Memory-Centric Computing for Data-Intensive Workloads

Dr. Mohammad Sadrosadati
Prof. Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu

11 December 2023

EFCL Mini-Conference

SAFARI

ETH zürich

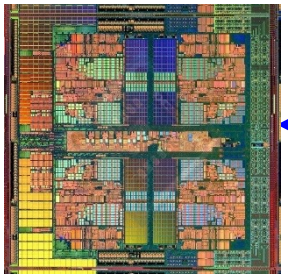
Brief Self Introduction

- Mohammad Sadrosadati
  - Senior Researcher and Lecturer @ SAFARI Research Group, ETHZ
  - PhD from Sharif University of Technology, 2014-2019
  - mohammad.sadrosadati@safari.ethz.ch

- Research Areas
  - Computer Architecture
  - Memory & Storage Systems
  - Near-Data Processing
  - Heterogeneous System Architecture
  - Bioinformatics
  - Interconnection Networks

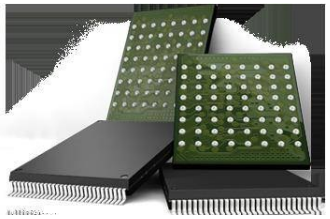
Current Research Mission

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

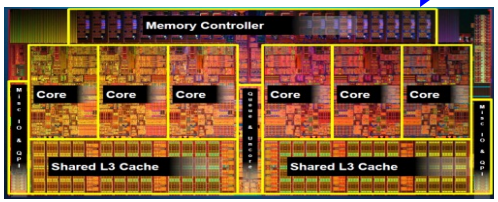


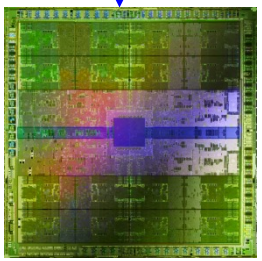
Heterogeneous Processors and Accelerators

Hybrid Main Memory



Persistent Memory/Storage





Graphics and Vision Processing

Build fundamentally better architectures

Four Key Current Directions

- Fundamentally Secure/Reliable/Safe Architectures
- Fundamentally Energy-Efficient Architectures
  - Memory-centric (Data-centric) Architectures
- Fundamentally Low-Latency and Predictable Architectures
- Architectures for AI/ML, Genomics, Medicine, Health, …

Fundamentally Better Architectures

Data-centric

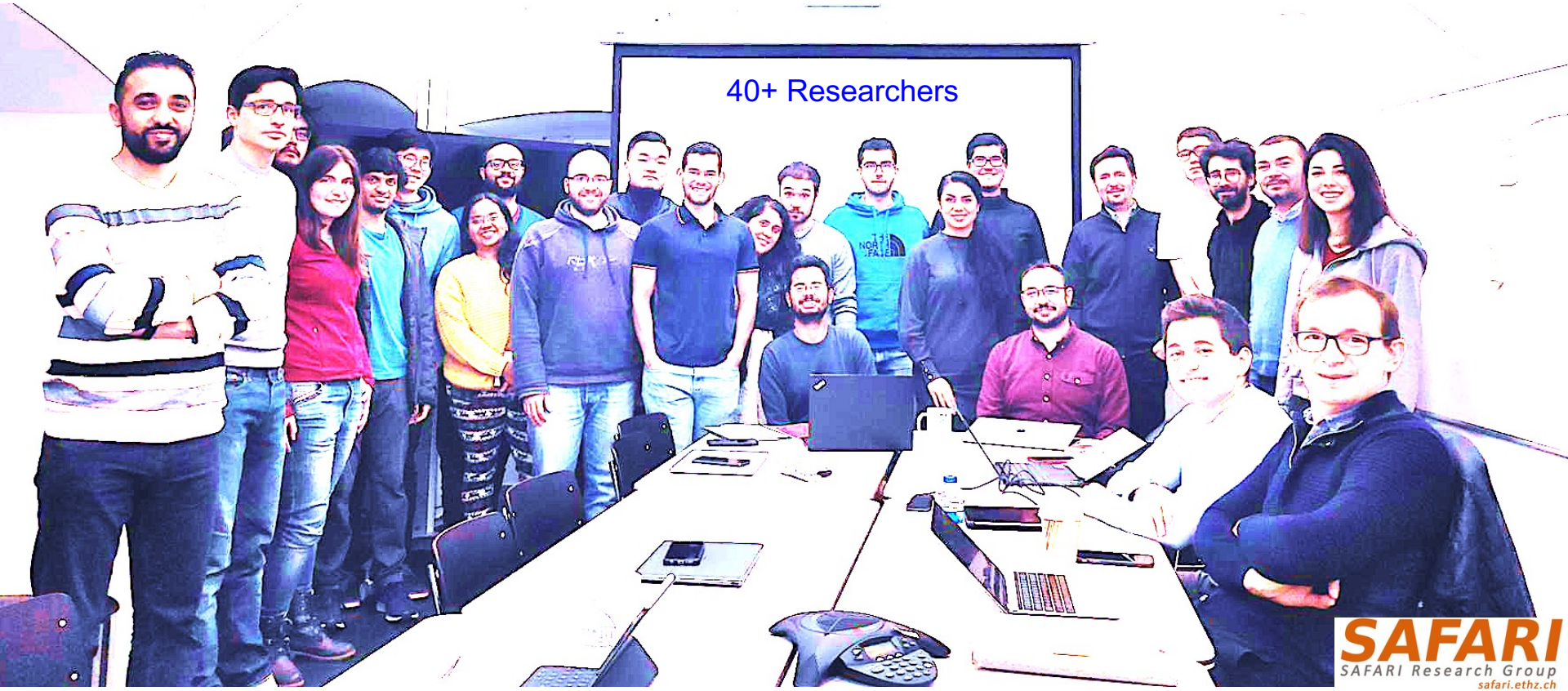
Data-driven

Data-aware

Onur Mutlu's SAFARI Research Group

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

https://safari.ethz.ch/safari-newsletter-june-2023/



Think BIG, Aim HIGH!

SAFARI Newsletter June 2023 Edition

- https://safari.ethz.ch/safari-newsletter-june-2023/

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/

Open-Source Artifacts

https://github.com/CMU-SAFARI

Open Source Tools: SAFARI GitHub

SAFARI Research Group at ETH Zurich and Carnegie Mellon University

Site for source code and tools distribution from SAFARI Research Group at ETH Zurich and Carnegie Mellon University.

ETH Zurich and Carnegie Mellon U... https://safari.ethz.ch/ omutlu@gmail.com

Overview | Repositories 62 | Projects | Packages | People 13

Pinned

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++ 304 153

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 50 21

DAMOV Public

DAMOV is a benchmark suite and a methodical framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processin...

C++ 26 3

SneakySnake Public

SneakySnake🐍 is the first and the only pre-alignment filtering algorithm that works efficiently and fast on modern CPU, FPGA, and GPU architectures. It greatly (by more than two orders of magnitude...

VHDL 40 8

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++ 143 90

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 188 41

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

SAFARI Overview at EFCL Huawei Day

- Onur Mutlu,
  "SAFARI Research Group: Introduction & Research"
  Invited Talk at the ETH Future Computing Laboratory Huawei Day, Virtual, 19 October 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (15 minutes)]

SAFARI Overview at EFCL Huawei Day

SAFARI Research Group: Introduction & Research -- ETH Future Computing Laboratory Talk - Onur Mutlu

1,939 views • Premiered Jan 15, 2022

29 DISLIKE SHARE CLIP SAVE ...

Onur Mutlu Lectures

24.9K subscribers

SUBSCRIBED

SAFARI Research Group: Introduction & Research -- ETH Future Computing Laboratory Event Talk - Onur Mutlu

https://youtu.be/mSr1QQmYuX0

Fundamentally Better Architectures

Data-centric

Data-driven

Data-aware

A Blueprint for Fundamentally Better Architectures

- Onur Mutlu,
  "Intelligent Architectures for Intelligent Computing Systems"
  *Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (**DATE**)*, Virtual, February 2021.
  [Slides (pptx) (pdf)]
  [IEDM Tutorial Slides (pptx) (pdf)]
  [Short DATE Talk Video (11 minutes)]
  [Longer IEDM Tutorial Video (1 hr 51 minutes)]

Intelligent Architectures for Intelligent Computing Systems

Onur Mutlu
ETH Zurich
omutlu@gmail.com

The Problem

Computing

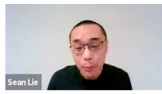
is Bottlenecked by Data

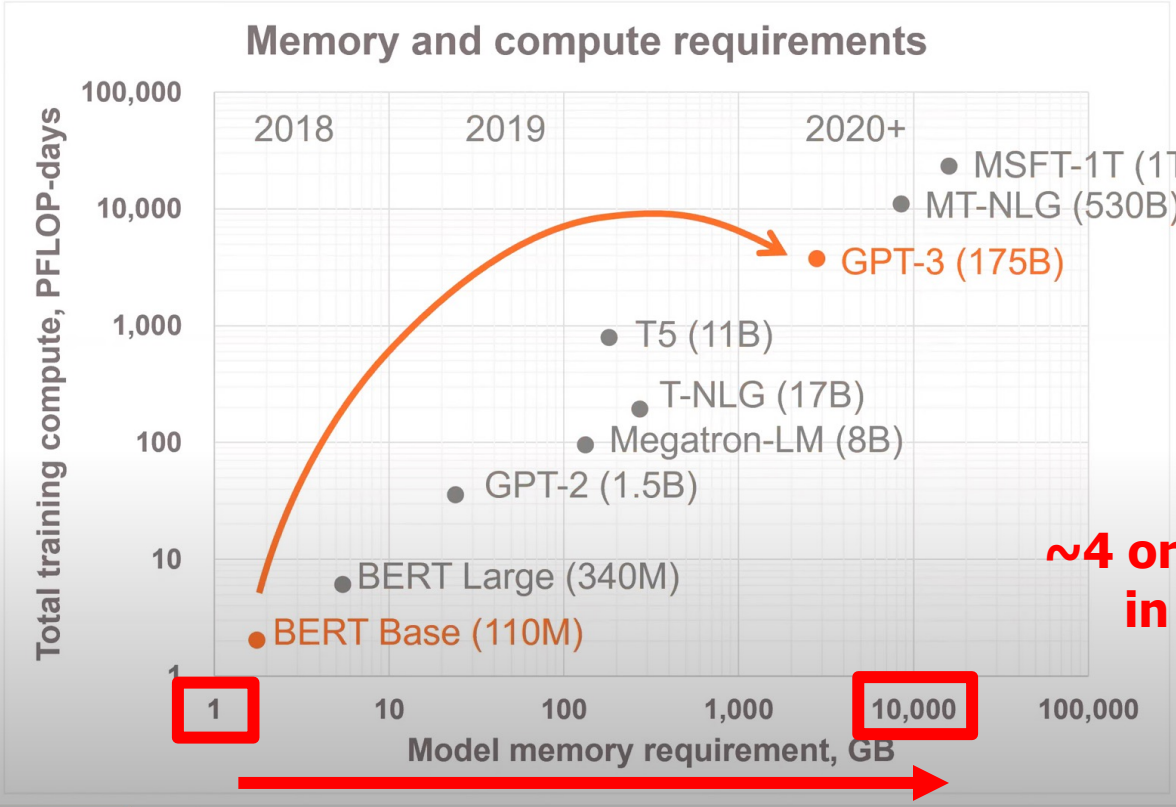
Data is Key for AI, ML, Genomics, …

- 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

Exponential Growth of Neural Networks





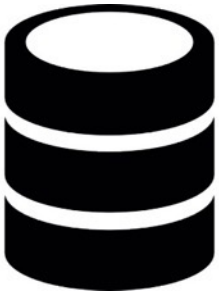
1800x more compute

In just **2 years**

Tomorrow, **multi-trillion** parameter models

~4 orders of magnitude increase in memory requirement in just two years!

Data is Key for Future Workloads



In-memory Databases

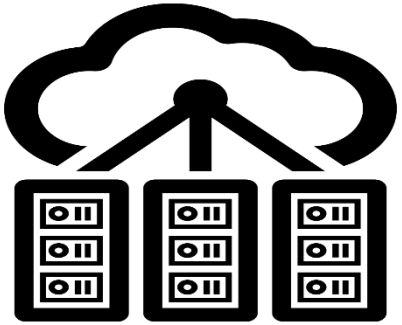
[Mao+, EuroSys'12;
Clapp+ (**Intel**), IISWC'15]

Graph/Tree Processing

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

In-Memory Data Analytics

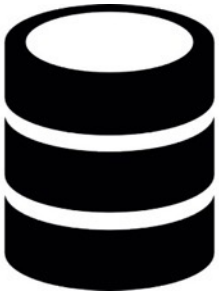
[Clapp+ (**Intel**), IISWC'15;
Awan+, BDCloud'15]



Datacenter Workloads

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

Data Overwhelms Modern Machines



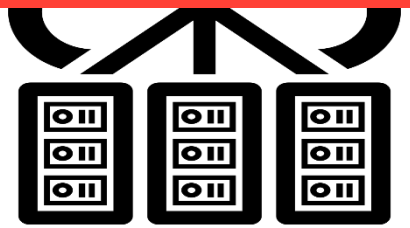
In-memory Databases

Graph/Tree Processing

Data → performance & energy bottleneck

In-Memory Data Analytics

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



Datacenter Workloads

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

Data is Key for Future Workloads

Chrome

Google's web browser

TensorFlow Mobile

Google's machine learning framework

Video Playback

Google's video codec

Video Capture

Google's video codec

Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data → performance & energy bottleneck

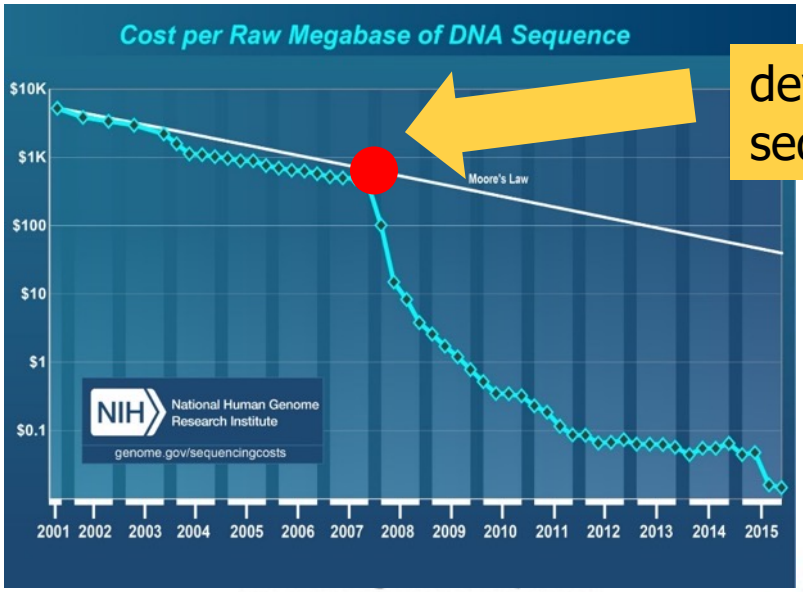
Video Playback

Google's **video codec**

Video Capture

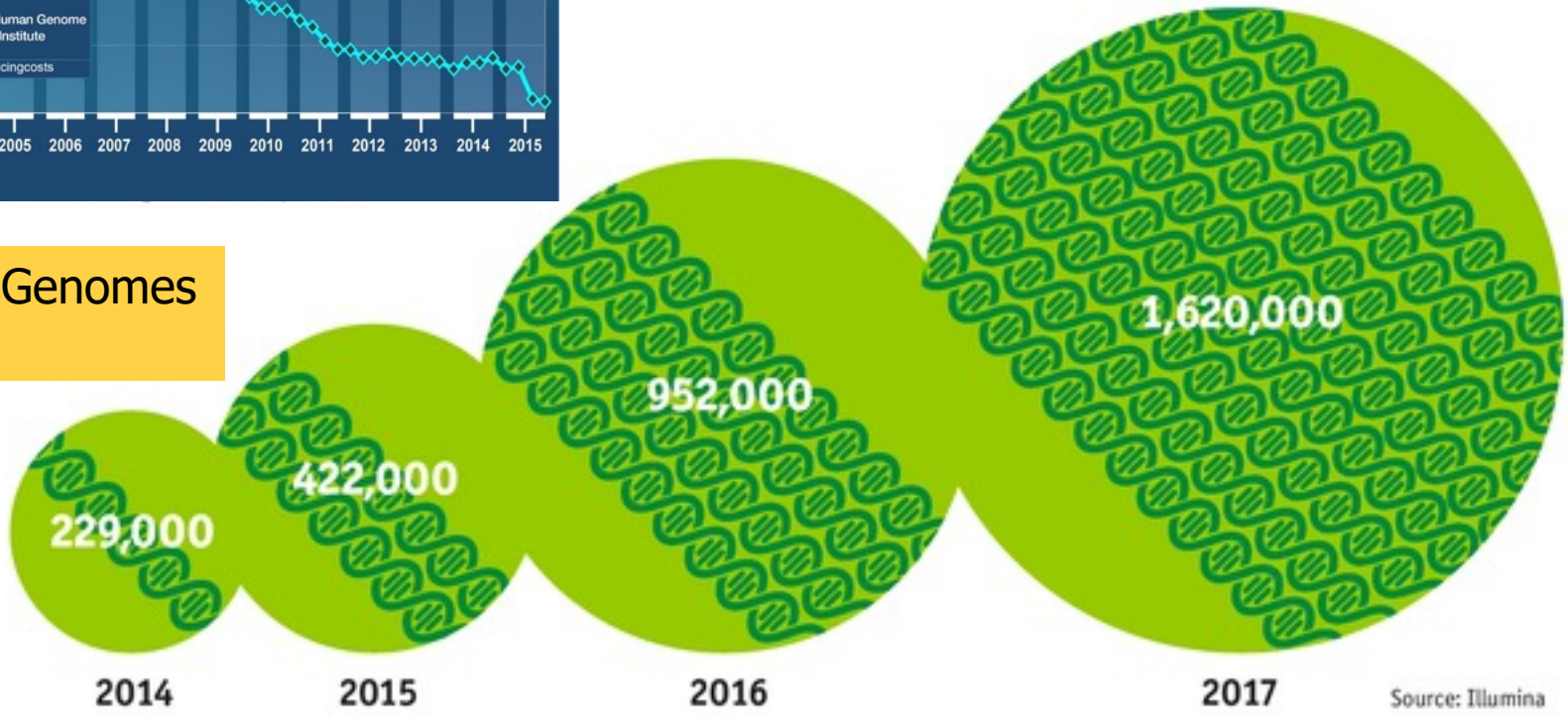
Google's **video codec**

Data is Key for Future Workloads



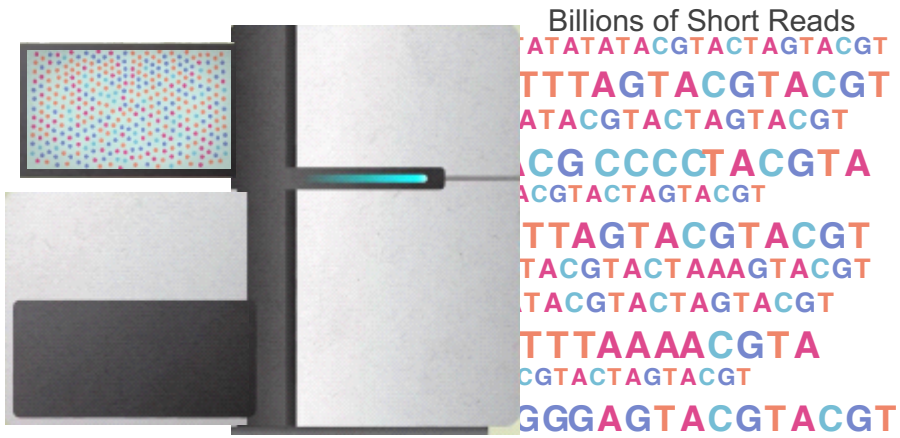
development of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced

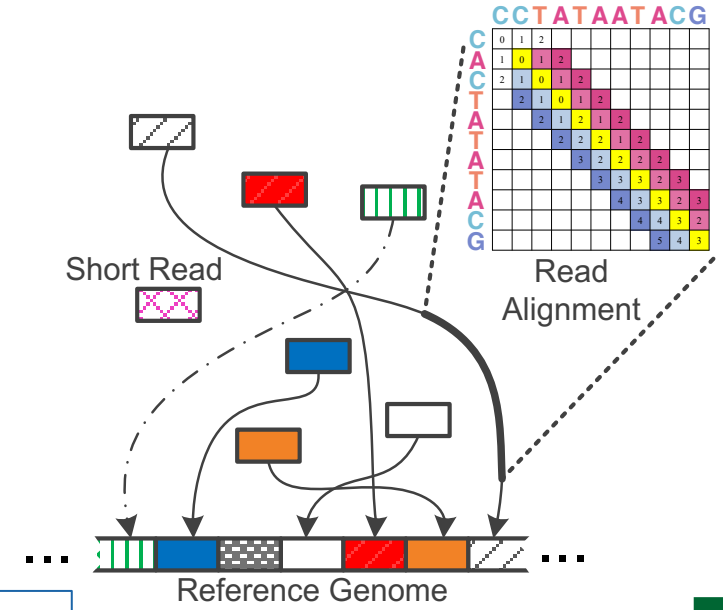


The Economist

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



1 Sequencing



Read Mapping 2

Genome Analysis

Data → performance & energy bottleneck

```
read4:          CGCTTCCAT
read5:                CCATGACGC
read6:              TTCCATGAC
```

