that make a workload suitable for a PIM architecture, (2) propose optimization strategies for PIM kernels, and (3)

Table of Contents

Tutorial Materials

Time	Speaker	Title	Materials
9:00am-10:20am	Prof. Onur Mutlu	Memory-Centric Computing	(PDF) (PPT)
10:40am-12:00pm	Dr. Juan Gómez Luna	Processing-Near-Memory: Real PNM Architectures Programming General-purpose PIM	(PDF) (PPT)
1:40pm-2:20pm	Prof. Alexandra (Sasha) Fedorova (UBC)	Processing in Memory in the Wild	(PDF) (PPT)
2:20pm-3:20pm	Dr. Juan Gómez Luna & Ataberk Olgun	Processing-Using-Memory: Exploiting the Analog Operational Properties of Memory Components	(PDF) (PPT) (PDF) (PPT)
3:40pm-4:10pm	Dr. Juan Gómez Luna	Adoption issues: How to enable PIM? Accelerating Modern Workloads on a General-purpose PIM System	(PDF) (PPT) (PDF) (PPT)
4:10pm-4:50pm	Dr. Yongkee Kwon & Eddy (Chanwook) Park (SK Hynix)	System Architecture and Software Stack for GDDR6-AiM	(PDF) (PPT)
4:50pm-5:00pm	Dr. Juan Gómez Luna	Hands-on Lab: Programming and Understanding a Real Processing-in-Memory Architecture	(Handout) (PDF) (PPT)

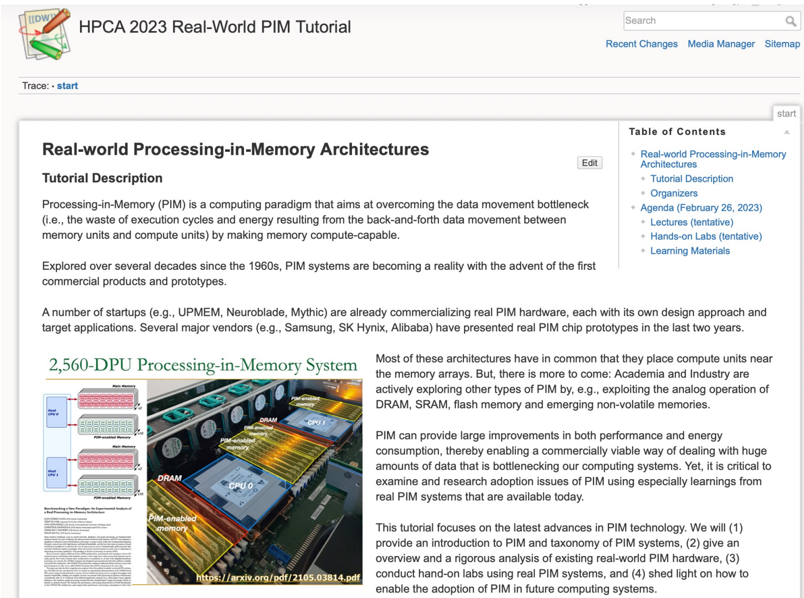


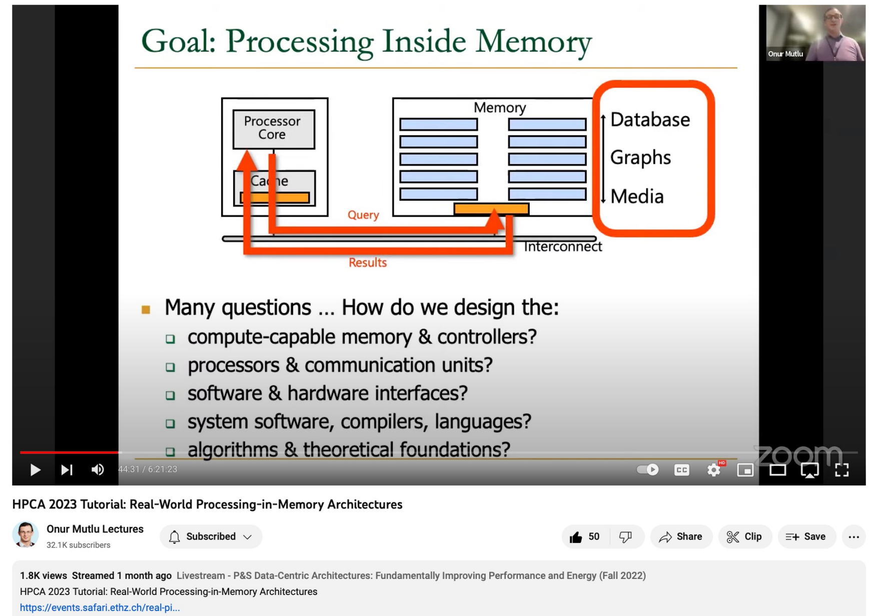
https://www.youtube.com/watch?v=oYCaLcT0Kmo

https://events.safari.ethz.ch/asplos-pim-tutorial/

Real PIM Tutorial [HPCA 2023]

- February 26: Lectures + Hands-on labs + Invited Talks





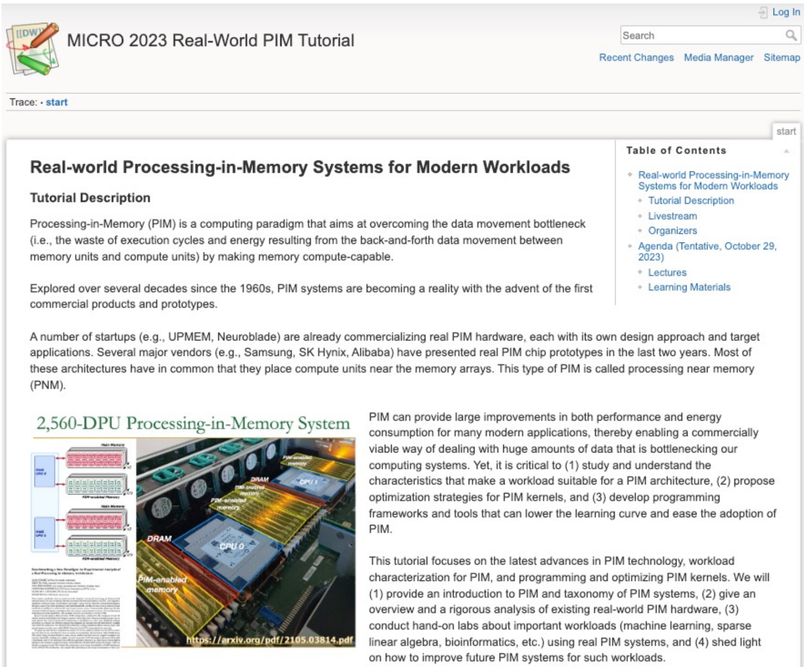
Time	Speaker	Title	Materials
8:00am-8:40am	Prof. Onur Mutlu	Memory-Centric Computing	(PDF) (PPT)
8:40am-10:00am	Dr. Juan Gómez Luna	Processing-Near-Memory: Real PNM Architectures Programming General-purpose PIM	(PDF) (PPT)
10:20am-11:00am	Dr. Dimin Niu	A 3D Logic-to-DRAM Hybrid Bonding Process-Near-Memory Chip for Recommendation System	
11:00am-11:40am	Dr. Christina Giannoula	SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures	(PDF) (PPT)
1:30pm-2:10pm	Dr. Juan Gómez Luna	Processing-Using-Memory: Exploiting the Analog Operational Properties of Memory Components	(PDF) (PPT)
2:10pm-2:50pm	Dr. Manuel Le Gallo	Deep Learning Inference Using Computational Phase-Change Memory	
2:50pm-3:30pm	Dr. Juan Gómez Luna	PIM Adoption Issues: How to Enable PIM Adoption?	(PDF) (PPT)
3:40pm-5:40pm	Dr. Juan Gómez Luna	Hands-on Lab: Programming and Understanding a Real Processing-in-Memory Architecture	(Handout) (PDF) (PPT)

https://www.youtube.com/watch?v=f5-nT1tbz5w

https://events.safari.ethz.ch/real-pim-tutorial/

Real PIM Tutorial [MICRO 2023]

- **October 29**: Lectures + Hands-on labs + Invited talks



Log In

MICRO 2023 Real-World PIM Tutorial

Search

Recent Changes Media Manager Sitemap

Trace: • start

start

Real-world Processing-in-Memory Systems for Modern Workloads

Tutorial Description

Processing-in-Memory (PIM) is a computing paradigm that aims at overcoming the data movement bottleneck (i.e., the waste of execution cycles and energy resulting from the back-and-forth data movement between memory units and compute units) by making memory compute-capable.

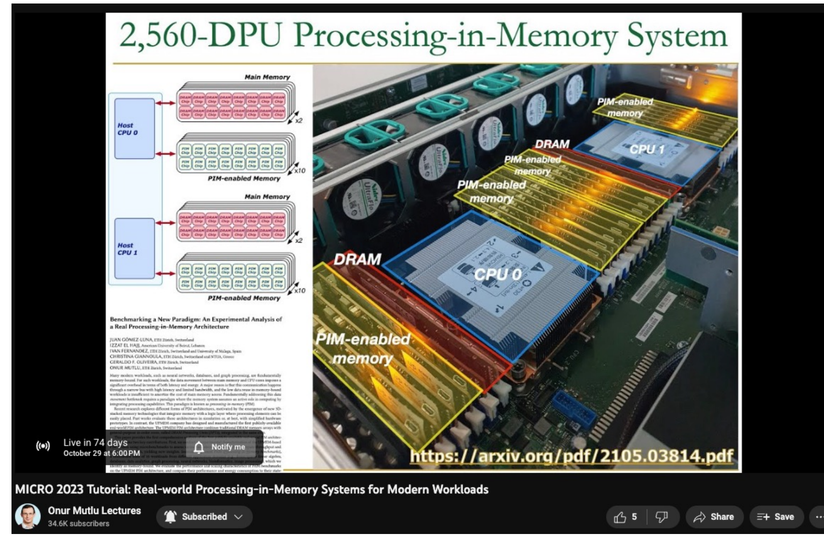
Explored over several decades since the 1960s, PIM systems are becoming a reality with the advent of the first commercial products and prototypes.

A number of startups (e.g., UPMEM, Neuroblade) are already commercializing real PIM hardware, each with its own design approach and target applications. Several major vendors (e.g., Samsung, SK Hynix, Alibaba) have presented real PIM chip prototypes in the last two years. Most of these architectures have in common that they place compute units near the memory arrays. This type of PIM is called processing near memory (PNM).

PIM can provide large improvements in both performance and energy consumption for many modern applications, thereby enabling a commercially viable way of dealing with huge amounts of data that is bottlenecking our computing systems. Yet, it is critical to (1) study and understand the characteristics that make a workload suitable for a PIM architecture, (2) propose optimization strategies for PIM kernels, and (3) develop programming frameworks and tools that can lower the learning curve and ease the adoption of PIM.

This tutorial focuses on the latest advances in PIM technology, workload characterization for PIM, and programming and optimizing PIM kernels. We will (1) provide an introduction to PIM and taxonomy of PIM systems, (2) give an overview and a rigorous analysis of existing real-world PIM hardware, (3) conduct hands-on labs about important workloads (machine learning, sparse linear algebra, bioinformatics, etc.) using real PIM systems, and (4) shed light on how to improve future PIM systems for such workloads.

Table of Contents



Agenda (Tentative, October 29, 2023)

Lectures

1. Introduction: PIM as a paradigm to overcome the data movement bottleneck.
2. PIM taxonomy: PNM (processing near memory) and PUM (processing using memory).
3. General-purpose PNM: UPMEM PIM.
4. PNM for neural networks: Samsung HBM-PIM, SK Hynix AiM.
5. PNM for recommender systems: Samsung AxDIMM, Alibaba PNM.
6. PUM prototypes: PiDRAM, SRAM-based PUM, Flash-based PUM.
7. Other approaches: Neuroblade, Mythic.
8. Adoption issues: How to enable PIM?
9. Hands-on labs: Programming a real PIM system.

https://www.youtube.com/watch?v=ohUooNSIxOI

https://events.safari.ethz.ch/micro-pim-tutorial

FPGA-based Processing Near Memory

- Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu,
"FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications"
IEEE Micro ***(IEEE MICRO)***, 2021.

FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh[◇] **Mohammed Alser**[◇] **Damla Senol Cali**[⋈]
Dionysios Diamantopoulos[▽] **Juan Gómez-Luna**[◇]
Henk Corporaal[⋆] **Onur Mutlu**[◇⋈]

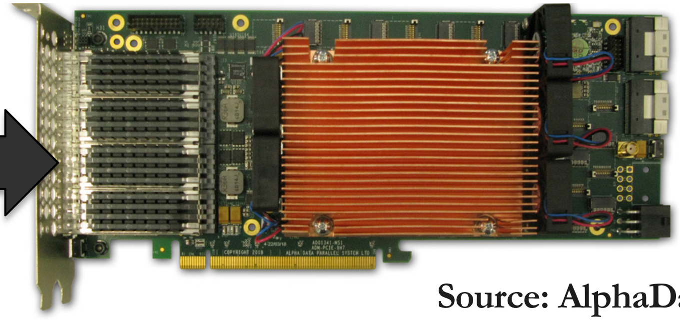
[◇]*ETH Zürich* [⋈]*Carnegie Mellon University*
[⋆]*Eindhoven University of Technology* [▽]*IBM Research Europe*

Near-Memory Acceleration using FPGAs

Source: IBM

IBM POWER9 CPU

OCAPI



Source: AlphaData

HBM-based FPGA board

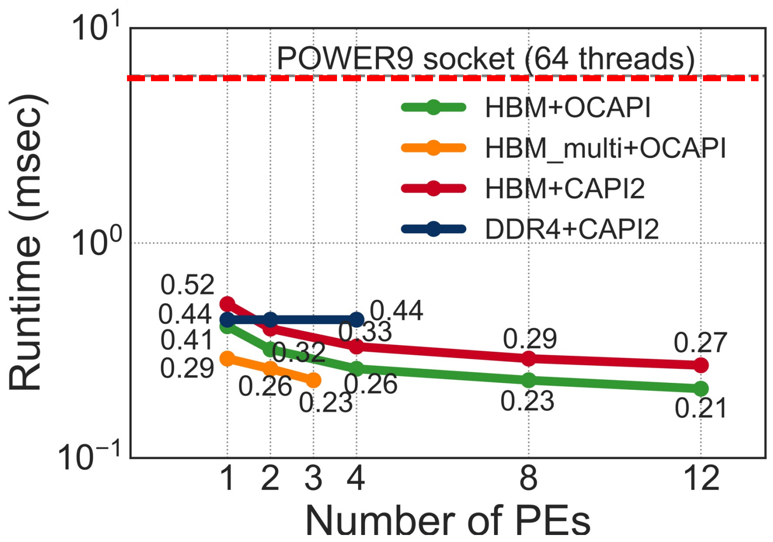
Near-HBM FPGA-based accelerator

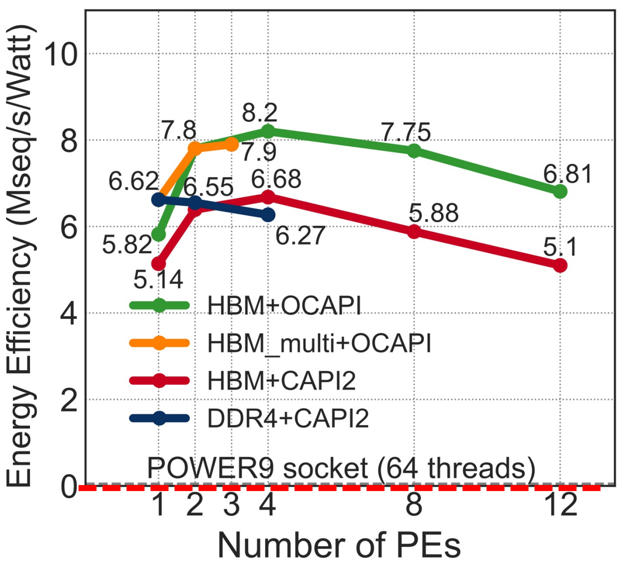
Two communication technologies: **CAPI2** and **OCAPI**

Two memory technologies: **DDR4** and **HBM**

Two workloads: **Weather Modeling** and **Genome Analysis**

Performance & Energy Greatly Improve





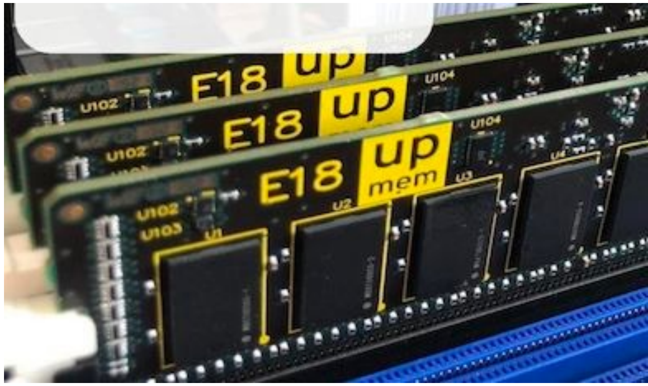
5-27× performance vs. a 16-core (64-thread) IBM POWER9 CPU

12-133× energy efficiency vs. a 16-core (64-thread) IBM POWER9 CPU

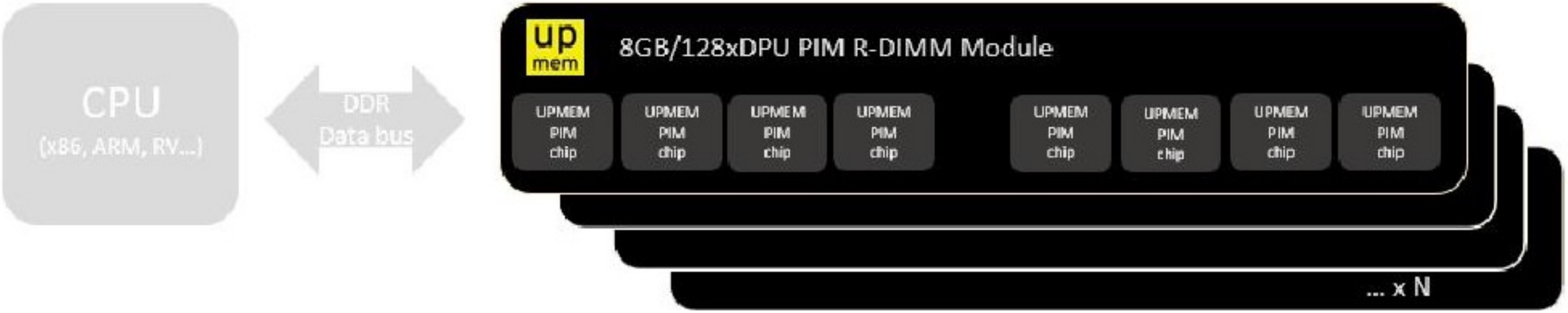
HBM alleviates memory bandwidth contention vs. DDR4

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/

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

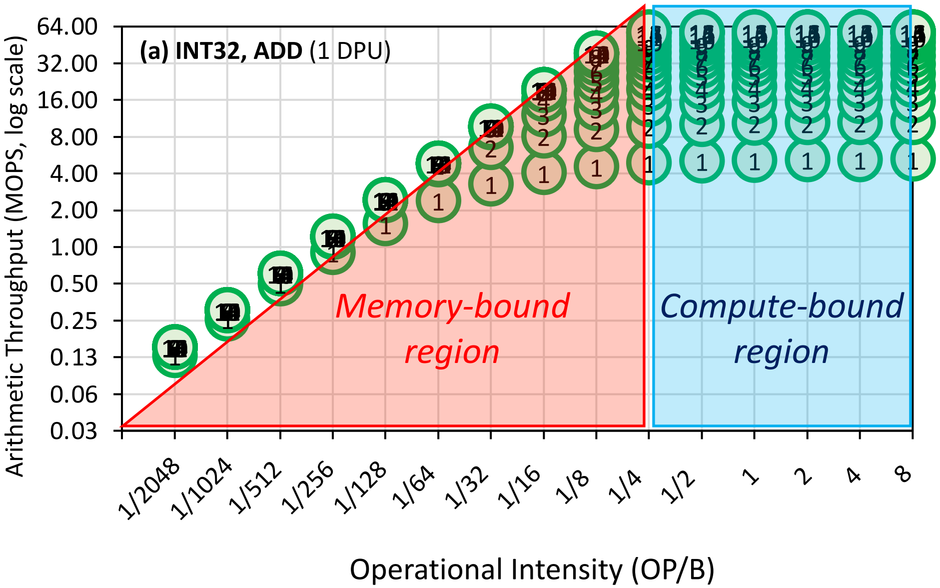
Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens 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.

https://arxiv.org/pdf/2105.03814.pdf

Key Takeaway 1



The throughput saturation point is as low as ¼ OP/B, i.e., 1 integer addition per every 32-bit element fetched

KEY TAKEAWAY 1

The UPMEM PIM architecture is fundamentally compute bound. As a result, **the most suitable workloads are memory-bound.**

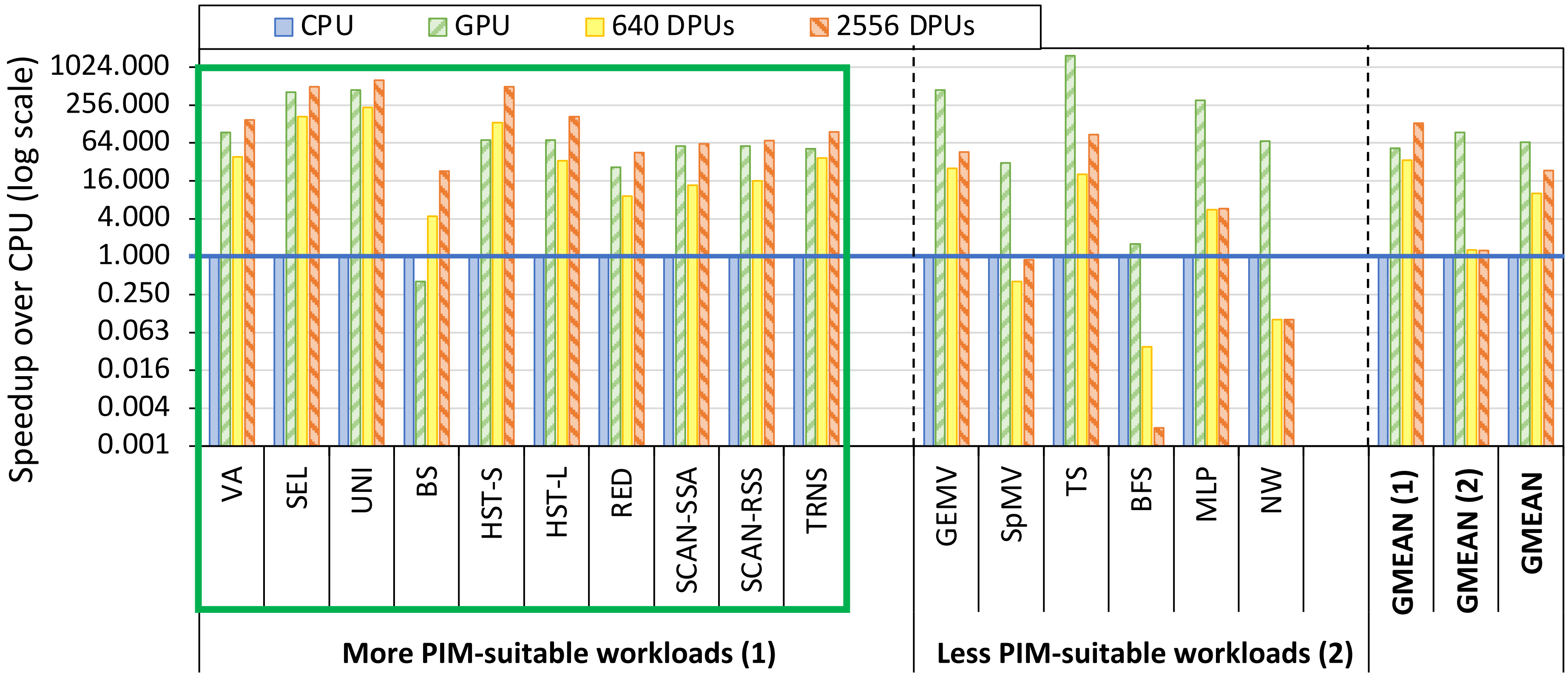
Key Takeaway 2

Table 4: Evaluated CPU, GPU, and UPMEM-based PIM Systems.

