

Storage-Centric Computing

for Modern Data-Intensive Workloads

Onur Mutlu

omutlu@gmail.com

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

23 May 2024

Huawei

SAFARI

ETH zürich

Computing

is Bottlenecked by Data

Data is Key for AI, ML, Genomics, ...

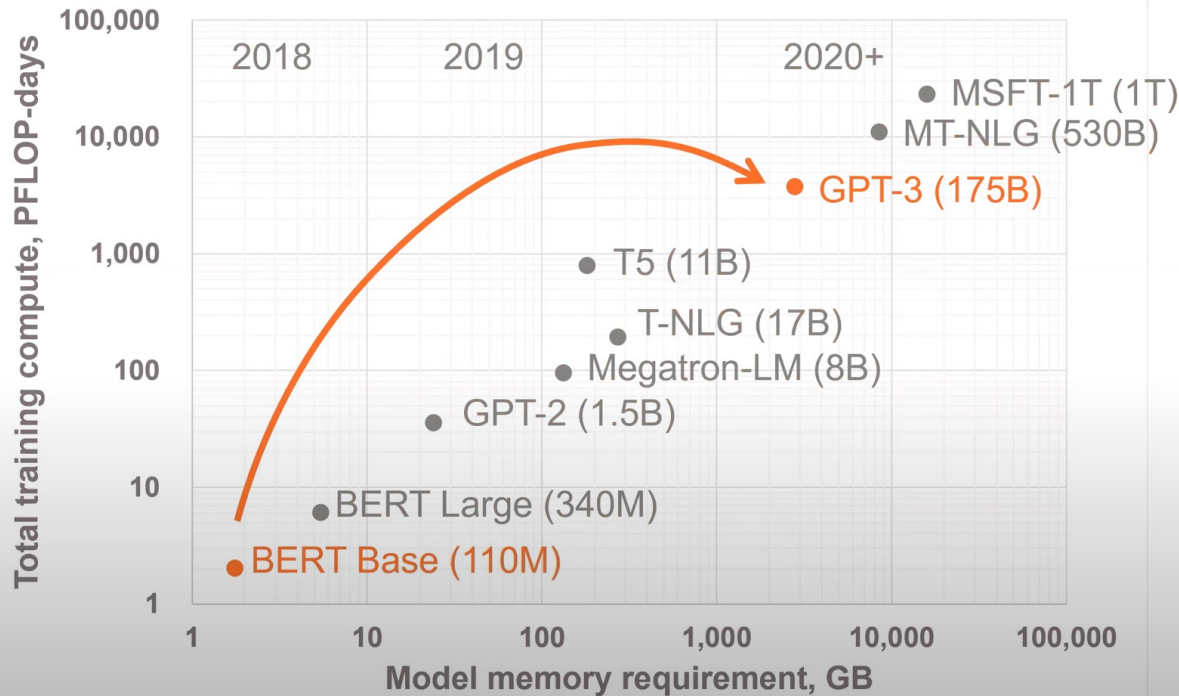
- Important workloads are all data intensive
- They require rapid and efficient processing of large amounts of data
- Data is increasing
 - We can generate more than we can process
 - We need to perform more sophisticated analyses on more data

Huge Demand for Performance & Efficiency

Exponential Growth of Neural Networks



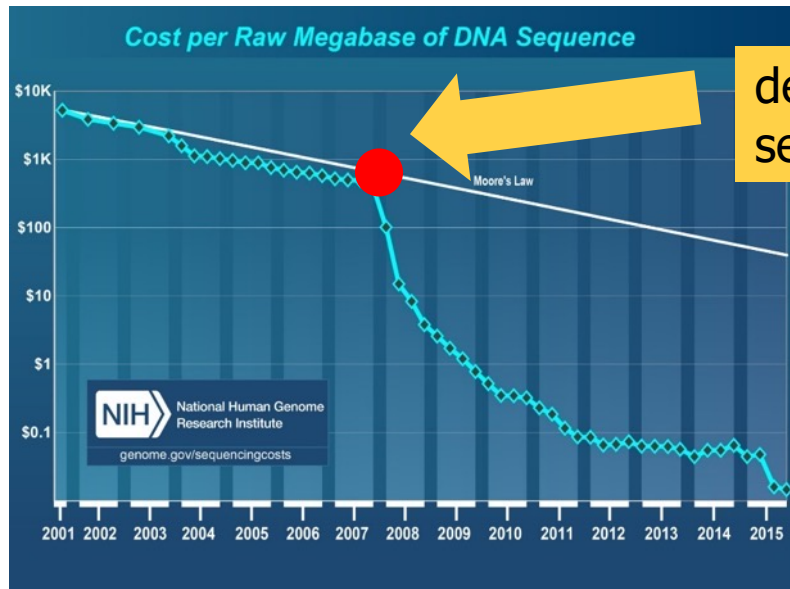
Memory and compute requirements



1800x more compute
In just **2 years**

Tomorrow, **multi-trillion** parameter models

Huge Demand for Performance & Efficiency

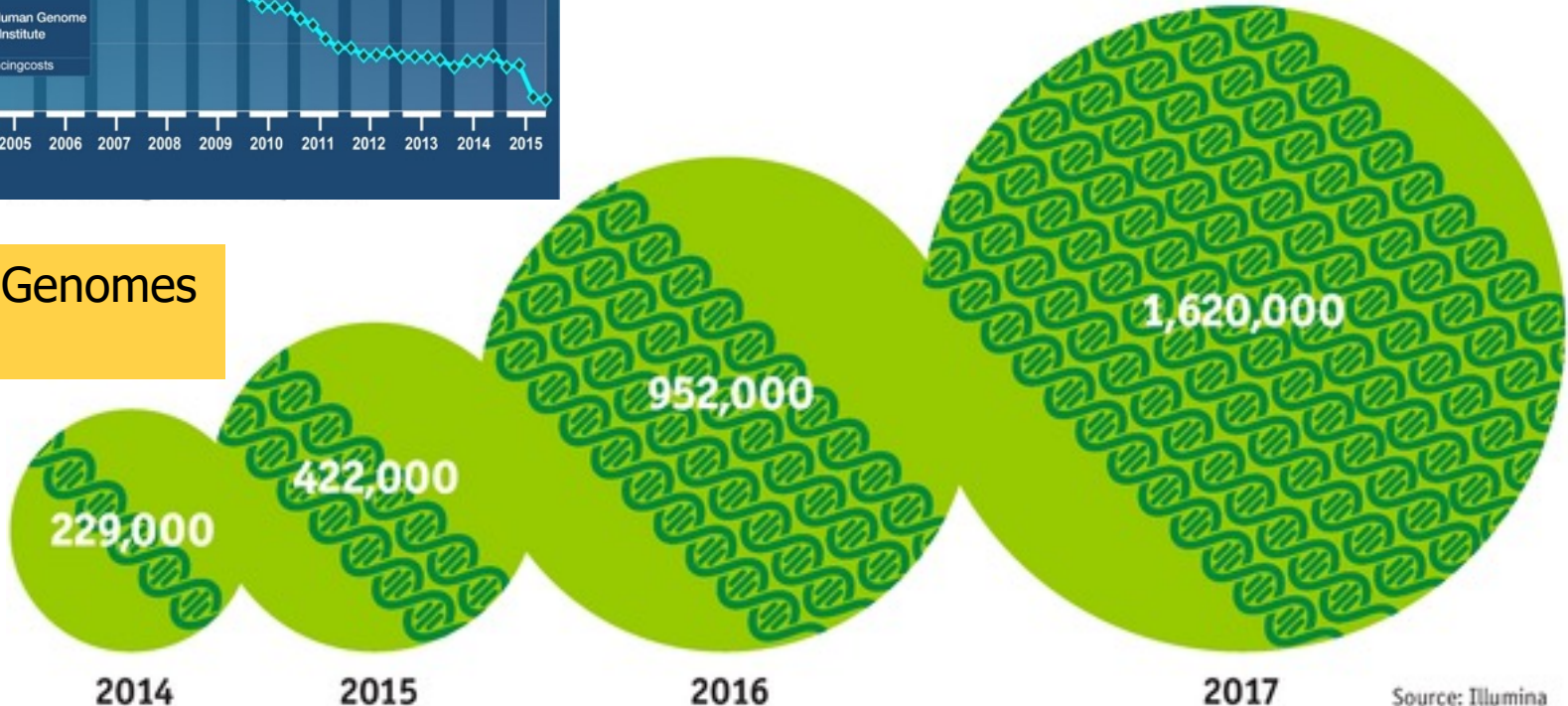


development of new sequencing technologies



Oxford Nanopore MinION

Number of Genomes Sequenced



The Economist

Do We Want This?



Or This?



High Performance,

Energy Efficient,

Sustainable

(All at the Same Time)

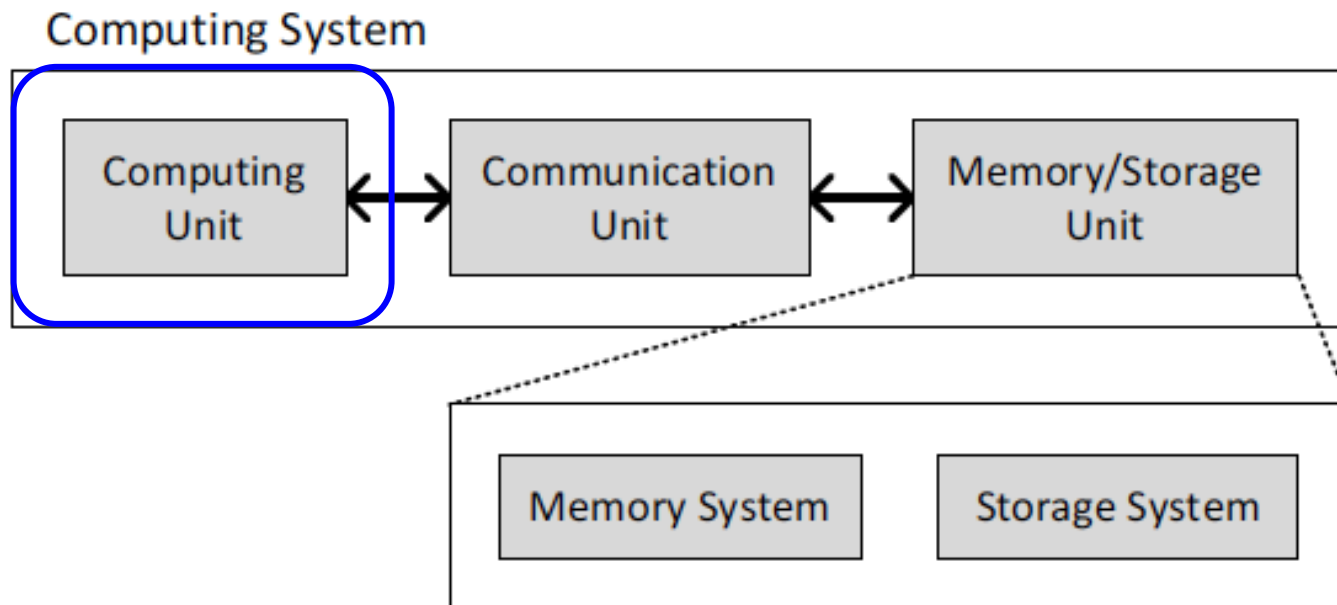
The Problem

Data access is the major performance and energy bottleneck

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

Today's Computing Systems

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



It's the Memory, Stupid!

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

RICHARD SITES

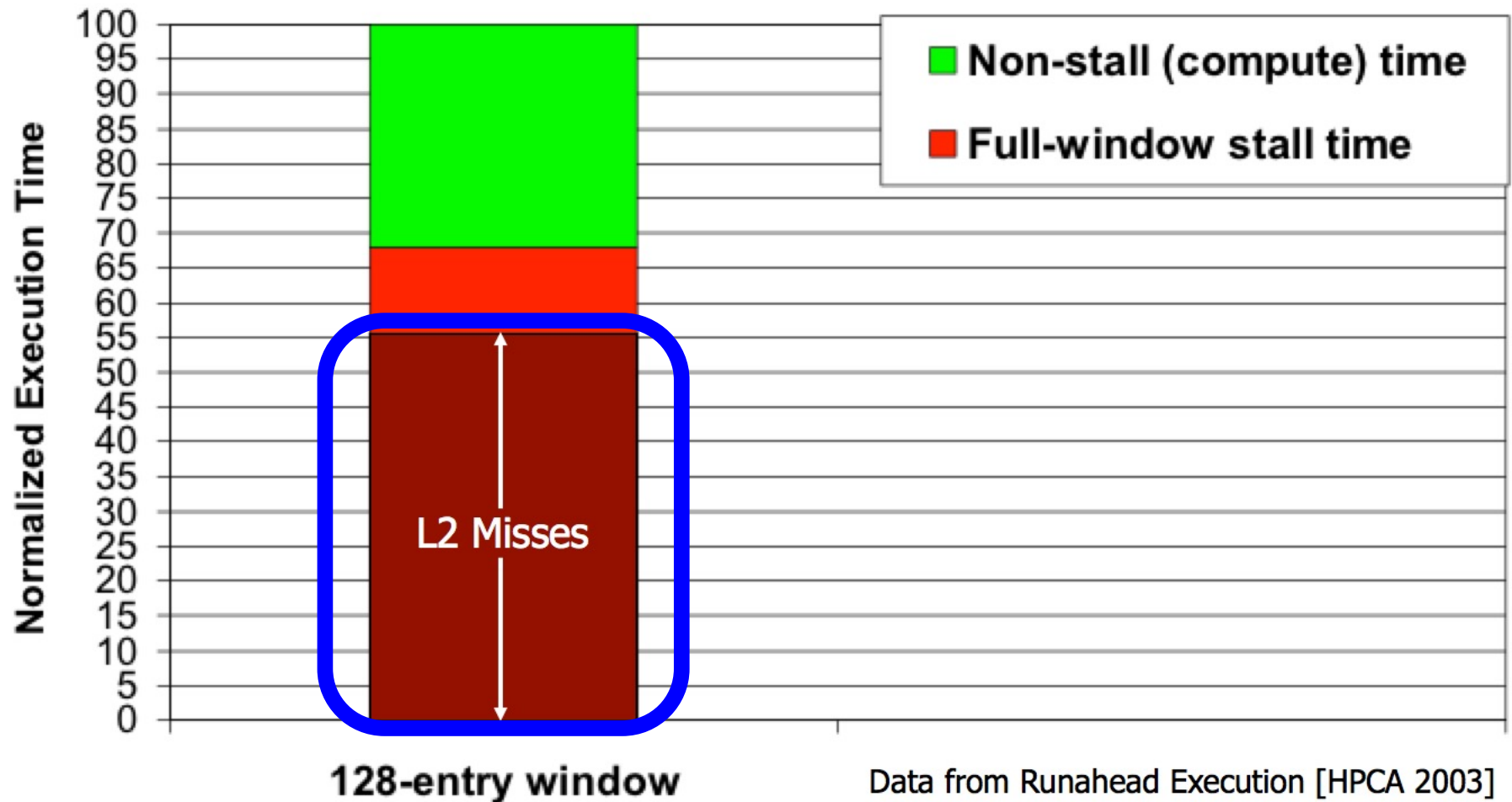
It's the Memory, Stupid!

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

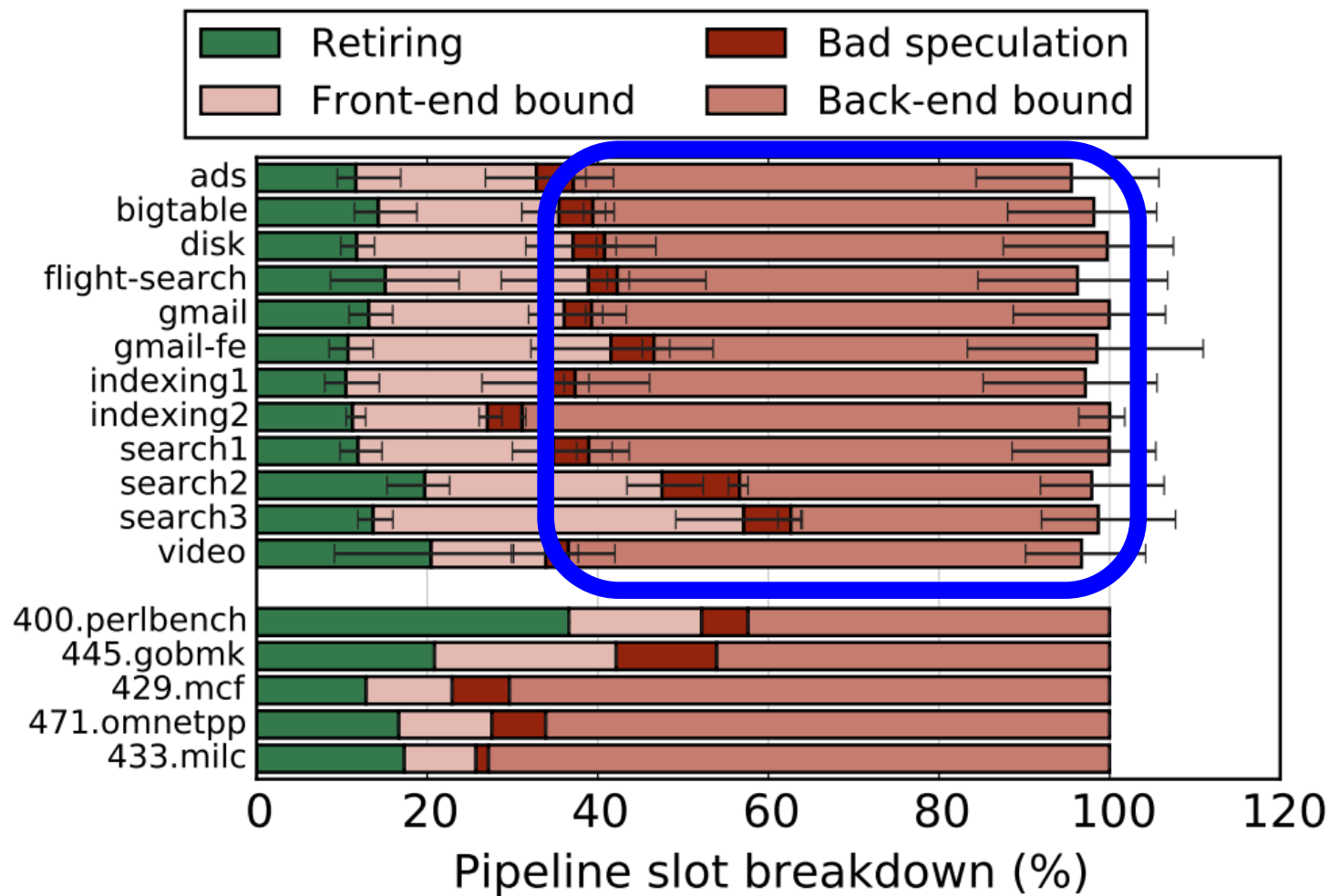
I expect that over the coming decade memory subsystem design will be the *only* important design issue for microprocessors.

Processor-Centric System Performance



Processor-Centric System Performance

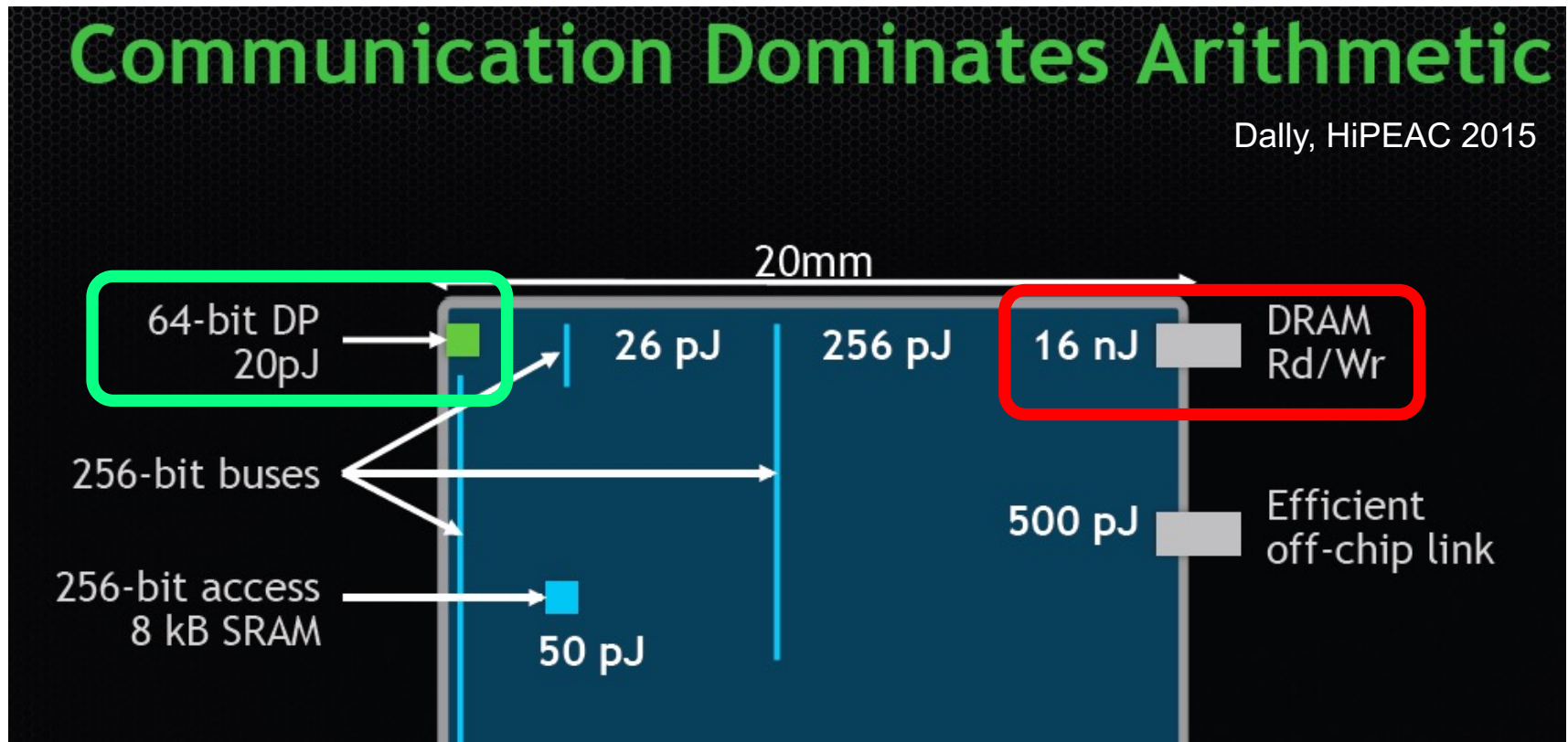
- All of Google's Data Center Workloads (2015):