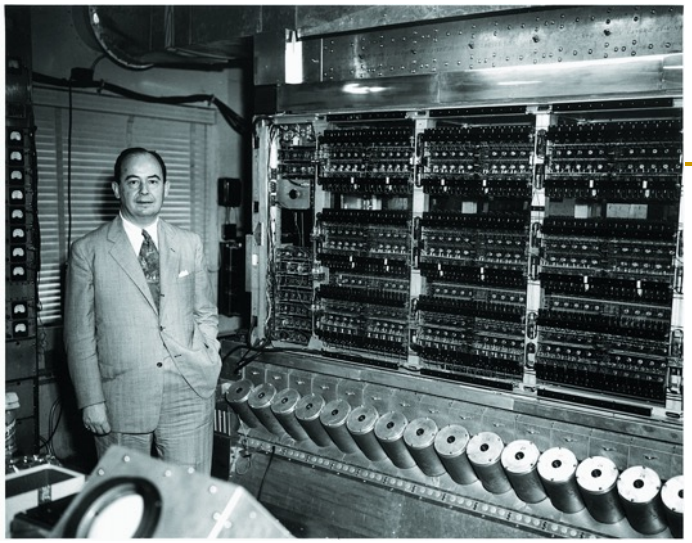
3 Variant Calling

Scientific Discovery 4

Data Overwhelms Modern Machines …

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

A Computing System



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

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

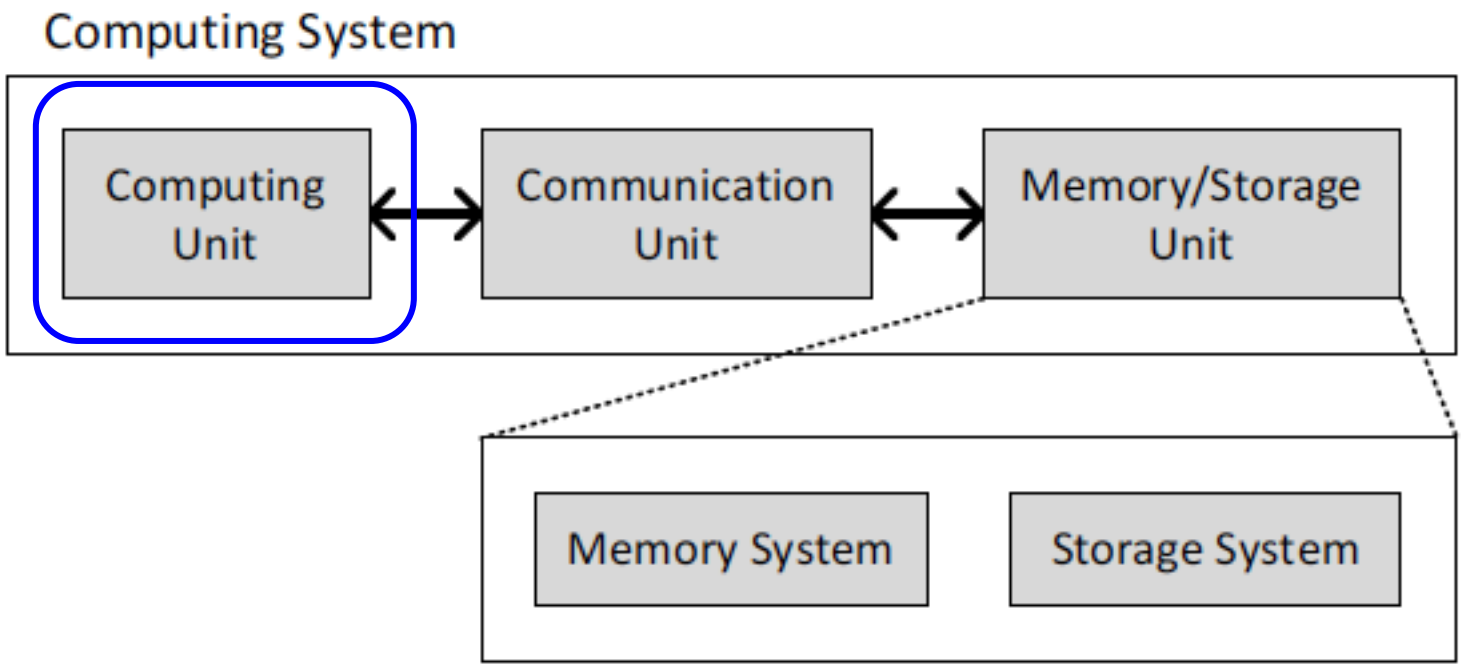
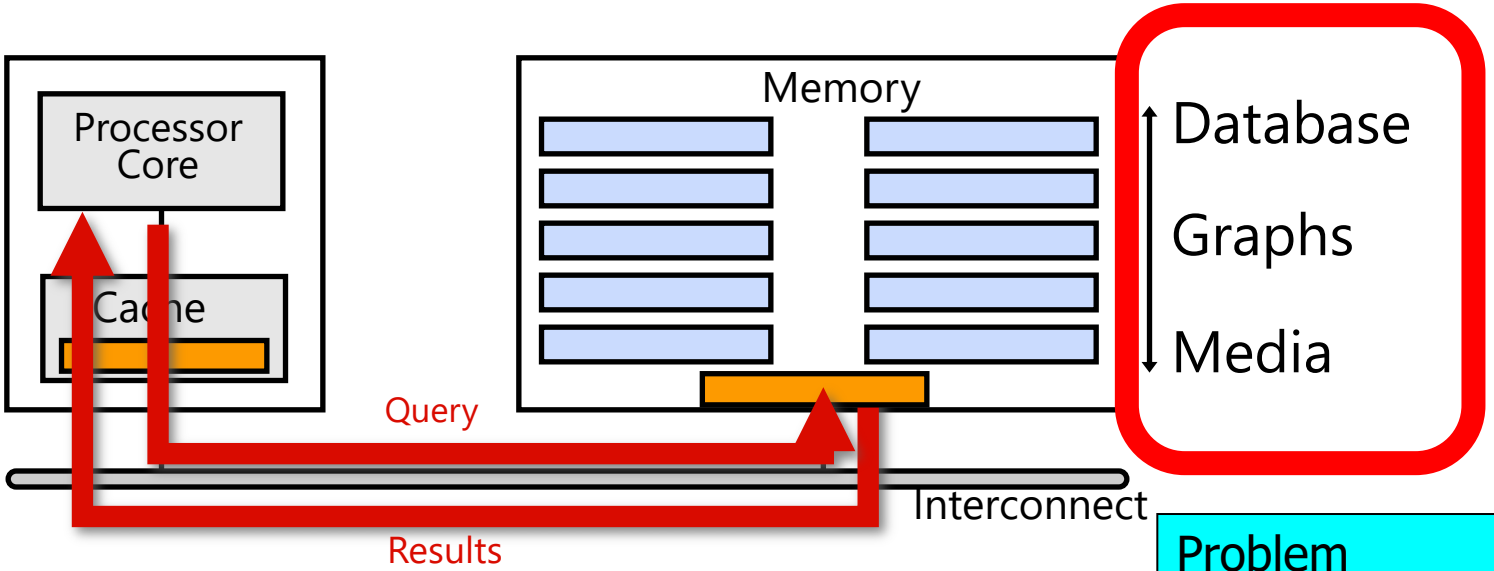


Image source: https://lbsitbytes2010.wordpress.com/2013/03/29/john-von-neumann-roll-no-15/

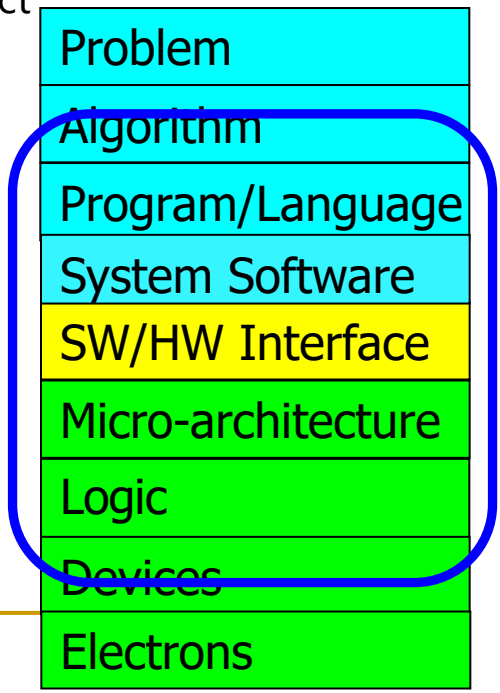
We Need A **Paradigm Shift** To …

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

Goal: Processing Inside Memory



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



Outline

1 Programming a Real PIM Architecture

Overview of recently published works

2 System Support for PuM Architectures

Overview of recently published works

3 Accelerating Key Applications with PIM

Outline

1 Programming a Real PIM Architecture

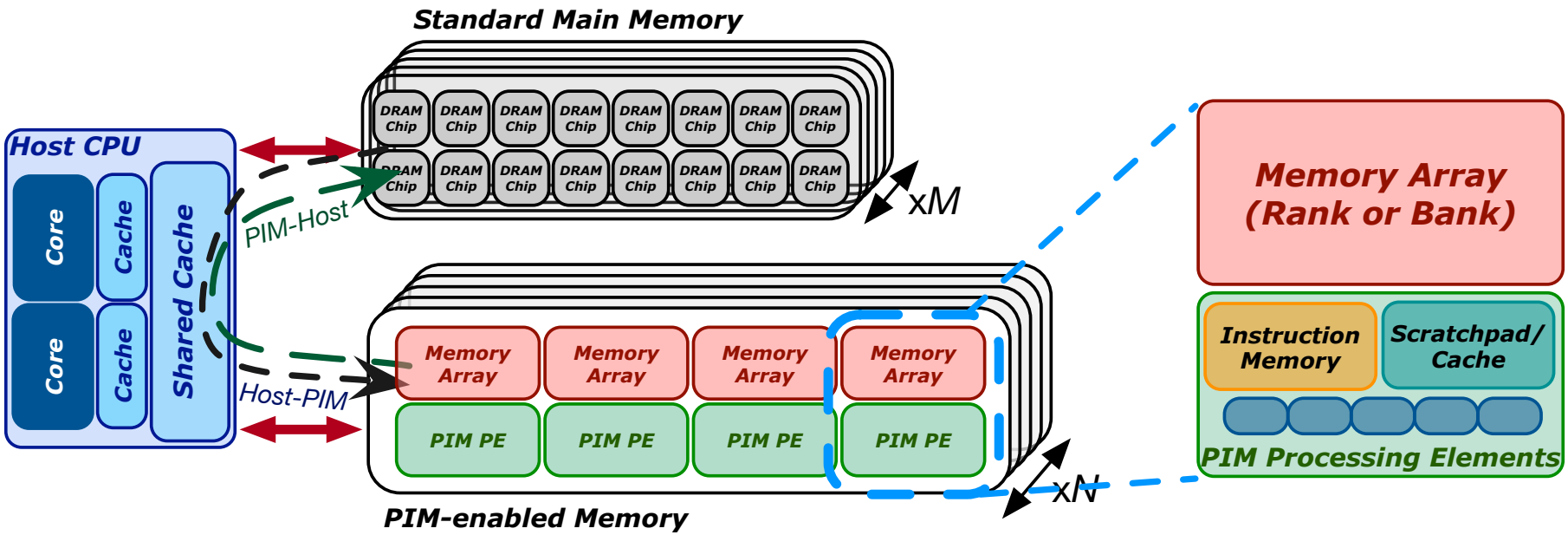
Overview of recently published works

2 System Support for PuM Architectures

Overview of recently published works

3 Accelerating Key Applications with PIM

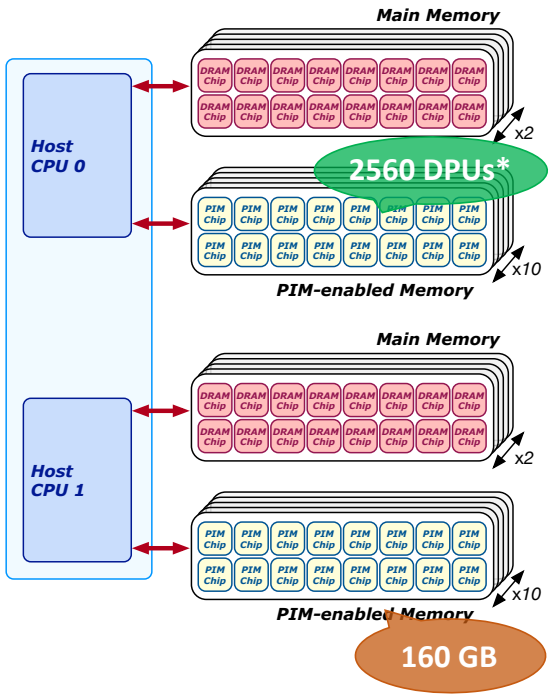
A State-of-the-Art PIM System

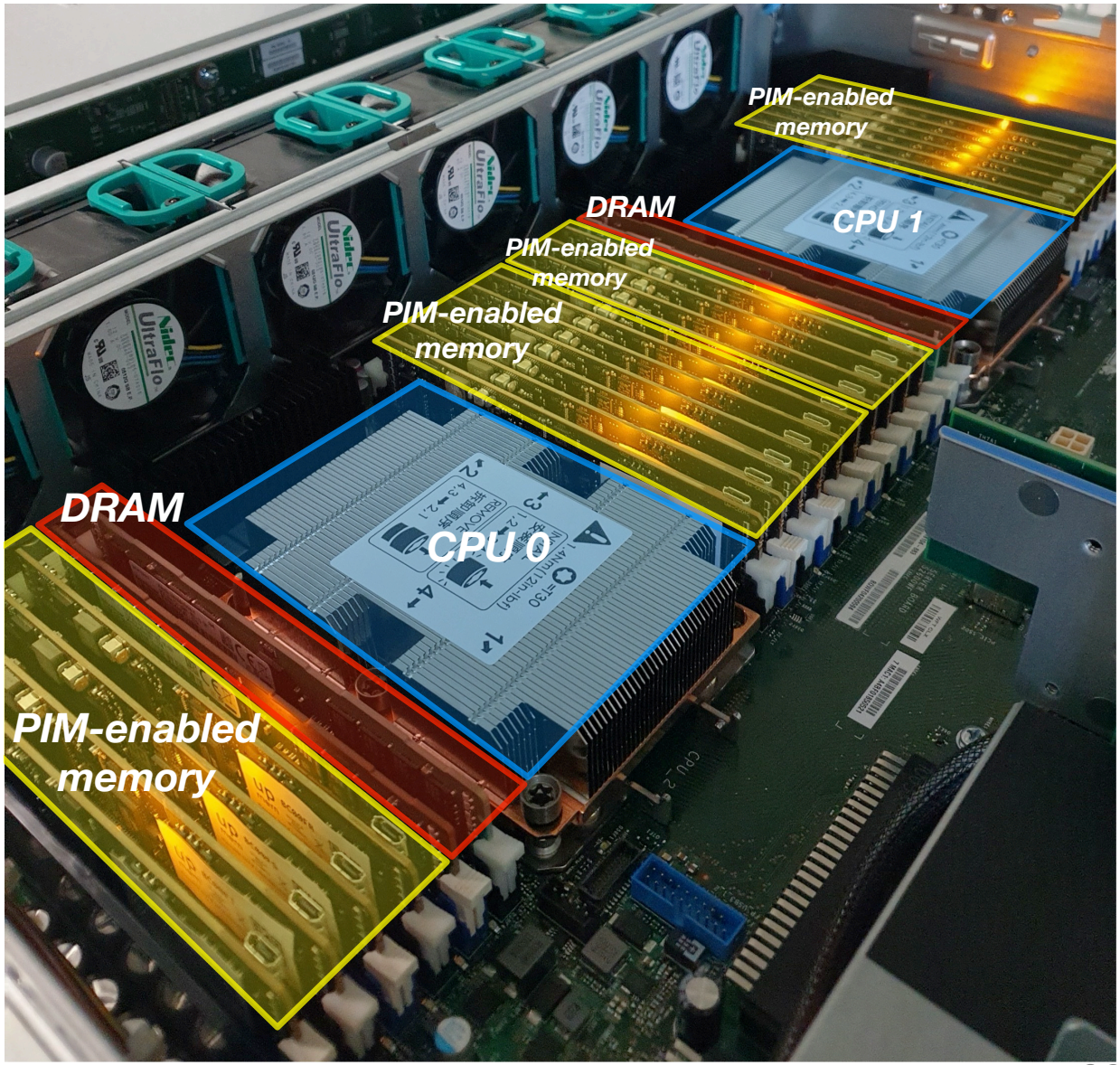


- In our work, we use the UPMEM PIM architecture
  - General-purpose processing cores called *DRAM Processing Units* (*DPUs*)
    - Up to 24 PIM threads, called *tasklets*
    - 32-bit integer arithmetic, but multiplication/division are emulated*, as well as floating-point operations
  - 64-MB DRAM bank (*MRAM*), 64-KB scratchpad (*WRAM*)

* 8-bit integer multiplication is natively supported

2,560-DPU UPMEM PIM System





- 20 UPMEM DIMMs of 16 chips each (40 ranks)
- Dual x86 socket
- UPMEM DIMMs coexist with regular DDR4 DIMMs
  - 2 memory controllers/socket
  - 2 conventional DDR4 DIMMs on one channel of one controller

SAFARI

* There are some faulty DPUs in the system that we use in our experiments. Thus, the maximum number of DPUs we can use is 2,524

Programming a Real PIM Architecture:
Overview of recently published works

1 Jinfan Chen, Juan Gómez-Luna, Izzat El Hajj, YuXin Guo, and Onur Mutlu, "***SimplePIM: A Software Framework for Productive and Efficient Processing in Memory***," in PACT, 2023.

2 Harshita Gupta, Mayank Kabra, Juan Gómez-Luna, Konstantinos Kanellopoulos, and Onur Mutlu, "***Evaluating Homomorphic Operations on a Real-World Processing-In-Memory System***," in IISWC, 2023.

3 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***," in ISPASS, 2023.

4 Maurus Item, Juan Gómez Luna, Yuxin Guo, Geraldo F. Oliveira, Mohammad Sadrosadati, and Onur Mutlu, "***TransPimLib: Efficient Transcendental Functions for Processing-in-Memory Systems***," in ISPASS, 2023.

Programming a Real PIM Architecture:
Overview of recently published works

1 Jinfan Chen, Juan Gómez-Luna, Izzat El Hajj, YuXin Guo, and Onur Mutlu, "***SimplePIM: A Software Framework for Productive and Efficient Processing in Memory***," in PACT, 2023.

2 Harshita Gupta, Mayank Kabra, Juan Gómez-Luna, Konstantinos Kanellopoulos, and Onur Mutlu, "***Evaluating Homomorphic Operations on a Real-World Processing-In-Memory System***," in IISWC, 2023.

3 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***," in ISPASS, 2023.

4 Maurus Item, Juan Gómez Luna, Yuxin Guo, Geraldo F. Oliveira, Mohammad Sadrosadati, and Onur Mutlu, "***TransPimLib: Efficient Transcendental Functions for Processing-in-Memory Systems***," in ISPASS, 2023.

2023 International Conference on Parallel Architectures and Compilation Techniques

SimplePIM:

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

Jinfan Chen, Juan Gómez Luna, Izzat El Hajj, Yuxin Guo, Onur Mutlu

https://arxiv.org/pdf/2310.01893.pdf

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

juang@ethz.ch

ETH Zürich

SAFARI

Monday, October 23, 2023

Executive Summary

- Processing-in-Memory (PIM) promises to alleviate the *data movement bottleneck*
- Real PIM hardware is now available, e.g., UPMEM PIM
- However, programming real PIM hardware is challenging, e.g.:
  - Distribute data across PIM memory banks,
  - Manage data transfers between host cores and PIM cores, and between PIM cores,
  - Launch PIM kernels on the PIM cores, etc.
- SimplePIM is a high-level programming framework for real PIM hardware
  - Iterators such as `map`, `reduce`, and `zip`
  - Collective communication with `broadcast`, `scatter`, and `gather`
- Implementation on UPMEM and evaluation with six different workloads
  - Reduction, vector add, histogram, linear/logistic regression, K-means
  - 4.4x fewer lines of code compared to hand-optimized code
  - Between 15% and 43% faster than hand-optimized code for three workloads
- Source code: https://github.com/CMU-SAFARI/SimplePIM

