

SAFARI EFCL Research Projects: Recent Results and Future Outlook

Onur Mutlu

omutlu@gmail.com

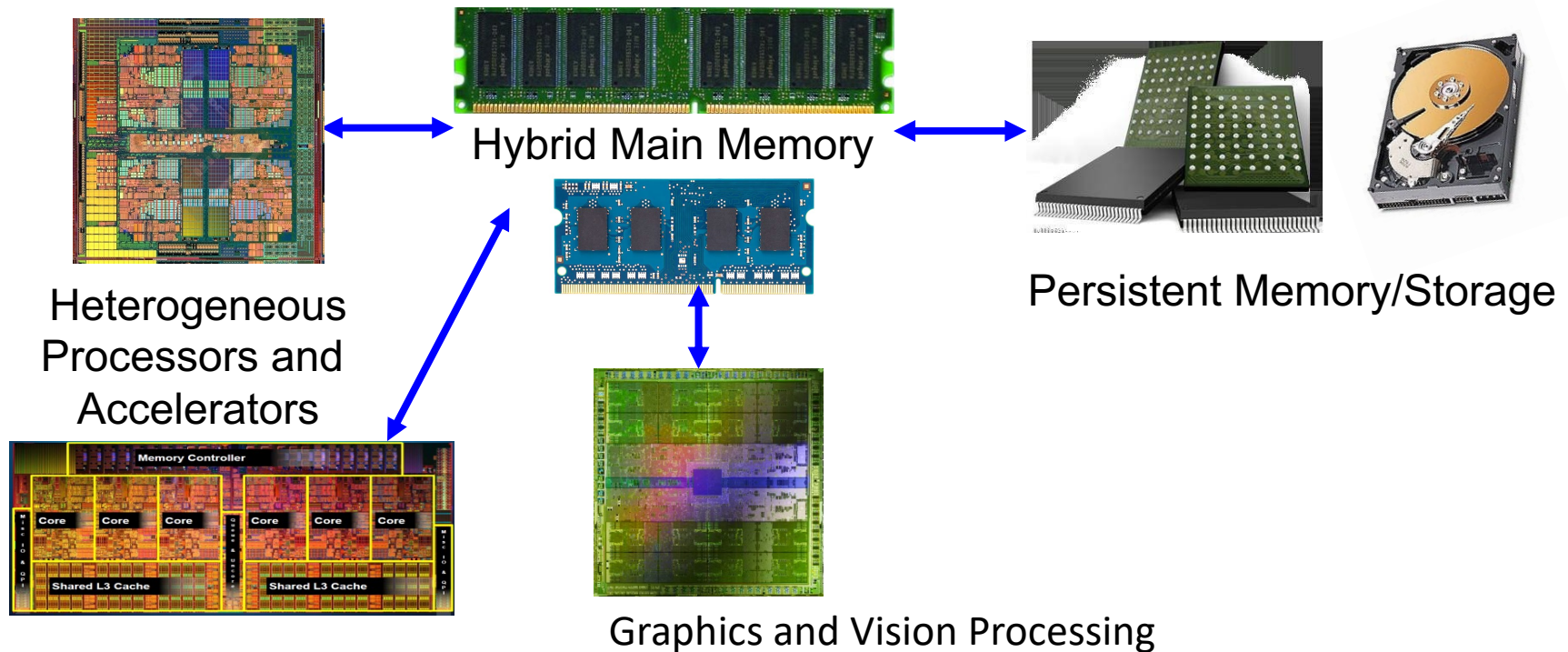
<https://people.inf.ethz.ch/omutlu>

23 May 2022

EFCL Mini-Conference

Current Research Mission

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



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

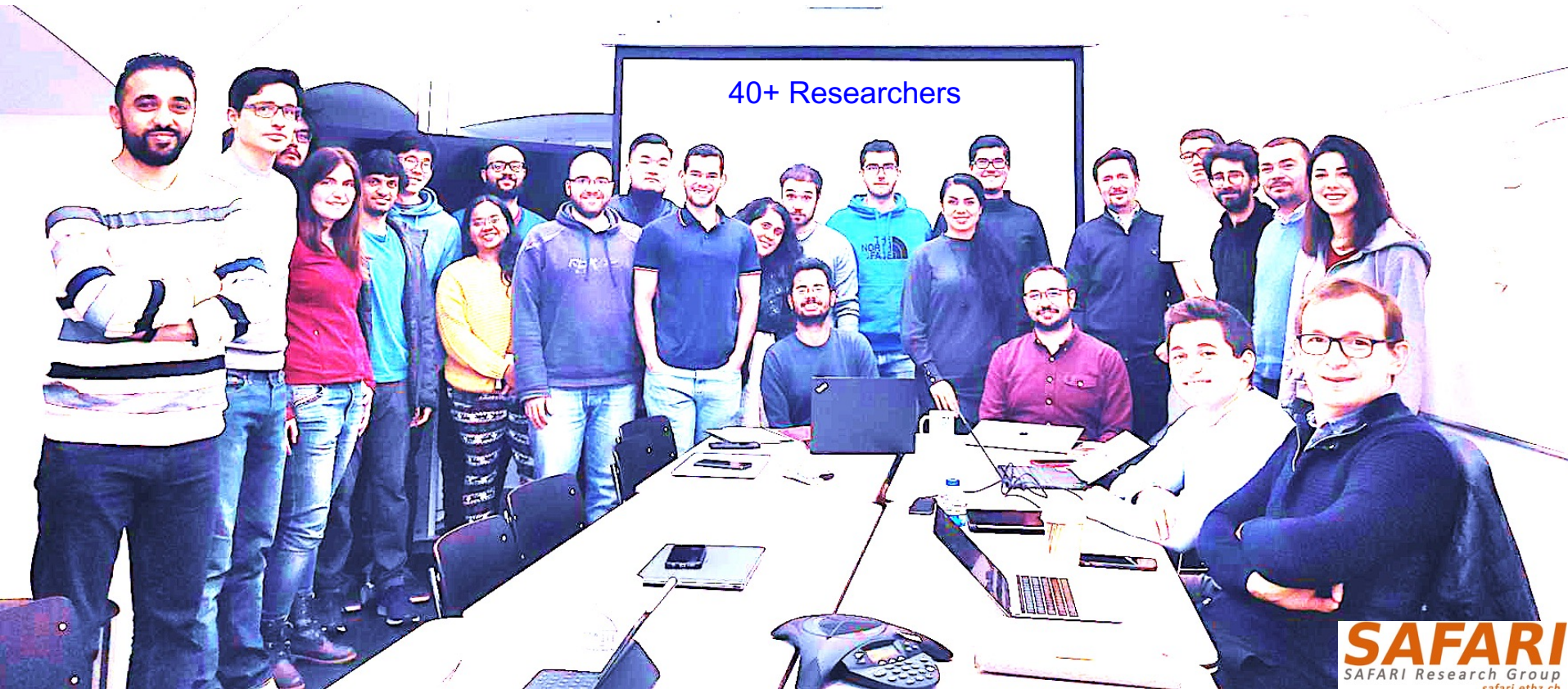
Data-driven

Data-aware

Onur Mutlu's SAFARI Research Group

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

<https://safari.ethz.ch/safari-newsletter-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



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/>

Onur Mutlu's SAFARI Research Group

SAFARI
SAFARI Research Group
safari.ethz.ch

Think BIG, Aim HIGH!

<https://safari.ethz.ch>

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>

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

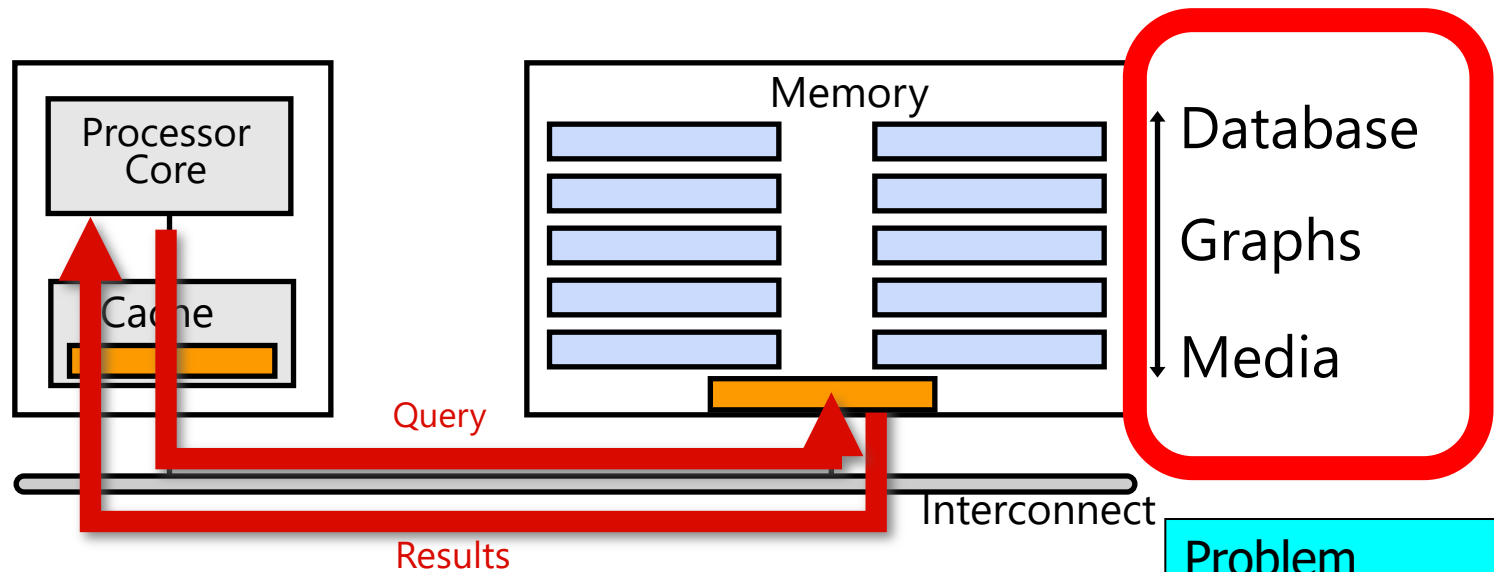
Current EFCL Projects

- “A New Methodology and Open-Source Benchmark Suite for Evaluating Data Movement Bottlenecks: A Processing-in-Memory Case Study”
 - Data-centric
- “Machine-Learning-Assisted Intelligent Microarchitectures to Reduce Memory Access Latency”
 - Data-driven
- “Cross-layer Hardware/Software Techniques to Enable Powerful Computation and Memory Optimizations”
 - Data-aware

A New Methodology and
Open-Source Benchmark Suite
for Evaluating Data Movement Bottlenecks:
A Processing-in-Memory Case Study

Juan Gómez Luna, Geraldo F. Oliveira,
Mohammad Sadr, Lois Orosa
Onur Mutlu

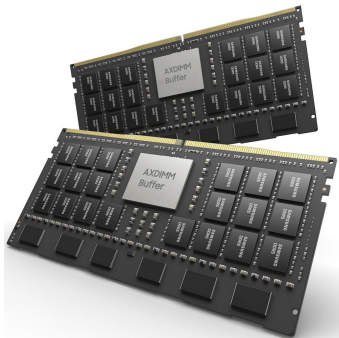
Goal: Processing Inside Memory



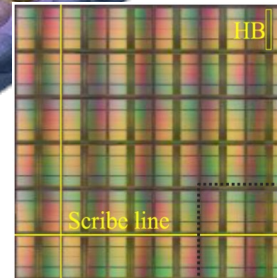
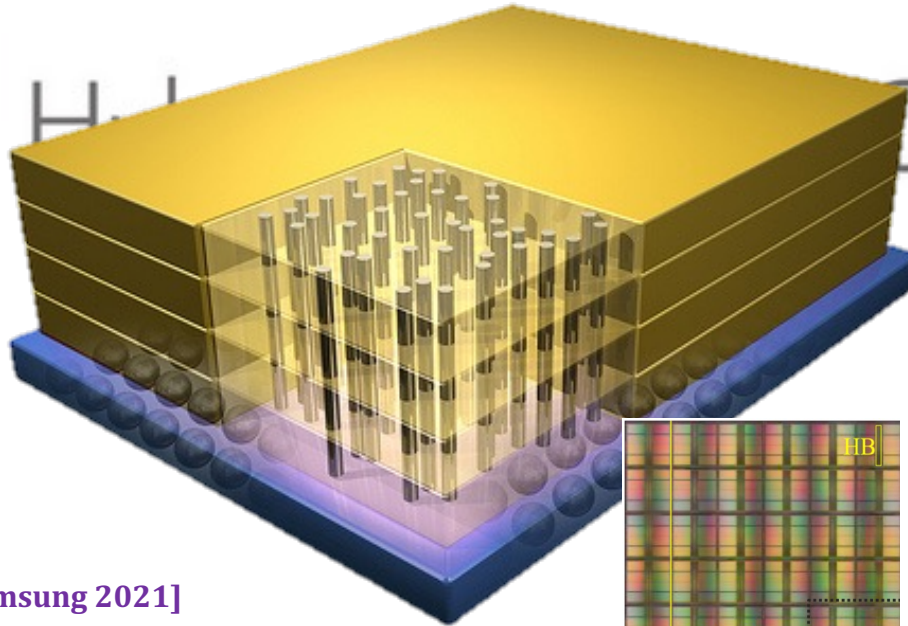
- Many questions ... How do we design the:
 - ❑ compute-capable memory & controllers?
 - ❑ processors & communication units?
 - ❑ software & hardware interfaces?
 - ❑ system software, compilers, languages?
 - ❑ algorithms & theoretical foundations?

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

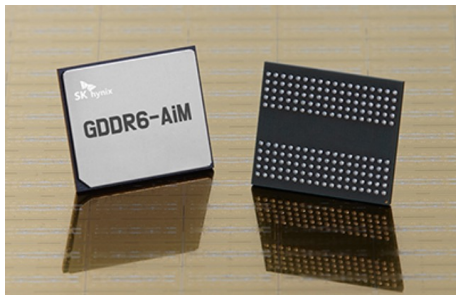
Processing-in-Memory Landscape Today



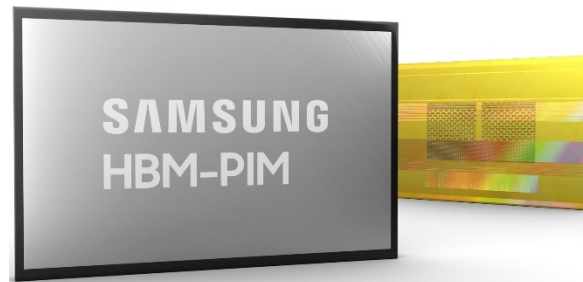
[Samsung 2021]



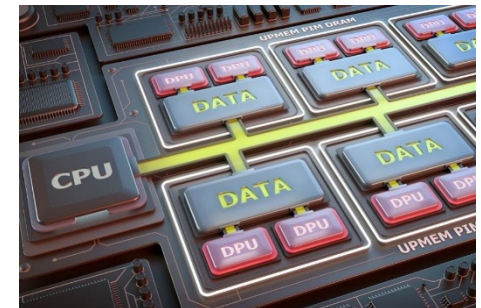
[Alibaba 2022]



[SK Hynix 2022]



[Samsung 2021]



[UPMEM 2019]

PIM Review and Open Problems

A Modern Primer on Processing in Memory

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

SAFARI Research Group

^a*ETH Zürich*

^b*Carnegie Mellon University*

^c*University of Illinois at Urbana-Champaign*

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

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

"A Modern Primer on Processing in Memory"

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

PIM Review and Open Problems (II)

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

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

[†]*Carnegie Mellon University*

[§]*ETH Zürich*

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu,

"Processing-in-Memory: A Workload-Driven Perspective"

Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence, to appear in November 2019.

[Preliminary arXiv version]

A Modern Primer on Processing in Memory

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

SAFARI Research Group

^aETH Zürich

^bCarnegie Mellon University

^cUniversity of Illinois at Urbana-Champaign

^dKing Mongkut's University of Technology North Bangkok

Abstract

Modern computing systems are overwhelmingly designed to move data to computation. This design choice goes directly against at least three key trends in computing that cause performance, scalability and energy bottlenecks: (1) data access is a key bottleneck as many important applications are increasingly data-intensive, and memory bandwidth and energy do not scale well, (2) energy consumption is a key limiter in almost all computing platforms, especially server and mobile systems, (3) data movement, especially off-chip to on-chip, is very expensive in terms of bandwidth, energy and latency, much more so than computation. These trends are especially severely-felt in the data-intensive server and energy-constrained mobile systems of today.

At the same time, conventional memory technology is facing many technology scaling challenges in terms of reliability, energy, and performance. As a result, memory system architects are open to organizing memory in different ways and making it more intelligent, at the expense of higher cost. The emergence of 3D-stacked memory plus logic, the adoption of error correcting codes inside the latest DRAM chips, proliferation of different main memory standards and chips, specialized for different purposes (e.g., graphics, low-power, high bandwidth, low latency), and the necessity of designing new solutions to serious reliability and security issues, such as the RowHammer phenomenon, are an evidence of this trend.

This chapter discusses recent research that aims to practically enable computation close to data, an approach we call *processing-in-memory* (PIM). PIM places computation mechanisms in or near where the data is stored (i.e., inside the memory chips, in the logic layer of 3D-stacked memory, or in the memory controllers), so that data movement between the computation units and memory is reduced or eliminated. While the general idea of PIM is not new, we discuss motivating trends in applications as well as memory circuits/technology that greatly exacerbate the need for enabling it in modern computing systems. We examine at least two promising new approaches to designing PIM systems to accelerate important data-intensive applications: (1) *processing using memory* by exploiting analog operational properties of DRAM chips to perform massively-parallel operations in memory, with low-cost changes, (2) *processing near memory* by exploiting 3D-stacked memory technology design to provide high memory bandwidth and low memory latency to in-memory logic. In both approaches, we describe and tackle relevant cross-layer research, design, and adoption challenges in devices, architecture, systems, and programming models. Our focus is on the development of in-memory processing designs that can be adopted in real computing platforms at low cost. We conclude by discussing work on solving key challenges to the practical adoption of PIM.

Keywords: memory systems, data movement, main memory, processing-in-memory, near-data processing, computation-in-memory, processing using memory, processing near memory, 3D-stacked memory, non-volatile memory, energy efficiency, high-performance computing, computer architecture, computing paradigm, emerging technologies, memory scaling, technology scaling, dependable systems, robust systems, hardware security, system security, latency, low-latency computing

1 Introduction	2
2 Major Trends Affecting Main Memory	4
3 The Need for Intelligent Memory Controllers to Enhance Memory Scaling	6
4 Perils of Processor-Centric Design	9
5 Processing-in-Memory (PIM): Technology Enablers and Two Approaches	12
5.1 New Technology Enablers: 3D-Stacked Memory and Non-Volatile Memory . . .	12
5.2 Two Approaches: Processing Using Memory (PUM) vs. Processing Near Memory (PNM)	13
6 Processing Using Memory (PUM)	14
6.1 RowClone	14
6.2 Ambit	15
6.3 Gather-Scatter DRAM	17
6.4 In-DRAM Security Primitives	17
7 Processing Near Memory (PNM)	18
7.1 Tesseract: Coarse-Grained Application-Level PNM Acceleration of Graph Processing	19
7.2 Function-Level PNM Acceleration of Mobile Consumer Workloads	20
7.3 Programmer-Transparent Function-Level PNM Acceleration of GPU Applications	21
7.4 Instruction-Level PNM Acceleration with PIM-Enabled Instructions (PEI) . .	21
7.5 Function-Level PNM Acceleration of Genome Analysis Workloads	22
7.6 Application-Level PNM Acceleration of Time Series Analysis	23
8 Enabling the Adoption of PIM	24
8.1 Programming Models and Code Generation for PIM	24
8.2 PIM Runtime: Scheduling and Data Mapping	25
8.3 Memory Coherence	27
8.4 Virtual Memory Support	27
8.5 Data Structures for PIM	28
8.6 Benchmarks and Simulation Infrastructures	29
8.7 Real PIM Hardware Systems and Prototypes	30
8.8 Security Considerations	30
9 Conclusion and Future Outlook	31

Main memory, built using the Dynamic Random Access Memory (DRAM) technology, is a major component in nearly all computing systems, including servers, cloud platforms, mobile/embedded devices, and sensor systems. Across all of these systems, the data working set sizes of modern applications are rapidly growing, while the need for fast analysis of such data is increasing. Thus, main memory is becoming an increasingly significant bottleneck across a wide variety of computing systems and applications [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Alleviating the main memory bottleneck requires the memory capacity, energy, cost, and performance to all scale in an efficient manner across technology generations. Unfortunately, it has become increasingly difficult in recent years, especially the past decade, to scale all of these dimensions [1, 2, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49], and thus the main memory bottleneck has been worsening.

A major reason for the main memory bottleneck is the high energy and latency cost associated with *data movement*. In modern computers, to perform any operation on data that resides in main memory, the processor must retrieve the data from main memory. This requires the memory controller to issue commands to a DRAM module across a relatively slow and power-hungry off-chip bus (known as the *memory channel*). The DRAM module sends the requested data across the memory channel, after which the data is placed in the caches and registers. The CPU can perform computation on the data once the data is in its registers. Data movement from the DRAM to the CPU incurs long latency and consumes a significant amount of energy [7, 50, 51, 52, 53, 54]. These costs are often exacerbated by the fact that much of the data brought into the caches is *not reused* by the CPU [52, 53, 55, 56], providing little benefit in return for the high latency and energy cost.

The cost of data movement is a fundamental issue with the *processor-centric* nature of contemporary computer systems. The CPU is considered to be the master in the system, and computation is performed only in the processor (and accelerators). In contrast, data storage and communication units, including the main memory, are treated as unintelligent workers that are incapable of computation. As a result of this processor-centric design paradigm, data moves a lot in the system between the computation units and communication/ storage units so that computation can be done on it. With the increasingly *data-centric* nature of contemporary and emerging appli-

How to Enable Adoption of Processing in Memory

Potential Barriers to Adoption of PIM

1. **Applications & software** for PIM
2. Ease of **programming** (interfaces and compiler/HW support)
3. **System** and **security** support: coherence, synchronization, virtual memory, isolation, communication interfaces, ...
4. **Runtime** and **compilation** systems for adaptive scheduling, data mapping, access/sharing control, ...
5. **Infrastructures** to assess benefits and feasibility

All can be solved with change of mindset

Our Goal

- To enable adoption of Processing-in-Memory (PIM) systems by solving various key challenges
 - Identifying workloads and functions suitable for PIM
 - Fundamental (architecture-independent) characterization
 - Architecture-specific suitability
 - Overcoming the problem of lack of useful tools
 - Profiling tools
 - Analytical performance and energy models (or ML-based models)
 - Simulation tools
- The project is organized in two phases
 - Phase 1: Methodology and open-source benchmark suite(s)
 - Phase 2: Follow-up project to enable PIM adoption

Results So Far (2021-2022)

- DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks [IEEE Access 2021]
- Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System [IEEE Access 2022]
- Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks [PACT 2021]
- Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Cooperation [ICDE 2022]
- FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications [IEEE Micro 2021]

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

Geraldo F. Oliveira

Juan Gómez-Luna Lois Orosa Saugata Ghose

Nandita Vijaykumar Ivan Fernandez Mohammad Sadrosadati

Onur Mutlu

SAFARI



UNIVERSITY OF
ILLINOIS
URBANA-CHAMPAIGN



UNIVERSITY OF
TORONTO



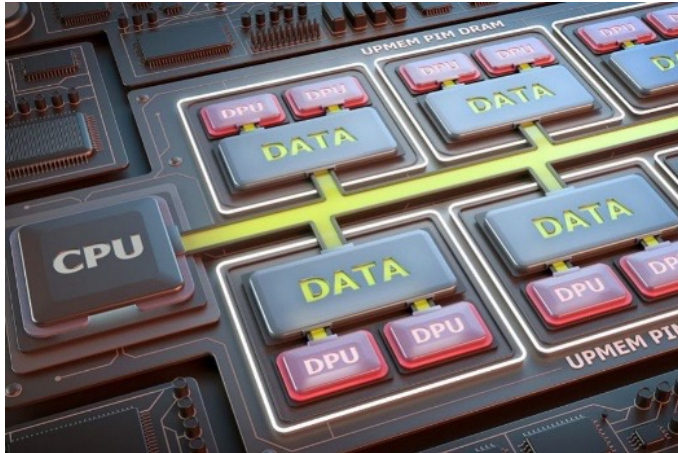
UNIVERSIDAD
DE MÁLAGA

Executive Summary

- **Problem**: Data movement is a major bottleneck in modern systems. However, it is **unclear** how to identify:
 - **different sources** of data movement bottlenecks
 - the **most suitable** mitigation technique (e.g., caching, prefetching, near-data processing) for a given data movement bottleneck
- **Goals**:
 1. Design a methodology to **identify** sources of data movement bottlenecks
 2. **Compare** compute- and memory-centric data movement mitigation techniques
- **Key Approach**: Perform a large-scale application characterization to identify **key metrics** that reveal the sources of data movement bottlenecks
- **Key Contributions**:
 - **Experimental characterization** of 77K functions across 345 applications
 - A **methodology** to characterize applications based on data movement bottlenecks and their relation with different data movement mitigation techniques
 - **DAMOV**: a **benchmark suite** with **144 functions** for data movement studies
 - **Four case-studies** to highlight DAMOV's applicability to open research problems

Near-Data Processing

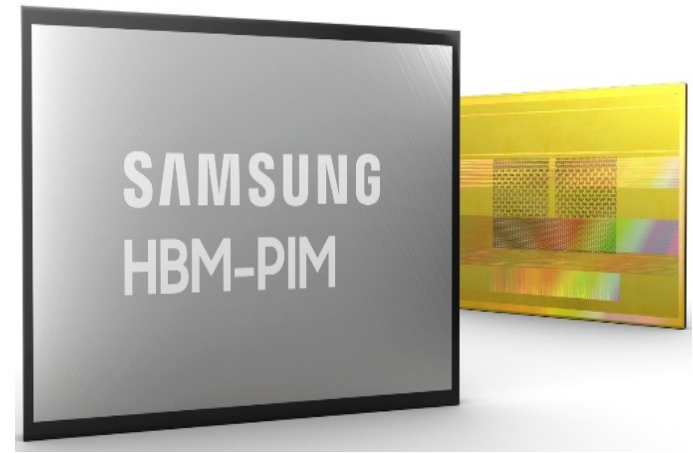
UPMEM (2019)



Near-DRAM-banks processing
for general-purpose computing

0.9 TOPS compute throughput¹

Samsung FIMDRAM (2021)

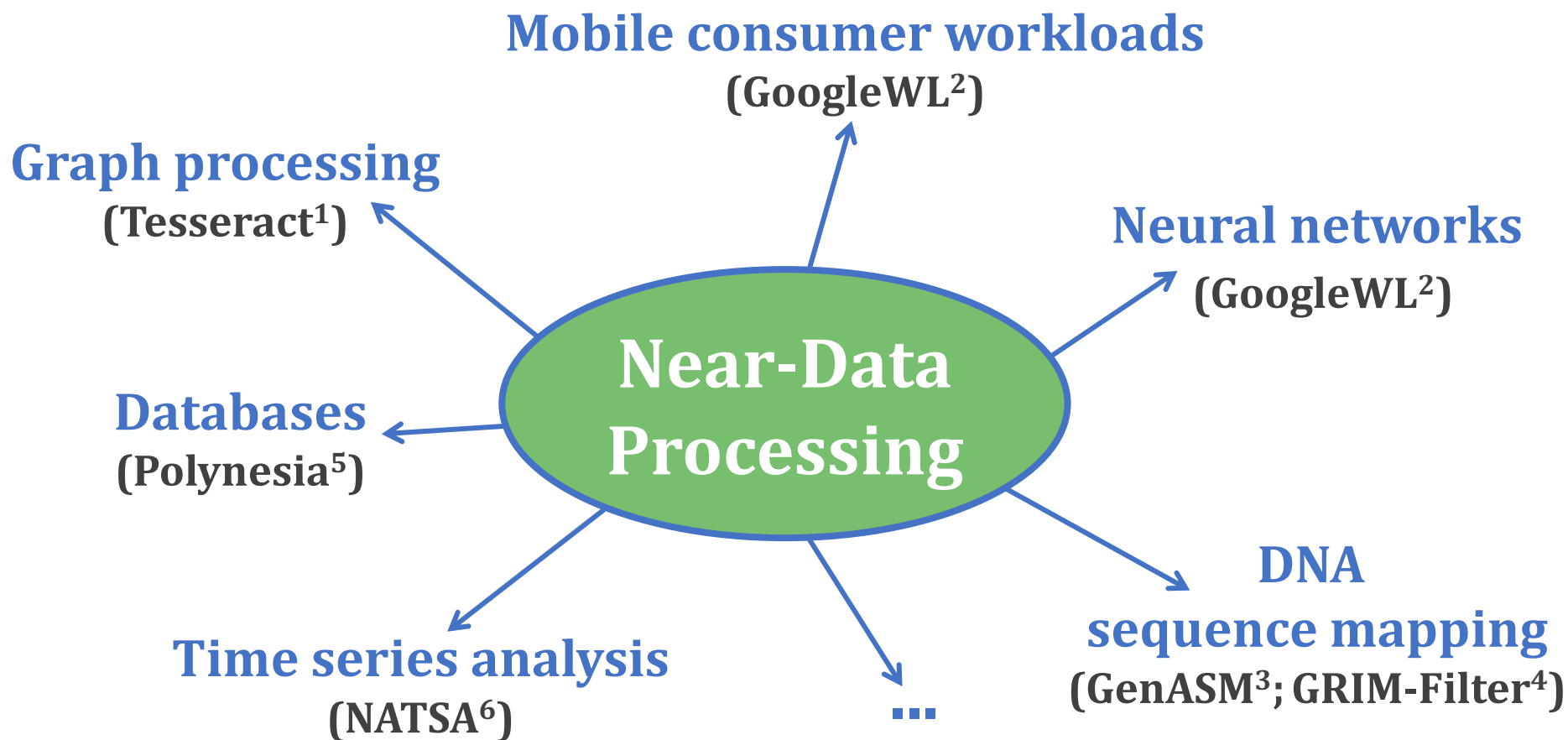


Near-DRAM-banks processing
for neural networks

1.2 TFLOPS compute throughput²

The goal of Near-Data Processing (NDP) is
to mitigate data movement

When to Employ Near-Data Processing?



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

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

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

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

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

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

Key Approach

- New **workload characterization methodology** to analyze:
 - data movement bottlenecks
 - suitability of different data movement mitigation mechanisms
- Two main profiling strategies:

Architecture-independent profiling:

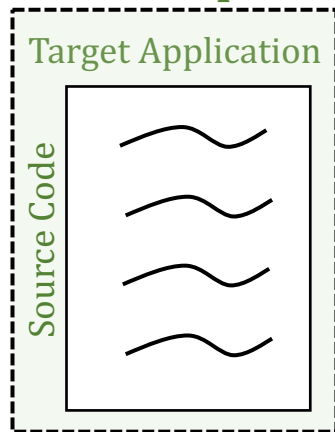
characterizes the memory behavior **independently**
of the underlying **hardware**

Architecture-dependent profiling:

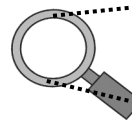
evaluates the **impact of the system configuration**
on the memory behavior

Methodology Overview

User Input



Step 1 Application Profiling



Profiler

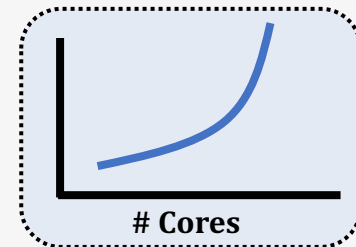
roi_begin

roi_end

DAMOV-SIM Simulator

```
ld 0xFF
st 0xAF
ld 0xFF
st 0xAF
ld 0xFF
```

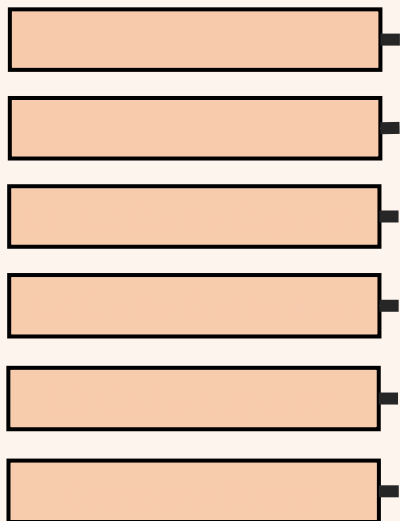
Memory Traces



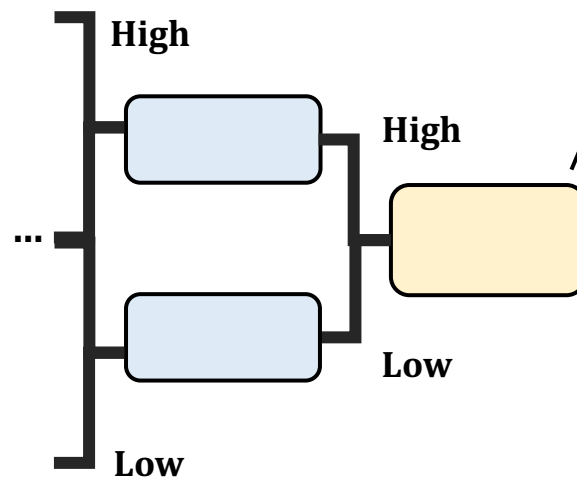
Scalability Analysis

Methodology Output

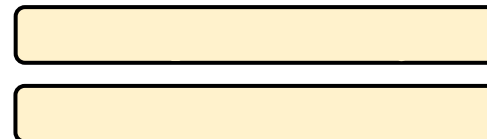
Memory Bottleneck Classes



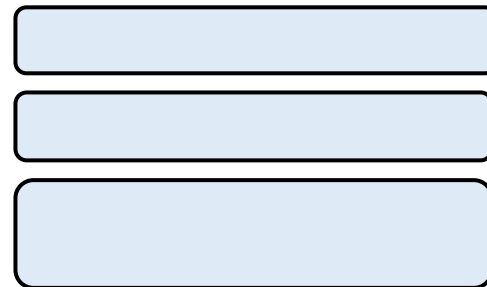
SAFARI



Step 2 Locality-based Clustering



Step 3 Memory Bottleneck Class.



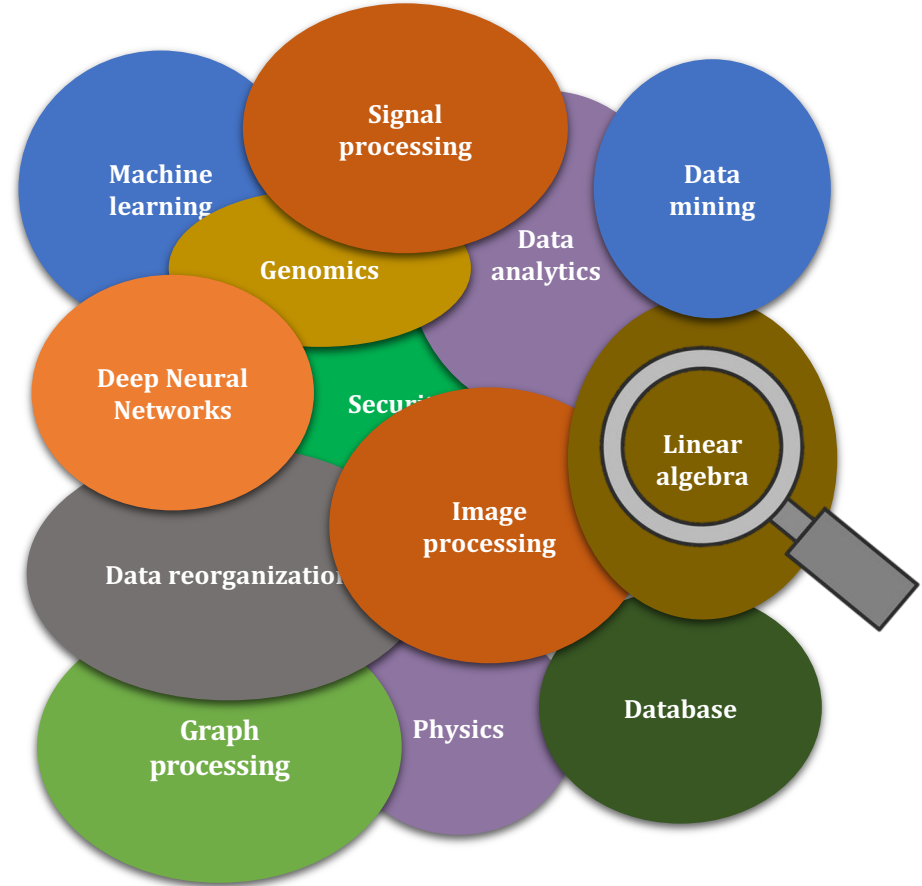
Step 1: Application Profiling

- We analyze 345 applications from distinct domains:

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

...

SAFARI



Step 3: Memory Bottleneck Analysis

**Six classes of
data movement bottlenecks:**

each class \leftrightarrow data movement
mitigation mechanism

Memory Bottleneck Class

1a: *DRAM
Bandwidth*

1b: *DRAM Latency*

1c: *L1/L2
Cache Capacity*

2a: *L3 Cache
Contention*

2b: *L1 Cache
Capacity*

2c: *Compute-Bound*

DAMOV is Open Source

- We open-source our **benchmark suite** and our **toolchain**

CMU-SAFARI / DAMOV

<> Code Issues Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags

Go to file

Add file

Code

About



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 processing. Described by Oliveira et al. (preliminary version at <https://arxiv.org/pdf/2105.03725.pdf>)

Readme

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages



omutlu Update README.md

ce1b4ea 17 days ago 5 commits

simulator	Cleaning	19 days ago
README.md	Update README.md	17 days ago
get_workloads.sh	DAMOV -- first commit	19 days ago

README.md

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

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 processing.

The DAMOV benchmark suite is the first open-source benchmark suite for main memory data movement-related studies, based on our systematic characterization methodology. This suite consists of 144 functions representing different sources of data movement bottlenecks and can be used as a baseline benchmark set for future data-movement mitigation research. The applications in the DAMOV benchmark suite belong to popular benchmark suites, including [BWA](#), [Chai](#), [Darknet](#), [GASE](#), [Hardware Effects](#), [Hashjoin](#), [HPCC](#), [HPCG](#), [Ligra](#), [PARSEC](#), [Parboil](#), [PolyBench](#), [Phoenix](#), [Rodinia](#), [SPLASH-2](#), [STREAM](#).

DAMOV-SIM

DAMOV
Benchmarks

SAFARI

DAMOV is Open Source

- We open-source our [benchmark suite](https://github.com/CMU-SAFARI/DAMOV) and our [toolchain](https://github.com/CMU-SAFARI/DAMOV)

CMU-SAFARI / DAMOV

<> Code Issues Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags

Go to file

Add file

Code

About

DAMOV is a benchmark suite and a

Get DAMOV at:

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

README.md

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

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 processing.

The DAMOV benchmark suite is the first open-source benchmark suite for main memory data movement-related studies, based on our systematic characterization methodology. This suite consists of 144 functions representing different sources of data movement bottlenecks and can be used as a baseline benchmark set for future data-movement mitigation research. The applications in the DAMOV benchmark suite belong to popular benchmark suites, including [BWA](#), [Chai](#), [Darknet](#), [GASE](#), [Hardware Effects](#), [Hashjoin](#), [HPCC](#), [HPCG](#), [Ligra](#), [PARSEC](#), [Parboil](#), [PolyBench](#), [Phoenix](#), [Rodinia](#), [SPLASH-2](#), [STREAM](#).

Readme

Releases

No releases published
[Create a new release](#)

Packages

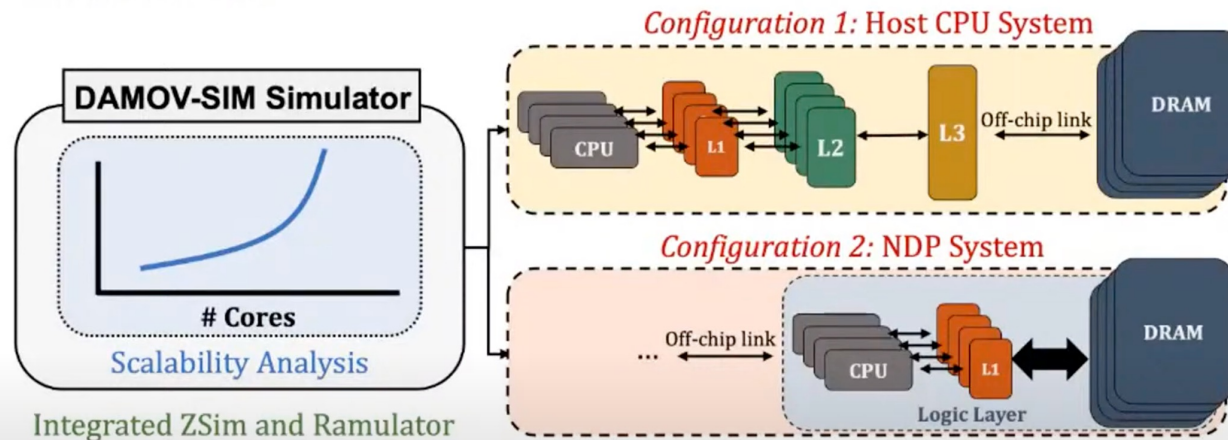
No packages published
[Publish your first package](#)

Languages

More on DAMOV Analysis Methodology & Workloads

Step 3: Memory Bottleneck Classification (2/)

- **Goal:** identify the specific sources of data movement bottlenecks



- **Scalability Analysis:**
 - 1, 4, 16, 64, and 256 out-of-order/in-order host and NDP CPU cores
 - 3D-stacked memory as main memory

SAFARI DAMOV-SIM: <https://github.com/CMU-SAFARI/DAMOV> 30

SAFARI Live Seminar: DAMOV: A New Methodology & Benchmark Suite for Data Movement Bottlenecks

352 views • Streamed live on Jul 22, 2021

18 0 SHARE SAVE ...