System	Process Node	Processor Cores			Memory		TDP
		Total Cores	Frequency	Peak Performance	Capacity	Total Bandwidth	
Intel Xeon E3-1225 v6 CPU [241]	14 nm	4 (8 threads)	3.3 GHz	26.4 GFLOPS*	32 GB	37.5 GB/s	73 W
NVIDIA Titan V GPU [277]	14 nm	80 (5,120 SIMD lanes)	1.2 GHz	12,288.0 GFLOPS	12 GB	652.8 GB/s	250 W
2,556-DPU PIM System	2x nm	2,556[9]	350 MHz	894.6 GOPS	159.75 GB	1.7 TB/s	383 W†
640-DPU PIM System	2x nm	640	267 MHz	170.9 GOPS	40 GB	333.75 GB/s	96 W†

*Estimated GFLOPS = 3.3 GHz × 4 cores × 2 instructions per cycle.

†Estimated TDP = $\frac{Total\ DPUs}{DPUs/chip} \times 1.2\ W/chip$ [199].



KEY TAKEAWAY 2

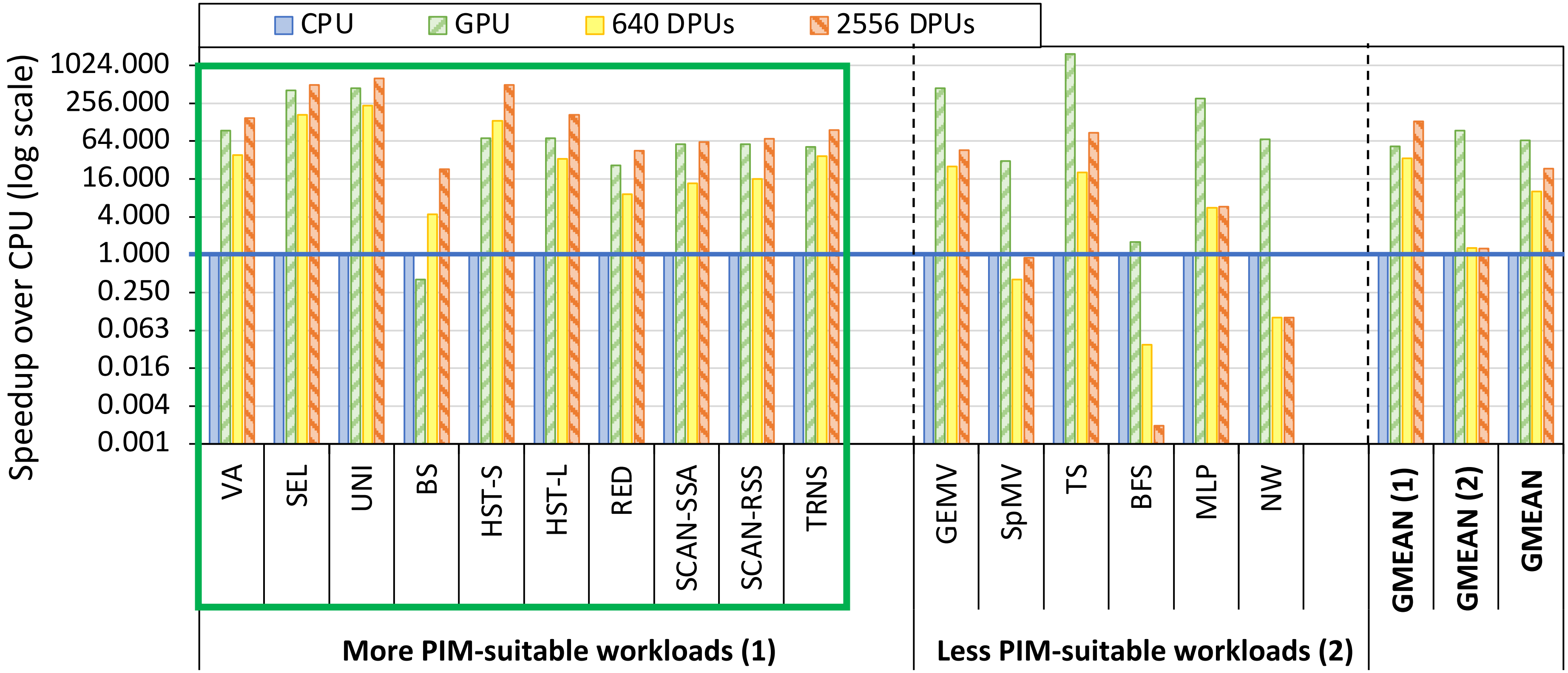
The most well-suited workloads for the UPMEM PIM architecture use no arithmetic operations or use only simple operations (e.g., bitwise operations and integer addition/subtraction).

Key Takeaway 3

Table 4: Evaluated CPU, GPU, and UPMEM-based PIM Systems.

System	Process Node	Processor Cores			Memory		TDP
		Total Cores	Frequency	Peak Performance	Capacity	Total Bandwidth	
Intel Xeon E3-1225 v6 CPU [241]	14 nm	4 (8 threads)	3.3 GHz	26.4 GFLOPS*	32 GB	37.5 GB/s	73 W
NVIDIA Titan V GPU [277]	14 nm	80 (5,120 SIMD lanes)	1.2 GHz	12,288.0 GFLOPS	12 GB	652.8 GB/s	250 W
2,556-DPU PIM System	2x nm	2,556[9]	350 MHz	894.6 GOPS	159.75 GB	1.7 TB/s	383 W†
640-DPU PIM System	2x nm	640	267 MHz	170.9 GOPS	40 GB	333.75 GB/s	96 W†

Estimated GFLOPS = 3.3 GHz × 4 cores × 2 instructions per cycle.
†*Estimated TDP =* $\frac{Total\ DPUs}{DPUs/chip} \times 1.2\ W/chip$ [199].



KEY TAKEAWAY 3

The most well-suited workloads for the UPMEM PIM architecture require little or no communication across DPUs (inter-DPU communication).

UPMEM PIM System Summary & Analysis

- Juan Gomez-Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu,
  "Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware"
  *Invited Paper at Workshop on Computing with Unconventional Technologies (**CUT**)*, Virtual, October 2021.
  [arXiv version]
  [PrIM Benchmarks Source Code]
  [Slides (pptx) (pdf)]
  [Talk Video (37 minutes)]
  [Lightning Talk Video (3 minutes)]

Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware

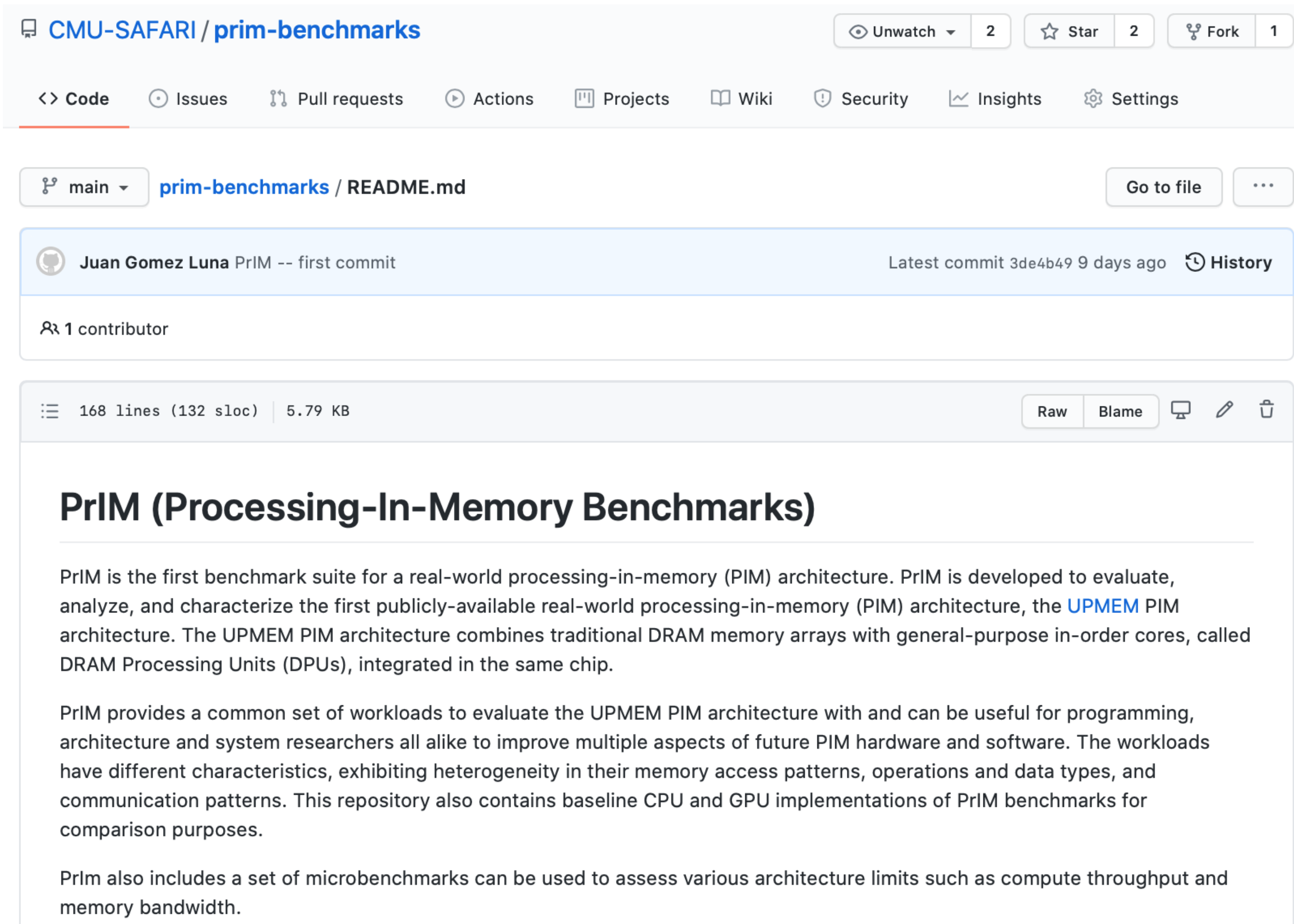
Juan Gómez-Luna, *ETH Zürich*
Izzat El Hajj, *American University of Beirut*
Ivan Fernandez, *University of Malaga*
Christina Giannoula, *National Technical University of Athens*
Geraldo F. Oliveira, *ETH Zürich*
Onur Mutlu, *ETH Zürich*

PrIM Benchmarks: Application Domains

Domain	Benchmark	Short name
Dense linear algebra	Vector Addition	VA
	Matrix-Vector Multiply	GEMV
Sparse linear algebra	Sparse Matrix-Vector Multiply	SpMV
Databases	Select	SEL
	Unique	UNI
Data analytics	Binary Search	BS
	Time Series Analysis	TS
Graph processing	Breadth-First Search	BFS
Neural networks	Multilayer Perceptron	MLP
Bioinformatics	Needleman-Wunsch	NW
Image processing	Image histogram (short)	HST-S
	Image histogram (large)	HST-L
Parallel primitives	Reduction	RED
	Prefix sum (scan-scan-add)	SCAN-SSA
	Prefix sum (reduce-scan-scan)	SCAN-RSS
	Matrix transposition	TRNS

PrIM Benchmarks are Open Source

- All microbenchmarks, benchmarks, and scripts
- https://github.com/CMU-SAFARI/prim-benchmarks



Understanding a Modern PIM Architecture

Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System

JUAN GÓMEZ-LUNA[1], IZZAT EL HAJJ[2], IVAN FERNANDEZ[1,3], CHRISTINA GIANNOULA[1,4], GERALDO F. OLIVEIRA[1], AND ONUR MUTLU[1]
[1]ETH Zürich
[2]American University of Beirut
[3]University of Malaga
[4]National Technical University of Athens