Programming a PIM System (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;
}
```

Programming a PIM System (II)

- 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)

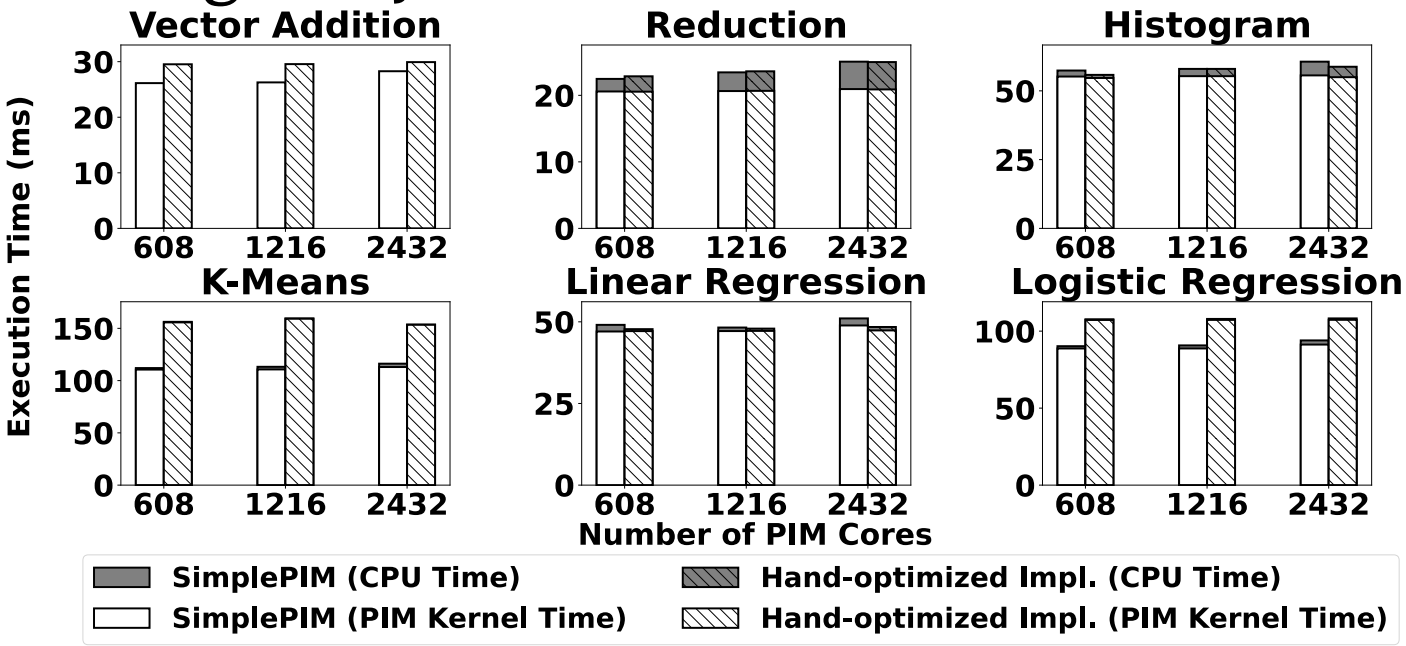
- Lines of code (LoC) reduction

	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 (I)

- Weak scaling analysis

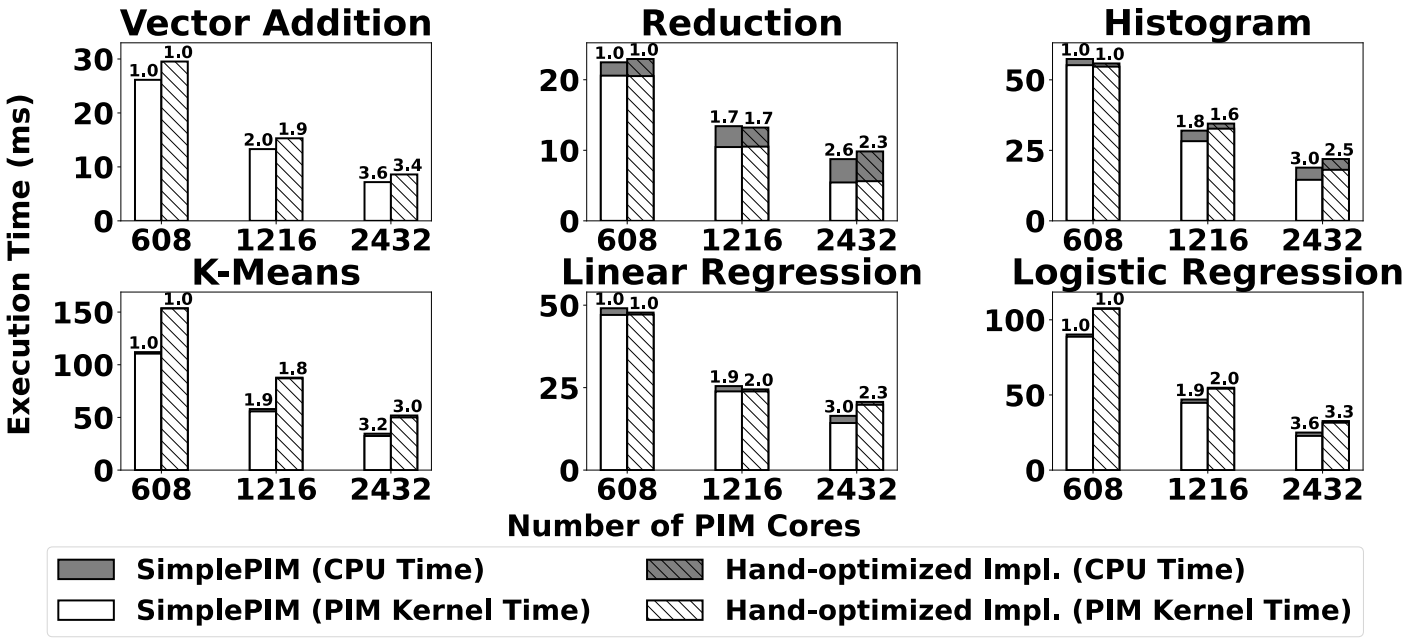


SimplePIM achieves comparable performance for `reduction`, `histogram`, and `linear regression`

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

Performance Evaluation (II)

- Strong scaling analysis



SimplePIM scales better than hand-optimized implementations for `reduction`, `histogram`, and `linear regression`

SimplePIM outperforms hand-optimized implementations for `vector addition`, `logistic regression`, and `k-means` by 15%-43%

Discussion

- SimplePIM is devised for PIM architectures with
  - A host processor with access to standard main memory and PIM-enabled memory
  - PIM processing elements (PEs) that communicate via the host processor
  - The number of PIM PEs scales with memory capacity
- SimplePIM emulates the communication between PIM cores via the host processor
- Other parallel patterns can be incorporated in future work
  - Prefix sum and filter can be easily added
  - Stencil and convolution would require fine-grained scatter-gather for halo cells
  - Random access patterns would be hard to support

SimplePIM: arXiv Version

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

https://arxiv.org/pdf/2310.01893.pdf

Source Code

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

SimplePIM Private

Edit Pins | Unwatch 3

main | 1 branch | 0 tags | Go to file | Add file | Code

Wangsitu98 interface cleanups, added allreduce and allgather 3421614 2 days ago 7 commits

benchmarks	interface cleanups, added allreduce and allgather	2 days ago
lib	interface cleanups, added allreduce and allgather	2 days ago
.gitignore	some cleanups	3 weeks ago
README.md	pushed SimplePIM	last month

README.md

SimplePIM

This project implements SimplePIM, a software framework for easy and efficient in-memory-hardware programming. The code is implemented on UPMEM, an actual, commercially available PIM hardware that combines traditional DRAM memory with general-purpose in-order cores inside the same chip. SimplePIM processes arrays of arbitrary elements on a PIM device by calling iterator functions from the host and provides primitives for communication among PIM cores and between PIM and the host system.

We implement six applications with SimplePIM on UPMEM:

- Vector Addtition
- Reduction
- K-Means Clustering
- Histogram
- Linear Regression
- Logistic Regression

Previous manual UPMEM implementations of the same applications can be found in PrIM benchmark (https://github.com/CMU-SAFARI/prim-benchmarks), dpu_kmeans (https://github.com/upmem/dpu_kmeans) and prim-ml (https://github.com/CMU-SAFARI/pim-ml). These previous implementations can serve as baseline for measuring SimplePIM's performance as well as productivity improvements.

Programming a Real PIM Architecture:
Overview of recently published works

1 Jinfan Chen, Juan Gómez-Luna, Izzat El Hajj, YuXin Guo, and Onur Mutlu, "***SimplePIM: A Software Framework for Productive and Efficient Processing in Memory***," in PACT, 2023.

2 Harshita Gupta, Mayank Kabra, Juan Gómez-Luna, Konstantinos Kanellopoulos, and Onur Mutlu, "***Evaluating Homomorphic Operations on a Real-World Processing-In-Memory System***," in IISWC, 2023.

3 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***," in ISPASS, 2023.

4 Maurus Item, Juan Gómez Luna, Yuxin Guo, Geraldo F. Oliveira, Mohammad Sadrosadati, and Onur Mutlu, "***TransPimLib: Efficient Transcendental Functions for Processing-in-Memory Systems***," in ISPASS, 2023.

2023 IEEE International Symposium on Performance Analysis of Systems and Software

Evaluating Machine Learning Workloads on Memory-Centric Computing Systems

Juan Gómez Luna, Yuxin Guo, Sylvan Brocard,
Julien Legriel, Remy Cimadomo, Geraldo F. Oliveira,
Gagandeep Singh, Onur Mutlu

https://arxiv.org/pdf/2207.07886.pdf
https://github.com/CMU-SAFARI/pim-ml
juang@ethz.ch

ETH Zürich SAFARI up mem

Monday, April 24, 2023

Executive Summary

Problem: Training machine learning (ML) algorithms is a computationally expensive process, frequently memory-bound

- Memory-centric computing systems can alleviate data movement bottlenecks
- Real-world PIM systems have only been manufactured and commercialized
- UPMEM has designed and fabricated the first publicly-available PIM architecture

Goal: Understand the potential of modern general-purpose PIM architectures to accelerate machine learning training

Our main contributions:

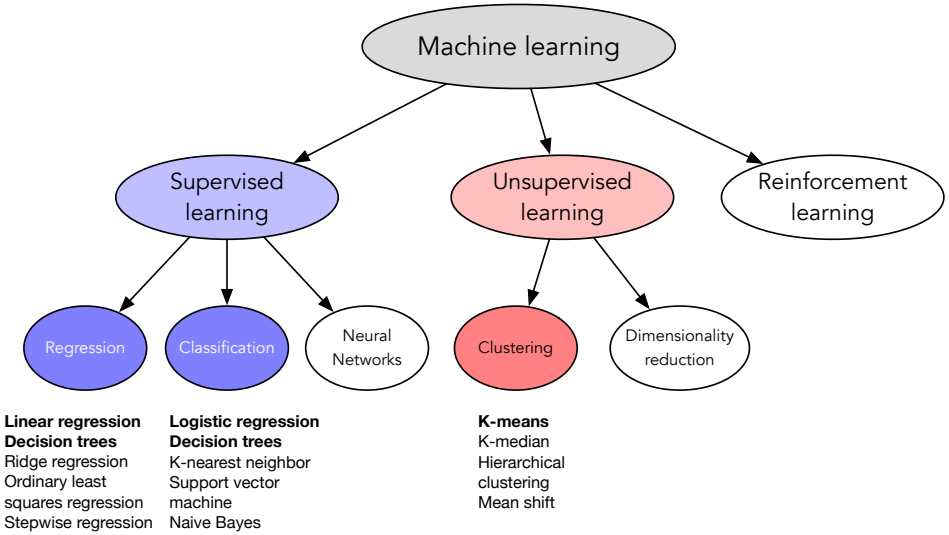
- PIM implementation of several classic machine learning algorithms: linear regression, logistic regression, decision tree, K-means clustering
- Workload characterization in terms of quality, performance, and scaling
- Comparison to their counterpart implementations on processor-centric systems (CPU and GPU)

Key Results:

- PIM version of DTR is 27x / 1.34x faster than the CPU / GPU version, respectively
- PIM version of KME is 2.8x / 3.2x faster than the CPU / GPU version, respectively

Machine Learning Workloads

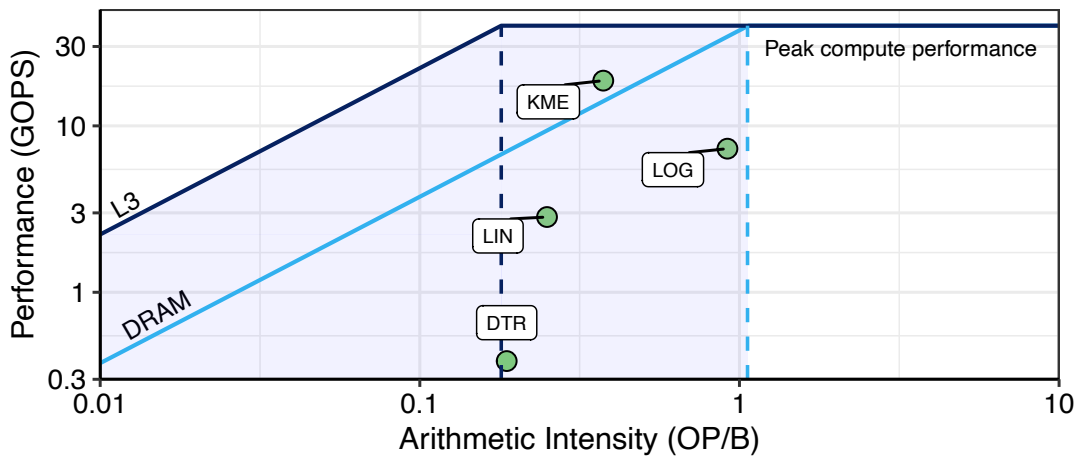
- Machine learning training with large amounts of data is a computationally expensive process, which requires many iterations to update an ML model's parameters