Data Movement vs. Computation Energy

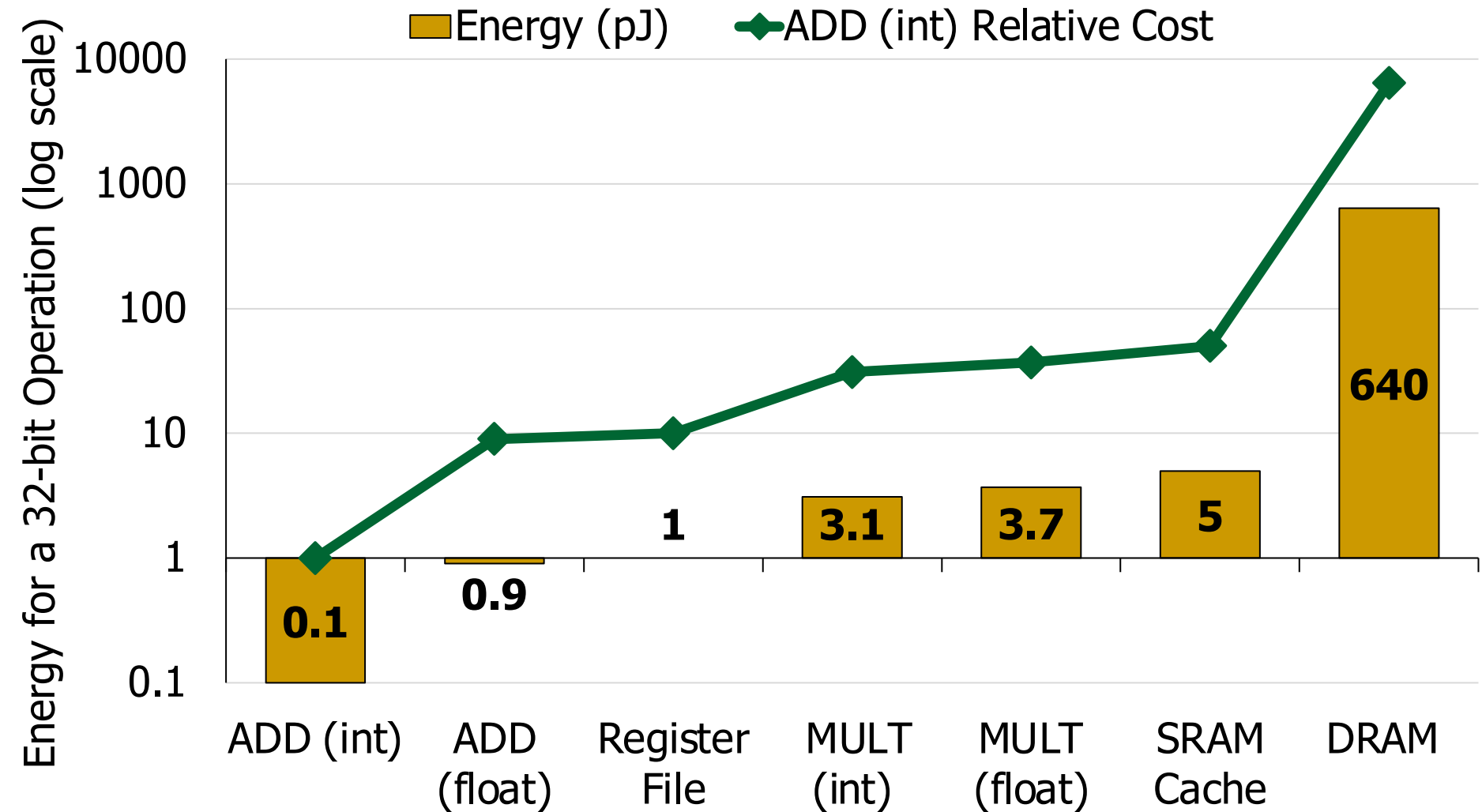
Communication Dominates Arithmetic

Dally, HiPEAC 2015

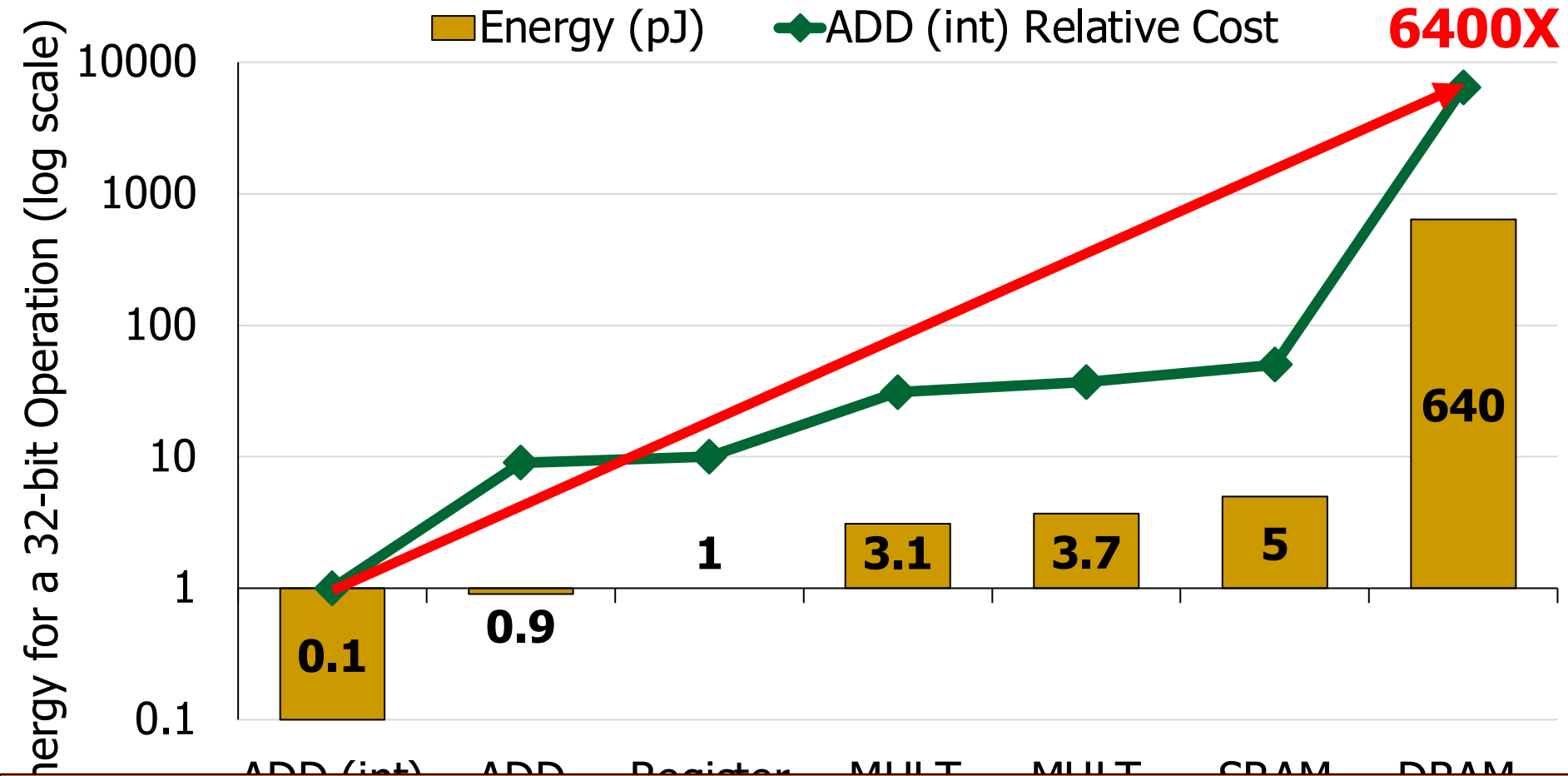


A memory access consumes $\sim 100\text{-}1000\times$ the energy of a complex addition

Data Movement vs. Computation Energy



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 Boroumand¹

Saugata Ghose¹

Youngsok Kim²

Rachata Ausavarungnirun¹

Eric Shiu³

Rahul Thakur³

Daehyun Kim^{4,3}

Aki Kuusela³

Allan Knies³

Parthasarathy Ranganathan³

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^{†◇}
Geraldo F. Oliveira^{*}

Saugata Ghose[‡]
Xiaoyu Ma[§]

Berkin Akin[§]
Eric Shiu[§]

Ravi Narayanaswami[§]
Onur Mutlu^{*†}

[†]Carnegie Mellon Univ.

[◇]Stanford Univ.

[‡]Univ. of Illinois Urbana-Champaign

[§]Google

^{*}ETH Zürich

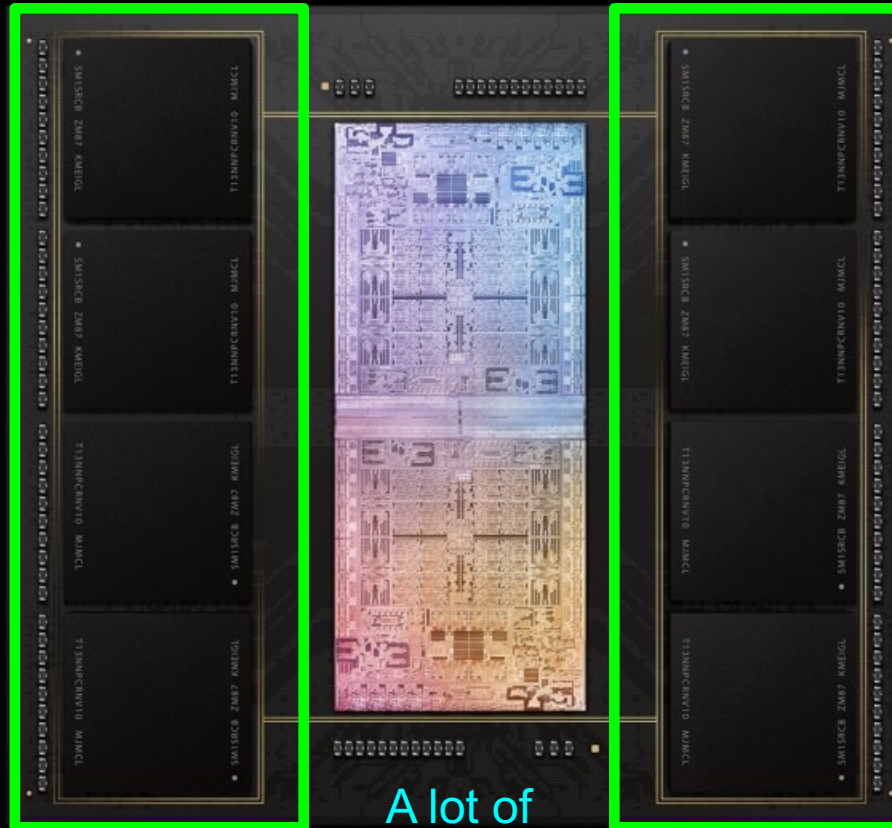
Processing of data
is performed
far away from the data

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

Sensors



A lot of
SRAM

Storage

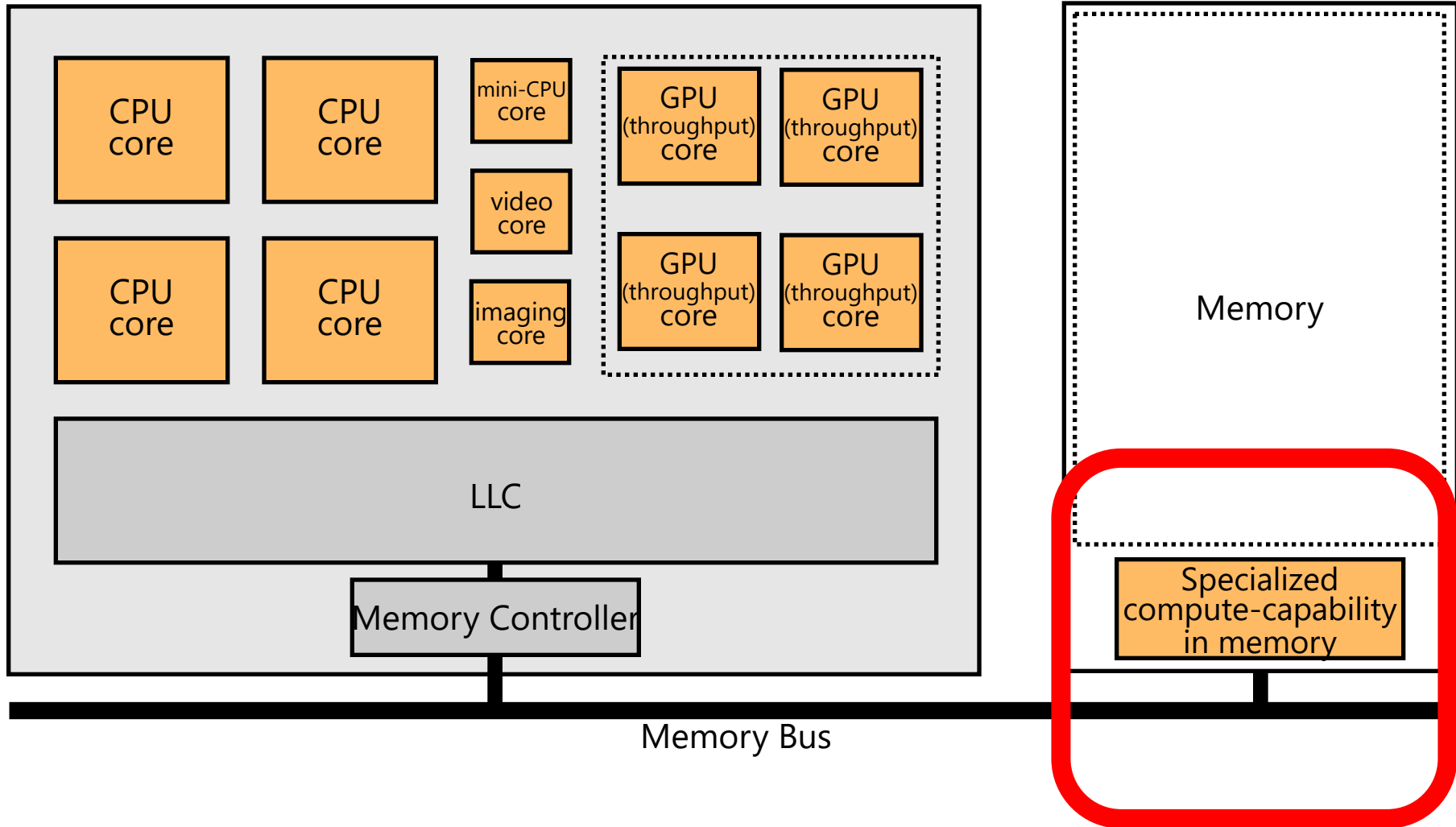
DRAM

DRAM

Storage

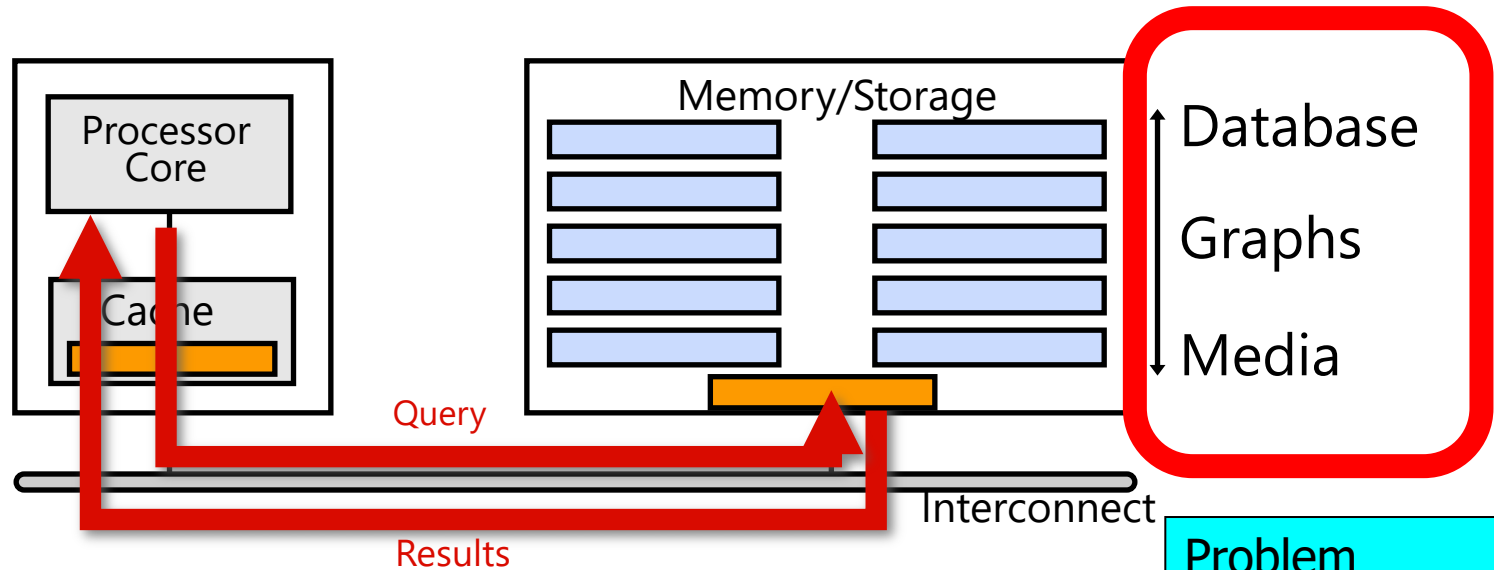
Apple M1 Ultra System (2022)

Memory/Storage as an Accelerator



Memory similar to a "conventional" accelerator

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?

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

An Intelligent Architecture Handles Data Well

Computing Systems Today ...

- Are **processor-centric** vs. **data-centric**
- Make **designer-dictated** decisions vs. **data-driven**
- Make **component-based myopic** decisions vs. **data-aware**

Data-centric

Data-driven

Data-aware

Vision: Storage-Centric Computing (I)

Storage system is a heterogeneous computing device with hybrid memory

Storage system enables data-centric design of systems & workloads

Application-driven customization enables a powerful data-centric engine

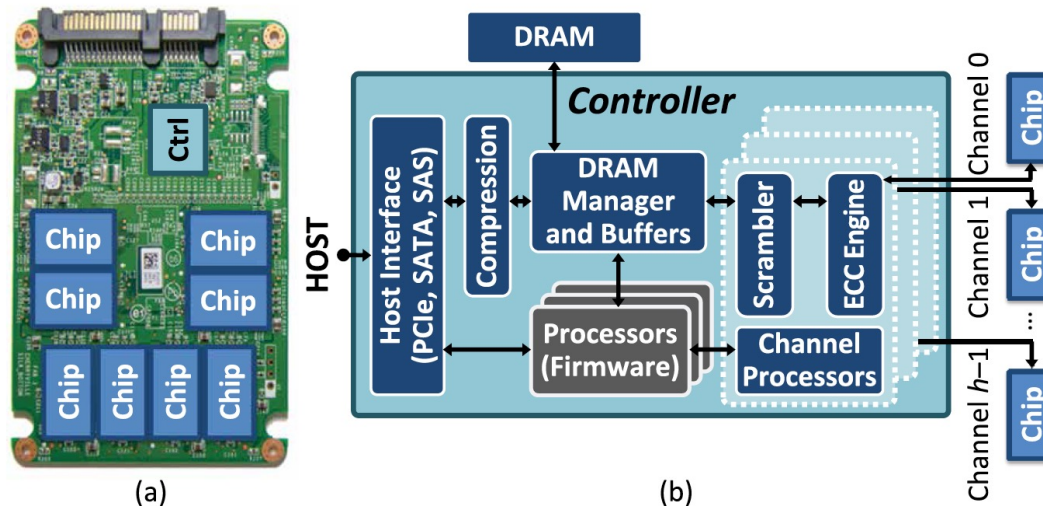
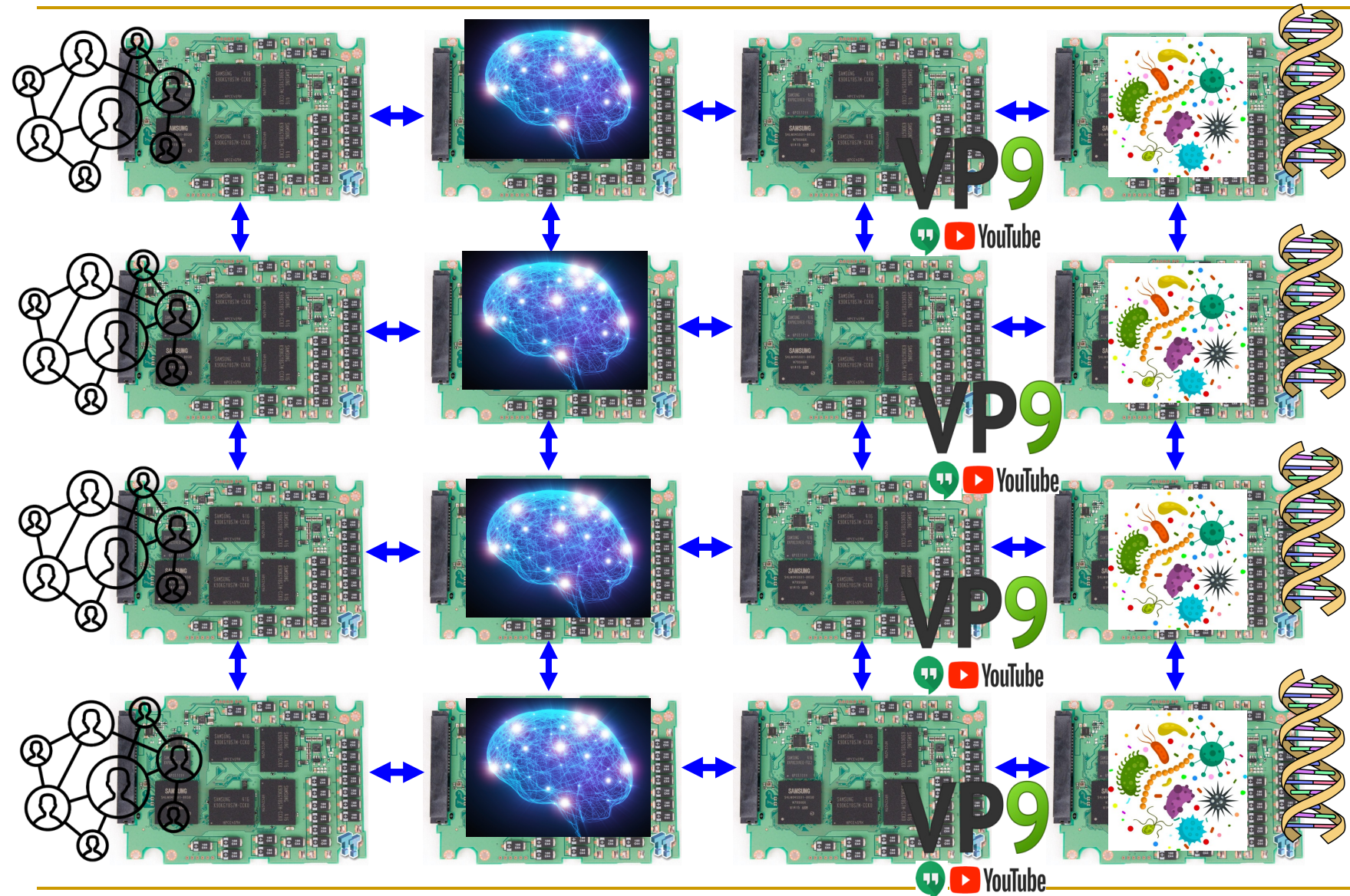


Fig. 1. (a) SSD system architecture, showing controller (Ctrl) and chips. (b) Detailed view of connections between controller components and chips.

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

Vision: Storage-Centric Computing (II)



Workload-Customized Storage-Centric Computing

- Software and hardware customized for major workloads
 - Genomics
 - Video analytics
 - Data & graph analytics
 - Machine learning
 - ...
- Data-centric (processing capability in all memories)
- Data-driven (design & decision making)
- Data-aware (optimization & design)
- Unified interfaces for efficient & fast communication

Processing in Storage: Two Approaches

1. Processing using Storage
2. Processing **near** Storage

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 Ghiasi¹ Jisung Park¹ Harun Mustafa¹ Jeremie Kim¹ Ataberk Olgun¹
Arvid Gollwitzer¹ Damla Senol Cali² Can Firtina¹ Haiyu Mao¹ Nour Almadhoun Alserr¹
Rachata Ausavarungnirun³ Nandita Vijaykumar⁴ Mohammed Alser¹ Onur Mutlu¹

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

Genome Sequence Analysis

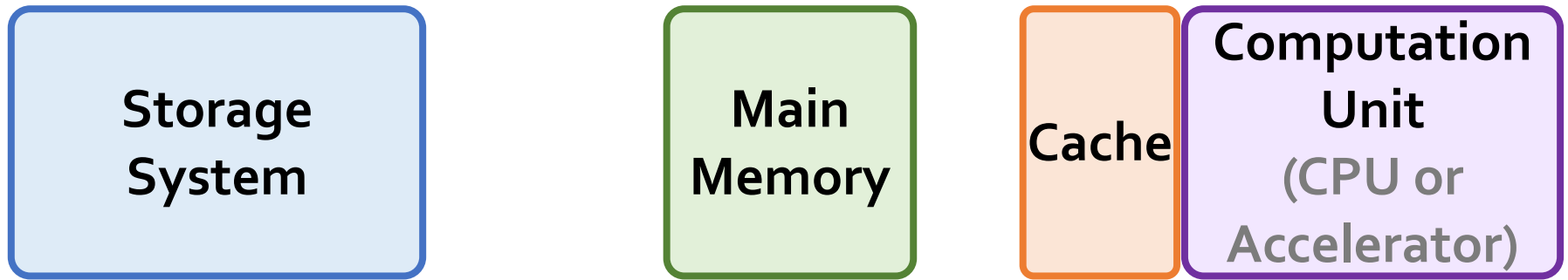
- **Read mapping:** first key step in genome sequence analysis
 - Aligns reads to potential matching locations in the reference genome
 - For each matching location, the alignment step finds the degree of similarity (alignment score)



- 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

Data Movement from Storage

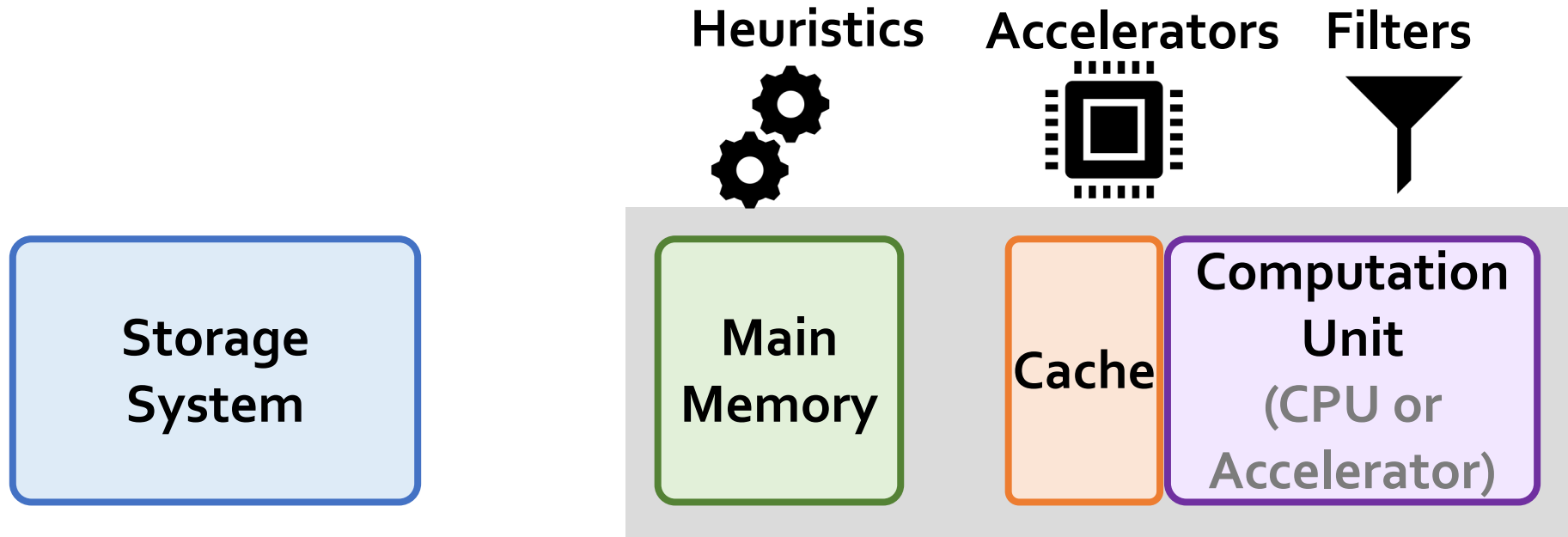


Computation overhead



Data movement overhead

Compute-Centric Accelerators



Computation overhead

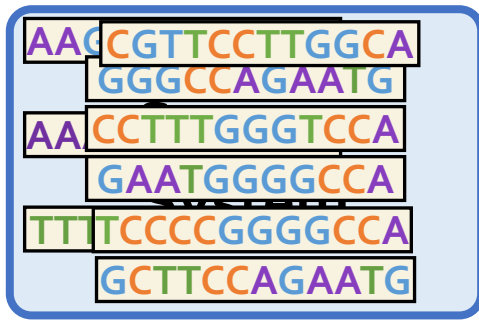


Data movement overhead

Key Idea: In-Storage Filtering



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



Filtered Reads

**Main
Memory**

Cache

**Computation
Unit**
(CPU or
Accelerator)

Exactly-matching reads

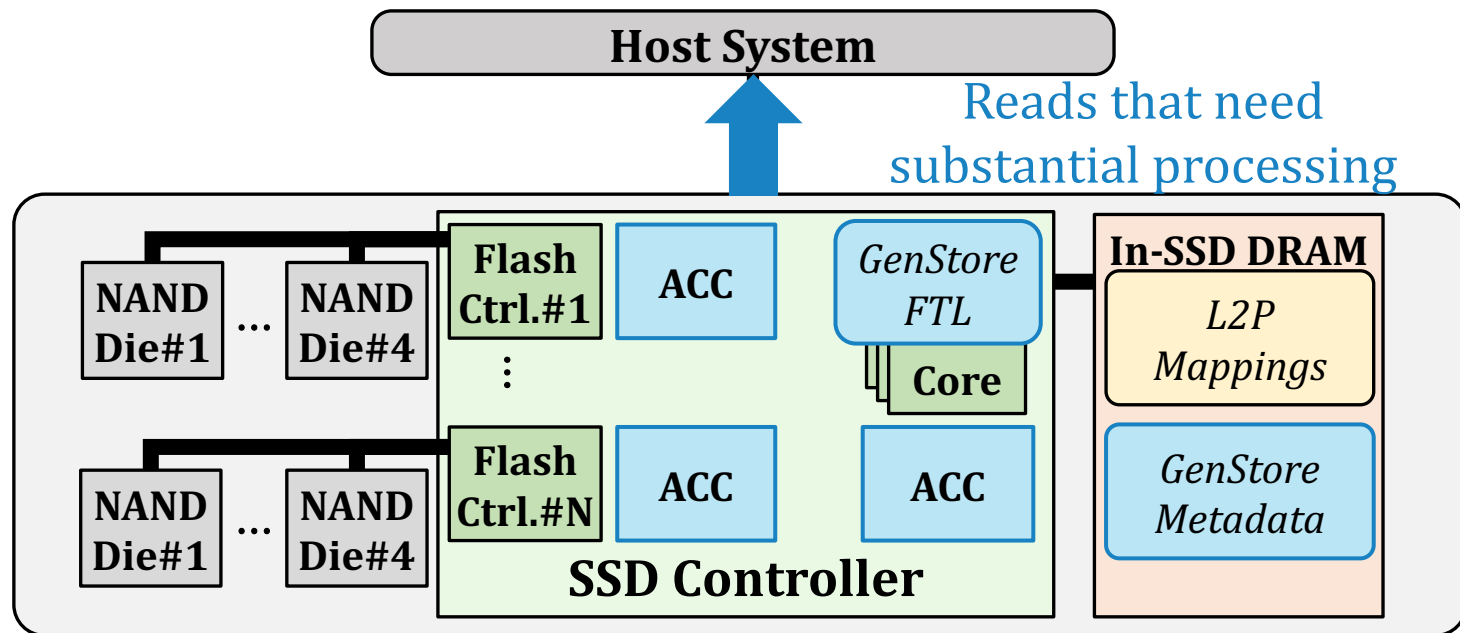
Do not need expensive approximate string matching during alignment