Corresponding author: Juan Gómez-Luna (e-mail: juang@ethz.ch).

https://arxiv.org/pdf/2105.03814.pdf

https://github.com/CMU-SAFARI/prim-benchmarks

Understanding a Modern PIM Architecture

https://www.youtube.com/watch?v=D8Hjy2iU9l4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9

ML Training on a Real PIM System

Machine Learning Training on a Real Processing-in-Memory System

Juan Gómez-Luna[1] Yuxin Guo[1] Sylvan Brocard[2] Julien Legriel[2]
Remy Cimadomo[2] Geraldo F. Oliveira[1] Gagandeep Singh[1] Onur Mutlu[1]

[1]ETH Zürich [2]UPMEM

An Experimental Evaluation of Machine Learning Training on a Real Processing-in-Memory System

Juan Gómez-Luna[1] Yuxin Guo[1] Sylvan Brocard[2] Julien Legriel[2]
Remy Cimadomo[2] Geraldo F. Oliveira[1] Gagandeep Singh[1] Onur Mutlu[1]

[1]ETH Zürich [2]UPMEM

Short version: https://arxiv.org/pdf/2206.06022.pdf

Long version: https://arxiv.org/pdf/2207.07886.pdf

https://www.youtube.com/watch?v=qeukNs5XI3g&t=11226s

ML Training on a Real PIM System

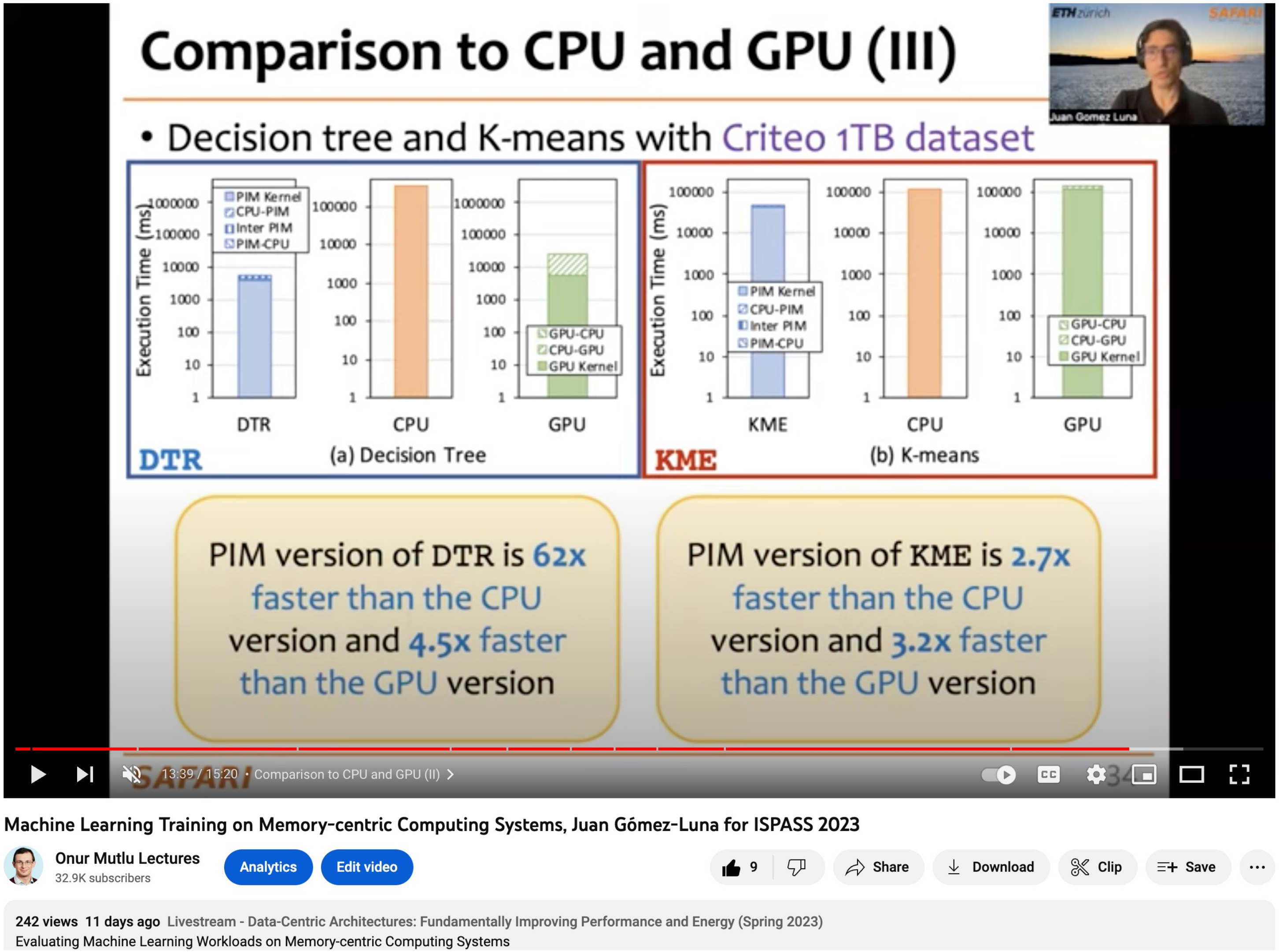
- **Need to optimize data representation**
  (1) fixed-point
  (2) quantization
  (3) hybrid precision

- **Use lookup tables (LUTs) to implement complex functions (e.g., sigmoid)**

- **Optimize data placement & layout for streaming**

- **Large speedups:** 2.8X/27X vs. CPU, 1.3x/3.2x vs. GPU

ML Training on Real PIM Talk Video



https://www.youtube.com/watch?v=60pkaI5AeM4

ML Training on Real PIM Systems

- Juan Gómez Luna, Yuxin Guo, Sylvan Brocard, Julien Legriel, Remy Cimadomo, Geraldo F. Oliveira, Gagandeep Singh, and Onur Mutlu,
  "Evaluating Machine Learning Workloads on Memory-Centric Computing Systems"
  *Proceedings of the 2023 IEEE International Symposium on Performance Analysis of Systems and Software (**ISPASS**)*, Raleigh, North Carolina, USA, April 2023.
  [arXiv version, 16 July 2022.]
  [PIM-ML Source Code]
  Best paper session.

An Experimental Evaluation of Machine Learning Training on a Real Processing-in-Memory System

Juan Gómez-Luna[1] Yuxin Guo[1] Sylvan Brocard[2] Julien Legriel[2]
Remy Cimadomo[2] Geraldo F. Oliveira[1] Gagandeep Singh[1] Onur Mutlu[1]

[1]ETH Zürich [2]UPMEM

https://github.com/CMU-SAFARI/pim-ml

https://arxiv.org/pdf/2207.07886.pdf

SpMV Multiplication on Real PIM Systems

- Appears at SIGMETRICS 2022

SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Systems

CHRISTINA GIANNOULA, ETH Zürich, Switzerland and National Technical University of Athens, Greece
IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
NECTARIOS KOZIRIS, National Technical University of Athens, Greece
GEORGIOS GOUMAS, National Technical University of Athens, Greece
ONUR MUTLU, ETH Zürich, Switzerland

https://arxiv.org/pdf/2201.05072.pdf

https://github.com/CMU-SAFARI/SparseP

https://www.youtube.com/watch?v=5kaOsJKlGrE

SparseP

Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures

Christina Giannoula

Ivan Fernandez, Juan Gomez-Luna,

Nectarios Koziris, Georgios Goumas, Onur Mutlu

SparseP: Key Contributions

1. Efficient SpMV kernels for current & future PIM systems
   - SparseP library = 25 SpMV kernels
     - Compression, data types, data partitioning, synchronization, load balancing

SparseP is Open-Source

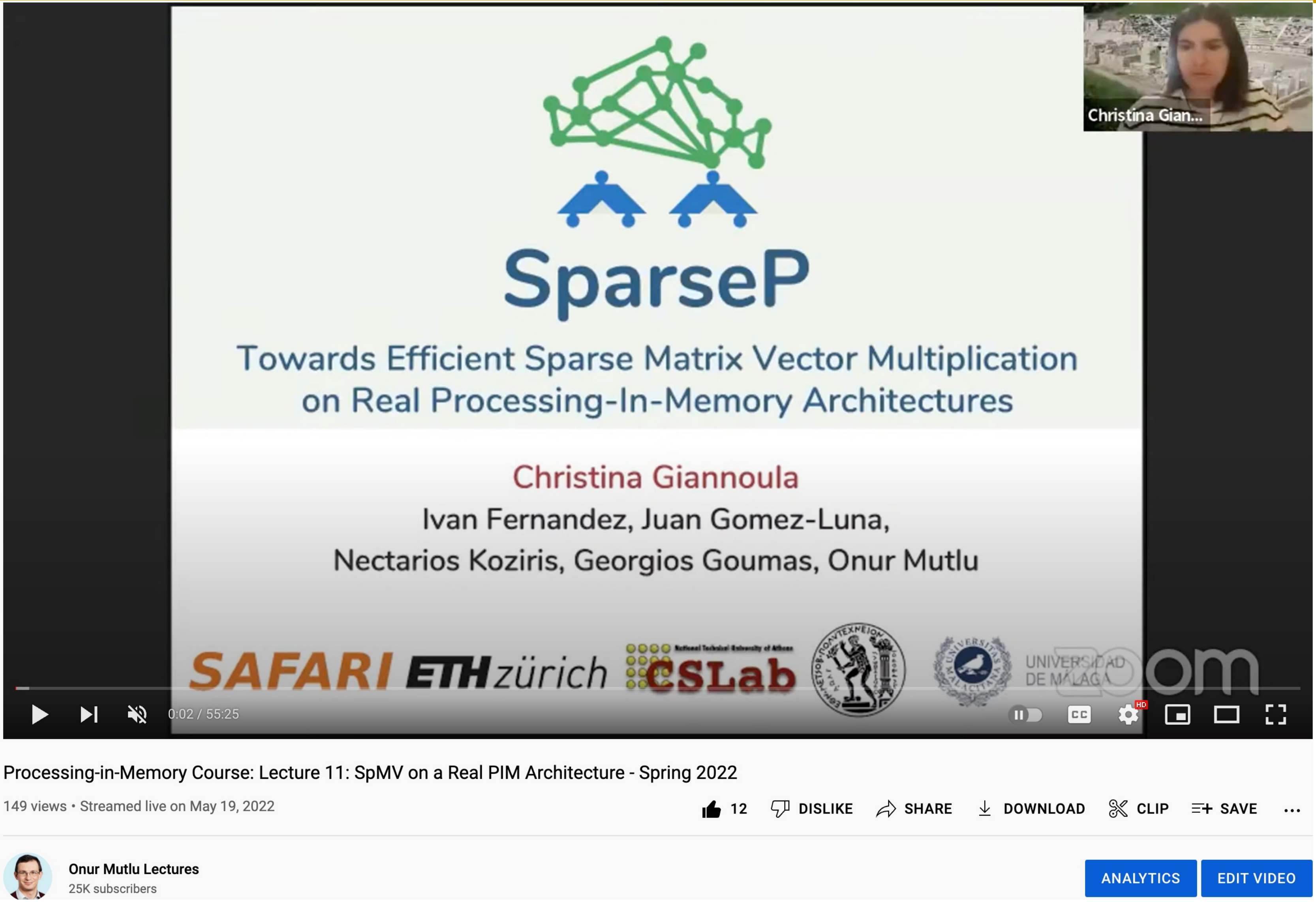
SparseP: https://github.com/CMU-SAFARI/SparseP

2. Comprehensive analysis of SpMV on the first commercially-available real PIM system
   - 26 sparse matrices
   - Comparisons to state-of-the-art CPU and GPU systems
   - Recommendations for software, system and hardware designers