- Frequent data movement between memory and processing elements to access training data
- The amount of computation is not enough to amortize the cost of moving training data to the processing elements
  - Low arithmetic intensity
  - Low temporal locality
  - Irregular memory accesses

Machine Learning Workloads: Our Goal

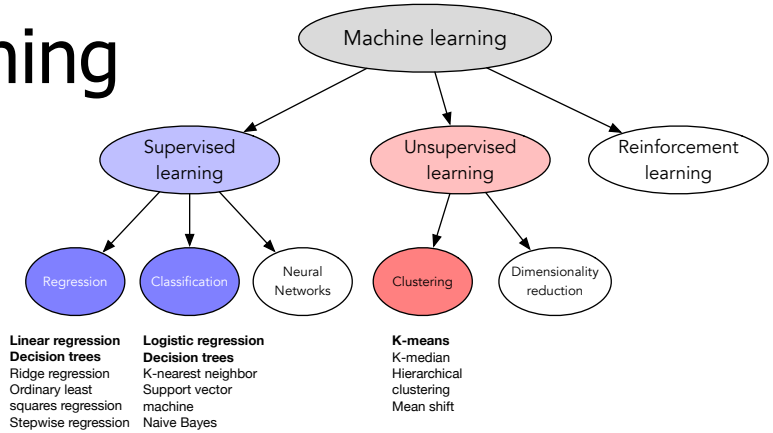
- Our goal is to study and analyze how real-world general-purpose PIM can accelerate ML training
- Four representative ML algorithms: linear regression, logistic regression, decision tree, K-means
- Roofline model to quantify memory boundedness of CPU versions



All workloads fall in the memory-bound area of the Roofline

ML Training Workloads

- Four widely-used machine learning workloads:
  - Linear regression (LIN)
  - Logistic regression (LOG)
  - Decision tree (DTR)
  - K-means clustering (KME)



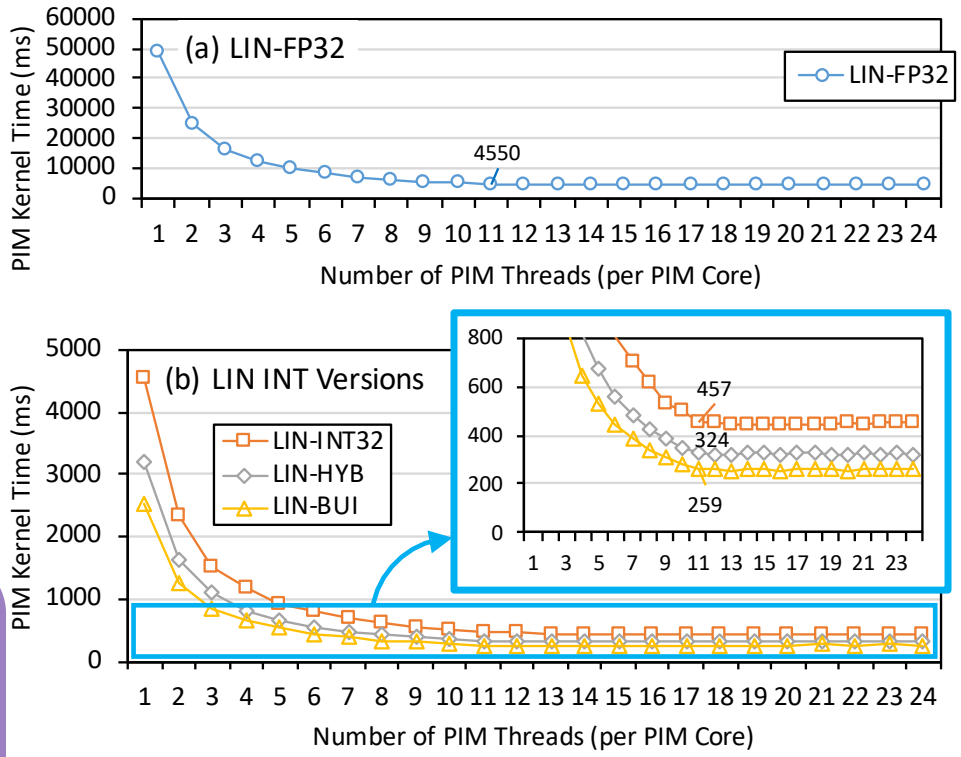
- Diversity of our ML training workloads:
  - Memory access patterns
  - Operations and datatypes
  - Communication/synchronization

Learning approach	Application	Algorithm	Short name	Memory access pattern			Computation pattern		Communication/synchronization	
				Sequential	Strided	Random	Operations	Datatype	Intra PIM Core	Inter PIM Core
Supervised	Regression	**Linear Regression**	LIN	Yes	No	No	mul, add	float, int32_t	barrier	Yes
	Classification	**Logistic Regression**	LOG	Yes	No	No	mul, add, exp, div	float, int32_t	barrier	Yes
		Decision Tree	DTR	Yes	No	No	compare, add	float	barrier, mutex	Yes
Unsupervised	Clustering	**K-Means**	KME	Yes	No	No	mul, compare, add	int16_t, int64_t	barrier, mutex	Yes

Evaluation: Analysis of PIM Kernels (I)

- Linear regression

Fixed-point representation accelerates the kernel by an order of magnitude over FP32



Key Takeaway 1. Workloads with **arithmetic operations or datatypes not natively supported** by PIM cores run at low performance due to instruction emulation (e.g., FP in UPMEM PIM).

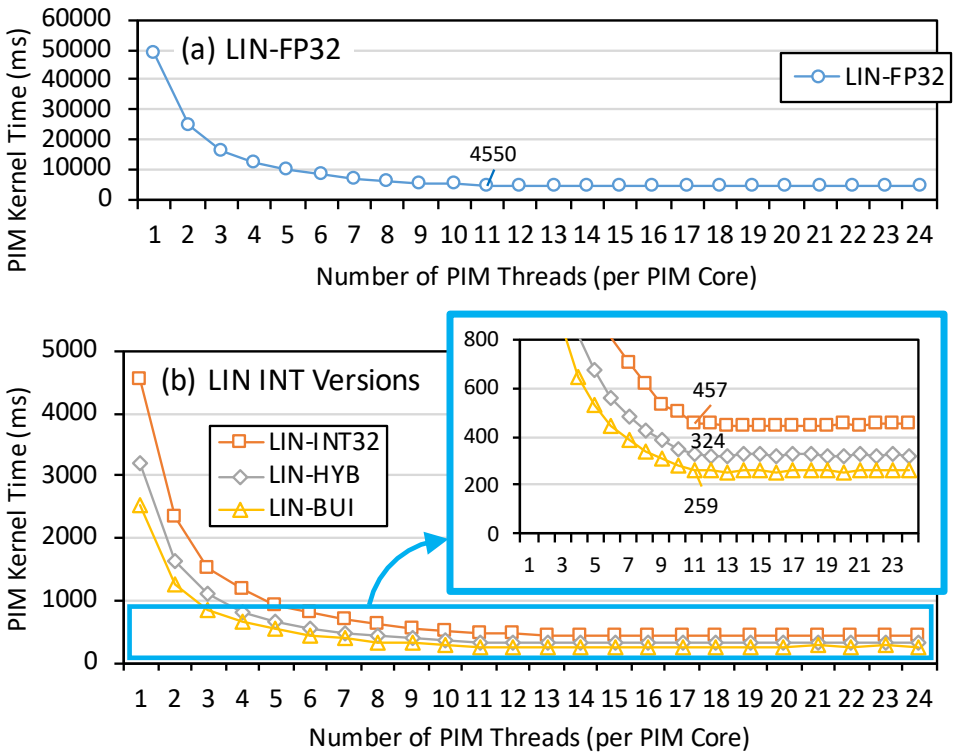
Recommendation 1. Use **fixed-point representation**, without much accuracy loss, if PIM cores do not support FP.

Evaluation: Analysis of PIM Kernels (II)

- Linear regression

LIN-HYB is 41% faster than LIN-INT32

LIN-BUI provides an additional 25% speedup



Recommendation 2.

Quantization can take advantage of native hardware support. **Hybrid precision** can significantly improve performance.

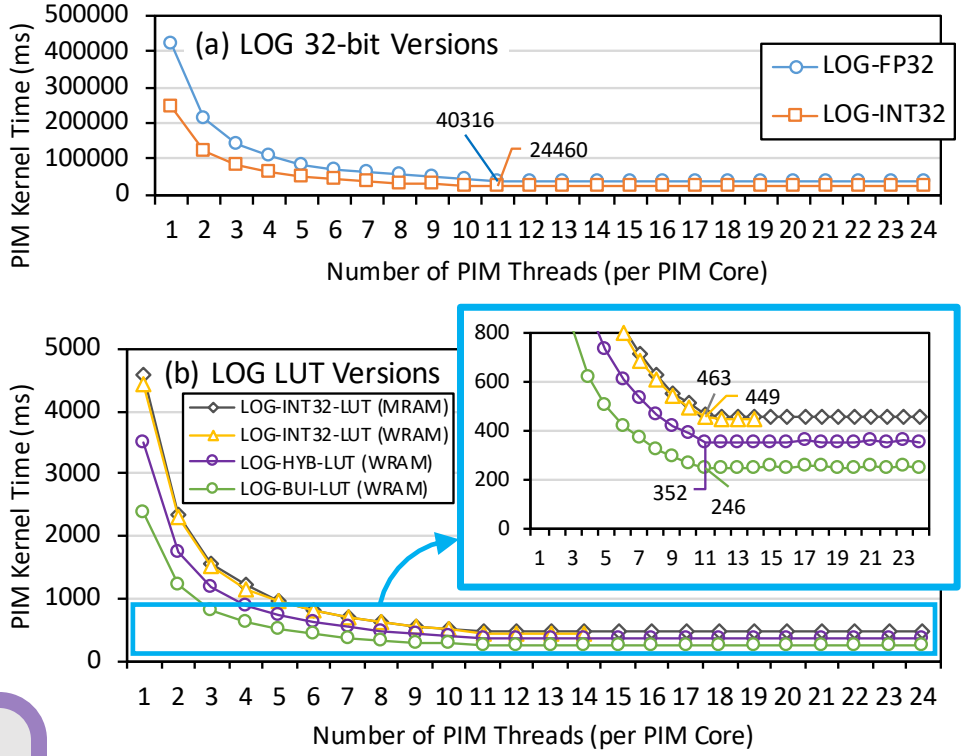
Recommendation 3.
Programmers/better compilers can **optimize code by leveraging native instructions** (e.g., 8-bit integer multiplication in UPMEM) .

Evaluation: Analysis of PIM Kernels (III)

- Logistic regression

Very high kernel time of LOG-FP32 and LOG-INT32 due to Sigmoid approximation

LOG-INT32-LUT(MRAM) is 53x faster than LOG-INT32



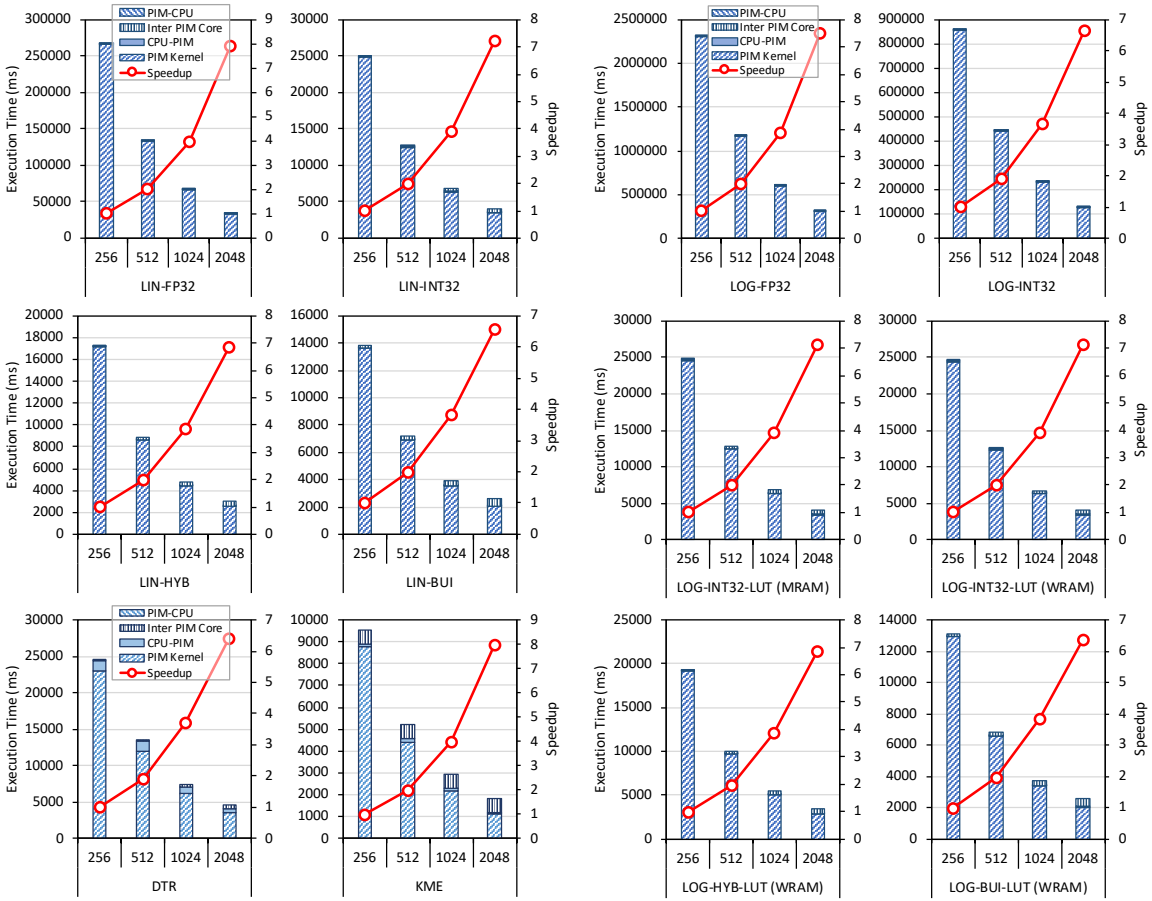
LOG-HYB-LUT is 28% faster than LOG-INT32-LUT

LOG-BUI-LUT provides an additional 43% speedup

Recommendation 4.
Convert computation to memory accesses by **keeping pre-calculated operation results** (e.g., LUTs, memoization) **in memory**.

Evaluation: Performance Scaling (I)

- Strong scaling: 256 to 2,048 PIM cores



PIM kernel time scales linearly with the number of PIM cores

Little overhead from inter PIM core communication and communication between host and PIM cores

Conclusion

Problem: Training machine learning (ML) algorithms is a computationally expensive process, frequently memory-bound

- Memory-centric computing systems can alleviate data movement bottlenecks
- Real-world PIM systems have only been manufactured and commercialized
- UPMEM has designed and fabricated the first publicly-available PIM architecture

Goal: Understand the potential of modern general-purpose PIM architectures to accelerate machine learning training

Our main contributions:

- PIM implementation of several classic machine learning algorithms: linear regression, logistic regression, decision tree, K-means clustering
- Workload characterization in terms of quality, performance, and scaling
- Comparison to their counterpart implementations on processor-centric systems (CPU and GPU)

Key Results:

- PIM version of DTR is 27x / 1.34x faster than the CPU / GPU version, respectively
- PIM version of KME is 2.8x / 3.2x faster than the CPU / GPU version, respectively

Next Steps for Real PIM Systems

- **Frameworks to ease PIM programmability**
  - Goal: A framework that can automatically distribute input and gather output data, handle memory management, and parallelize work across PIM cores
- **Benchmark and analyze other real PIM architectures**
  - Samsung's HBM-PIM
  - SK Hynix's AiM
- **Design Other Applications on PIM Systems**
  - Database primitives
  - Genomics
  - DNN training
  - Homomorphic encryption

Real PIM Tutorial (ISCA 2023)

- June 18th: Lectures + Hands-on labs + Invited lectures

ISCA 2023 Real-World PIM Tutorial

Search

Recent Changes Media Manager Sitemap

Trace: • start

start

Table of Contents

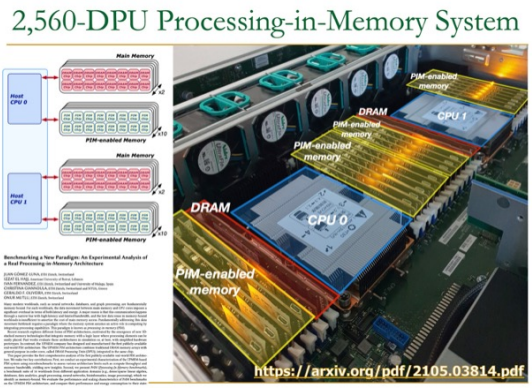
Real-world Processing-in-Memory Systems for Modern Workloads

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.

Real PIM Tutorial (MICRO 2023)

- Oct. 28th: Lectures + Hands-on labs + Invited lectures

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.

Table of Contents

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.

Outline