Non-matching reads

Do not have potential matching locations and can skip alignment

GenStore

- **Key idea:** Filter reads that do not require alignment *inside the storage system*
- **Challenges**
 - **Different behavior** across read mapping workloads
 - **Limited** hardware resources in the SSD



Filtering Opportunities

- Sequencing machines produce one of two kinds of reads
 - Short reads: highly accurate and short
 - Long reads: less accurate and long

Reads that do not require the expensive alignment step:

Exactly-matching reads

Do not need expensive approximate string matching during alignment

- Low sequencing error rates (short reads) combined with
- Low genetic variation

Non-matching reads

Do not have potential matching locations, so they skip alignment

- High sequencing error rates (long reads) or
- High genetic variation (short or long reads)

GenStore

GenStore-**EM** for Exactly-Matching Reads

GenStore-**NM** for Non-Matching Reads

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

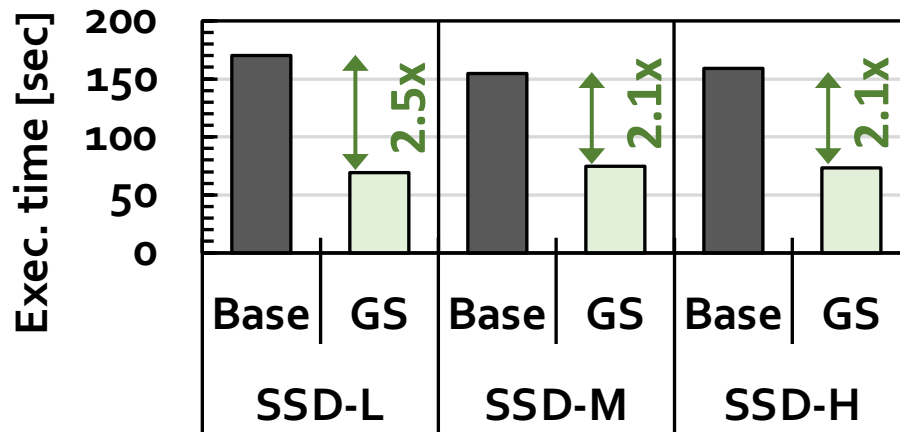


Data movement overhead

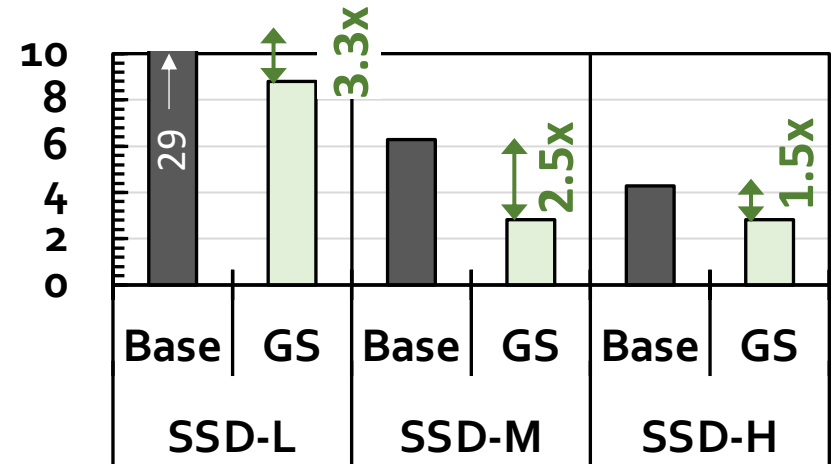
GenStore provides significant speedup (1.4x - 33.6x) and
energy reduction (3.9x - 29.2x) at low cost

Performance – GenStore-EM

With the Software Mapper



With the Hardware Mapper

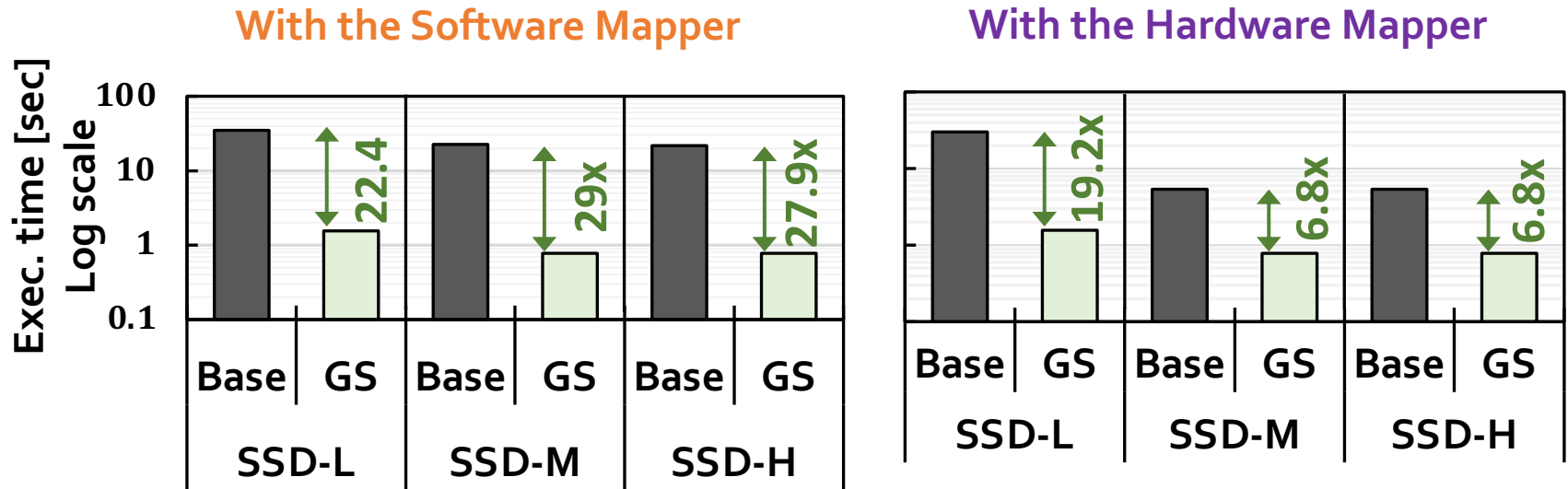


2.1x - 2.5x speedup compared to the software Base

1.5x – 3.3x speedup compared to the hardware Base

On average 3.92x energy reduction

Performance – GenStore-NM



22.4x – 27.9x speedup compared to the software Base

6.8x – 19.2x speedup compared to the hardware Base

On average **27.2x energy reduction**

Area and Power Consumption

- Based on **Synthesis** of **GenStore** accelerators using the Synopsys Design Compiler @ 65nm technology node

| Logic unit | # of instances | Area [mm ²] | Power [mW] |
|-----------------------------------|----------------|-------------------------|------------|
| Comparator | 1 per SSD | 0.0007 | 0.14 |
| K -mer Window | 2 per channel | 0.0018 | 0.27 |
| Hash Accelerator | 2 per SSD | 0.008 | 1.8 |
| Location Buffer | 1 per channel | 0.00725 | 0.37375 |
| Chaining Buffer | 1 per channel | 0.008 | 0.95 |
| Chaining PE | 1 per channel | 0.004 | 0.98 |
| Control | 1 per SSD | 0.0002 | 0.11 |
| <i>Total for an 8-channel SSD</i> | - | 0.2 | 26.6 |

Only **0.006%** of a **14nm Intel Processor**, less than **9.5%** of the three **ARM processors** in a **SATA SSD controller**

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 Ghiasi¹ Jisung Park¹ Harun Mustafa¹ Jeremie Kim¹ Ataberk Olgun¹
Arvid Gollwitzer¹ Damla Senol Cali² Can Firtina¹ Haiyu Mao¹ Nour Almadhoun Alserr¹
Rachata Ausavarungnirun³ Nandita Vijaykumar⁴ Mohammed Alser¹ Onur Mutlu¹

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

Tight Integration of Genome Analysis Tasks

- 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 Mao¹ Mohammed Alser¹ Mohammad Sadrosadati¹ Can Firtina¹ Akanksha Baranwal¹
Damla Senol Cali² Aditya Manglik¹ Nour Almadhoun Alserr¹ Onur Mutlu¹
¹ETH Zürich ²Bionano Genomics

MegIS: High-Performance and Low-Cost Metagenomic Analysis with In-Storage Processing

Nika Mansouri Ghiasi, Mohammad Sadrosadati, Harun Mustafa, Arvid Gollwitzer,
Can Firtina, Julien Eudine, Haiyu Mao, Joël Lindegger, Meryem Banu Cavlak,
Mohammed Alser, Jisung Park, Onur Mutlu

SAFARI

ETH zürich

POSTECH

What is Metagenomics?

- **Metagenomics:** *Analysis of genome sequences of diverse organisms within a shared environment (e.g., blood, soil, oceans)*
- **Overcomes the limitations of traditional genomics**
 - Bypasses the need for isolating and culturing individual species
 - Addresses clinical/environmental challenges where culturing is impractical
- **Has led to ground-breaking advancements**
 - Precision medicine
 - Understanding microbial diversity of an environment
 - Discovering early warnings of communicable diseases

Key Steps of the Metagenomic Workflow

Sequencing

- Extracts the genomic information of all organisms in a sample into digital data
- A sequencing machine generates randomly sampled, inexact fragments of genomic information, called reads

Basecalling

- Converts the raw sequencer data into sequences of characters (A, C, G, and T)

Analysis

- Determines the species present/absent in the sample
- Determines their relative abundances

Why Accelerating the Analysis Step?

*Metagenomic analysis is typically performed
much more frequently compared to the other two steps*

*Even for one round of analysis,
the analysis step bottlenecks
the **end-to-end performance** and **energy efficiency***

*Due to rapid advancements in sequencing technologies,
the gap between sequencing and analysis is widening*

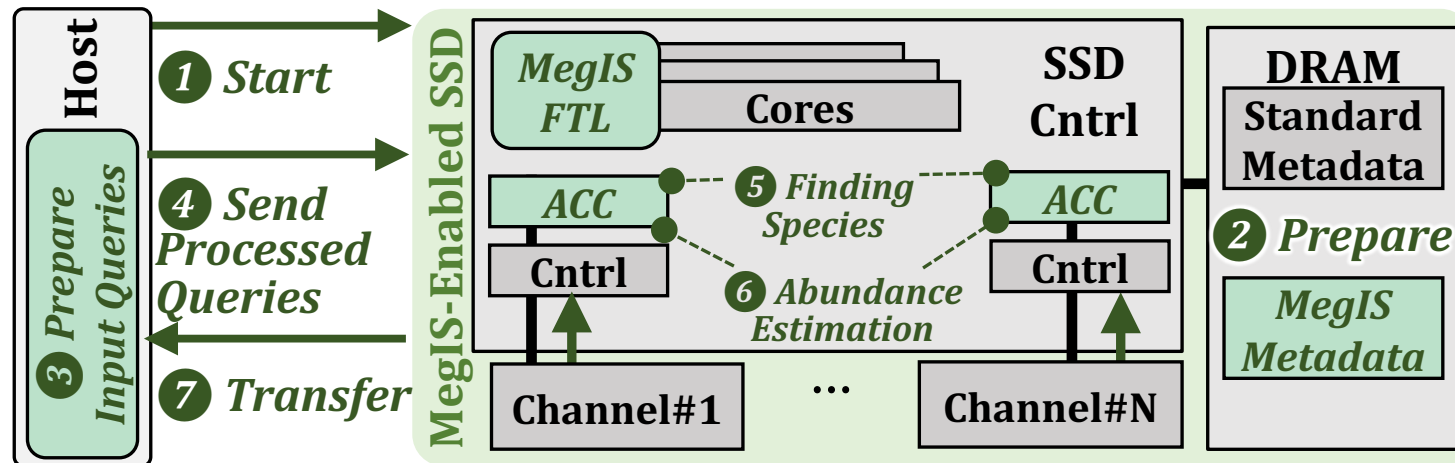
Executive Summary

- **Problem**: Metagenomic analysis suffers from significant storage I/O data movement overhead due to accessing large amounts of low-reuse data
- **Goal**: Improve the performance of metagenomic analysis by reducing data movement overhead right from the storage system, in a cost-effective manner
- **Challenge**: While in-storage processing can be a promising direction, none of the existing approaches can be effectively implemented inside the storage system due to the limited resources available in storage devices
- **Idea**: Enable cooperative ISP for metagenomics, where we do not merely focus on the storage system and, instead, capitalize on the strengths of processing both inside and outside the storage system via a synergistic design
- **MegIS**: *The first in-storage processing system tailored for metagenomic analysis*
 - Leverages and orchestrates processing inside and outside the storage system
 - 1) Task partitioning, 2) Data/computation flow coordination, 3) Storage-aware algorithms, 4) Lightweight in-storage accelerators, and 5) Data mapping
- **Results**: Significant speedup (1.5x – 100.2x) and energy reduction (1.9x – 25.7x) with high accuracy and at low cost

MegIS

The *first* ISP system for end-to-end metagenomic analysis

- Cooperative ISP
- Enables an efficient pipeline between the host and the storage
- Maximally leverage and orchestrate their capabilities



Evaluation Methodology

Metagenomic Analysis Tools

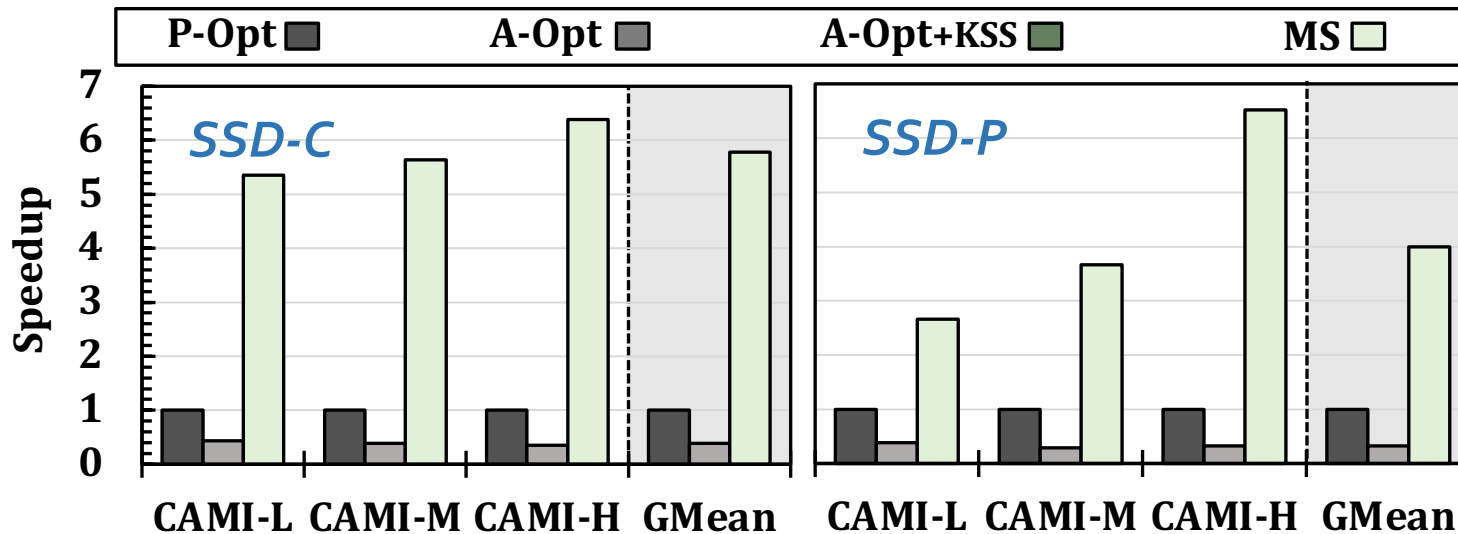
- **Performance-Optimized SW:** Kraken2 [Genome Biology'19]
- **Accuracy-Optimized SW:** Metalign [Genome Biology'20]
- **PIM HW:** Sieve [ISCA'21]

SSD Configurations

- **SSD-C:** with **SATA₃** interface (**0.5 GB/s** sequential read bandwidth)
- **SSD-P:** with **PCIe Gen₄** interface (**7 GB/s** sequential read bandwidth)

Performance

- With inputs with Low, Medium, and High genetic diversity



2.66x – 6.38x speedup compared to P-Opt

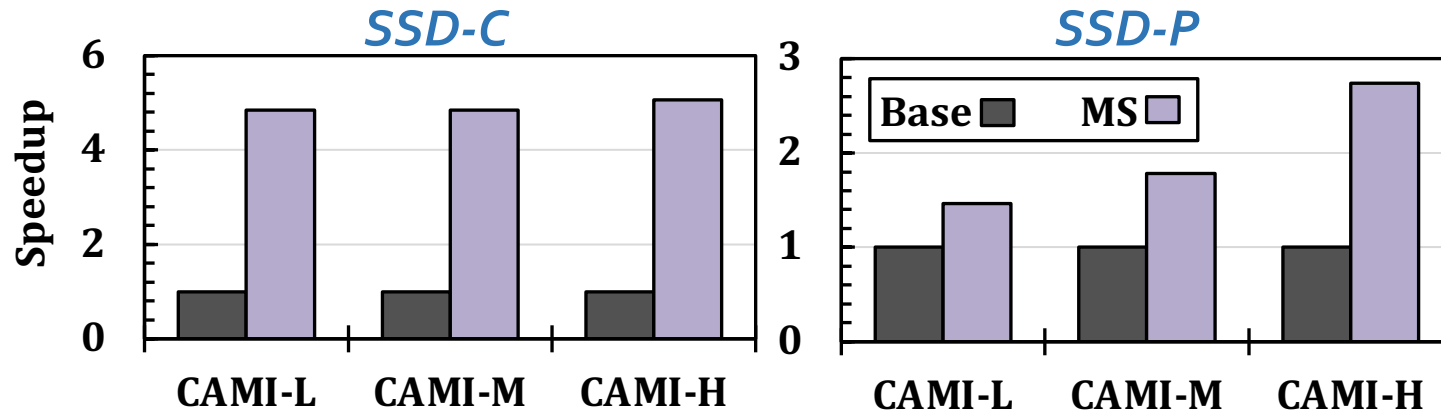
6.93x – 20.39x speedup compared to A-Opt

5.36x average energy reduction compared to P-Opt

15.16x average energy reduction compared to A-Opt

Performance Compared to PIM

- With inputs with Low, Medium, and High genetic diversity

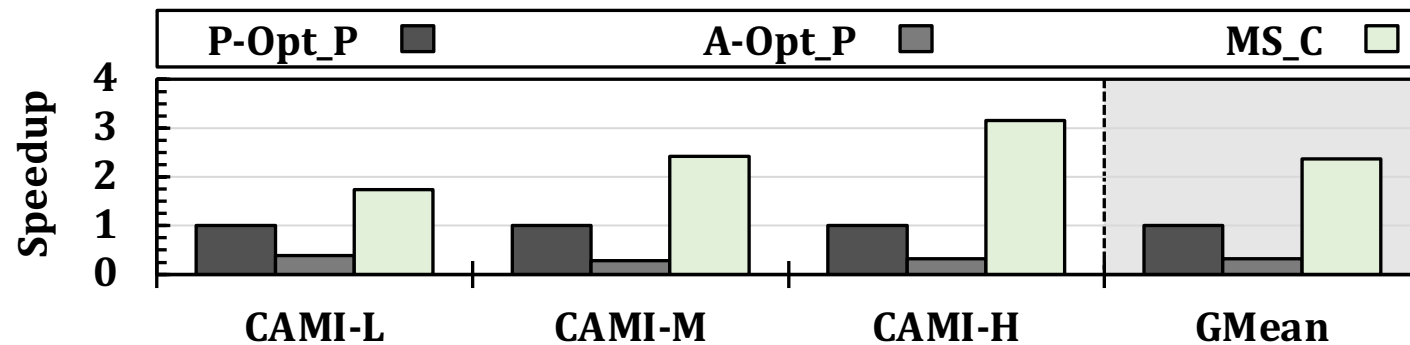


1.46x – 5.06x speedup compared to the PIM baseline

1.88x (3.49x) average (max) energy reduction
compared to the PIM baseline

System Cost Efficiency

- MegIS on a cost-optimized system
 - With SSD-C and 64-GB DRAM
- Baselines on a performance-optimized system
 - With SSD-P and 1-TB DRAM



2.37x average speedup compared to P-Opt

7.20x average speedup compared to A-Opt

Area and Power

- Based on **Synthesis** of **MegIS** accelerators using the Synopsys Design Compiler @ 65nm technology node

| Logic unit | # of instances | Area [mm ²] | Power [mW] |
|-----------------------------------|----------------|-------------------------|--------------|
| Intersect (120-bit) | 1 per channel | 0.001361 | 0.284 |
| k-mer Registers (2× 120-bit) | 1 per channel | 0.002821 | 0.645 |
| Index Generator (64-bit) | 1 per channel | 0.000272 | 0.025 |
| Control Unit | 1 per SSD | 0.000188 | 0.026 |
| Total for an 8-channel SSD | - | 0.04 | 7.658 |

Only **1.7%** of the area of three 28-nm ARM Cortex R4 cores
in a SATA SSD controller

Other Results in the Paper

- MegIS's performance with the SSD cores
- MegIS's performance outside SSD
- Performance with varying
 - Database sizes
 - Memory capacities
 - #SSDs
 - #Channels
 - #Samples
- MegIS's performance for abundance estimation

Processing in Storage: Two Approaches

1. Processing **using** Storage
2. Processing near Storage

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](#)]

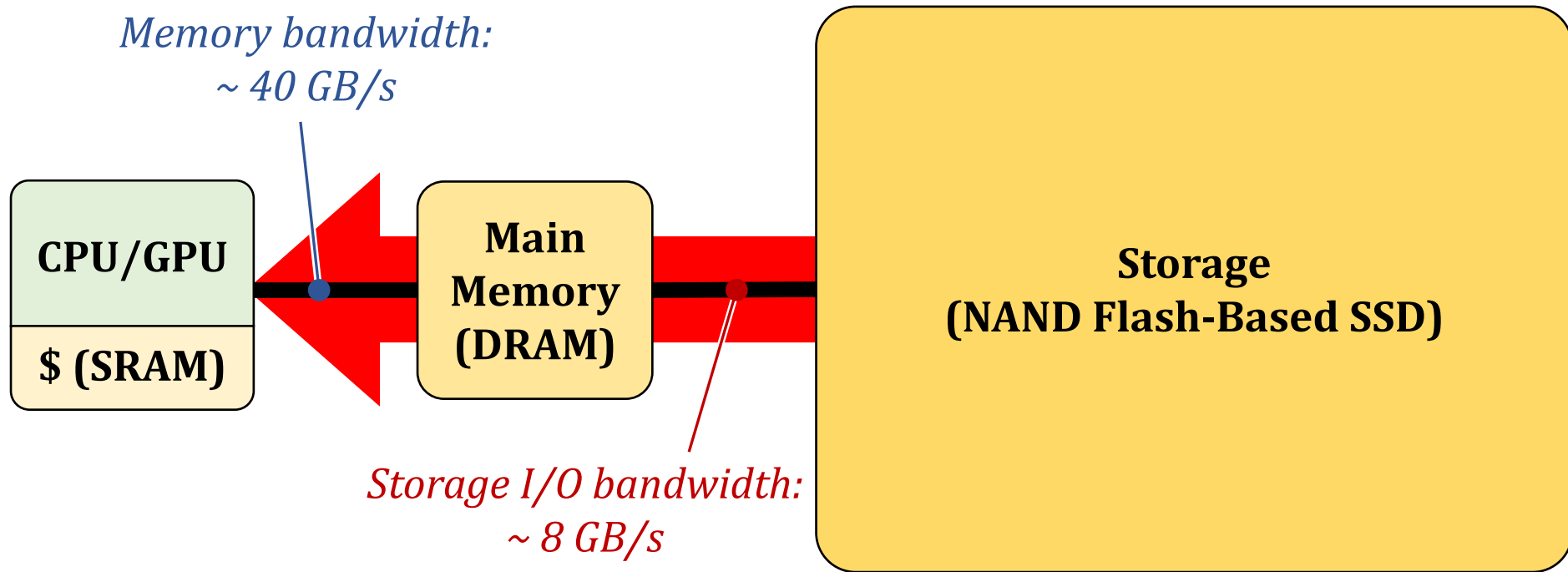
Flash-Cosmos: In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory

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

Data-Movement Bottleneck

- Conventional systems: Outside-storage processing (OSP) that must move the entire data to CPUs/GPUs through the memory hierarchy

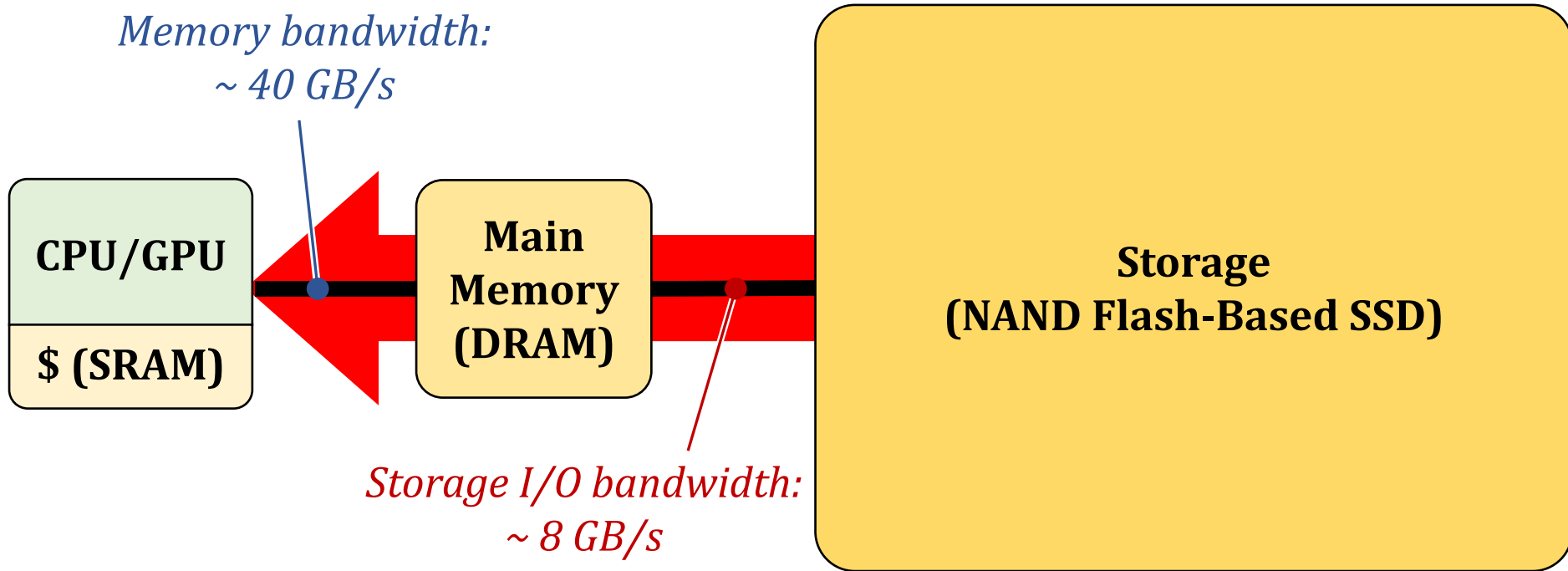


External I/O bandwidth of storage systems is the **main bottleneck** in conventional systems (OSP)

In-Storage Processing (ISP)

- Uses **in-storage compute units** (embedded cores or FPGA) to send **only the computation results**

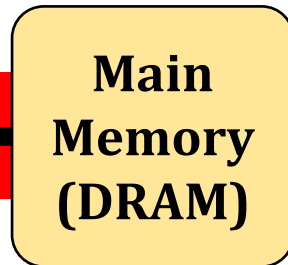
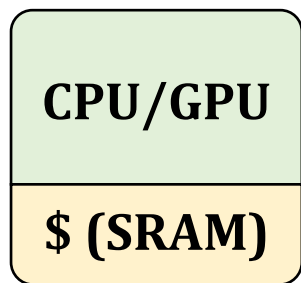
*Memory bandwidth:
~ 40 GB/s*



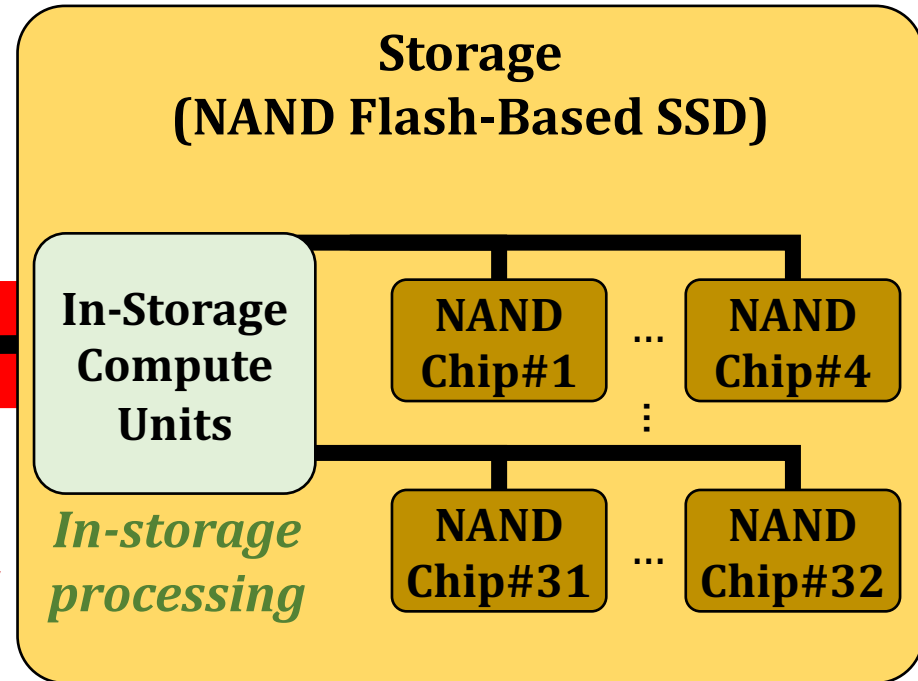
In-Storage Processing (ISP)

- Uses **in-storage compute units** (embedded cores or FPGA) to send **only the computation results**

*Memory bandwidth:
~ 40 GB/s*

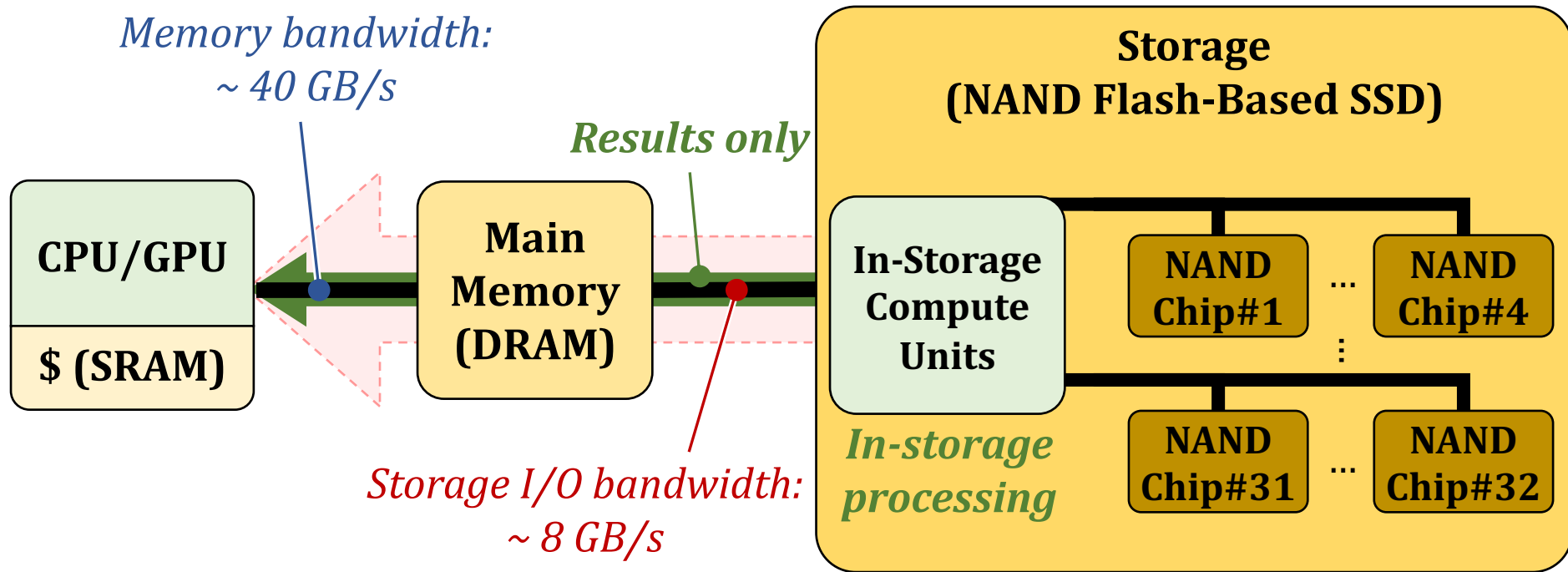


*Storage I/O bandwidth:
~ 8 GB/s*



In-Storage Processing (ISP)

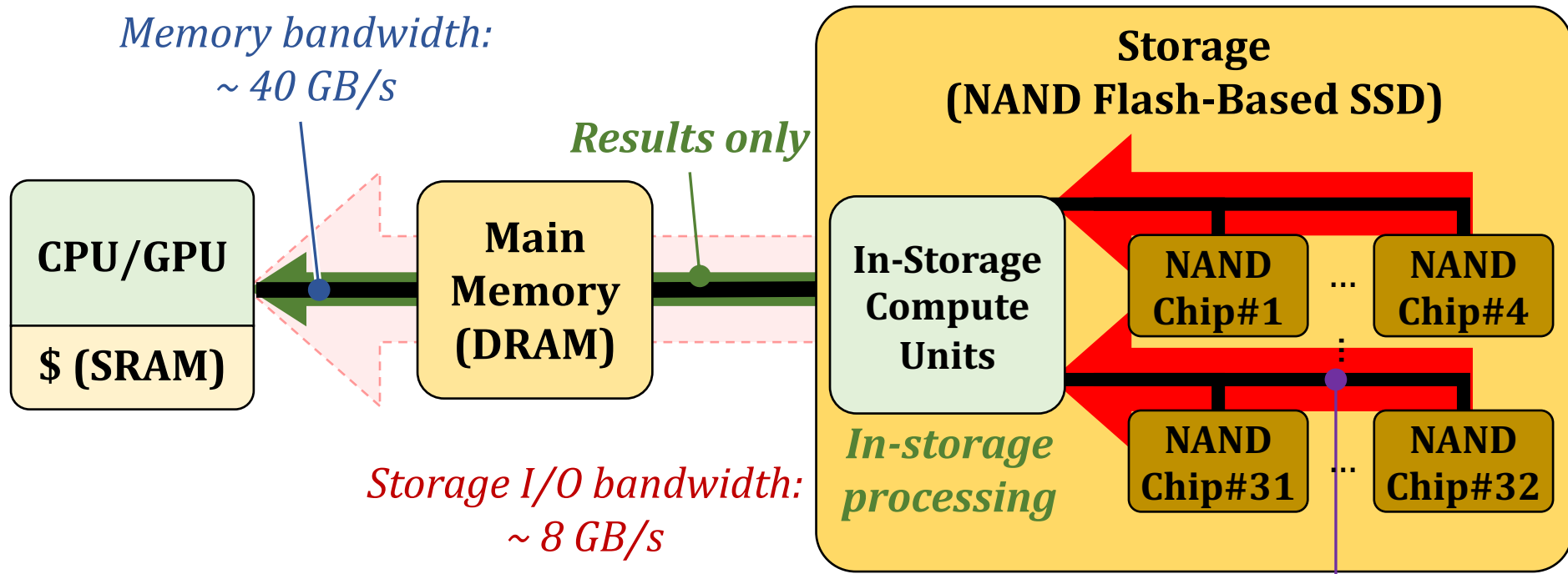
- Uses **in-storage compute units** (embedded cores or FPGA) to send **only the computation results**



ISP can mitigate data movement overhead by **reducing SSD-external data movement**

In-Storage Processing (ISP)

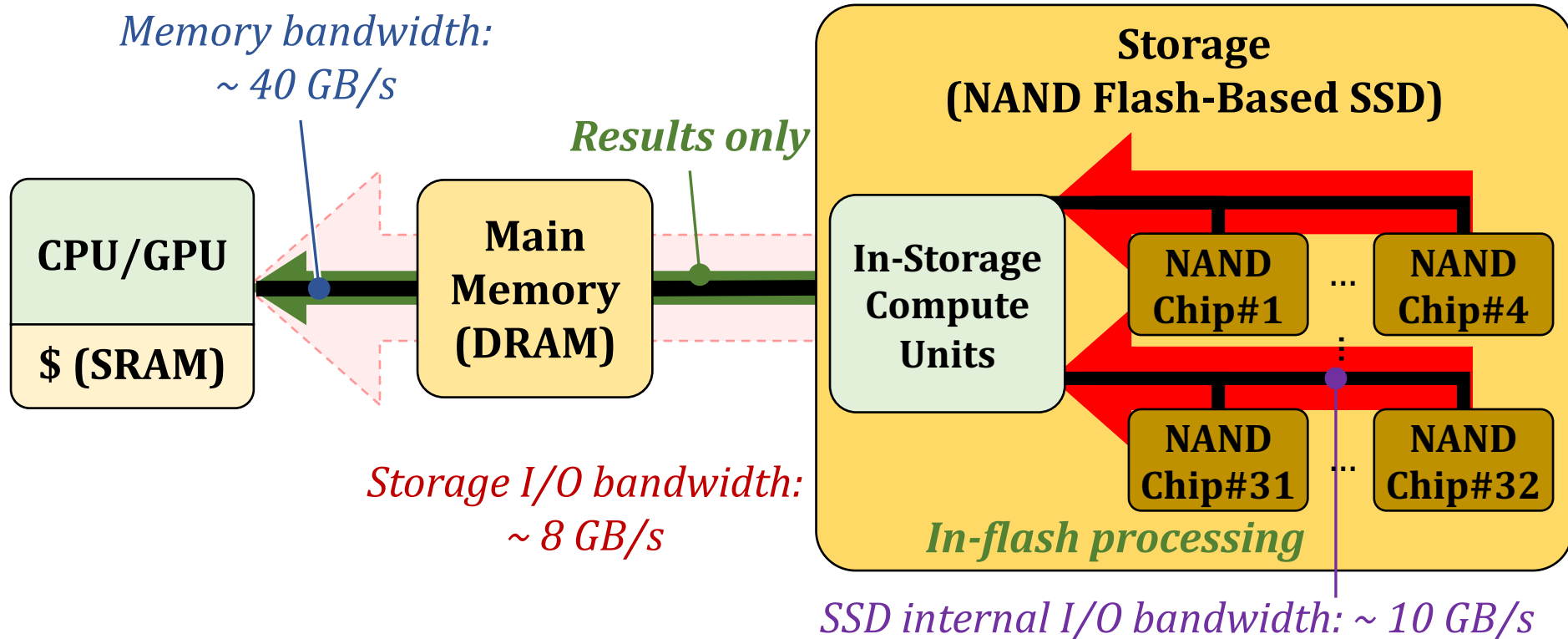
- Uses **in-storage compute units** (embedded cores or FPGA) to send **only the computation results**



SSD-internal bandwidth
becomes the **new bottleneck** in ISP

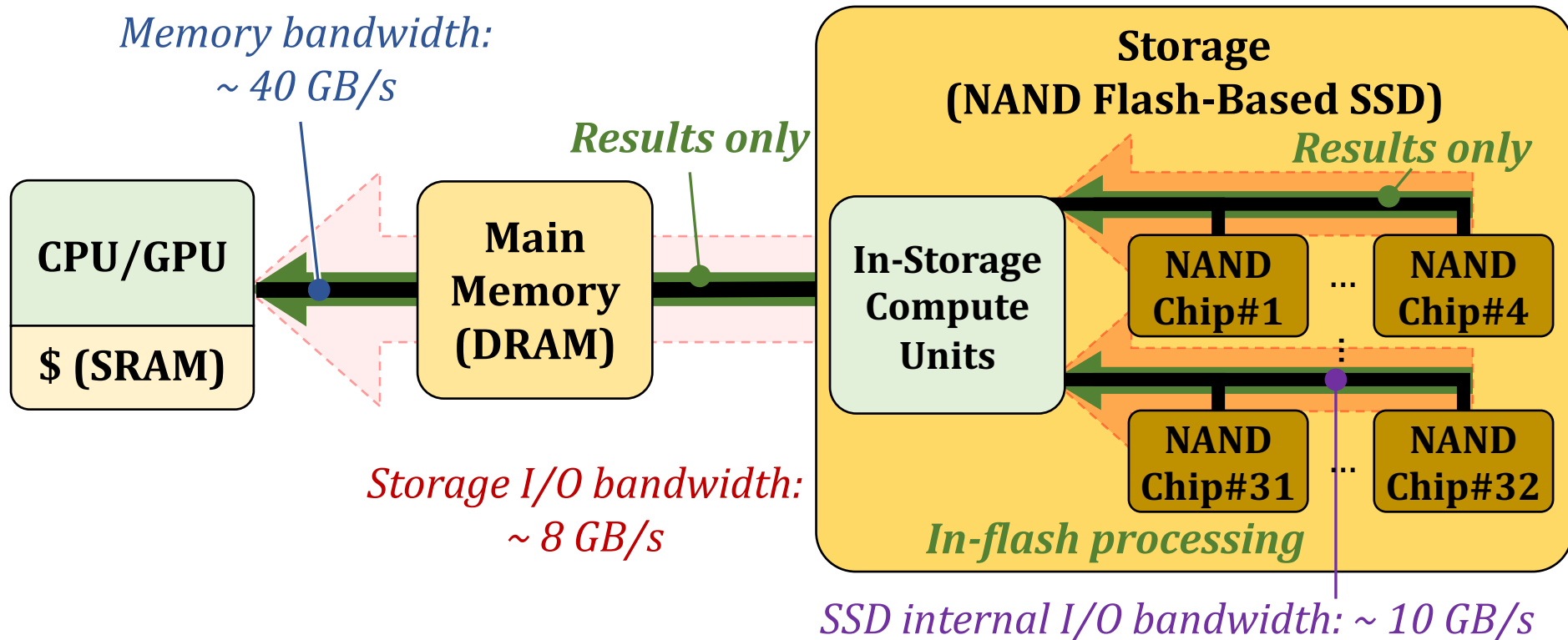
In-Flash Processing (IFP)

- Performs computation *inside* NAND flash chips



In-Flash Processing (IFP)

- Performs computation *inside* NAND flash chips

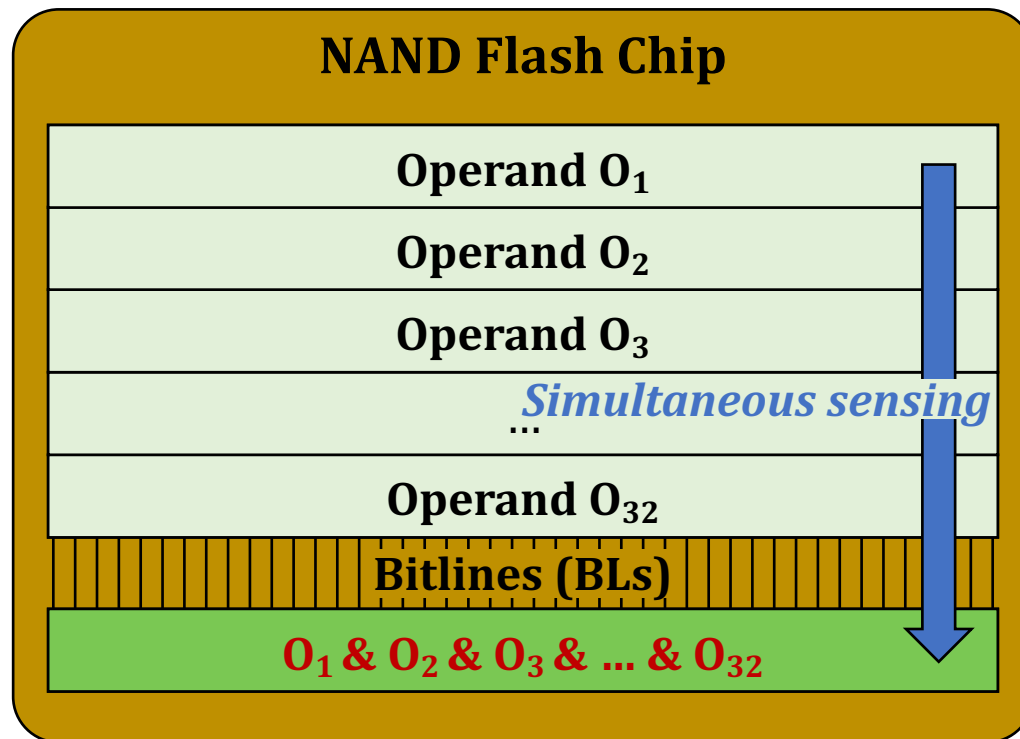


IFP fundamentally mitigates data movement

Our Proposal: Flash-Cosmos

▪ 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

*Operate
as a resistance (1)
or an open switch (0)*

WL₂

WL₃

WL₄

BL₁

BL₂

BL₃

BL₄

Result: 0

0

0

0

Multi-Wordline Sensing (MWS): Bitwise AND

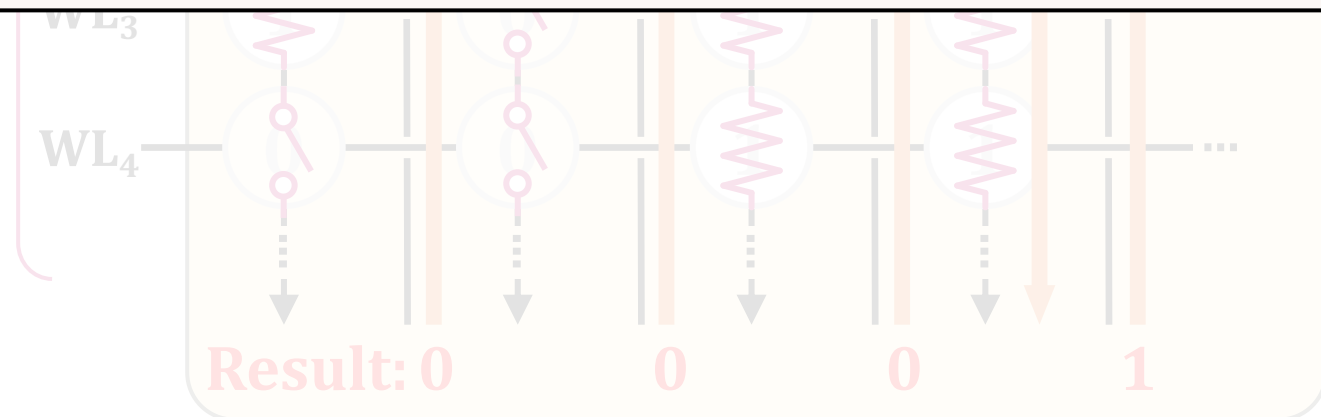
- Intra-Block MWS:

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



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>

Key Ideas



Multi-Wordline Sensing (MWS)
to enable in-flash bulk bitwise operations
via a single sensing operation



Enhanced SLC-Mode Programming (ESP)
to eliminate raw bit errors in stored data
(and thus in computation results)

Enhanced SLC-Mode Programming (ESP)

- **Goal:** eliminate raw bit errors in stored data (and computation results)
- **Key ideas**
 - Programs only a single bit per cell (SLC-mode programming)
 - Trades storage density for reliable computation
 - Performs more precise programming of the cells
 - Trades programming latency for reliable computation

Maximizes the reliability margin
between the different states of flash cells

Enhanced SLC-Mode Programming (ESP)

- To eliminate raw bit errors in stored data (and computation results)

Flash-Cosmos (ESP) enables
reliable in-flash computation
by trading storage density & programming latency

Storage & latency overheads affect
only data used in in-flash computation

Evaluation Methodology

▪ Real-device characterization

- To validate the feasibility and reliability of Flash-Cosmos
- Using 160 48-WL-layer 3D Triple-Level Cell NAND flash chips
 - 3,686,400 tested wordlines
- Under worst-case operating conditions
 - Under a 1-year retention time at 10K P/E cycles
 - Worst-case data patterns

▪ System-level evaluation

- Using the state-of-the-art SSD simulator (MQSim [Tavakkol+, FAST'18])
- Three real-world applications
 - Bitmap Indices (BMI): Bitwise AND of up to ~1,000 operands
 - Image Segmentation (IMS): Bitwise AND of 3 operands
 - K-clique Star Listing (KCS): Bitwise OR of up to 32 operands
- Baselines
 - Outside-Storage Processing (OSP): A multi-core CPU (Intel i7-11700K)
 - In-Storage Processing (ISP): An in-storage hardware accelerator
 - ParaBit [Gao+, MICRO'21]: State-of-the-art in-flash processing mechanism