Recommendations for Architects and Programmers

Full Paper: https://arxiv.org/pdf/2201.05072.pdf

SparseP Talk Video



https://www.youtube.com/watch?v=5kaOsJKlGrE

More on SparseP

Christina Giannoula, Ivan Fernandez, Juan Gomez-Luna, Nectarios Koziris, Georgios Goumas, and Onur Mutlu,
"SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures"
*Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**)*, Mumbai, India, June 2022.
[Extended arXiv Version]
[Abstract]
[Slides (pptx) (pdf)]
[Long Talk Slides (pptx) (pdf)]
[SparseP Source Code]
[Talk Video (16 minutes)]
[Long Talk Video (55 minutes)]

***SparseP*: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Systems**

CHRISTINA GIANNOULA, ETH Zürich, Switzerland and National Technical University of Athens, Greece
IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
NECTARIOS KOZIRIS, National Technical University of Athens, Greece
GEORGIOS GOUMAS, National Technical University of Athens, Greece
ONUR MUTLU, ETH Zürich, Switzerland

https://github.com/CMU-SAFARI/SparseP

https://arxiv.org/pdf/2201.05072.pdf

Transcendental Functions on Real PIM Systems

- Maurus Item, Juan Gómez Luna, Yuxin Guo, Geraldo F. Oliveira, Mohammad Sadrosadati, and Onur Mutlu,
  "TransPimLib: Efficient Transcendental Functions for Processing-in-Memory Systems"
  *Proceedings of the 2023 IEEE International Symposium on Performance Analysis of Systems and Software (**ISPASS**)*, Raleigh, North Carolina, USA, April 2023.
  [arXiv version]
  [Slides (pptx) (pdf)]
  [TransPimLib Source Code]
  [Talk Video (17 minutes)]

TransPimLib: Efficient Transcendental Functions for Processing-in-Memory Systems

Maurus Item Juan Gómez-Luna Yuxin Guo
Geraldo F. Oliveira Mohammad Sadrosadati Onur Mutlu

ETH Zürich

https://github.com/CMU-SAFARI/transpimlib

https://arxiv.org/pdf/2304.01951.pdf

Sequence Alignment on Real PIM Systems

- Safaa Diab, Amir Nassereldine, Mohammed Alser, Juan Gómez Luna, Onur Mutlu, and Izzat El Hajj,
  "A Framework for High-throughput Sequence Alignment using Real Processing-in-Memory Systems"
  Bioinformatics, [published online on] 27 March 2023.
  [Online link at Bioinformatics Journal]
  [arXiv preprint]
  [AiM Source Code]

A Framework for High-throughput Sequence Alignment using Real Processing-in-Memory Systems

Safaa Diab [1] Amir Nassereldine [1] Mohammed Alser [2] Juan Gómez Luna [2]
Onur Mutlu [2] Izzat El Hajj [1]

[1]American University of Beirut [2]ETH Zürich

https://github.com/CMU-SAFARI/alignment-in-memory

https://arxiv.org/pdf/2208.01243.pdf

Summary

- Sequence alignment on traditional systems is limited by the **memory bandwidth bottleneck**
- **Processing-in-memory (PIM)** overcomes this bottleneck by placing cores near the memory
- Our framework, **Alignment-in-Memory (AIM)**, is a PIM framework that supports multiple alignment algorithms (NW, SWG, GenASM, WFA)
  - Implemented on UPMEM, the first real PIM system
- Results show **substantial speedups over both CPUs (1.8X-28X) and GPUs (1.2X-2.7X)**
- AIM is available at:
  - https://github.com/CMU-SAFARI/alignment-in-memory

Homomorphic Operations on Real PIM Systems

- Harshita Gupta, Mayank Kabra, Juan Gómez-Luna, Konstantinos Kanellopoulos, and Onur Mutlu,
  "Evaluating Homomorphic Operations on a Real-World Processing-In-Memory System"
  *Proceedings of the 2023 IEEE International Symposium on Workload Characterization Poster Session (**IISWC**)*, Ghent, Belgium, October 2023.
  [arXiv version]
  [Lightning Talk Slides (pptx) (pdf)]
  [Poster (pptx) (pdf)]

Evaluating Homomorphic Operations on a Real-World Processing-In-Memory System

Harshita Gupta* Mayank Kabra* Juan Gómez-Luna Konstantinos Kanellopoulos Onur Mutlu
ETH Zürich

Accelerating ML Training on Real PIM Systems

- **https://arxiv.org/pdf/2404.07164**

Analysis of Distributed Optimization Algorithms on a Real Processing-In-Memory System

Steve Rhyner[1] Haocong Luo[1] Juan Gómez-Luna[2] Mohammad Sadrosadati[1]
Jiawei Jiang[3] Ataberk Olgun[1] Harshita Gupta[1] Ce Zhang[4] Onur Mutlu[1]

[1]ETH Zurich [2]NVIDIA [3]Wuhan University [4]University of Chicago

Accelerating ML Training on Real PIM Systems

- https://arxiv.org/pdf/2404.07164

8. Conclusion

We evaluate and train ML models on large-scale datasets with centralized parallel optimization algorithms on a *real-world* PIM architecture. We show the importance of carefully *choosing* the distributed optimization algorithm that fits PIM and analyze tradeoffs. We demonstrate that *commercial* general-purpose PIM systems can be a viable alternative for many ML training workloads on large-scale datasets to processor-centric architectures. Our results demonstrate the necessity of adapting PIM architectures to enable inter-DPU communication to overcome scalability challenges for many ML training workloads and discuss decentralized parallel SGD optimization algorithms as a potential solution.

- https://arxiv.org/pdf/2402.16731

Accelerating Graph Neural Networks on Real Processing-In-Memory Systems

Christina Giannoula[*,†], Peiming Yang[*], Ivan Fernandez Vega[§,†], Jiacheng Yang[*], Yu Xin Li[*],
Juan Gomez Luna[¶,†], Mohammad Sadrosadati[†], Onur Mutlu[†,‡], Gennady Pekhimenko[*,‖]
[*]University of Toronto [†]ETH Zürich [§]Barcelona Supercomputing Center
[¶]NVIDIA [‡]Stanford [‖]CentML

Accelerating GNNs on Real PIM Systems

- https://arxiv.org/pdf/2402.16731

Abstract—Graph Neural Networks (GNNs) are emerging ML models to analyze graph-structure data. Graph Neural Network (GNN) execution involves both compute-intensive and memory-intensive kernels, the latter dominates the total time, being significantly bottlenecked by data movement between memory and processors. Processing-In-Memory (PIM) systems can alleviate this data movement bottleneck by placing simple processors near or inside to memory arrays. In this work, we introduce PyGim, an efficient ML framework that accelerates GNNs on real PIM systems. We propose intelligent parallelization techniques for memory-intensive kernels of GNNs tailored for real PIM systems, and develop handy Python API for them. We provide hybrid GNN execution, in which the compute-intensive and memory-intensive kernels are executed in processor-centric and memory-centric computing systems, respectively, to match their algorithmic nature. We extensively evaluate PyGim on a real-world PIM system with 1992 PIM cores using emerging GNN models, and demonstrate that it outperforms its state-of-the-art CPU counterpart on Intel Xeon by on average 3.04×, and achieves higher resource utilization than CPU and GPU systems. Our work provides useful recommendations for software, system and hardware designers. PyGim will be open-sourced to enable the widespread use of PIM systems in GNNs.

Samsung Function-in-Memory DRAM (2021)

Samsung
Newsroom

CORPORATE | PRODUCTS | PRESS RESOURCES | VIEWS | ABOUT US

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%

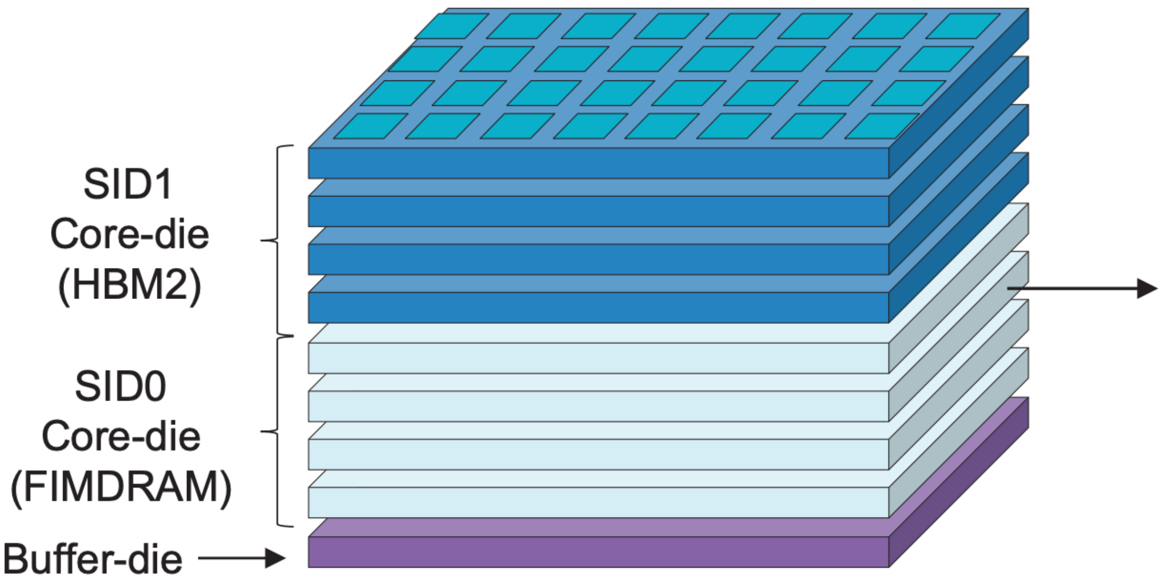
Samsung Electronics, the world leader in advanced memory technology, today announced that it has developed the industry's first High Bandwidth Memory (HBM) integrated with artificial intelligence (AI) processing power — the HBM-PIM. The new processing-in-memory (PIM) architecture brings powerful AI computing 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."

https://news.samsung.com/global/samsung-develops-industrys-first-high-bandwidth-memory-with-ai-processing-power

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

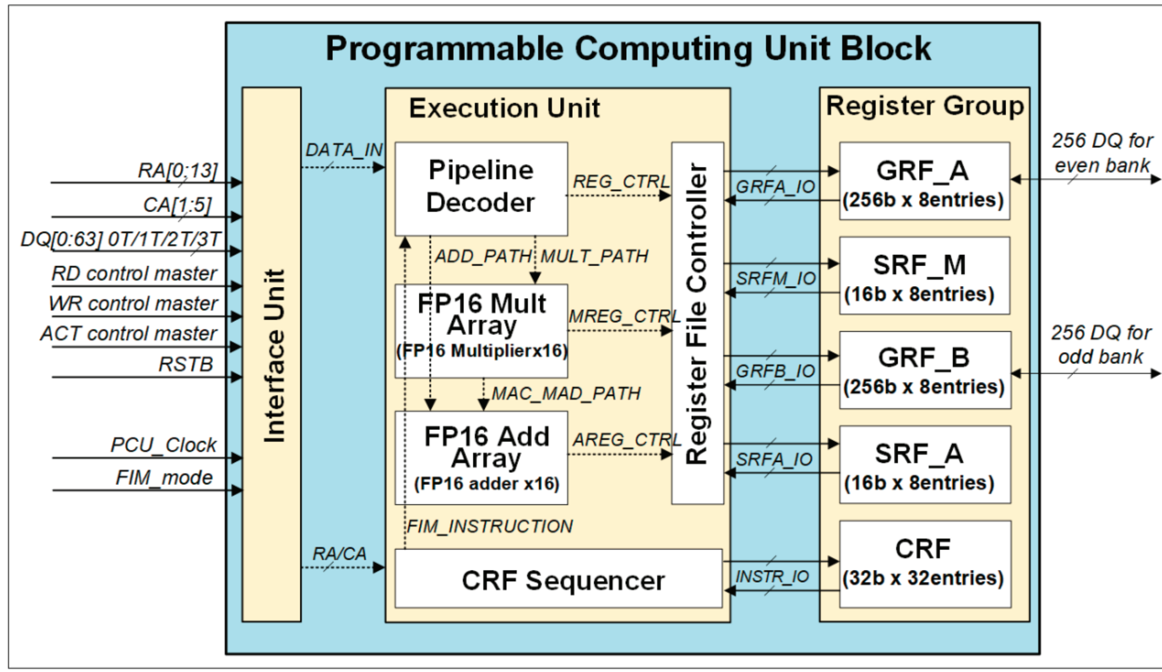
Young-Cheon Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1], Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1], Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1], Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1], Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1], David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[3], JoonHo Song[3], Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