Onur Mutlu Lectures
17.7K subscribers

ANALYTICS

EDIT VIDEO

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

More on DAMOV Methods & Benchmarks

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

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

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

LOIS OROSA, ETH Zürich, Switzerland

SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA

NANDITA VIJAYKUMAR, University of Toronto, Canada

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

MOHAMMAD SADROSADATI, ETH Zürich, Switzerland

ONUR MUTLU, ETH Zürich, Switzerland

DAMOV Analysis Methodology & Workloads

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

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

LOIS OROSA, ETH Zürich, Switzerland

SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA

NANDITA VIJAYKUMAR, University of Toronto, Canada

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

MOHAMMAD SADROSADATI, Institute for Research in Fundamental Sciences (IPM), Iran & ETH Zürich, Switzerland

ONUR MUTLU, ETH Zürich, Switzerland

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

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

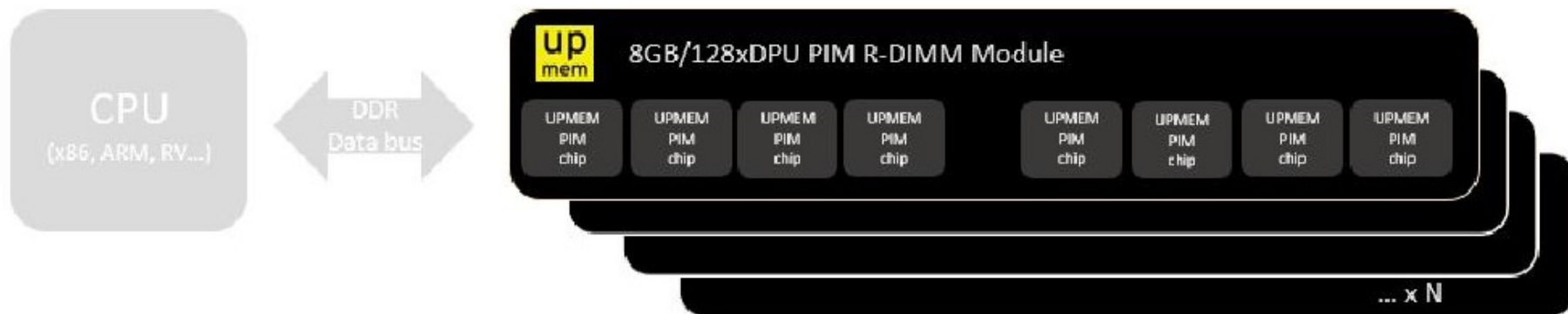
Results So Far (2021-2022)

- DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks [IEEE Access 2021]
- Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System [IEEE Access 2022]
- Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks [PACT 2021]
- Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Cooperation [ICDE 2022]
- FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications [IEEE Micro 2021]

Processing-in-Memory in the Real World

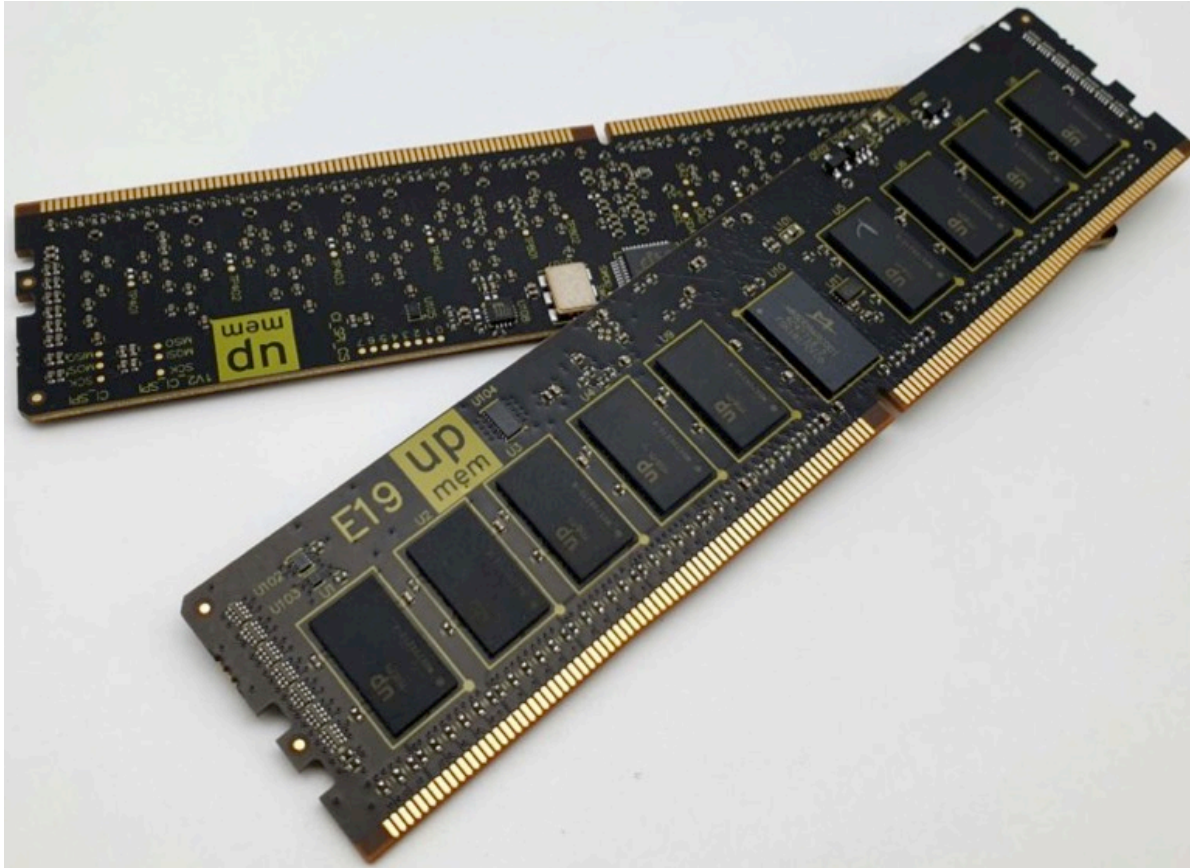
UPMEM Processing-in-DRAM Engine (2019)

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

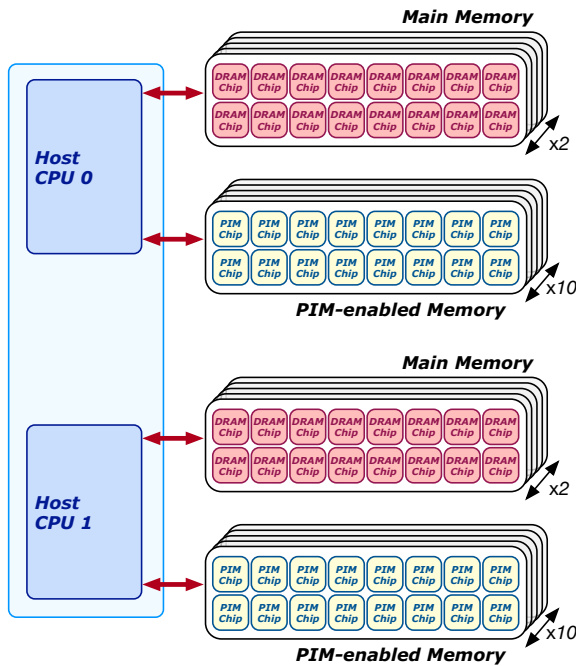


UPMEM Memory Modules

- E19: 8 chips DIMM (1 rank). DPUs @ 267 MHz
- P21: 16 chips DIMM (2 ranks). DPUs @ 350 MHz



2,560-DPU Processing-in-Memory System



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

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
 IZZAT EL HAJJ, American University of Beirut, Lebanon
 IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain
 CHRISTINA GIANNOULA, ETH Zürich, Switzerland and NTUA, Greece
 GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
 ONUR MUTLU, ETH Zürich, Switzerland

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

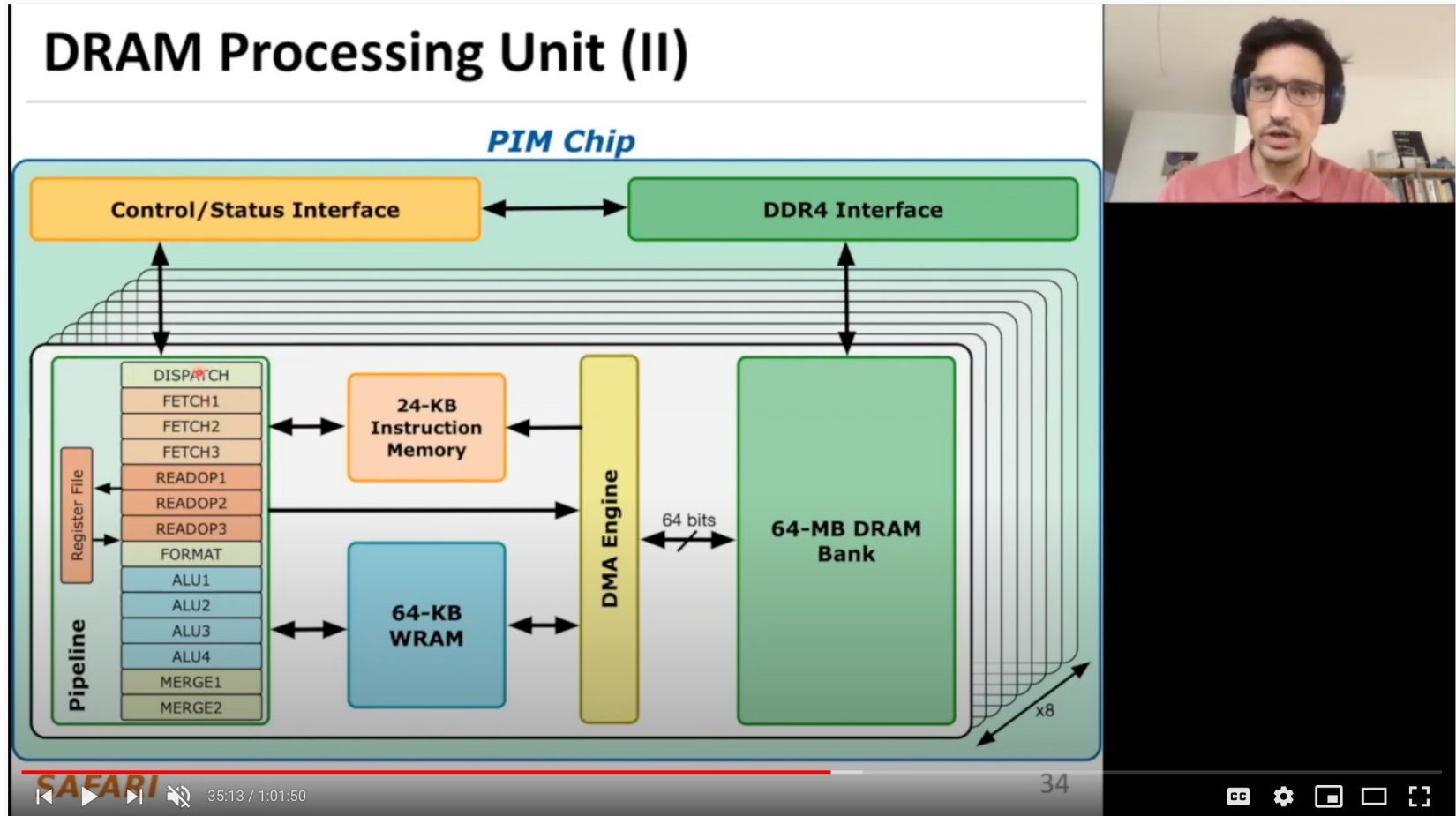
Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware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

This paper provides the first comprehensive analysis of the first publicly-available real-world PIM architecture. We make two key contributions. First, we conduct an experimental characterization of the UPMEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present *PrIM (Processing-In-Memory benchmarks)*, a benchmark suite of 16 workloads from different application domains (e.g., dense/sparse linear algebra, databases, data analytics, graph processing, neural networks, bioinformatics, image processing), which we identify as memory-bound. We evaluate the performance and scaling characteristics of PrIM benchmarks on the UPMEM PIM architecture, and compare their performance and energy consumption to their state-of-the-art CPU and GPU counterparts. Our extensive evaluation conducted on two real UPMEM-based PIM systems with 640 and 2,560 DPUs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.



<https://arxiv.org/pdf/2105.03814.pdf>

More on the UPMEM PIM System



ETH ZÜRICH HAUPTGEBÄUDE

Computer Architecture - Lecture 12d: Real Processing-in-DRAM with UPMEM (ETH Zürich, Fall 2020)

1,120 views • Oct 31, 2020

30 0 SHARE SAVE ...



Onur Mutlu Lectures
16.7K subscribers

ANALYTICS

EDIT VIDEO

<https://www.youtube.com/watch?v=Sscy1Wrr22A&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=26>

UPMEM PIM System Summary & Analysis

- Juan Gomez-Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu,
"Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware"
*Invited Paper at Workshop on Computing with Unconventional Technologies (**CUT**), Virtual, October 2021.*
[[arXiv version](#)]
[[PrIM Benchmarks Source Code](#)]
[[Slides \(pptx\)](#) ([pdf](#))]
[[Talk Video](#) (37 minutes)]
[[Lightning Talk Video](#) (3 minutes)]

Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware

Juan Gómez-Luna
ETH Zürich

Izzat El Hajj
*American University
of Beirut*

Ivan Fernandez
*University
of Malaga*

Christina Giannoula
*National Technical
University of Athens*

Geraldo F. Oliveira
ETH Zürich

Onur Mutlu
ETH Zürich

Understanding a Modern Processing-in-Memory Architecture:

Benchmarking and Experimental Characterization

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

<https://arxiv.org/pdf/2105.03814.pdf>

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

Executive Summary

- **Data movement** between memory/storage units and compute units is a major contributor to execution time and energy consumption
- **Processing-in-Memory** (PIM) is a paradigm that can tackle the **data movement bottleneck**
 - Though explored for +50 years, technology challenges prevented the successful materialization
- UPMEM has designed and fabricated **the first publicly-available real-world PIM architecture**
 - DDR4 chips embedding in-order multithreaded DRAM Processing Units (DPUs)
- Our work:
 - **Introduction** to UPMEM programming model and PIM architecture
 - **Microbenchmark-based characterization** of the DPU
 - Benchmarking and **workload suitability** study
- **Main contributions:**
 - Comprehensive **characterization and analysis of the first commercially-available PIM architecture**
 - **PrIM** (**P**rocessing-**I**n-**M**emory) benchmarks:
 - 16 workloads that are memory-bound in conventional processor-centric systems
 - Strong and weak scaling characteristics
 - Comparison to **state-of-the-art CPU and GPU**
- **Takeaways:**
 - Workload characteristics for **PIM suitability**
 - **Programming** recommendations
 - Suggestions and hints for **hardware and architecture designers** of future PIM systems
 - **PrIM**: (a) programming samples, (b) evaluation and comparison of current and future PIM systems

PrIM Benchmarks: Application Domains

Domain	Benchmark	Short name
Dense linear algebra	Vector Addition	VA
	Matrix-Vector Multiply	GEMV
Sparse linear algebra	Sparse Matrix-Vector Multiply	SpMV
Databases	Select	SEL
	Unique	UNI
Data analytics	Binary Search	BS
	Time Series Analysis	TS
Graph processing	Breadth-First Search	BFS
Neural networks	Multilayer Perceptron	MLP
Bioinformatics	Needleman-Wunsch	NW
Image processing	Image histogram (short)	HST-S
	Image histogram (large)	HST-L
Parallel primitives	Reduction	RED
	Prefix sum (scan-scan-add)	SCAN-SSA
	Prefix sum (reduce-scan-scan)	SCAN-RSS
	Matrix transposition	TRNS

PrIM Benchmarks are Open Source

- All microbenchmarks, benchmarks, and scripts
- <https://github.com/CMU-SAFARI/prim-benchmarks>

CMU-SAFARI / **prim-benchmarks** Unwatch 2 Star 2 Fork 1

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main prim-benchmarks / README.md Go to file ...

Juan Gomez Luna PrIM -- first commit Latest commit 3de4b49 9 days ago History

1 contributor

168 lines (132 sloc) 5.79 KB Raw Blame

PrIM (Processing-In-Memory Benchmarks)

PrIM is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. PrIM is developed to evaluate, analyze, and characterize the first publicly-available real-world processing-in-memory (PIM) architecture, the [UPMEM](#) PIM architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called DRAM Processing Units (DPUs), integrated in the same chip.

PrIM provides a common set of workloads to evaluate the UPMEM PIM architecture with and can be useful for programming, architecture and system researchers all alike to improve multiple aspects of future PIM hardware and software. The workloads have different characteristics, exhibiting heterogeneity in their memory access patterns, operations and data types, and communication patterns. This repository also contains baseline CPU and GPU implementations of PrIM benchmarks for comparison purposes.

PrIm also includes a set of microbenchmarks can be used to assess various architecture limits such as compute throughput and memory bandwidth.

Understanding a Modern PIM Architecture

Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System

**JUAN GÓMEZ-LUNA¹, IZZAT EL HAJJ², IVAN FERNANDEZ^{1,3}, CHRISTINA GIANNOULA^{1,4},
GERALDO F. OLIVEIRA¹, AND ONUR MUTLU¹**

¹ETH Zürich

²American University of Beirut

³University of Malaga

⁴National Technical University of Athens

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

<https://doi.org/10.1109/ACCESS.2022.3174101>

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

Observations, Recommendations, Takeaways

GENERAL PROGRAMMING RECOMMENDATIONS

1. Execute on the *DRAM Processing Units (DPUs)* **portions of parallel code** that are as long as possible.
2. Split the workload into **independent data blocks**, which the DPUs operate on independently.
3. Use **as many working DPUs** in the system as possible.
4. Launch at least **11 tasklets (i.e., software threads)** per DPU.

PROGRAMMING RECOMMENDATION 1

For data movement between the DPU's MRAM bank and the WRAM, **use large DMA transfer sizes when all the accessed data is going to be used.**

KEY OBSERVATION 7

Larger CPU-DPU and DPU-CPU transfers between the host main memory and the DRAM Processing Unit's Main memory (MRAM) banks **result in higher sustained bandwidth.**

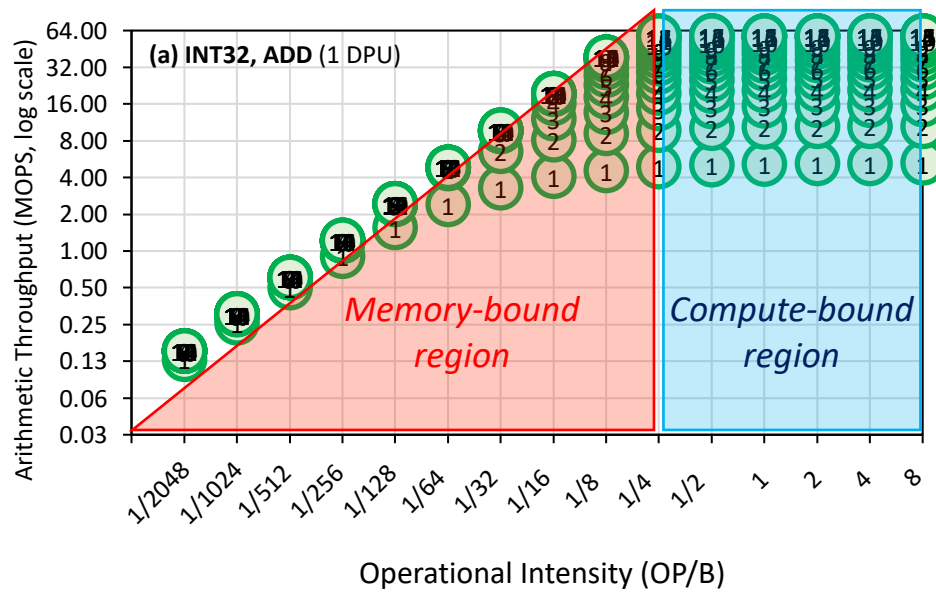
KEY TAKEAWAY 1

The UPMEM PIM architecture is fundamentally compute bound. As a result, **the most suitable work-loads are memory-bound.**

Outline

- Introduction
 - Accelerator Model
 - UPMEM-based PIM System Overview
- UPMEM PIM Programming
 - Vector Addition
 - CPU-DPU Data Transfers
 - Inter-DPU Communication
 - CPU-DPU/DPU-CPU Transfer Bandwidth
- DRAM Processing Unit
 - Arithmetic Throughput
 - WRAM and MRAM Bandwidth
- PRIM Benchmarks
 - Roofline Model
 - Benchmark Diversity
- Evaluation
 - Strong and Weak Scaling
 - Comparison to CPU and GPU
- Key Takeaways

Key Takeaway 1

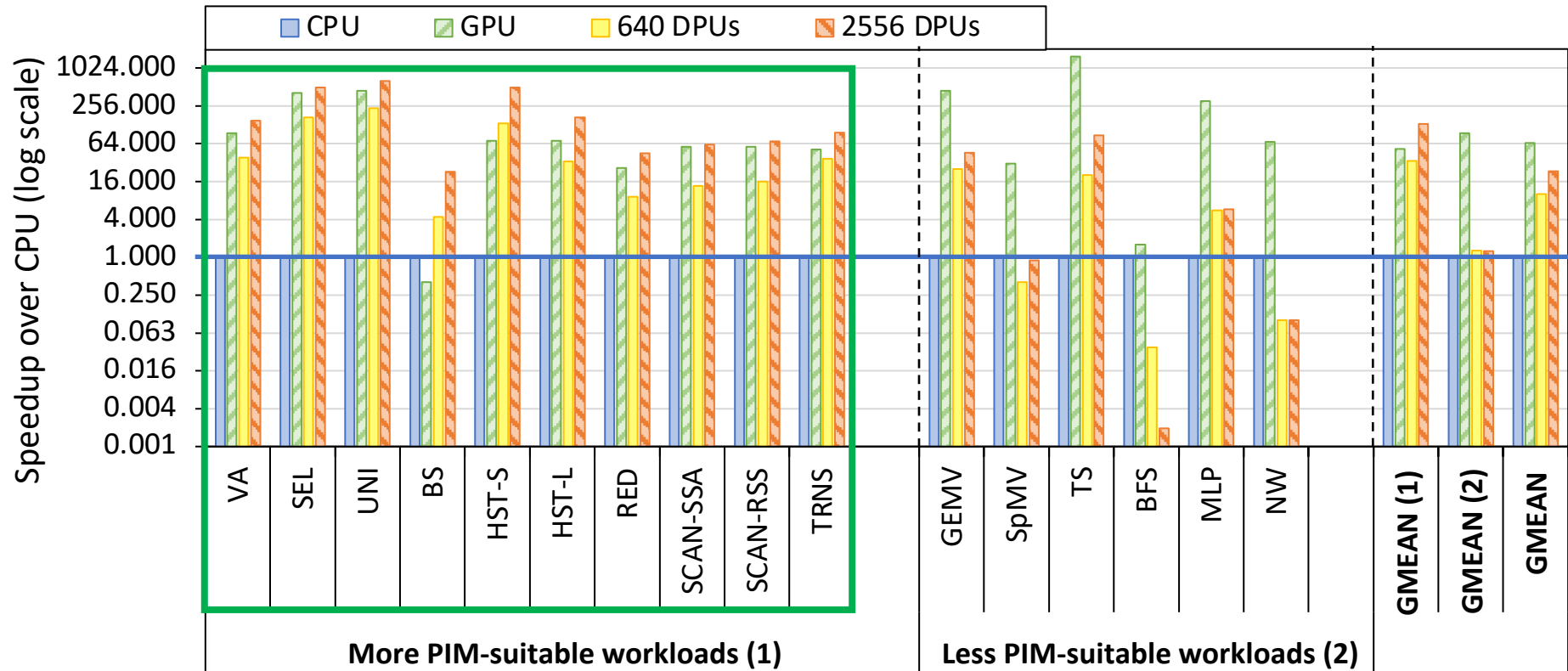


The throughput saturation point is as low as $\frac{1}{4}$ OP/B, i.e., 1 integer addition per every 32-bit element fetched

KEY TAKEAWAY 1

The UPMEM PIM architecture is fundamentally compute bound. As a result, the most suitable workloads are memory-bound.

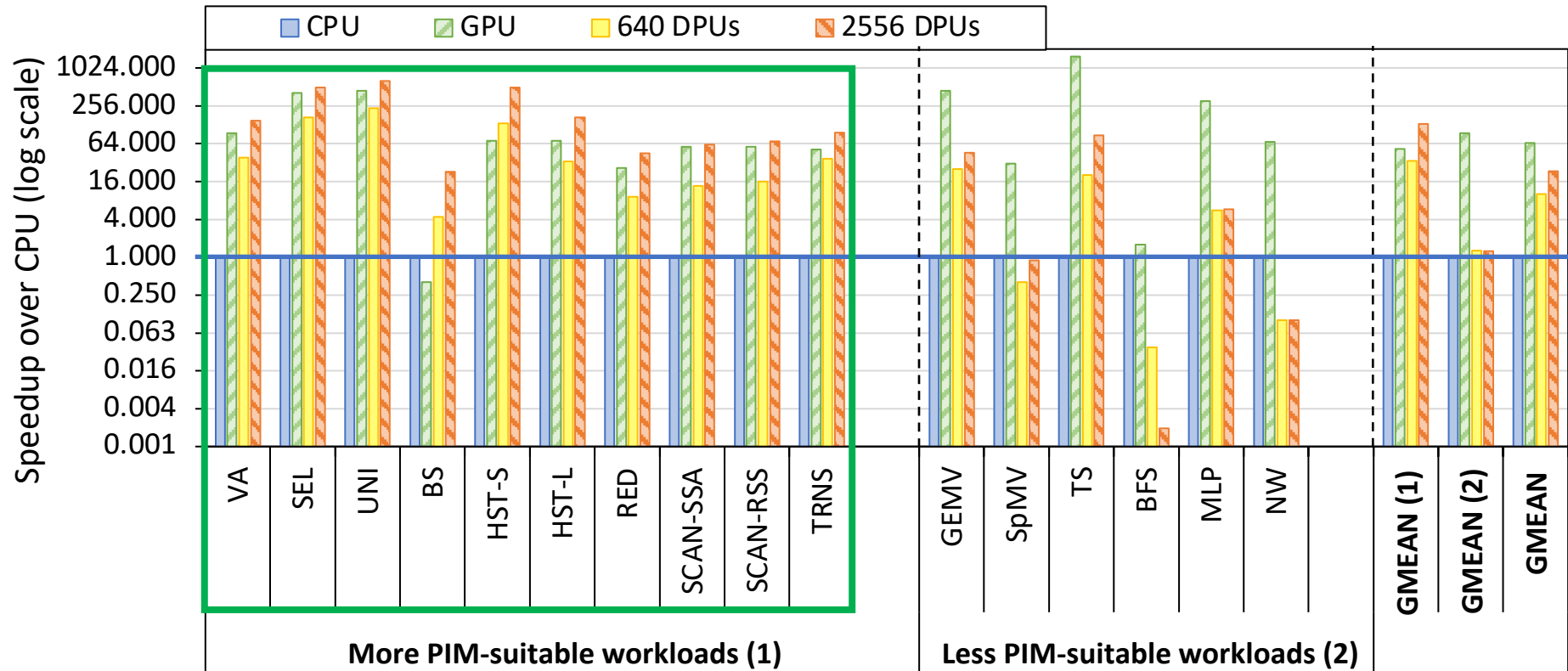
Key Takeaway 2



KEY TAKEAWAY 2

The most well-suited workloads for the UPMEM PIM architecture use no arithmetic operations or use only simple operations (e.g., bitwise operations and integer addition/subtraction).

Key Takeaway 3



KEY TAKEAWAY 3

The most well-suited workloads for the UPMEM PIM architecture require little or no communication across DPUs (inter-DPU communication).

Key Takeaway 4

KEY TAKEAWAY 4

- UPMEM-based PIM systems **outperform state-of-the-art CPUs** in terms of performance and energy efficiency on most of PrIM benchmarks.
- UPMEM-based PIM systems **outperform state-of-the-art GPUs** on a majority of PrIM benchmarks, and the outlook is even more positive for future PIM systems.
- UPMEM-based PIM systems are **more energy-efficient than state-of-the-art CPUs and GPUs on workloads that they provide performance improvements** over the CPUs and the GPUs.

Understanding a Modern PIM Architecture

Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System

**JUAN GÓMEZ-LUNA¹, IZZAT EL HAJJ², IVAN FERNANDEZ^{1,3}, CHRISTINA GIANNOULA^{1,4},
GERALDO F. OLIVEIRA¹, AND ONUR MUTLU¹**

¹ETH Zürich

²American University of Beirut

³University of Malaga

⁴National Technical University of Athens

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

<https://doi.org/10.1109/ACCESS.2022.3174101>

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

Understanding a Modern Processing-in-Memory Architecture:

Benchmarking and Experimental Characterization

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

el1goluj@gmail.com

<https://arxiv.org/pdf/2105.03814.pdf>

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

Experimental Analysis of the UPMEM PIM Engine

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

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

IZZAT EL HAJJ, American University of Beirut, Lebanon

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

CHRISTINA GIANNOULA, ETH Zürich, Switzerland and NTUA, Greece

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland

ONUR MUTLU, ETH Zürich, Switzerland

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

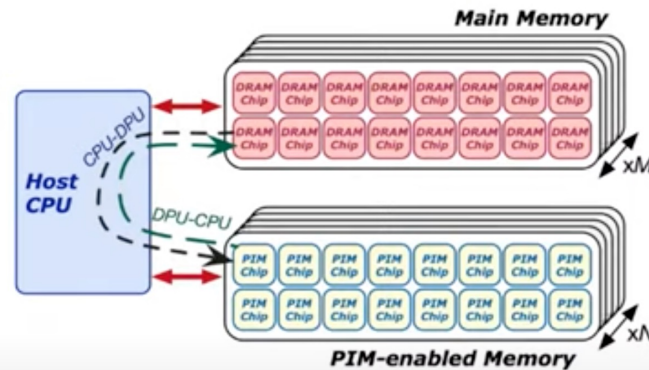
Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware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

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

More on Analysis of the UPMEM PIM Engine

Inter-DPU Communication

- There is **no direct communication channel between DPUs**



- Inter-DPU communication takes place via the host CPU using CPU-DPU and DPU-CPU transfers
- Example communication patterns:
 - Merging of partial results to obtain the final result
 - Only DPU-CPU transfers
 - Redistribution of intermediate results for further computation
 - DPU-CPU transfers and CPU-DPU transfers



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

1,868 views • Streamed live on Jul 12, 2021

81 0 SHARE SAVE ...



Onur Mutlu Lectures
17.6K subscribers

Talk Title: Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization
Dr. Juan Gómez-Luna, SAFARI Research Group, D-ITET, ETH Zurich

ANALYTICS

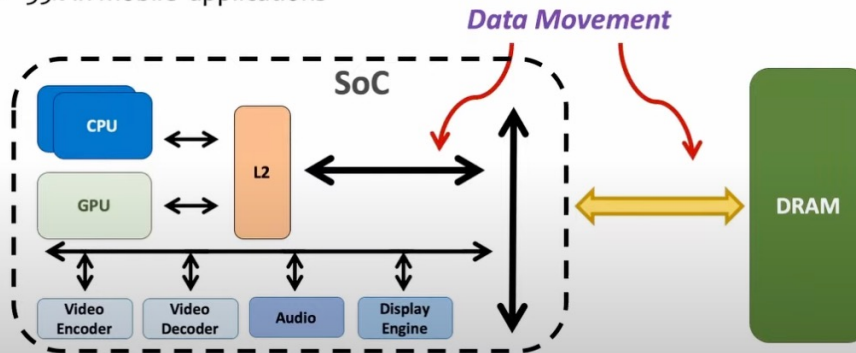
EDIT VIDEO

https://www.youtube.com/watch?v=D8Hjy2iU9l4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9

More on Analysis of the UPMEM PIM Engine

Data Movement in Computing Systems

- **Data movement** dominates **performance** and is a major system **energy bottleneck**
- **Total system energy**: data movement accounts for
 - 62% in consumer applications*,
 - 40% in scientific applications*,
 - 35% in mobile applications*



* Boroumand et al., "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," ASPLOS 2018

* Kestor et al., "Quantifying the Energy Cost of Data Movement in Scientific Applications," IISWC 2013

* Pandiyan and Wu, "Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms," IISWC 2014

SAFARI

3

Understanding a Modern Processing-in-Memory Arch: Benchmarking & Experimental Characterization; 21m

3,482 views • Premiered Jul 25, 2021

38 0 SHARE SAVE ...



Onur Mutlu Lectures
17.9K subscribers

ANALYTICS

EDIT VIDEO

https://www.youtube.com/watch?v=Pp9jSU2b9oM&list=PL5Q2soXY2Zi8_VVChACnON4sfh2bJ5IrD&index=159

More on PRIM Benchmarks

- Juan Gomez-Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu,
"Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture"

*Preprint in **arXiv**, 9 May 2021.*

[[arXiv preprint](#)]

[[PrIM Benchmarks Source Code](#)]

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

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

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

[[SAFARI Live Seminar Slides \(pptx\) \(pdf\)](#)]

[[SAFARI Live Seminar Video](#) (2 hrs 57 mins)]

[[Lightning Talk Video](#) (3 minutes)]

UPMEM PIM System Summary & Analysis

- Juan Gomez-Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu,
"Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware"
*Invited Paper at Workshop on Computing with Unconventional Technologies (**CUT**), Virtual, October 2021.*
[[arXiv version](#)]
[[PrIM Benchmarks Source Code](#)]
[[Slides \(pptx\)](#) ([pdf](#))]
[[Talk Video](#) (37 minutes)]
[[Lightning Talk Video](#) (3 minutes)]

Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware

Juan Gómez-Luna	Izzat El Hajj	Ivan Fernandez	Christina Giannoula	Geraldo F. Oliveira	Onur Mutlu
<i>ETH Zürich</i>	<i>American University of Beirut</i>	<i>University of Malaga</i>	<i>National Technical University of Athens</i>	<i>ETH Zürich</i>	<i>ETH Zürich</i>

Results So Far (2021-2022)

- DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks [IEEE Access 2021]
- Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System [IEEE Access 2022]
- Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks [PACT 2021]
- Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Cooperation [ICDE 2022]
- FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications [IEEE Micro 2021]

Data-Centric 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^{†◇}

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*

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

Amirali Boroumand

Saugata Ghose

Berkin Akin

Ravi Narayanaswami

Geraldo F. Oliveira

Xiaoyu Ma

Eric Shiu

Onur Mutlu

PACT 2021

SAFARI

Carnegie Mellon



UNIVERSITY OF
ILLINOIS
URBANA-CHAMPAIGN



ETH zürich

Executive Summary

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

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

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

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

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

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

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

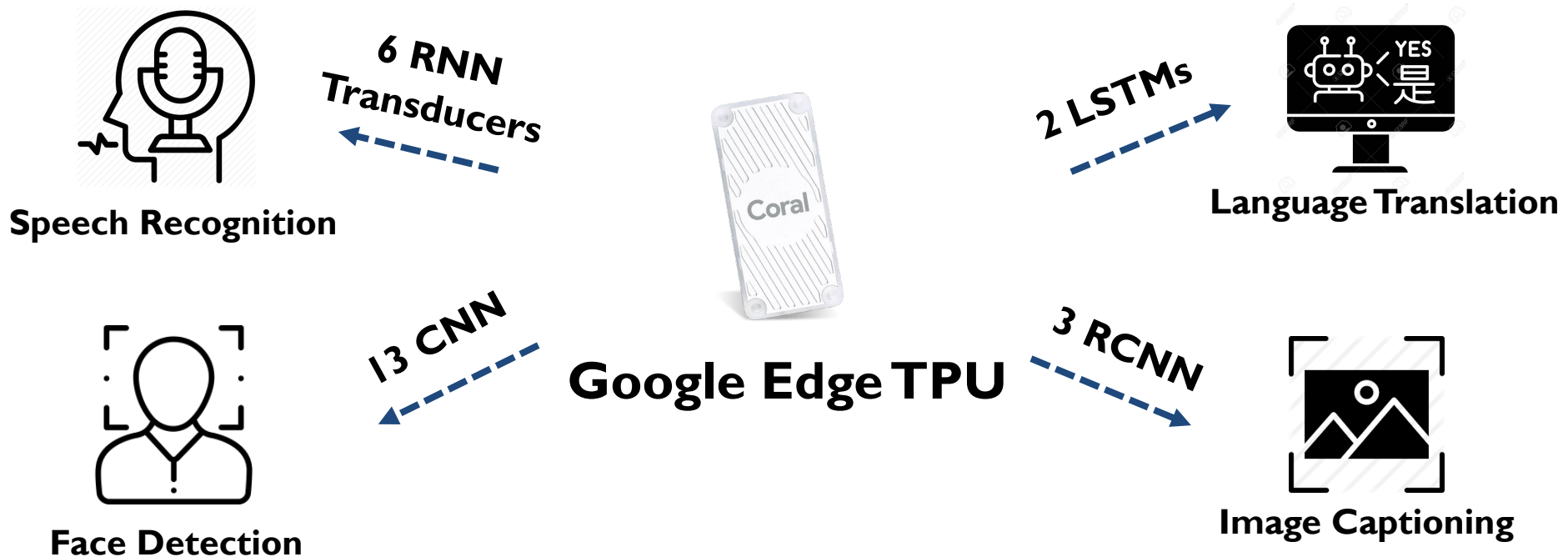
- Mensa consists of heterogeneous accelerators whose dataflow and hardware are specialized for specific families of layers

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

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

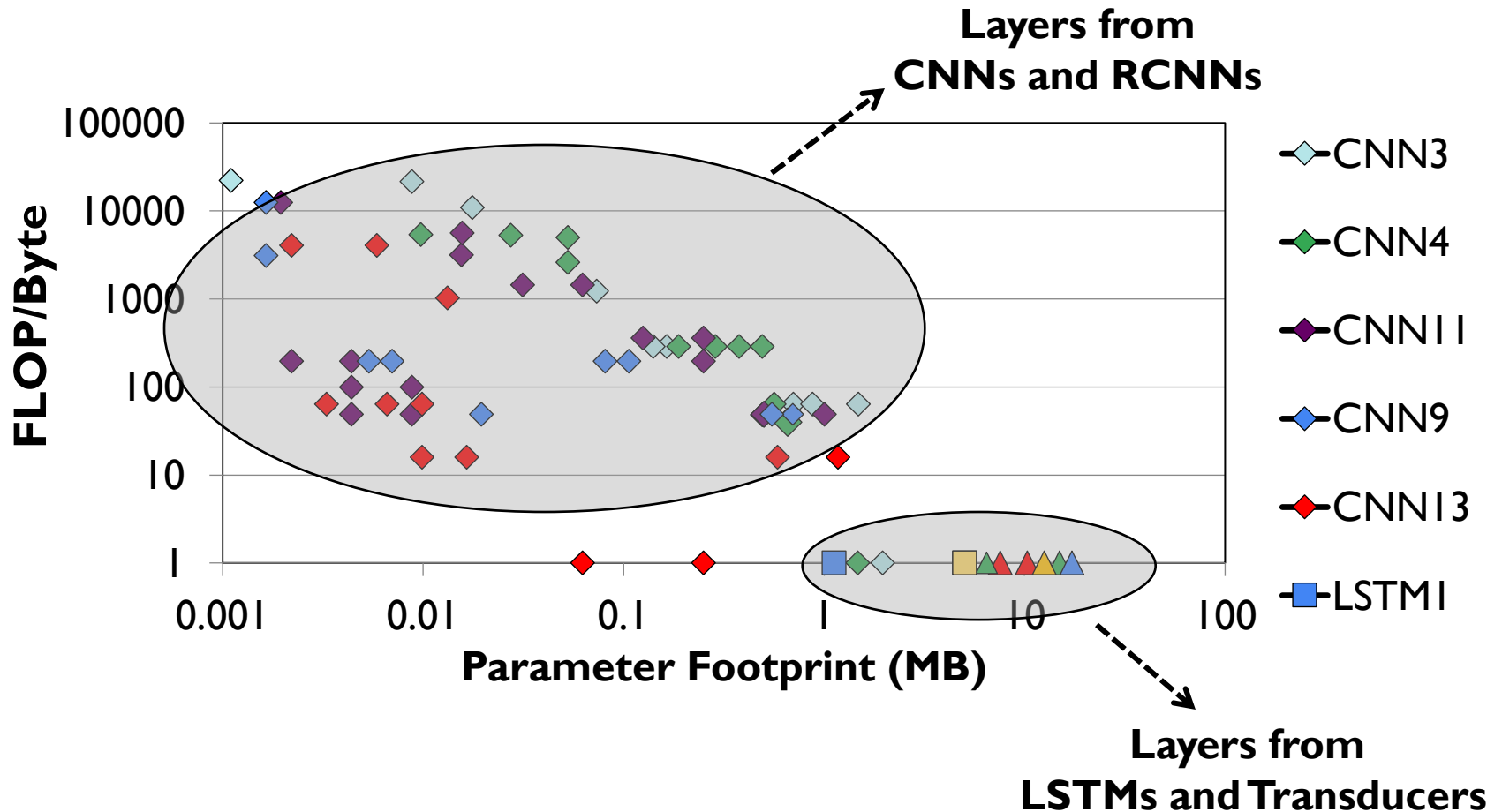
Google Edge NN Models

We analyze inference execution using 24 edge NN models



Diversity Across the Models

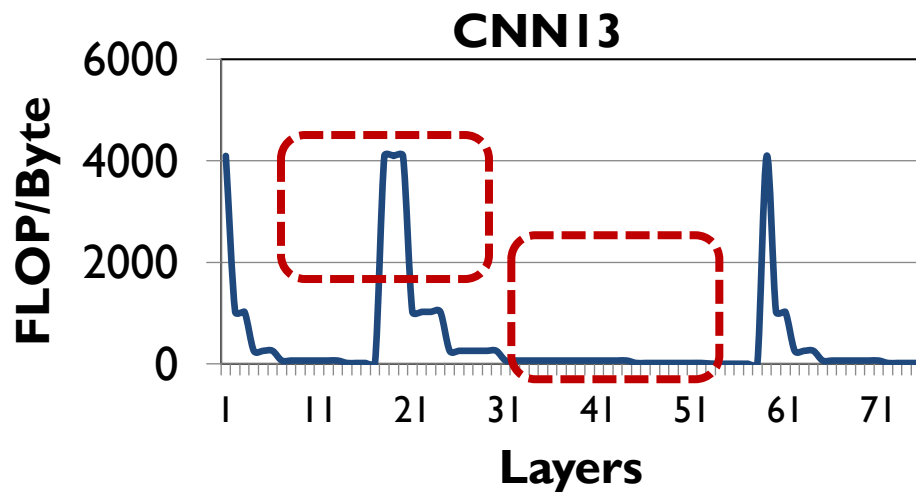
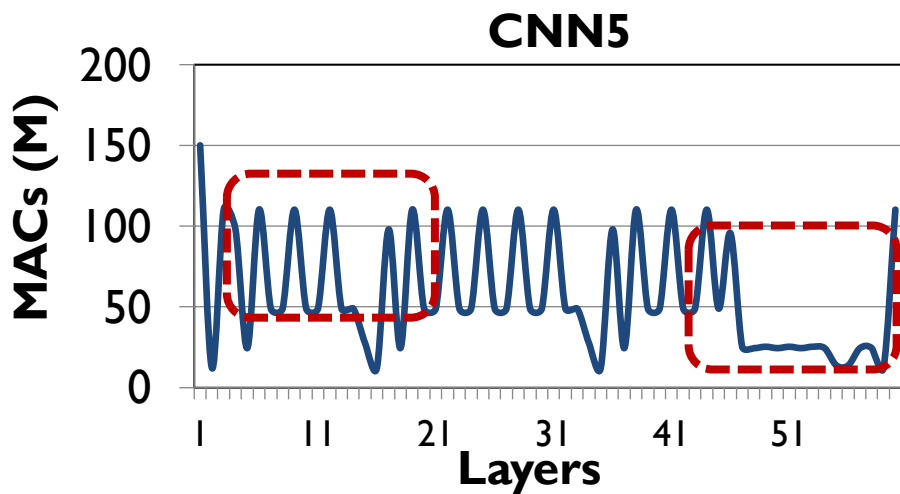
Insight 1: there is **significant variation** in terms of layer characteristics **across the models**



Diversity Within the Models

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

For example, our analysis of edge **CNN** models shows:

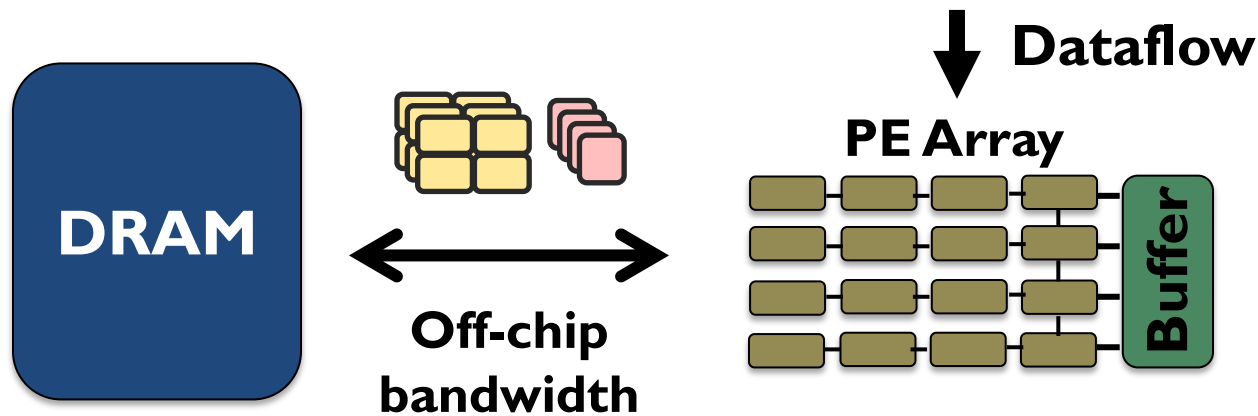


Variation in **MAC intensity**: up to **200x** across layers

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

Root Cause of Accelerator Challenges

The **key components** of Google Edge TPU are completely **oblivious** to **layer heterogeneity**



Edge accelerators typically take **a monolithic** approach: equip the accelerator with **an over-provisioned PE array** and on-chip buffer, **a rigid dataflow**, and **fixed off-chip bandwidth**



While this approach might work for a specific group of layers, it fails to efficiently execute inference across a wide variety of edge models

Mensa Framework

Goal: design an edge accelerator that can efficiently run inference across **a wide range of different models** and **layers**

Instead of running the entire NN model on
a monolithic accelerator:

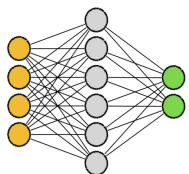


Mensa: a new acceleration framework for edge NN inference

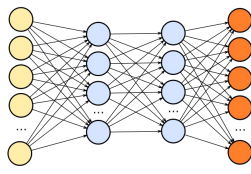
Mensa High-Level Overview

Edge TPU Accelerator

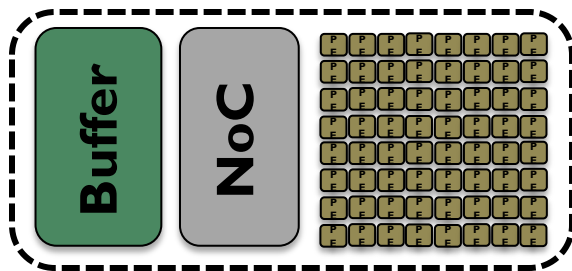
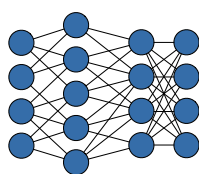
Model A