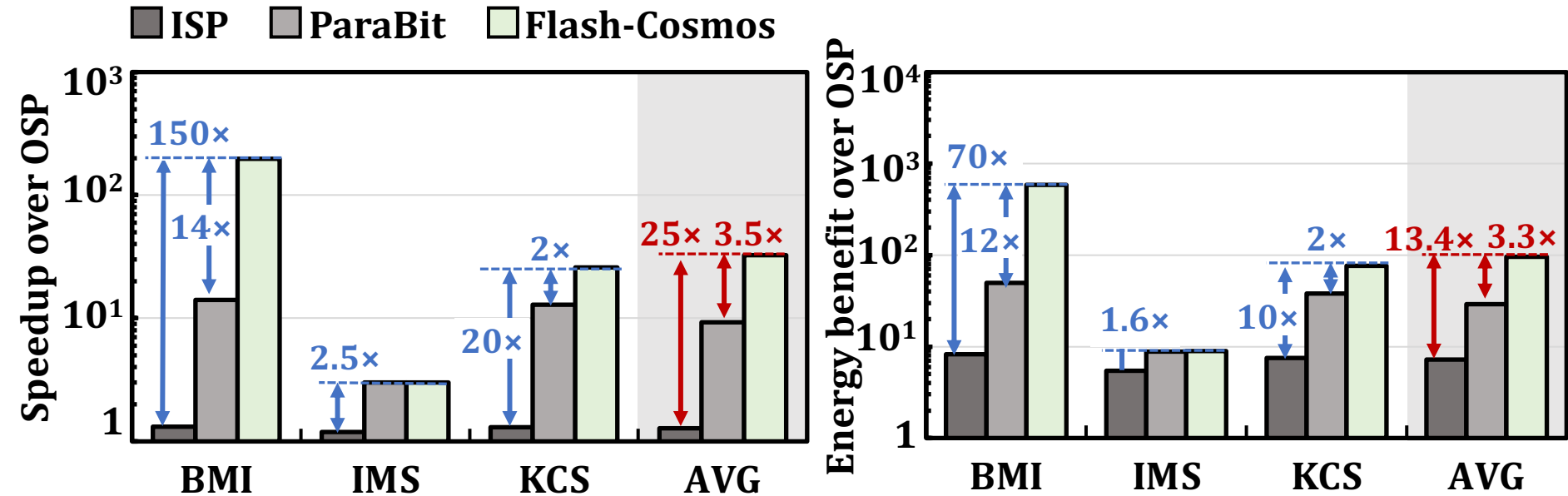
Results: Real-Device Characterization

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

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

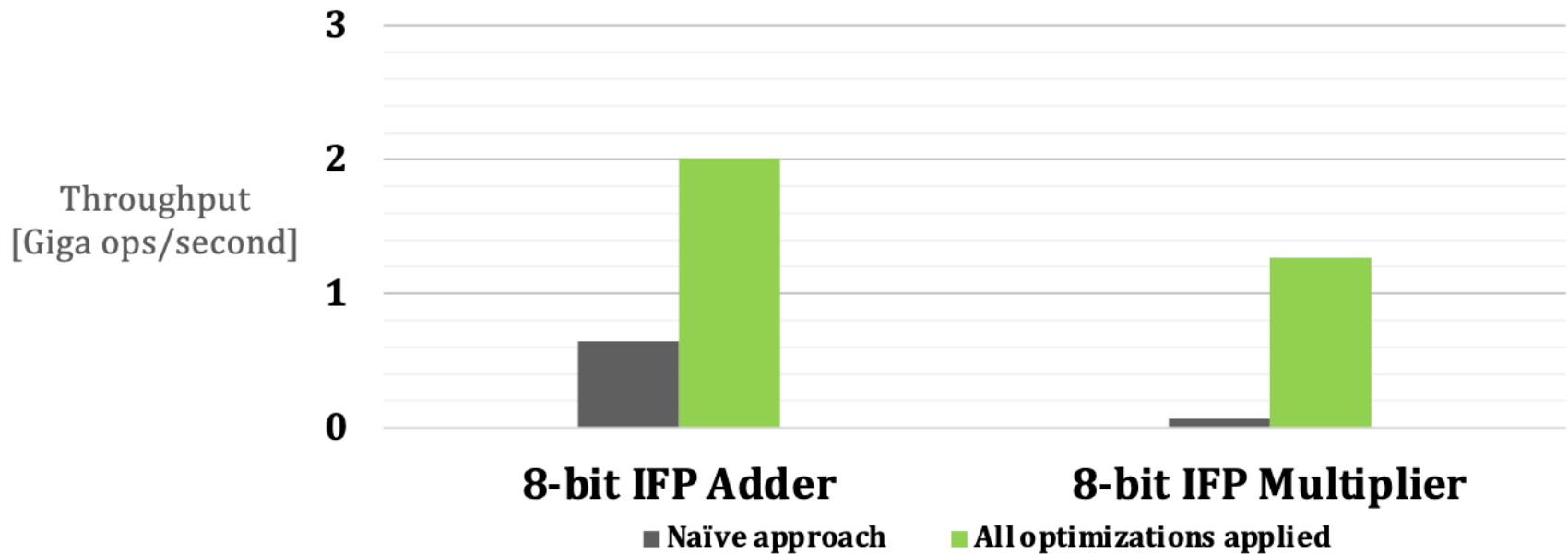
Jisung Park^{§∇} Roknoddin Azizi[§] Geraldo F. Oliveira[§] Mohammad Sadrosadati[§]
Rakesh Nadig[§] David Novo[†] Juan Gómez-Luna[§] Myungsuk Kim[‡] Onur Mutlu[§]

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

Complex In-Flash Arithmetic: Summary

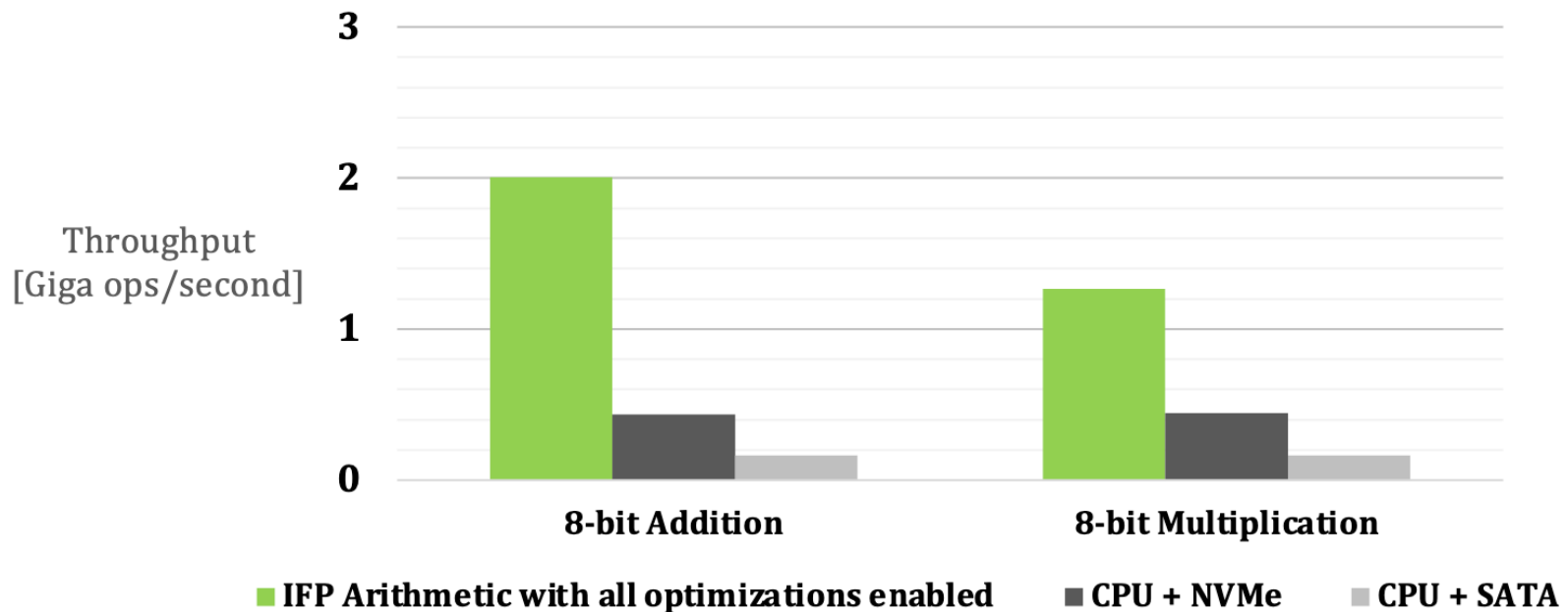
- **Motivation:**
 - Prior **IFP** techniques do not accelerate complex arithmetic operations
 - Arithmetic operations require handling of intermediate results
 - Storing intermediate results reduces flash cell lifespan and increases latency overhead
- **Key Goal:** Implement **in-flash arithmetic operations** with minimal effect on the flash cell lifespan
- **Key Ideas:**
 - Combine prior IFP approaches (e.g., Flash-Cosmos, ParaBit) to implement arithmetic operations
 - Logic unrolling to avoid storing the intermediate results
 - Algebraic/disjunctive normal form (DNF) hybrid representation of output bits
 - Exploit Latch Dump operation to store intermediate results in data latches
- **Key Results:** We use an **analytical model** to evaluate IFP performance
 - Speedup of **up to 19.7x** over naïve IFP arithmetic
 - Speedup of **up to 12.3x** over read/write bound CPU
 - **No programming** needed **to perform addition and multiplication**

Comparison Against Naïve Implementation



- Optimized approach outperforms pure Flash-Cosmos by **3.1x for addition** and **19.7x for multiplication**
- Number of **ESP commands** goes from 14 for addition and 154 for multiplication to **zero**

Comparison Against CPU



- Comparison against CPU bound by read/write bandwidth of NVMe/SATA SSD
- IFP with all optimizations enabled outperforms CPU by 4.6x/12.3x for addition and 2.8x/7.7x for multiplication

Processing in Storage: Adoption Challenges

1. Processing **using** Storage
2. Processing **near** Storage

Eliminating the Adoption Barriers

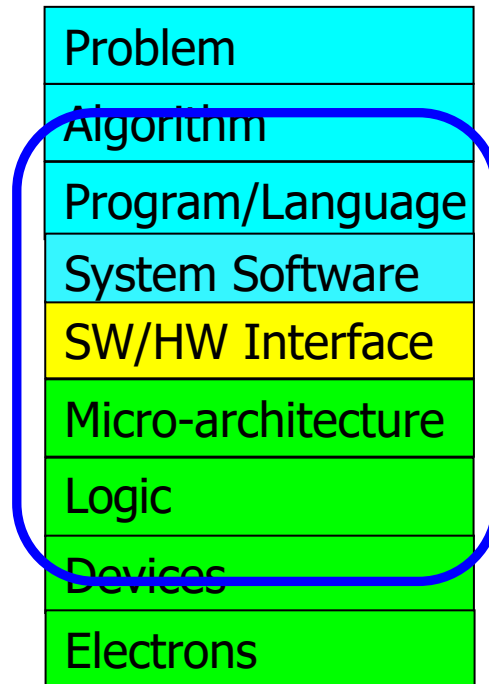
How to Enable Adoption of Processing in Storage

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



We can get there step by step

Automated Framework for Near-Data Processing in Storage Systems

Ongoing Work

Vamanan Arulchelvan & Rakesh Nadig

21 May 2024

Ongoing Work: Summary

- **Background:** Storage devices (SSDs) enable three key types of near-data processing (NDP): (1) **in-storage processing**, (2) **in-flash processing**, and (3) **processing in on-chip DRAM**
- **Problem:** Applications offload functions or code blocks for NDP in SSDs in a **static manner**. There is **no automated approach** to identify (1) what to offload, and (2) where the computation should be performed, within the storage system.
- **Goal:**
 - Perform **profiling** of the workload and **extract accesses to SSD**
 - **Automate identification** and **offloading of function blocks** to different compute units within the SSD in a manner that is transparent to the application.
- **Key Idea:** We propose an **automated end-to-end NDP framework** for SSDs that
 1. To perform **profiling of the workload** and extract accesses (a) **from application to DRAM** (e.g., memory loads and stores), (b) **DRAM to SSD** (via page faults and mmap), and (c) **applications to SSD** (via Direct I/O and file accesses)
 2. **Identify instructions** which can be offloaded to different compute units within the SSD based on (a) the computation to be performed and (b) where the data resides

RawGAINS

A Heterogeneous Storage-Centric Processing System for Raw Signal Genome Analysis

Melina Soysal

Master Thesis Presentation

14.12.2023

Supervisor: Prof. Onur Mutlu, ETH Zürich

Advisor: Dr. Mohammad Sadrosadati, ETH Zürich

Examiner: Prof. Ulf Schlichtmann, TU München

Executive Summary

Motivation

Raw signal genome analysis (RSGA) operates on **large datasets** and is exposed to large **data movement overhead**, especially from storage to host limiting end-to-end performance

Goal

Design a **compact, end-to-end In-Storage Processing system** to improve the performance and energy efficiency of the state-of-the-art RSGA algorithm scalable to larger genomes

Key Contributions

RawGAINS, the first HW-Algorithm Co-Design for RSGA scalable to large genomes by **orchestrating multiple data-centric compute primitives** in a NAND flash-based **SSD**

Key Results

RawGAINS improves the **performance** compared to the state-of-the-art RSGA algorithm on a server-class CPU by **14.9x** and improves **energy consumption** by **170.3x** across five real-world genomic datasets

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

Fundamentally High-Performance **(Data-Centric)** Computing Architectures

Computing Architectures with Minimal Data Movement

Data-Driven (Self-Optimizing) Memory/Storage Architectures

System Architecture Design Today

- Human-driven
 - Humans design the policies (how to do things)
- Many (too) simple, short-sighted policies all over the system
- No automatic data-driven policy learning
- (Almost) no learning: cannot take lessons from past actions

**Can we design
fundamentally intelligent architectures?**

An Intelligent Architecture

- Data-driven
 - Machine learns the “best” policies (how to do things)
- Sophisticated, workload-driven, changing, far-sighted policies
- Automatic data-driven policy learning
- All controllers are intelligent data-driven agents

**We need to rethink design
(of all controllers)**

Self-Optimizing Memory Controllers

- 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.

Self-Optimizing Memory Controllers: A Reinforcement Learning Approach

Engin İpek^{1,2} Onur Mutlu² José F. Martínez¹ Rich Caruana¹

¹Cornell University, Ithaca, NY 14850 USA

²Microsoft Research, Redmond, WA 98052 USA

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](#)]

Officially artifact evaluated as available, reusable and reproducible.



Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera¹

Konstantinos Kanellopoulos¹

Anant V. Nori²

Taha Shahroodi^{3,1}

Sreenivas Subramoney²

Onur Mutlu¹

¹ETH Zürich

²Processor Architecture Research Labs, Intel Labs

³TU Delft

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

Learning-Based Off-Chip Load Predictors

- 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 \(Officially Artifact Evaluated with All Badges\)](#)]

***Officially artifact evaluated as available, reusable and reproducible.
Best paper award at MICRO 2022.***



Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction

Rahul Bera¹ Konstantinos Kanellopoulos¹ Shankar Balachandran² David Novo³
Ataberk Olgun¹ Mohammad Sadrosadati¹ Onur Mutlu¹

¹ETH Zürich ²Intel Processor Architecture Research Lab ³LIRMM, Univ. Montpellier, CNRS

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

Self-Optimizing Storage Controllers

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 |

¹ETH Zürich

²Eindhoven University of Technology

³LIRMM, Univ. Montpellier, CNRS

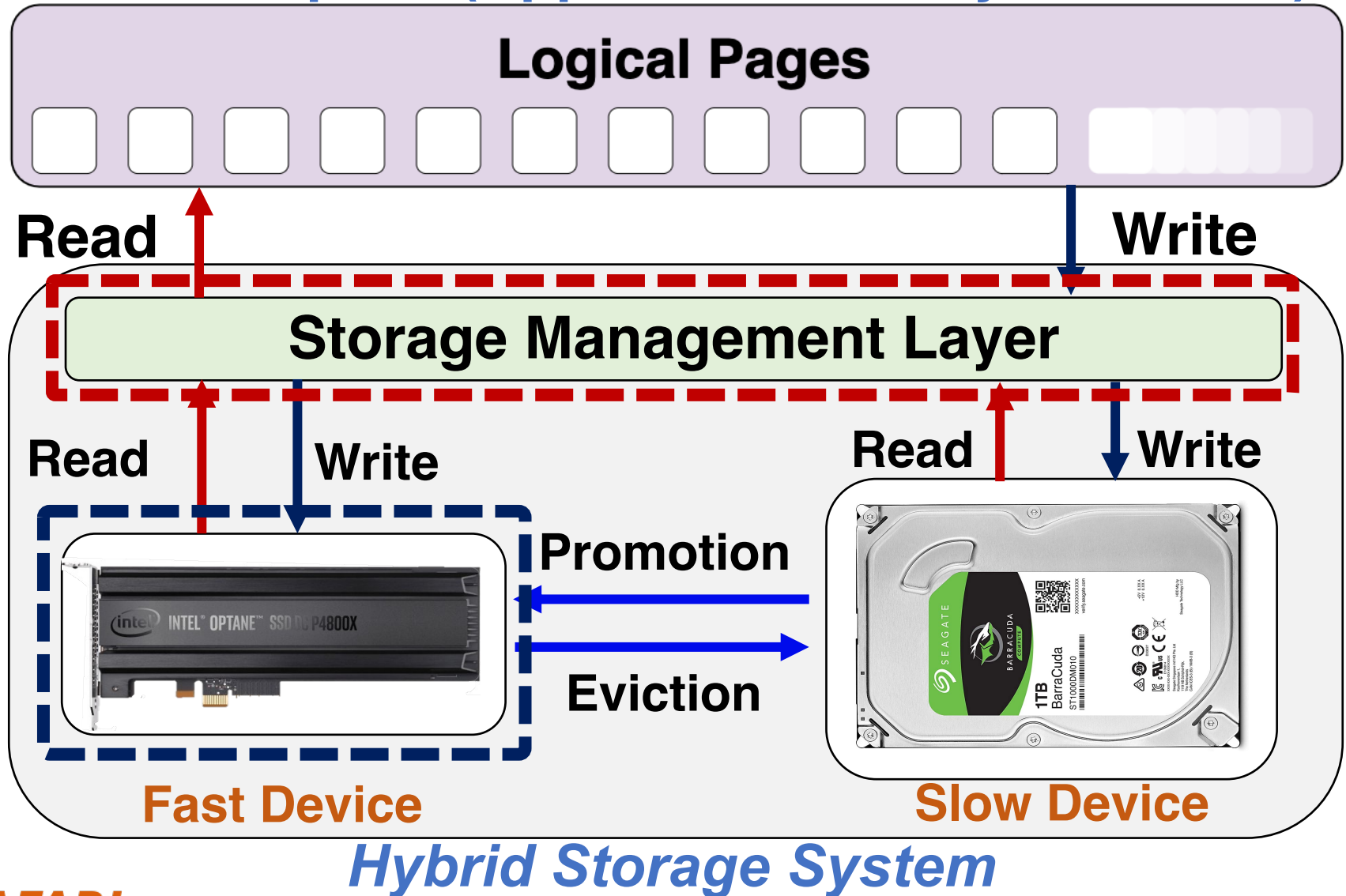
Sibyl:

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

Gagandeep Singh, Rakesh Nadig, Jisung Park,
Rahul Bera, Nastaran Hajinazar, David Novo,
Juan Gómez Luna, Sander Stuijk, Henk Corporaal,
Onur Mutlu

Hybrid Storage System Basics

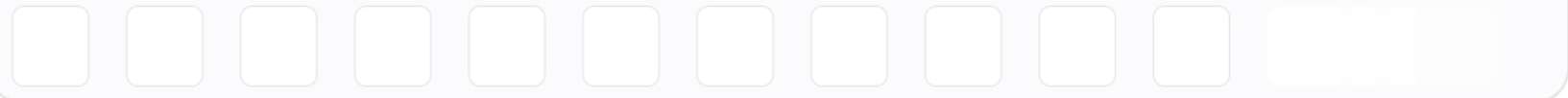
Address Space (Application/File System View)



Hybrid Storage System Basics

Logical Address Space (Application/File System View)

Logical Pages



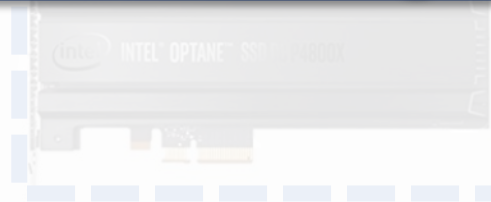
Read



Write



Performance of a hybrid storage system
highly depends on the ability of the
storage management layer



Hybrid Storage System

Key Shortcomings in Prior Techniques

We observe **two key shortcomings** that significantly limit the performance benefits of prior techniques

1. Lack of **adaptivity to**:
 - a) Workload changes
 - b) Changes in device types and configuration
2. Lack of **extensibility** to more devices

Our Goal

A **data-placement mechanism**
that can provide:

1. **Adaptivity**, by **continuously learning** and **adapting** to the application and underlying device characteristics
2. **Easy extensibility** to incorporate a wide range of hybrid storage configurations

Our Proposal

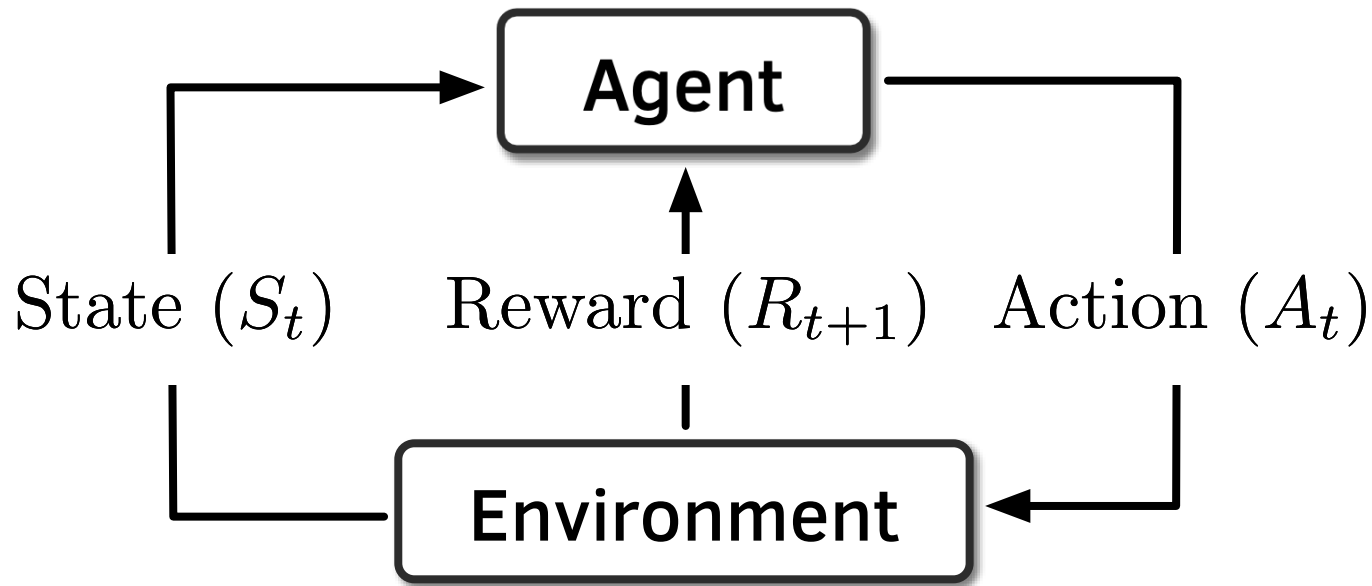


Sibyl

Formulates data placement in
hybrid storage systems as a
reinforcement learning problem

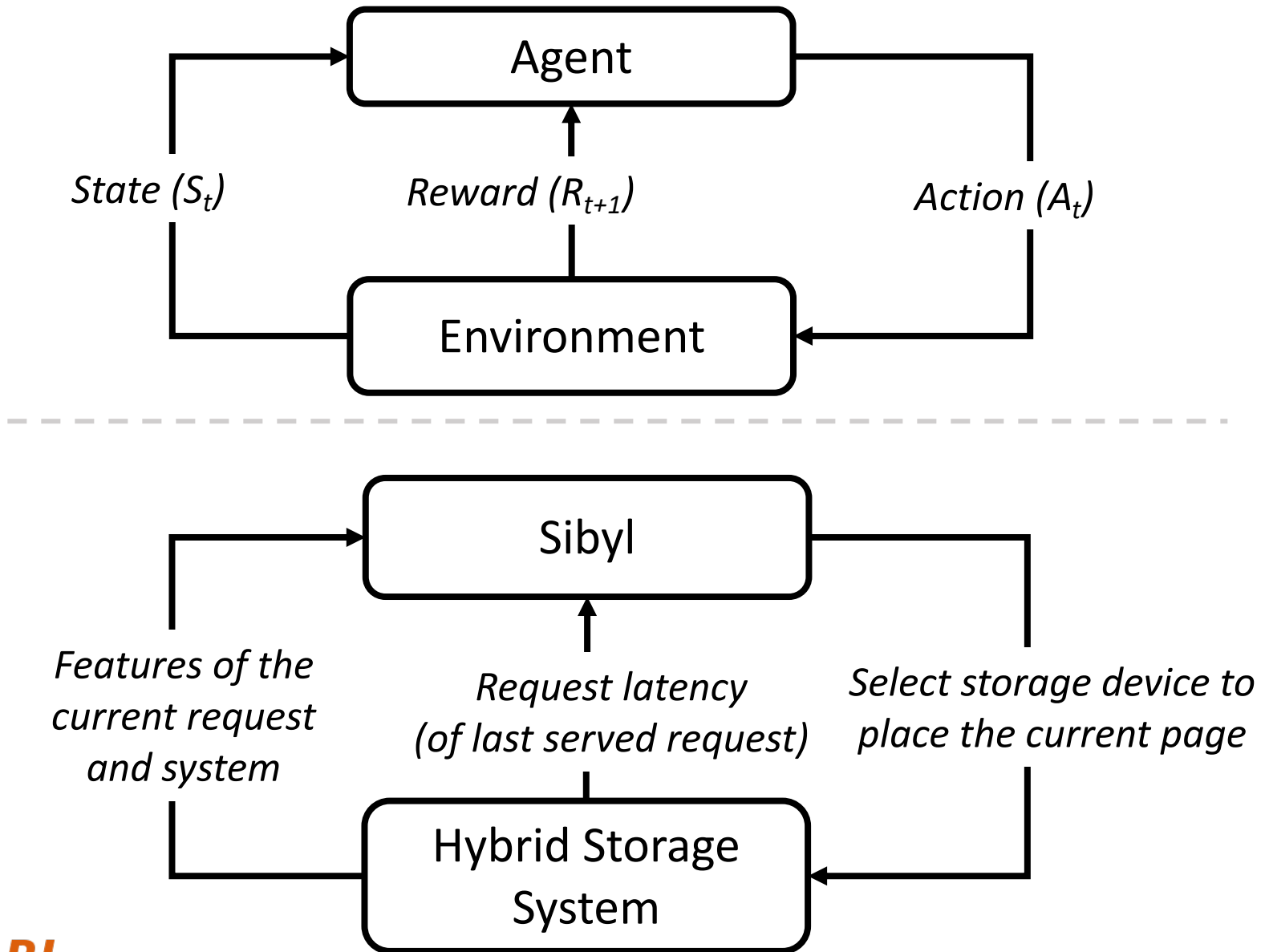
Sybil is an oracle that makes accurate prophecies
<https://en.wikipedia.org/wiki/Sibyl>