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

Samsung Function-in-Memory DRAM (2021)

Programmable Computing Unit

- Configuration of PCU block
  - Interface unit to control data flow
  - Execution unit to perform operations
  - Register group
    - 32 entries of CRF for instruction memory
    - 16 GRF for weight and accumulation
    - 16 SRF to store constants for MAC operations



[Block diagram of PCU in FIMDRAM]

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 Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1], Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1], Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1], Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1], Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1], David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[3], JoonHo Song[3], Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

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

Samsung Function-in-Memory DRAM (2021)

[Available instruction list for FIM operation]

Type	CMD	Description
Floating Point	ADD	FP16 addition
	MUL	FP16 multiplication
	MAC	FP16 multiply-accumulate
	MAD	FP16 multiply and add
Data Path	MOVE	Load or store data
	FILL	Copy data from bank to GRFs
Control Path	NOP	Do nothing
	JUMP	Jump instruction
	EXIT	Exit instruction

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

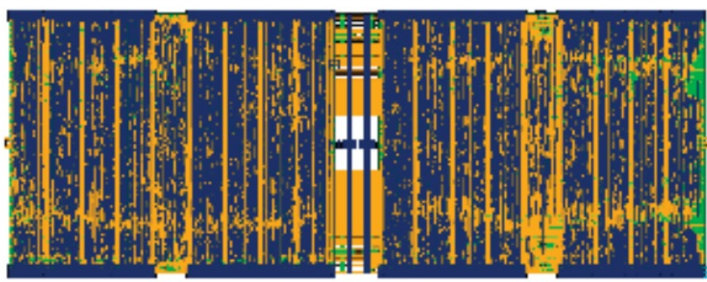
Young-Cheon Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1], Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1], Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1], Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1], Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1], David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[3], JoonHo Song[3], Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

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

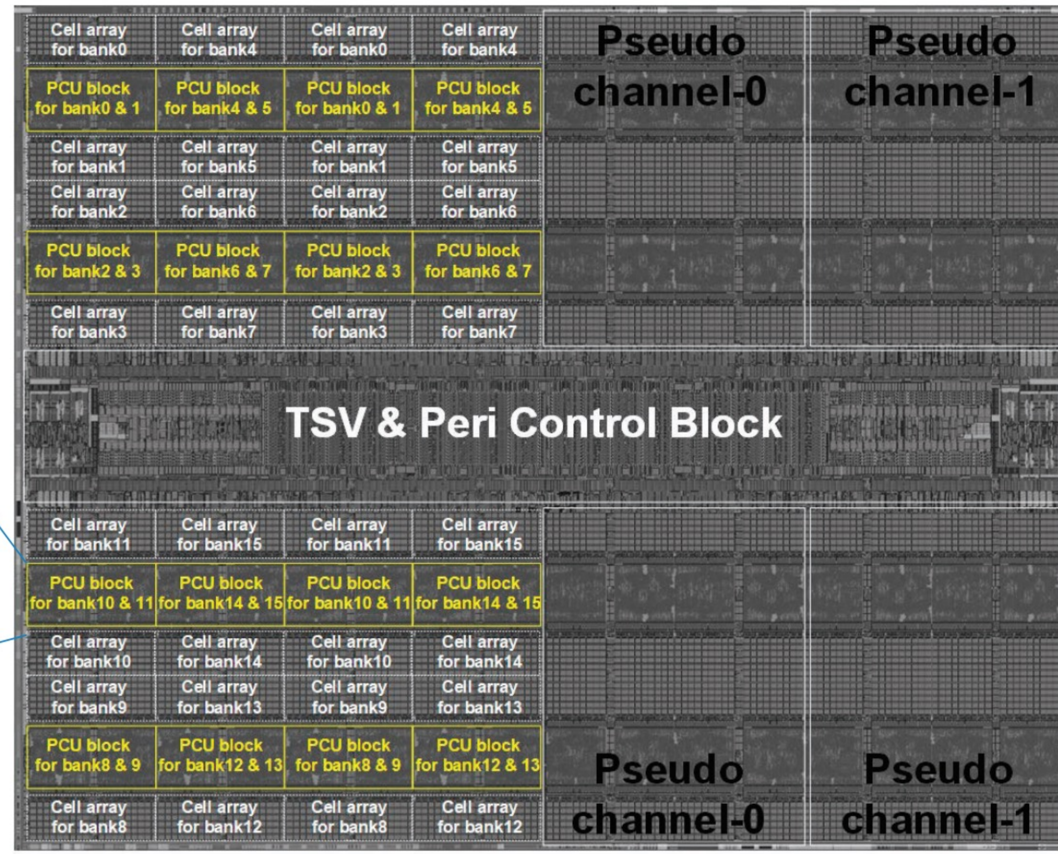
Samsung Function-in-Memory DRAM (2021)

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

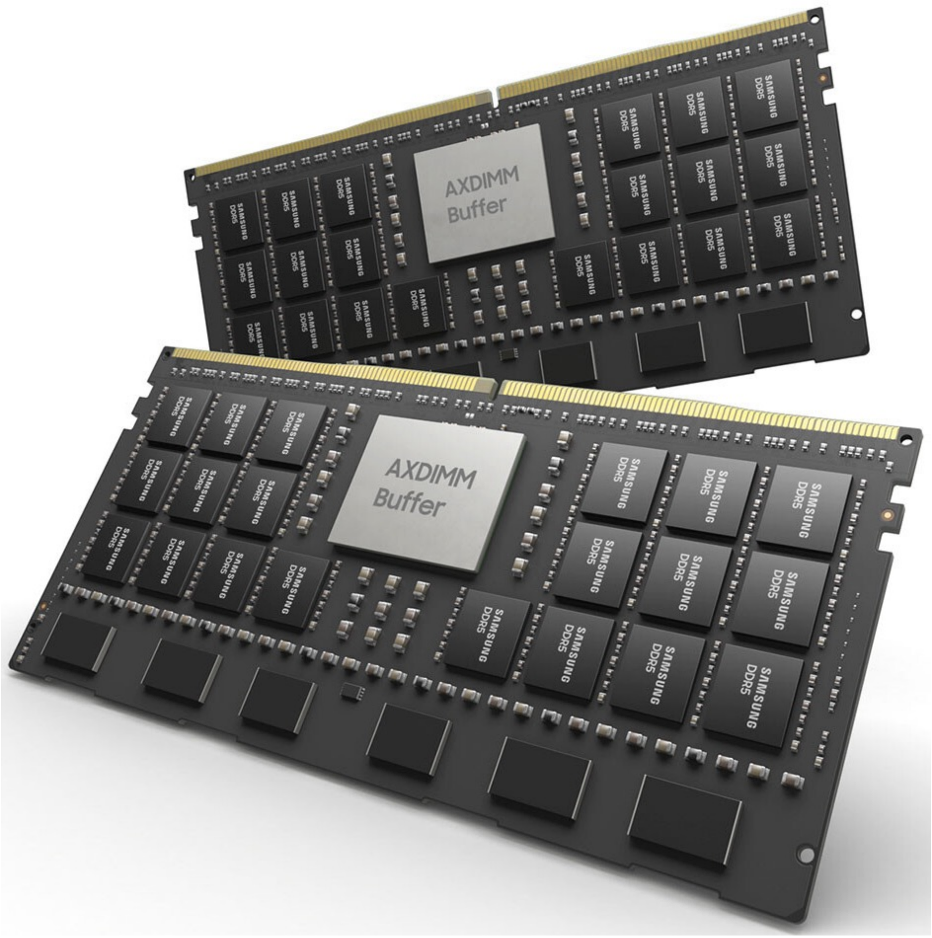
25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

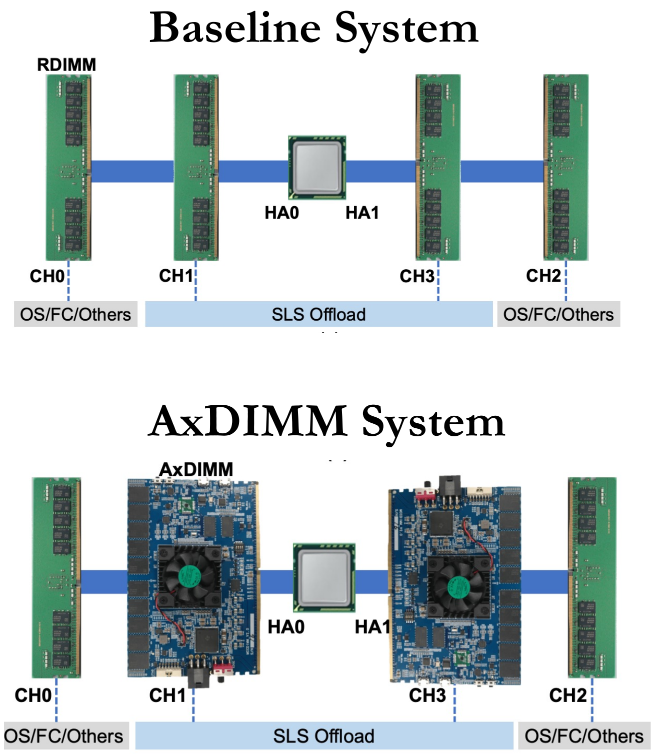
Young-Cheon Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1], Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1], Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1], Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1], Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1], David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[3], JoonHo Song[3], Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

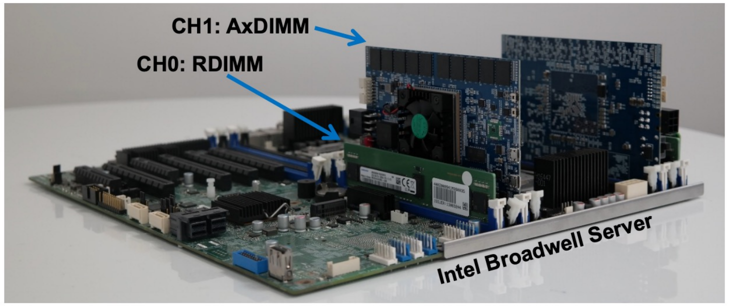
[1]Samsung Electronics, Hwaseong, Korea
[2]Samsung Electronics, San Jose, CA
[3]Samsung Electronics, Suwon, Korea

Samsung AxDIMM (2021)

- DDRx-PIM
  - DLRM recommendation system







Ke et al. "Near-Memory Processing in Action: Accelerating Personalized Recommendation with AxDIMM", IEEE Micro (2021)

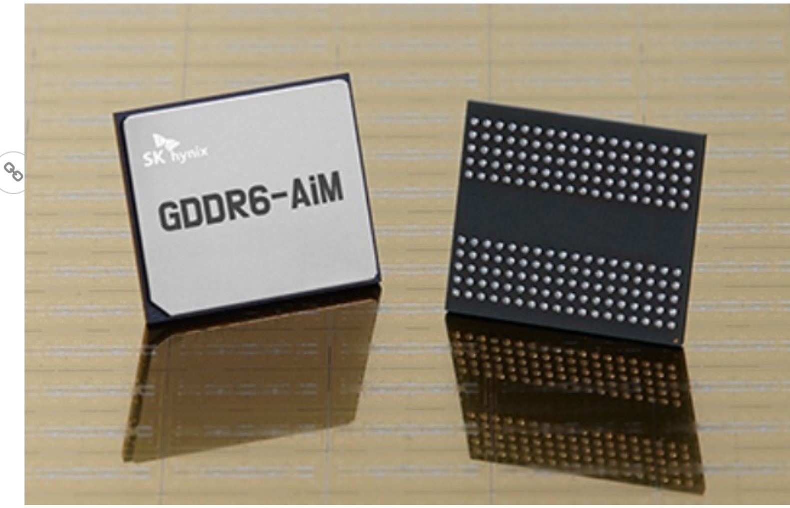
SK Hynix Accelerator-in-Memory (2022)

SKhynix NEWSROOM ENG

INSIGHT | SK hynix STORY | PRESS CENTER | MULTIMEDIA | Search

SK hynix Develops PIM, Next-Generation AI Accelerator

February 16, 2022



Seoul, February 16, 2022

SK hynix (or "the Company", www.skhynix.com) announced on February 16 that it has developed PIM*, a next-generation memory chip with computing capabilities.