Model B



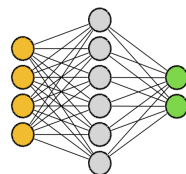
Model C



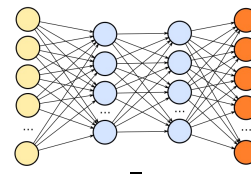
Monolithic Accelerator

Mensa

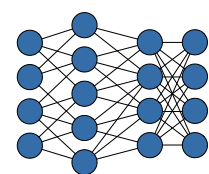
Model A



Model B



Model C

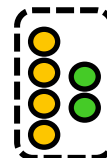


Runtime

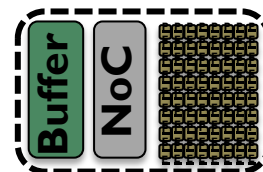
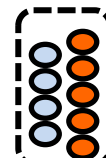
Family 1



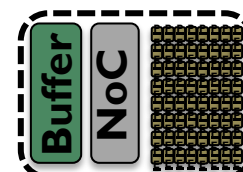
Family 2



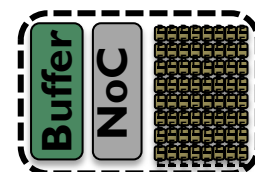
Family 3



Acc. 1



Acc. 2

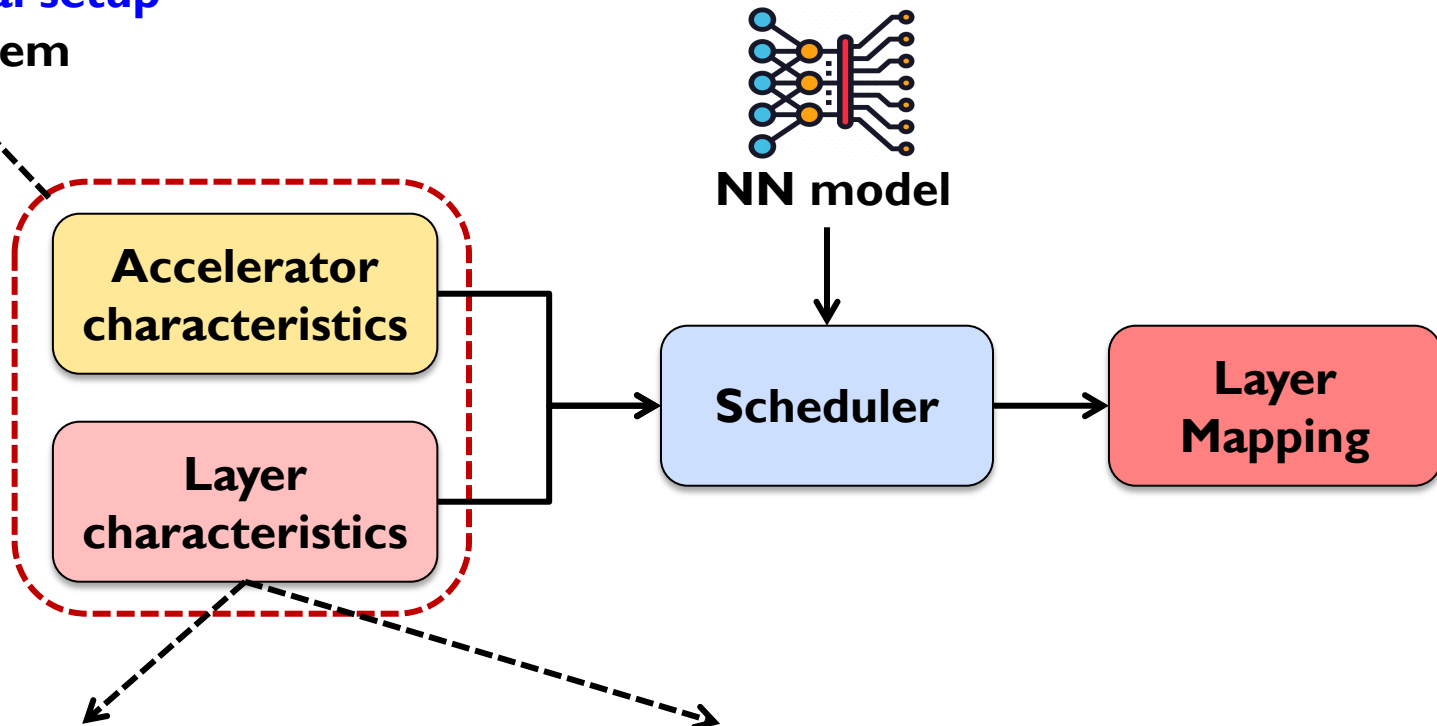


Acc. 3

Mensa Runtime Scheduler

The **goal** of Mensa's software **runtime scheduler** is to **identify** **which accelerator** each **layer** in an NN model should run on

Generated **once**
during **initial setup**
of a system



Each of the accelerators
caters to
a specific family of layers

Layers tend to **group**
together into a small
number of **families**

Mensa Runtime Scheduler

The **goal** of Mensa's software **runtime scheduler** is to **identify** which accelerator each **layer** in an NN model should run on

Generated **once**
during **initial setup**

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*

Layer
characteristics

Each of the accelerators
caters to
a specific family of layers

Layers tend to **group**
together into a small
number of **families**

SAFARI

Mensa: Highly-Efficient ML Inference

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

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

Amirali Boroumand^{†◇}

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*

Results So Far (2021-2022)

- DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks [IEEE Access 2021]
- Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System [IEEE Access 2022]
- Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks [PACT 2021]
- Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Cooperation [ICDE 2022]
- FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications [IEEE Micro 2021]

Accelerating HTAP Database Systems

- Appears in ICDE 2022

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

Amirali Boroumand[†]
[†]*Google*

Saugata Ghose[◇]
[◇]*Univ. of Illinois Urbana-Champaign*

Geraldo F. Oliveira[‡]
[‡]*ETH Zürich*

Onur Mutlu[‡]

<https://arxiv.org/pdf/2204.11275.pdf>

Results So Far (2021-2022)

- DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks [IEEE Access 2021]
- Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System [IEEE Access 2022]
- Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks [PACT 2021]
- Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Cooperation [ICDE 2022]
- FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications [IEEE Micro 2021]

FPGA-based Processing Near Memory

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

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

Gagandeep Singh[◇] Mohammed Alser[◇] Damla Senol Cali[✕]

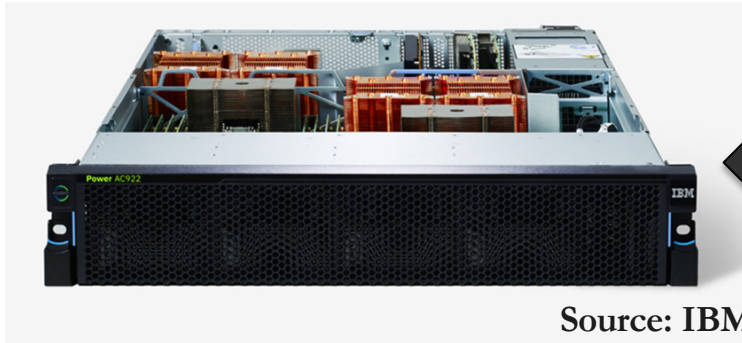
Dionysios Diamantopoulos[▽] Juan Gómez-Luna[◇]

Henk Corporaal[★] Onur Mutlu^{◇✕}

[◇]*ETH Zürich* [✕]*Carnegie Mellon University*

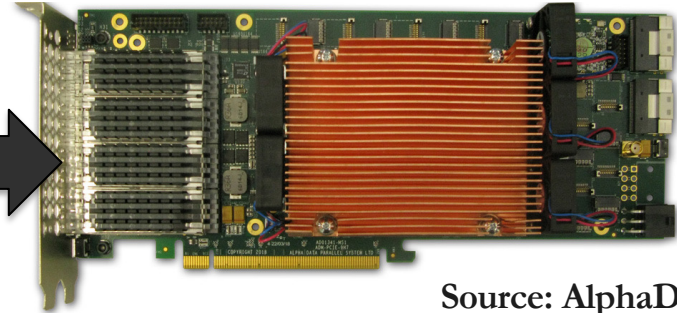
[★]*Eindhoven University of Technology* [▽]*IBM Research Europe*

Near-Memory Acceleration Using FPGAs



Source: IBM

IBM POWER9 CPU



Source: AlphaData

HBM-based FPGA board

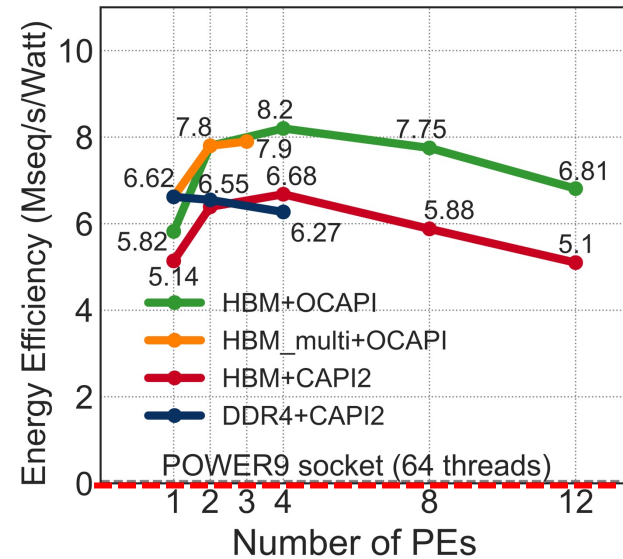
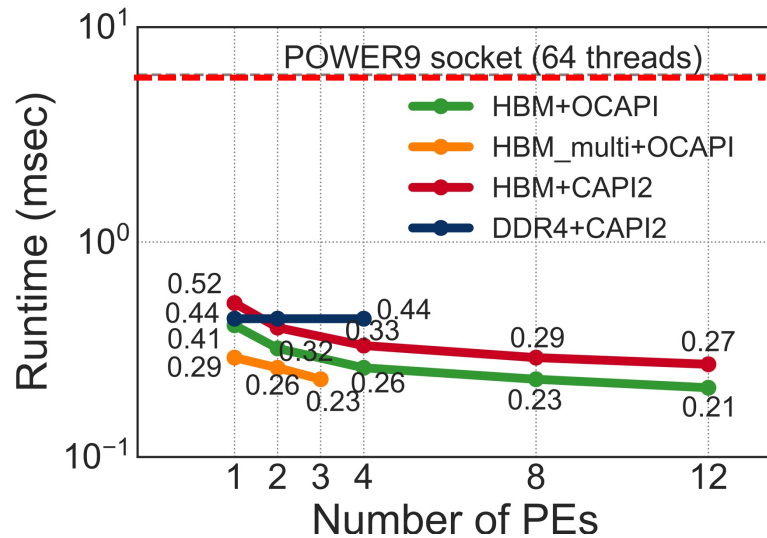
Near-HBM FPGA-based accelerator

Two communication technologies: CAPI2 and OCAPI

Two memory technologies: DDR4 and HBM

Two workloads: Weather Modeling and Genome Analysis

Performance & Energy Greatly Improve

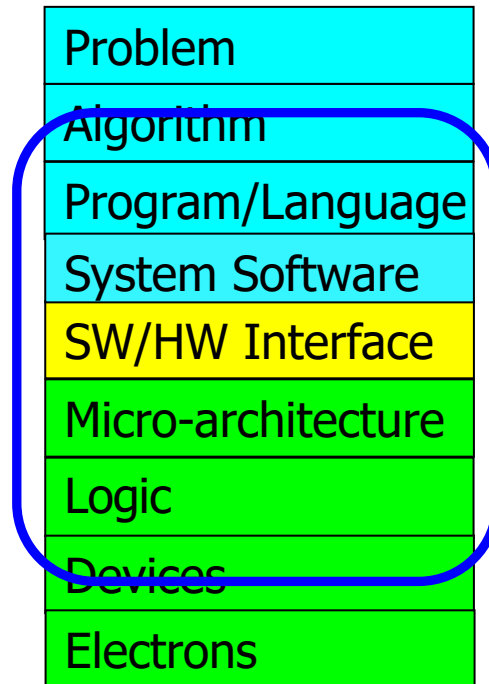


5-27× performance vs. a 16-core (64-thread) IBM POWER9 CPU

12-133× energy efficiency vs. a 16-core (64-thread) IBM POWER9 CPU

HBM alleviates memory bandwidth contention vs. DDR4

We Need to Revisit the Entire Stack



We can get there step by step

Coming Up: SpMV on Real PIM Systems

- To appear in SIGMETRICS 2022

***SparseP*: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Systems**

CHRISTINA GIANNOULA, ETH Zürich, Switzerland and National Technical University of Athens, Greece

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

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

NECTARIOS KOZIRIS, National Technical University of Athens, Greece

GEORGIOS GOUMAS, National Technical University of Athens, Greece

ONUR MUTLU, ETH Zürich, Switzerland

<https://arxiv.org/pdf/2201.05072.pdf>

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

Coming Up: SpMV on Real PIM Systems



The video player displays a presentation slide for 'SparseP'. At the top right, a small video feed shows a woman, identified as Christina Giannoula. The slide features a green network diagram above the title 'SparseP' in large blue letters. Below the title is the subtitle 'Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures'. The presenter's name, Christina Giannoula, is listed in red, followed by the co-authors: Ivan Fernandez, Juan Gomez-Luna, Nectarios Koziris, Georgios Goumas, and Onur Mutlu. The bottom of the slide includes logos for SAFARI, ETH zürich, CSLab, and several university seals. The video player controls at the bottom show a play button, a progress bar at 0:02 / 55:25, and various icons for volume, settings, and full screen.

Processing-in-Memory Course: Lecture 11: SpMV on a Real PIM Architecture - Spring 2022

149 views • Streamed live on May 19, 2022

👍 12 🗑 DISLIKE ➦ SHARE ⬇ DOWNLOAD ✂ CLIP ≡+ SAVE ...



Onur Mutlu Lectures
25K subscribers

ANALYTICS

EDIT VIDEO

Coming Up: FPGA Framework for PuM

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

Ataberk Olgun^{§†}

Juan Gómez Luna[§]

Konstantinos Kanellopoulos[§]

Behzad Salami^{§*}

Hasan Hassan[§]

Oğuz Ergin[†]

Onur Mutlu[§]

[§]ETH Zürich

[†]TOBB ETÜ

^{*}BSC

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

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

Comp Arch (Fall'21)

- Lectures/Schedule
- Lecture Buzzwords
- Readings
- HWs
- Labs
- Exams
- Related Courses
- Tutorials

- Computer Architecture FS20: Course Webpage
- Computer Architecture FS20: Lecture Videos
- Digitaltechnik SS21: Course Webpage
- Digitaltechnik SS21: Lecture Videos
- Moodle
- HotCRP
- Verilog Practice Website (HDLBits)

Fall 2021 Edition:

- <https://safari.ethz.ch/architecture/fall2021/doku.php?id=schedule>

Fall 2020 Edition:

- <https://safari.ethz.ch/architecture/fall2020/doku.php?id=schedule>

Youtube Livestream (2021):

- https://www.youtube.com/watch?v=4yfkM_5EFg0&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF

Youtube Livestream (2020):

- <https://www.youtube.com/watch?v=c3mPdZA-Fmc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN>

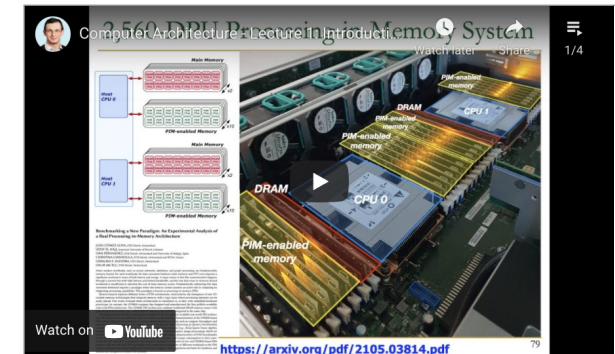
Master's level course

- Taken by Bachelor's/Masters/PhD students
- Cutting-edge research topics + fundamentals in Computer Architecture
- 5 Simulator-based Lab Assignments
- Potential research exploration
- Many research readings

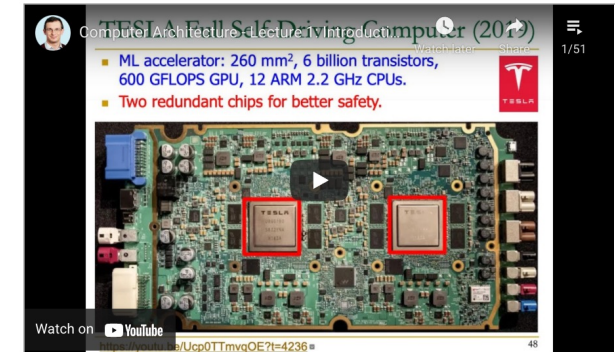
<https://www.youtube.com/onurmutlulectures>

Lecture Video Playlist on YouTube

Livestream Lecture Playlist



Recorded Lecture Playlist



Fall 2021 Lectures & Schedule

Week	Date	Livestream	Lecture	Readings	Lab	HW
W1	30.09 Thu.	YouTube Live	L1: Introduction and Basics (PDF) (PPT)	Required Mentioned	Lab 1 Out	HW 0 Out
	01.10 Fri.	YouTube Live	L2: Trends, Tradeoffs and Design Fundamentals (PDF) (PPT)	Required Mentioned		
W2	07.10 Thu.	YouTube Live	L3a: Memory Systems: Challenges and Opportunities (PDF) (PPT)	Described Suggested		HW 1 Out
			L3b: Course Info & Logistics (PDF) (PPT)			
			L3c: Memory Performance Attacks (PDF) (PPT)	Described Suggested		
	08.10 Fri.	YouTube Live	L4a: Memory Performance Attacks (PDF) (PPT)	Described Suggested	Lab 2 Out	
			L4b: Data Retention and Memory Refresh (PDF) (PPT)	Described Suggested		
			L4c: RowHammer (PDF) (PPT)	Described Suggested		

PIM Course (Fall'21)

Fall 2021 Edition:

- https://safari.ethz.ch/projects_and_seminars/fall2021/doku.php?id=processing_in_memory

Youtube Livestream:

- <https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi-841fUYYUK9EsXKhQKRPyX>

Project course

- Taken by Bachelor's/Master's students
- Processing-in-Memory lectures
- Hands-on research exploration
- Many research readings

PIM Review and Open Problems
Processing in Memory Course: Meeting 1: Ex...

Watch later Share 1/10

A Modern Primer on Processing in Memory

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

SAFARI Research Group

^aETH Zürich
^bCarnegie Mellon University
^cUniversity of Illinois at Urbana-Champaign
^dKing Mongkut's University of Technology North Bangkok

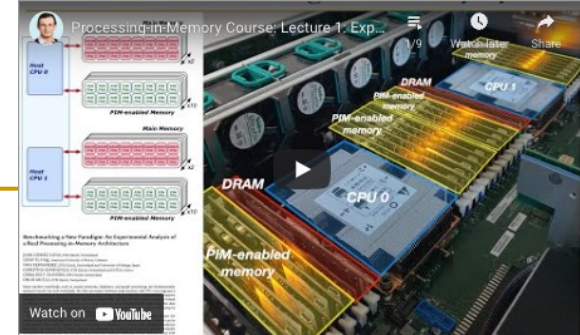
Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun, "A Modern Primer on Processing in Memory" Invited Book Chapter in *Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann*, Springer, to be published in 2021.

Watch on YouTube <https://arxiv.org/pdf/1903.03988.pdf> 108

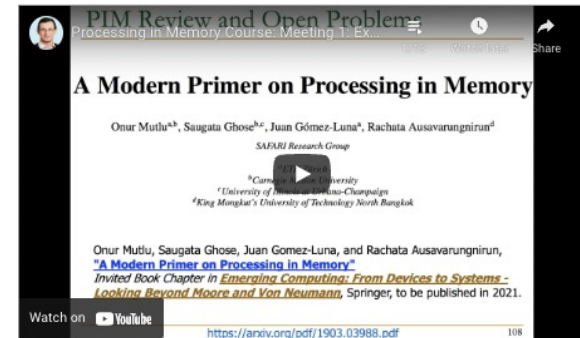
Fall 2021 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	05.10 Tue.	YouTube Live	M1: P&S PIM Course Presentation PDF (PDF) PPT (PPT)	Required Materials Recommended Materials	HW 0 Out
W2	12.10 Tue.	YouTube Live	M2: Real-World PIM Architectures PDF (PDF) PPT (PPT)		
W3	19.10 Tue.	YouTube Live	M3: Real-World PIM Architectures II PDF (PDF) PPT (PPT)		
W4	26.10 Tue.	YouTube Live	M4: Real-World PIM Architectures III PDF (PDF) PPT (PPT)		
W5	02.11 Tue.	YouTube Live	M5: Real-World PIM Architectures IV PDF (PDF) PPT (PPT)		
W6	09.11 Tue.	YouTube Live	M6: End-to-End Framework for Processing-using-Memory PDF (PDF) PPT (PPT)		
W7	16.11 Tue.	YouTube Live	M7: How to Evaluate Data Movement Bottlenecks PDF (PDF) PPT (PPT)		
W8	23.11 Tue.	YouTube Live	M8: Programming PIM Architectures PDF (PDF) PPT (PPT)		
W9	30.11 Tue.	YouTube Live	M9: Benchmarking and Workload Suitability on PIM PDF (PDF) PPT (PPT)		
W10	07.12 Tue.	YouTube Live	M10: Bit-Serial SIMD Processing using DRAM PDF (PDF) PPT (PPT)		

PIM Course (Current)



Recorded Lecture Playlist



Spring 2022 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	10.03 Thu.	Live	M1: P&S PIM Course Presentation (PDF) (PPT)	Required Materials Recommended Materials	HW 0 Out
W2	15.03 Tue.		Hands-on Project Proposals		
	17.03 Thu.	Premiere	M2: Real-world PIM: UPMEM PIM (PDF) (PPT)		
W3	24.03 Thu.	Live	M3: Real-world PIM: Microbenchmarking of UPMEM PIM (PDF) (PPT)		
W4	31.03 Thu.	Live	M4: Real-world PIM: Samsung HBM-PIM (PDF) (PPT)		
W5	07.04 Thu.	Live	M5: How to Evaluate Data Movement Bottlenecks (PDF) (PPT)		
W6	14.04 Thu.	Live	M6: Real-world PIM: SK Hynix AiM (PDF) (PPT)		
W7	21.04 Thu.	Premiere	M7: Programming PIM Architectures (PDF) (PPT)		
W8	28.04 Thu.	Premiere	M8: Benchmarking and Workload Suitability on PIM (PDF) (PPT)		
W9	05.05 Thu.	Premiere	M9: Real-world PIM: Samsung AxDIMM (PDF) (PPT)		
W10	12.05 Thu.		M10: Real-world PIM: Alibaba HB-PNM (PDF) (PPT)		

Spring 2022 Edition:

- https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=processing_in_memory

Youtube Livestream:

- https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi-841fUYYUK9EsXKhQKRPyX

Project course

- Taken by Bachelor's/Master's students
- Processing-in-Memory lectures
- Hands-on research exploration
- Many research readings

Current EFCL Projects

- “A New Methodology and Open-Source Benchmark Suite for Evaluating Data Movement Bottlenecks: A Processing-in-Memory Case Study”
 - Data-centric
- “Machine-Learning-Assisted Intelligent Microarchitectures to Reduce Memory Access Latency”
 - Data-driven
- “Cross-layer Hardware/Software Techniques to Enable Powerful Computation and Memory Optimizations”
 - Data-aware

Machine-Learning-Assisted Intelligent Microarchitectures to Reduce Memory Access Latency

Rahul Bera, Gagandeep Singh,
Rakesh Nadig, Konstantinos Kanellopoulos,
Onur Mutlu

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

How do we start?

Data-Driven **(Self-Optimizing)** **Computing Architectures**

Results So Far (2021-2022)

- Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning [MICRO 2021]
- Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning [ISCA 2022]

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 Bera¹ Konstantinos Kanellopoulos¹ Anant V. Nori² Taha Shahroodi^{3,1}
Sreenivas Subramoney² Onur Mutlu¹

¹ETH Zürich

²Processor Architecture Research Labs, Intel Labs

³TU Delft



Pythia

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera, Konstantinos Kanellopoulos, Anant V. Nori,
Taha Shahroodi, Sreenivas Subramoney, Onur Mutlu

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



1

Mainly use one
program context info.
for prediction

2

Lack system
awareness

3

Lack in-silicon
customizability



Why prefetchers do
not perform well?





Pythia

Autonomously learns to prefetch using multiple program context information and system-level feedback

In-silicon customizable to change program context information or prefetching objective on the fly



Basics of Reinforcement Learning (RL)

- Algorithmic approach to learn to take an **action** in a given **situation** to maximize a numerical **reward**

Agent

Environment

- Agent stores **Q-values** for *every* state-action pair
 - **Expected return** for taking an action in a state
 - Given a state, selects action that provides **highest** Q-value

Brief Overview of Pythia

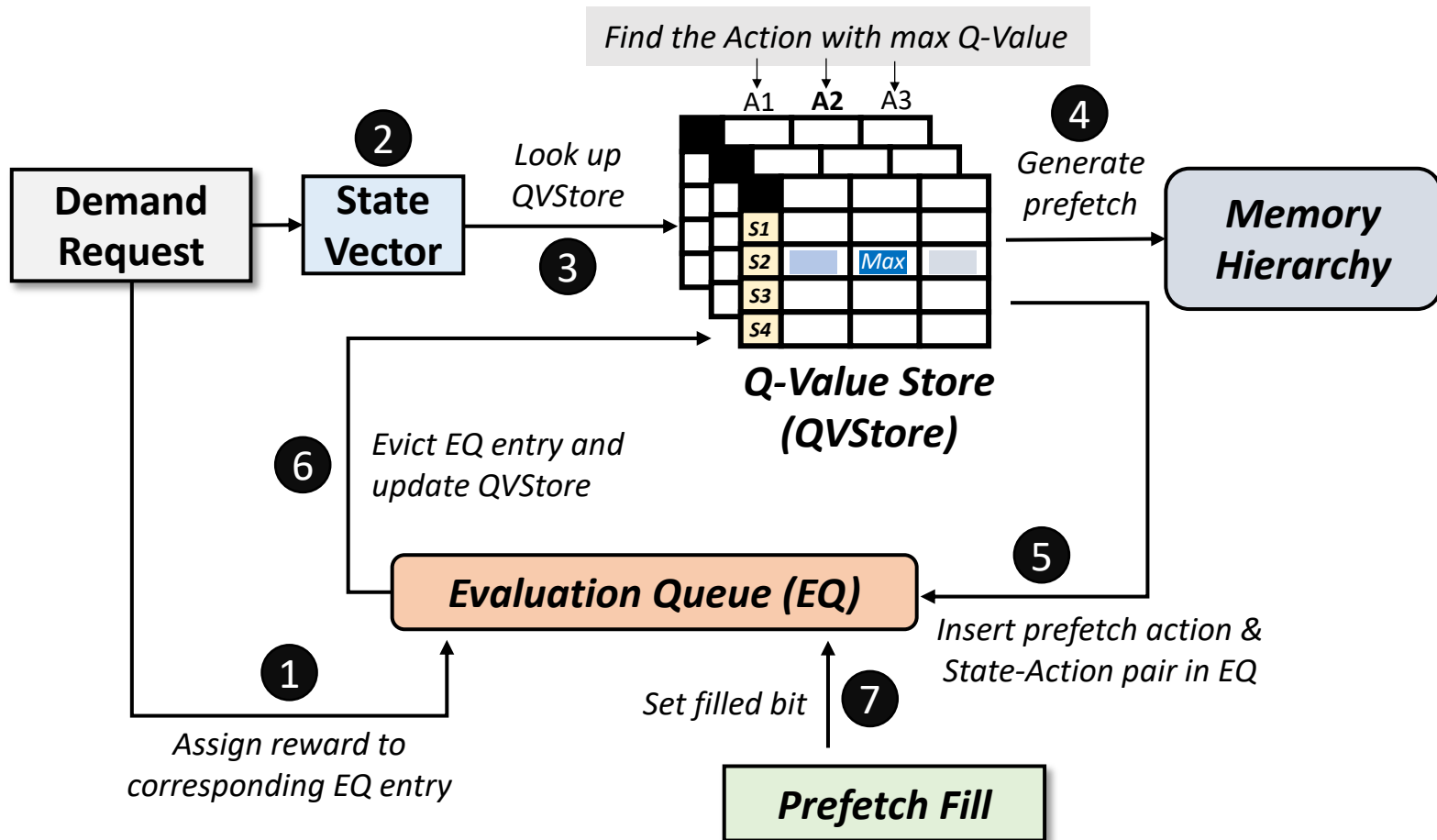
Pythia formulates prefetching as a **reinforcement learning** problem

Basic Pythia Configuration

- Derived from **automatic design-space exploration**
- **State:** 2 features
 - PC+Delta
 - Sequence of last-4 deltas
- **Actions:** 16 prefetch offsets
 - Ranging between -6 to +32. Including 0.
- **Rewards:**
 - $R_{AT} = +20$; $R_{AL} = +12$; $R_{NP-H} = -2$; $R_{NP-L} = -4$;
 - $R_{IN-H} = -14$; $R_{IN-L} = -8$; $R_{CL} = -12$

More Detailed Pythia Overview

- **Q-Value Store**: Records Q-values for *all* state-action pairs
- **Evaluation Queue**: A FIFO queue of recently-taken actions



Rigorous Evaluation Methodology

- **Champsim** [3] trace-driven simulator
- **150** single-core memory-intensive workload traces
 - SPEC CPU2006 and CPU2017
 - PARSEC 2.1
 - Ligra
 - Cloudsuite
- Homogeneous and heterogeneous multi-core mixes
- **Five** state-of-the-art prefetchers
 - SPP [Kim+, MICRO'16]
 - Bingo [Bakhshalipour+, HPCA'19]
 - MLOP [Shakerinava+, 3rd Prefetching Championship, 2019]
 - SPP+DSPatch [Bera+, MICRO'19]
 - SPP+PPF [Bhatia+, ISCA'20]

1

We evaluate Pythia using a wide-range of workloads

Pythia improves performance by

3.4% and 3.8% in single-core

7.7% and 9.6% in twelve-core

16.9% and 20.2% in memory bandwidth-constrained core

over state-of-the-art MLOP and Bingo prefetchers

2

We gain 7.8% more performance on top of basic Pythia configuration by simply customizing reward values for graph workloads

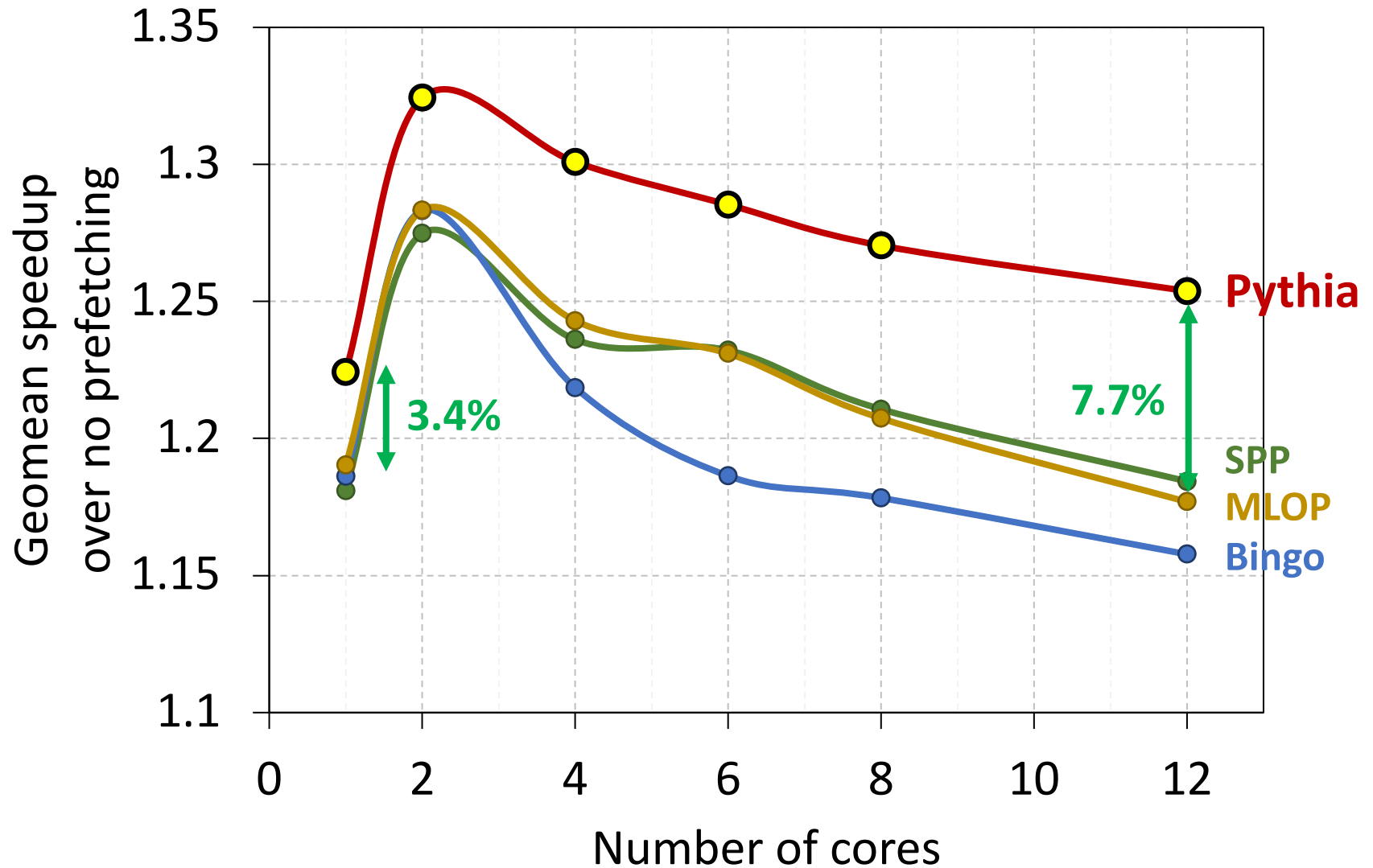
3

Realistic, practical implementation

No complex structures, only simple tables.

Only 1.03% area and 0.4% power of a desktop-class processor

Performance with Varying Core Count



Performance with Varying Core Count

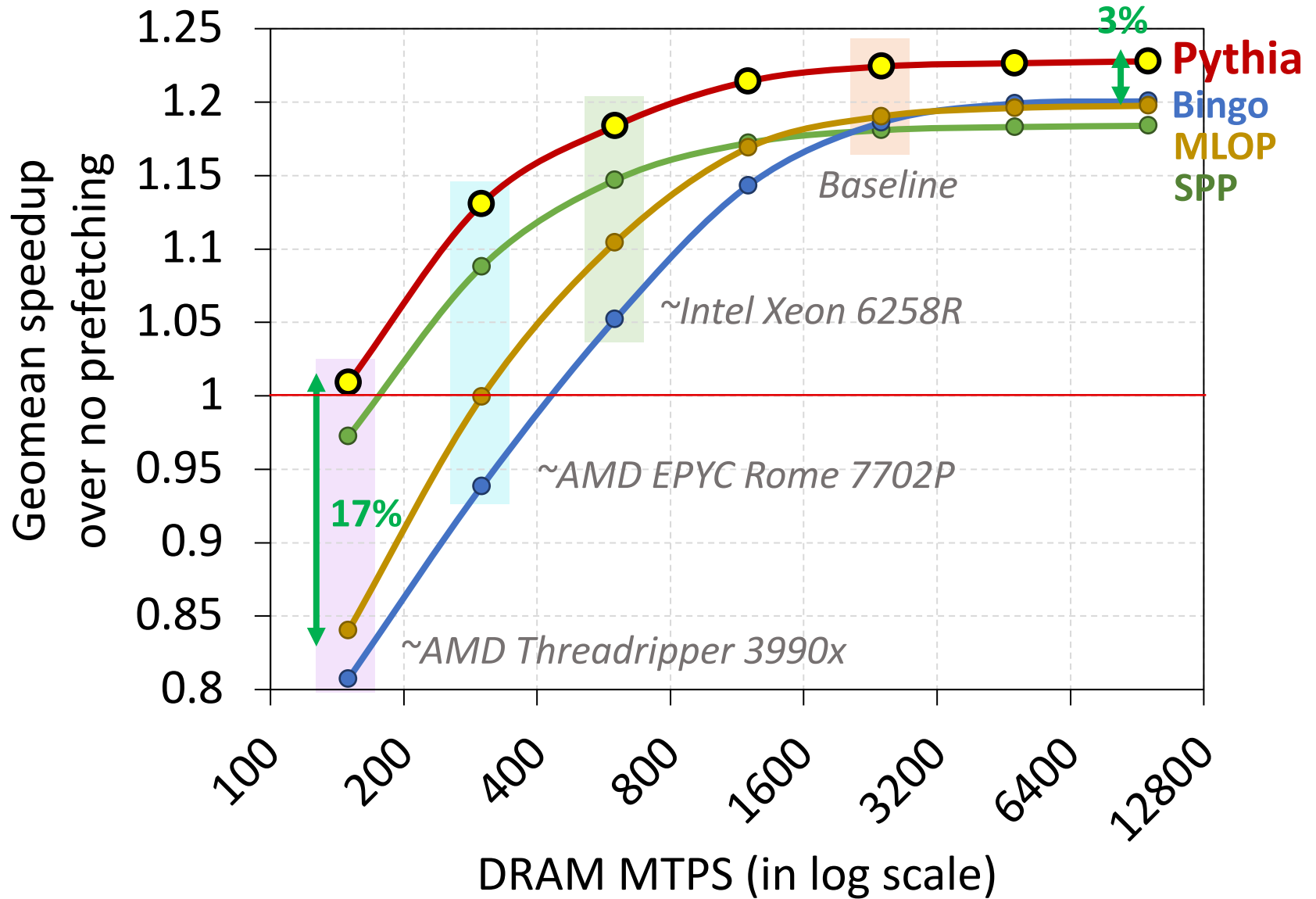


The graph shows performance on the y-axis (ranging from 1.1 to 1.35) against the number of cores on the x-axis (ranging from 0 to 12). Pythia (red line) starts at ~1.28 at 2 cores, peaks at ~1.32 at 4 cores, and then declines. Other models (blue, green, orange lines) show lower performance, with a green line showing a 3.4% gain at 8 cores. A vertical green line at 12 cores is labeled 'SDD'.

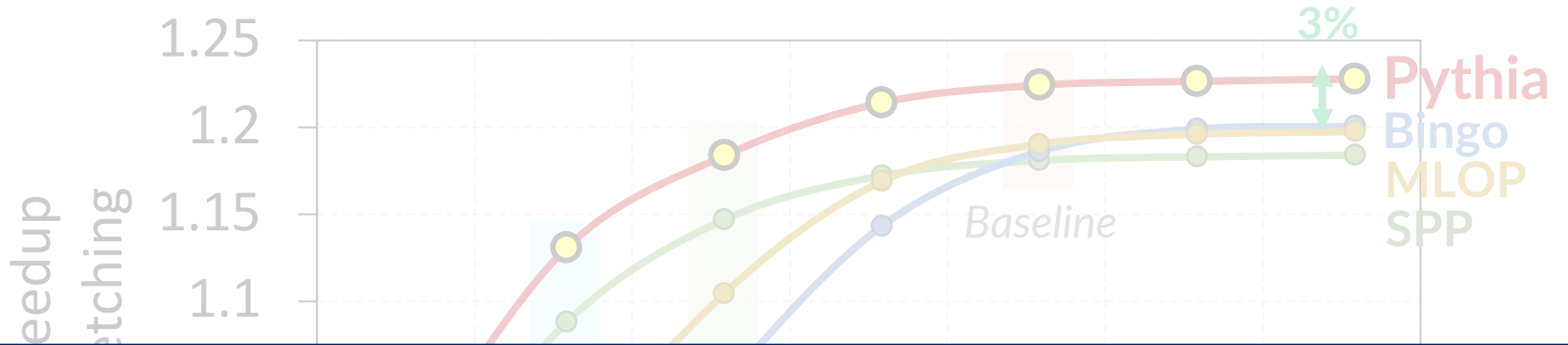
1. Pythia consistently provides the highest performance in **all core configurations**

2. Pythia's gain **increases with core count**

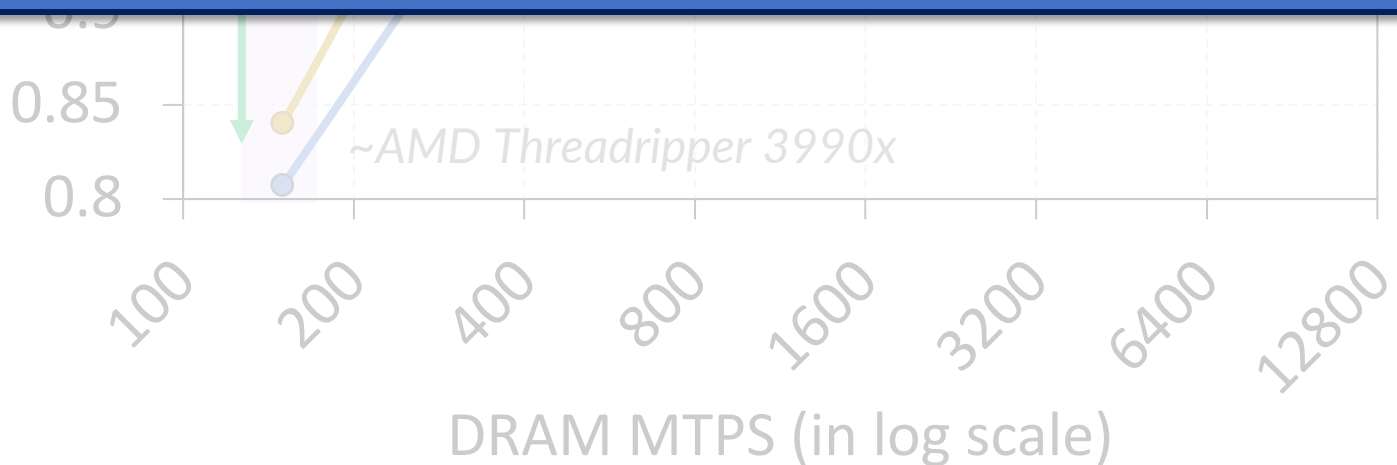
Performance with Varying DRAM Bandwidth



Performance with Varying DRAM Bandwidth



Pythia outperforms prior best prefetchers for a wide range of DRAM bandwidth configurations



A Lot More in the Paper

- Performance comparison with **unseen traces**
 - Pythia provides equally high performance benefits

• Comparison against **multi-level prefetchers**

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>

- **Performance sensitivity** towards different features and hyperparameter values

- Detailed single-core and four-core performance

Pythia is Open Source



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

- MICRO'21 **artifact evaluated**
- **Champsim source** code + **Chisel** modeling code
- **All traces** used for evaluation

The screenshot shows the GitHub repository for CMU-SAFARI/Pythia. The repository is public and has 3 unwatchers, 7 stars, and 2 forks. The main navigation bar includes links to Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The repository is currently on the master branch, with 1 branch and 5 tags. The commit history shows a recent update to the README by rahulbera 2 days ago, with 38 commits in total. The file list includes directories for branch, config, experiments, inc, prefetcher, replacement, scripts, src, tracer, and files for .gitignore, CITATION.cff, LICENSE, and LICENSE.champsim. The right sidebar contains an 'About' section describing Pythia as a customizable hardware prefetching framework, a link to the arXiv paper, and a list of related topics like machine-learning, reinforcement-learning, and computer-architecture. There are also links to the README, View license, and Cite this repository.