Basics of Reinforcement Learning (RL)

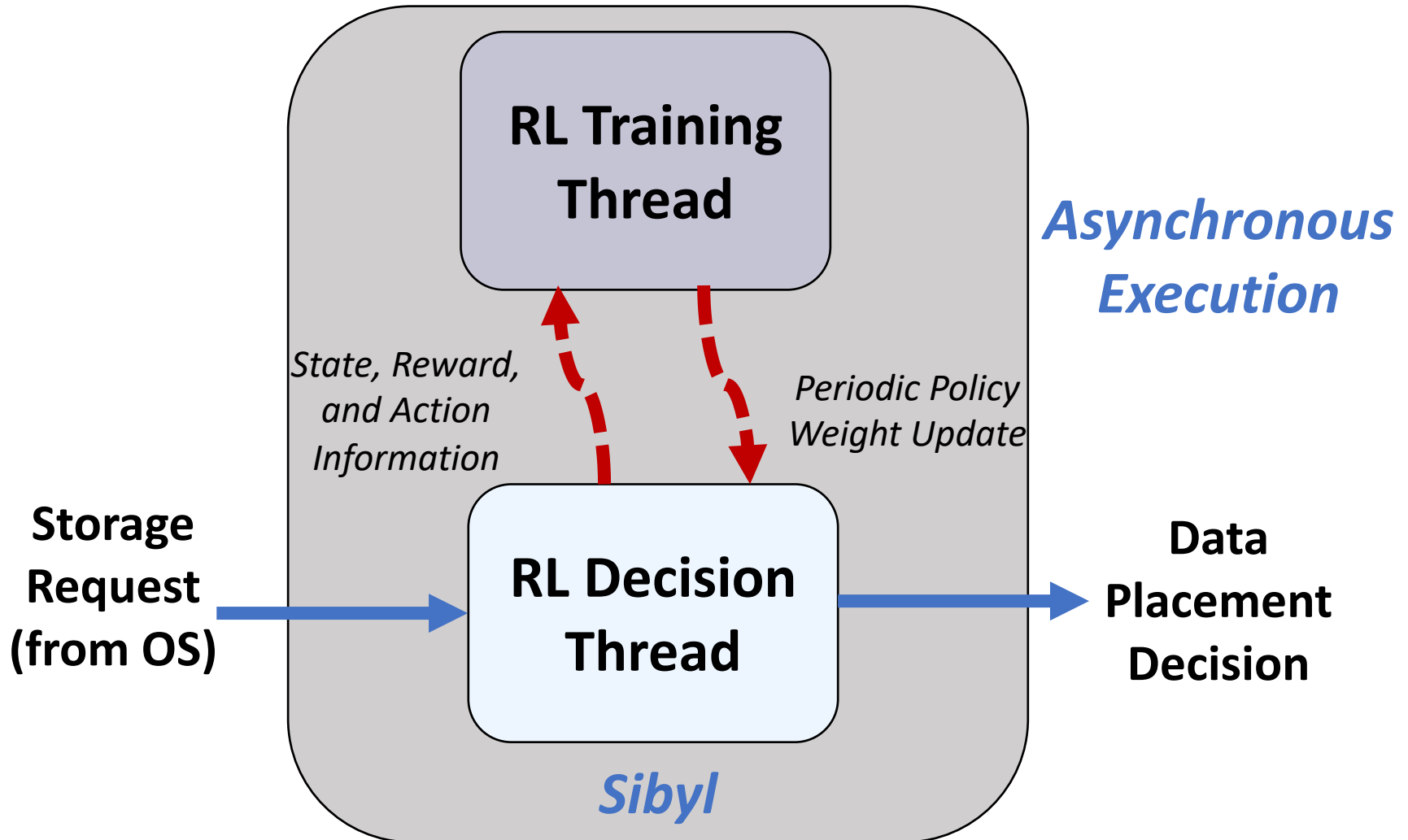


Agent learns to take an **action** in a given **state** to maximize a numerical **reward**

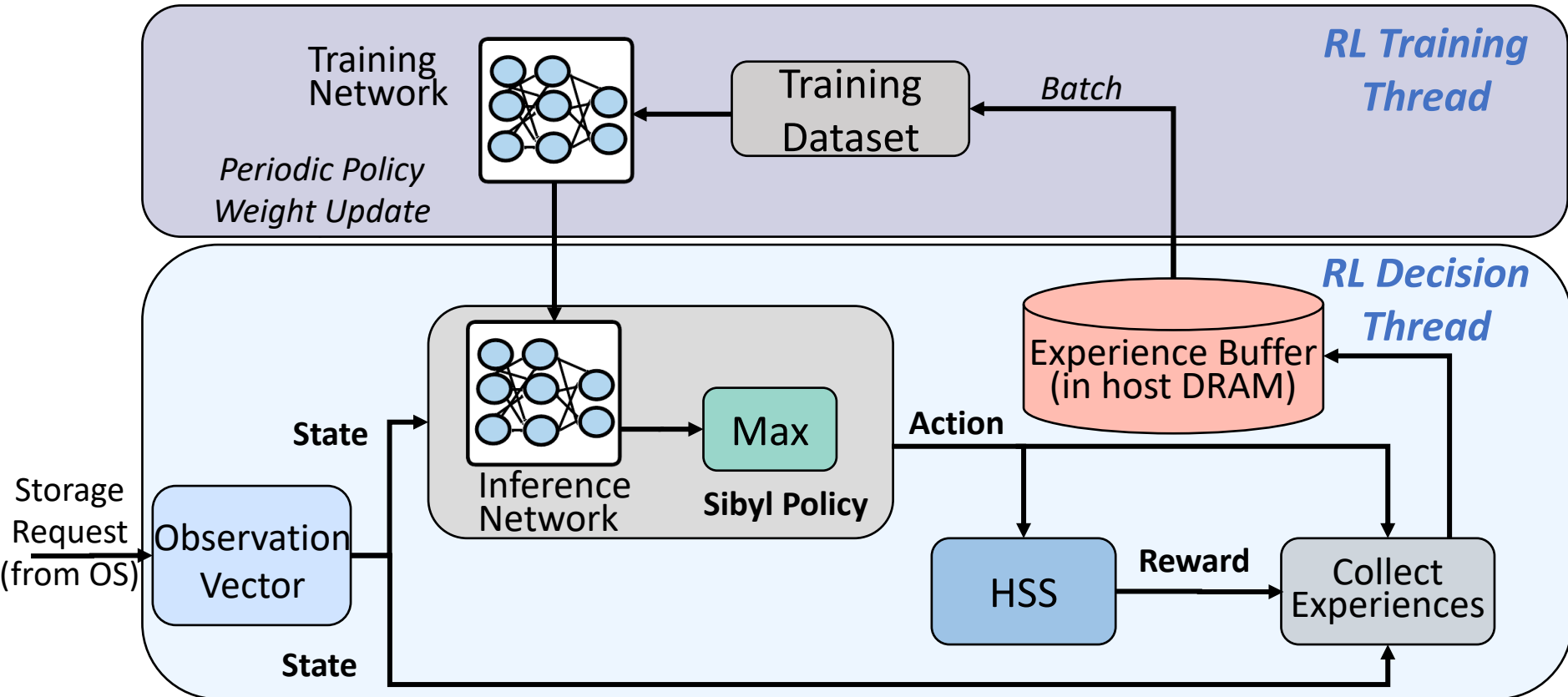
Formulating Data Placement as RL



Sibyl Execution



Sibyl Design: Overview



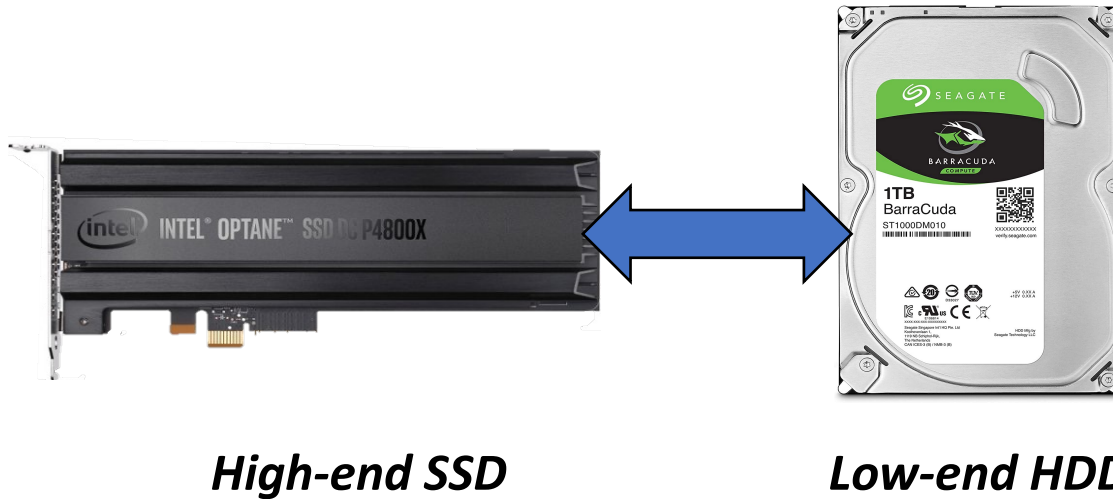
Evaluation Methodology (1/3)

- **Real system** with various HSS configurations
 - Dual-hybrid and tri-hybrid systems

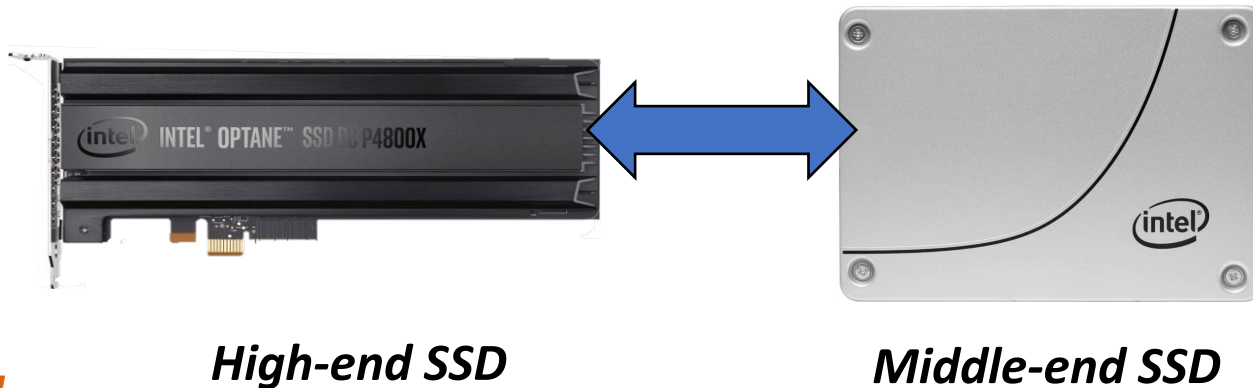


Evaluation Methodology (2/3)

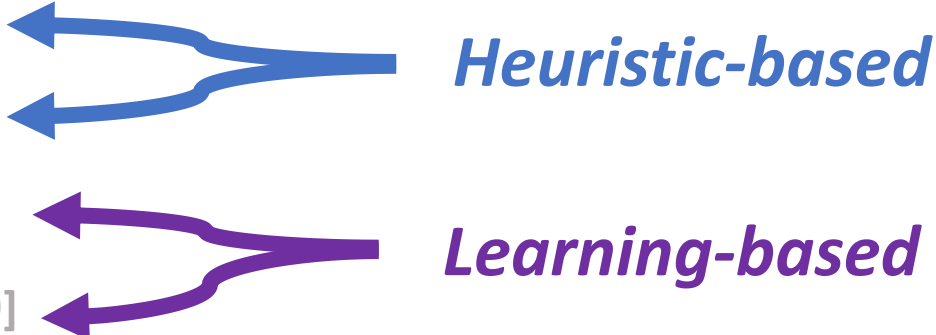
Cost-Oriented HSS Configuration



Performance-Oriented HSS Configuration



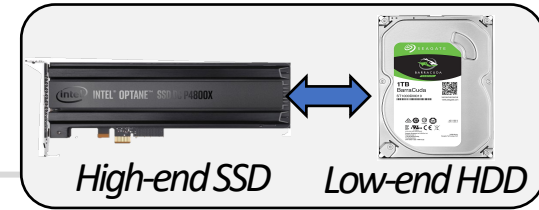
Evaluation Methodology (3/3)

- **18 different workloads** from:
 - MSR Cambridge and Filebench Suites
- **Four** state-of-the-art data placement baselines:
 - CDE [Matsui+, Proc. IEEE'17]
 - HPS [Meswani+, HPCA'15]
 - Archivist [Ren+, ICCD'19]
 - RNN-HSS [Doudali+, HPDC'19]

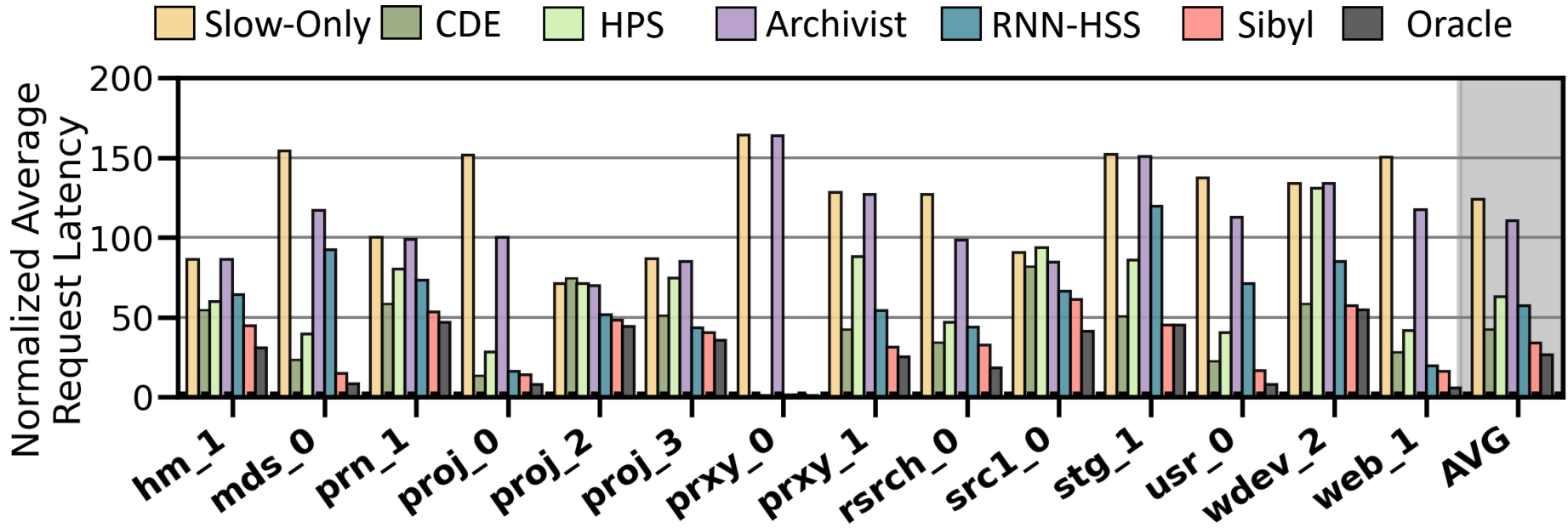
Heuristic-based

Learning-based

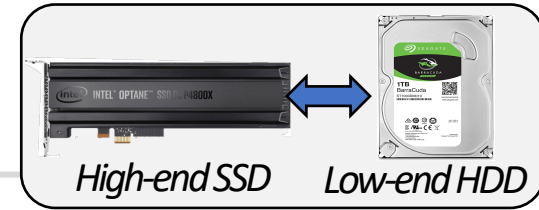
Performance Analysis



Cost-Oriented HSS Configuration

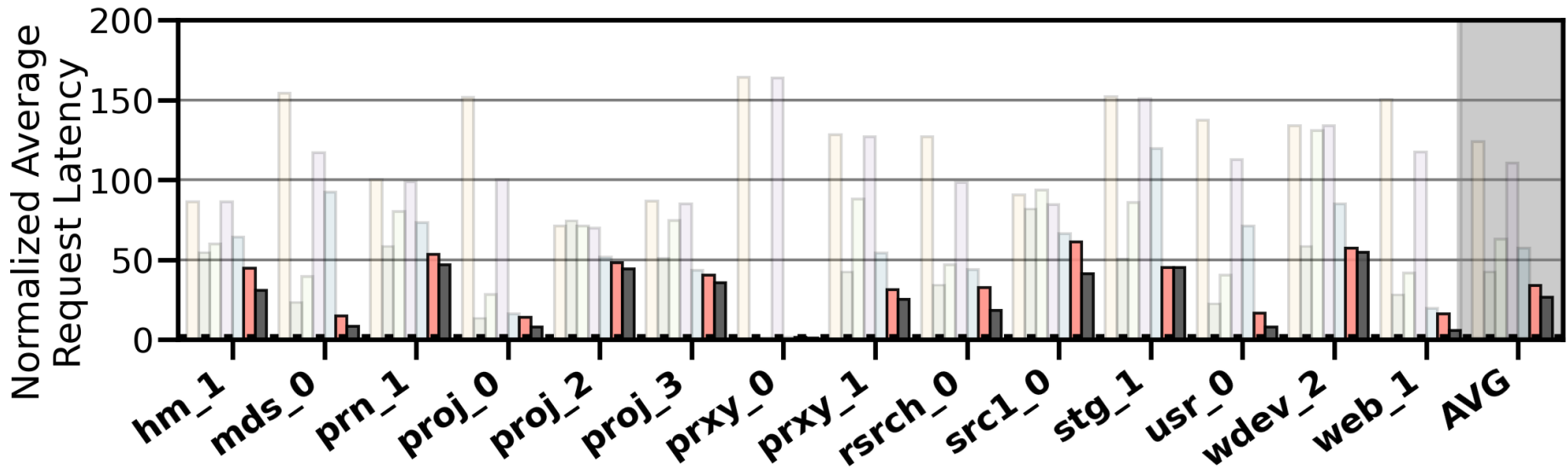


Performance Analysis



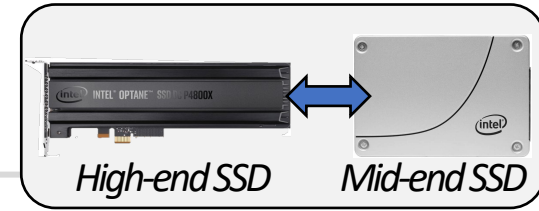
Cost-Oriented HSS Configuration

Slow-Only CDE HPS Archivist RNN-HSS Sibyl Oracle

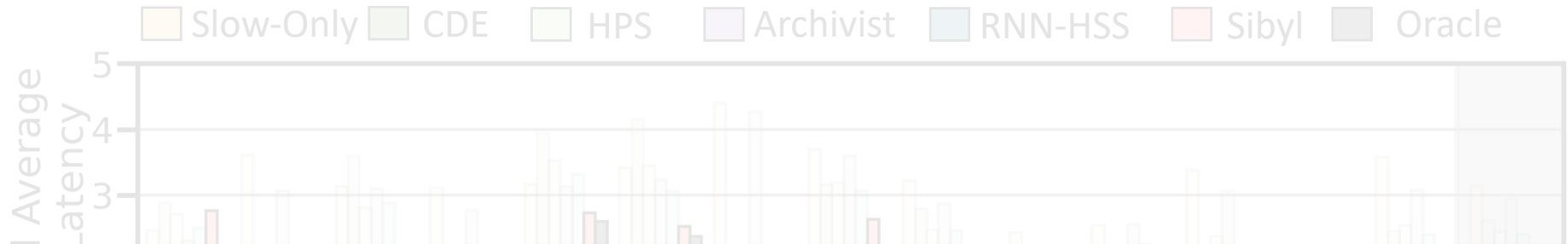


Sibyl consistently **outperforms all the baselines**
for all the workloads

Performance Analysis

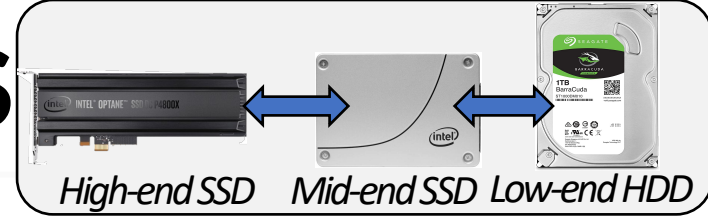


Performance-Oriented HSS Configuration



Sibyl achieves **80% of the performance of an oracle policy** that has complete knowledge of future access patterns

Performance on Tri-HSS



Extending Sibyl for **more devices**:

1. Add a new action

Sibyl **outperforms** the state-of-the-art data placement policy by **48.2% in a real tri-hybrid system**

Sibyl reduces the system architect's burden by providing **ease of extensibility**

Sibyl: Summary

- **We introduced Sibyl**, the first reinforcement learning-based data placement technique in hybrid storage systems that provides
 - **Adaptivity**
 - **Easily extensibility**
 - **Ease of design and implementation**
- **We evaluated Sibyl** on **real systems** using many different workloads
 - In a tri-HSS configuration, Sibyl **outperforms** the state-of-the-art-data placement policy by **48.2%**
 - Sibyl achieves **80% of the performance** of an oracle policy with a storage overhead of only **124.4 KiB**

Data-Driven (Self-Optimizing) Computing Architectures

Sibyl Paper, Slides, Videos [ISCA 2022]

- 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 Singh¹ Rakesh Nadig¹ Jisung Park¹ Rahul Bera¹ Nastaran Hajinazar¹
David Novo³ Juan Gómez-Luna¹ Sander Stuijk² Henk Corporaal² Onur Mutlu¹

¹ETH Zürich

²Eindhoven University of Technology

³LIRMM, Univ. Montpellier, CNRS

Janus

**Coordinated Data Placement and Data Migration
in Hybrid Storage Systems Using
Multi-Agent Online Reinforcement Learning**

Ongoing Work

Janus: Summary



First work to use multi-agent online reinforcement learning to optimize data management in HSS



Improves performance by 26%/23% over the best-performing prior work on performance-optimized/cost-optimized HSS



Achieves 64%/62% of an Oracle's performance on a performance-optimized/cost-optimized HSS

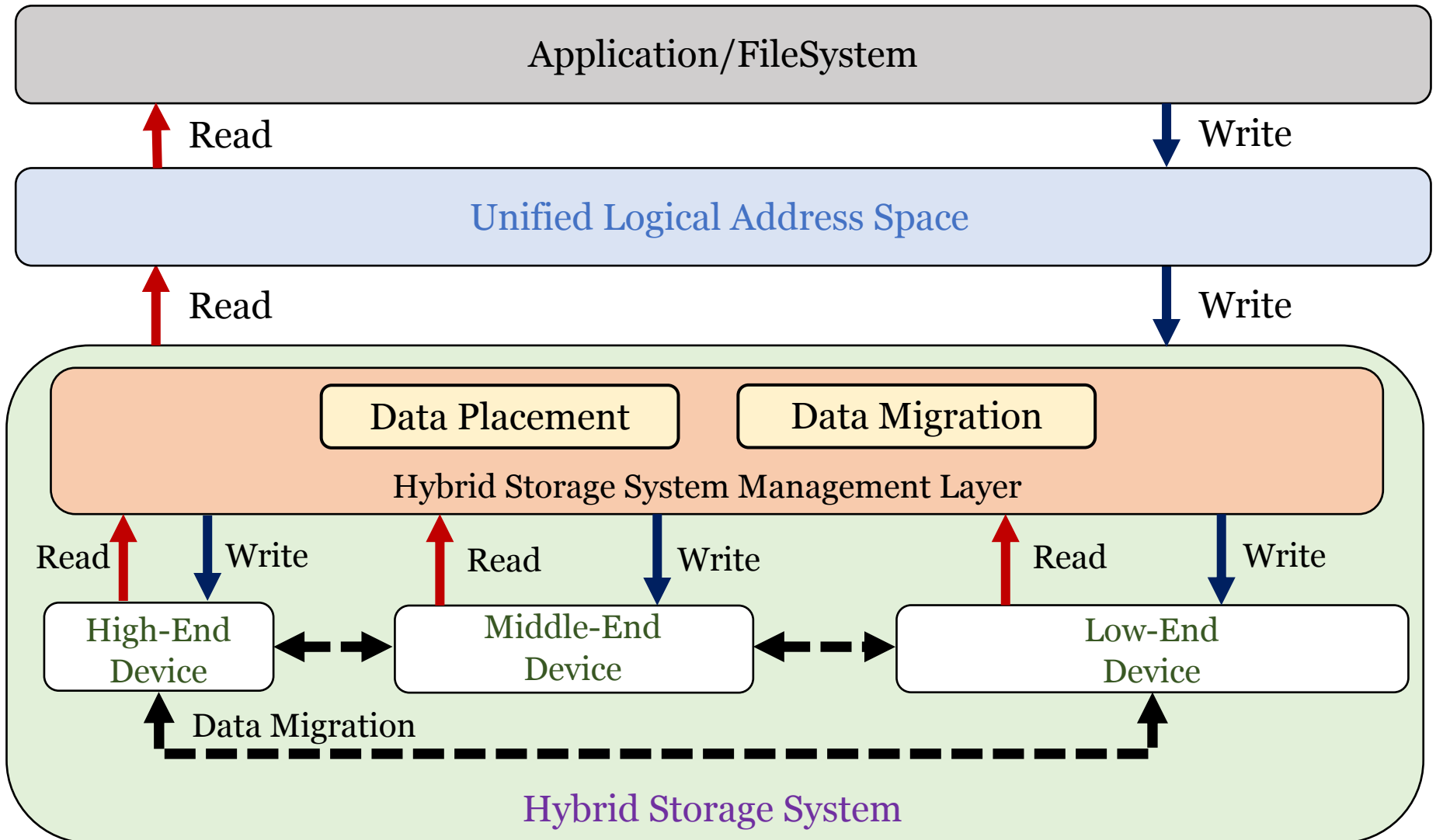


Low inference latency and storage overheads

Executive Summary

- **Background:** A hybrid storage system (HSS) uses multiple different storage devices to provide high and scalable storage capacity at high performance
- **Problem:** Prior data placement and data migration policies (1) lack a **holistic data management policy**, (2) Lack **adaptivity** to: (i) workload changes, and (ii) changes in device types and configurations, and (iii) Lack of **extensibility** to more devices
- **Goal:** Design a holistic data management technique that provides:
 - **Combined optimization** of both data placement and data migration policies
 - **Coordination** between data placement and data migration
 - **Adaptivity**, by **continuously learning and adapting** to the **application and underlying device characteristics**
 - **Easy extensibility** to incorporate a wide range of hybrid storage configurations
- **Contribution:** Janus, the first multi-agent online reinforcement learning-based data management technique in HSS that:
 - Performs **combined optimization** of both data placement and data migration policies
 - **Coordinates** between data placement and data migration
 - Provides **adaptivity** to changing workload demands and underlying device characteristics
 - Can **easily extend** to any number of storage devices
- **Key Results:** Evaluate on **real systems** using fifteen data-intensive workloads
 - Janus **improves performance by 26%/22%** compared to the best previous data placement technique in performance-optimized/cost-optimized dual-HSS configuration
 - In a tri-HSS configuration, Janus outperforms the state-of-the-art policy by **31%**
 - Janus achieves **64%/62% of an Oracle's performance** on a performance-optimized/cost-optimized HSS with a storage overhead of only **151.2 KiB**