1 Programming a Real PIM Architecture

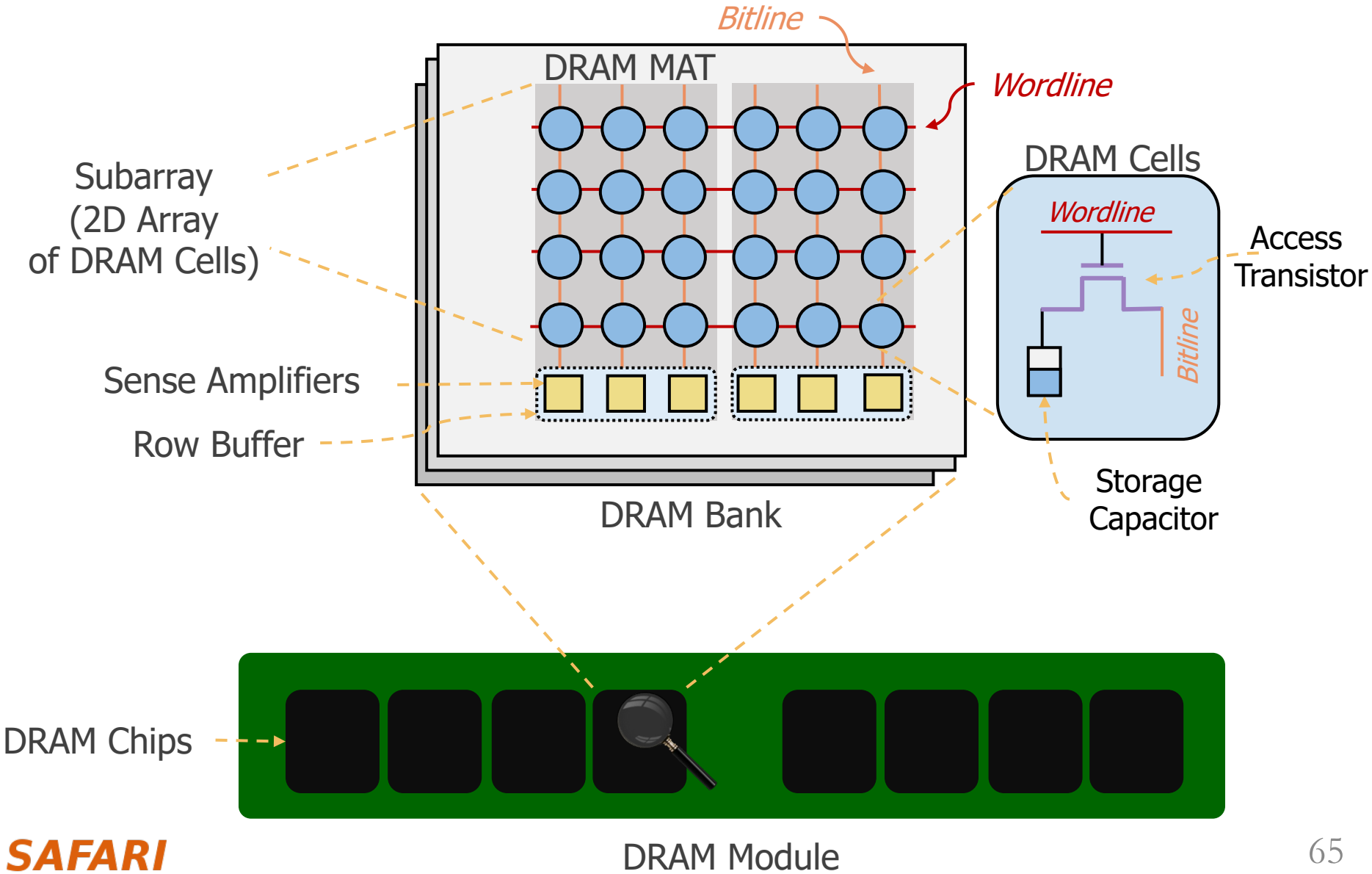
Overview of recently published works

2 System Support for PuM Architectures

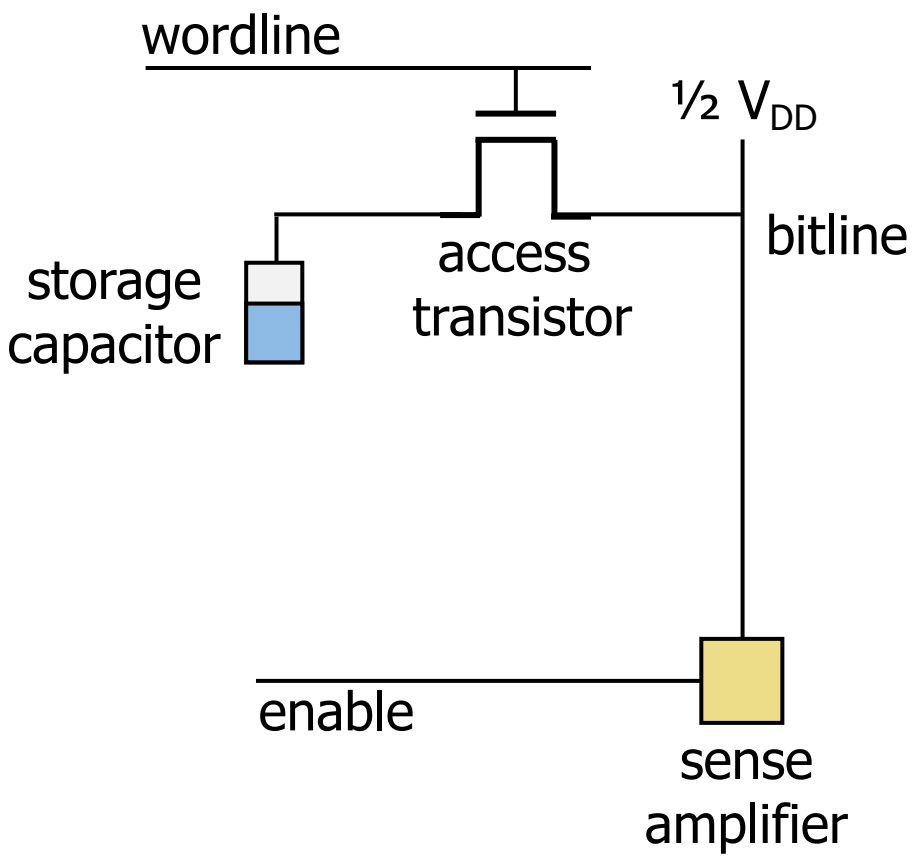
Overview of recently published works

3 Accelerating Key Applications with PIM

Inside a DRAM Chip



DRAM Cell Operation

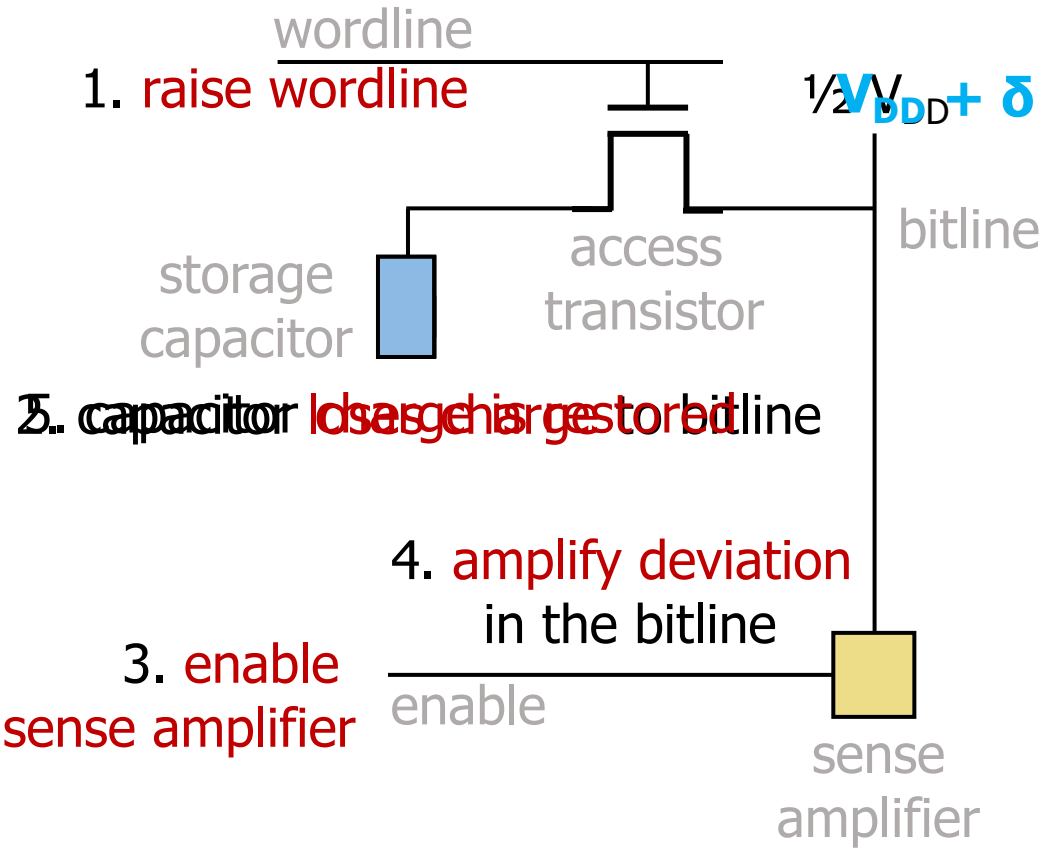


1. ACTIVATE (ACT)

2. READ/WRITE

3. PRECHARGE (PRE)

DRAM Cell Operation (1/3)

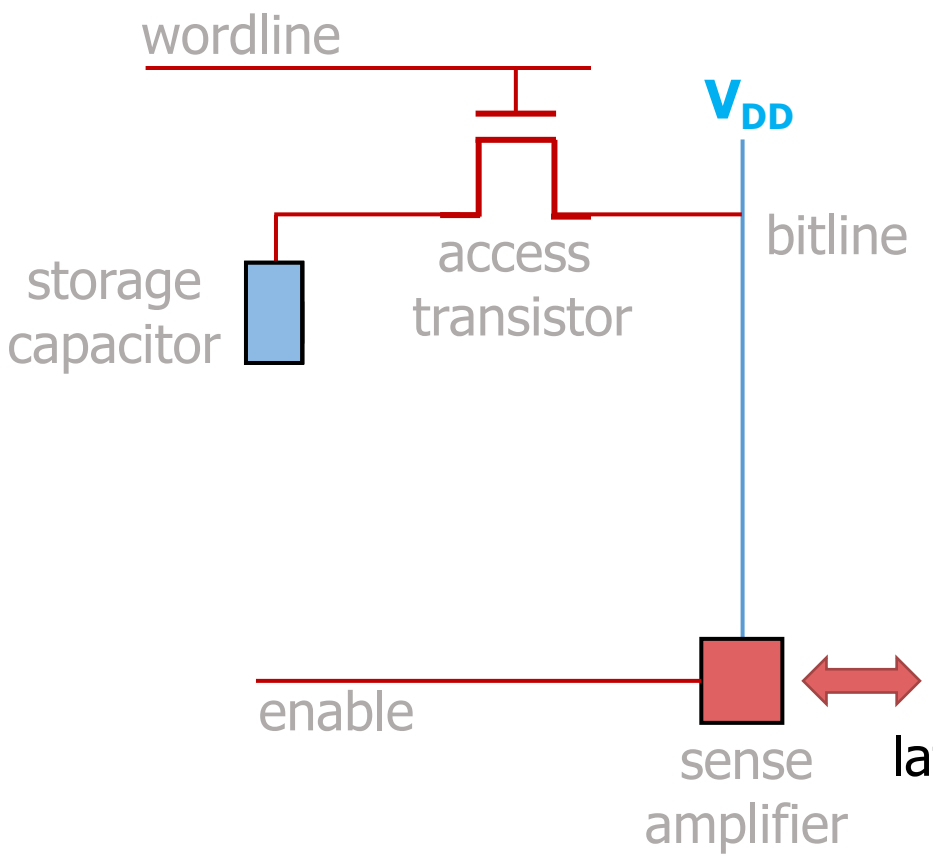


1. ACTIVATE (ACT)

2. READ/WRITE

3. PRECHARGE (PRE)

DRAM Cell Operation (2/3)



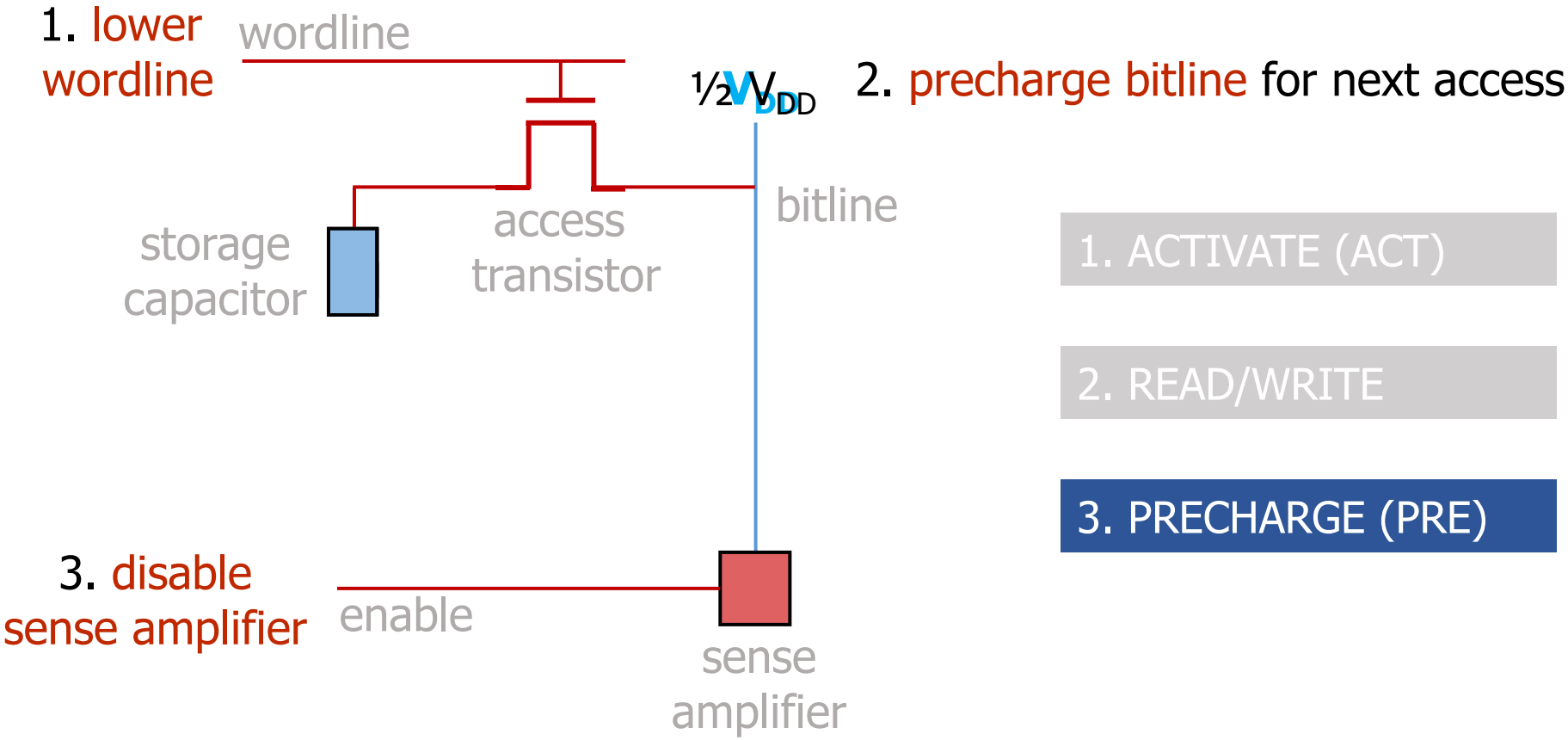
1. ACTIVATE (ACT)

2. READ/WRITE

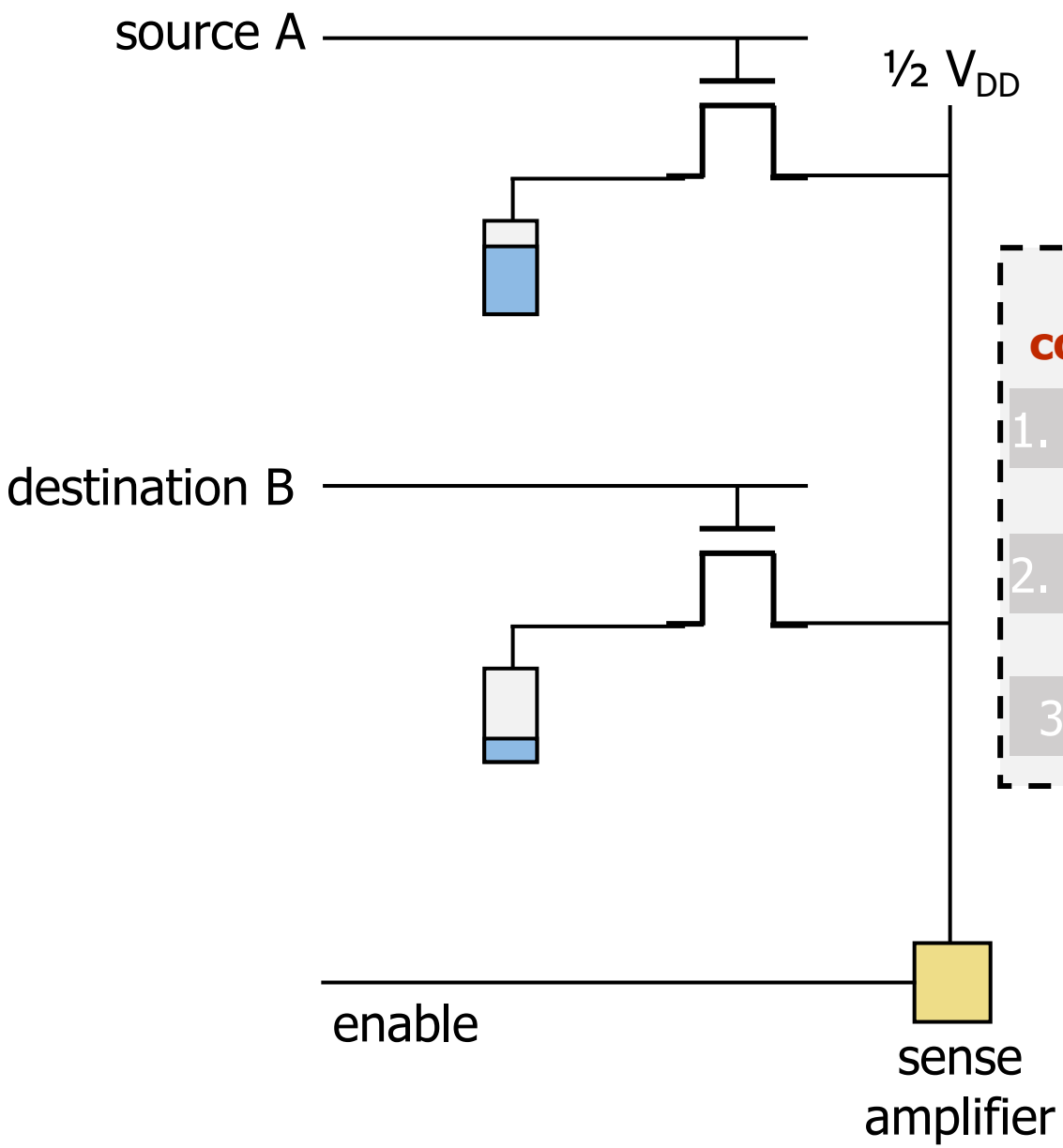
3. PRECHARGE (PRE)

read/write charge latched in sense amplifier

DRAM Cell Operation (3/3)



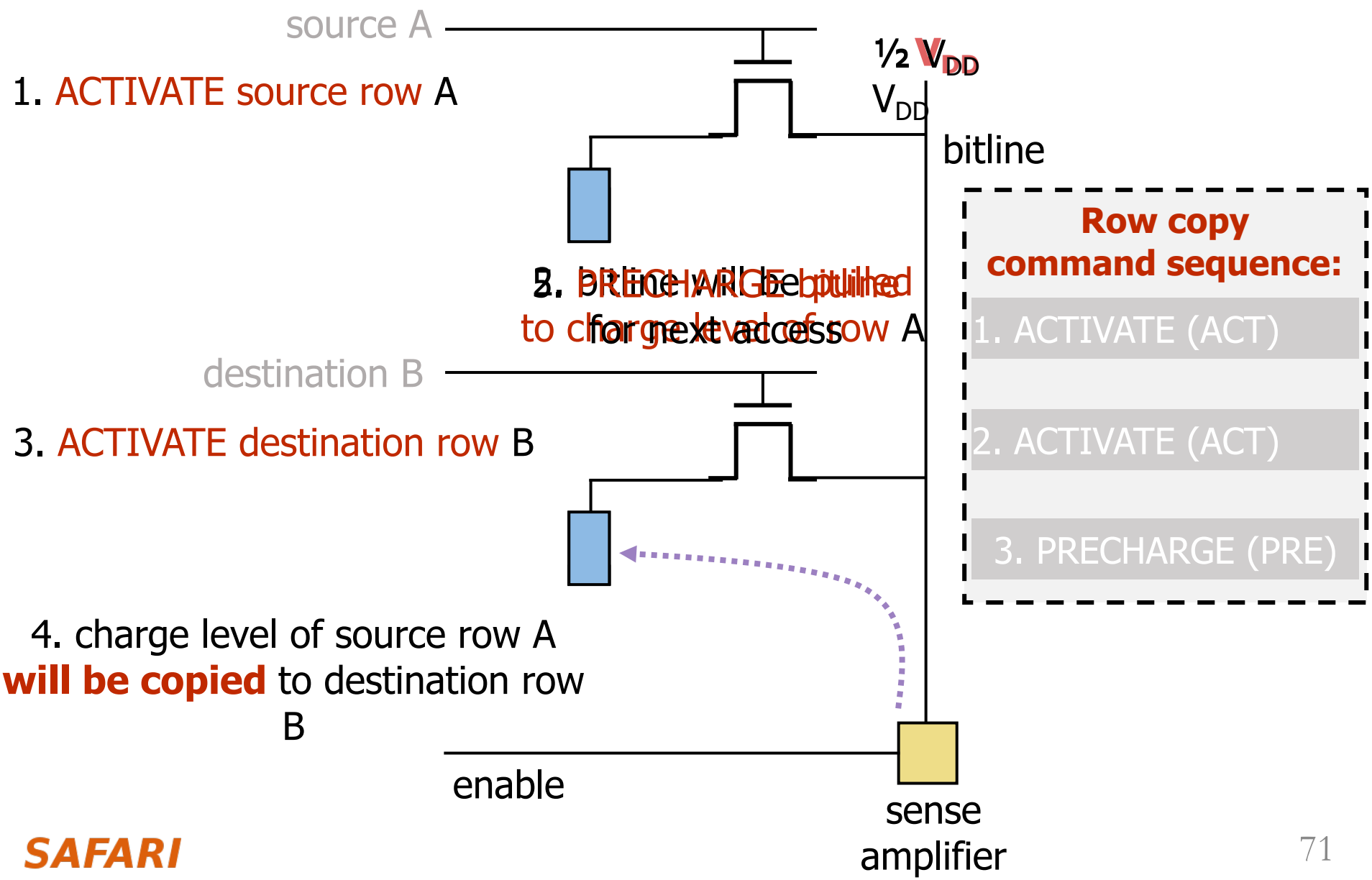
RowClone: In-DRAM Row Copy (1/2)



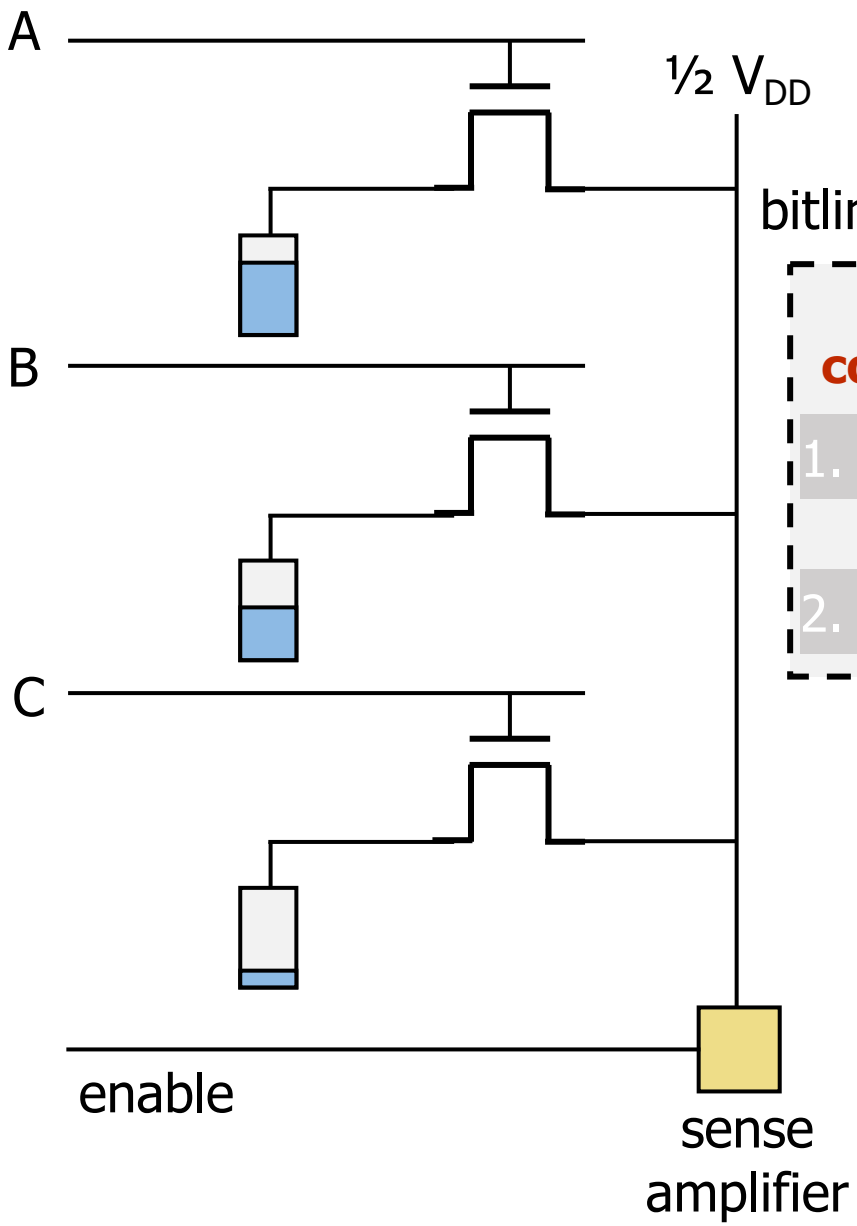
Row copy command sequence:

1. ACTIVATE (ACT)
2. ACTIVATE (ACT)
3. PRECHARGE (PRE)

RowClone: In-DRAM Row Copy (2/2)



Triple-Row Activation: Majority Function



Majority function command sequence:

1. ACTIVATE (ACT)
2. PRECHARGE (PRE)

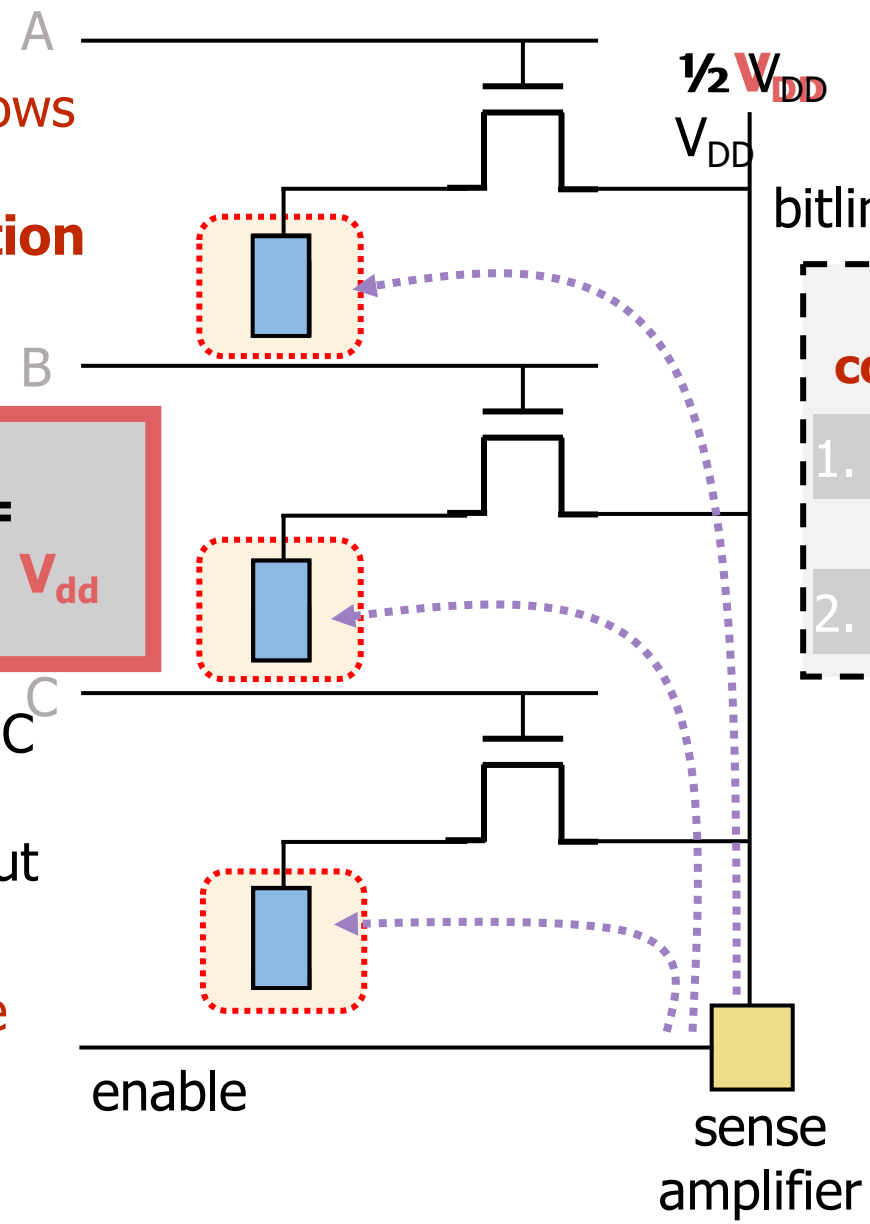
Triple-Row Activation: Majority Function

1. ACTIVATE three rows simultaneously
→ **triple-row activation**

MAJ(A, B, C) =
MAJ(V_{dd}, V_{dd}, 0) = V_{dd}

3. values in cells A, B, C will be overwritten with the majority output

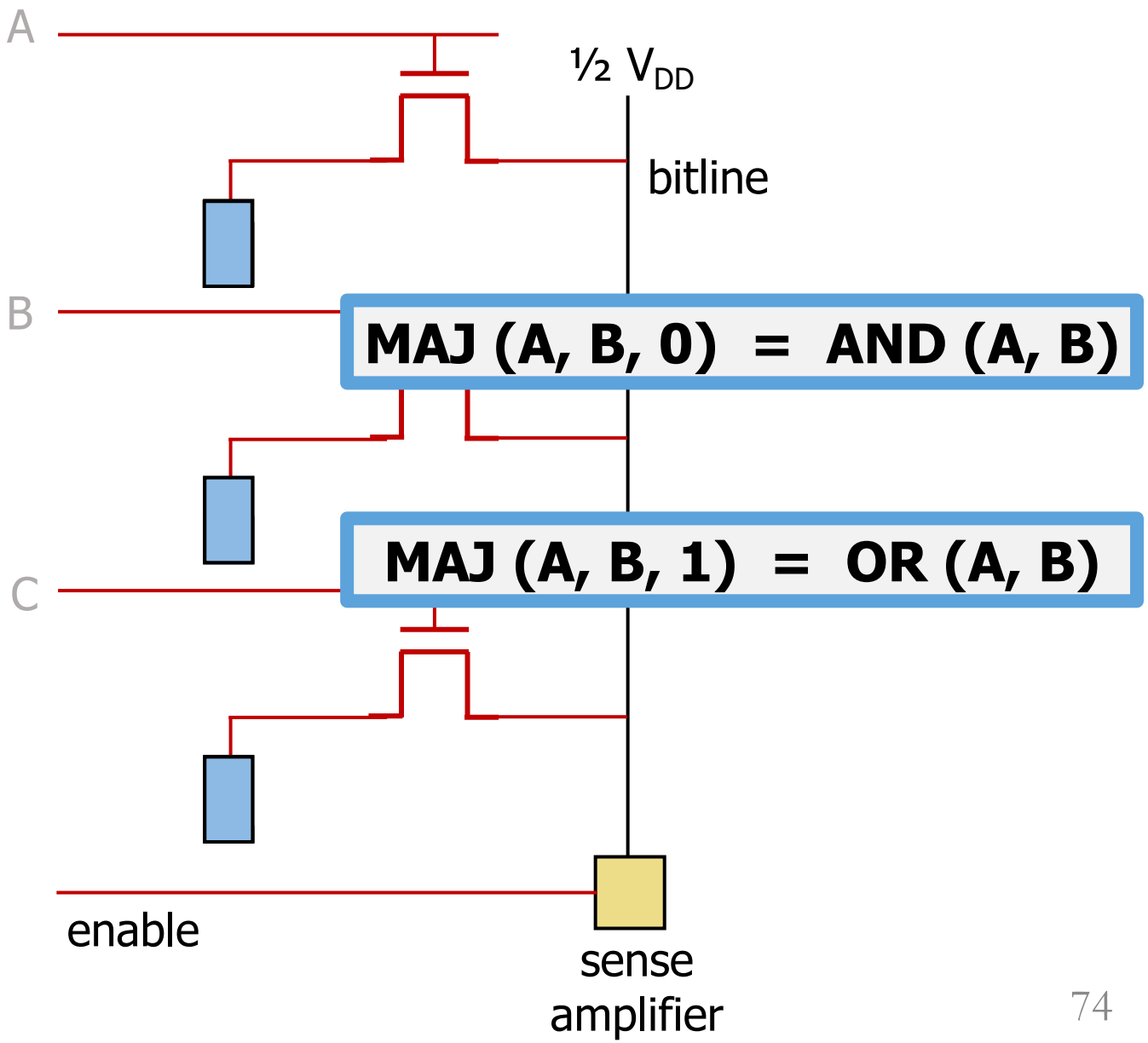
4. PRECHARGE bitline for next access



Majority function command sequence:

1. ACTIVATE (ACT)
2. PRECHARGE (PRE)

Ambit: In-DRAM Bulk Bitwise AND/OR



Outline

1 Programming a Real PIM Architecture

Overview of recently published works

2 System Support for PuM Architectures

Overview of recently published works

3 Accelerating Key Applications with PIM

System Support for PuM Architectures:
Overview of recently published works

1 Geraldo F. Oliveira, Ataberk Olgun, Giray Yaglikci, Nisa Bostanci, Juan Gómez-Luna, Saugata Ghose, and Onur Mutlu, "***MIMDRAM: An End-to-End Processing-using-DRAM System for Energy-Efficient and Programmer-Transparent MIMD Computing***," in HPCA, 2024.

2 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, 2023.

3 Geraldo F. Oliveira, Emanuele G. Esposito, Juan Gómez-Luna, and Onur Mutlu, "***PUMA: Efficient and Low-Cost Memory Allocation and Alignment Support for Processing-Using-Memory Architectures***," in MICRO SRC, 2023.

System Support for PuM Architectures:
Overview of recently published works

1 Geraldo F. Oliveira, Ataberk Olgun, Giray Yaglikci, Nisa Bostanci, Juan Gómez-Luna, Saugata Ghose, and Onur Mutlu, "***MIMDRAM: An End-to-End Processing-using-DRAM System for Energy-Efficient and Programmer-Transparent MIMD Computing***," in HPCA, 2024.

2 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, 2023.

3 Geraldo F. Oliveira, Emanuele G. Esposito, Juan Gómez-Luna, and Onur Mutlu, "***PUMA: Efficient and Low-Cost Memory Allocation and Alignment Support for Processing-Using-Memory Architectures***," in MICRO SRC, 2023.

2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)

MIMDRAM: An End-to-End Processing-using-DRAM System for Energy-Efficient and Programmer-Transparent MIMD Computing

Geraldo F. Oliveira† Ataberk Olgun† A. Giray Yaglikci† Nisa Bostanci†
Juan Gómez-Luna† Saugata Ghose‡ Onur Mutlu†

† ETH Zürich ‡ University of Illinois at Urbana-Champaign

Geraldo F. Oliveira, Ataberk Olgun, Giray Yaglikci, Nisa Bostanci, Juan Gómez-Luna, Saugata Ghose, and Onur Mutlu, "***MIMDRAM: An End-to-End Processing-using-DRAM System for Energy-Efficient and Programmer-Transparent MIMD Computing***," in HPCA, 2024.

Executive Summary

Problem: Processing-using-DRAM (PuD) suffers from three main issues caused by DRAM's large and rigid access granularity

- Under-utilization due to varying degrees of SIMD parallelism in an application
- Limited computation pattern due to a lack of interconnecting networks
- Challenging programming model due to a lack of compilers

Goal: Design a flexible PuD system that overcomes the three limitations caused by the large and rigid granularity of PuD

Key Mechanism: MIMDRAM, a hardware/software co-designed PuD

- *Key idea*: leverage fine-grained DRAM (i.e., the ability to access portions of a DRAM row)
- ***Hardware side***: (*i*) *latches and isolation transistors* to enable concurrent execution of PuD operations in a DRAM row; (*ii*) *interconnect networks* to enable PuD reduction
- ***Software side***: compiler passes to (*i*) *identify and generate* the PuD operations with the appropriate granularity; (*ii*) *schedule* the concurrent execution of PuD operations

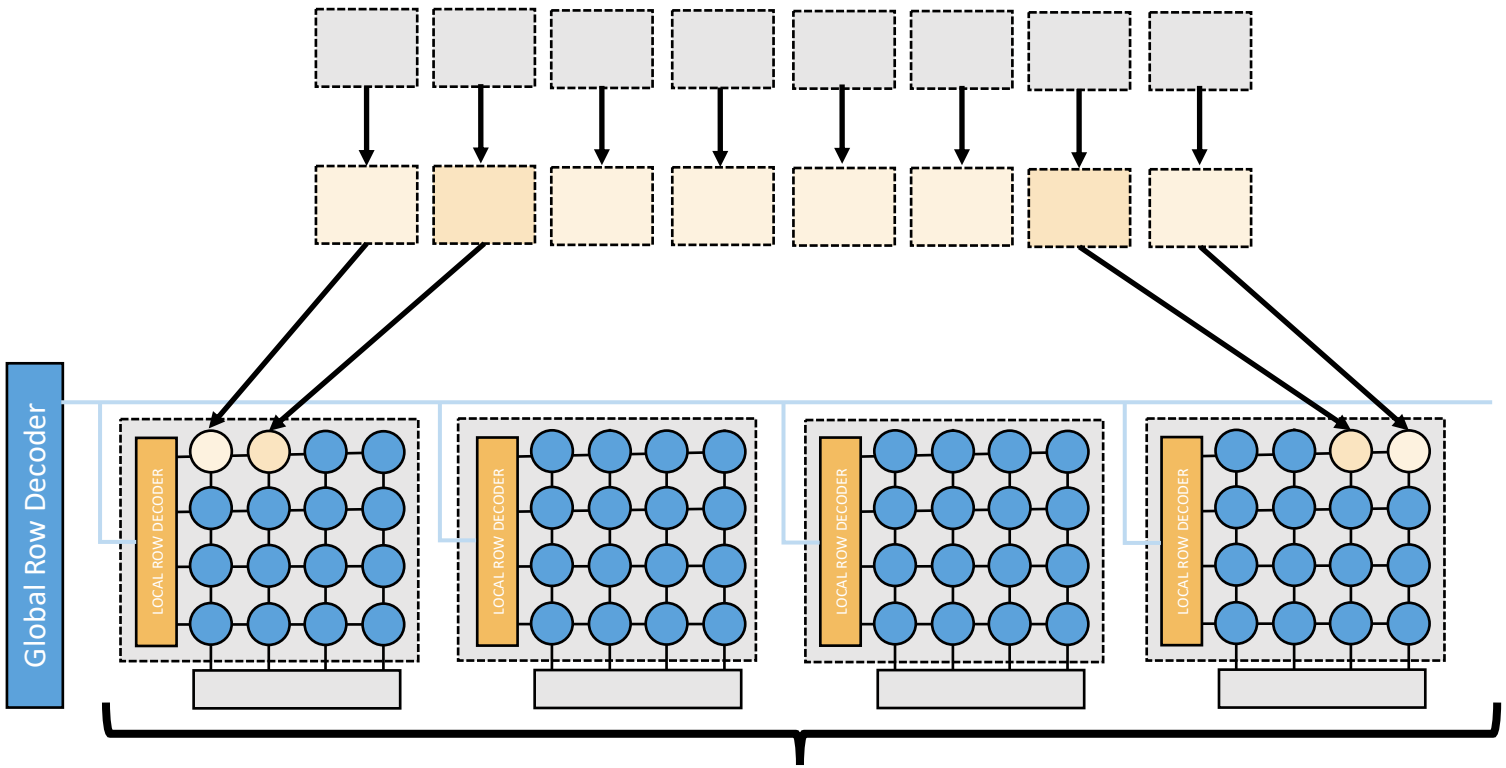
Key Results: MIMDRAM achieves

- 18.6x the utilization, 152x the energy efficiency, 1.7x the throughput, and 1.3x the fairness of a state-of-the-art PuD framework;
- 130x the energy efficiency of a high-end CPU

Problem & Goal:
Application Analysis

Application analysis: quantify the amount of SIMD parallelism real-world applications inherently display

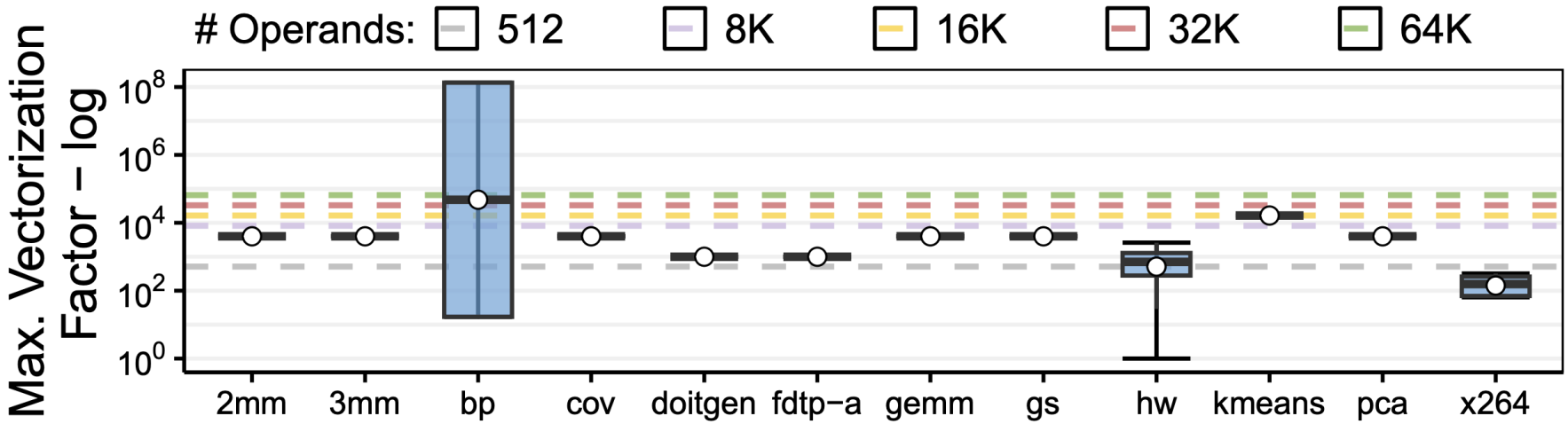
Maximum Vectorization Factor:
how many operands can be executed in parallel



Ideal maximum vectorization factor = # bitlines (e.g., 65'536)

Problem & Goal:
Application Analysis

Application analysis: quantify the amount of SIMD parallelism real-world applications inherently display

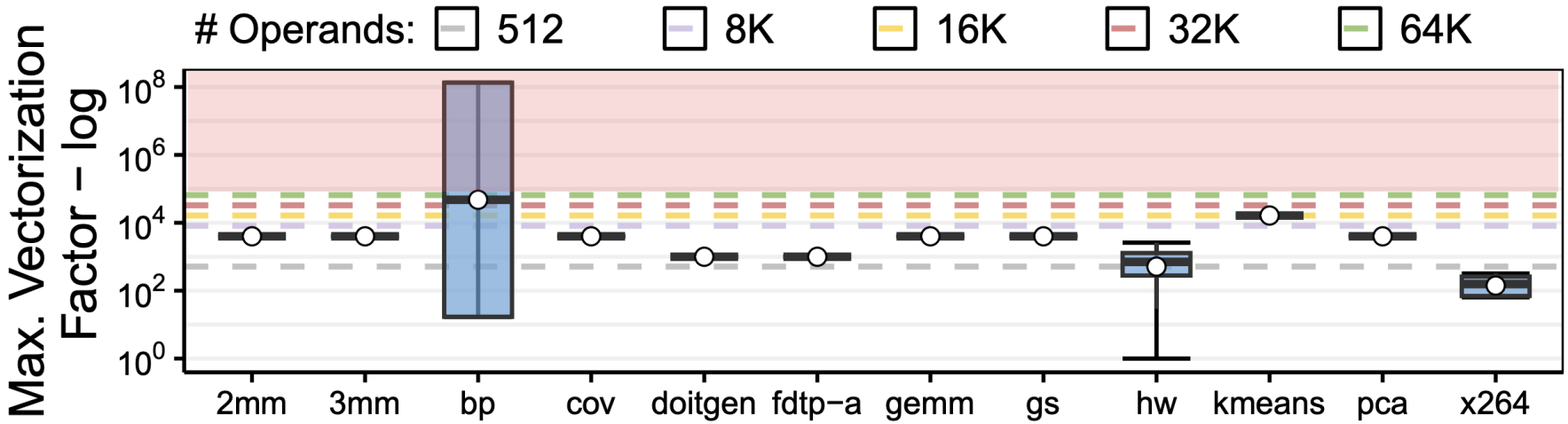


Takeaway