File	Commit Message	Time
branch	Initial commit for MICRO'21 artifact evaluation	2 months ago
config	Initial commit for MICRO'21 artifact evaluation	2 months ago
experiments	Added chart visualization in Excel template	2 months ago
inc	Updated README	6 days ago
prefetcher	Initial commit for MICRO'21 artifact evaluation	2 months ago
replacement	Initial commit for MICRO'21 artifact evaluation	2 months ago
scripts	Added md5 checksum for all artifact traces to verify download	2 months ago
src	Initial commit for MICRO'21 artifact evaluation	2 months ago
tracer	Initial commit for MICRO'21 artifact evaluation	2 months ago
.gitignore	Initial commit for MICRO'21 artifact evaluation	2 months ago
CITATION.cff	Added citation file	6 days ago
LICENSE	Updated LICENSE	2 months ago
LICENSE.champsim	Initial commit for MICRO'21 artifact evaluation	2 months ago



Pythia

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera, Konstantinos Kanellopoulos, Anant V. Nori,
Taha Shahroodi, Sreenivas Subramoney, Onur Mutlu

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



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)**

Results So Far (2021-2022)

- Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning [MICRO 2021]
- Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning [ISCA 2022]

Self-Optimizing Hybrid Storage Systems

- To appear in ISCA 2022

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

<https://arxiv.org/pdf/2205.07394.pdf>

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, Onur Mutlu

Executive Summary

Background: Hybrid storage systems (HSSs) complement different storage technologies to extend the overall capacity and reduce the system cost with minimal effect on the application performance

Problem: Accurately identify the performance-critical data of an application and placing it in the “best-fit” storage device. Three key shortcomings of prior data placement policies (heuristic-based and supervised learning-based) of hybrid storage systems:

- Lack of **adaptability**
- Lack of **device awareness (e.g., read/write latencies of each device)**
- Lack of **extensibility**

Goal: Develop a new, efficient, and high performance data-placement mechanism for hybrid storage systems that can:

- Dynamically derive an adaptive data-placement strategy by **continuously learning and adapting** to the **application and underlying device characteristics**
- **Easily extensible** to incorporate a wide range of hybrid storage configurations.

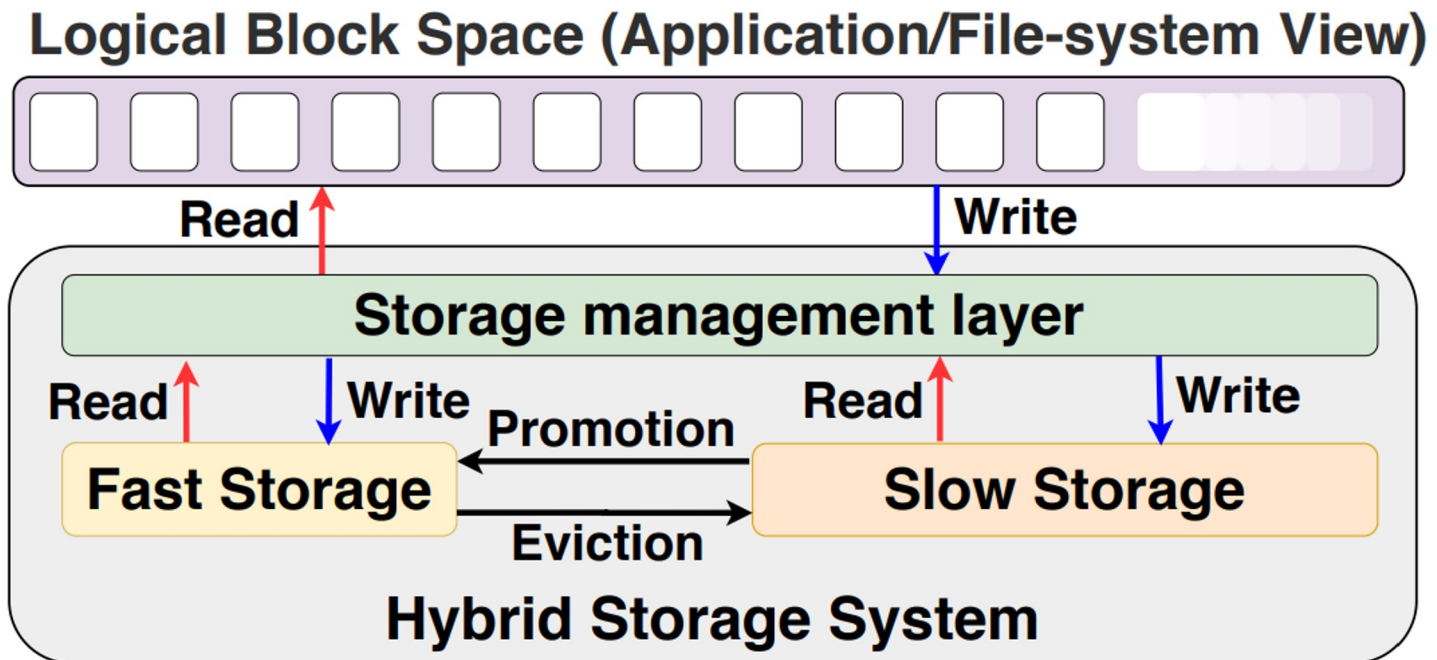
Key Idea: Sibyl, an online reinforcement learning-based self-optimizing mechanism for data placement that:

- **Dynamically learns** from past experiences and **continuously adapts** its policy to improve long-term performance by interacting with the hybrid storage system
- **Learns the asymmetry in the read/write latencies** present in modern hybrid storage devices while **taking into account** the **inherent characteristics of an application**

Key Results: Sibyl is evaluated on a real system with multiple device configurations

- Evaluated using a **wide range of workloads** from MSR Cambridge and Filebench
- In a performance (cost) optimized hybrid storage configuration, Sibyl provides up to **21.6% (19.9%)** performance improvement compared to prior data placement policies
- On a tri-hybrid storage system, Sibyl outperforms a heuristics-based policy by **23.9% -48.2%**
- Sibyl achieves **80%** performance of an oracle policy with storage overhead of **124.4 KiB**

Hybrid Storage Systems



Key Shortcomings of Prior Data Placement Techniques

We observe **three key shortcomings** that significantly limit performance benefits of data-placement techniques

Lack of **adaptability**

Lack of **device awareness**

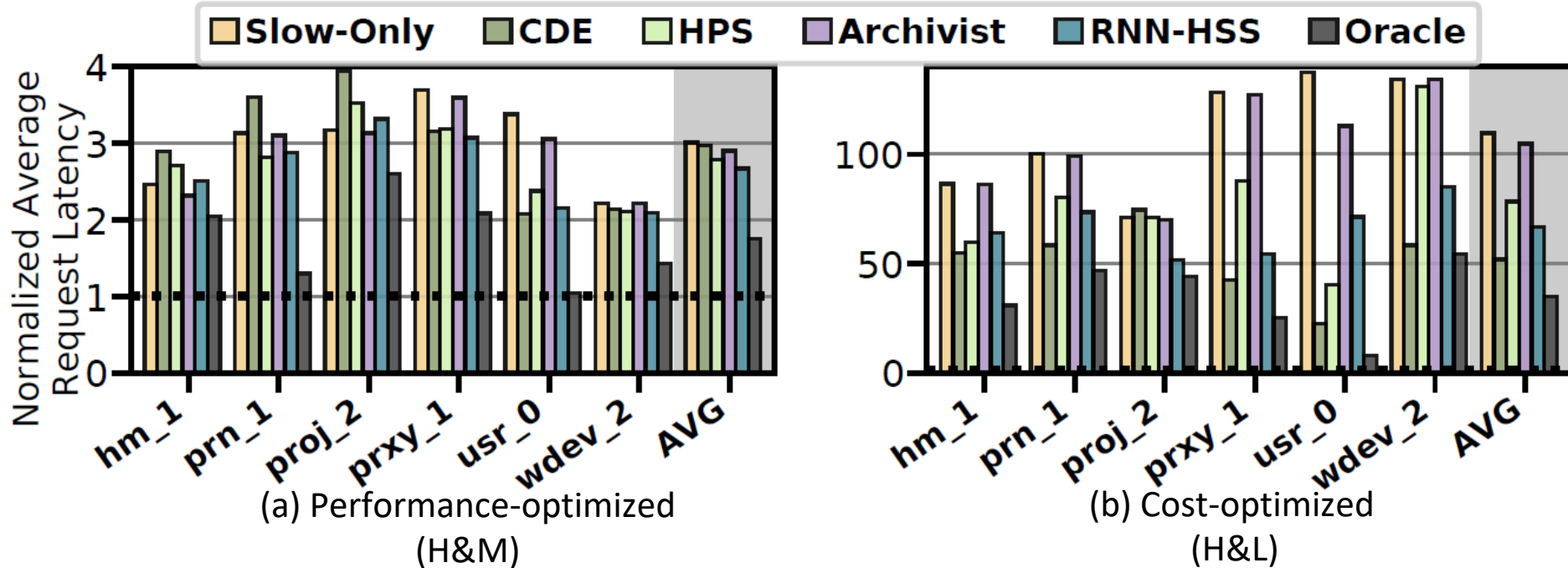
Lack of **extensibility**

Lack of Adaptability (1/2)

- Prior heuristic-based techniques consider only a **few characteristics** (e.g., access frequency) to perform data placement
- **Statically tuned characteristics** (based on **fixed thresholds**) are ineffective when used on a wide range of applications and system configurations
- Supervised learning techniques need **labeled data** and **frequent retraining** to adapt to varying workloads and system conditions

Prior techniques offer **41.1% lower performance** compared to an Oracle policy

Lack of Adaptability (2/2)



CDE shows an average performance gap of 41.1% (32.6%) to Oracle for H&M (H&L)

HPS shows an average performance gap of 37.2% (55.5%) to Oracle for H&M (H&L)

Lack of Device Awareness

Prior data placement techniques:

- **do not adapt** well to changes in underlying device characteristics (e.g., storage read latency)
- **do not consider the data migration cost** between storage devices while making a data placement decision
- **are highly inefficient** in hybrid storage systems that have devices with significantly different read/write latencies

Lack of Extensibility

- Prior data placement techniques are typically **designed** for a hybrid storage system with **only two storage devices**
- **Significant effort** is required to extend the data placement policies for more than two devices

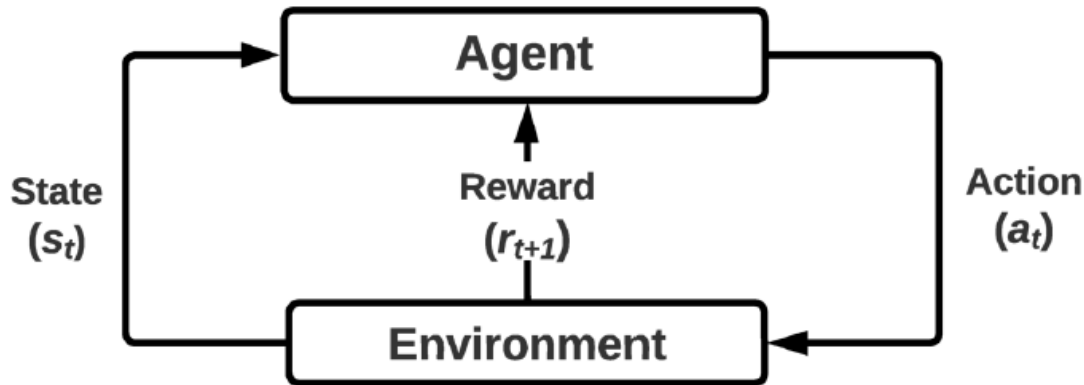
Compared to a RL-based solution, a heuristic-based policy provides **48.2% lower performance** when extended from two to three devices

Our Goal

A **data-placement mechanism** that can

- dynamically derive an adaptive data-placement strategy by **continuously learning** and **adapting** to the application and underlying device characteristics
- be **easily extended** to incorporate a wide range of hybrid storage configurations

Basics of Reinforcement Learning



- RL is a framework for decision making
 - An **autonomous agent observes** the current **state** of the environment
 - It **interacts** with the environment by taking **actions**
 - Agent is **rewarded** or **penalized** based on the consequences of its actions
 - Agent tries to maximize the cumulative reward

Applying RL to Data Placement

Key factors in applying RL for data placement in a hybrid storage system

- RL agent needs to be **aware of:**
 - **asymmetry in read/write latencies** of a storage device
 - **differences in latencies** across hybrid storage devices
 - **application access patterns**
- Data placement module should decide which actions to reward and penalize (credit assignment)
- Low implementation overhead

RL State

- Feature selection is performed to select only the most correlated features that affect data placement
- Divide the states into a small number of bins to reduce the state space

Feature	Description	# of bins	Encoding (bits)
$size_t$	Size of the requested page (in pages)	8	8
$type_t$	Type of the current request (read/write)	2	4
$intr_t$	Access interval of the requested page	64	8
cnt_t	Access count of the requested page	64	8
cap_t	Remaining capacity in the fast storage device	8	8
$curr_t$	Current placement of the requested page (fast/slow)	2	4

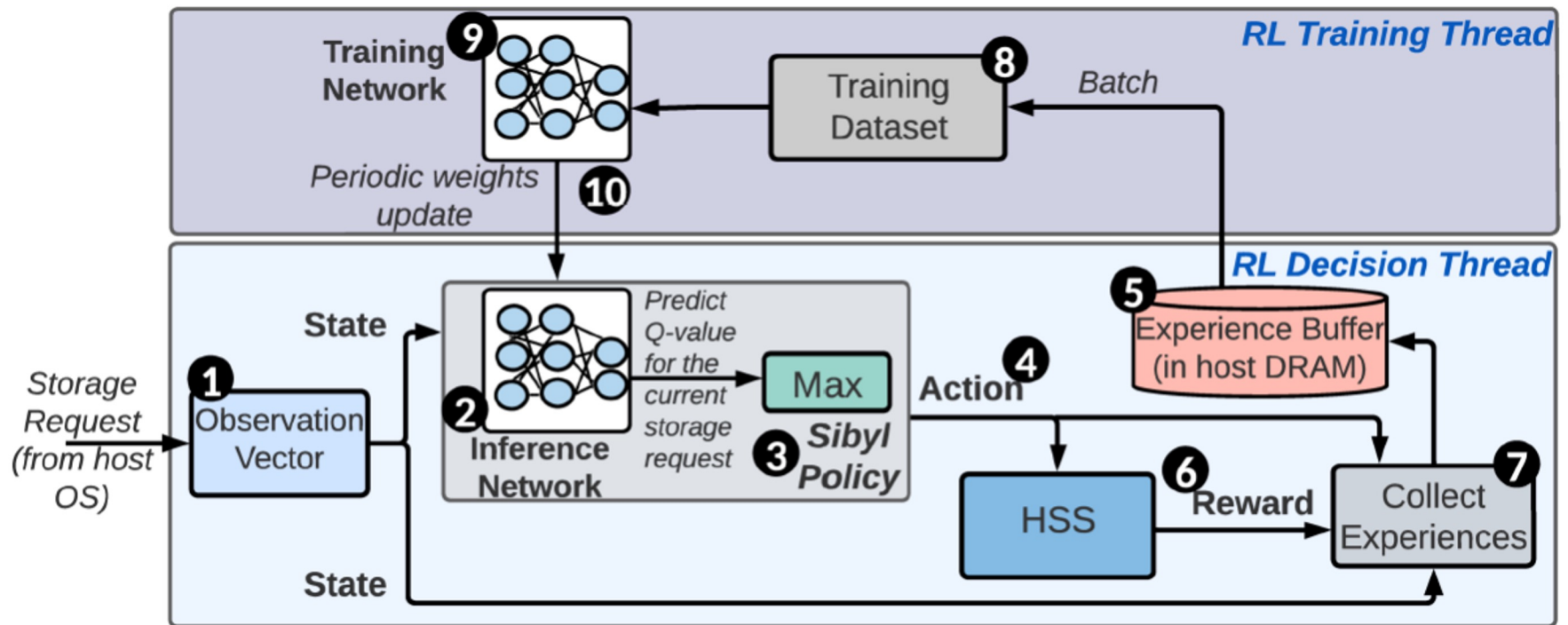
Reward

- For every action at time-step t , Sibyl gets a reward from the environment at time-step $t + 1$
- **Reward** acts as a **feedback** to the agent's past action
- **Request latency** faithfully captures the status of the hybrid storage system
- **Penalty** value is chosen to prevent the agent from aggressively servicing all the requests from the faster device

$$R = \begin{cases} \frac{1}{L_t} & \text{if no eviction} \\ \max(0, \frac{1}{L_t} - R_p) & \text{if an eviction happens} \end{cases}$$

L_t = latency of the request
 R_p = eviction penalty

Overview of Sibyl



The two threads run asynchronously to prevent training delay from affecting the inference time

Hyper-parameter Tuning

- Different hyper-parameter configurations were chosen using the design of experiments (DoE) technique

Hyper-parameter	Design Space	Chosen Value
Discount factor (γ)	0-1	0.9
Learning rate (α)	$1e^{-5} - 1e^0$	$1e^{-4}$
Exploration rate (ϵ)	0-1	0.001
Batch size	64-256	128
Experience buffer size (e_{EB})	10-10000	1000

Evaluation Methodology

- Evaluated on a **real system** with different hybrid storage configurations
- Hybrid storage system constitutes one contiguous logical block address space
- A custom block driver was implemented to manage the I/O requests to the storage devices
- We evaluate **three** different hybrid storage configurations
 - Performance-optimized (H&M)
 - Cost-optimized (H&L)
 - Tri-hybrid storage system

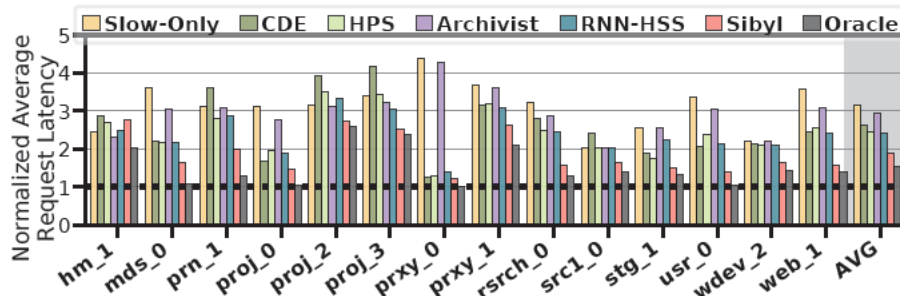
Evaluation Methodology

Host System	AMD Ryzen 7 2700G [146], 8-cores@3.5 GHz, 8×64/32 KiB L1-I/D, 4 MiB L2, 8 MiB L3, 16 GiB RDIMM DDR4 2666 MHz	
Storage Devices	Characteristics	
H: Intel Optane SSD P4800X [94]	375 GB, PCIe 3.0 NVMe, SLC, R/W: 2.4/2 GB/s, random R/W: 550000/500000 IOPS	
M: Intel SSD D3-S4510 [96]	1.92 TB, SATA TLC (3D), R/W: 550/510 MB/s, random R/W: 895000/21000 IOPS	
L: Seagate HDD ST1000DM010 [98]	1 TB, SATA 6Gb/s 7200 RPM Max. Sustained Transfer Rate: 210 MB/s	
L _{SSD} : ADATA SU630 SSD [99]	960 GB, SATA 6 Gb/s, TLC, Max R/W: 520/450 MB/s	
HSS Configurations	Fast Device	Slow Device
H&M (Performance-oriented)	high-end (H)	middle-end (M)
H&L (Cost-oriented)	high-end (H)	low-end (L)

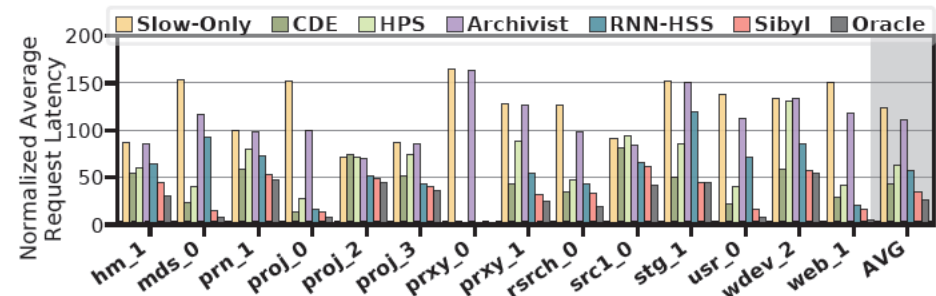
Evaluation Methodology

- 18 different workloads from MSR Cambridge and FileBench suites
- Sibyl is compared against **four baselines**
 - Heuristic-based policies
 - Cold data eviction (CDE) [Matsui et. al., "Design of Hybrid SSDs With Storage Class Memory and NAND Flash Memory," IEEE 2017]
 - History Page Scheduler (HPS) [Meswani et.al., "Heterogeneous Memory Architectures: A HW/SW Approach for Mixing Die-stacked and Off-package Memories," HPCA, 2015]
 - Supervised learning-based policies
 - Recurrent neural network (RNN)-based technique adapted from Kleio [Doudali et.al., "Kleio: A Hybrid Memory Page Scheduler with Machine Intelligence," HPDC, 2019]
 - Neural network-based classifier based on Archivist [Ren et.al., "Archivist: A Machine Learning Assisted Data Placement Mechanism for Hybrid Storage Systems," ICCD, 2019]

Latency Improvement



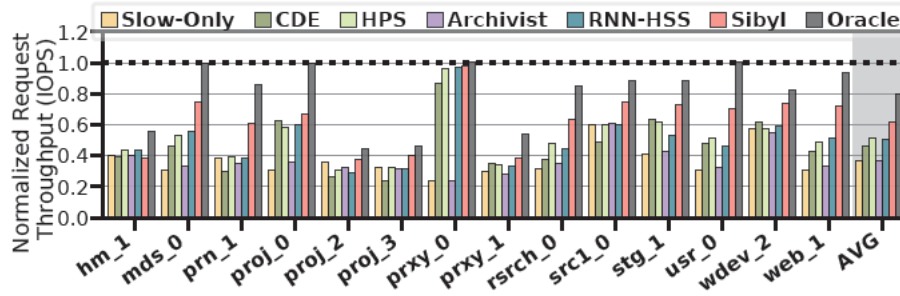
(a) Performance-optimized
(H&M)



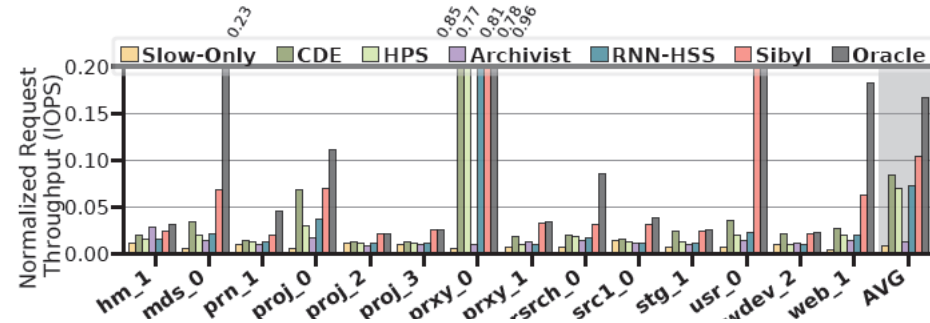
(b) Cost-optimized
(H&L)

Configuration	CDE	HPS	Archivist	RNN-HSS
H&M	28.1%	23.2%	36.1%	21.6%
H&L	19.9%	45.9%	68.8%	34.1%

Throughput Improvement



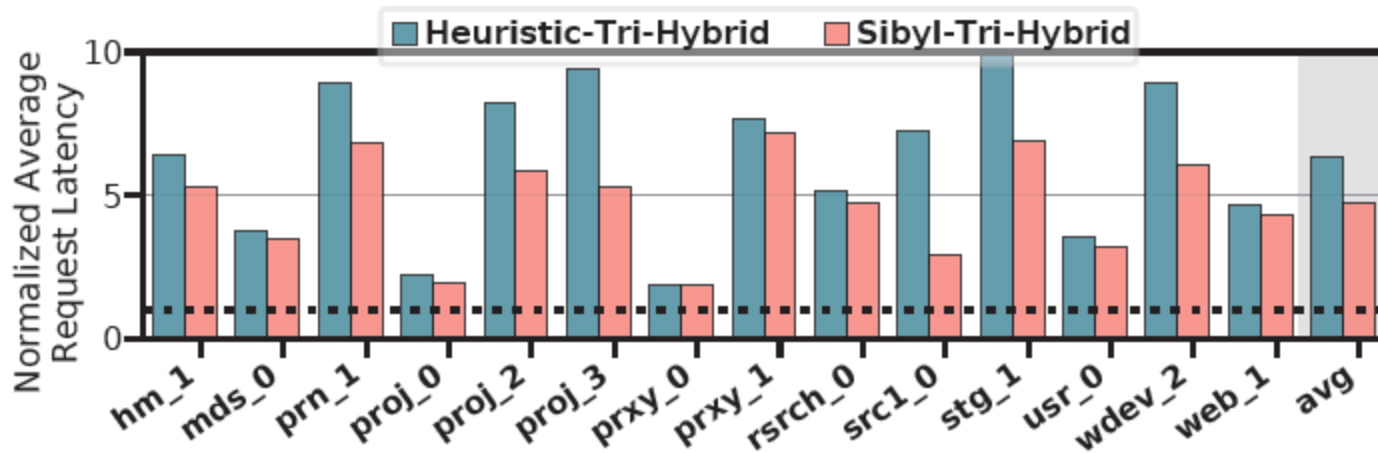
(a) Performance-optimized
(H&M)



(b) Cost-optimized
(H&L)

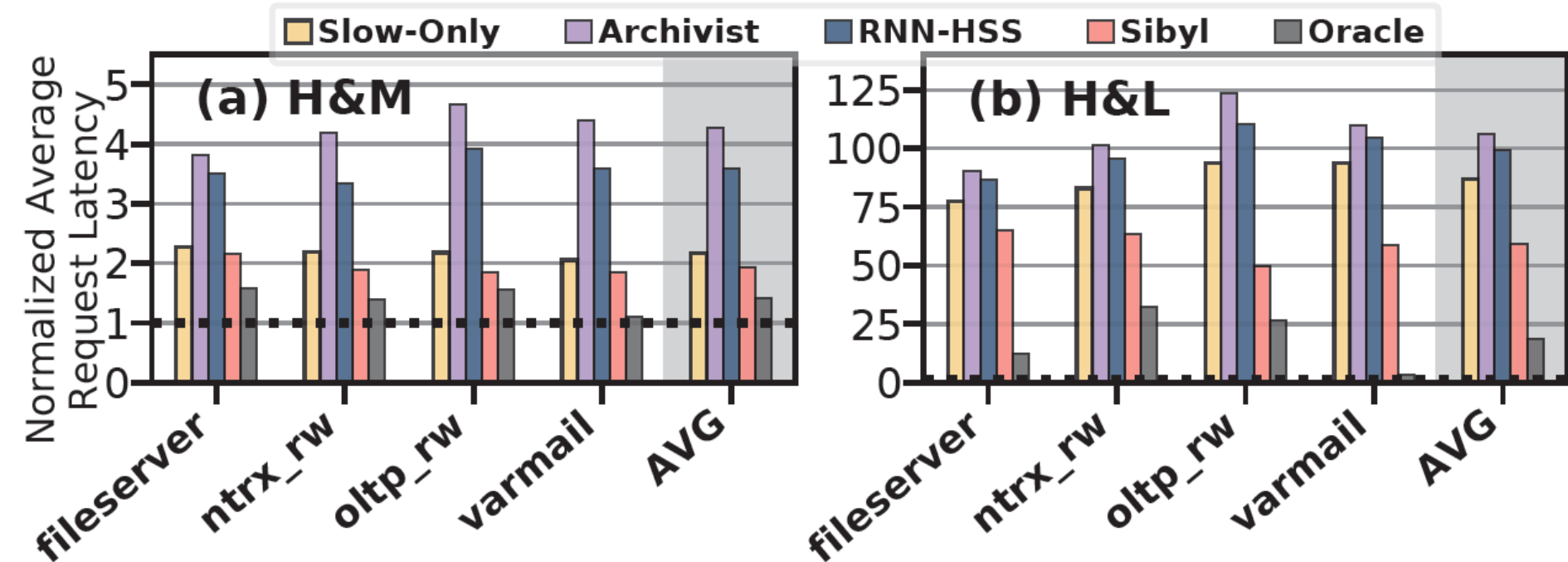
Configuration	CDE	HPS	Archivist	RNN-HSS
H&M	32.6%	21.9%	54.2%	22.7%
H&L	22.8%	49.1%	86.9%	41.9%

Latency in Tri-Hybrid System



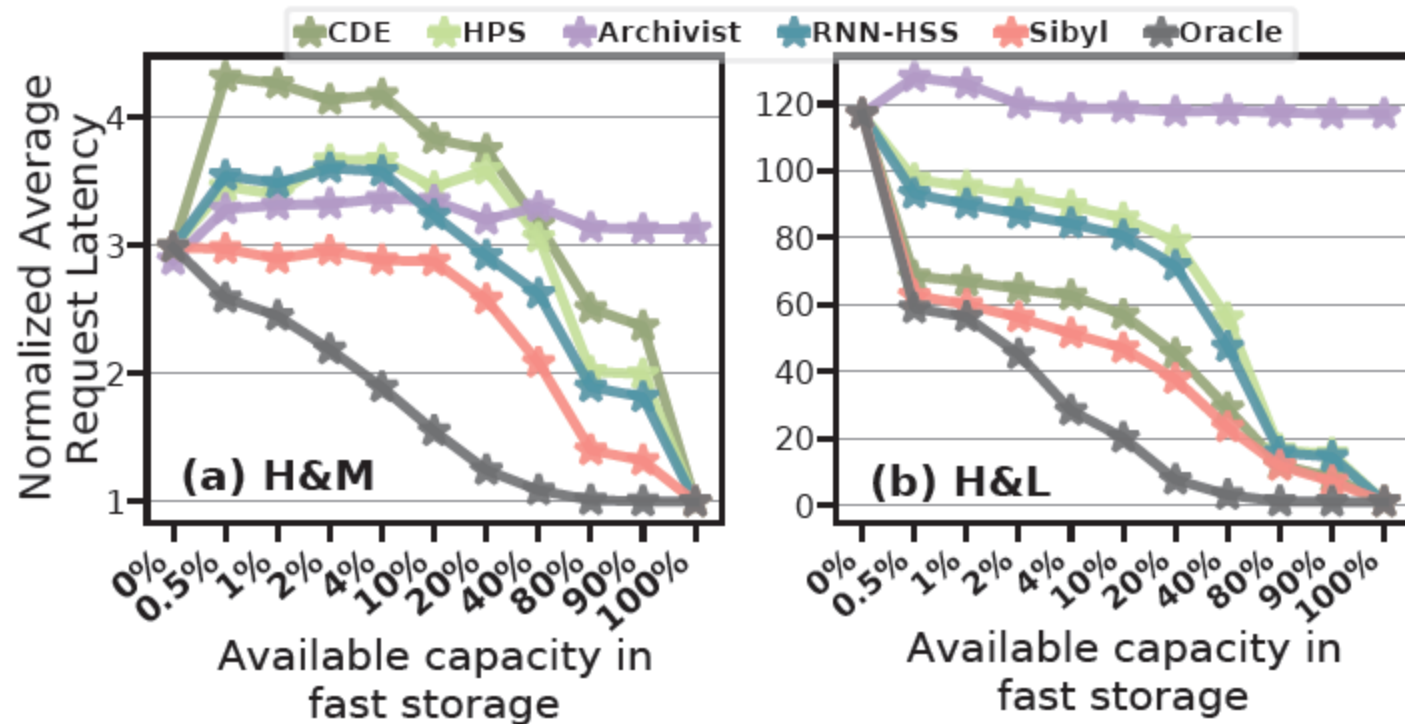
Sibyl outperforms the heuristic-based data placement policy for tri-hybrid system by 48.2% on average across all workloads

Latency for Unseen Workloads

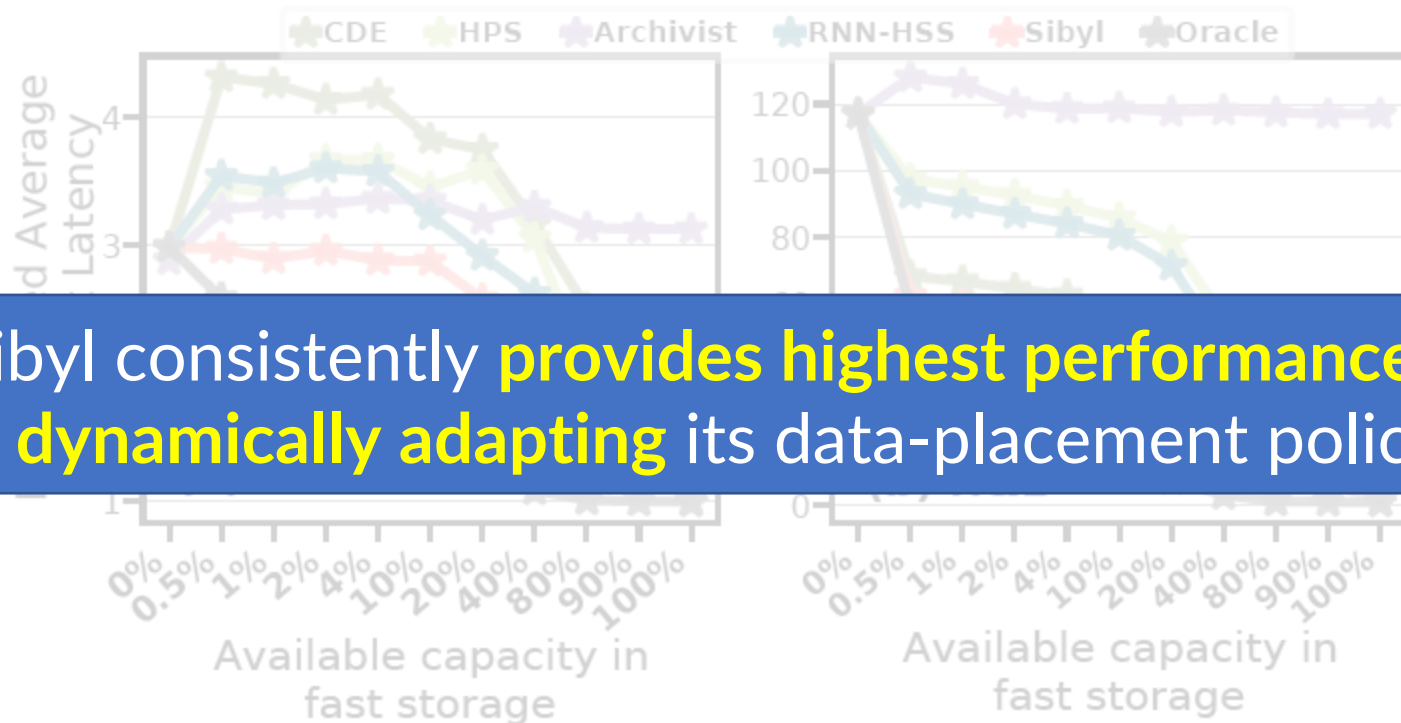


In H&M (H&L) configurations, Sibyl outperforms RNN-HSS and Archivist by 46.1% (54.6%) and 8.5% (44.1%) respectively

Sensitivity to Fast Storage Capacity



Sensitivity to Fast Storage Capacity



Sibyl consistently **provides highest performance** by **dynamically adapting** its data-placement policy

Overhead Analysis

- **Performance Overhead**
 - **~10ns** for every inference on the evaluated system; this is several orders of magnitude less than I/O latency of high-end SSD
- **Implementation Overhead**
 - **124.4 KiB** of implementation overhead
- **Metadata overhead**
 - 0.1% of the total storage capacity when using a 4 KiB data placement granularity
 - **40-bit** metadata overhead per data placement unit

For More on Sybil

- To appear in ISCA 2022

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

<https://arxiv.org/pdf/2205.07394.pdf>

Current EFCL Projects

- “A New Methodology and Open-Source Benchmark Suite for Evaluating Data Movement Bottlenecks: A Processing-in-Memory Case Study”
 - Data-centric
- “Machine-Learning-Assisted Intelligent Microarchitectures to Reduce Memory Access Latency”
 - Data-driven
- “Cross-layer Hardware/Software Techniques to Enable Powerful Computation and Memory Optimizations”
 - Data-aware

Cross-layer Hardware/Software Techniques to Enable Powerful Computation and Memory Optimizations

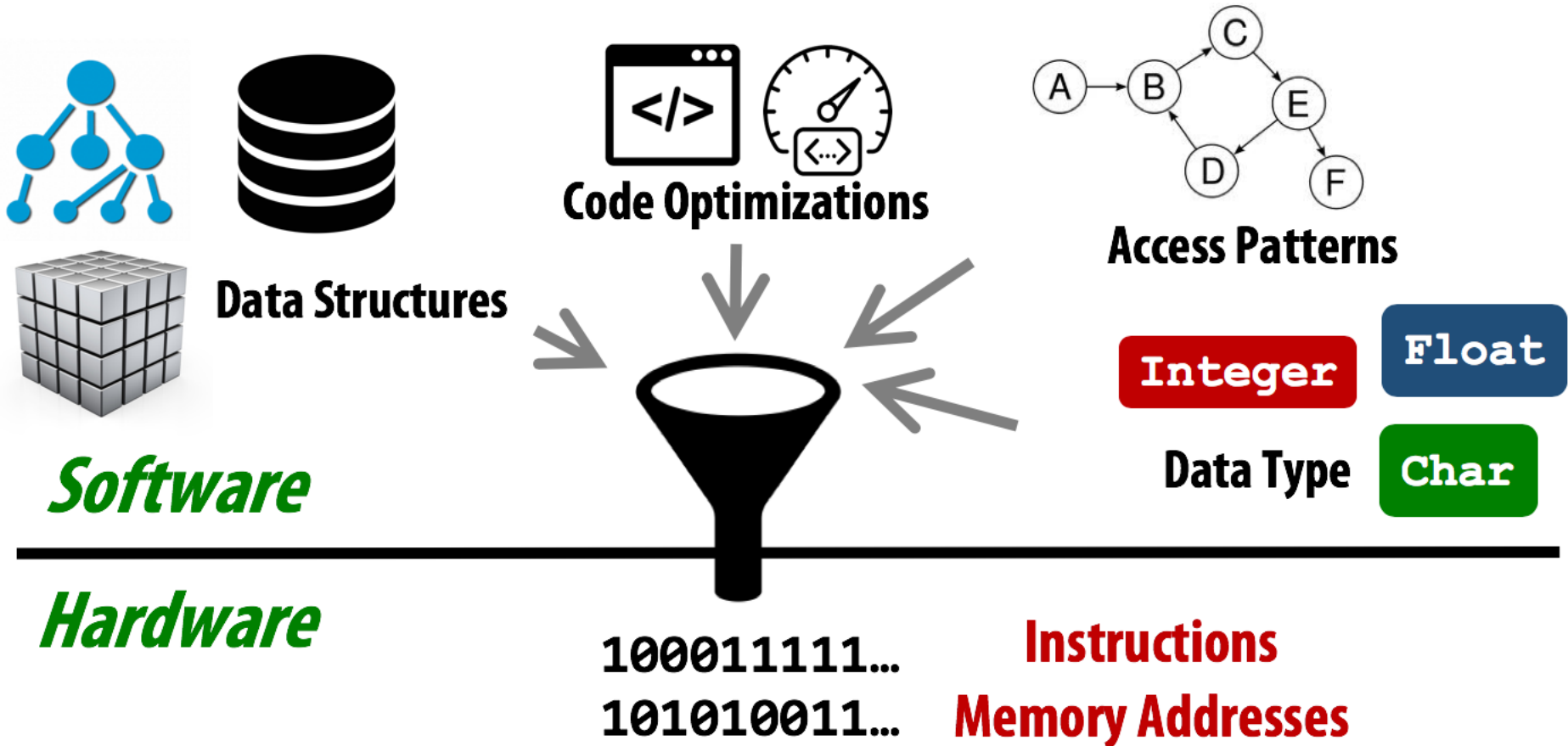
Ataberk Olgun, Konstantinos Kanellopoulos, Nisa Bostanci
Onur Mutlu

Data-Aware Architectures

- A data-aware architecture understands what it can do with and to each piece of data
- It makes use of different properties of data to improve performance, efficiency and other metrics
 - Compressibility
 - Approximability
 - Locality
 - Sparsity
 - Criticality for Computation X
 - Access Semantics
 - ...