Hybrid Storage System Basics



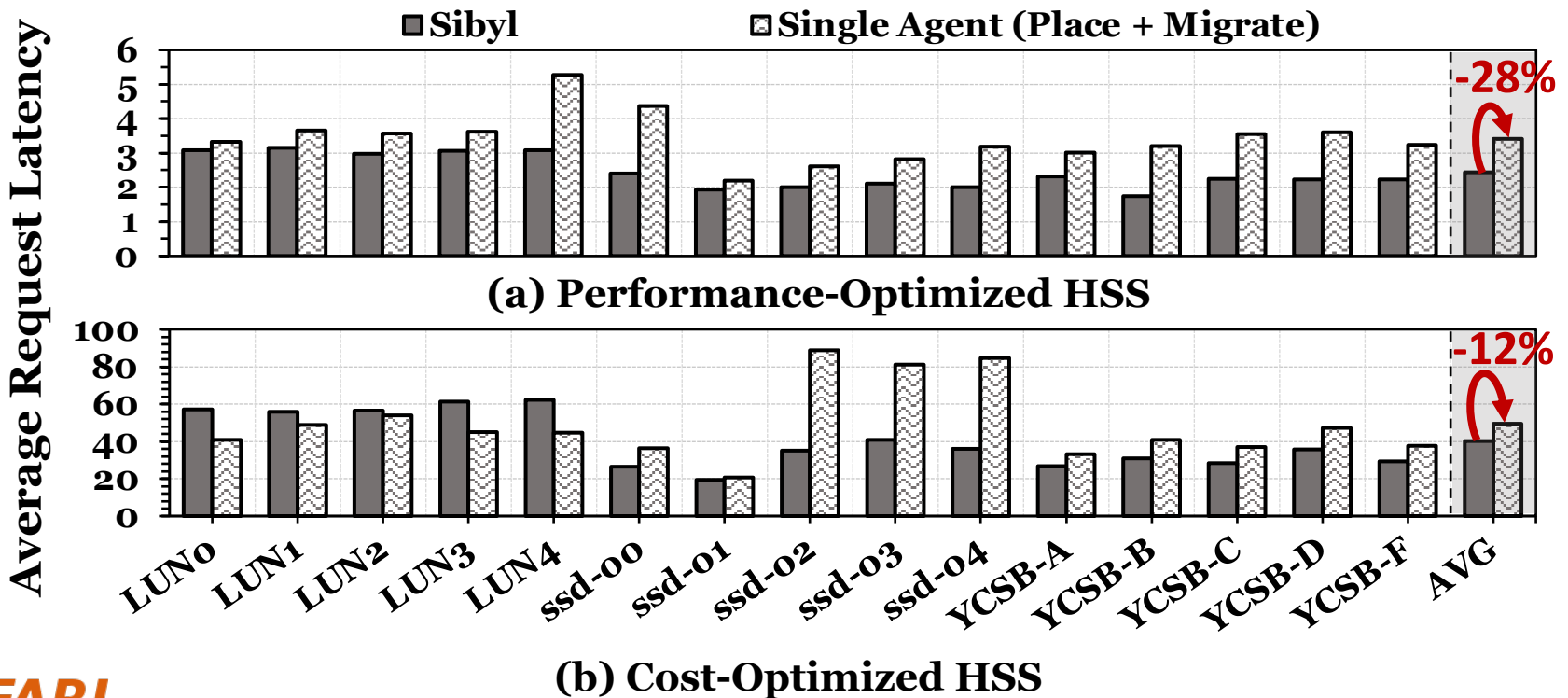
Our Goal

A **holistic data management mechanism** that can provide:

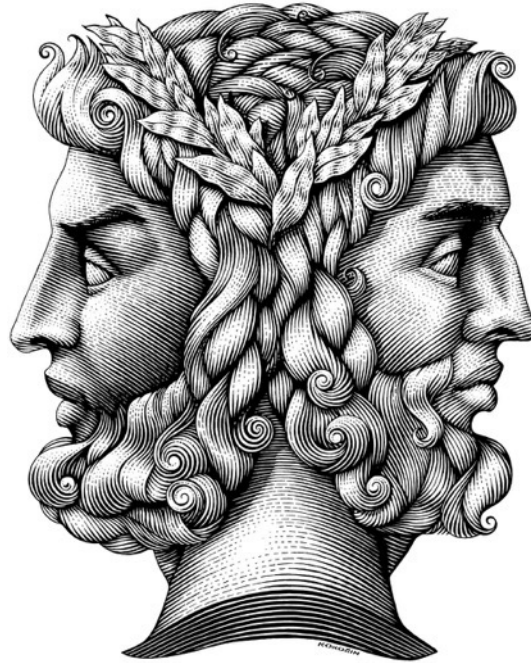
1. **Combined optimization** of both data placement and data migration policies
2. **Coordination** between data placement and data migration
3. **Adaptivity**, by **continuously learning** and **adapting** to the application and underlying device characteristics
4. **Easy extensibility** to incorporate a wide range of hybrid storage configurations

Need for a Multi-Agent RL Technique

- An RL-based technique can adapt to changes in workload and HSS configurations
- A **single RL agent** based technique **cannot optimize multiple policies** (e.g., data placement and data migration) concurrently because the goals of these policies are different



Our Proposal

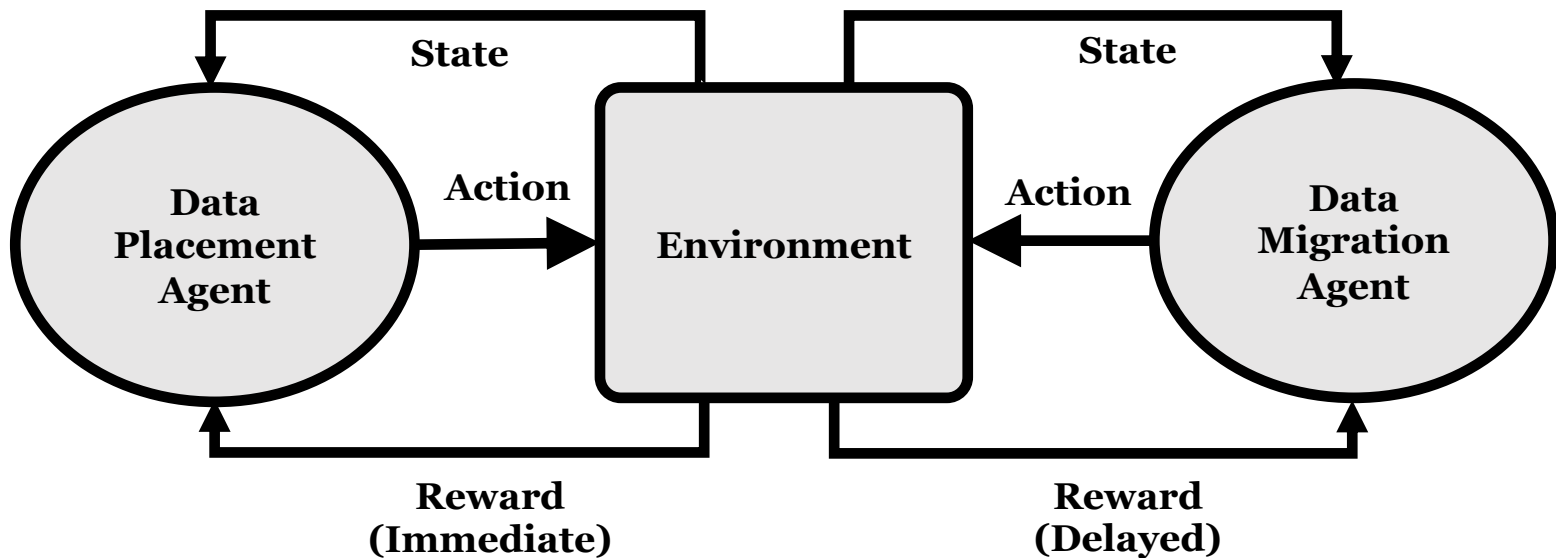
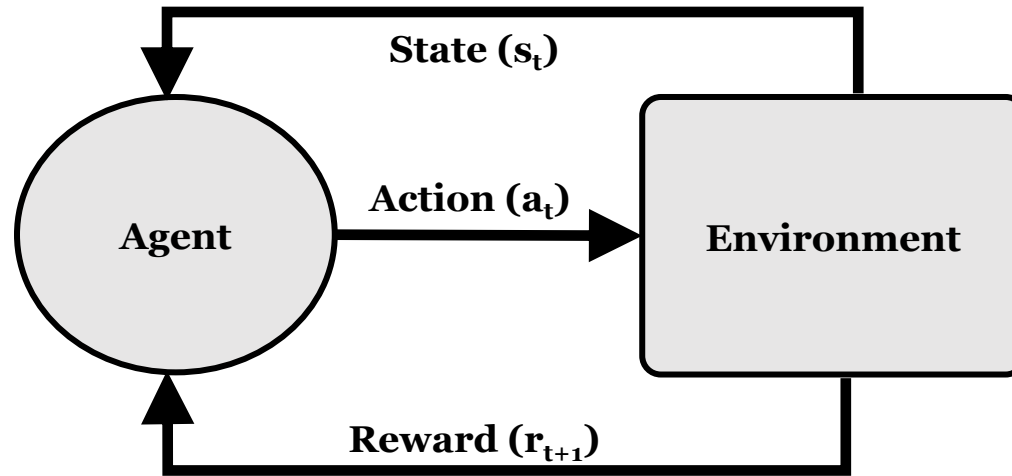


Janus

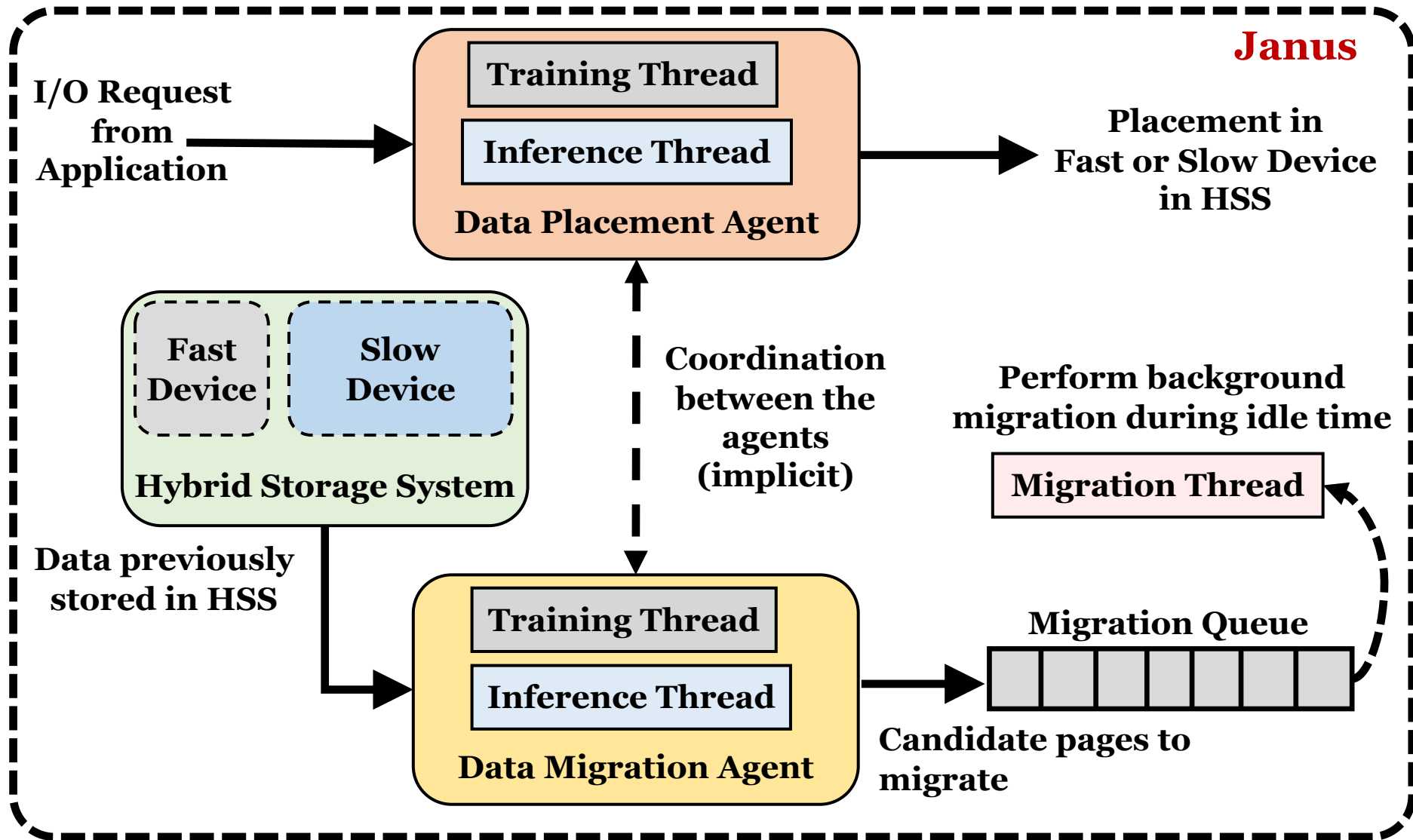
A holistic data management technique
that uses
multi-agent online reinforcement learning

Janus is the god of beginnings, transitions and endings
<https://en.wikipedia.org/wiki/Janus>

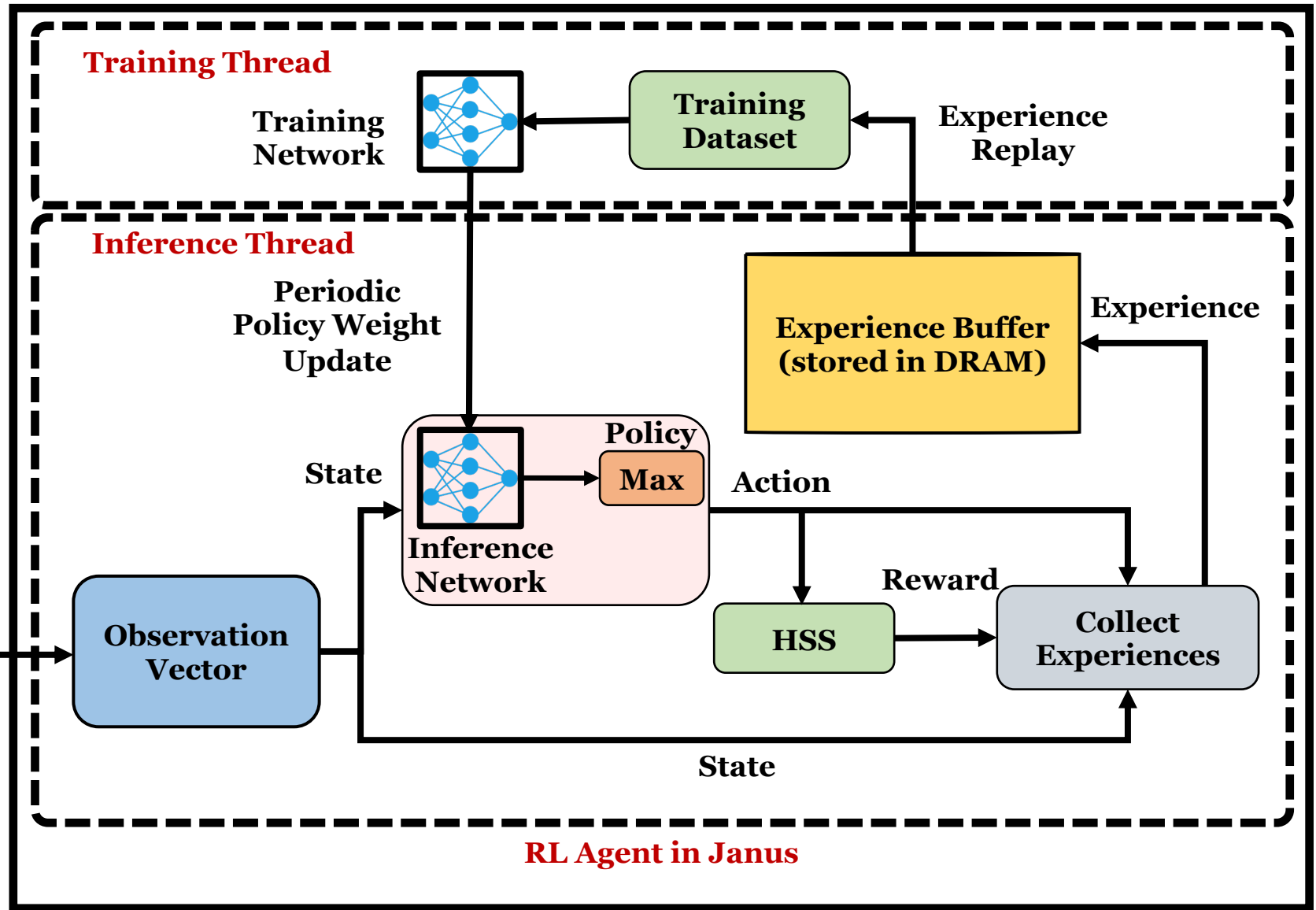
Formulating Data Management as RL



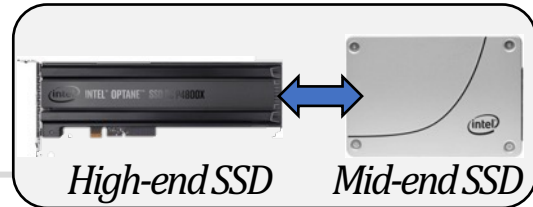
Janus: Design



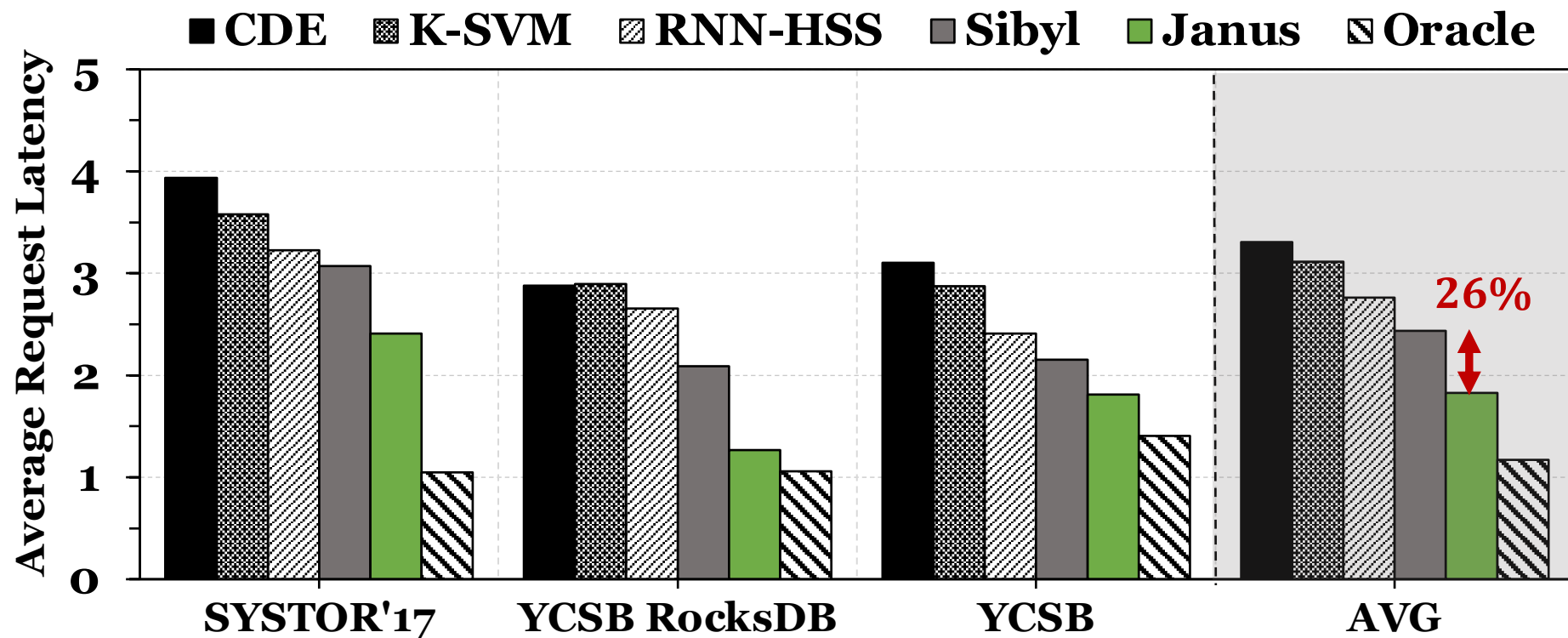
Janus: RL Agent Design



Performance Analysis (1/2)



Performance-Optimized HSS

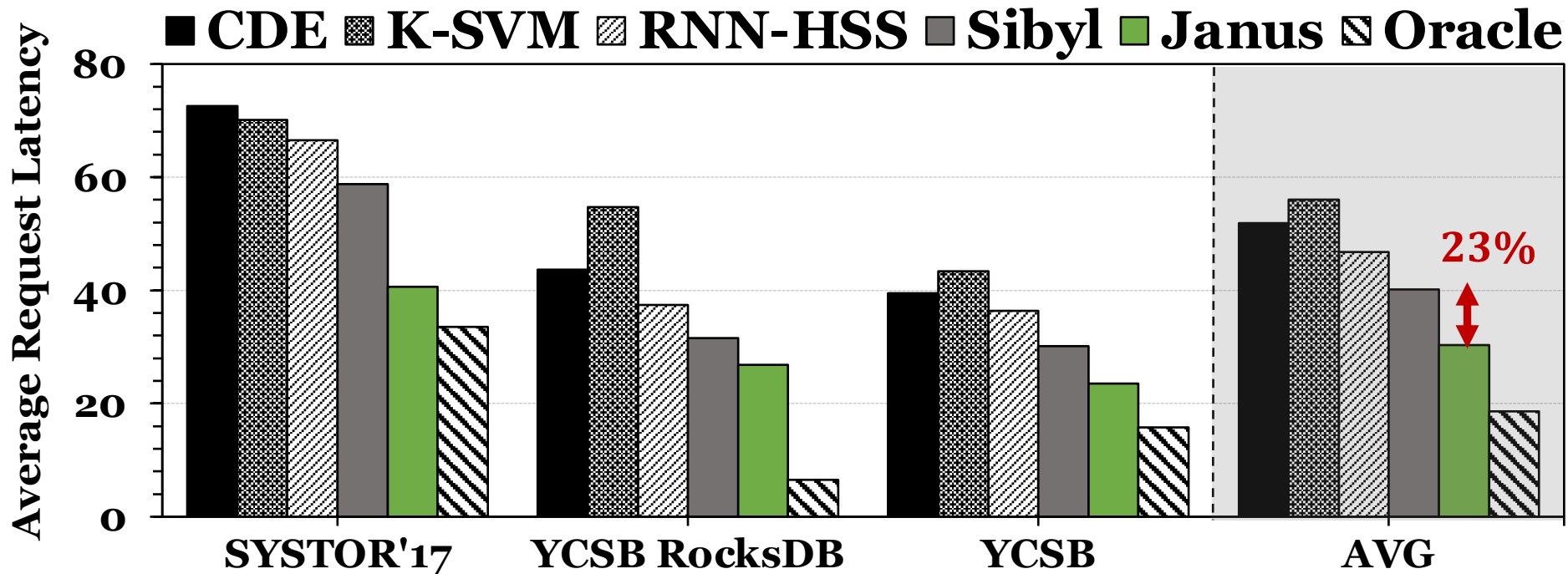


Janus achieves 64% of an Oracle's performance on average across different workloads

Performance Analysis (2/2)