The maximum vectorization factor <u>varies</u> <u>within</u> a single application and <u>across</u> different applications

Problem & Goal:
Application Analysis

Application analysis: quantify the amount of SIMD parallelism real-world applications inherently display



Takeaway

A small amount of vectorized loops have a large enough maximum vectorization factor to fully exploit the SIMD parallelism of PuD

Problem & Goal

Problem

The rigid granularity of PuD architectures limits their applicability and efficiency for many applications.

The underlying PuD architecture often suffers from SIMD underutilization and consequentially energy and throughput waste

Goal

Design a Processing-using-DRAM architecture that:

1. adapts to the SIMD parallelism an application displays
2. maximizes the utilization of the PuD engine

MIMDRAM:
Overview

MIMDRAM is a hardware/software co-designed PuD system that enables fine-grained PuD computation at low cost and low programming effort

Main components in MIMDRAM

1 Hardware-side

- subarray modification to enable MIMD-like fine-grained DRAM computation
- inter- and intra-mat network to enable PuD vector reduction

2 Software-side

- new compiler support to transparently generate PuD instructions
- system support to to enable the orchestration of PuD instructions

MIMDRAM:
Overview

MIMDRAM is a hardware/software co-designed PuD system that enables fine-grained PuD computation at low cost and low programming effort

Main components in MIMDRAM

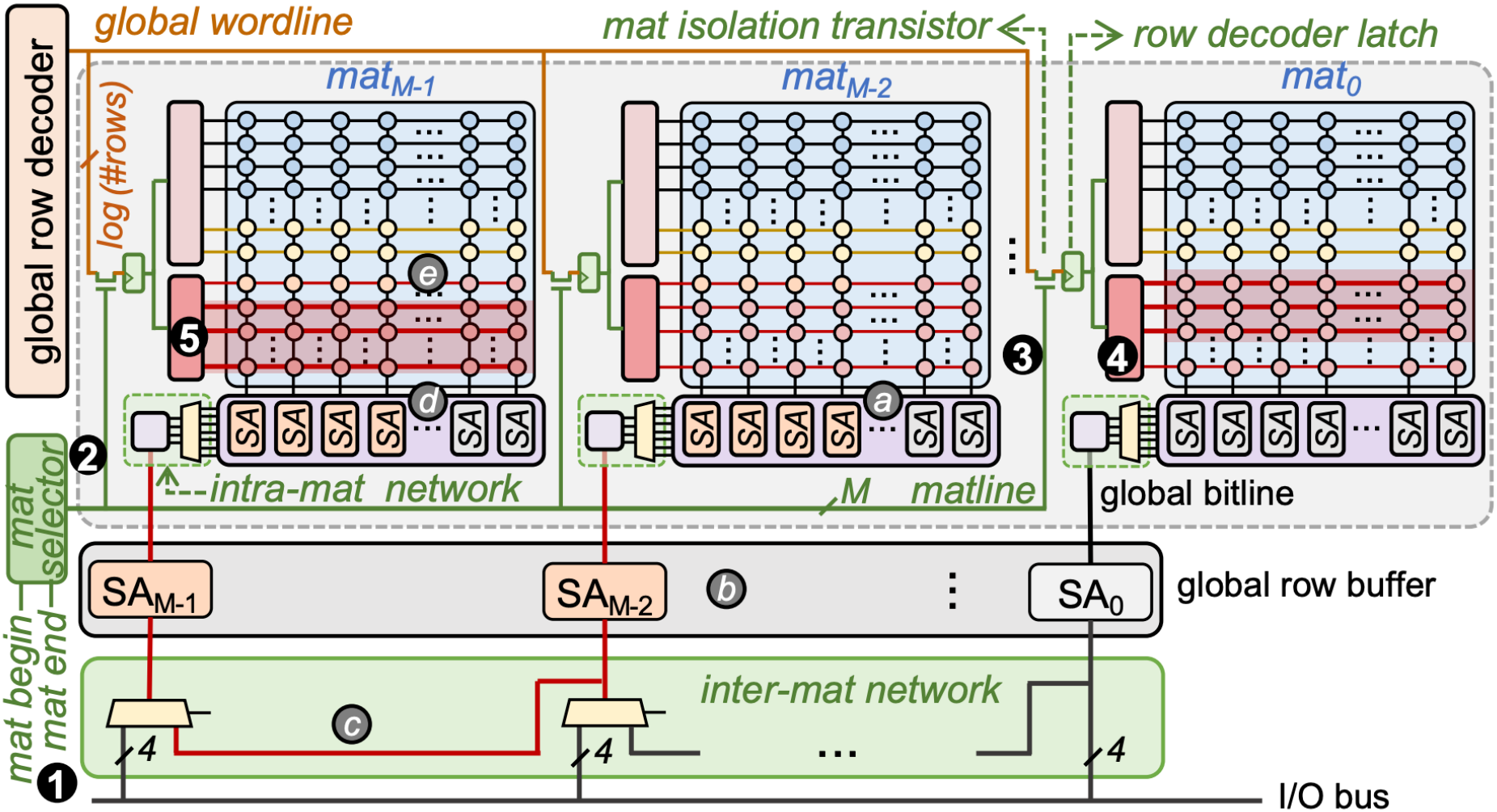
1 Hardware-side

- subarray modification to enable MIMD-like fine-grained DRAM computation
- inter- and intra-mat network to enable PuD vector reduction

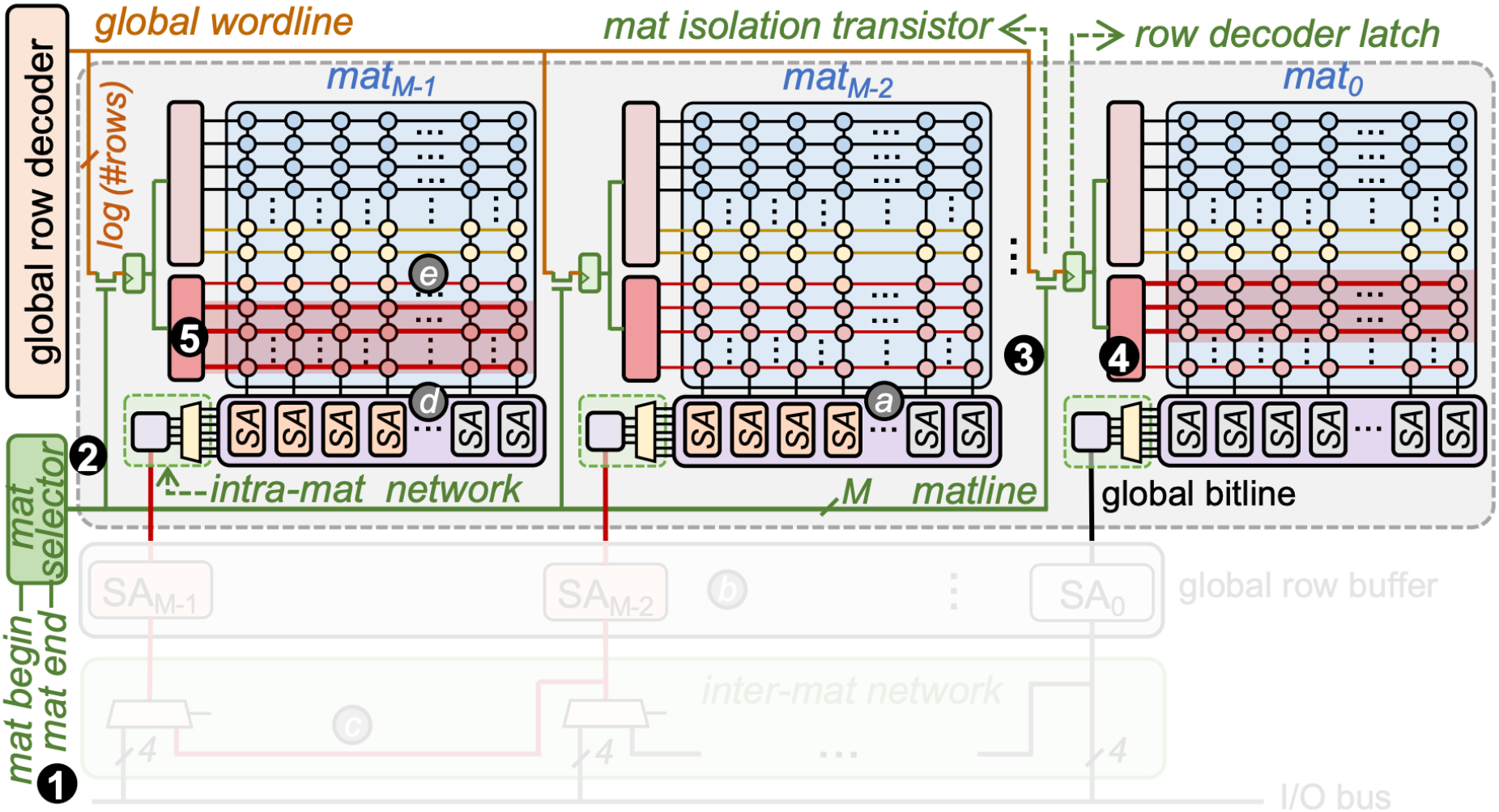
2 Software-side

- new compiler support to transparently generate PuD instructions
- system support to to enable the orchestration of PuD instructions

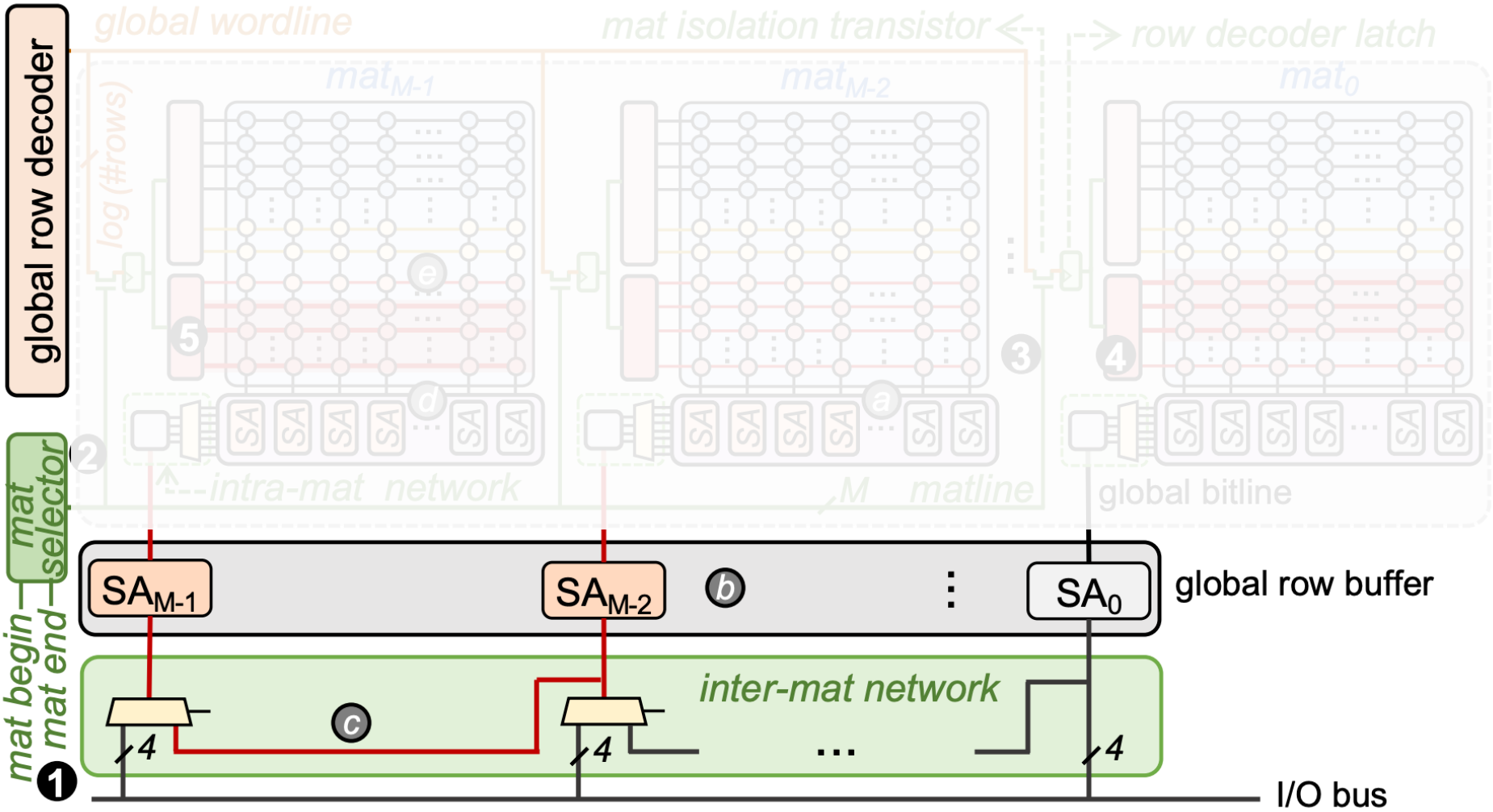
MIMDRAM:
DRAM Hardware Overview



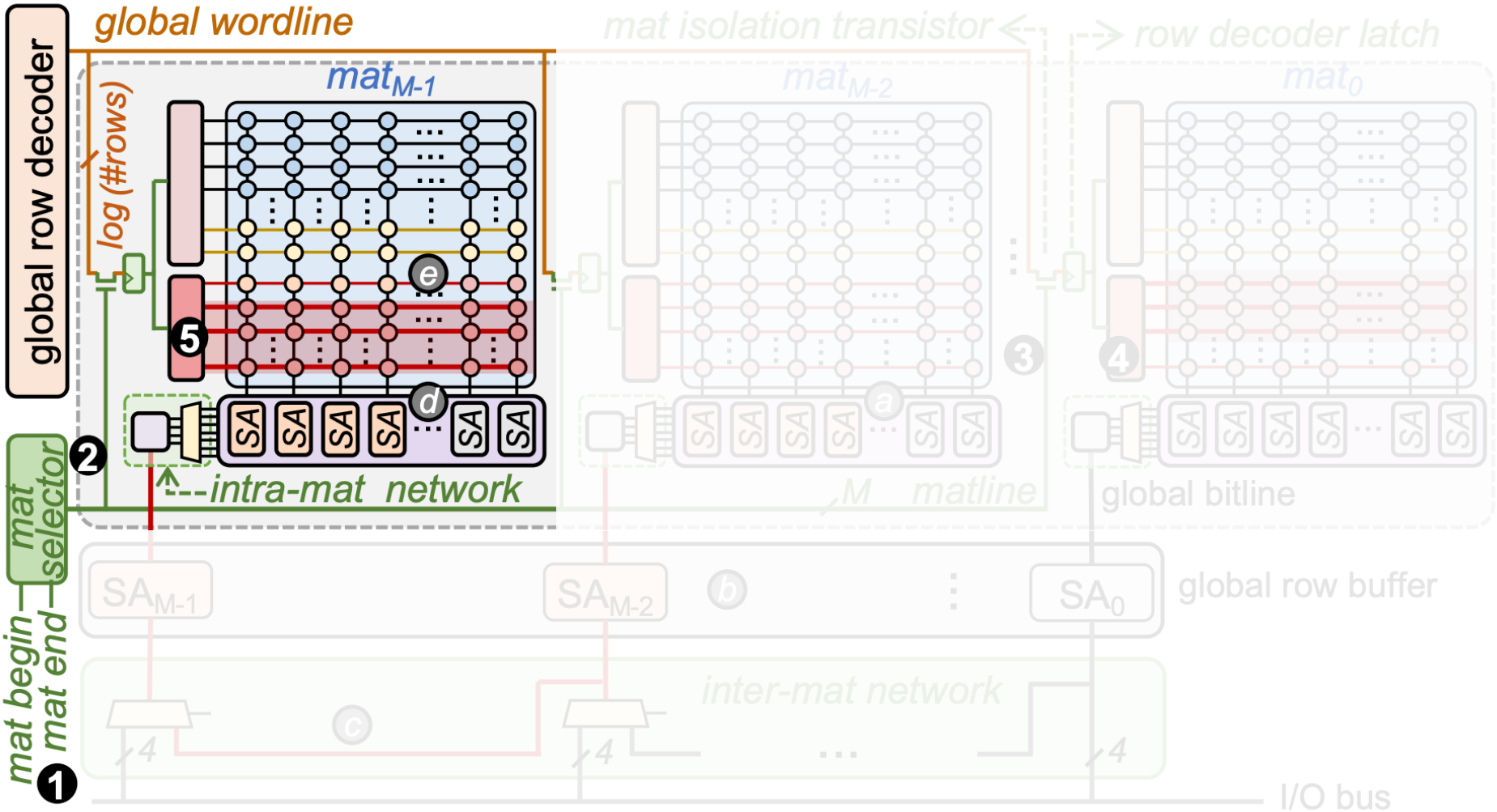
MIMDRAM:
DRAM Hardware Overview



MIMDRAM:
DRAM Hardware Overview

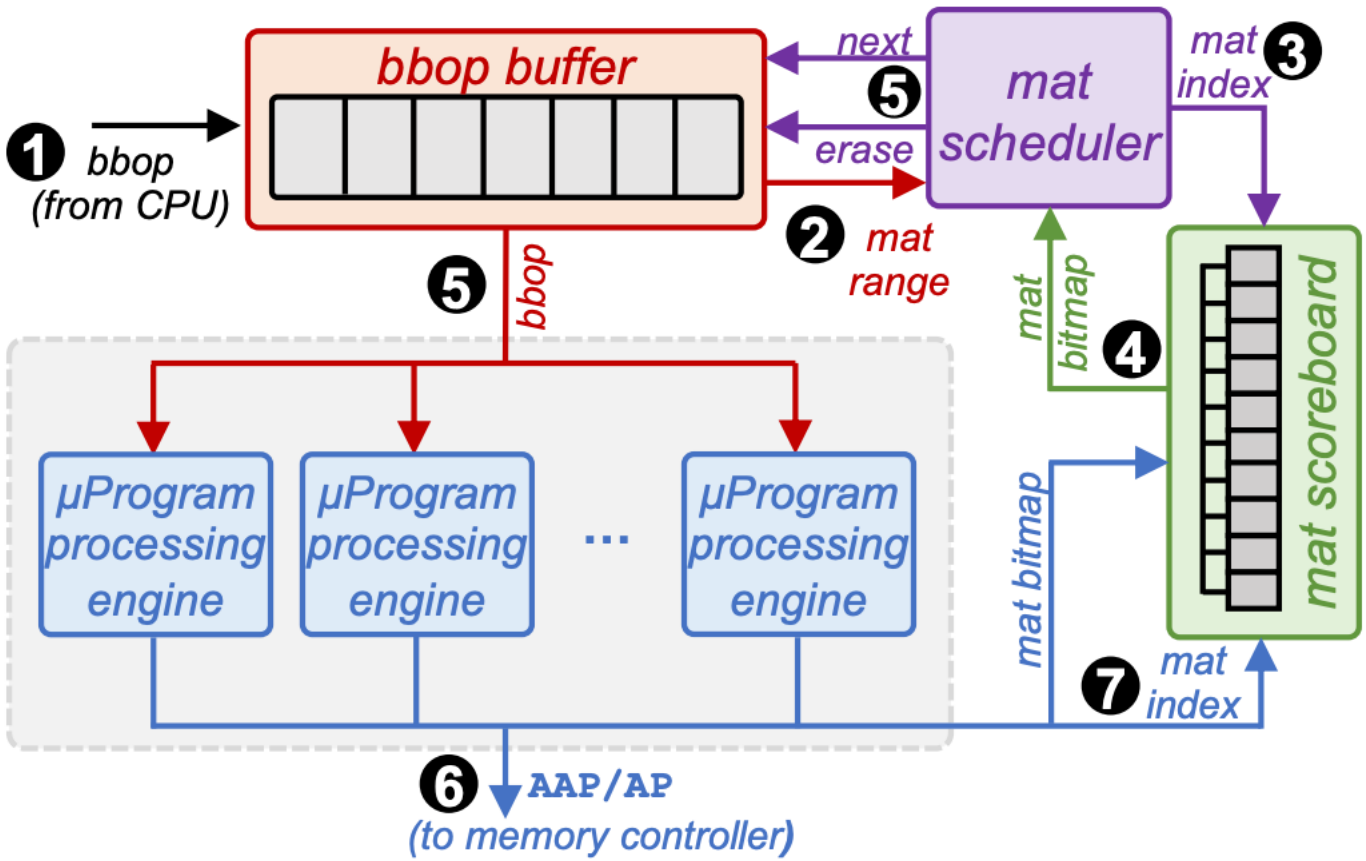


MIMDRAM:
DRAM Hardware Overview



MIMDRAM:
Control Unit

Goal: schedule and orchestrate the execution of multiple PuD instructions transparently



SAFAR.

MIMDRAM:
Overview

MIMDRAM is a hardware/software co-designed PuD system that enables fine-grained PuD computation at low cost and low programming effort

Main components in MIMDRAM

1 Hardware-side

- subarray modification to enable MIMD-like fine-grained DRAM computation
- inter- and intra-mat network to enable PuD vector reduction

2 Software-side

- new compiler support to transparently generate PuD instructions
- system support to to enable the orchestration of PuD instructions

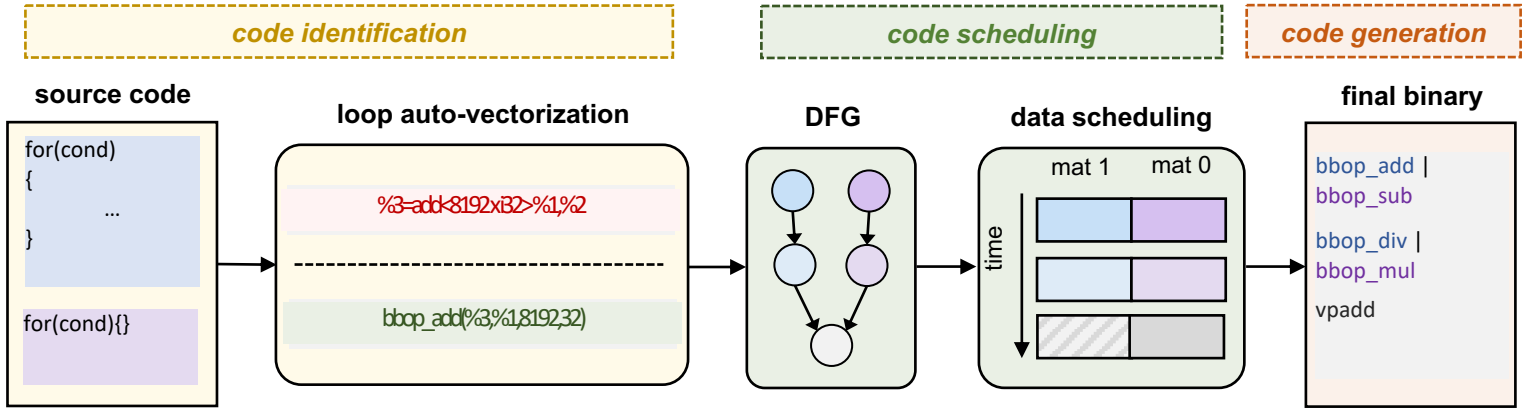
MIMDRAM:

Software Overview

Goal

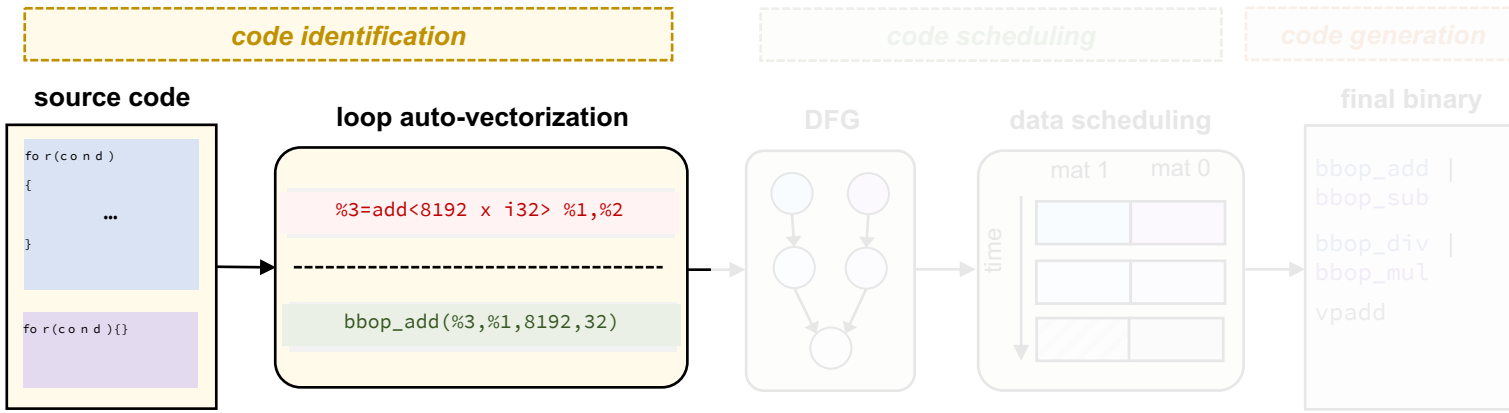
Transparently:
(1) extract SIMD parallelism from an application,
(2) schedule PuD operations while maximizing MAT utilization

Three new LLVM-based passes targeting PuD execution



MIMDRAM:

Software Overview



Goal

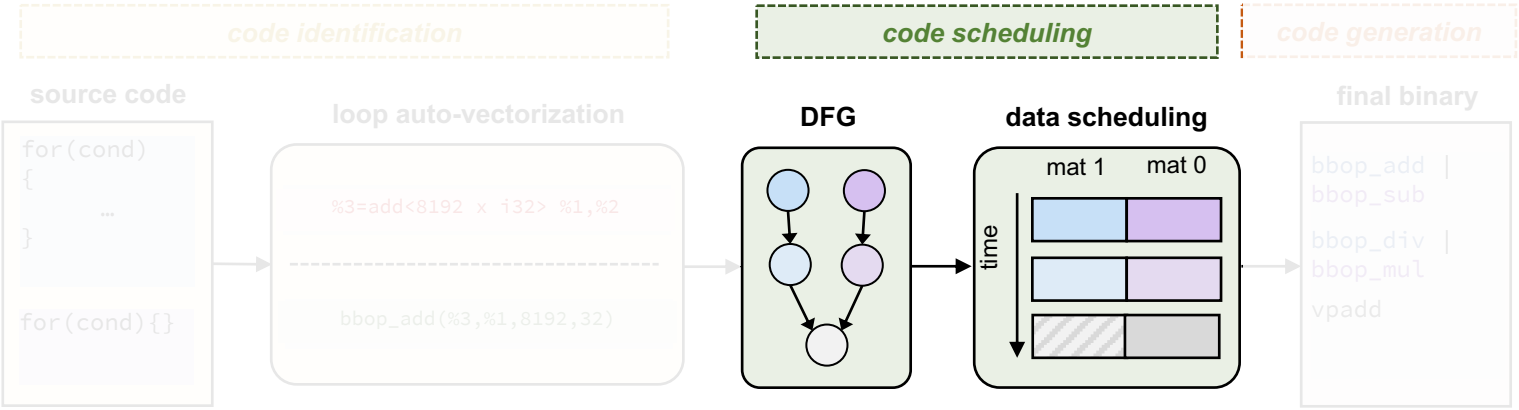
Identify SIMD parallelism and generate appropriate PuD instructions with best vectorization factor

Changes to LLVM's auto-vectorization pass:

- Selection of **the best-performing vectorization factor** for a given loop → always select as vectorization factor the maximum vectorization factor
- **Code generation routine** for a given vectorized loop → identify and remove memory instructions related to an arithmetic SIMD operation

MIMDRAM:

Software Overview



Goal

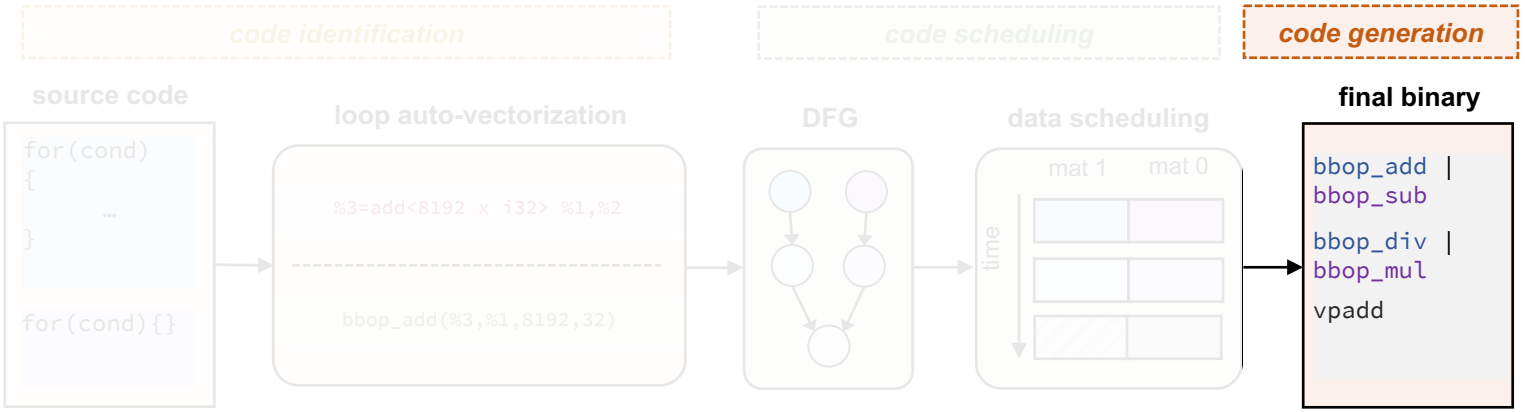
Provide load balance across MATs and minimal inter-MAT data movement

Key Idea: use a new **malloc operation** that informs the OS about MAT allocation requirement

New code scheduling algorithm to schedule computation across MATs

MIMDRAM:

Software Overview



Goal

Generate the appropriate code for data allocation and PuD execution

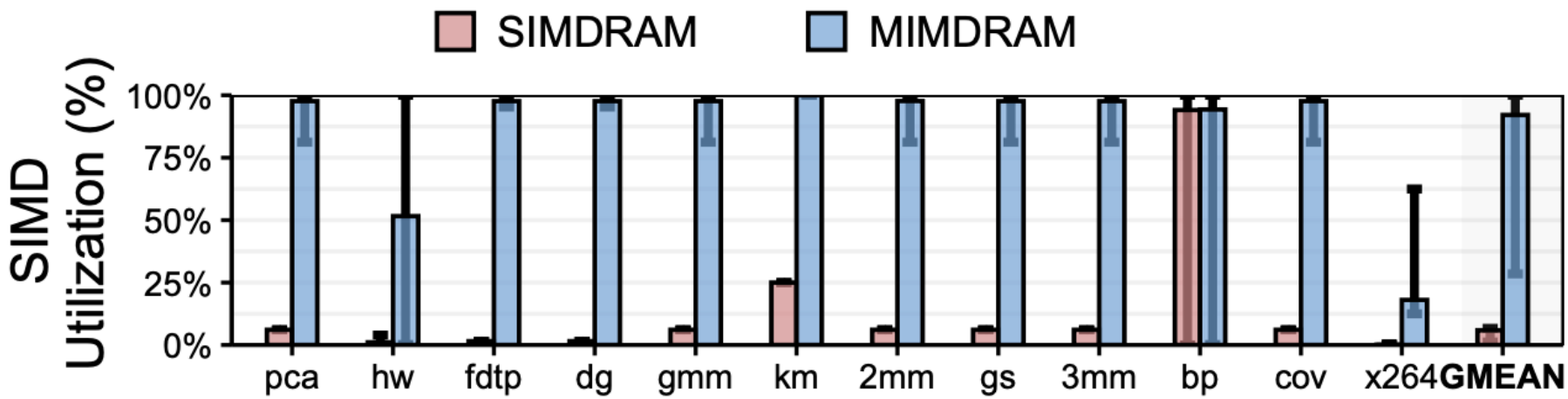
Evaluation:
Methodology

- We implement MIMDRAM using the gem5 simulator

- **Comparison points:**
  - real multicore CPU (Intel Skylake)
  - state-of-the-art PuD framework (SIMDRAM)

- **Workloads:**
  - 12 applications from various benchmark suites

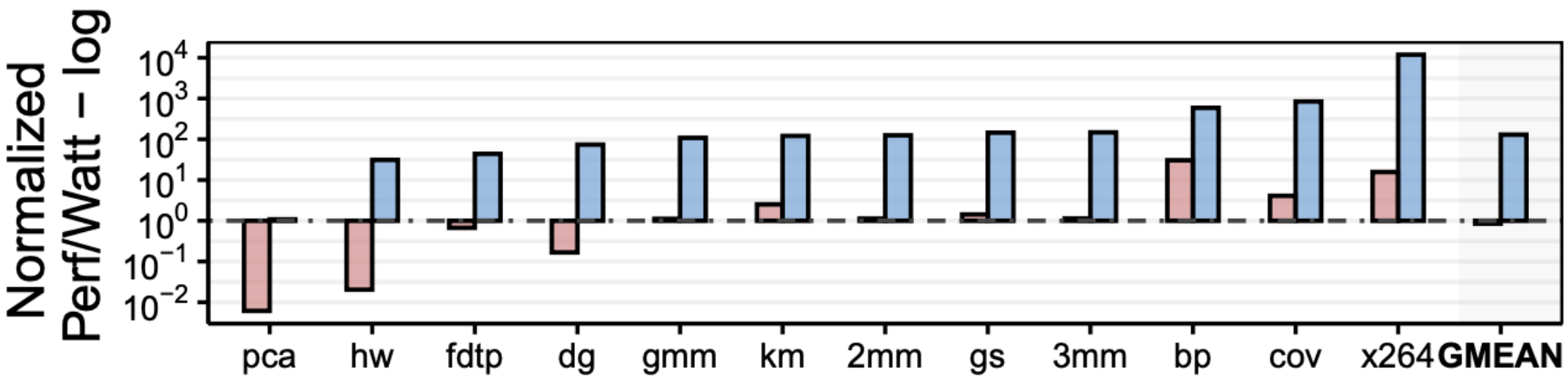
Evaluation:
Single Application Analysis – SIMD Utilization



Takeaway

MIMDRAM significantly improves SIMD utilization compared with SIMDRAM by 15.6x, on average

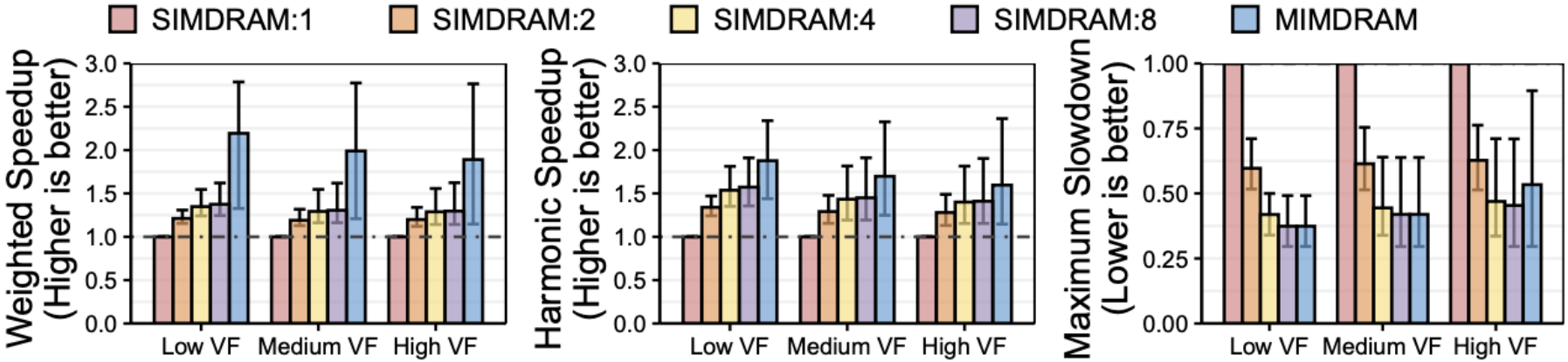
Evaluation:
Single Application Analysis – Energy Efficiency



Takeaway

MIMDRAM significantly improves energy efficiency compared with SIMDRAM (by 152x) and to the CPU (by 130x)

Evaluation:
Multi Application Analysis



Takeaway

MIMDRAM significantly improves system throughput (1.68x), job turnaround time (1.33x) , and fairness (1.32x) compared with SIMDRAM

Conclusion

Problem: Processing-using-DRAM (PuD) suffers from three main issues caused by DRAM's large and rigid access granularity

- Under-utilization due to varying degrees of SIMD parallelism in an application
- Limited computation pattern due to a lack of interconnecting networks
- Challenging programming model due to a lack of compilers

Goal: Design a flexible PuD system that overcomes the three limitations caused by the large and rigid granularity of PuD

Key Mechanism: MIMDRAM, a hardware/software co-designed PuD