One Problem: Limited Expressiveness

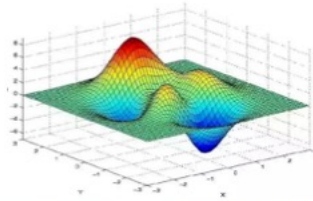
Higher-level information is not visible to HW



A Solution: More Expressive Interfaces

Performance

Software



Functionality



**ISA
Virtual Memory**

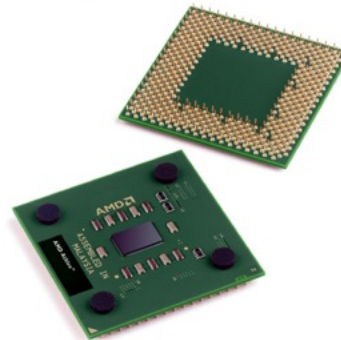
**Higher-level
Program
Semantics**

**Expressive
Memory
"XMem"**

Hardware



wiseGEEK



Data-Aware (Expressive)

Computing Architectures

Results So Far (2021-2022)

- MetaSys: A Practical Open-Source Metadata Management System to Implement and Evaluate Cross-Layer Optimizations [ACM TACO 2022]

MetaSys Open Source Framework

- Appears in ACM TACO 2022

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

NANDITA VIJAYKUMAR, University of Toronto, Canada

ATABERK OLGUN, ETH Zurich, TOBB ETU, Turkey

KONSTANTINOS KANELLOPOULOS, ETH Zurich, Switzerland

F. NISA BOSTANCI, ETH Zurich, TOBB ETU, Turkey

HASAN HASSAN, ETH Zurich, Switzerland

MEHRSHAD LOTFI, Max Plank Institute, Germany

PHILLIP B. GIBBONS, Carnegie Mellon University, USA

ONUR MUTLU, ETH Zurich, Switzerland

MetaSys

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

Nandita Vijaykumar

Ataberk Olgun, Konstantinos Kanellopoulos, F. Nisa Bostanci

Hasan Hassan, Mehrshad Lotfi, Phillip B. Gibbons, Onur Mutlu

SAFARI



UNIVERSITY OF
TORONTO

ETH *Zürich*



TOBB ETÜ
University of Economics & Technology

Executive Summary

Problem

- Cross-layer techniques are **challenging** to **implement** because they require **full-stack changes**
- **Existing open-source infrastructures** for implementing cross-layer techniques are **not** designed to provide **key features**

Key Idea – Provide:

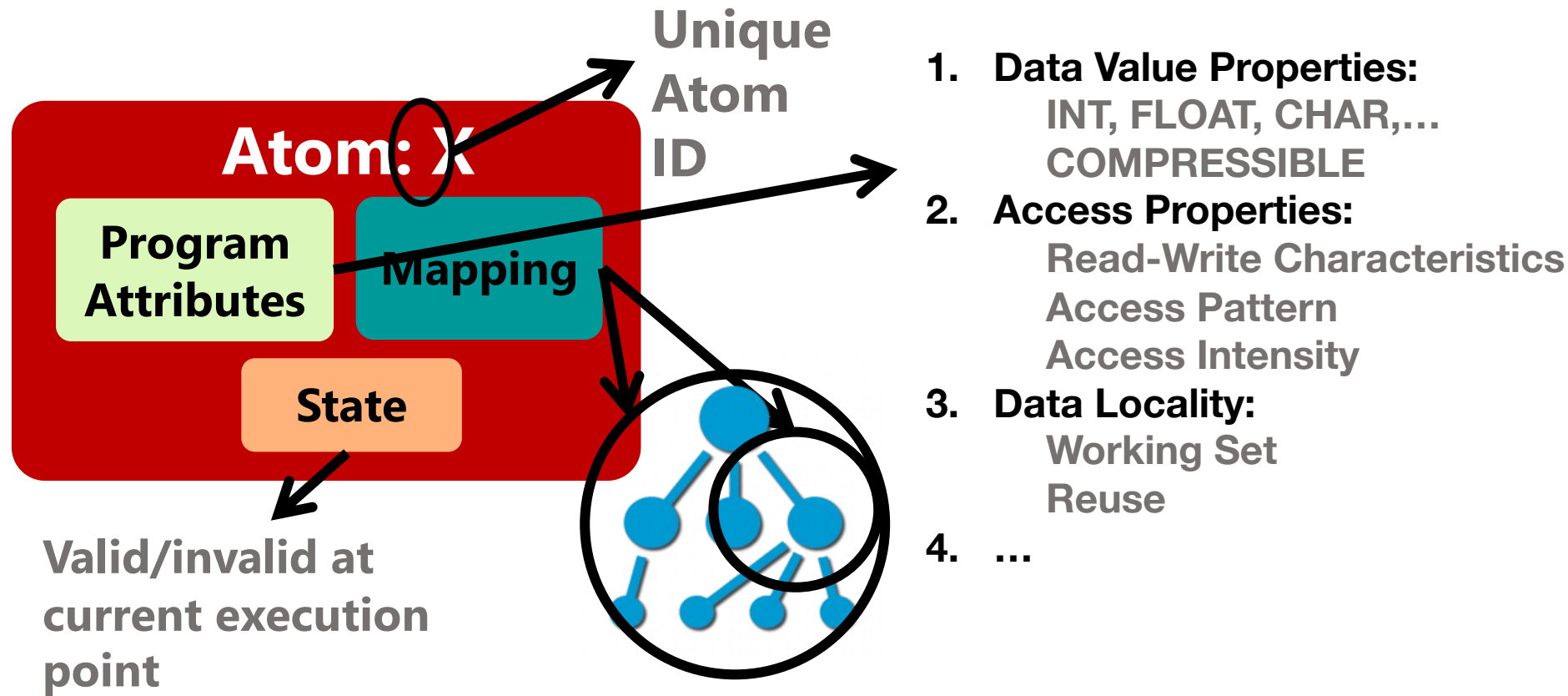
- Rich dynamic HW/SW interfaces
- Low-overhead metadata management
- Interfaces to key hardware components (e.g., prefetcher, caches)

Our **goal** is twofold:

1. Develop an **efficient** and **flexible** framework to enable **rapid implementation** of new cross-layer techniques
2. Perform a detailed limit study to quantify the overheads associated with **general** metadata systems

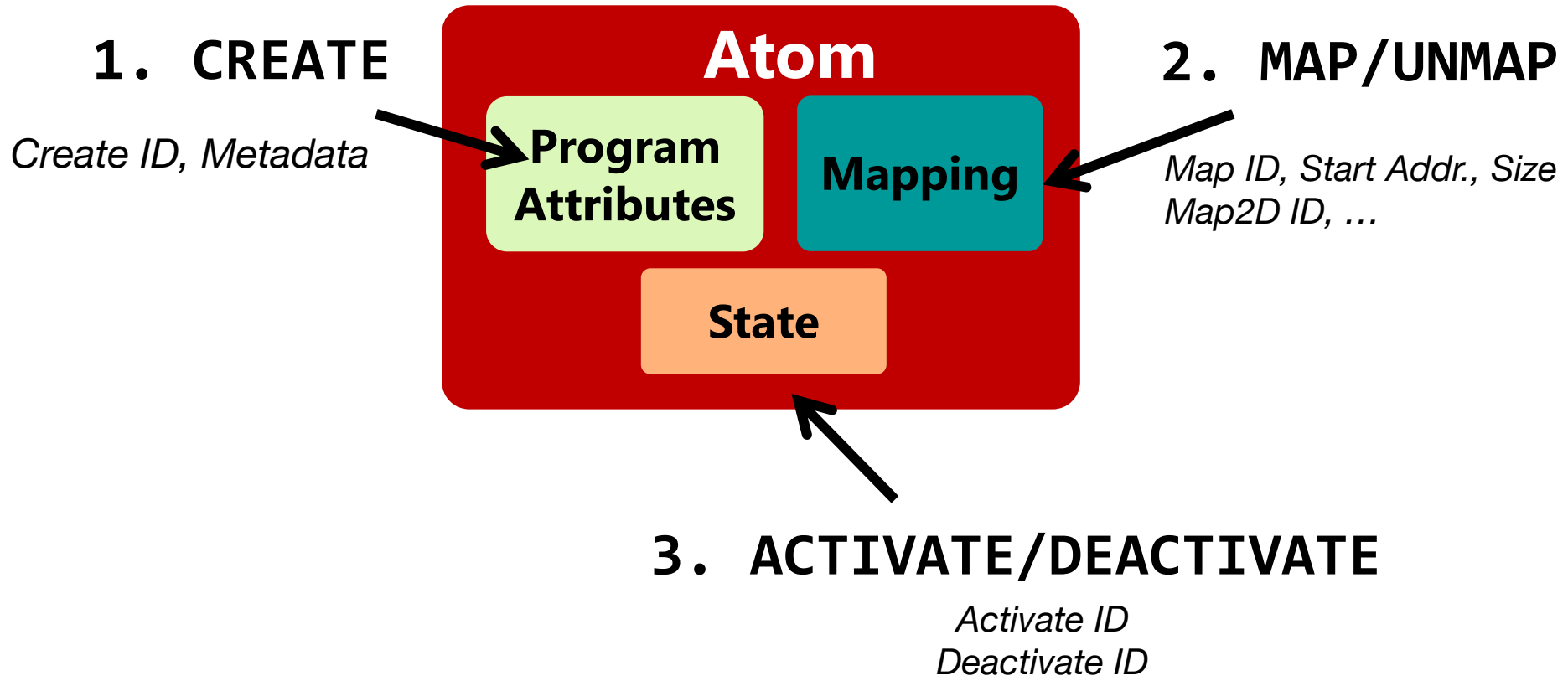
The Atom Abstraction

An abstraction to express data semantics

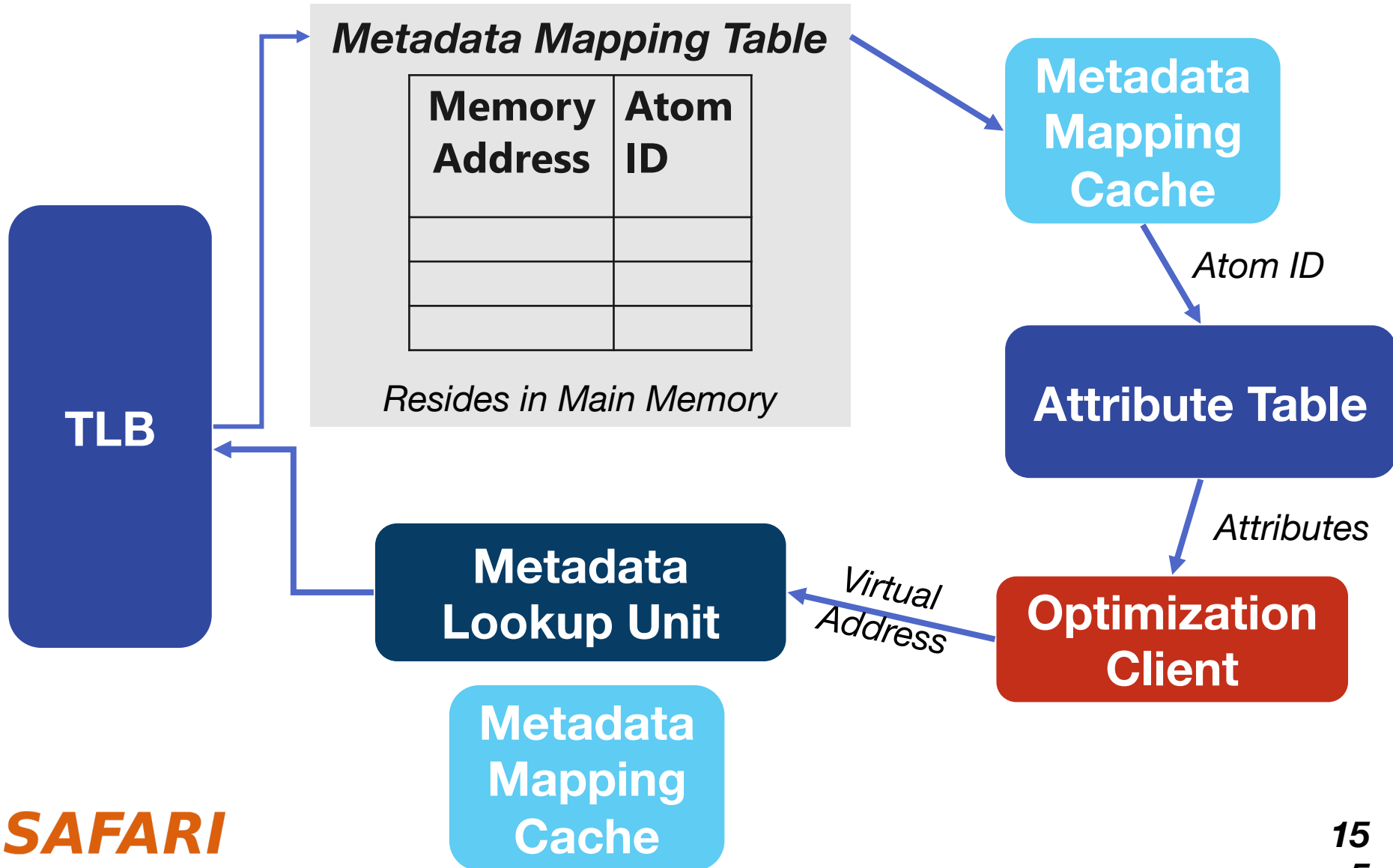


The Software Interface

Three Atom operators



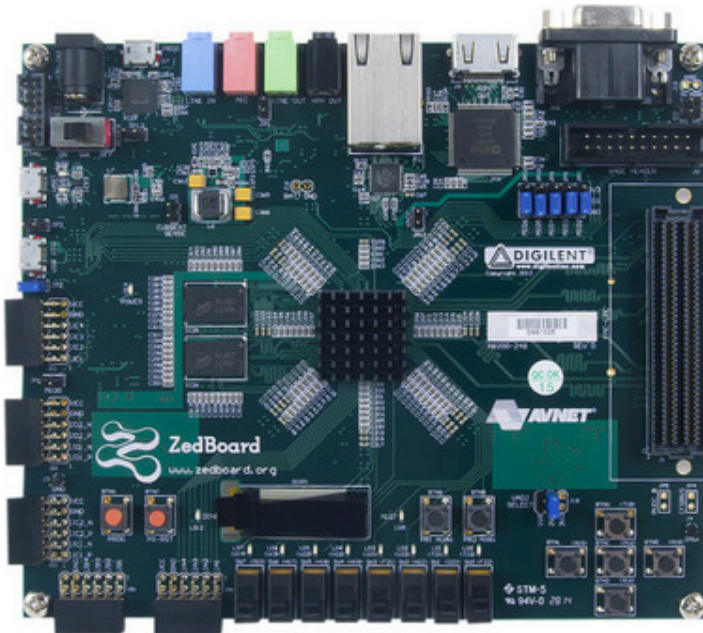
MetaSys Key Structures



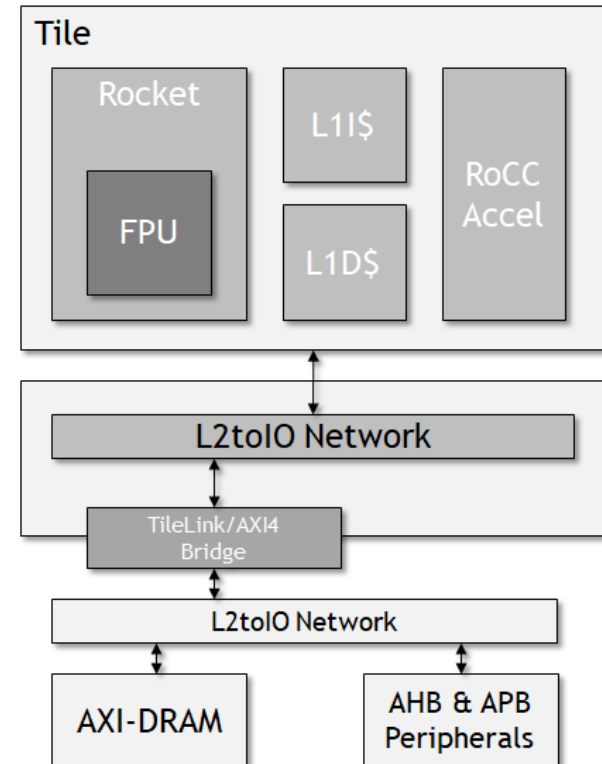
FPGA Prototype

Prototype on Xilinx Zedboard
within a **real RISC-V system** (Rocket Chip)

Zedboard



Rocket Chip



MetaSys in Rocket Chip

Implement two **main** components:

1. Atom Controller


- Manages the **attribute table** (CREATE – (DE)ACTIVATE)
- Performs atom mapping (MAP/UNMAP)
 - Physical address → Atom ID





2. Metadata Lookup Unit









- Responds to clients:
 - Provides atom attributes
- Contains the metadata mapping cache




MetaSys is Open Source



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


 **CMU-SAFARI / MetaSys** Public

 Notifications  Fork 2  Star 0 


 Code  Issues  Pull requests  Actions  Projects  Wiki  Security  Insights

 main  1 branch  0 tags

 Go to file  Code

 **olgunataberk** Update README.md e21ccd2 on Jul 9, 2021 🕒 12 commits

common	Initial commit	9 months ago
riscv-tools	Add tools directory	7 months ago
rocket-chip	Initial commit	9 months ago
testchipip	Initial commit	9 months ago
zedboard	Initial commit	9 months ago
LICENSE	Initial commit	9 months ago
README.md	Update README.md	7 months ago
metasys_readme.md	Update metasys_readme.md	7 months ago

 README.md



MetaSys




We refer the developers of the MetaSys repository to [metasys_readme.md](#), where we describe our modifications to the existing rocket-chip code base, and present a walkthrough of an implementation of the prefetching use case described in our paper.

For more details, please read our [preprint on arXiv](#).

About

Metasys is the first open-source FPGA-based infrastructure with a prototype in a RISC-V core, to enable the rapid implementation and evaluation of a wide range of cross-layer software/hardware cooperative techniques techniques in real hardware. Described in our preprint: <https://arxiv.org/abs/2105.08123>

 Readme  View license

 0 stars  3 watching  2 forks

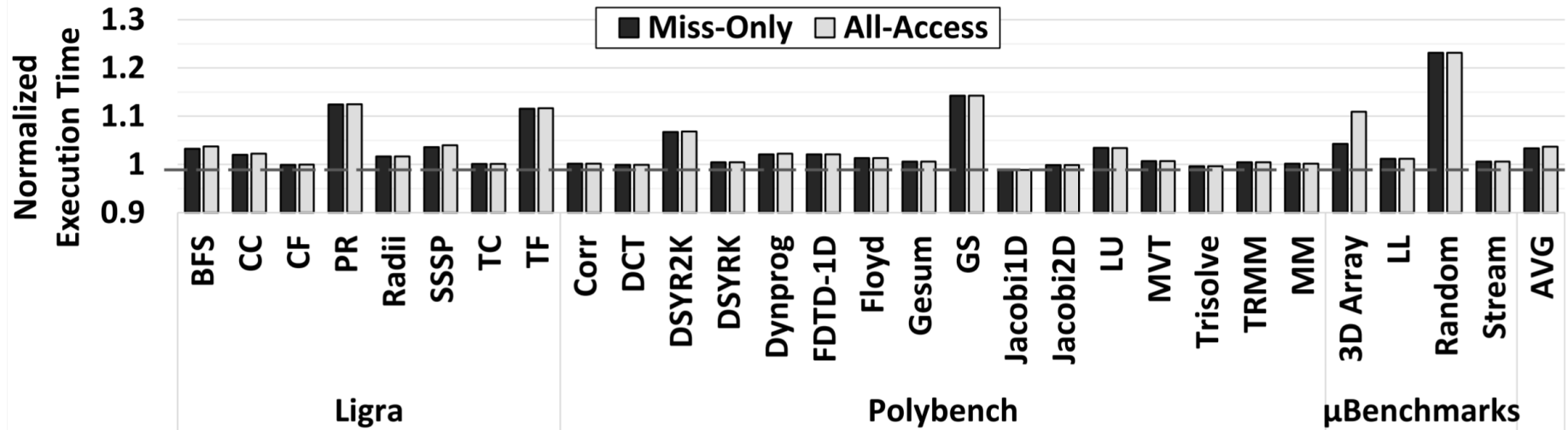
Releases

No releases published

Packages

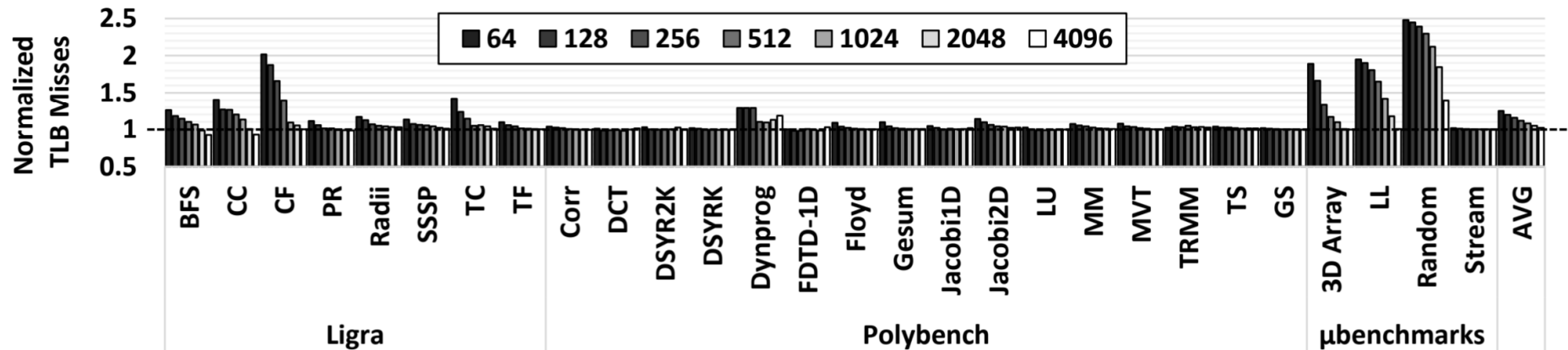
No packages published

Performance Overhead



Metadata lookups occur low performance overheads
2.7% on average

Impact of Tagging Granularity on TLB misses



Fine tagging granularities increase TLB misses

MetaSys

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

Nandita Vijaykumar

Ataberk Olgun, Konstantinos Kanellopoulos, F. Nisa Bostanci

Hasan Hassan, Mehrshad Lotfi, Phillip B. Gibbons, Onur Mutlu

SAFARI



UNIVERSITY OF
TORONTO

ETH *Zürich*



TOBB ETÜ
University of Economics & Technology

Results So Far (2021-2022)

- **Many Educational and Outreach Efforts**

- 11 Different Courses (livestreamed & recorded)
- Tutorials
- Keynote Talks
- Invited Talks
- Conference & Workshop Talks
- ...

- All are freely available on our YouTube Channel

- <https://www.youtube.com/onurmutlulectures>

A Detailed Tutorial

- Onur Mutlu,
"Memory-Centric Computing"
Education Class at Embedded Systems Week (ESWEEK),
Virtual, 9 October 2021.
[Slides (pptx) (pdf)]
[Abstract (pdf)]
[Talk Video (2 hours, including Q&A)]
[Invited Paper at DATE 2021]
["A Modern Primer on Processing in Memory" paper]

<https://www.youtube.com/watch?v=N1Ac1ov1JOM>

Memory-Centric Computing

Onur Mutlu

omutlu@gmail.com

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

9 October 2021

ESWEEK Education Class

SAFARI

ETH zürich

Carnegie Mellon



1:08 / 2:00:10



Embedded Systems Week (ESWEEK) 2021 Lecture - Memory-Centric Computing - Onur Mutlu - 9 October 2021

509 views • Premiered Dec 6, 2021

28 DISLIKE SHARE SAVE ...



Onur Mutlu Lectures
20.7K subscribers

<https://www.youtube.com/watch?v=N1Ac1ov1JOM>

ANALYTICS

EDIT VIDEO

<https://www.youtube.com/onurmutlulectures>

164

Online Courses & Lectures

■ **First Computer Architecture & Digital Design Course**

- ❑ Digital Design and Computer Architecture
- ❑ **Spring 2022 Livestream** Edition:
<https://www.youtube.com/watch?v=cpXdE3HwvK0&list=PL5Q2soXY2Zi97Ya5DEUpMpO2bbAoaG7c6>
- ❑ **Spring 2021 Livestream** Edition:
https://www.youtube.com/watch?v=LbC0EZY8yw4&list=PL5Q2soXY2Zi_uej3aY39YB5pfW4SJ7LIN

■ **Advanced Computer Architecture Course**

- ❑ Computer Architecture
- ❑ **Fall 2021 Livestream** Edition:
https://www.youtube.com/watch?v=4yfkM_5EFgo&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF
- ❑ **Fall 2020** Edition: <https://www.youtube.com/watch?v=c3mPdZA-Fmc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN>

Comp Arch (Fall'21)

- Lectures/Schedule
- Lecture Buzzwords
- Readings
- HWs
- Labs
- Exams
- Related Courses
- Tutorials

- Computer Architecture FS20: Course Webpage
- Computer Architecture FS20: Lecture Videos
- Digitaltechnik SS21: Course Webpage
- Digitaltechnik SS21: Lecture Videos
- Moodle
- HotCRP
- Verilog Practice Website (HDLBits)

Fall 2021 Edition:

- <https://safari.ethz.ch/architecture/fall2021/doku.php?id=schedule>

Fall 2020 Edition:

- <https://safari.ethz.ch/architecture/fall2020/doku.php?id=schedule>

Youtube Livestream (2021):

- https://www.youtube.com/watch?v=4yfkM_5EFg0&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF

Youtube Livestream (2020):

- <https://www.youtube.com/watch?v=c3mPdZA-Fmc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN>

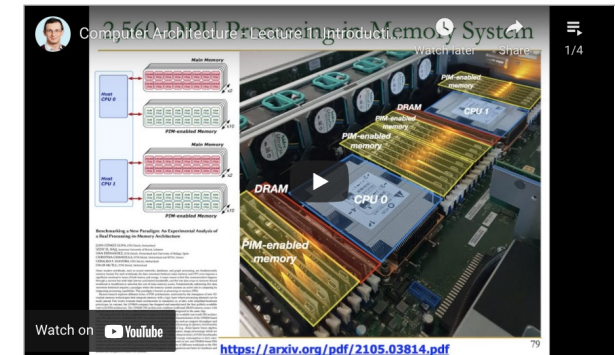
Master's level course

- Taken by Bachelor's/Masters/PhD students
- Cutting-edge research topics + fundamentals in Computer Architecture
- 5 Simulator-based Lab Assignments
- Potential research exploration
- Many research readings

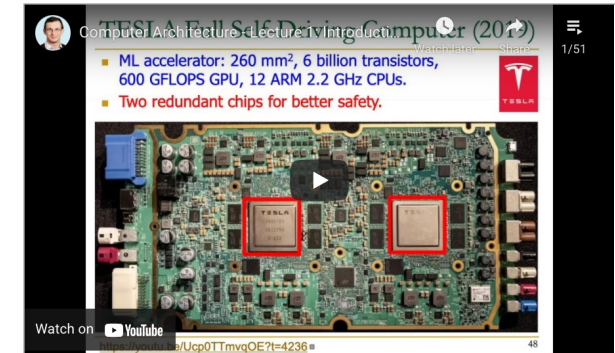
<https://www.youtube.com/onurmutlulectures>

Lecture Video Playlist on YouTube

Livestream Lecture Playlist



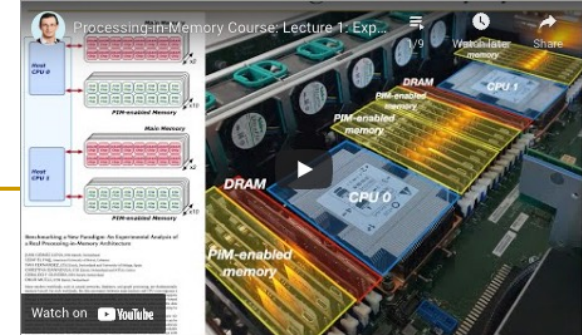
Recorded Lecture Playlist



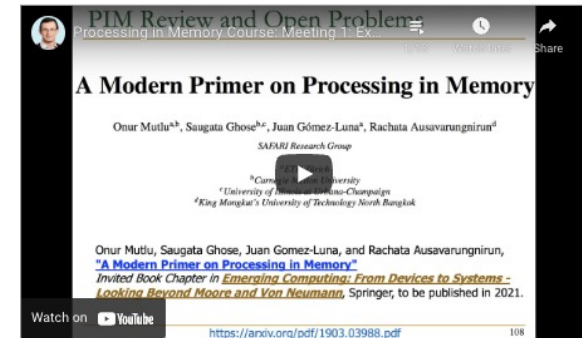
Fall 2021 Lectures & Schedule

Week	Date	Livestream	Lecture	Readings	Lab	HW
W1	30.09 Thu.	YouTube Live	L1: Introduction and Basics PDF PPT	Required Mentioned	Lab 1 Out	HW 0 Out
	01.10 Fri.	YouTube Live	L2: Trends, Tradeoffs and Design Fundamentals PDF PPT	Required Mentioned		
W2	07.10 Thu.	YouTube Live	L3a: Memory Systems: Challenges and Opportunities PDF PPT	Described Suggested		HW 1 Out
			L3b: Course Info & Logistics PDF PPT			
			L3c: Memory Performance Attacks PDF PPT	Described Suggested		
	08.10 Fri.	YouTube Live	L4a: Memory Performance Attacks PDF PPT	Described Suggested	Lab 2 Out	
			L4b: Data Retention and Memory Refresh PDF PPT	Described Suggested		
			L4c: RowHammer PDF PPT	Described Suggested		

PIM Course (Current)



Recorded Lecture Playlist



Spring 2022 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	10.03 Thu.	Live	M1: P&S PIM Course Presentation (PDF) (PPT)	Required Materials Recommended Materials	HW 0 Out
W2	15.03 Tue.		Hands-on Project Proposals		
	17.03 Thu.	Premiere	M2: Real-world PIM: UPMEM PIM (PDF) (PPT)		
W3	24.03 Thu.	Live	M3: Real-world PIM: Microbenchmarking of UPMEM PIM (PDF) (PPT)		
W4	31.03 Thu.	Live	M4: Real-world PIM: Samsung HBM-PIM (PDF) (PPT)		
W5	07.04 Thu.	Live	M5: How to Evaluate Data Movement Bottlenecks (PDF) (PPT)		
W6	14.04 Thu.	Live	M6: Real-world PIM: SK Hynix A1M (PDF) (PPT)		
W7	21.04 Thu.	Premiere	M7: Programming PIM Architectures (PDF) (PPT)		
W8	28.04 Thu.	Premiere	M8: Benchmarking and Workload Suitability on PIM (PDF) (PPT)		
W9	05.05 Thu.	Premiere	M9: Real-world PIM: Samsung AxDIMM (PDF) (PPT)		
W10	12.05 Thu.		M10: Real-world PIM: Alibaba HB-PIM (PDF) (PPT)		

Spring 2022 Edition:

- https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=processing_in_memory

Youtube Livestream:

- https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi-841fUYYUK9EsXKhQKRPyX

Project course

- Taken by Bachelor's/Master's students
- Processing-in-Memory lectures
- Hands-on research exploration
- Many research readings

PIM Course (Fall'21)

Fall 2021 Edition:

- https://safari.ethz.ch/projects_and_seminars/fall2021/doku.php?id=processing_in_memory

Youtube Livestream:

- <https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi-841fUYYUK9EsXKhQKRPyX>

Project course

- Taken by Bachelor's/Master's students
- Processing-in-Memory lectures
- Hands-on research exploration
- Many research readings

PIM Review and Open Problems
Processing in Memory Course: Meeting 1: Ex...

Watch later Share 1/10

A Modern Primer on Processing in Memory

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

SAFARI Research Group

^aETH Zürich
^bCarnegie Mellon University
^cUniversity of Illinois at Urbana-Champaign
^dKing Mongkut's University of Technology North Bangkok

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun, "A Modern Primer on Processing in Memory" Invited Book Chapter in *Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann*, Springer, to be published in 2021.

Watch on YouTube <https://arxiv.org/pdf/1903.03988.pdf> 108

Fall 2021 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	05.10 Tue.	YouTube Live	M1: P&S PIM Course Presentation PDF (PDF) PPT (PPT)	Required Materials Recommended Materials	HW 0 Out
W2	12.10 Tue.	YouTube Live	M2: Real-World PIM Architectures PDF (PDF) PPT (PPT)		
W3	19.10 Tue.	YouTube Live	M3: Real-World PIM Architectures II PDF (PDF) PPT (PPT)		
W4	26.10 Tue.	YouTube Live	M4: Real-World PIM Architectures III PDF (PDF) PPT (PPT)		
W5	02.11 Tue.	YouTube Live	M5: Real-World PIM Architectures IV PDF (PDF) PPT (PPT)		
W6	09.11 Tue.	YouTube Live	M6: End-to-End Framework for Processing-using-Memory PDF (PDF) PPT (PPT)		
W7	16.11 Tue.	YouTube Live	M7: How to Evaluate Data Movement Bottlenecks PDF (PDF) PPT (PPT)		
W8	23.11 Tue.	YouTube Live	M8: Programming PIM Architectures PDF (PDF) PPT (PPT)		
W9	30.11 Tue.	YouTube Live	M9: Benchmarking and Workload Suitability on PIM PDF (PDF) PPT (PPT)		
W10	07.12 Tue.	YouTube Live	M10: Bit-Serial SIMD Processing using DRAM PDF (PDF) PPT (PPT)		

Hetero. Systems (Fall'21)

Fall 2021 Edition:

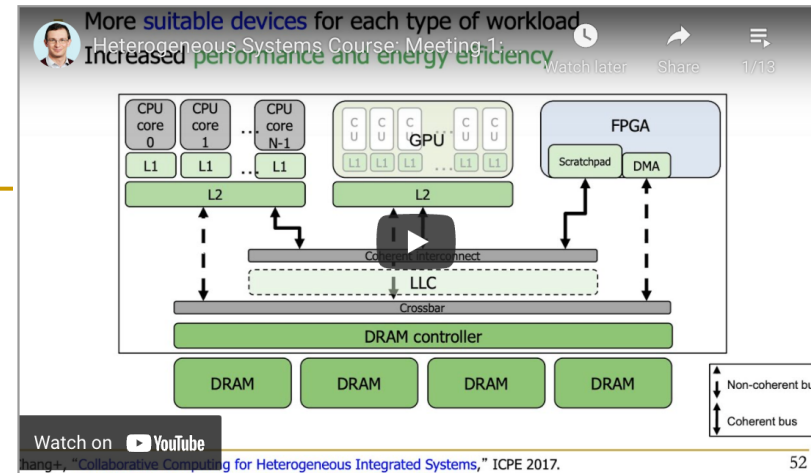
- https://safari.ethz.ch/projects_and_seminars/fall2021/doku.php?id=heterogeneous_systems

Youtube Livestream:

- https://www.youtube.com/watch?v=QYbjwzsfMM&list=PL5Q2soXY2Zi_OwkTgEyA6tk3UsoPBH737

Project course

- Taken by Bachelor's/Master's students
- GPU and Parallelism lectures
- Hands-on research exploration
- Many research readings



Fall 2021 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	07.10 Thu.	YouTube Live	M1: P&S Course Presentation (PDF) (PPT)	Required Materials Recommended Materials	HW 0 Out
W2	14.10 Thu.	YouTube Live	M2: SIMD Processing and GPUs (PDF) (PPT)		
W3	21.10 Thu.	YouTube Live	M3: GPU Software Hierarchy (PDF) (PPT)		
W4	28.10 Thu.	YouTube Live	M4: GPU Memory Hierarchy (PDF) (PPT)		
W5	04.11 Thu.	YouTube Live	M5: GPU Performance Considerations (PDF) (PPT)		
W6	11.11 Thu.	YouTube Live	M6: Parallel Patterns: Reduction (PDF) (PPT)		
W7	18.11 Thu.	YouTube Live	M7: Parallel Patterns: Histogram (PDF) (PPT)		
W8	25.11 Thu.	YouTube Live	M8: Parallel Patterns: Convolution (PDF) (PPT)		
W9	02.12 Thu.	YouTube Live	M9: Parallel Patterns: Prefix Sum (Scan) (PDF) (PPT)		
W10	09.12 Thu.	YouTube Live	M10: Parallel Patterns: Sparse Matrices (PDF) (PPT)		
W11	16.12 Thu.	YouTube Live	M11: Parallel Patterns: Graph Search (PDF) (PPT)		
W12	22.12 Thu.	YouTube Live	M12: Dynamic Parallelism (PDF) (PPT)		
W13	06.01 Thu.	YouTube Live	M13: Collaborative Computing (PDF) (PPT)		

Genomics (Fall 2021)

Fall 2021 Edition:

- https://safari.ethz.ch/projects_and_seminars/fall2021/doku.php?id=bioinformatics

Youtube Livestream:

- <https://www.youtube.com/watch?v=MnogTeMjY8k&list=PL5Q2soXY2Zi8sngH-TrNZnDhDkPq55J9J>

Project course

- Taken by Bachelor's/Master's students
- Genomics lectures
- Hands-on research exploration
- Many research readings

Mobile Genomics Course - Meeting 1: Course...

Understanding **genetic variations**

Predicting the **presence and relative abundances of microbes** in a sample

Watch on **YouTube**

Rapid surveillance of **disease outbreaks**

Developing **personalized medicine**

Fall 2021 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	5.10 Tue.	YouTube Live	M1: P&S Accelerating Genomics Course Introduction & Project Proposals PDF (PDF) PPT (PPT) YouTube Video	Required Materials Recommended Materials	
W2	20.10 Wed.	YouTube Live	M2: Introduction to Sequencing PDF (PDF) PPT (PPT)		
W3	27.10 Wed.	YouTube Live	M3: Read Mapping PDF (PDF) PPT (PPT)		
W4	3.11 Wed.	YouTube Live	M4: GateKeeper PDF (PDF) PPT (PPT)		
W5	10.11 Wed.	YouTube Live	M5: MAGNET & Shouji PDF (PDF) PPT (PPT)		
W6	17.11 Wed.		M6.1: SneakySnake PDF (PDF) PPT (PPT) YouTube Video		
			M6.2: GRIM-Filter PDF (PDF) PPT (PPT) YouTube Video		
W7	24.11 Wed.		M7: GenASM PDF (PDF) PPT (PPT) YouTube Video		
W8	01.12 Wed.	YouTube Live	M8: Genome Assembly PDF (PDF) PPT (PPT)		
W9	13.12 Mon.	YouTube Live	M9: GRIM-Filter PDF (PDF) PPT (PPT)		
W10	15.12 Wed.	YouTube Live	M10: Genomic Data Sharing Under Differential Privacy PDF (PDF) PPT (PPT)		

HW/SW Co-Design (Spring 2022)

Spring 2022 Edition:

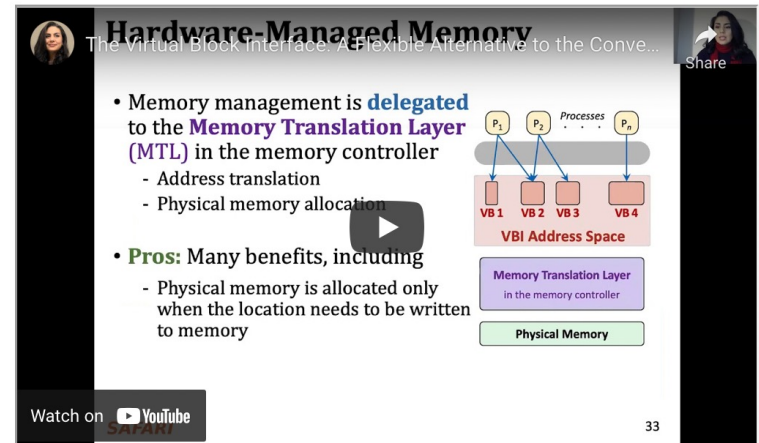
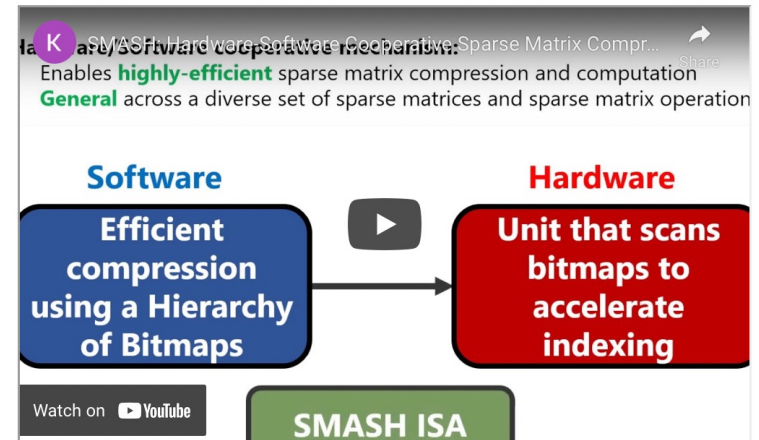
- https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=hw_sw_co_design

Youtube Livestream:

- https://youtube.com/playlist?list=PL5Q2soXY2Zi8nH7un3ghD2nutKWWDk-NK

Project course

- Taken by Bachelor's/Master's students
- HW/SW co-design lectures
- Hands-on research exploration
- Many research readings



2022 Meetings/Schedule (Tentative)

Week	Date	Livestream	Meeting	Materials	Assignments
W0	16.03	Live	Intro to HW/SW Co-Design (PPTX) (PDF)	Required	HW 0 Out
W1	23.03		Project selection	Required	
W2	30.03	Live	Virtual Memory (I) (PPTX) (PDF)		
W3	13.04	Live	Virtual Memory (II) (PPTX) (PDF)		

SSD Course (Spring 2022)

■ Spring 2022 Edition:

- https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=modern_sds

■ Youtube Livestream:

- <https://www.youtube.com/watch?v=q4rm71DsY4&list=PL5Q2soXY2Zi8vabcse1kL22DEcgMI2RAq>

■ Project course

- Taken by Bachelor's/Master's students
- SSD Basics and Advanced Topics
- Hands-on research exploration
- Many research readings

The image displays two screenshots of a Zoom video player interface, showing lecture slides from the 'P&S Modern SSDs' course. The top screenshot is from 'Meeting 2: Basics of NAND Flash-Based SSDs (Spring 2022)', dated 25 March 2021. The slide content includes the title 'P&S Modern SSDs', the subtitle 'Basics of NAND Flash-Based SSDs', and the speakers 'Dr. Jisung Park' and 'Prof. Onur Mutlu' from 'ETH Zürich'. The bottom screenshot is from 'Meeting 4: Introduction to MQSim (Spring 2022)', dated 8th April 2022. The slide content includes the title 'P&S Modern SSDs', the subtitle 'Introduction to MQSim', and the speakers 'Rakesh Nadig', 'Dr. Jisung Park', and 'Prof. Onur Mutlu' from 'ETH Zürich'. Both screenshots show a Zoom video player with a progress bar, volume control, and a small video feed of the speaker in the top right corner. The video player interface also shows the video title, view count, and streaming date.

P&S Modern SSDs
Basics of NAND Flash-Based SSDs

Dr. Jisung Park
Prof. Onur Mutlu
ETH Zürich
Spring 2022
25 March 2021

Modern Solid-State Drives (SSDs) Course - Meeting 2: Basics of NAND Flash-Based SSDs (Spring 2022)
807 views • Streamed live on Mar 25, 2022

P&S Modern SSDs
Introduction to MQSim

Rakesh Nadig
Dr. Jisung Park
Prof. Onur Mutlu
ETH Zürich
Spring 2022
8th April 2022

Modern Solid-State Drives (SSDs) Course - Meeting 4: Introduction to MQSim (Spring 2022)
310 views • Streamed live on Apr 8, 2022

Hands-On Projects & Seminars Courses

- https://safari.ethz.ch/projects_and_seminars



SAFARI Project & Seminars Courses
(Spring 2022)



[Recent Changes](#) [Media Manager](#) [Sitemap](#)

Trace: • [start](#)

[Home](#)

Courses

- [SoftMC](#)
- [Ramulator](#)
- [Accelerating Genomics](#)
- [Mobile Genomics](#)
- [Processing-in-Memory](#)
- [Heterogeneous Systems](#)
- [Modern SSDs](#)
- [Hardware/Software Co-design](#)

[start](#)

SAFARI Projects & Seminars Courses (Spring 2022)

Welcome to the wiki for Project and Seminar courses SAFARI offers.

Courses we offer:

- [Understanding and Improving Modern DRAM Performance, Reliability, and Security with Hands-On Experiments](#)
- [Designing and Evaluating Memory Systems and Modern Software Workloads with Ramulator](#)
- [Accelerating Genome Analysis with FPGAs, GPUs, and New Execution Paradigms](#)
- [Genome Sequencing on Mobile Devices](#)
- [Exploring the Processing-in-Memory Paradigm for Future Computing Systems](#)
- [Hands-on Acceleration on Heterogeneous Computing Systems](#)
- [Understanding and Designing Modern NAND Flash-Based Solid-State Drives \(SSDs\)](#)
- [Intelligent Architectures using Hardware/Software Cooperative Techniques](#)



Onur Mutlu Lectures

16.9K subscribers

CUSTOMIZE CHANNEL

MANAGE VIDEOS

HOME

VIDEOS

PLAYLISTS


COMMUNITY

CHANNELS

ABOUT




Popular uploads ▶ PLAY ALL




How Computers Work
(from the ground up)

1:33:25



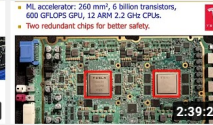
Digital Design & Computer
Architecture: Lecture 1:...

49K views • 1 year ago



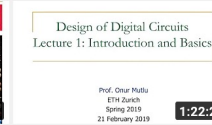
Computer Architecture -
Lecture 1: Introduction and...

36K views • 3 years ago




Computer Architecture -
Lecture 1: Introduction and...

31K views • 1 year ago




Computer Architecture -
Lecture 1: Introduction and...

30K views • 8 months ago



Design of Digital Circuits -
Lecture 1: Introduction and...

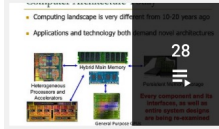
22K views • 2 years ago



Computer Architecture -
Lecture 2: Fundamentals,...


17K views • 3 years ago

First Course in Computer Architecture & Digital Design 2021-2013




Livestream - Digital Design and
Computer Architecture - ETH...

Onur Mutlu Lectures
VIEW FULL PLAYLIST



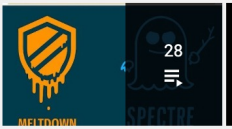
Digital Design & Computer
Architecture - ETH Zürich...

Onur Mutlu Lectures
VIEW FULL PLAYLIST



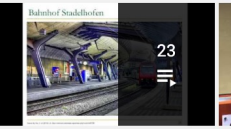
Design of Digital Circuits - ETH
Zürich - Spring 2019

Onur Mutlu Lectures
VIEW FULL PLAYLIST




Design of Digital Circuits - ETH
Zürich - Spring 2018

Onur Mutlu Lectures
VIEW FULL PLAYLIST



Digital Circuits and Computer
Architecture - ETH Zürich - ...

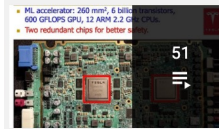
Onur Mutlu Lectures
VIEW FULL PLAYLIST



Spring 2015 -- Computer
Architecture Lectures -- ...

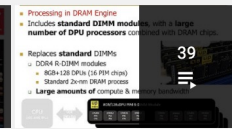
Carnegie Mellon Computer Architec...
VIEW FULL PLAYLIST

Advanced Computer Architecture Courses 2020-2012




Computer Architecture - ETH
Zürich - Fall 2020

Onur Mutlu Lectures
VIEW FULL PLAYLIST




Computer Architecture - ETH
Zürich - Fall 2019

Onur Mutlu Lectures
VIEW FULL PLAYLIST




Computer Architecture - ETH
Zürich - Fall 2018

Onur Mutlu Lectures
VIEW FULL PLAYLIST



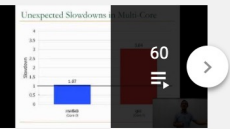
Computer Architecture - ETH
Zürich - Fall 2017

Onur Mutlu Lectures
VIEW FULL PLAYLIST



Fall 2015 - 740 Computer
Architecture


Carnegie Mellon Computer Archite...
VIEW FULL PLAYLIST



Fall 2013 - 740 Computer
Architecture - Carnegie Mellon


Carnegie Mellon Computer Archite...
VIEW FULL PLAYLIST

Special Courses on Memory Systems



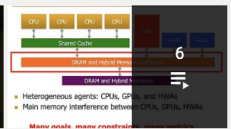
Memory Technology Lectures

Onur Mutlu Lectures
VIEW FULL PLAYLIST




Champéry Winter School 2020 -
Memory Systems and Memory...

Onur Mutlu Lectures
VIEW FULL PLAYLIST



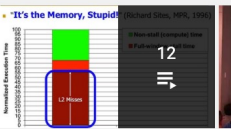
Perugia NIPS Summer School
2019

Onur Mutlu Lectures
VIEW FULL PLAYLIST



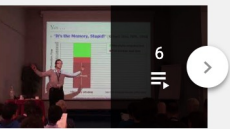
SAMOS Tutorial 2019 - Memory
Systems

Onur Mutlu Lectures
VIEW FULL PLAYLIST



TU Wien 2019 - Memory
Systems and Memory-Centric...