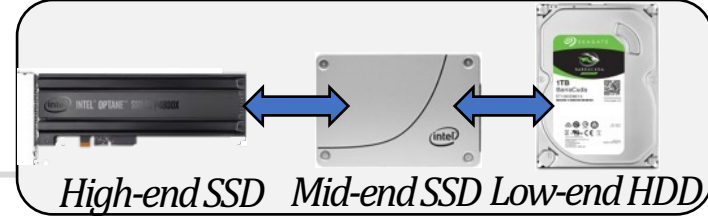


Cost-Optimized HSS



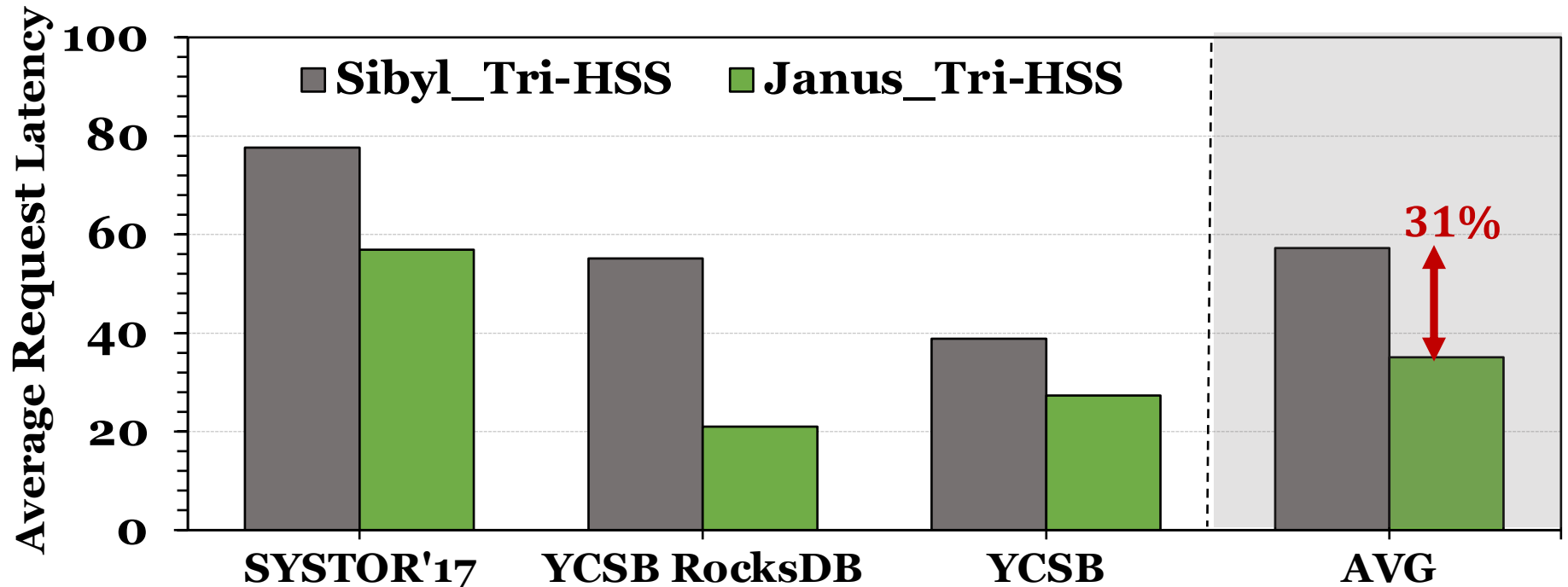
Janus achieves 62% of an Oracle's performance on average across all workloads

Performance on Tri-HSS



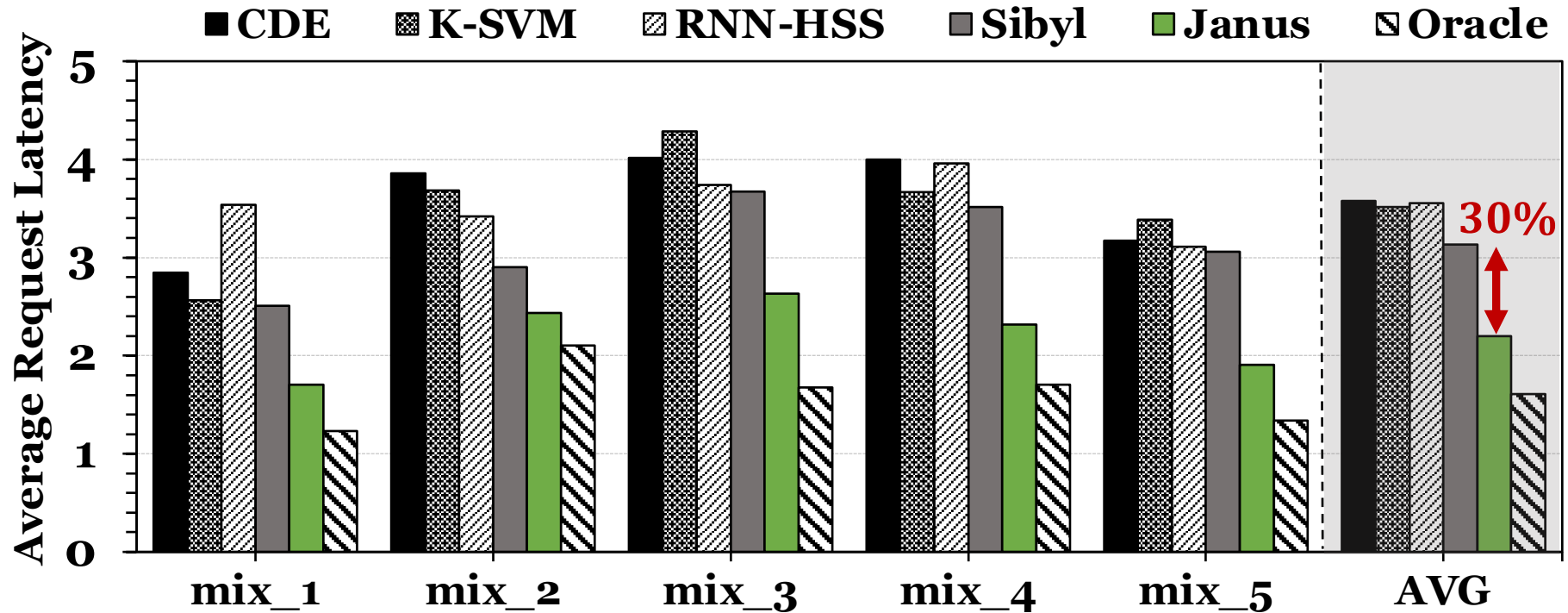
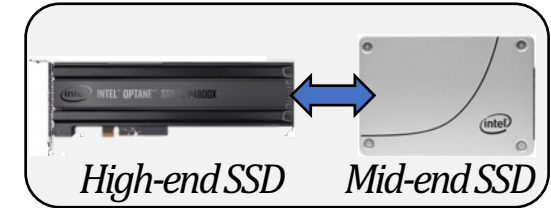
Extending Janus for **more devices**:

1. **Add a new action** in each agent
2. **Add the remaining capacity** of the new device as a state feature for each agent



Multi-Programmed Workloads (1/2)

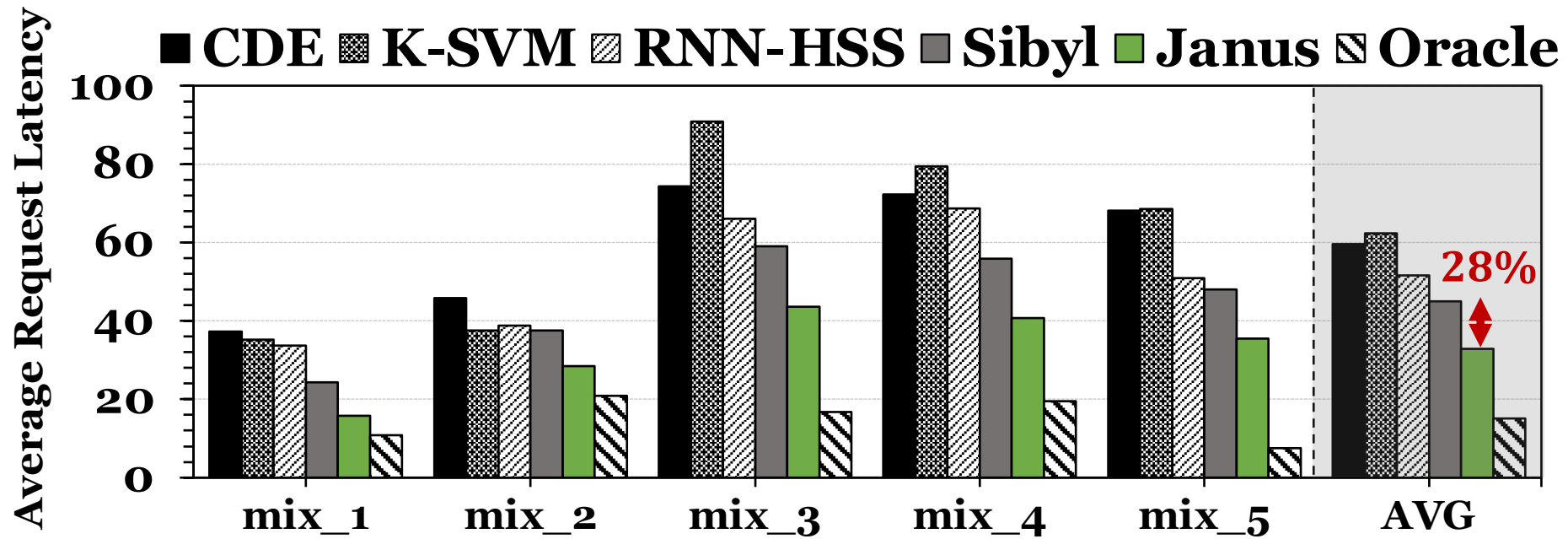
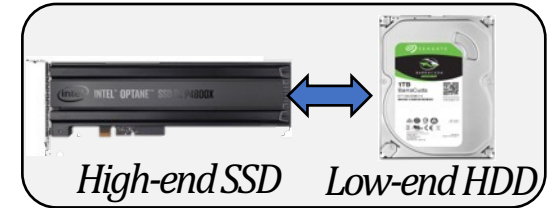
Performance-Optimized HSS



Janus achieves 73% of Oracle's performance on average across different workloads

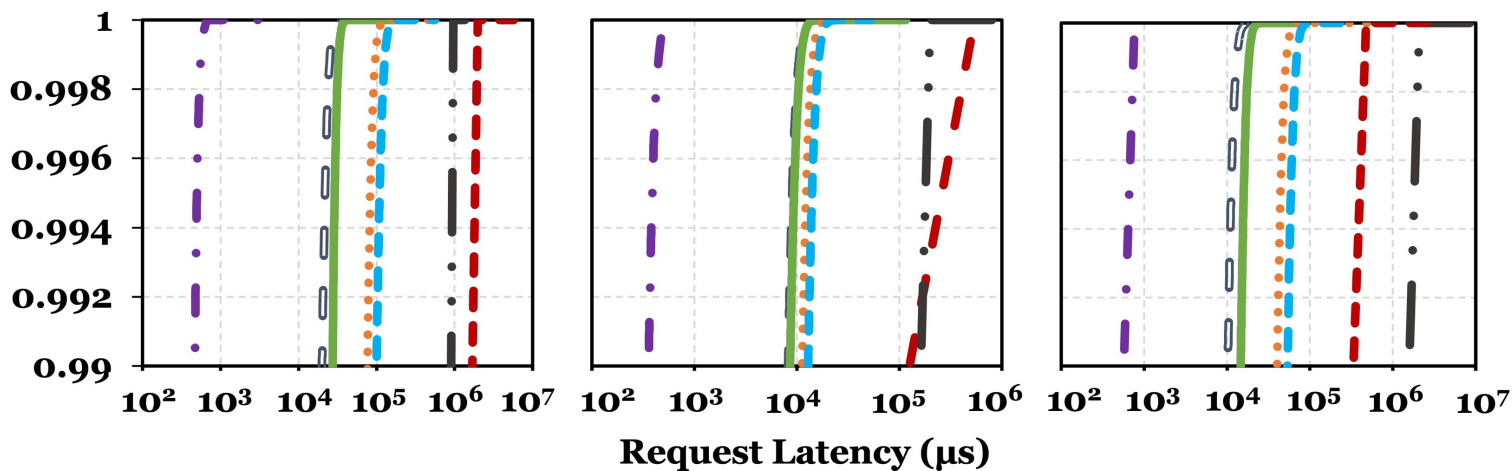
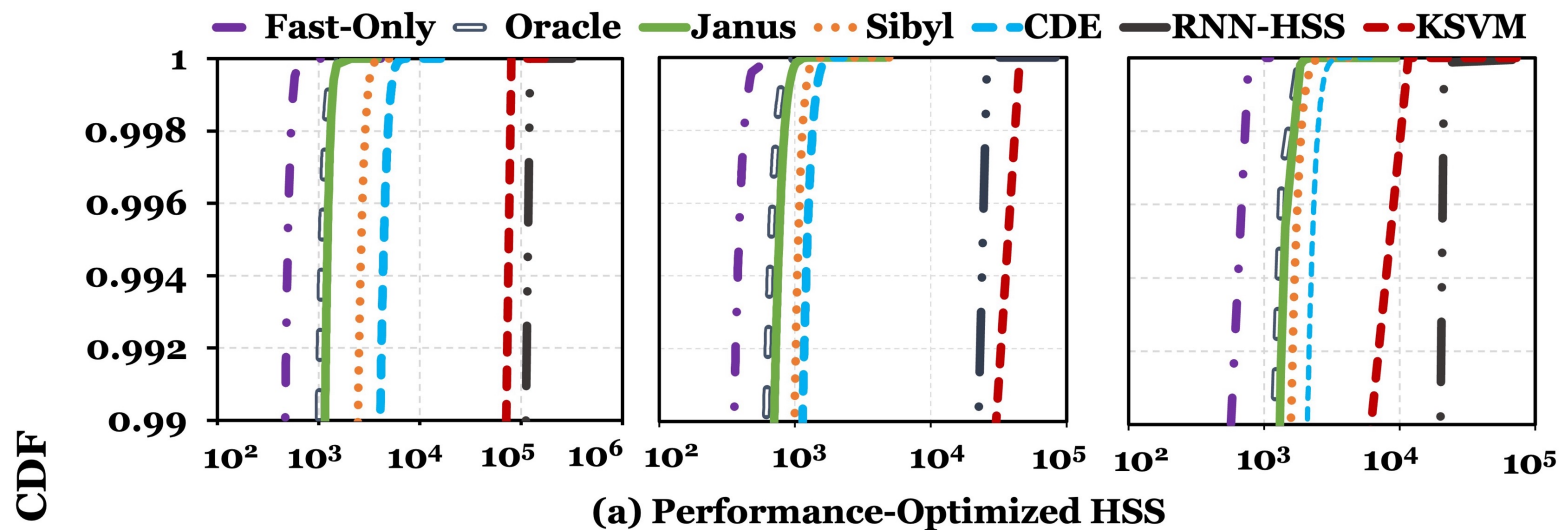
Multi-Programmed Workloads (2/2)

Cost-Optimized HSS

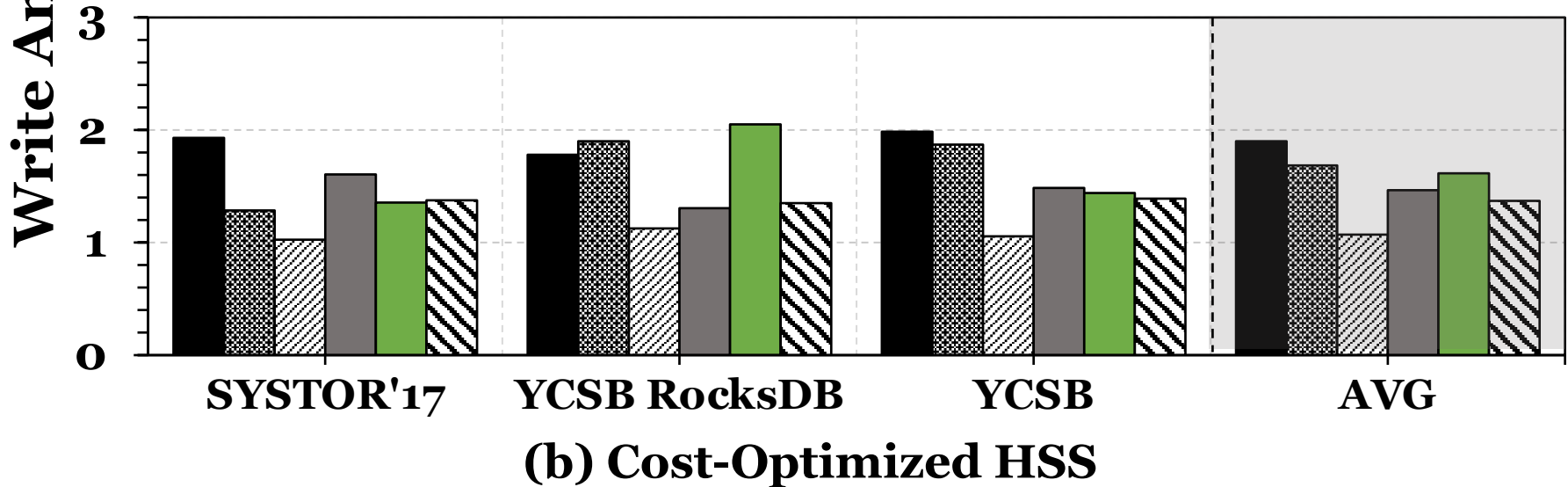
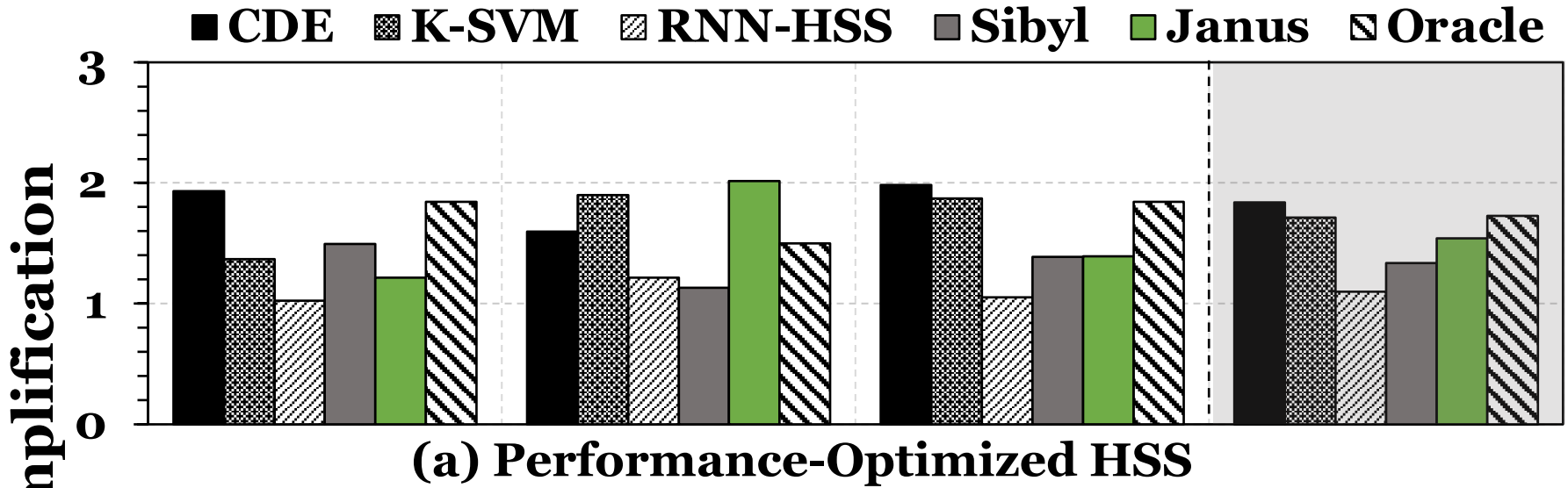


Janus achieves 46% of Oracle's performance on average across different workloads

Tail Latency



Impact on Device Lifetimes



Impact on Device Lifetimes

■ CDE ■ K-SVM ■ RNN-HSS ■ Sibyl ■ Janus ■ Oracle

3

To improve performance, Janus migrates more data in **read-intensive workloads**, which causes **higher write amplification**

(a) Performance-Optimized HSS

3

Janus migrates less data in **write-intensive workloads** as data is moved during updates from host, which causes **lower write amplification**

(b) Cost-Optimized HSS

Janus: Overhead Analysis

- **151.2 KiB** of total storage cost
 - Experience buffer, inference and training network
- **48-bit** metadata overhead per page for state features
- Inference Latency **$\sim 10\text{ns}$**
- Training Latency **$\sim 2\mu\text{s}$ per training step**



Small storage overhead



Small inference overhead

Janus: Summary



First work to use multi-agent online reinforcement learning to optimize data management in HSS



Improves performance by 26%/23% over the best-performing prior work on performance-optimized/cost-optimized HSS



Achieves 64%/62% of an Oracle's performance on a performance-optimized/cost-optimized HSS



Low inference latency and storage overheads

Future Work

- Adding more agents to optimize other policies such as data prefetching across storage devices.
- Applicability of these techniques to other memory/storage systems (e.g., distributed storage, hybrid memory)
- Look at the other objectives
 - endurance, fairness, inter-application interference, QoS, tail latency.

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 storage
- This can
 - Lead to **orders-of-magnitude** improvements
 - **Enable new applications & computing platforms**
 - **Enable better understanding of nature**
 - ...
- Future of **truly storage-centric computing** is bright
 - We need to do research & design across the computing stack

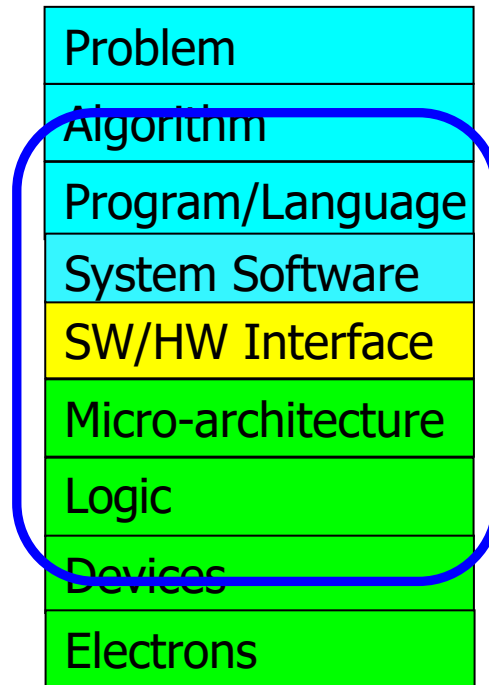
Data-centric

Data-driven

Data-aware



We Need to Revisit the Entire Stack



We can get there step by step

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

Acknowledgments

SAFARI

SAFARI Research Group

safari.ethz.ch

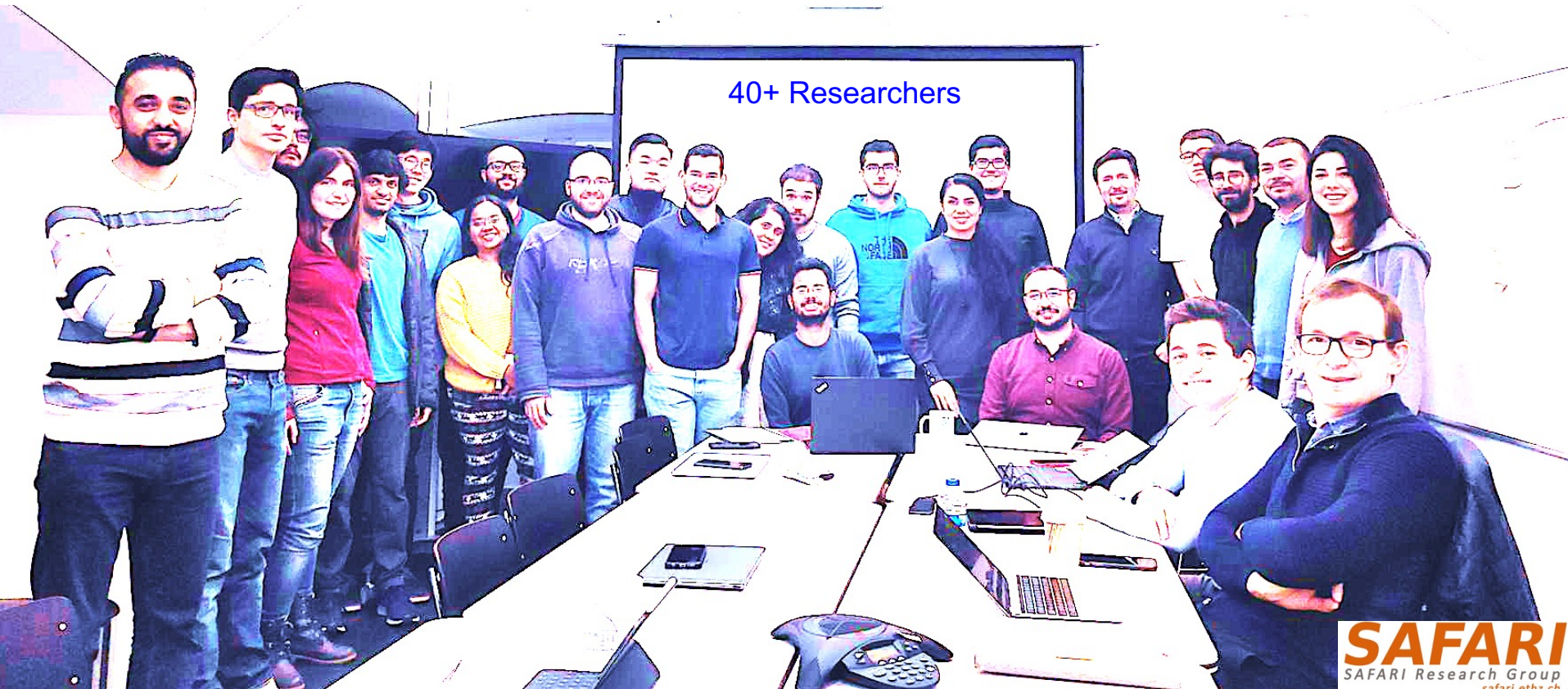
Think BIG, Aim HIGH!

<https://safari.ethz.ch>

Onur Mutlu's SAFARI Research Group

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

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



SAFARI
SAFARI Research Group
safari.ethz.ch

Think BIG, Aim HIGH!

SAFARI

<https://safari.ethz.ch>

SAFARI Newsletter December 2021 Edition

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

SAFARI
SAFARI Research Group

Think Big, Aim High

ETH zürich



View in your browser
December 2021



SAFARI Newsletter June 2023 Edition

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

SAFARI
SAFARI Research Group

Think Big, Aim High

ETH zürich



[View in your browser](#)

June 2023



SAFARI Introduction & Research

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



Seminar in Computer Architecture - Lecture 5: Potpourri of Research Topics (Spring 2023)



Onur Mutlu Lectures
32.6K subscribers

Subscribed

17



Share

Download

Clip



719 views Streamed 1 month ago Livestream - Seminar in Computer Architecture - ETH Zürich (Spring 2023)

SAFARI
SAFARI Research Group
safari.ethz.ch

THINK BIG, AIM HIGH!

SAFARI

<https://www.youtube.com/watch?v=mV2OuB2djEs>

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 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.

👤 440 followers

📍 ETH Zurich and Carnegie Mellon U...

🔗 <https://safari.ethz.ch/>

✉ omutlu@gmail.com

🏠 Overview

📁 Repositories 80

📁 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++ ☆ 502 🍴 204

📁 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 ☆ 122 🍴 46

📁 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++ ☆ 258 🍴 142

📁 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 ☆ 210 🍴 43

📁 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 ☆ 118 🍴 26

📁 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++ ☆ 105 🍴 34

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

Storage-Centric Computing

for Modern Data-Intensive Workloads

Onur Mutlu

omutlu@gmail.com

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

23 May 2024

Huawei

SAFARI

ETH zürich