- *Key idea*: leverage fine-grained DRAM (i.e., the ability to access portions of a DRAM row)
- ***Hardware side***: (*i*) *latches and isolation transistors* to enable concurrent execution of PuD operations in a DRAM row; (*ii*) *interconnect networks* to enable PuD reduction
- ***Software side***: compiler passes to (*i*) *identify and generate* the PuD operations with the appropriate granularity; (*ii*) *schedule* the concurrent execution of PuD operations

Key Results: MIMDRAM achieves

- 18.6x the utilization, 152x the energy efficiency, 1.7x the throughput, and 1.3x the fairness of a state-of-the-art PuD framework;
- 130x the energy efficiency of a high-end CPU

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

Summary: Flash-Cosmos

The first work that enables
in-flash multi-operand bulk bitwise operations
with a single sensing operation and high reliability

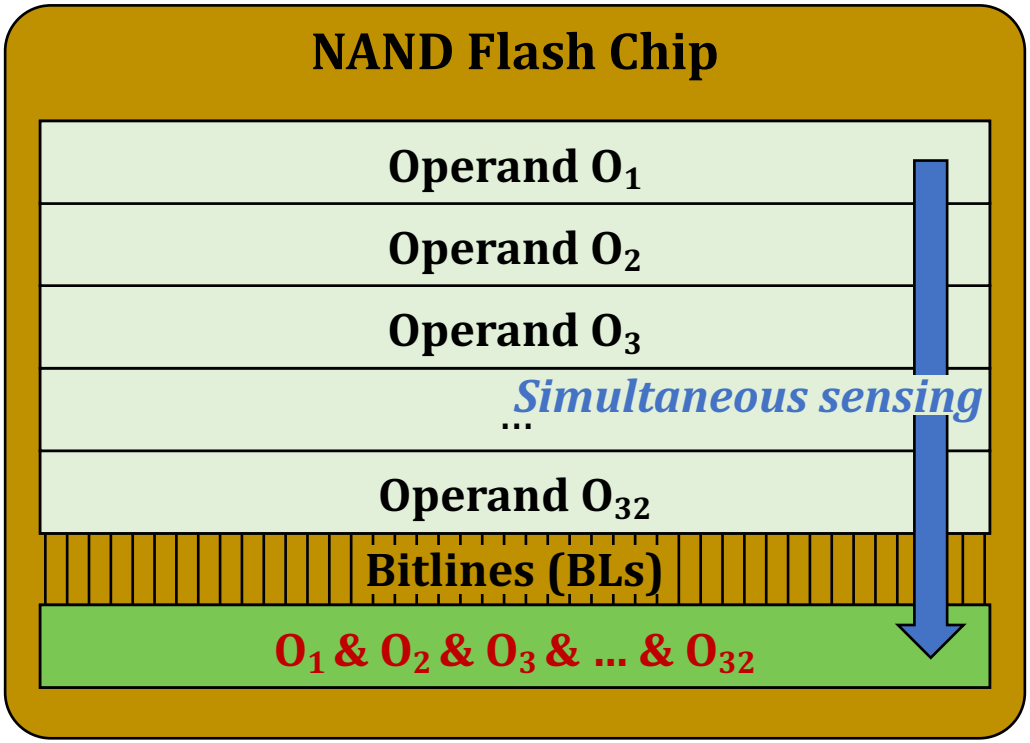
Improves performance
by 32x/25x/3.5x over OSP/ISP/ParaBit

Improves energy efficiency
by 95x/13.4x/3.3x over OSP/ISP/ParaBit

Low-cost & requires no changes to flash cell arrays

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



Next Steps for PuM

- **Executing processing-using-DRAM operations in real, off-the-shelf DRAM chips**
  - We experimentally demonstrate that off-the-shelf DRAM chips are capable of performing
    - 1) NOT, NAND, and NOR operations
    - 2) AND and OR operations with more than 2 inputs
  - We present an extensive characterization of new bulk bitwise operations in **224 off-the-shelf modern DDR4 DRAM chips**

- **System support for processing-using-DRAM**

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

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

ETH Zurich

Outline

1 Programming a Real PIM Architecture

Overview of recently published works

2 System Support for PuM Architectures

Overview of recently published works

3 Accelerating Key Applications with PIM

Casper, IEEE ACCESS 2023

RESEARCH ARTICLE

Casper: Accelerating Stencil Computations Using Near-Cache Processing

ALAIN DENZLER, GERALDO F. OLIVEIRA, NASTARAN HAJINAZAR, RAHUL BERA, GAGANDEEP SINGH, JUAN GÓMEZ-LUNA, (Member, IEEE), AND ONUR MUTLU, (Fellow, IEEE)

Department of Information Technology and Electrical Engineering (D-ITET), ETH Zürich, 8092 Zürich, Switzerland

Corresponding author: Juan Gómez-Luna (juang@ethz.ch)

ABSTRACT Stencil computations are commonly used in a wide variety of scientific applications, ranging from large-scale weather prediction to solving partial differential equations. Stencil computations are characterized by three properties: 1) low arithmetic intensity, 2) limited temporal data reuse, and 3) regular and predictable data access pattern. As a result, stencil computations are typically bandwidth-bound workloads, which experience only limited benefits from the deep cache hierarchy of modern CPUs. In this work, we propose Casper, a near-cache accelerator consisting of specialized stencil computation units connected to the last-level cache (LLC) of a traditional CPU. Casper is based on two key ideas: 1) avoiding the cost of moving rarely reused data throughout the cache hierarchy, and 2) exploiting the regularity of the data accesses and the inherent parallelism of stencil computations to increase overall performance. With small changes in LLC address decoding logic and data placement, Casper performs stencil computations at the peak LLC bandwidth. We show that by tightly coupling lightweight stencil computation units near LLC, Casper improves performance of stencil kernels by 1.65× on average (up to 4.16×) compared to a commercial high-performance multi-core processor, while reducing system energy consumption by 35% on average (up to 65%). Casper provides 37× (up to 190×) improvement in performance-per-area compared to a state-of-the-art GPU.

https://arxiv.org/pdf/2112.14216.pdf

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 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*

SAFARI **https://arxiv.org/pdf/2204.11275.pdf**

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

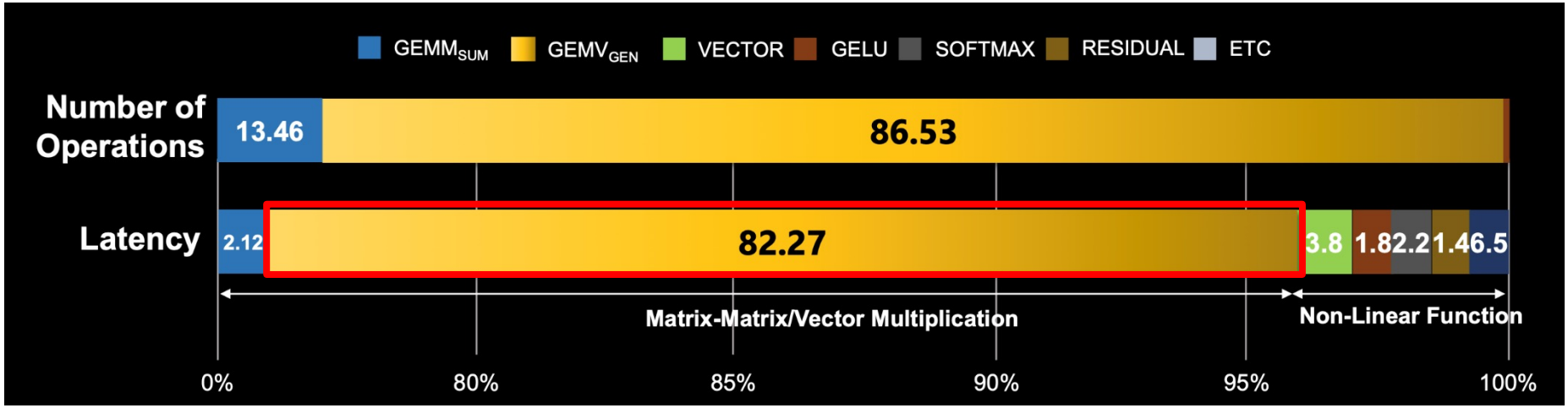
Amirali Boroumand[†◇] Saugata Ghose[‡] Berkin Akin[§] Ravi Narayanaswami[§]
Geraldo F. Oliveira[⋆] Xiaoyu Ma[§] Eric Shiu[§] Onur Mutlu[⋆†]

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

Samsung PNM Solutions for Generative AI (2023)

- Main target: transformer decoders used in ChatGPT, GPT-3
  - Compute-bound step: Summarization
  - Memory-bound step: Generation
    - Most of the execution time is spent on the memory copy from the host CPU memory to the CPU memory

- GEMV portion can be 60%-80% of total generation latency, which is the target of PIM/PNM



From: J. H. Kim, "Samsung AI-cluster system with HBM-PIM and CXL-based Processing-near-Memory for transformer-based LLMs," HC, 2023.

In-Storage Genome 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.
  [Talk Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Lightning Talk Video (90 seconds)]
  [Talk Video (17 minutes)]

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

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

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

SAFARI

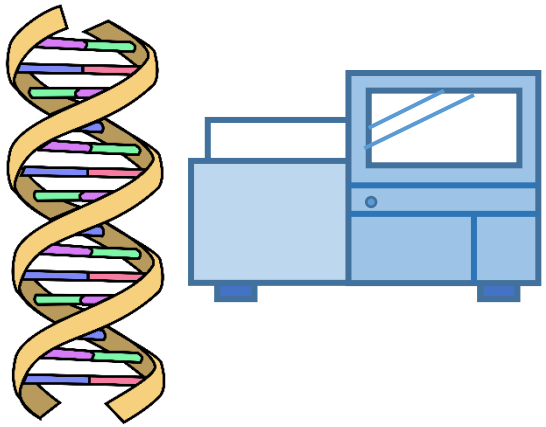
ETH zürich

UNIVERSITY OF TORONTO

Genome Sequence Analysis

- Genome sequence analysis is critical for many applications
  - Personalized medicine
  - Outbreak tracing
  - Evolutionary studies