Onur Mutlu Lectures
VIEW FULL PLAYLIST



ACACES 2018 Lectures --
Memory Systems and Memory...

Onur Mutlu Lectures
VIEW FULL PLAYLIST

Research Talks

<https://www.youtube.com/onurmutlulectures>

SAFARI

SAFARI EFCL Research Projects: Recent Results and Future Outlook

Onur Mutlu

omutlu@gmail.com

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

23 May 2022

EFCL Mini-Conference

Backup Slides (for More Detail)

Brief Self Introduction



■ Onur Mutlu

- ❑ Full Professor @ ETH Zurich ITET (INFK), since Sept 2015
- ❑ Strecker Professor @ Carnegie Mellon University ECE (CS), 2009-2016, 2016-...
- ❑ Started the Comp Arch Research Group @ Microsoft Research, 2006-2009
- ❑ Worked @ Google, VMware, Microsoft Research, Intel, AMD
- ❑ PhD in Computer Engineering from University of Texas at Austin in 2006
- ❑ BS in Computer Engineering & Psychology from University of Michigan in 2000
- ❑ <https://people.inf.ethz.ch/omutlu/> omutlu@gmail.com

■ Research and Teaching in:

- ❑ **Computer architecture, systems, hardware security, bioinformatics**
- ❑ Memory and storage systems
- ❑ Robust & dependable hardware systems: security, safety, predictability, reliability
- ❑ Hardware/software cooperation
- ❑ New computing paradigms; architectures with emerging technologies/devices
- ❑ Architectures for bioinformatics, genomics, health, medicine, AI/ML
- ❑ ...

A New Methodology and
Open-Source Benchmark Suite
for Evaluating Data Movement Bottlenecks:
A Processing-in-Memory Case Study

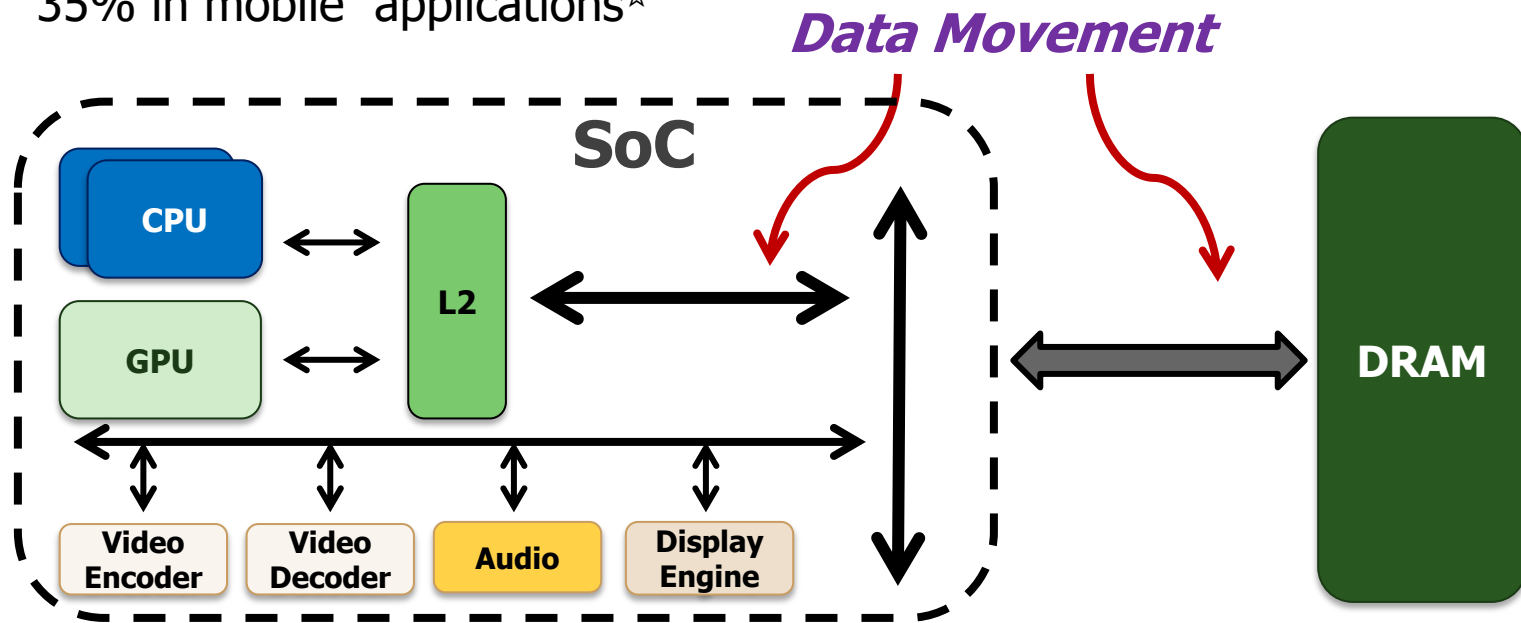
Backup Slides

Executive Summary

- Data movement between memory units and compute units is a major system bottleneck
- Many workloads suffer from the **data movement bottleneck**
 - Machine learning, computational biology, graph processing, databases, video analytics, real-time data analytics...
- The traditional processor-centric design should evolve to **a more data-centric design where processing elements are closer to where the data resides**
 - Near-data processing (NDP), processing-in-memory (PIM)
 - Processing-immersed-with-memory (monolithic 3D integration)
- Some **main challenges for adoption**
 - Identifying suitable workloads and functions
 - Lack of profiling tools, analytical models, simulators...

Data Movement in Computing Systems

- **Data movement** dominates **performance** and is a major system **energy bottleneck**
- **Total system energy**: data movement accounts for
 - ❑ 62% in consumer applications*,
 - ❑ 40% in scientific applications*,
 - ❑ 35% in mobile applications☆



* Boroumand et al., "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," ASPLOS 2018

* Kestor et al., "Quantifying the Energy Cost of Data Movement in Scientific Applications," IISWC 2013

☆ Pandiyan and Wu, "Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms," IISWC 2014

Our Goal

- To **enable adoptable PIM systems** by solving various key challenges
 - **Identifying workloads and functions** that are suitable for processing near/inside where the data resides
 - Fundamental (architecture-independent) characterization
 - Architecture-specific suitability
 - **Lack of tools**
 - Profiling tools
 - Analytical performance and energy models (or ML-based models)
 - Simulation tools
- The project is organized in **two phases**
 - Phase 1: Methodology and benchmark suite
 - Phase 2: Follow-up project to enable PIM adoption

Phase 1: Methodology and Benchmark Suite

- We aim to develop understanding into modern workloads with the key goal of identifying workload characteristics and portions that would be beneficial to offload to a PIM engine
- We intend to develop:
 - ❑ New profiling tools
 - ❑ Analytical and simulation models
 - ❑ Benchmark suites
- Three anticipated steps:
 - ❑ 1. Application profiling on modern processors
 - ❑ 2. Application characterization
 - ❑ 3. Performance analysis and validation

Phase 1: Three Anticipated Steps (I)

- Step 1: Application profiling on modern processors
 - We will use
 - Profiling tools: VTune, perf, nvprof
 - Relevant metrics (e.g., cache misses, DRAM accesses, data locality)
 - Outcomes
 - Motivate the need for PIM and type of PIM
 - Strong empirical understanding about workload characteristics and suitable portions

Phase 1: Three Anticipated Steps (II)

- Step 2: **Application characterization**
 - Analysis of metrics
 - Architecture-independent and architecture-dependent metrics
 - Identification of key metrics
 - Huge design space
 - Types of cores, accelerators, functional units
 - Processing models on host and PIM side

Phase 1: Three Anticipated Steps (III)

- Step 3: Performance analysis and validation
 - Rigorous analysis and validation
 - Commonalities between functions
 - Development and evaluation of general-purpose and special-purpose PIM

Phase 1: Target Applications

- We will apply our methodology to **more than 500 applications**
 - ❑ Benchmark suites, frameworks
 - ❑ Custom open-source version
- **Major domains**
 - ❑ Machine learning
 - ❑ Graph processing
 - ❑ Data analytics
 - ❑ Databases
 - ❑ Bioinformatics
 - ❑ Image/video processing
 - ❑ Physics simulation
 - ❑ Etc.

Phase 1: Types of PIM

■ Where?

- ❑ Processing in logic layer of 3D stacked memories
- ❑ Processing in memory controllers on die
- ❑ In-DRAM processing ala Ambit
- ❑ In-DRAM processing ala UPMEM
- ❑ In-cache processing

■ What?

- ❑ General-purpose and special-purpose PIM
- ❑ Host: CPU, GPU, FPGA, accelerators

Phase 1: Tools

- Profiling tools
 - VTune
 - Perf
 - Pin
- Simulation tools
 - Ramulator
 - Gem5
 - GPGPU-Sim
 - Ramulator-PIM

DAMOV Analysis Methodology & Workloads

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

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

LOIS OROSA, ETH Zürich, Switzerland

SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA

NANDITA VIJAYKUMAR, University of Toronto, Canada

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

MOHAMMAD SADROSADATI, Institute for Research in Fundamental Sciences (IPM), Iran & ETH Zürich, Switzerland

ONUR MUTLU, ETH Zürich, Switzerland

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

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

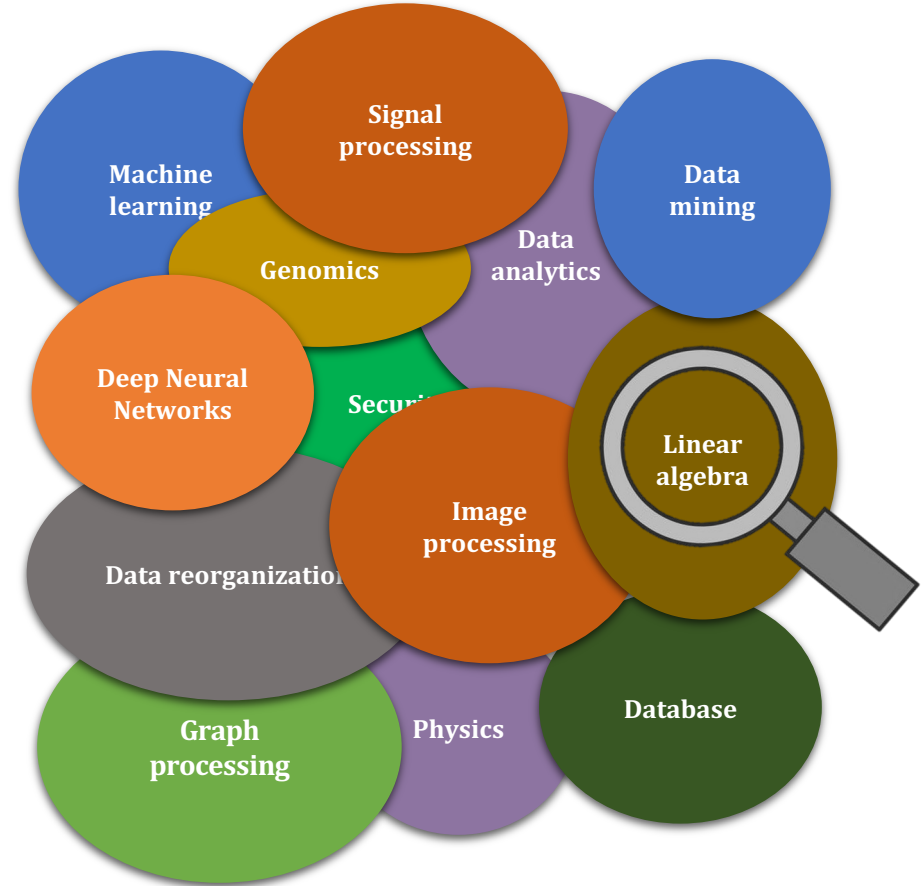
Step 1: Application Profiling

- We analyze 345 applications from distinct domains:

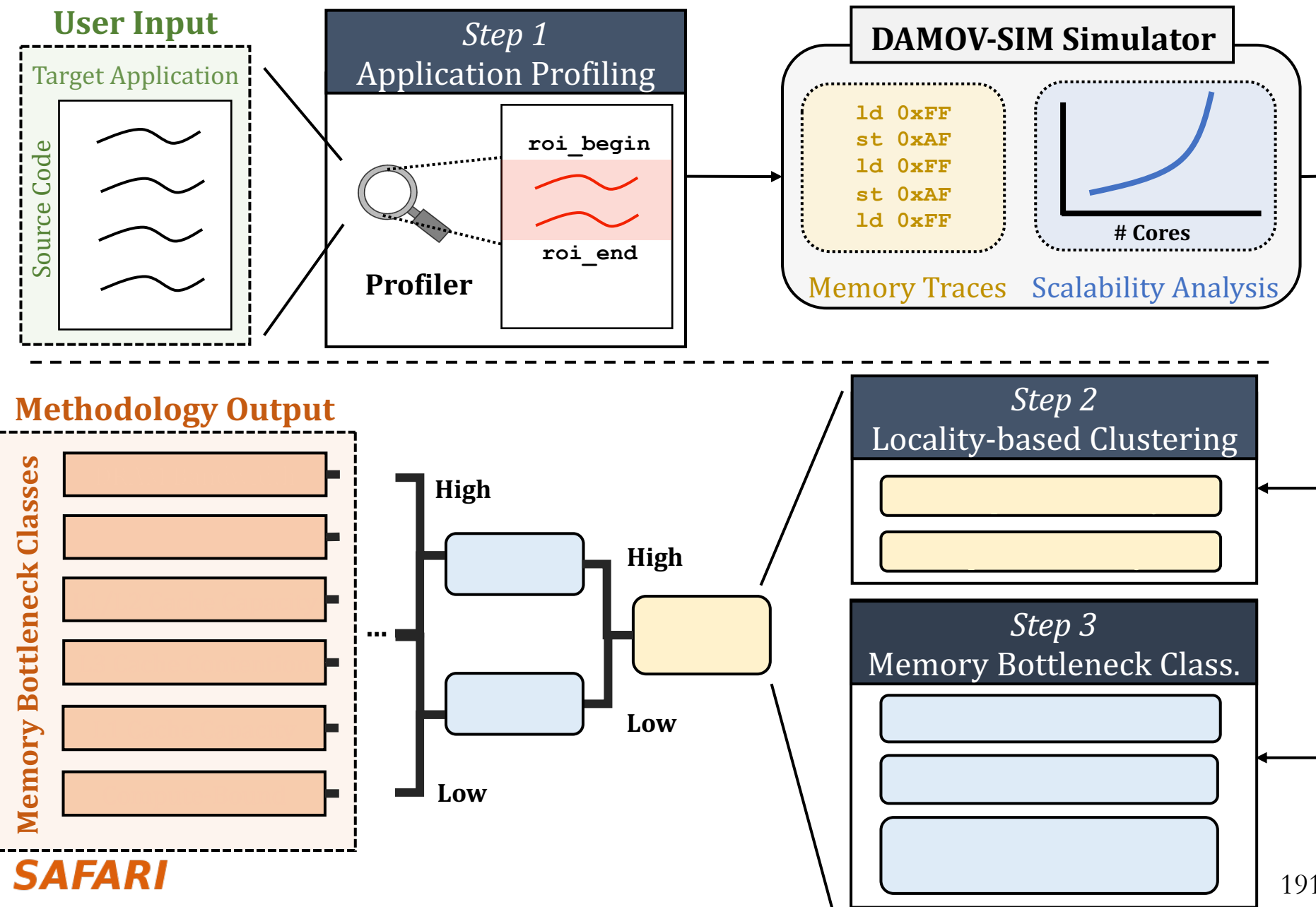
- Graph Processing
- Deep Neural Networks
- Physics
- High-Performance Computing
- Genomics
- Machine Learning
- Databases
- Data Reorganization
- Image Processing
- Map-Reduce
- Benchmarking
- Linear Algebra

...

SAFARI



Methodology Overview



DAMOV is Open Source

- We open-source our **benchmark suite** and our **toolchain**

CMU-SAFARI / DAMOV

<> Code Issues Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags

Go to file

Add file

Code

About

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 processing. Described by Oliveira et al. (preliminary version at <https://arxiv.org/pdf/2105.03725.pdf>)

Readme

Releases

No releases published
[Create a new release](#)

Packages

omutlu Update README.md

ce1b4ea 17 days ago 5 commits

simulator	Cleaning	19 days ago
README.md	Update README.md	17 days ago
get_workloads.sh	DAMOV -- first commit	19 days ago

README.md

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

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 processing.

The DAMOV benchmark suite is the first open-source benchmark suite for near-data processing-related

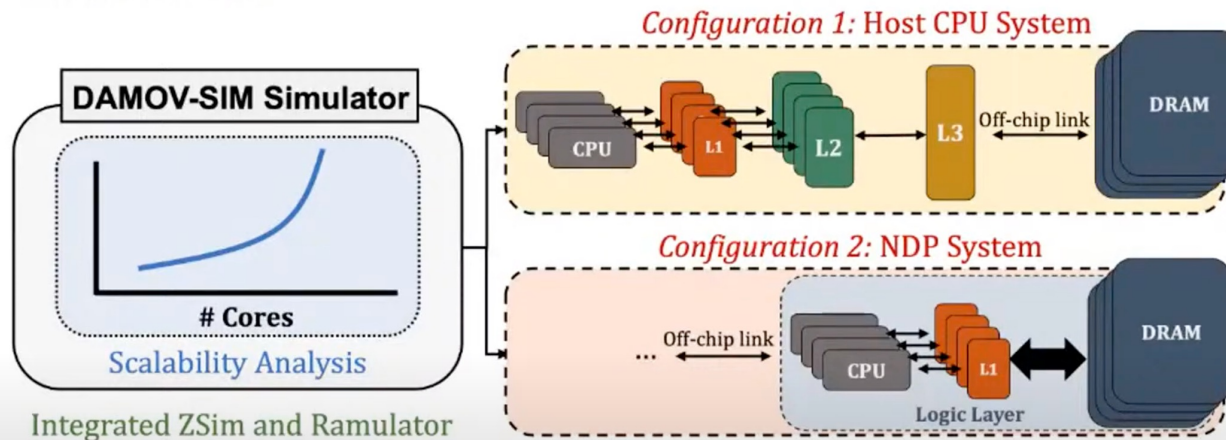
Get DAMOV at:

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

More on DAMOV Analysis Methodology & Workloads

Step 3: Memory Bottleneck Classification (2/)

- **Goal:** identify the specific sources of data movement bottlenecks



- **Scalability Analysis:**
 - 1, 4, 16, 64, and 256 out-of-order/in-order host and NDP CPU cores
 - 3D-stacked memory as main memory

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

SAFARI Live Seminar: DAMOV: A New Methodology & Benchmark Suite for Data Movement Bottlenecks

352 views • Streamed live on Jul 22, 2021



Onur Mutlu Lectures
17.7K subscribers

18 0 SHARE SAVE ...

ANALYTICS

EDIT VIDEO

SAFARI

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

Experimental Analysis of the UPMEM PIM Engine

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

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

IZZAT EL HAJJ, American University of Beirut, Lebanon

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

CHRISTINA GIANNOULA, ETH Zürich, Switzerland and NTUA, Greece

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland

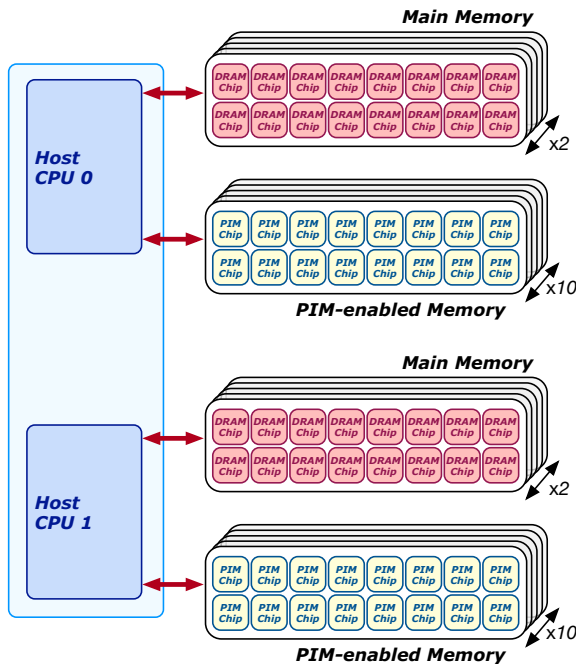
ONUR MUTLU, ETH Zürich, Switzerland

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

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

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

2,560-DPU Processing-in-Memory System



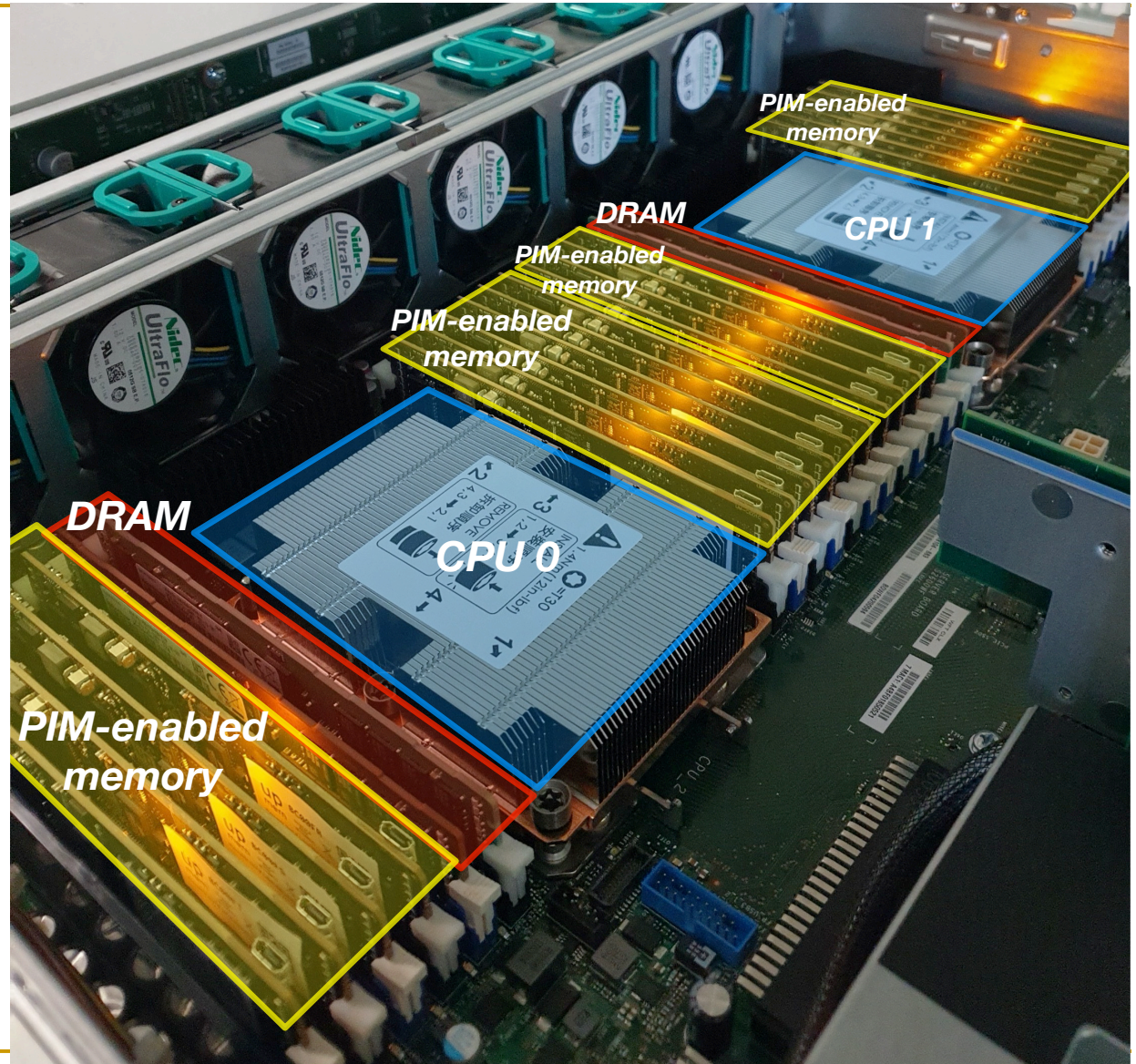
Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
 IZZAT EL HAJJ, American University of Beirut, Lebanon
 IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain
 CHRISTINA GIANNOULA, ETH Zürich, Switzerland and NTUA, Greece
 GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
 ONUR MUTLU, ETH Zürich, Switzerland

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

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

This paper provides the first comprehensive analysis of the first publicly-available real-world PIM architecture. We make two key contributions. First, we conduct an experimental characterization of the UPMEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present *PrIM (Processing-In-Memory benchmarks)*, a benchmark suite of 16 workloads from different application domains (e.g., dense/sparse linear algebra, databases, data analytics, graph processing, neural networks, bioinformatics, image processing), which we identify as memory-bound. We evaluate the performance and scaling characteristics of PrIM benchmarks on the UPMEM PIM architecture, and compare their performance and energy consumption to their state-of-the-art CPU and GPU counterparts. Our extensive evaluation conducted on two real UPMEM-based PIM systems with 640 and 2,560 DPUs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.



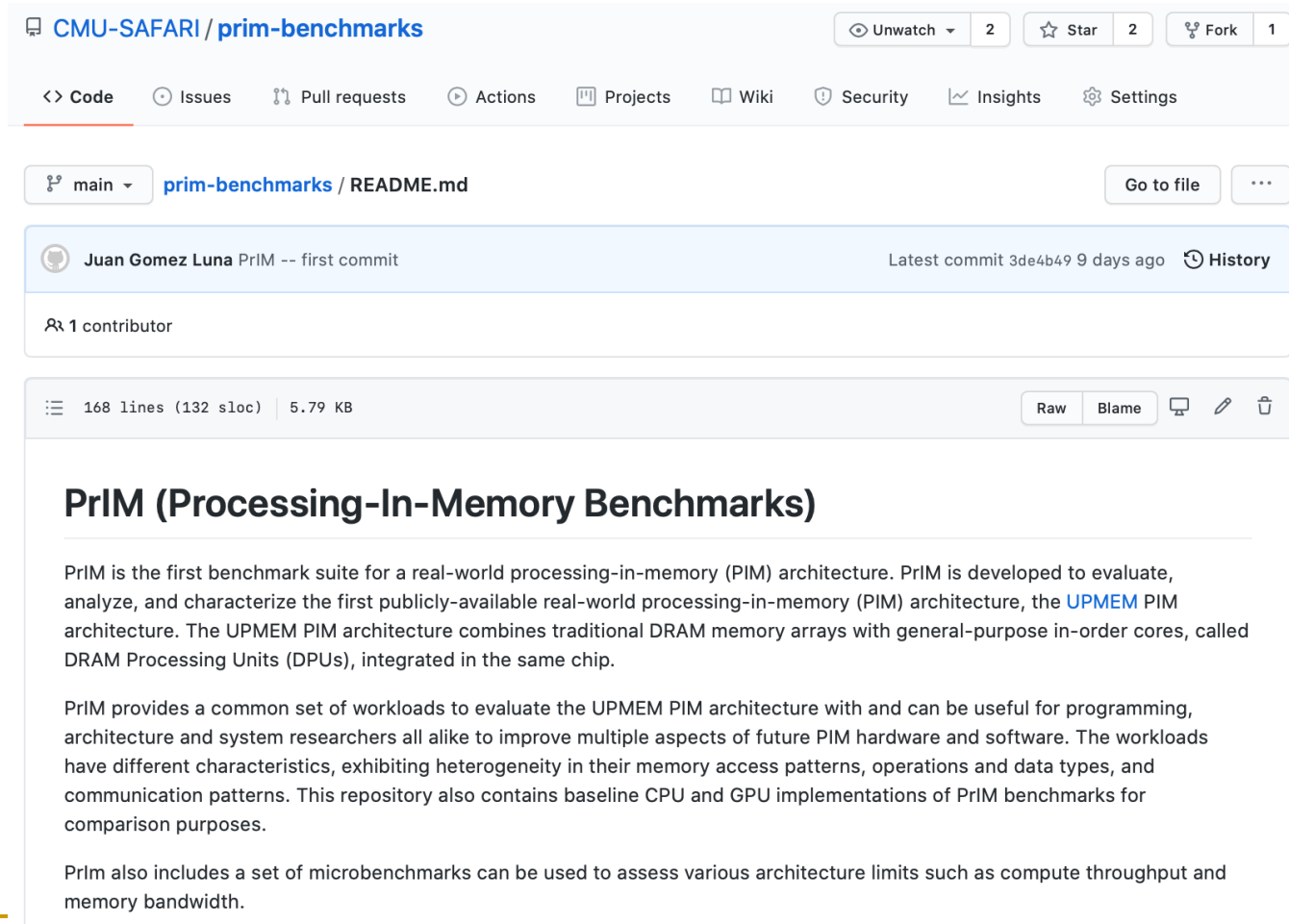
<https://arxiv.org/pdf/2105.03814.pdf>

PrIM Benchmarks: Application Domains

Domain	Benchmark	Short name
Dense linear algebra	Vector Addition	VA
	Matrix-Vector Multiply	GEMV
Sparse linear algebra	Sparse Matrix-Vector Multiply	SpMV
Databases	Select	SEL
	Unique	UNI
Data analytics	Binary Search	BS
	Time Series Analysis	TS
Graph processing	Breadth-First Search	BFS
Neural networks	Multilayer Perceptron	MLP
Bioinformatics	Needleman-Wunsch	NW
Image processing	Image histogram (short)	HST-S
	Image histogram (large)	HST-L
Parallel primitives	Reduction	RED
	Prefix sum (scan-scan-add)	SCAN-SSA
	Prefix sum (reduce-scan-scan)	SCAN-RSS
	Matrix transposition	TRNS

PrIM Benchmarks are Open Source

- All microbenchmarks, benchmarks, and scripts
- <https://github.com/CMU-SAFARI/prim-benchmarks>



CMU-SAFARI / **prim-benchmarks** Unwatch 2 Star 2 Fork 1

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main prim-benchmarks / README.md Go to file ...

Juan Gomez Luna PrIM -- first commit Latest commit 3de4b49 9 days ago History

1 contributor

168 lines (132 sloc) 5.79 KB Raw Blame

PrIM (Processing-In-Memory Benchmarks)

PrIM is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. PrIM is developed to evaluate, analyze, and characterize the first publicly-available real-world processing-in-memory (PIM) architecture, the [UPMEM PIM](#) architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called DRAM Processing Units (DPUs), integrated in the same chip.

PrIM provides a common set of workloads to evaluate the UPMEM PIM architecture with and can be useful for programming, architecture and system researchers all alike to improve multiple aspects of future PIM hardware and software. The workloads have different characteristics, exhibiting heterogeneity in their memory access patterns, operations and data types, and communication patterns. This repository also contains baseline CPU and GPU implementations of PrIM benchmarks for comparison purposes.

PrIM also includes a set of microbenchmarks can be used to assess various architecture limits such as compute throughput and memory bandwidth.

Understanding a Modern PIM Architecture



The image shows a YouTube video player for a live seminar. The video title is "Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization". The speaker is Juan Gómez Luna, with co-speakers Izzat El Hajj, Ivan Fernandez, Christina Giannoula, and Geraldo F. Oliveira, Onur Mutlu. The video includes links to an arXiv paper and a GitHub repository. The player shows a progress bar at 2:26 / 2:57:10. The channel is "Onur Mutlu Lectures" with 18.7K subscribers. The video has 2,579 views and was streamed live on Jul 12, 2021. The player includes like, comment, share, and save buttons. The channel name "Onur Mutlu Lectures" and subscriber count "18.7K subscribers" are visible. The video has 93 likes and 0 comments. The channel is subscribed to.

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

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

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

ETH Zürich SAFARI

2:26 / 2:57:10

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

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

93 0 SHARE SAVE ...

Onur Mutlu Lectures
18.7K subscribers

SUBSCRIBED

Phase 2: Projects to Enable PIM Adoption (I)

- Learnings and artifacts (e.g., benchmark suites) from Phase 1 will allow us to start the **exploration of the following lines of research**

- 1. **Design space exploration for PIM hardware accelerators**
 - Understand hardware requirements of a vast range of dataflow-based PIM accelerators
 - Near-memory hardware accelerators for PIM-friendly workloads
 - Pre-RTL power-performance accelerator simulator
 - PIM hardware library to instantiate by workloads

Phase 2: Projects to Enable PIM Adoption (II)

■ 2. Runtime scheduling

- Runtime schedulers that can automatically identify when to offload a code segment to a PIM core
- Considerations:
 - Data movement
 - Memory coherence
 - Memory interference between applications and PIM cores
 - Offloading to different levels of the memory hierarchy

Phase 2: Projects to Enable PIM Adoption (III)

■ 3. PIM API

- Search for common computation patterns
- PIM API can facilitate PIM programmability
- Techniques to maximize performance, bandwidth utilization, to reduce communication

Phase 2: Projects to Enable PIM Adoption (IV)

- 4. Extending our methodology to other levels of the memory/storage hierarchy
 - Computation offloading to other parts of the memory hierarchy
 - Near-cache computing
 - Near-storage computing
 - Memory technologies other than DRAM

Term / Timeline

- The project plan spans 3 years

Project plan for Phase 1	Year 1				Year 2				Year 3			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Application profiling												
Application characterization												
Performance analysis and validation												

Project plan for Phase 2	Year 1				Year 2				Year 3			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Design space exploration												
Runtime scheduling												
PIM API												
Extending our methodology												

Expected Outcomes

- **At the completion of Phase 1** of this project
 - ❑ 1. Tools for identification of PIM suitability
 - ❑ 2. Models for understanding and evaluating PIM architectures
 - ❑ 3. PIM benchmark suites

- **At the completion of Phase 2** of this project
 - ❑ 1. Design space exploration of PIM accelerators
 - ❑ 2. Runtime schedulers
 - ❑ 3. Tools for PIM programming
 - ❑ 4. Tools for PIM suitability across other levels of memory/storage

Machine-Learning-Assisted Intelligent Microarchitectures to Reduce Memory Access Latency

Backup Slides



Pythia

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera, Konstantinos Kanellopoulos, Anant V. Nori,
Taha Shahroodi, Sreenivas Subramoney, Onur Mutlu

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



Executive Summary

- **Background:** Prefetchers predict addresses of future memory requests by associating memory access patterns with program context (called **feature**)
- **Problem:** Three key shortcomings of prior prefetchers:
 - Predict mainly using a **single program feature**
 - Lack **inherent system awareness** (e.g., memory bandwidth usage)
 - Lack **in-silicon customizability**
- **Goal:** Design a prefetching framework that:
 - Learns from **multiple features** and **inherent system-level feedback**
 - Can be **customized in silicon** to use different features and/or prefetching objectives
- **Contribution:** Pythia, which formulates prefetching as reinforcement learning problem
 - Takes **adaptive** prefetch decisions using multiple features and system-level feedback
 - Can be **customized in silicon** for target workloads via simple configuration registers
 - Proposes a **realistic and practical** implementation of RL algorithm in hardware
- **Key Results:**
 - Evaluated using a wide range of workloads from SPEC CPU, PARSEC, Ligra, Cloudsuite
 - Outperforms best prefetcher (in 1-core config.) by **3.4%, 7.7% and 17%** in 1/4/bw-constrained cores
 - Up to **7.8% more performance** over basic Pythia across Ligra workloads via simple customization

Key Shortcomings in Prior Prefetchers

- We observe **three key shortcomings** that significantly limit performance benefits of prior prefetchers

1 Predict mainly using a **single program feature**

2 Lack inherent **system awareness**

3 Lack **in-silicon customizability**

Our Goal

A **prefetching framework** that can:

1. Learn to prefetch using **multiple features** and **inherent system-level feedback** information
2. Be **easily customized in silicon** to use different features and/or change prefetcher's objectives

Our Proposal



Pythia

Formulates prefetching as a
reinforcement learning problem

Basics of Reinforcement Learning (RL)

- Algorithmic approach to learn to take an **action** in a given **situation** to maximize a numerical **reward**

Agent

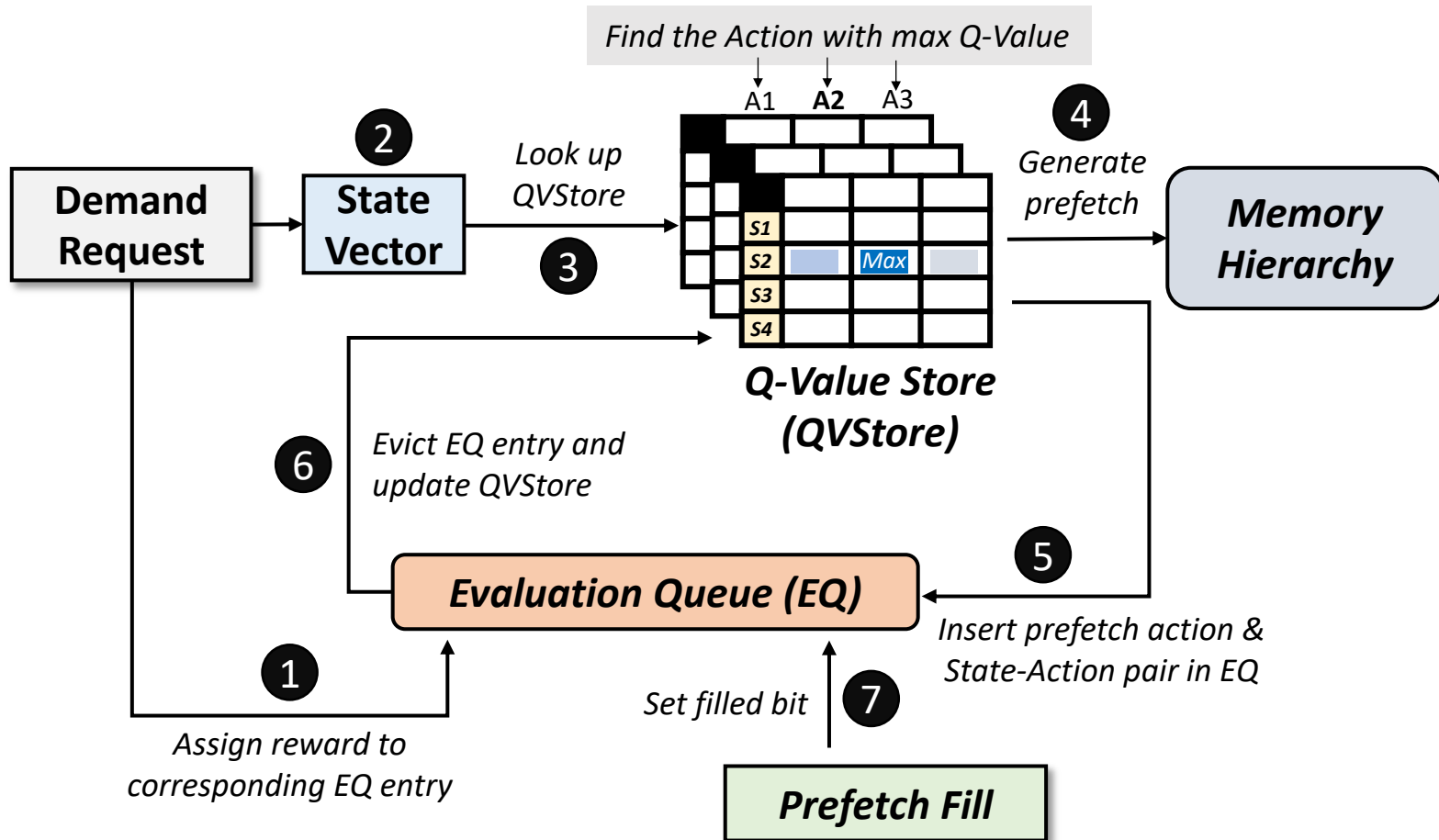
Environment

- Agent stores **Q-values** for *every* state-action pair
 - **Expected return** for taking an action in a state
 - Given a state, selects action that provides **highest** Q-value

Formulating Prefetching as RL

Pythia Overview

- **Q-Value Store**: Records Q-values for *all* state-action pairs
- **Evaluation Queue**: A FIFO queue of recently-taken actions



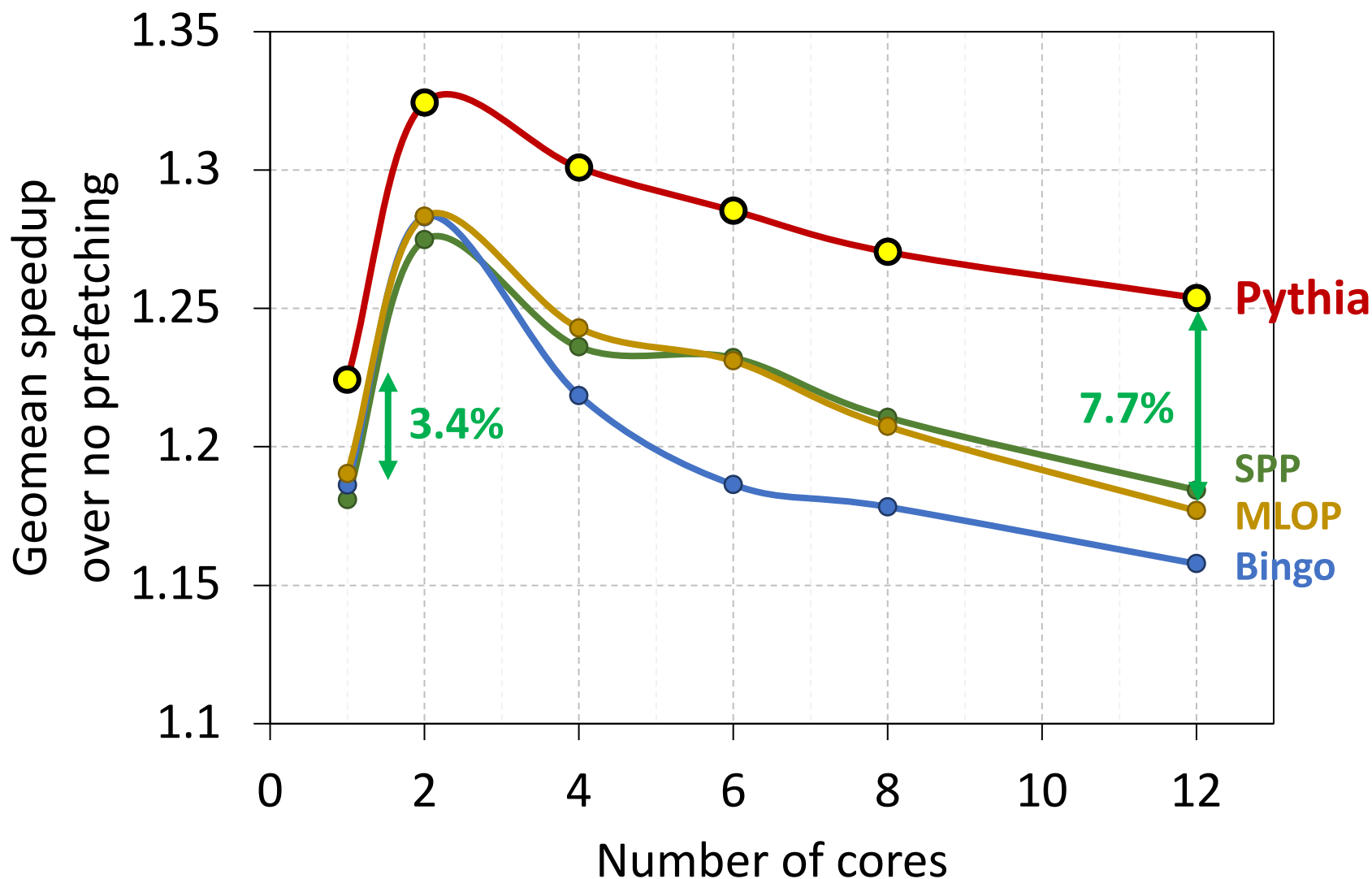
Simulation Methodology

- **Champsim** [3] trace-driven simulator
- **150** single-core memory-intensive workload traces
 - SPEC CPU2006 and CPU2017
 - PARSEC 2.1
 - Ligra
 - Cloudsuite
- Homogeneous and heterogeneous multi-core mixes
- **Five** state-of-the-art prefetchers
 - SPP [Kim+, MICRO'16]
 - Bingo [Bakhshalipour+, HPCA'19]
 - MLOP [Shakerinava+, 3rd Prefetching Championship, 2019]
 - SPP+DSPatch [Bera+, MICRO'19]
 - SPP+PPF [Bhatia+, ISCA'20]

Basic Pythia Configuration

- Derived from **automatic design-space exploration**
- **State:** 2 features
 - PC+Delta
 - Sequence of last-4 deltas
- **Actions:** 16 prefetch offsets
 - Ranging between -6 to +32. Including 0.
- **Rewards:**
 - $R_{AT} = +20$; $R_{AL} = +12$; $R_{NP-H} = -2$; $R_{NP-L} = -4$;
 - $R_{IN-H} = -14$; $R_{IN-L} = -8$; $R_{CL} = -12$

Performance with Varying Core Count



Performance with Varying Core Count

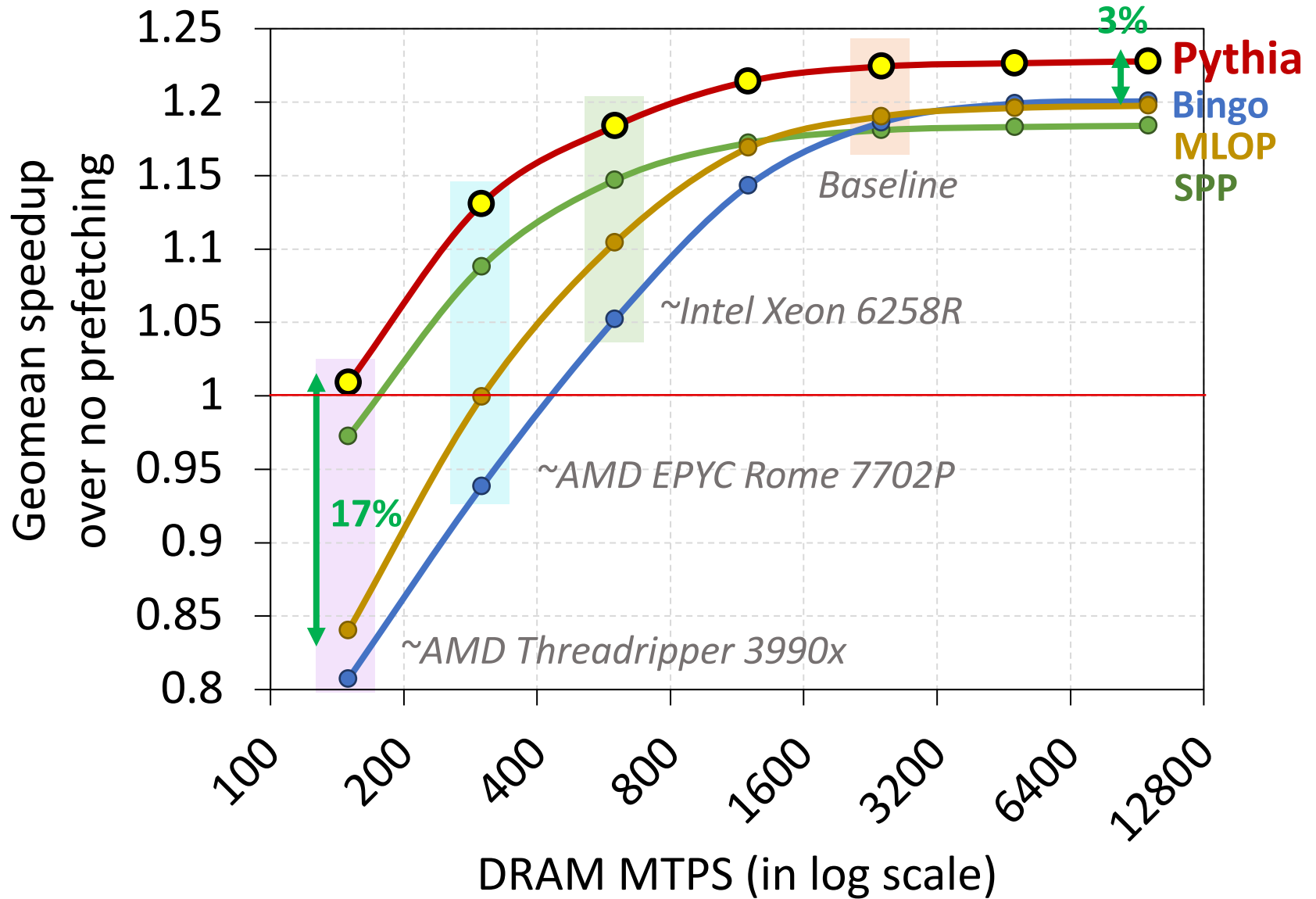


The graph shows performance on the y-axis (ranging from 1.1 to 1.35) against the number of cores on the x-axis (ranging from 0 to 12). Pythia (red line) starts at ~1.28 at 2 cores and peaks at ~1.32 at 4 cores. Other models (blue, green, orange lines) show lower performance, with a 3.4% gain noted for one model at 2 cores. A vertical green line at 12 cores is labeled 'SDD'.