**PIM(Processing In Memory): A next-generation technology that provides a solution for data congestion issues for AI and big data by adding computational functions to semiconductor memory*

It has been generally accepted that memory chips store data and CPU or GPU, like human brain, process data. SK hynix, following its challenge to such notion and efforts to pursue innovation in the next-generation smart memory, has found a breakthrough solution with the development of the latest technology.

SK hynix plans to showcase its PIM development at the world's most prestigious semiconductor conference, 2022 ISSCC*, in San Francisco at the end of this month. The company expects continued efforts for innovation of this technology to bring the memory-centric computing, in which semiconductor memory plays a central role, a step closer to the reality in devices such as smartphones.

**ISSCC: The International Solid-State Circuits Conference will be held virtually from Feb. 20 to Feb. 24 this year with a theme of "Intelligent Silicon for a Sustainable World"*

For the first product that adopts the PIM technology, SK hynix has developed a sample of GDDR6-AiM (Accelerator* in memory). The GDDR6-AiM adds computational functions to GDDR6* memory chips, which process data at 16Gbps. A combination of GDDR6-AiM with CPU or GPU instead of a typical DRAM makes certain computation speed 16 times faster. GDDR6-AiM is widely expected to be adopted for machine learning, high-performance computing, and big data computation and storage.

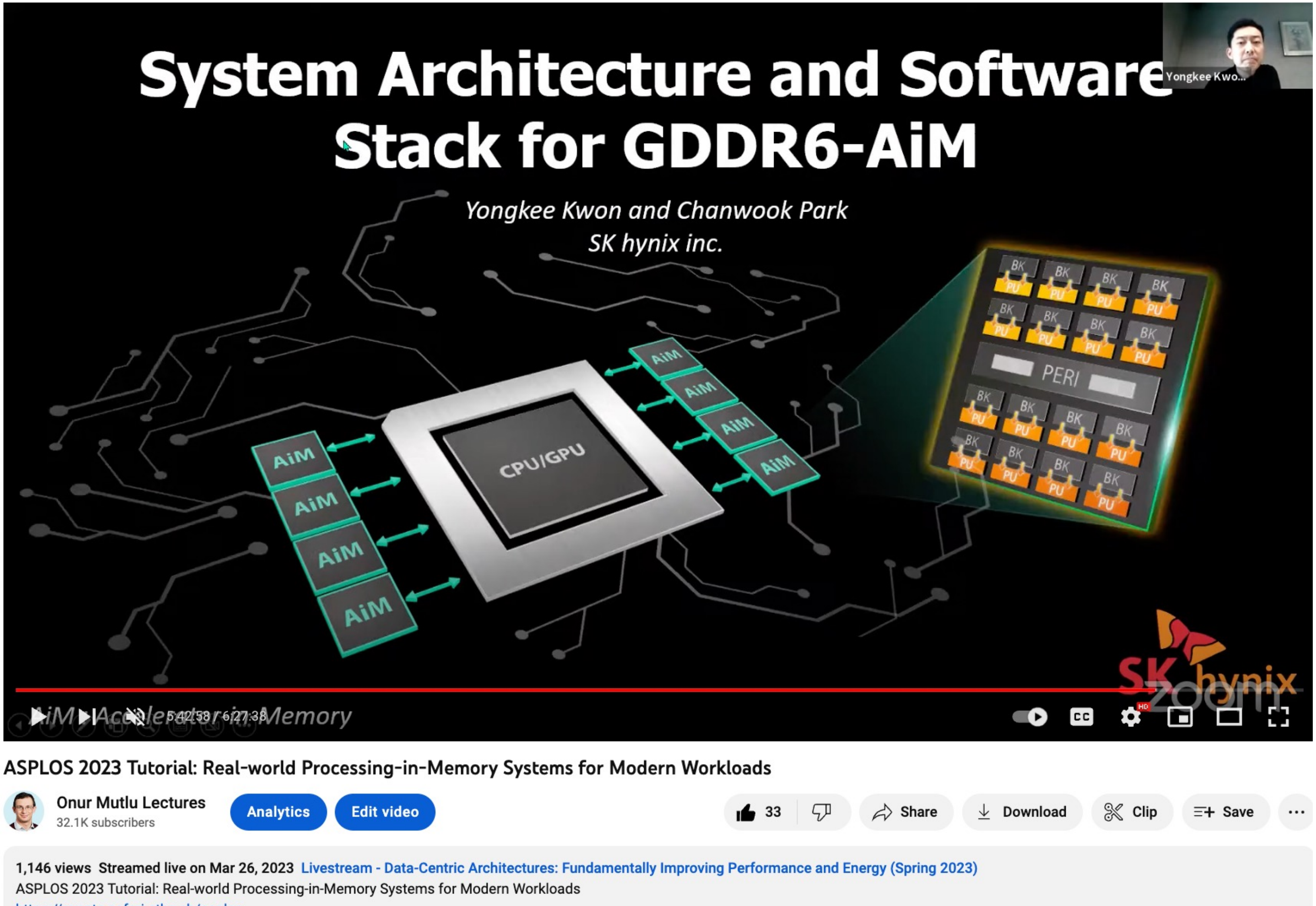
11.1 A 1ynm 1.25V 8Gb, 16Gb/s/pin GDDR6-based Accelerator-in-Memory supporting 1TFLOPS MAC Operation and Various Activation Functions for Deep-Learning Applications

Seongju Lee, SK hynix, Icheon, Korea

In Paper 11.1, SK Hynix describes an 1ynm, GDDR6-based accelerator-in-memory with a command set for deep-learning operation. The 8Gb design achieves a peak throughput of 1TFLOPS with 1GHz MAC operations and supports major activation functions to improve accuracy.

https://news.skhynix.com/sk-hynix-develops-pim-next-generation-ai-accelerator/

SK Hynix Accelerator-in-Memory (2022)



https://www.youtube.com/watch?v=oYCaLcT0Kmo

AliBaba PIM Recommendation System (2022)

ISSCC 2022 / February 24, 2022 / 8:30 AM

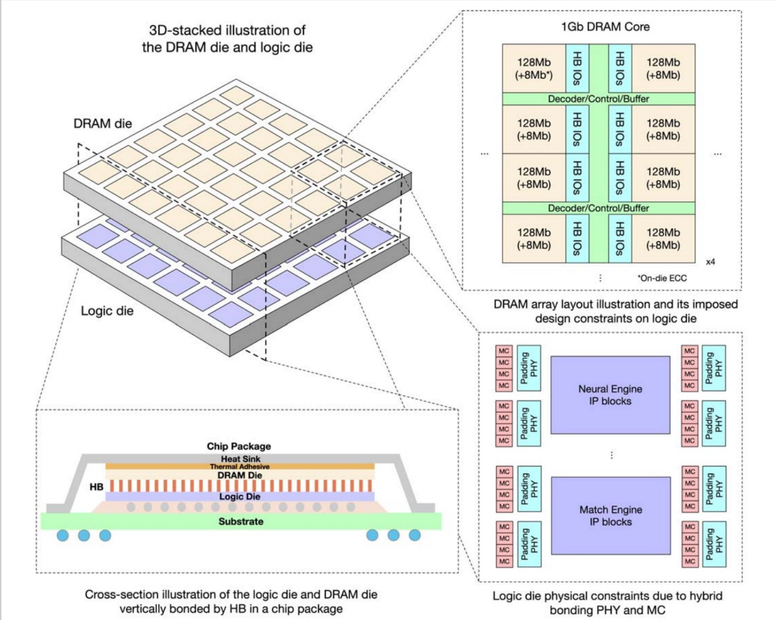
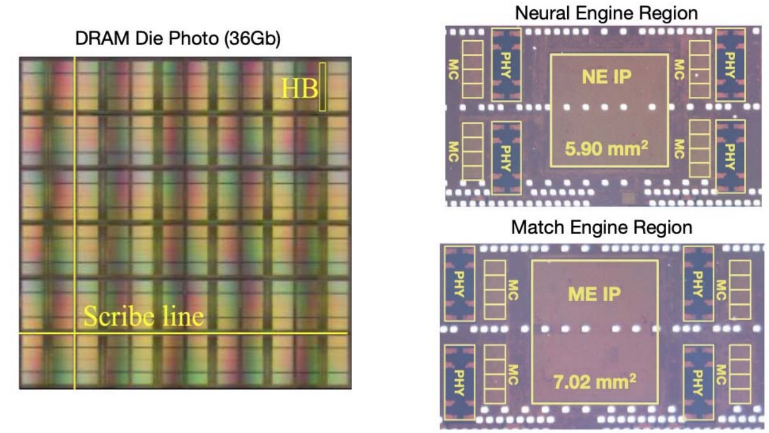


Figure 29.1.2: Illustration of 3D-stacked chip, cross-illustration of package, DRAM array layout and design blocks on logic die.



DRAM Die		
Technology	25nm	
Area	Total*	602.22 mm²
	Neural Engine	32 mm²
	Match Engine	32 mm²
Voltage	1.1 V	
Frequency (max)	150 MHz	
Power	300 mW per 1Gb	
Bandwidth**	153.60 GB/s / 1.38 TB/s	

* Total die size of test chip
** Memory bandwidth of NE and ME / Total bandwidth of DRAM die

Logic Die		
Technology	55nm	
Area	Total*	602.22 mm²
	Neural Engine	5.90 mm²
	Match Engine	7.02 mm²
# of MC	16 per IP	
Voltage	1.2 V	
Frequency	300 MHz	
Power	977.70 mW	
Precision	INT8	

Figure 29.1.7: Die micrographs of DRAM die, NE and ME. Detailed specifications of DRAM die and logic die.

29.1 184QPS/W 64Mb/mm² 3D Logic-to-DRAM Hybrid Bonding with Process-Near-Memory Engine for Recommendation System

Dimin Niu[1], Shuangchen Li[1], Yuhao Wang[1], Wei Han[1], Zhe Zhang[2], Yijin Guan[2], Tianchan Guan[3], Fei Sun[1], Fei Xue[1], Lide Duan[1], Yuanwei Fang[1], Hongzhong Zheng[1], Xiping Jiang[4], Song Wang[4], Fengguo Zuo[4], Yubing Wang[4], Bing Yu[4], Qiwei Ren[4], Yuan Xie[1]

SK Hynix CXL Processing Near Memory (2023)

IEEE COMPUTER ARCHITECTURE LETTERS, VOL. 22, NO. 1, JANUARY-JUNE

Computational CXL-Memory Solution for Accelerating Memory-Intensive Applications

Joonseop Sim, Soohong Ahn, Taeyoung Ahn, Seungyong Lee, Myunghyun Rhee, Jooyoung Kim, Kwangsik Shin, Donguk Moon, Euiseok Kim, and Kyoung Park

Abstract—CXL interface is the up-to-date technology that enables effective memory expansion by providing a memory-sharing protocol in configuring heterogeneous devices. However, its limited physical bandwidth can be a significant bottleneck for emerging data-intensive applications. In this work, we propose a novel CXL-based memory disaggregation architecture with a real-world prototype demonstration, which overcomes the bandwidth limitation of the CXL interface using near-data processing. The experimental results demonstrate that our design achieves up to 1.9× better performance/power efficiency than the existing CPU system.