- Genome sequencing machines extract smaller fragments of the original DNA sequence, known as reads

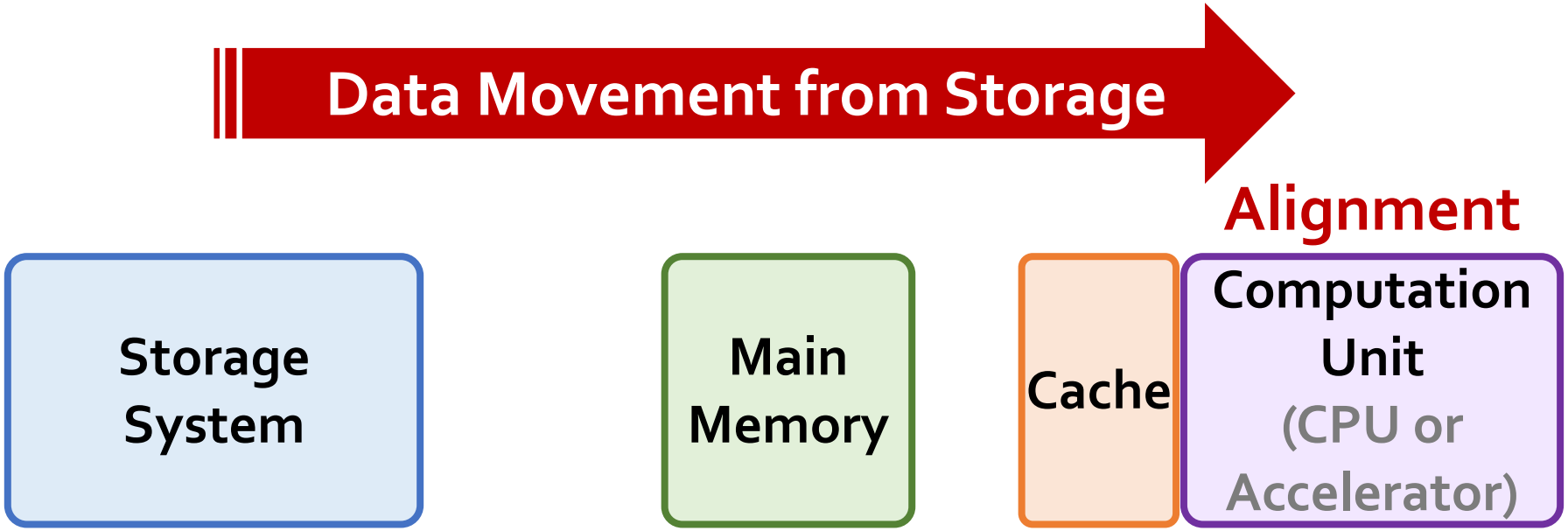


Genome Sequence Analysis

- **Read mapping:** first key step in genome sequence analysis
  - Aligns reads to potential matching locations in the reference genome
  - For each matching location, the alignment step finds the degree of similarity (alignment score)

- Calculating the alignment score requires computationally-expensive approximate string matching (ASM) to account for differences between reads and the reference genome due to:
  - Sequencing errors
  - Genetic variation

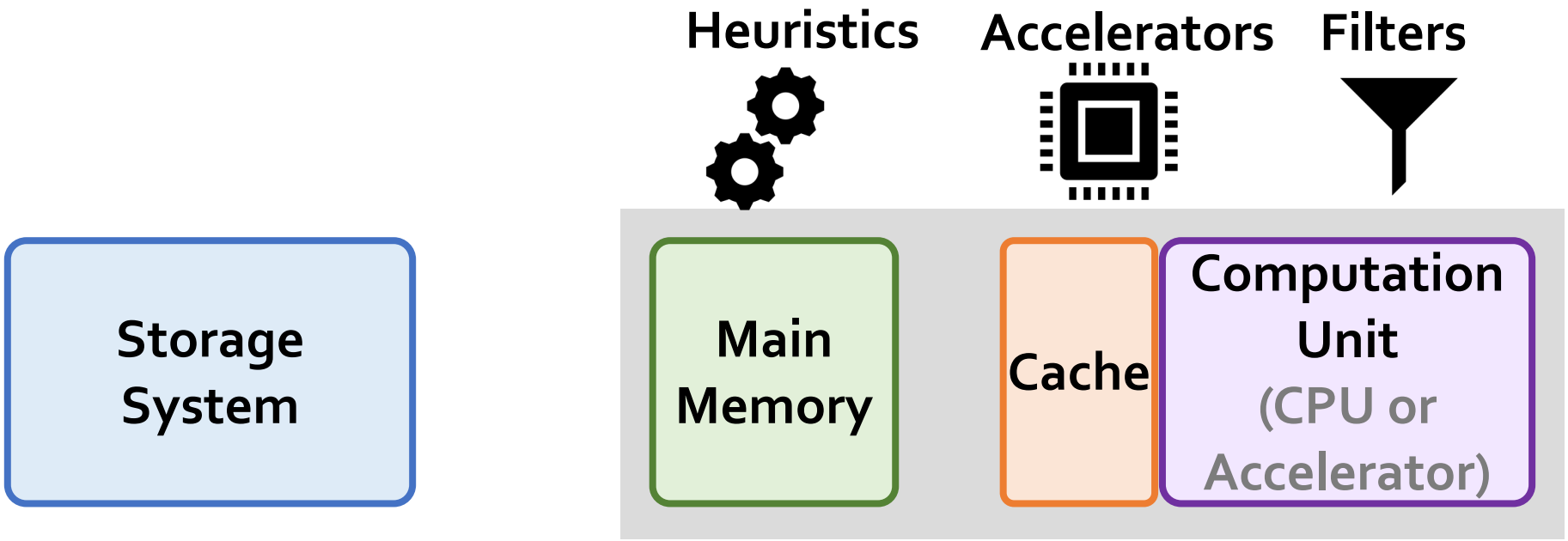
Genome Sequence Analysis



✗ Computation overhead

✗ Data movement overhead

Accelerating Genome Sequence Analysis

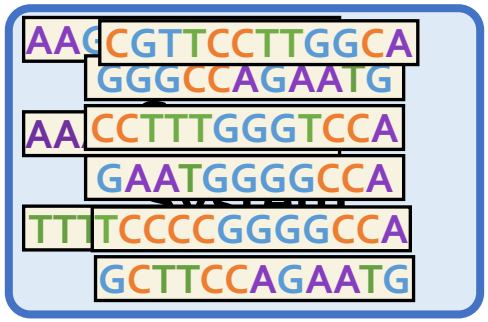


✓ Computation overhead

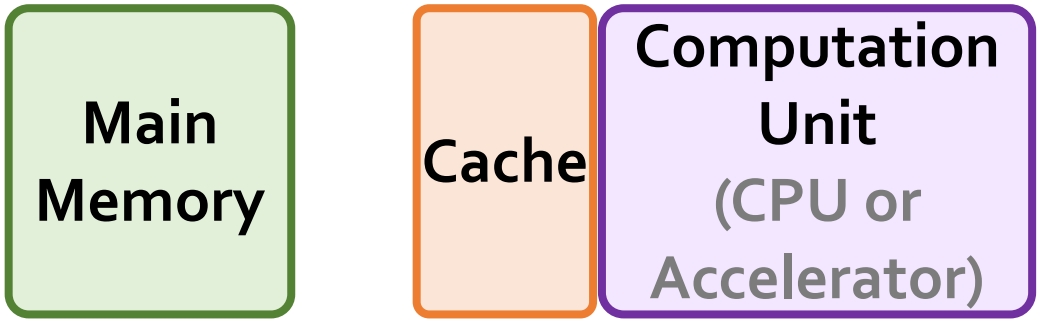
✗ Data movement overhead

Key Idea

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



Filtered Reads



Exactly-matching reads

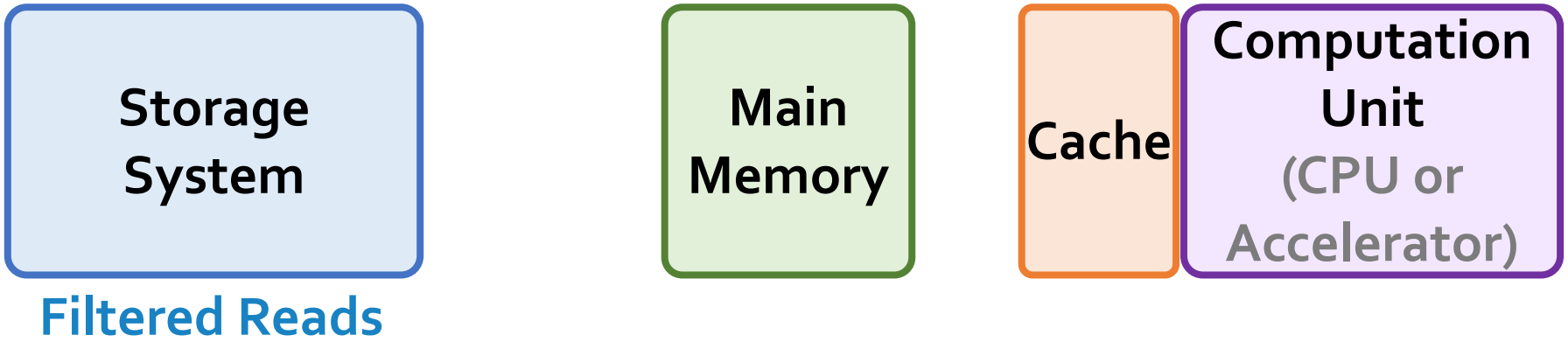
Do not need expensive approximate string matching during alignment

Non-matching reads

Do not have potential matching locations and can skip alignment

Challenges

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



Read mapping workloads can exhibit different behavior

There are limited hardware resources in the storage system

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

Concluding Remarks

Concluding Remarks

- We must design systems to be **balanced**, **high-performance**, **energy-efficient** (all at the same time) → intelligent systems
  - **Data-centric, data-driven, data-aware**
- Enable computation capability inside and close to memory/storage
- This can
  - Lead to **orders-of-magnitude** improvements
  - **Enable new applications & computing platforms**
  - **Enable better understanding of nature**
  - **…**
- Future of **truly data-centric computing** is bright
  - We need to do research & design across the computing stack

Fundamentally Better Architectures

Data-centric

Data-driven

Data-aware

Self-Optimizing Memory Prefetchers

- Rahul Bera, Konstantinos Kanellopoulos, Anant Nori, Taha Shahroodi, Sreenivas Subramoney, and Onur Mutlu,
  "Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning"
  Proceedings of the 54th International Symposium on Microarchitecture (***MICRO***), Virtual, October 2021.
  [Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (20 minutes)]
  [Lightning Talk Video (1.5 minutes)]
  [Pythia Source Code (Officially Artifact Evaluated with All Badges)]
  [arXiv version]

Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

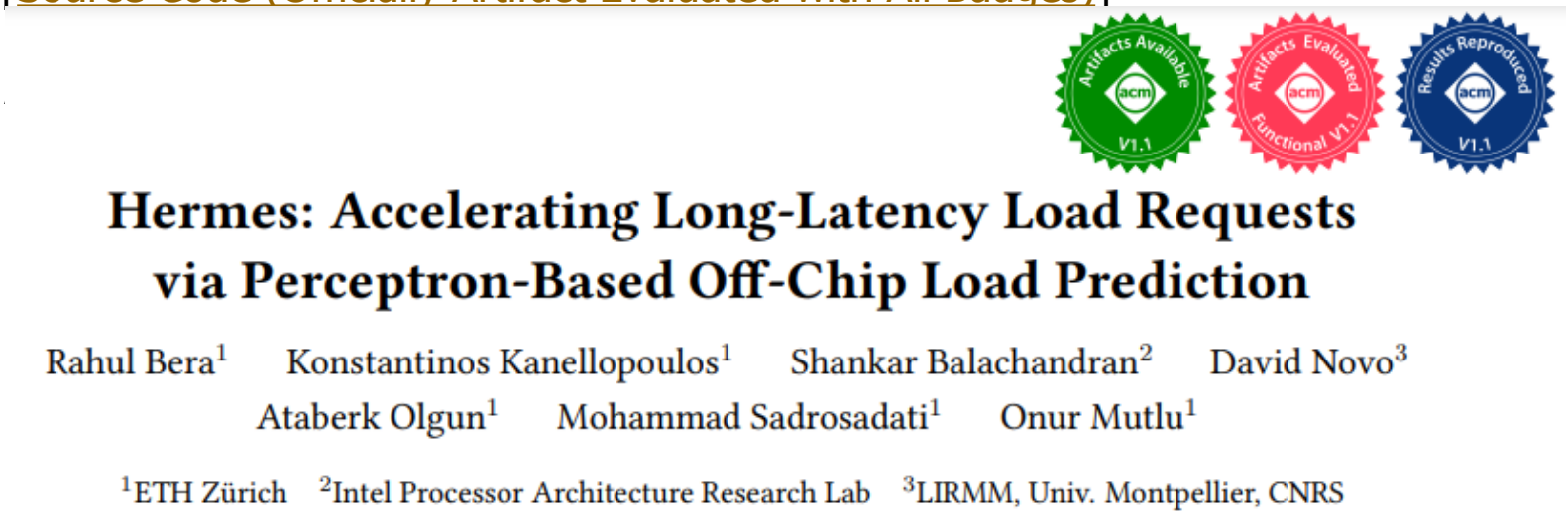
Rahul Bera[1] Konstantinos Kanellopoulos[1] Anant V. Nori[2] Taha Shahroodi[3,1]
Sreenivas Subramoney[2] Onur Mutlu[1]

[1]ETH Zürich [2]Processor Architecture Research Labs, Intel Labs [3]TU Delft

https://arxiv.org/pdf/2109.12021.pdf

Perceptron-Based Off-Chip Load Prediction

- Rahul Bera, Konstantinos Kanellopoulos, Shankar Balachandran, David Novo, Ataberk Olgun, Mohammad Sadrosadati, and Onur Mutlu,
  "Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction"
  Proceedings of the 55th International Symposium on Microarchitecture (***MICRO***), Chicago, IL, USA, October 2022.
  [Slides (pptx) (pdf)]
  [Longer Lecture Slides (pptx) (pdf)]
  [Talk Video (12 minutes)]
  [Lecture Video (25 minutes)]
  [arXiv version]
  [Source Code (Officially Artifact Evaluated with All Badges)]



https://arxiv.org/pdf/2209.00188.pdf

Self-Optimizing Hybrid Storage Systems

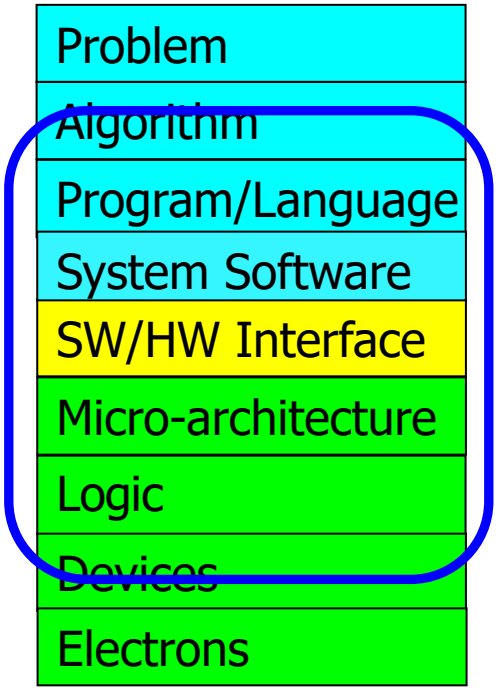
- Gagandeep Singh, Rakesh Nadig, Jisung Park, Rahul Bera, Nastaran Hajinazar, David Novo, Juan Gomez-Luna, Sander Stuijk, Henk Corporaal, and Onur Mutlu, **"Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning"**
  *Proceedings of the 49th International Symposium on Computer Architecture (**ISCA**)*, New York, June 2022.
  [Slides (pptx) (pdf)]
  [arXiv version]
  [Sibyl Source Code]
  [Talk Video (16 minutes)]

Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning

Gagandeep Singh[1] Rakesh Nadig[1] Jisung Park[1] Rahul Bera[1] Nastaran Hajinazar[1]
David Novo[3] Juan Gómez-Luna[1] Sander Stuijk[2] Henk Corporaal[2] Onur Mutlu[1]

[1]ETH Zürich [2]Eindhoven University of Technology [3]LIRMM, Univ. Montpellier, CNRS

We Need to Revisit the Entire Stack



We can get there step by step

We Need to Exploit Good Principles

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

Open minds

A Blueprint for Fundamentally Better Architectures

- Onur Mutlu,
  "Intelligent Architectures for Intelligent Computing Systems"
  *Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (**DATE**)*, Virtual, February 2021.
  [Slides (pptx) (pdf)]
  [IEDM Tutorial Slides (pptx) (pdf)]
  [Short DATE Talk Video (11 minutes)]
  [Longer IEDM Tutorial Video (1 hr 51 minutes)]

Intelligent Architectures for Intelligent Computing Systems

Onur Mutlu
ETH Zurich
omutlu@gmail.com

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!

Acknowledgments

SAFARI

SAFARI Research Group

safari.ethz.ch

Think BIG, Aim HIGH!

https://safari.ethz.ch

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/

Memory-Centric Computing for Data-Intensive Workloads

Dr. Mohammad Sadrosadati
Prof. Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu

11 December 2023

EFCL Mini-Conference

SAFARI

ETH zürich