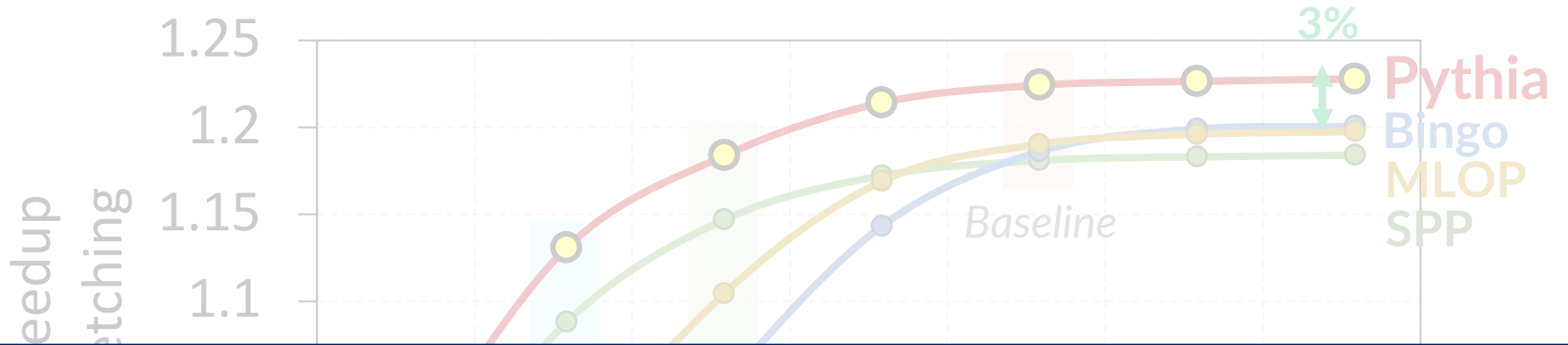
1. Pythia consistently provides the highest performance in **all core configurations**

2. Pythia's gain **increases with core count**

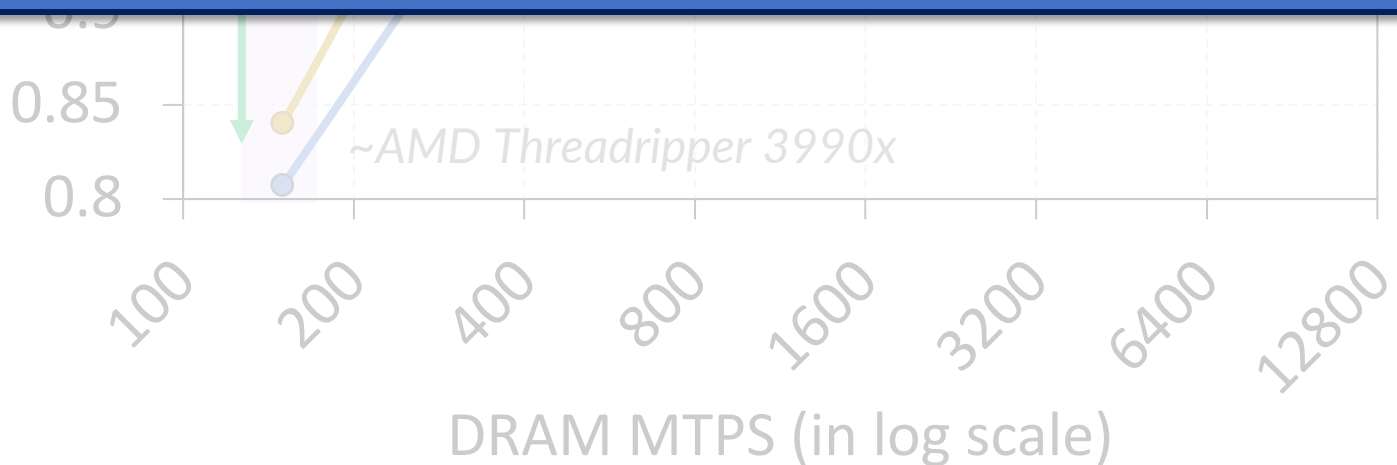
Performance with Varying DRAM Bandwidth



Performance with Varying DRAM Bandwidth



Pythia outperforms prior best prefetchers for a wide range of DRAM bandwidth configurations



Pythia's Overhead

- **25.5 KB** of total metadata storage **per core**
 - Only simple tables
- We also model functionally-accurate Pythia with full complexity in **Chisel** [4] HDL



1.03% area overhead



0.4% power overhead



Satisfies prediction latency

of a desktop-class 4-core Skylake processor (Xeon D2132IT, 60W)

More in the Paper

- Performance comparison with **unseen traces**
 - Pythia provides equally high performance benefits

• Comparison against **multi-level prefetchers**

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>

- **Performance sensitivity** towards different features and hyperparameter values

- Detailed single-core and four-core performance

Pythia is Open Source



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

- MICRO'21 **artifact evaluated**
- **Champsim source** code + **Chisel** modeling code
- **All traces** used for evaluation

The screenshot shows the GitHub repository for CMU-SAFARI/Pythia. The repository is public and has 3 unwatchers, 7 stars, and 2 forks. The main navigation bar includes links to Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The repository is currently on the master branch, with 1 branch and 5 tags. The commit history table shows the following entries:

Commit	Message	Time
f96dee9	Updated README	2 days ago
	branch	Initial commit for MICRO'21 artifact evaluation, 2 months ago
	config	Initial commit for MICRO'21 artifact evaluation, 2 months ago
	experiments	Added chart visualization in Excel template, 2 months ago
	inc	Updated README, 6 days ago
	prefetcher	Initial commit for MICRO'21 artifact evaluation, 2 months ago
	replacement	Initial commit for MICRO'21 artifact evaluation, 2 months ago
	scripts	Added md5 checksum for all artifact traces to verify download, 2 months ago
	src	Initial commit for MICRO'21 artifact evaluation, 2 months ago
	tracer	Initial commit for MICRO'21 artifact evaluation, 2 months ago
	.gitignore	Initial commit for MICRO'21 artifact evaluation, 2 months ago
	CITATION.cff	Added citation file, 6 days ago
	LICENSE	Updated LICENSE, 2 months ago
	LICENSE.champsim	Initial commit for MICRO'21 artifact evaluation, 2 months ago

The right sidebar contains the 'About' section, which describes Pythia as a customizable hardware prefetching framework using online reinforcement learning. It includes a link to the arXiv paper (arxiv.org/pdf/2109.12021.pdf) and a list of tags: machine-learning, reinforcement-learning, computer-architecture, prefetcher, microarchitecture, cache-replacement, branch-predictor, champsim-simulator, and champsim-tracer. Below the 'About' section are links to the README, View license, and Cite this repository. The 'Releases' section shows 5 releases.



Pythia

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera, Konstantinos Kanellopoulos, Anant V. Nori,
Taha Shahroodi, Sreenivas Subramoney, Onur Mutlu

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



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 Bera¹ Konstantinos Kanellopoulos¹ Anant V. Nori² Taha Shahroodi^{3,1}
Sreenivas Subramoney² Onur Mutlu¹

¹ETH Zürich

²Processor Architecture Research Labs, Intel Labs

³TU Delft

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 Hybrid Storage Systems

- To appear in ISCA 2022

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

<https://arxiv.org/pdf/2205.07394.pdf>

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, Onur Mutlu

Executive Summary

Background: Hybrid storage systems (HSSs) complement different storage technologies to extend the overall capacity and reduce the system cost with minimal effect on the application performance

Problem: Accurately identify the performance-critical data of an application and placing it in the “best-fit” storage device. Three key shortcomings of prior data placement policies (heuristic-based and supervised learning-based) of hybrid storage systems:

- Lack of **adaptability**
- Lack of **device awareness (e.g., read/write latencies of each device)**
- Lack of **extensibility**

Goal: Develop a new, efficient, and high performance data-placement mechanism for hybrid storage systems that can:

- Dynamically derive an adaptive data-placement strategy by **continuously learning and adapting** to the **application and underlying device characteristics**
- **Easily extensible** to incorporate a wide range of hybrid storage configurations.

Key Idea: Sibyl, an online reinforcement learning-based self-adaptable mechanism for data placement that:

- **Dynamically learns** from past experiences and **continuously adapts** its policy to improve long-term performance by interacting with the hybrid storage system
- **Learns the asymmetry in the read/write latencies** present in modern hybrid storage devices while **taking into account** the **inherent characteristics of an application**

Key Results: Sibyl is evaluated on a real system with multiple device configurations

- Evaluated using a **wide range of workloads** from MSR Cambridge and Filebench
- In a performance (cost) optimized hybrid storage configuration, Sibyl provides up to **21.6% (19.9%)** performance improvement compared to prior data placement policies
- On a tri-hybrid storage system, Sibyl outperforms a heuristics-based policy by **23.9% -48.2%**
- Sibyl achieves **80%** performance of an oracle policy with storage overhead of **124.4 KiB**

Outline

Background

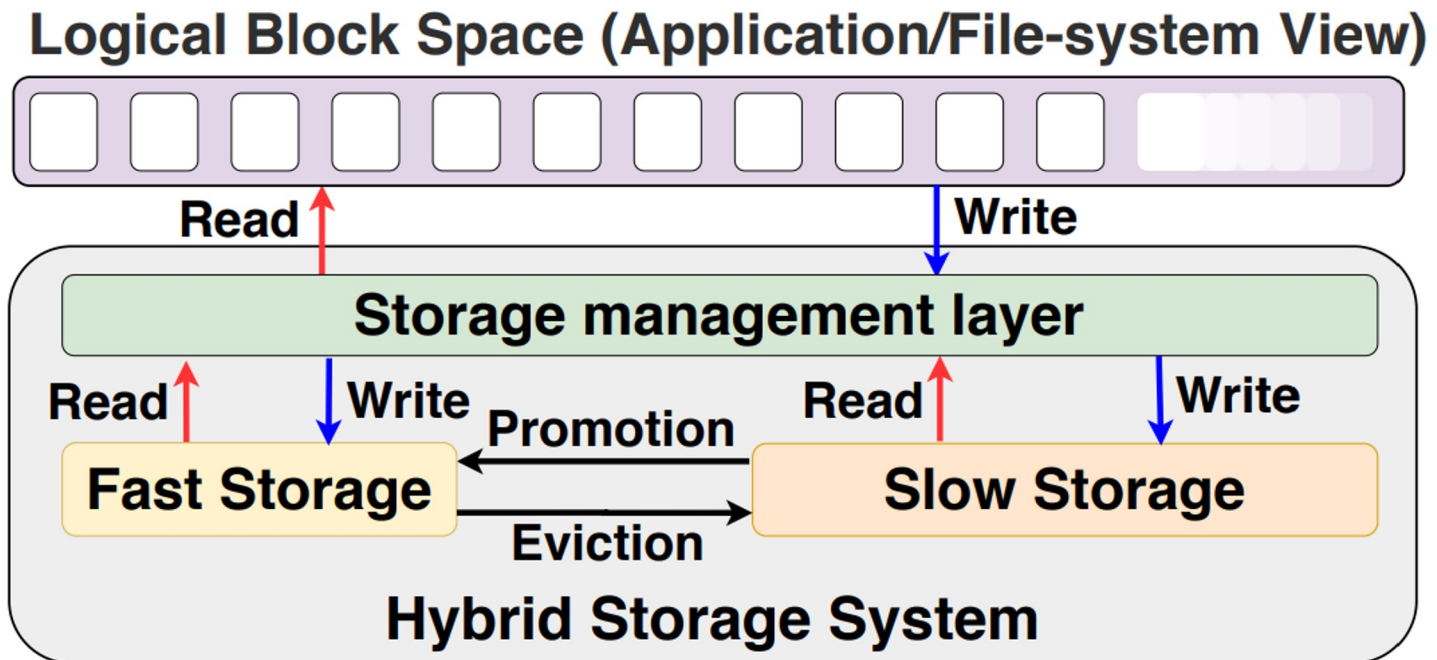
Formulating Data Placement as RL problem

Sibyl: Overview

Evaluation and Key Results

Conclusion

Hybrid Storage Systems



Key Shortcomings of Prior Data Placement Techniques

We observe **three key shortcomings** that significantly limit performance benefits of data-placement techniques

Lack of **adaptability**

Lack of **device awareness**

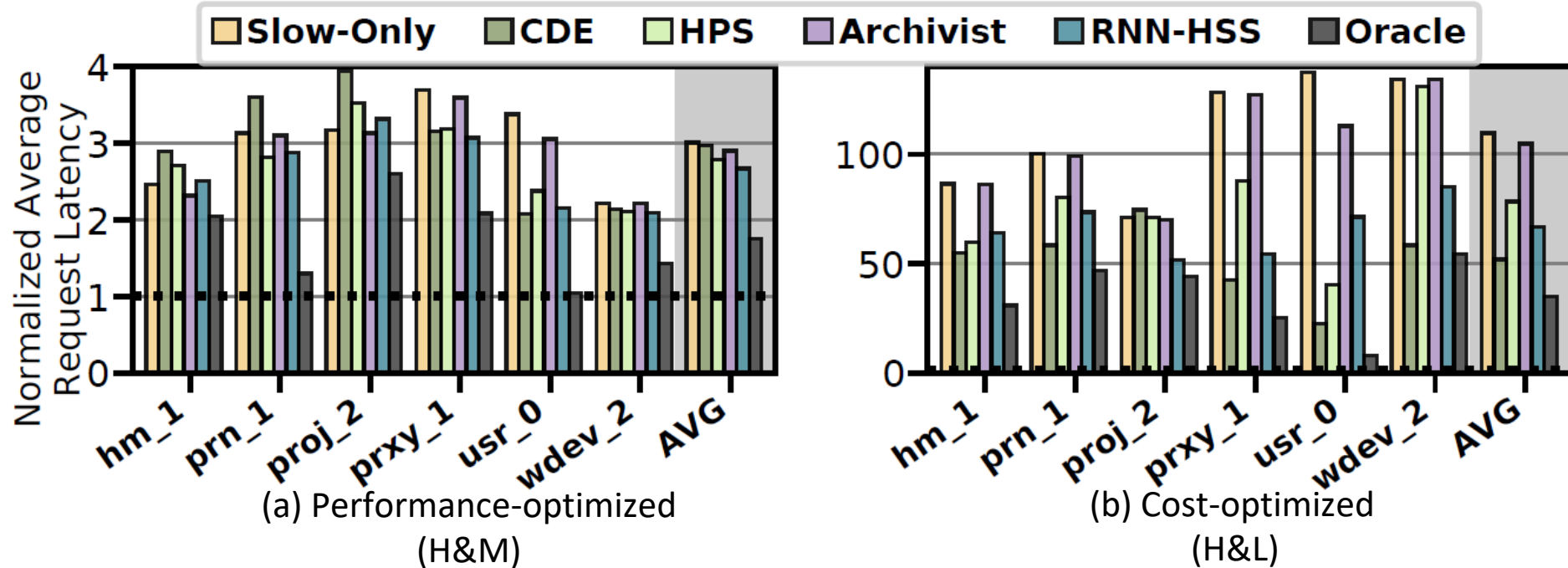
Lack of **extensibility**

Lack of Adaptability (1/2)

- Prior heuristic-based techniques consider only a **few characteristics** (e.g., access frequency) to perform data placement
- **Statically tuned characteristics** (based on **fixed thresholds**) are ineffective when used on a wide range of applications and system configurations
- Supervised learning techniques need **labeled data** and **frequent retraining** to adapt to varying workloads and system conditions

Prior techniques offer **41.1% lower performance** compared to an Oracle policy

Lack of Adaptability (2/2)



CDE shows an average performance gap of 41.1% (32.6%) to Oracle for H&M (H&L)

HPS shows an average performance gap of 37.2% (55.5%) to Oracle for H&M (H&L)

Lack of Device Awareness

Prior data placement techniques:

- **do not adapt** well to changes in underlying device characteristics (e.g., storage read latency)
- **do not consider the data migration cost** between storage devices while making a data placement decision
- **are highly inefficient** in hybrid storage systems that have devices with significantly different read/write latencies

Lack of Extensibility

- Prior data placement techniques are typically **designed** for a hybrid storage system with **only two storage devices**
- **Significant effort** is required to extend the data placement policies for more than two devices

Compared to a RL-based solution, a heuristic-based policy provides **48.2% lower performance** when extended from two to three devices

Our Goal

A **data-placement mechanism** that can

- dynamically derive an adaptive data-placement strategy by **continuously learning** and **adapting** to the application and underlying device characteristics
- be **easily extended** to incorporate a wide range of hybrid storage configurations

Outline

Background

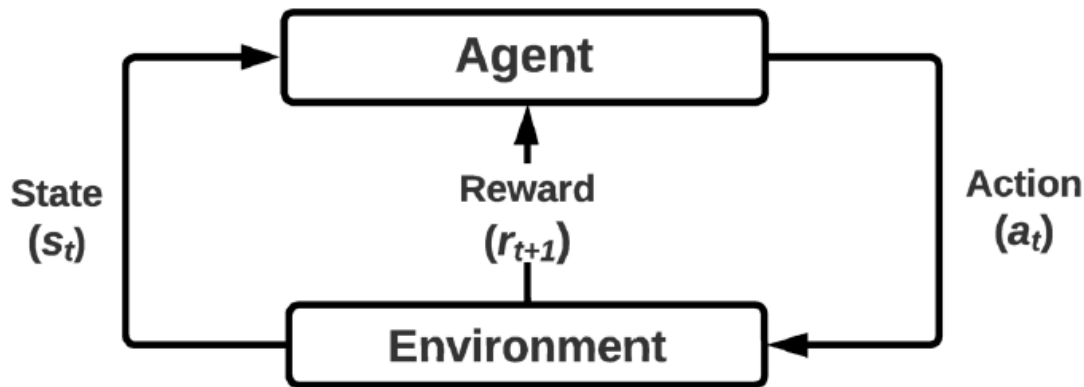
Formulating Data Placement as RL problem

Sibyl: Overview

Evaluation and Key Results

Conclusion

Basics of Reinforcement Learning



- RL is a framework for decision making
 - An **autonomous agent observes** the current **state** of the environment
 - It **interacts** with the environment by taking **actions**
 - Agent is **rewarded** or **penalized** based on the consequences of its actions
 - Agent tries to maximize the cumulative reward

Applying RL to Data Placement

Key factors in applying RL for data placement in a hybrid storage system

- RL agent needs to be **aware of:**
 - **asymmetry in read/write latencies** of a storage device
 - **differences in latencies** across hybrid storage devices
 - **application access patterns**
- Data placement module should decide which actions to reward and penalize (credit assignment)
- Low implementation overhead

RL Formulation

- Sibyl observes **multiple application/device features** for every storage request to make a data placement decision
- Possible actions - Placing data in fast or slow device
- For **every action**, Sibyl receives a **reward** that takes into account the data placement decision and state of the environment
- Sibyl finds an **optimal data placement policy** that increases the overall performance for any workload and system configuration

RL State

- Feature selection is performed to select only the most correlated features that affect data placement
- Divide the states into a small number of bins to reduce the state space

Feature	Description	# of bins	Encoding (bits)
$size_t$	Size of the requested page (in pages)	8	8
$type_t$	Type of the current request (read/write)	2	4
$intr_t$	Access interval of the requested page	64	8
cnt_t	Access count of the requested page	64	8
cap_t	Remaining capacity in the fast storage device	8	8
$curr_t$	Current placement of the requested page (fast/slow)	2	4

Reward

- For every action at time-step t , Sibyl gets a reward from the environment at time-step $t + 1$
- **Reward** acts as a **feedback** to the agent's past action
- **Request latency** faithfully captures the status of the hybrid storage system
- **Penalty** value is chosen to prevent the agent from aggressively servicing all the requests from the faster device

$$R = \begin{cases} \frac{1}{L_t} & \text{if no eviction} \\ \max(0, \frac{1}{L_t} - R_p) & \text{if an eviction happens} \end{cases}$$

L_t = latency of the request
 R_p = eviction penalty

Outline

Background

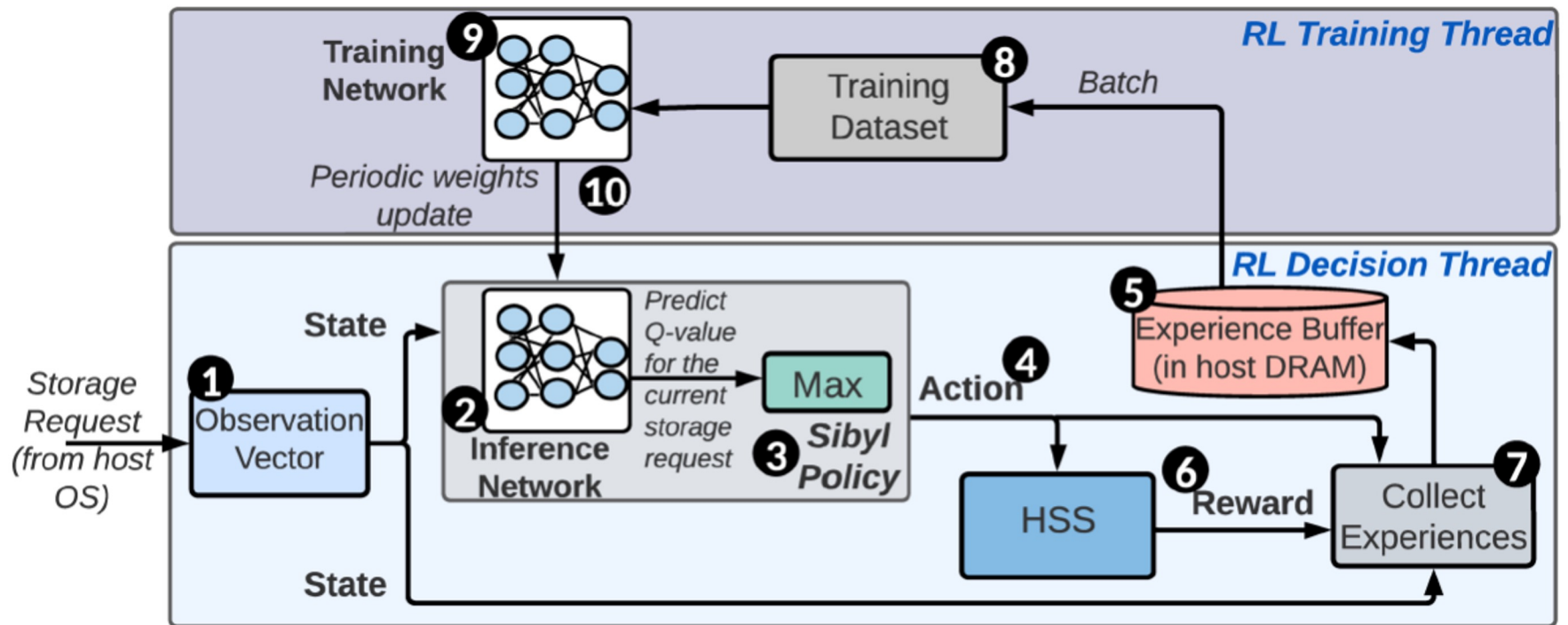
Formulating Data Placement as RL problem

Sibyl: Overview

Evaluation and Key Results

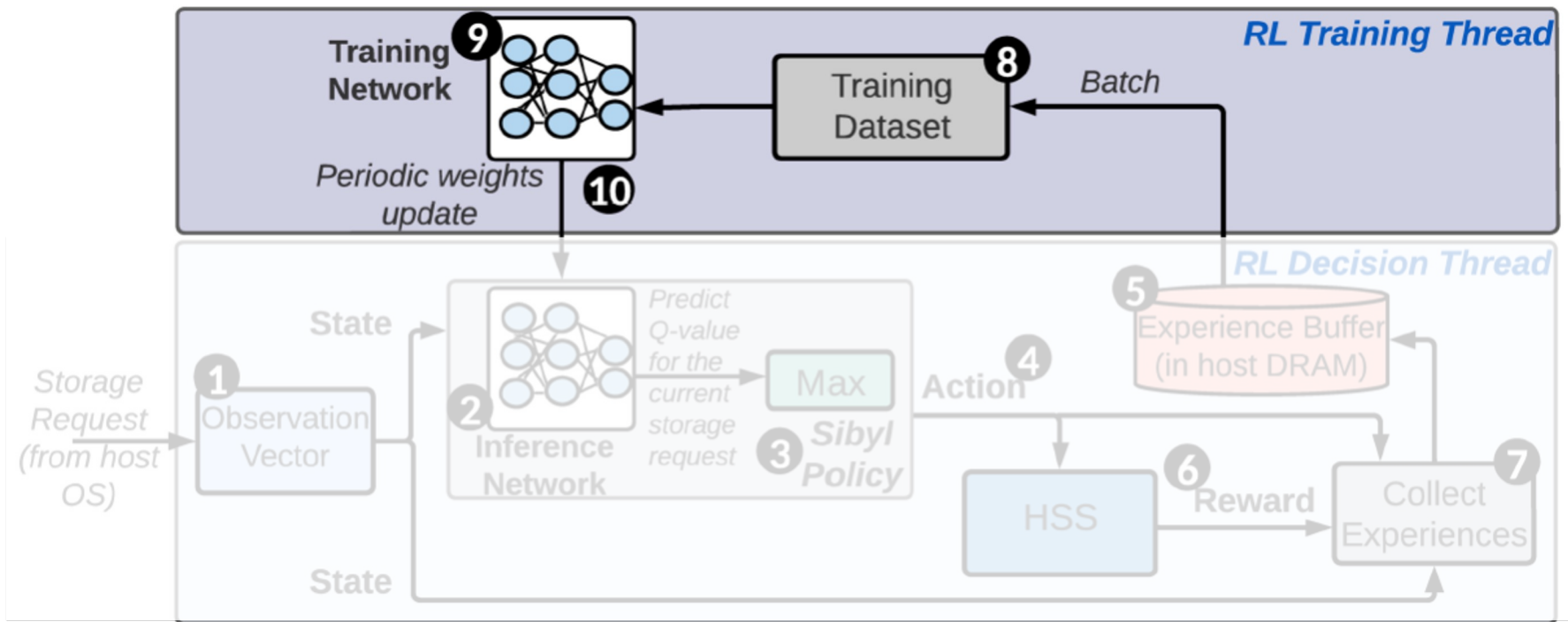
Conclusion

Overview of Sibyl



The two threads run asynchronously to prevent training delay from affecting the inference time

Overview of Sibyl



In the RL training thread, Sibyl uses explored experiences to autonomously update its decision-making policy

Hyper-parameter Tuning

- Different hyper-parameter configurations were chosen using the design of experiments (DoE) technique

Hyper-parameter	Design Space	Chosen Value
Discount factor (γ)	0-1	0.9
Learning rate (α)	$1e^{-5} - 1e^0$	$1e^{-4}$
Exploration rate (ϵ)	0-1	0.001
Batch size	64-256	128
Experience buffer size (e_{EB})	10-10000	1000

Outline

Background

Formulating Data Placement as RL problem

Sibyl: Overview

Evaluation and Key Results

Conclusion

Evaluation Methodology

- Evaluated on a **real system** with different hybrid storage configurations
- Hybrid storage system constitutes one contiguous logical block address space
- A custom block driver was implemented to manage the I/O requests to the storage devices
- We evaluate **three** different hybrid storage configurations
 - Performance-optimized (H&M)
 - Cost-optimized (H&L)
 - Tri-hybrid storage system

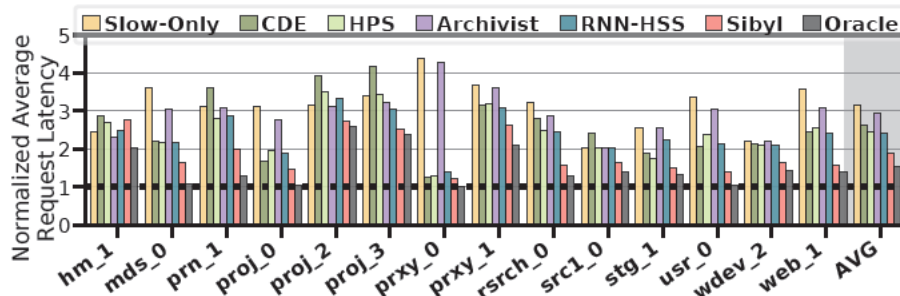
Evaluation Methodology

Host System	AMD Ryzen 7 2700G [146], 8-cores@3.5 GHz, 8×64/32 KiB L1-I/D, 4 MiB L2, 8 MiB L3, 16 GiB RDIMM DDR4 2666 MHz	
Storage Devices	Characteristics	
H: Intel Optane SSD P4800X [94]	375 GB, PCIe 3.0 NVMe, SLC, R/W: 2.4/2 GB/s, random R/W: 550000/500000 IOPS	
M: Intel SSD D3-S4510 [96]	1.92 TB, SATA TLC (3D), R/W: 550/510 MB/s, random R/W: 895000/21000 IOPS	
L: Seagate HDD ST1000DM010 [98]	1 TB, SATA 6Gb/s 7200 RPM Max. Sustained Transfer Rate: 210 MB/s	
L _{SSD} : ADATA SU630 SSD [99]	960 GB, SATA 6 Gb/s, TLC, Max R/W: 520/450 MB/s	
HSS Configurations	Fast Device	Slow Device
H&M (Performance-oriented)	high-end (H)	middle-end (M)
H&L (Cost-oriented)	high-end (H)	low-end (L)

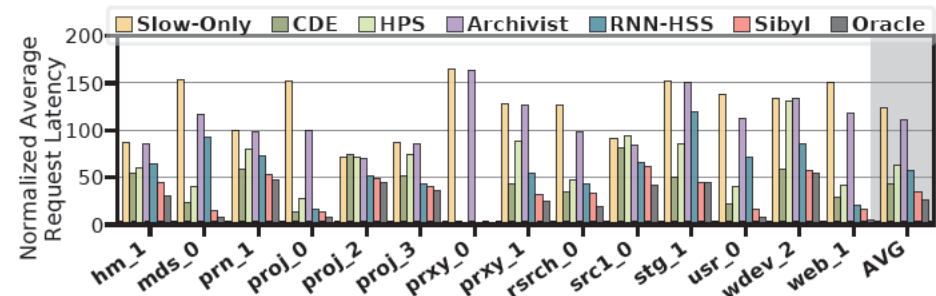
Evaluation Methodology

- 18 different workloads from MSR Cambridge and FileBench suites
- Sibyl is compared against **four baselines**
 - Heuristic-based policies
 - Cold data eviction (CDE) [Matsui et. al., "Design of Hybrid SSDs With Storage Class Memory and NAND Flash Memory," IEEE 2017]
 - History Page Scheduler (HPS) [Meswani et.al., "Heterogeneous Memory Architectures: A HW/SW Approach for Mixing Die-stacked and Off-package Memories," HPCA, 2015]
 - Supervised learning-based policies
 - Recurrent neural network (RNN)-based technique adapted from Kleio [Doudali et.al., "Kleio: A Hybrid Memory Page Scheduler with Machine Intelligence," HPDC, 2019]
 - Neural network-based classifier based on Archivist [Ren et.al., "Archivist: A Machine Learning Assisted Data Placement Mechanism for Hybrid Storage Systems," ICCD, 2019]

Latency for HSS Configurations



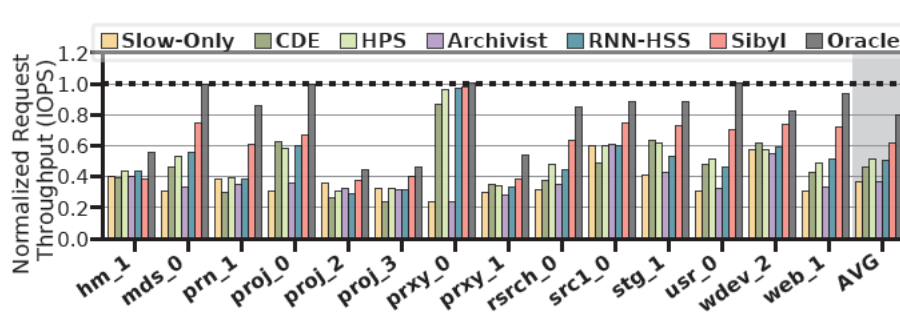
(a) Performance-optimized
(H&M)



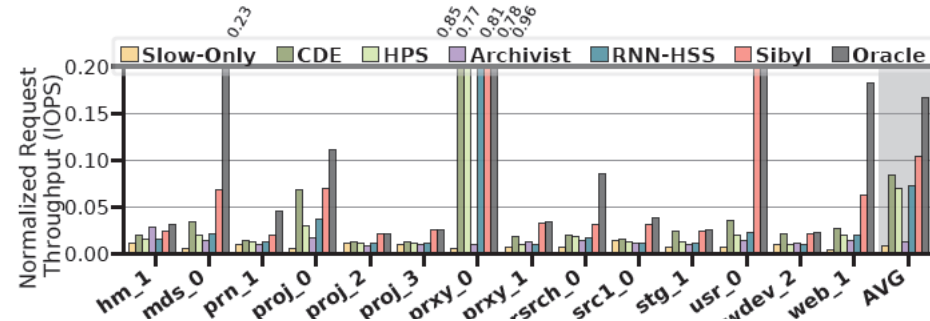
(b) Cost-optimized
(H&L)

Configuration	CDE	HPS	Archivist	RNN-HSS
H&M	28.1%	23.2%	36.1%	21.6%
H&L	19.9%	45.9%	68.8%	34.1%

Throughput for HSS Configurations



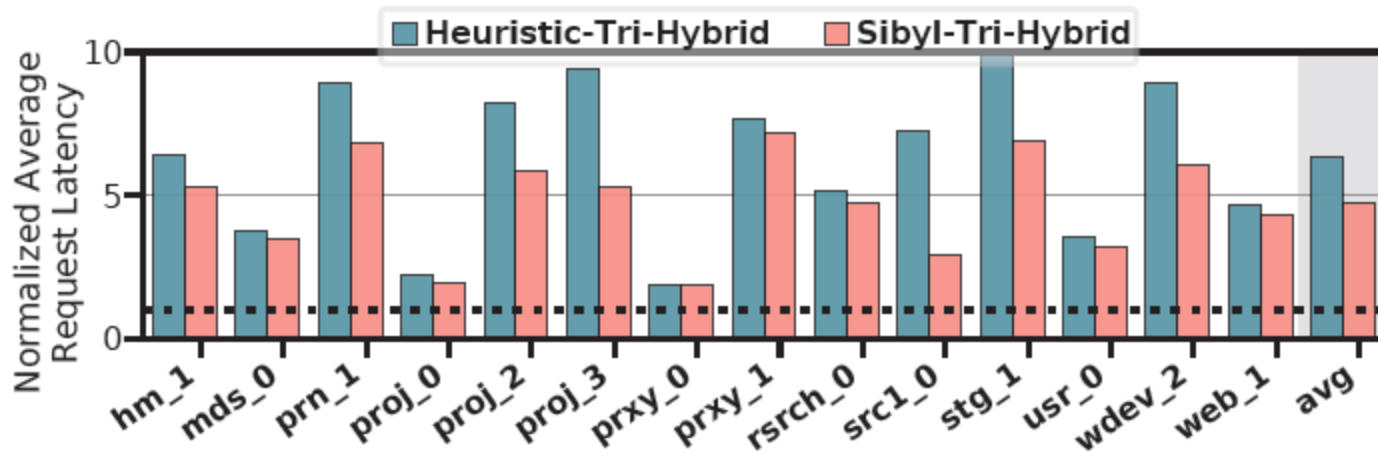
(a) Performance-optimized
(H&M)



(b) Cost-optimized
(H&L)

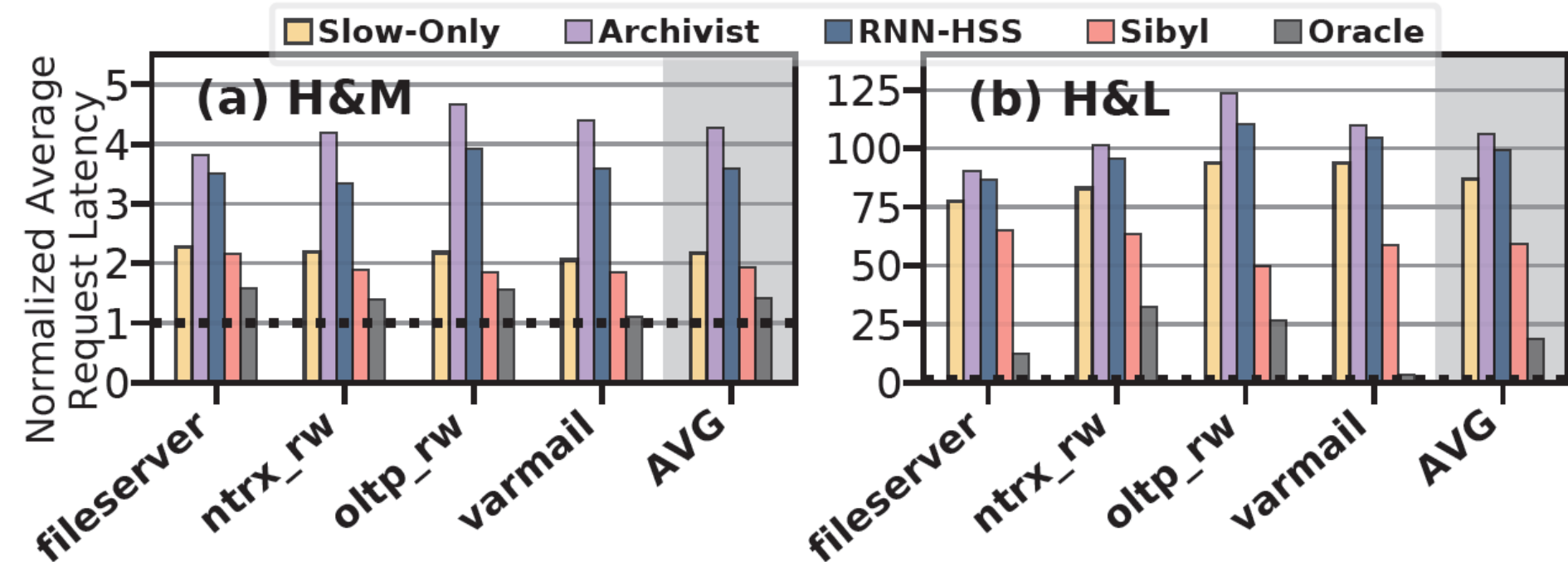
Configuration	CDE	HPS	Archivist	RNN-HSS
H&M	32.6%	21.9%	54.2%	22.7%
H&L	22.8%	49.1%	86.9%	41.9%