Index Terms—Compute express link (CXL), near-data-processing (NDP)

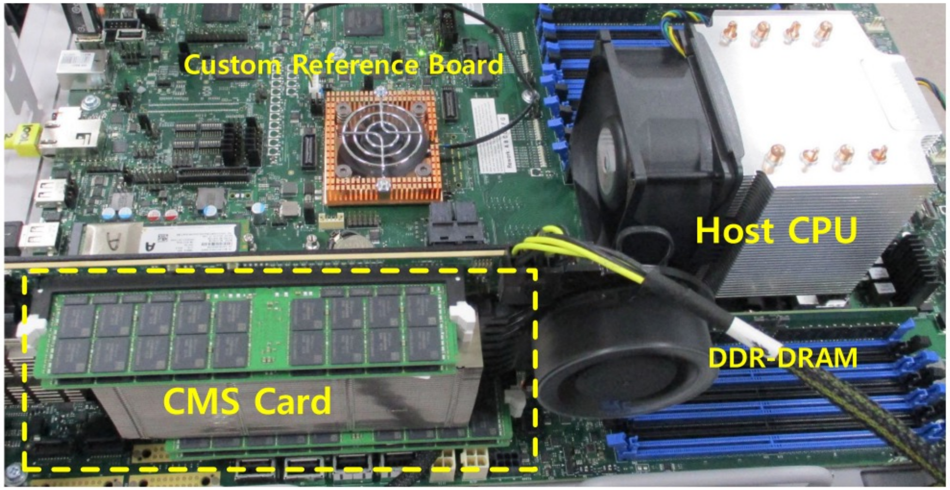


Fig. 6. FPGA prototype of proposed CMS card.

Samsung CXL Processing Near Memory (2023)

Samsung Processing in Memory Technology at Hot Chips 2023

By **Patrick Kennedy** - August 28, 2023

Samsung PIM PNM For Transformer Based AI HC35_Page_24

Concluding Remarks

Challenge and Opportunity for Future

Fundamentally

Energy-Efficient

(Data-Centric)

Computing Architectures

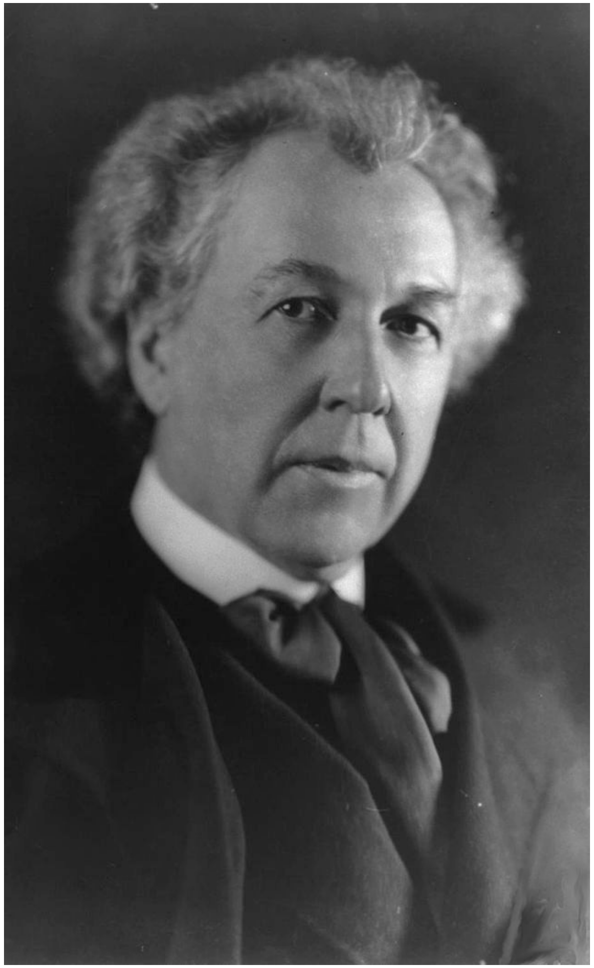
Challenge and Opportunity for Future

Fundamentally
High-Performance
(Data-Centric)
Computing Architectures

Computing Architectures
with
Minimal Data Movement

A Quote from A Famous Architect

- "architecture [...] based upon principle, and not upon precedent"



Precedent-Based Design?

- "architecture […] based upon principle, and not upon precedent"

Principled Design

- "architecture [...] based upon principle, and not upon precedent"


www.GreatBuildings.com

The Overarching Principle

Organic architecture

From Wikipedia, the free encyclopedia

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

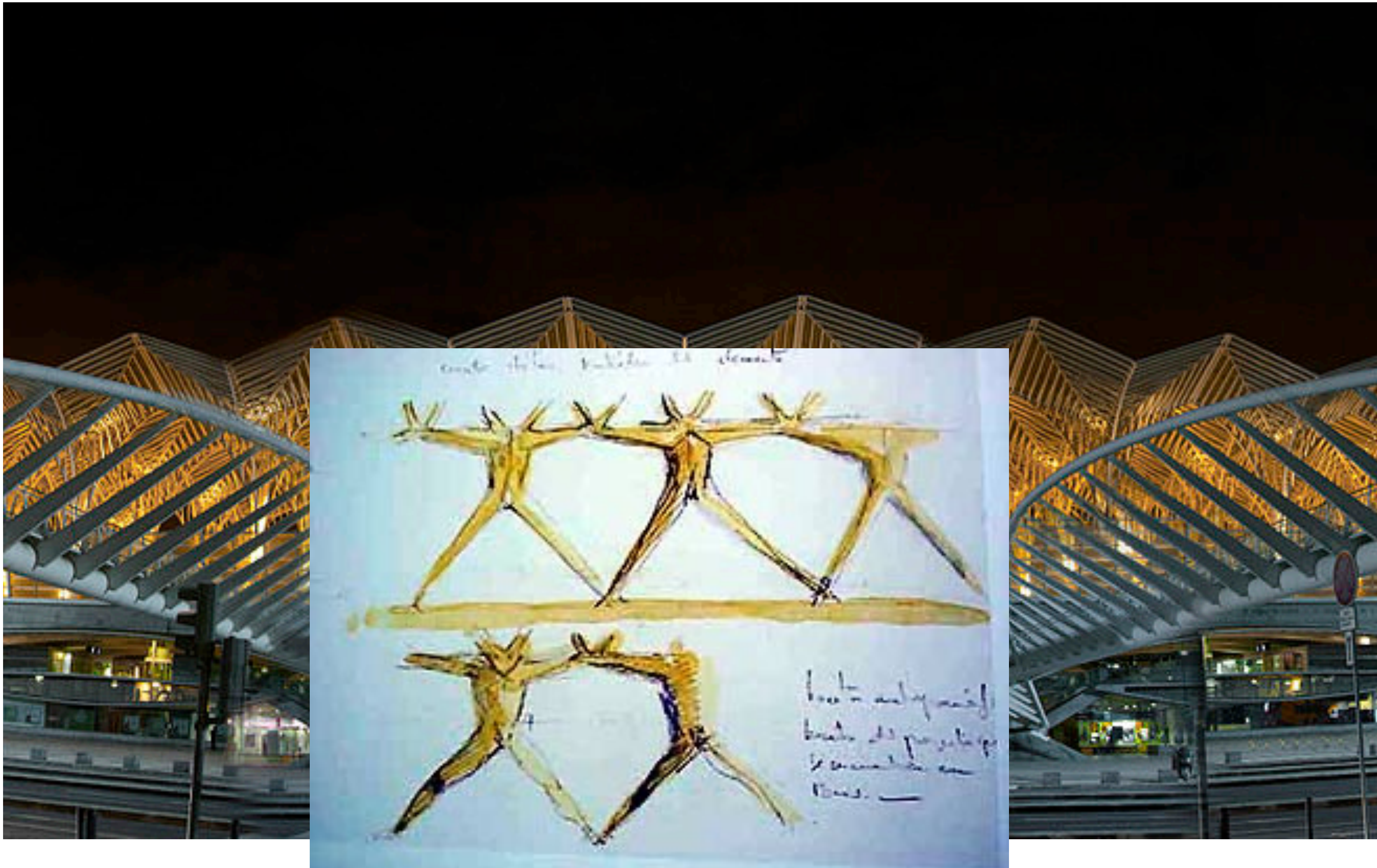
A well-known example of organic architecture is Fallingwater, the residence Frank Lloyd Wright designed for the Kaufmann family in rural Pennsylvania. Wright had many choices to locate a home on this large site, but chose to place the home directly over the waterfall and creek creating a close, yet noisy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

Another Example: Precedent-Based Design

Source: http://cookiemagik.deviantart.com/art/Train-station-207266944

Principled Design

Another Principled Design



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

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

Another Principled Design

Source: CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=172107

Principle Applied to Another Structure

Source: By 準建築人手札網站 Forgemind ArchiMedia - Flickr: IMG_2489.JPG, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=51493356, https://en.wikipedia.org/wiki/Santiago_Calatrava

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

The Overarching Principle

Zoomorphic architecture

From Wikipedia, the free encyclopedia

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

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

Overarching Principles for Computing?

Source: http://spectrum.ieee.org/image/MjYzMzAyMg.jpeg

Concluding Remarks

- It is time to design principled system architectures to solve the memory problem

- We must design systems to be **balanced, high-performance, and energy-efficient** → **memory-centric**
  - Enable computation capabilities in memory

- This can
  - Lead to **orders-of-magnitude** improvements
  - **Enable new applications & computing platforms**
  - **Enable better understanding of nature**
  - **…**

- Future of **truly memory-centric computing** is bright
  - We need to do research & design across the computing stack

Fundamentally Better Architectures

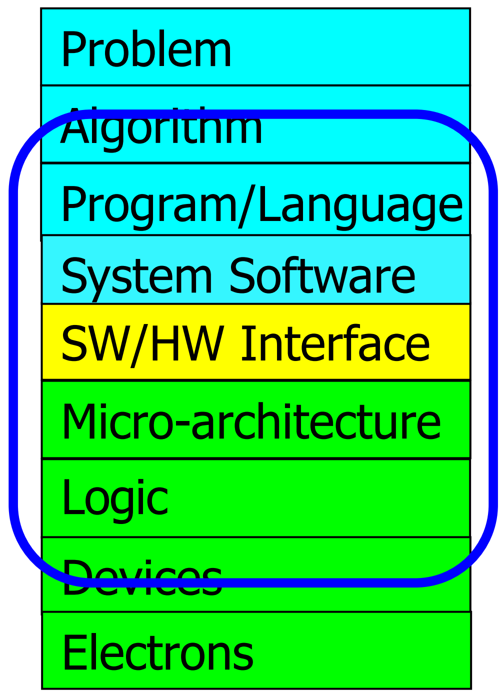
Data-centric

Data-driven

Data-aware

We Need to Revisit the Entire Stack

- With a **memory-centric mindset**



We can get there step by step

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.

https://arxiv.org/pdf/2012.03112.pdf

Open Source Tools: SAFARI GitHub

SAFARI Research Group

SAFARI Research Group at ETH Zurich and Carnegie Mellon University

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

440 followers · ETH Zurich and Carnegie Mellon U... · https://safari.ethz.ch/ · omutlu@gmail.com

Overview · Repositories 98 · Projects · Packages · People 13

ramulator Public

A Fast and Extensible DRAM Simulator, with built-in support for modeling many different DRAM technologies including DDRx, LPDDRx, GDDRx, WIOx, HBMx, and various academic proposals. Described in the...

C++ · 532 · 206

prim-benchmarks Public

PrIM (Processing-In-Memory benchmarks) is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. PrIM is developed to evaluate, analyze, and characterize the first publ...

C · 126 · 47

MQSim Public

MQSim is a fast and accurate simulator modeling the performance of modern multi-queue (MQ) SSDs as well as traditional SATA based SSDs. MQSim faithfully models new high-bandwidth protocol implement...

C++ · 268 · 143

rowhammer Public

Source code for testing the Row Hammer error mechanism in DRAM devices. Described in the ISCA 2014 paper by Kim et al. at http://users.ece.cmu.edu/~omutlu/pub/dram-row-hammer_isca14.pdf.

C · 211 · 42

SoftMC Public

SoftMC is an experimental FPGA-based memory controller design that can be used to develop tests for DDR3 SODIMMs using a C++ based API. The design, the interface, and its capabilities and limitatio...

Verilog · 120 · 27

Pythia Public

A customizable hardware prefetching framework using online reinforcement learning as described in the MICRO 2021 paper by Bera et al. (https://arxiv.org/pdf/2109.12021.pdf).

C++ · 109 · 34

https://github.com/CMU-SAFARI/

Referenced Papers, Talks, Artifacts

- All are available at

  https://people.inf.ethz.ch/omutlu/projects.htm

  https://www.youtube.com/onurmutlulectures

  https://github.com/CMU-SAFARI/

Funding Acknowledgments

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

Thank you!