Latency in Tri-Hybrid System



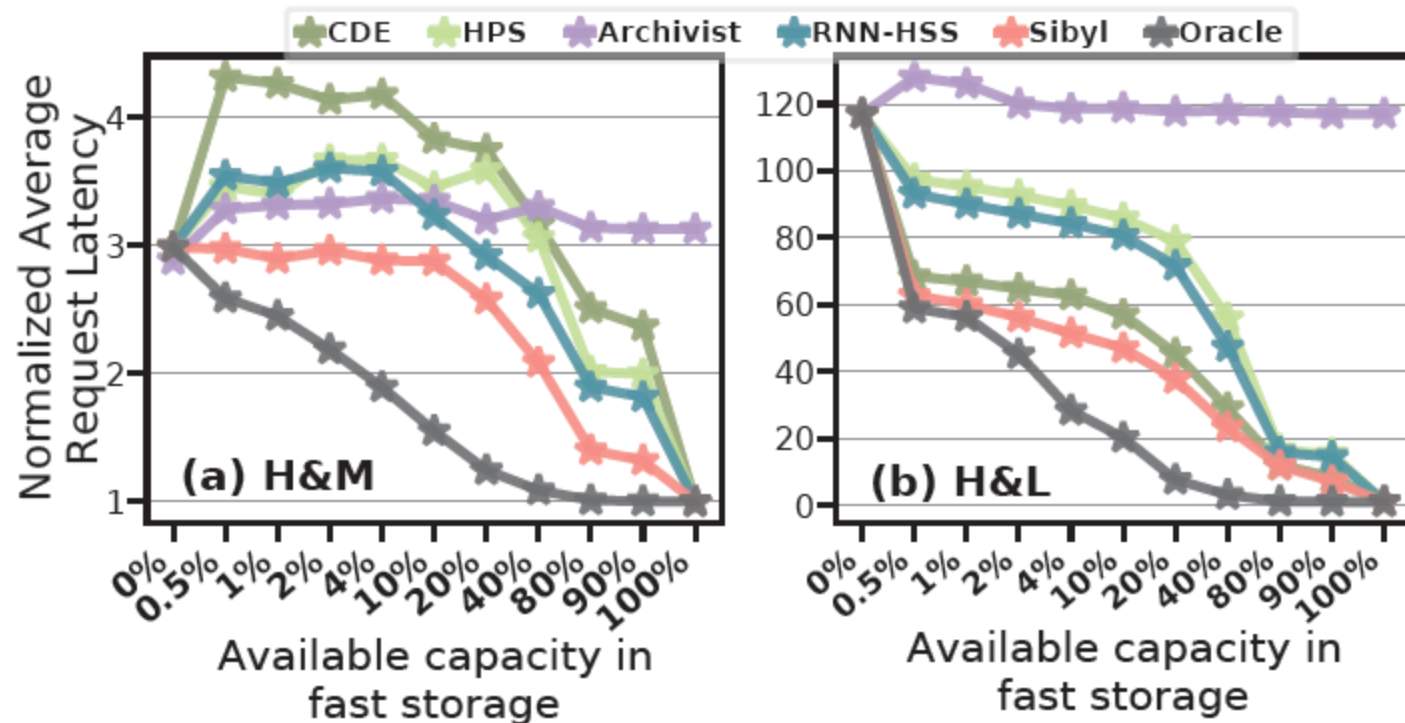
Sibyl outperforms the heuristic-based data placement policy for tri-hybrid system by 48.2% on average across all workloads

Latency for Unseen Workloads

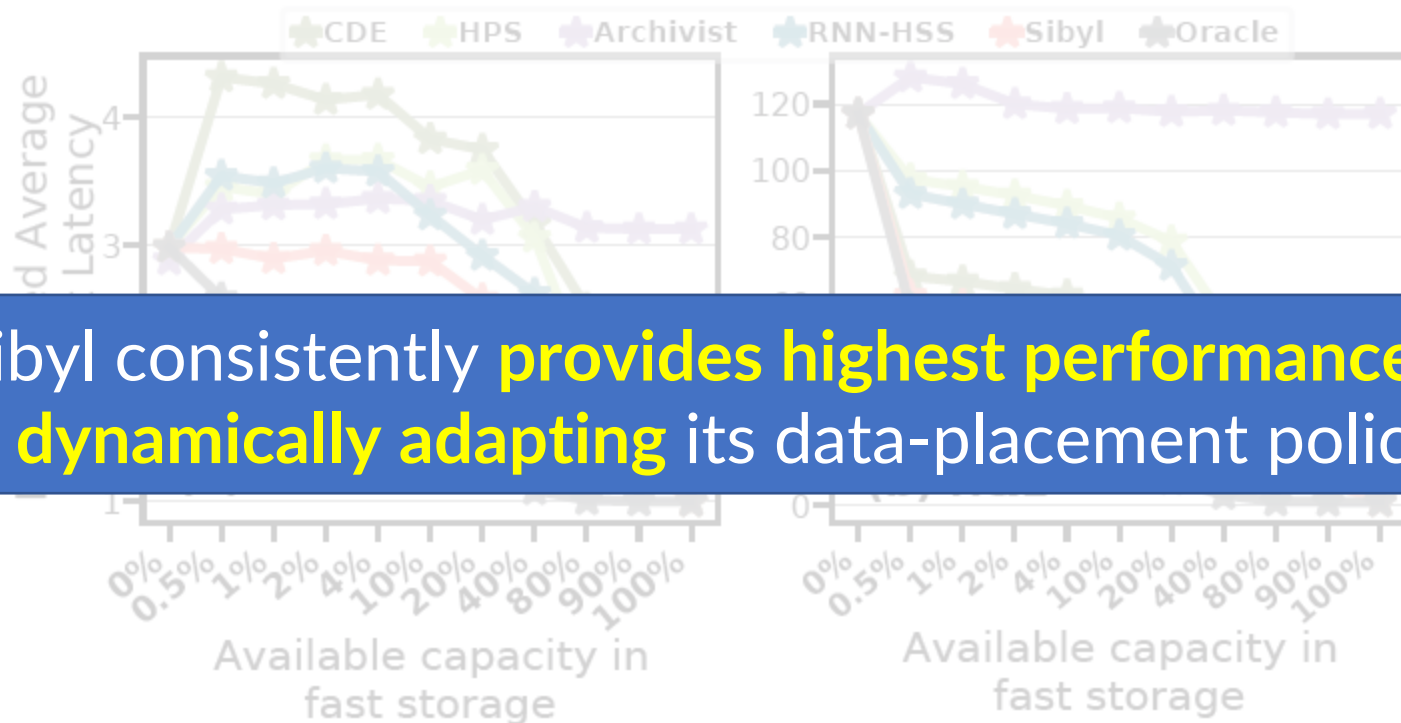


In H&M (H&L) configurations, Sibyl outperforms RNN-HSS and Archivist by 46.1% (54.6%) and 8.5% (44.1%) respectively

Sensitivity to Fast Storage Capacity



Sensitivity to Fast Storage Capacity



Sibyl consistently **provides highest performance** by **dynamically adapting** its data-placement policy

Overhead Analysis

- **Performance Overhead**
 - **~10ns** for every inference on the evaluated system; this is several orders of magnitude less than I/O latency of high-end SSD
- **Implementation Overhead**
 - **124.4 KiB** of implementation overhead
- **Metadata overhead**
 - 0.1% of the total storage capacity when using a 4 KiB data placement granularity
 - **40-bit** metadata overhead per data placement unit

For More on Sybil

- To appear in ISCA 2022

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

<https://arxiv.org/pdf/2205.07394.pdf>

Cross-layer Hardware/Software Techniques to Enable Powerful Computation and Memory Optimizations

Backup Slides

MetaSys

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

Nandita Vijaykumar

Ataberk Olgun, Konstantinos Kanellopoulos, F. Nisa Bostanci

Hasan Hassan, Mehrshad Lotfi, Phillip B. Gibbons, Onur Mutlu

SAFARI



UNIVERSITY OF
TORONTO

ETH *Zürich*



TOBB ETÜ
University of Economics & Technology

Executive Summary

Problem

- Cross-layer techniques are **challenging** to **implement** because they require **full-stack changes**
- **Existing open-source infrastructures** for implementing cross-layer techniques are **not** designed to provide **key features**

Key Idea – Provide:

- Rich dynamic HW/SW interfaces
- Low-overhead metadata management
- Interfaces to key hardware components (e.g., prefetcher)

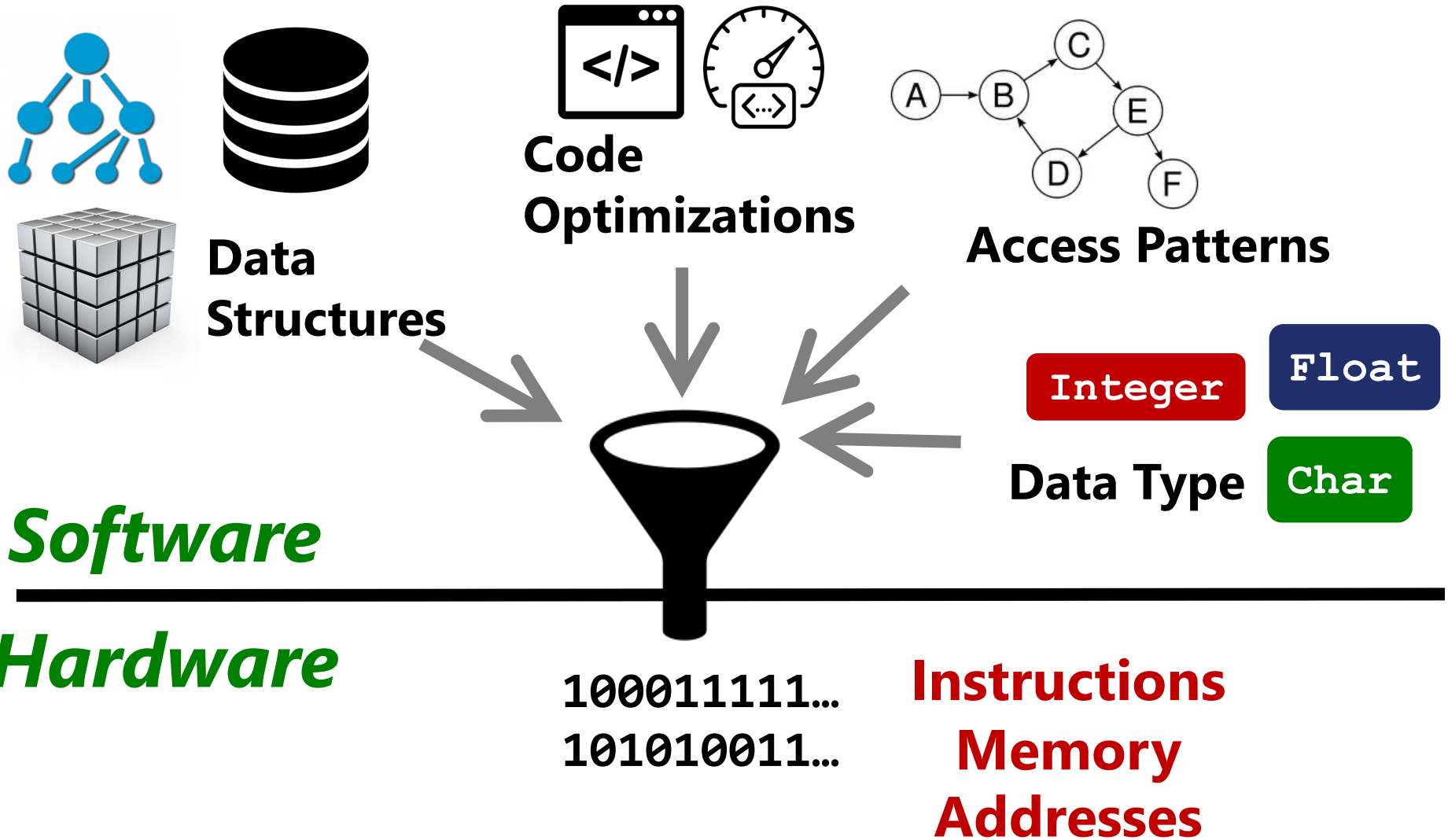
Our **goal** is twofold:

1. Develop an **efficient** and **flexible** framework to enable **rapid implementation** of new cross-layer techniques
2. Perform a detailed limit study to quantify the overheads associated with **general** metadata systems

Outline

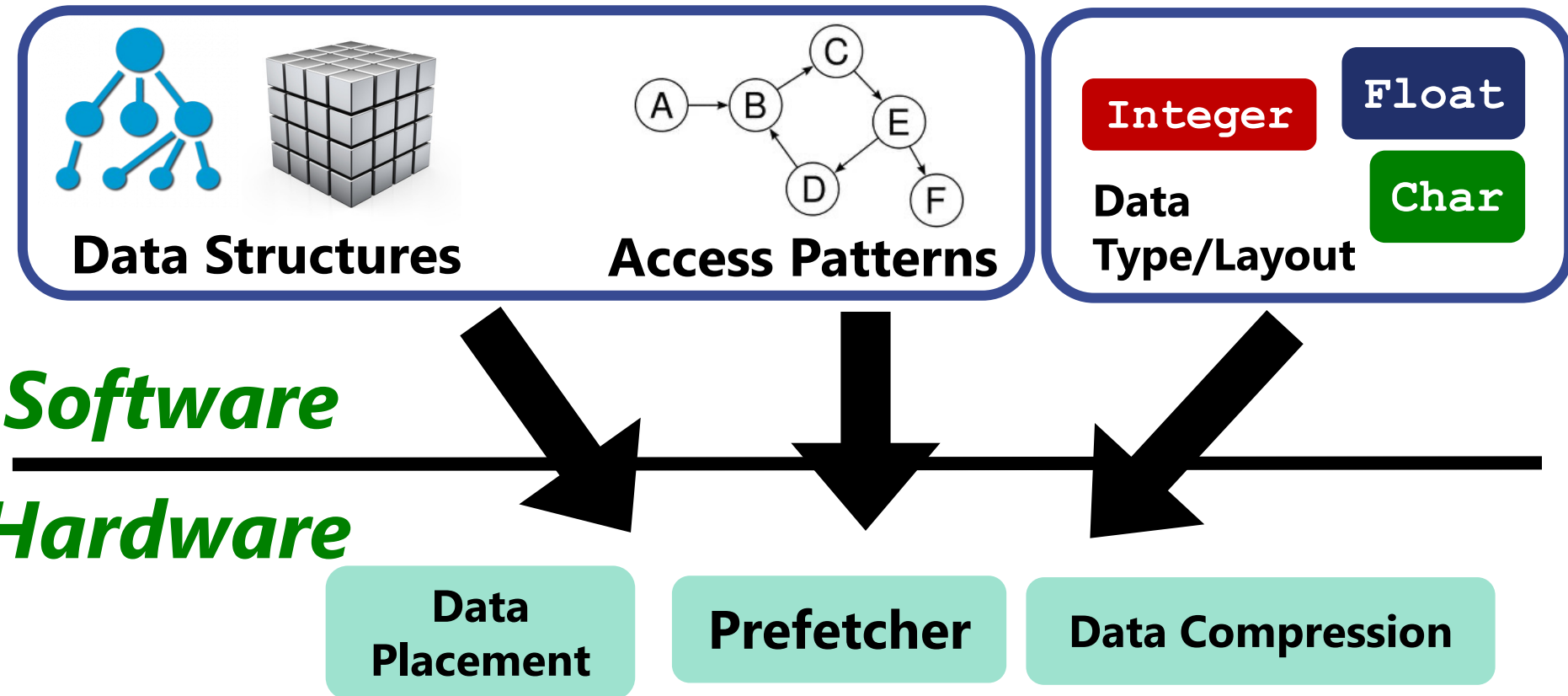
- **Background on Expressive Memory**
- MetaSys
 - Software Interface
 - Key Structures
- FPGA Implementation
- Evaluation

Higher-level information is not visible to HW



With a richer abstraction:

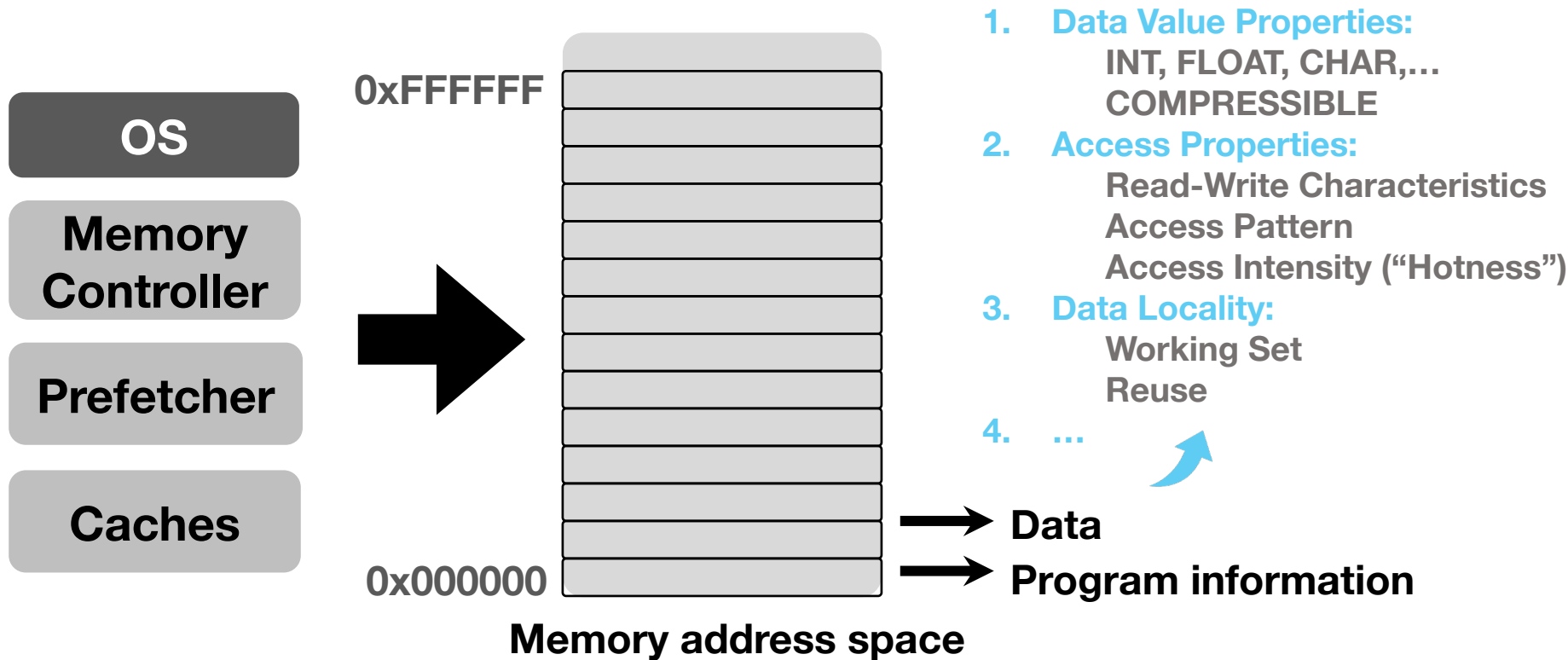
SW can provide program information
can significantly help hardware



Outline

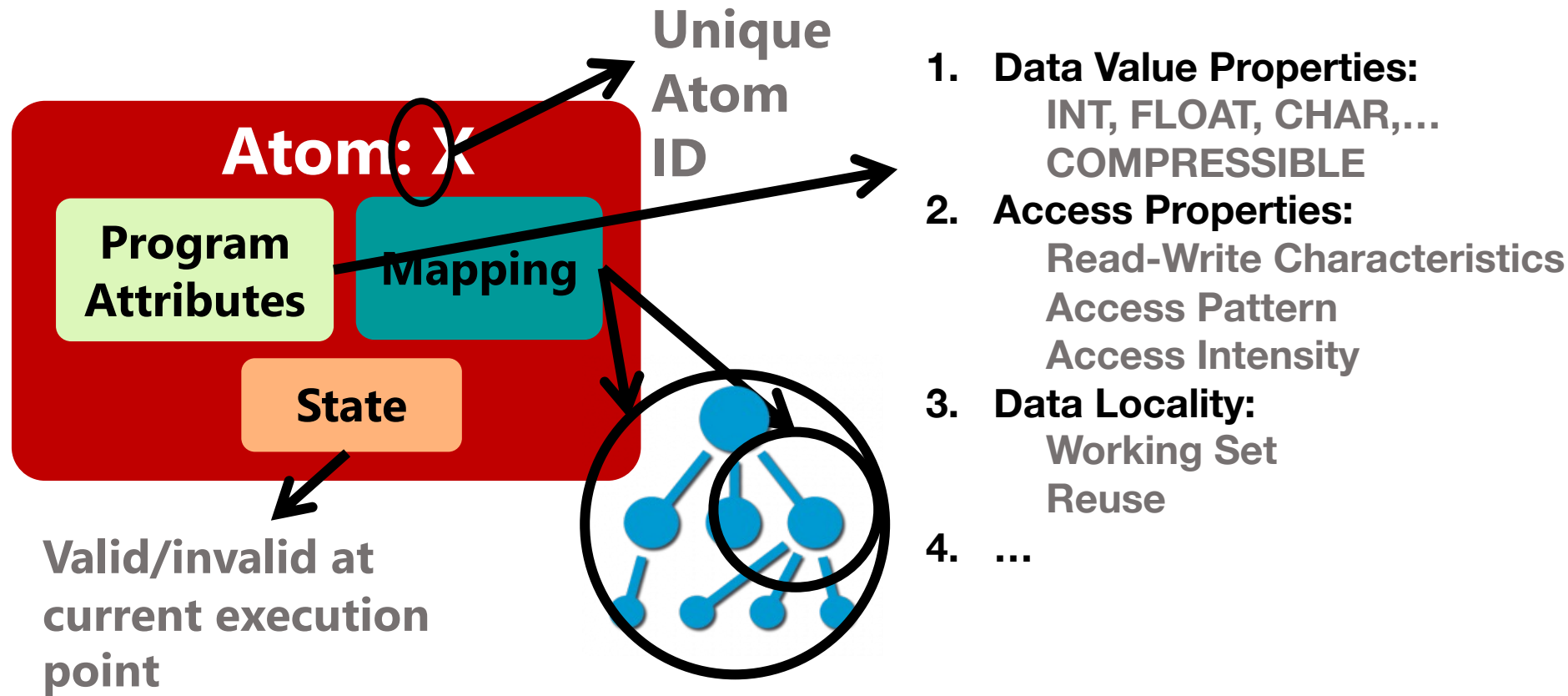
- Background on Expressive Memory
- **MetaSys**
 - **Software Interface**
 - **Key Structures**
- FPGA Implementation
- Evaluation

Metadata: Data Semantics



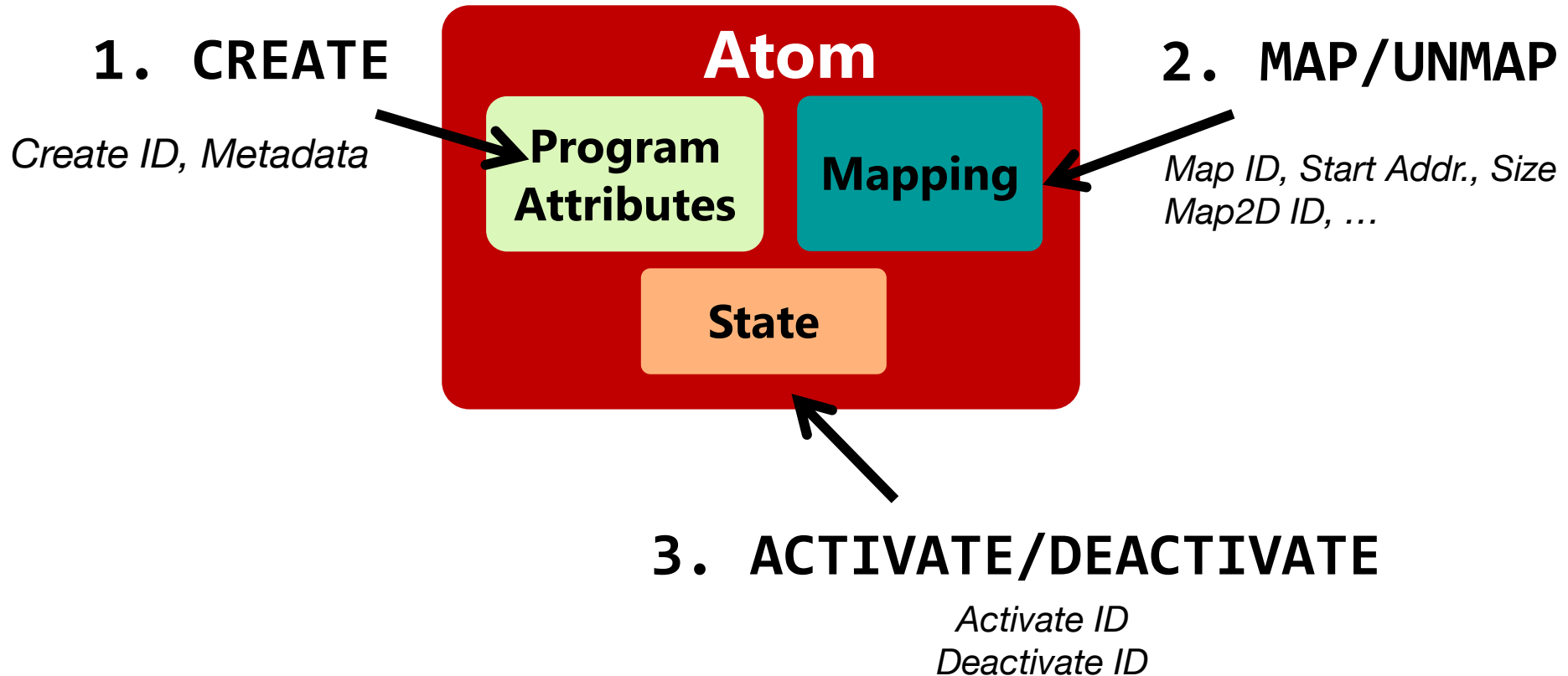
The ATOM

An abstraction to express data semantics

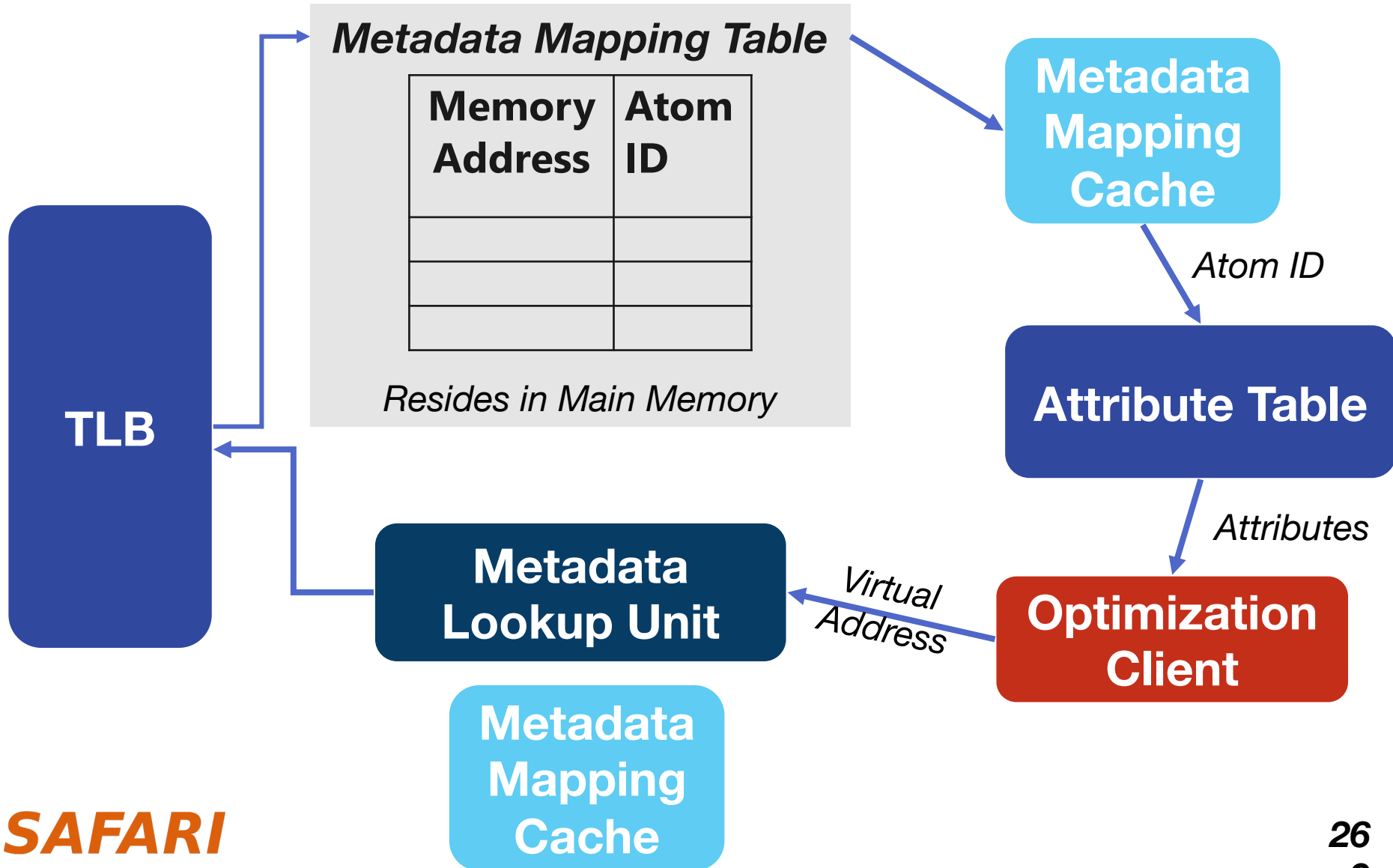


The Software Interface

Three Atom operators



MetaSys Key Structures



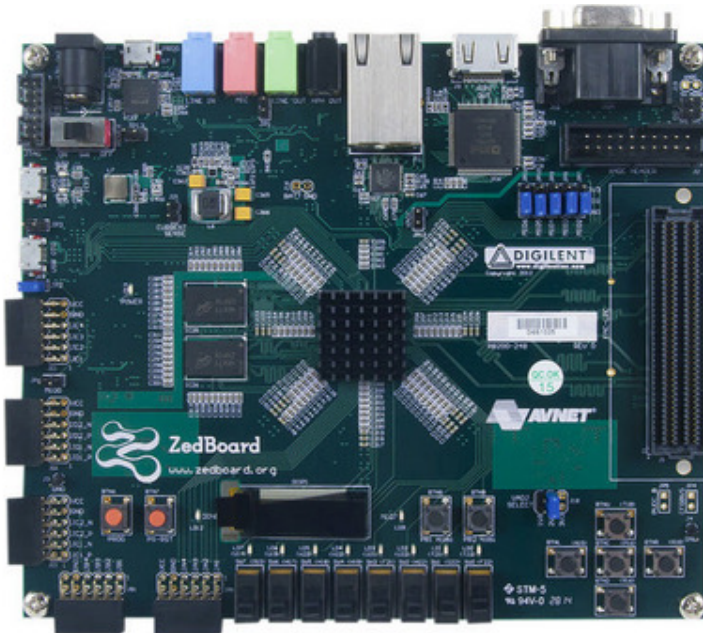
Outline

- Background on Expressive Memory
- MetaSys
 - Software Interface
 - Key Structures
- **FPGA Implementation**
- Evaluation

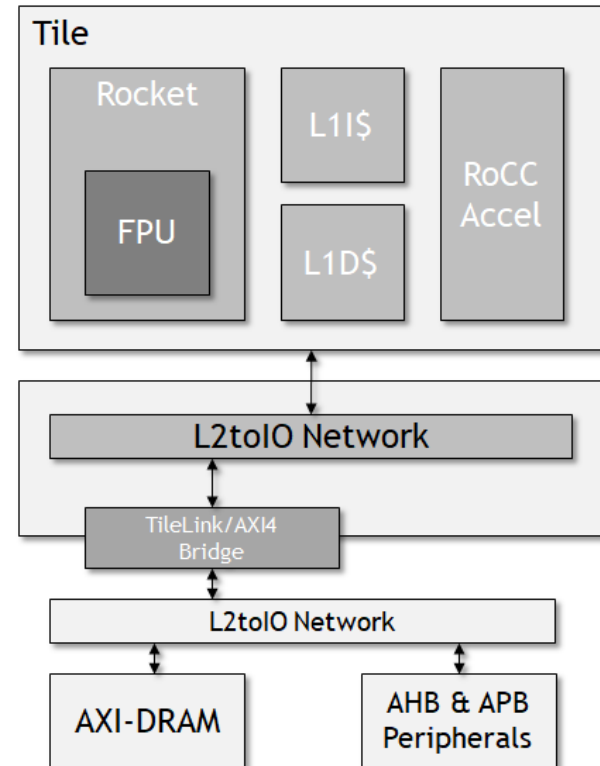
FPGA Prototype

Prototype on Xilinx Zedboard
within a **real RISC-V system** (Rocket Chip)

Zedboard



Rocket Chip



MetaSys in Rocket Chip

Implement two **main** components:

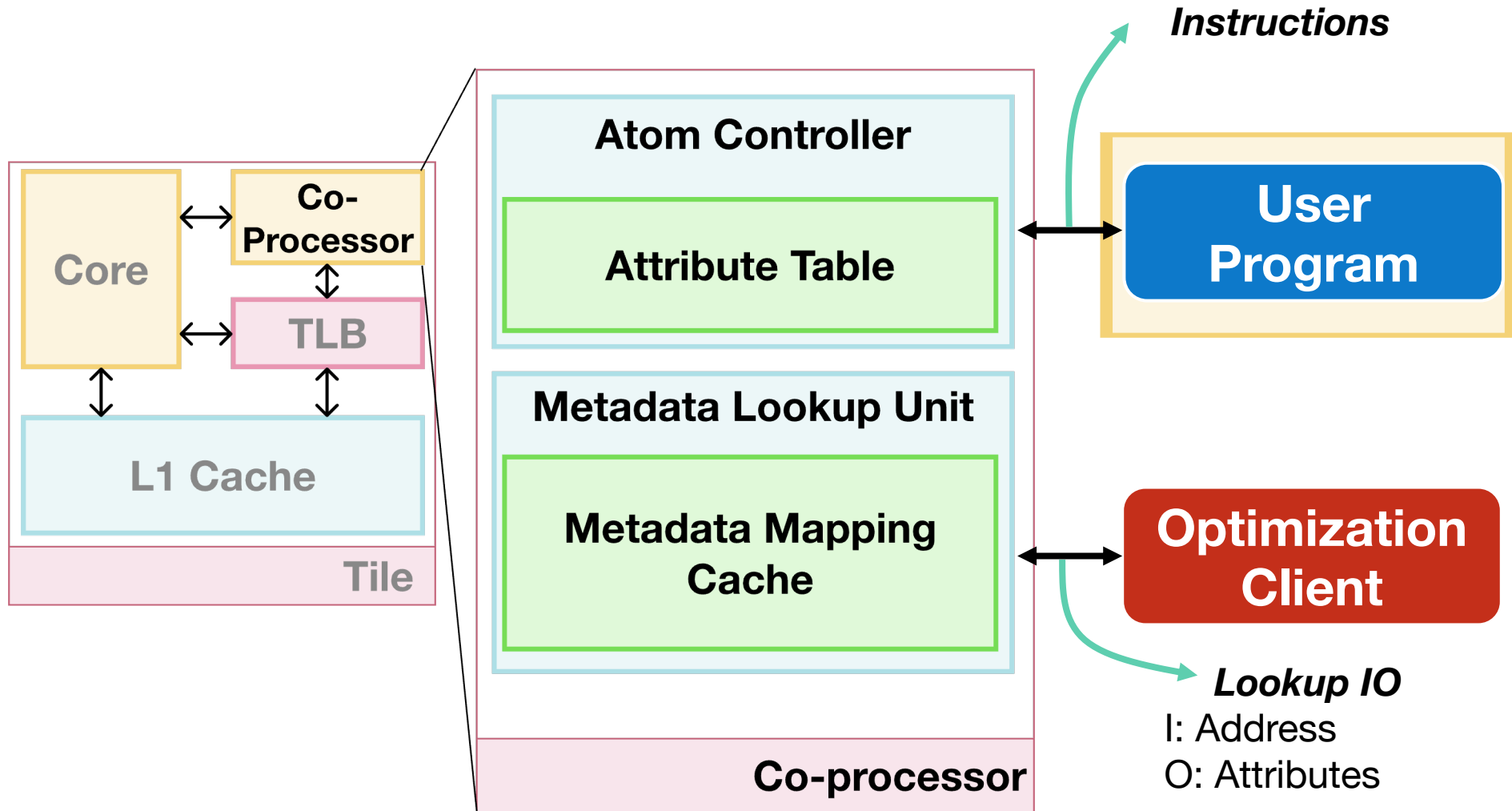
1. Atom Controller

- Manages the **attribute table** (CREATE – (DE)ACTIVATE)
- Performs atom mapping (MAP/UNMAP)
 - Physical address → Atom ID

2. Metadata Lookup Unit


- Responds to clients:
 - Provides atom attributes
- Contains the metadata mapping cache




Changes in Rocket Chip






Source on Github

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


 **CMU-SAFARI / MetaSys** Public

 Notifications  Fork 2  Star 0


[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)

 main  1 branch  0 tags

[Go to file](#) [Code](#)

 **olgunataberk** Update README.md e21ccd2 on Jul 9, 2021 🕒 12 commits

common	Initial commit	9 months ago
riscv-tools	Add tools directory	7 months ago
rocket-chip	Initial commit	9 months ago
testchipip	Initial commit	9 months ago
zedboard	Initial commit	9 months ago
LICENSE	Initial commit	9 months ago
README.md	Update README.md	7 months ago
metasys_readme.md	Update metasys_readme.md	7 months ago

 README.md



MetaSys




We refer the developers of the MetaSys repository to [metasys_readme.md](#), where we describe our modifications to the existing rocket-chip code base, and present a walkthrough of an implementation of the prefetching use case described in our paper.

For more details, please read our [preprint on arXiv](#).

About

Metasys is the first open-source FPGA-based infrastructure with a prototype in a RISC-V core, to enable the rapid implementation and evaluation of a wide range of cross-layer software/hardware cooperative techniques techniques in real hardware. Described in our preprint: <https://arxiv.org/abs/2105.08123>

 Readme  View license

 0 stars  3 watching  2 forks

Releases

No releases published

Packages

No packages published

Outline

- Background on Expressive Memory
- MetaSys
 - Software Interface
 - Key Structures
- FPGA Implementation
- **Evaluation**

Characterizing Metadata Management

Our **goal** is twofold:

1. ...
2. Perform a detailed limit study to quantify the overheads associated with **general** metadata systems

Quantify the overheads of performing lookups in MetaSys

Evaluation Methodology

Run workloads on MetaSys prototype (Zedboard):

Microbenchmarks: Represent a variety of memory access patterns

Polybench: Scientific computation kernels

Ligra: Graph workloads

CPU: 25 MHz; in-order Rocket core [21]; **TLB** 16 entries DTLB; LRU policy;

L1 Data + Inst. Cache: 16 KB, 4-way; 4-cycle; 64 B line; LRU policy; MSHR size: 2

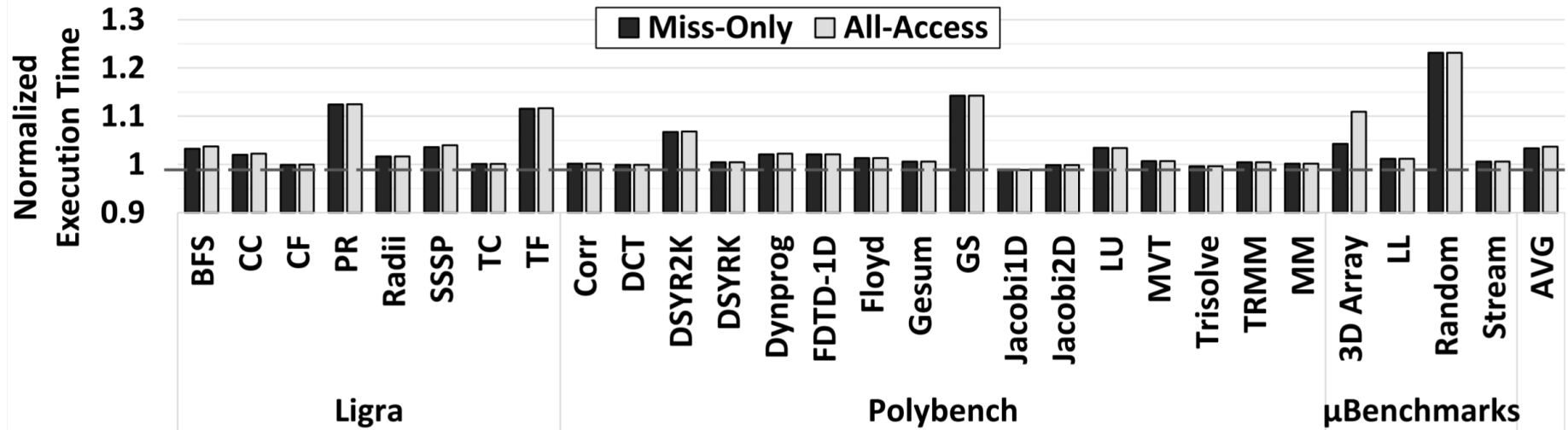
MMC: NMRU Policy; 128 entries; 38bits/entry; **Tagging Granularity:** 512B;

Private Metadata Table: 256 entries; 64B/entry; **DRAM:** 533MHz; V_{dd} : 1.5V;

Workloads: Ligra [36]: PageRank(PR), Shortest Path (SSSP), Collaborative Filtering (CF) Teenage Follower (TF), Triangle Counting (TC), Breadth-First Search (BFS) Radius Estimation (Radii), Connected Components (CC);

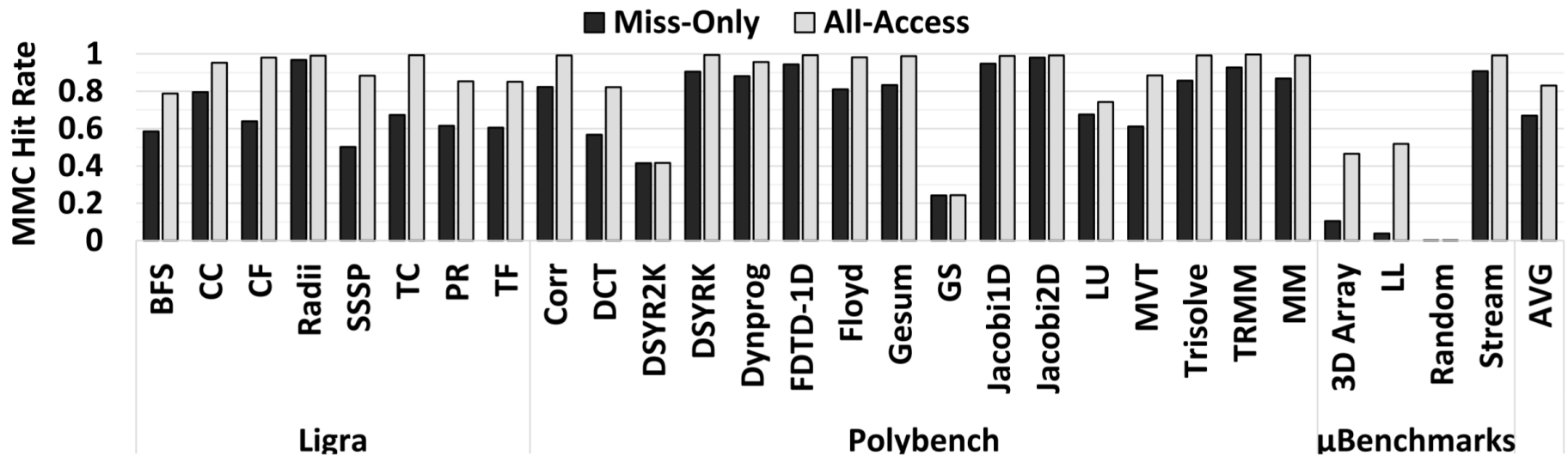
Polybench [37]; μ Benchmarks

Performance Overhead



Metadata lookups occur low performance overheads
2.7% on average

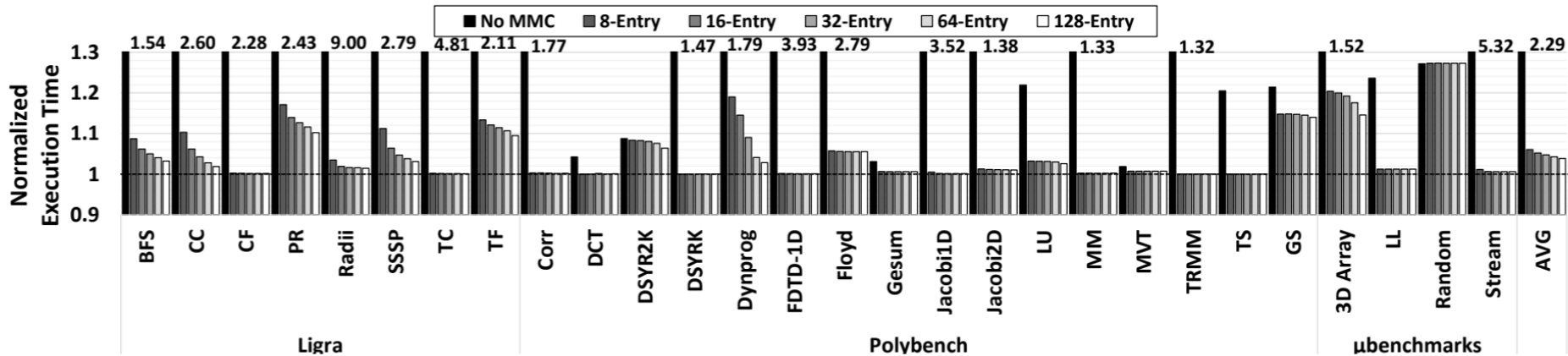
MMC hit rate



MMC can cover ~81% of all memory requests on average

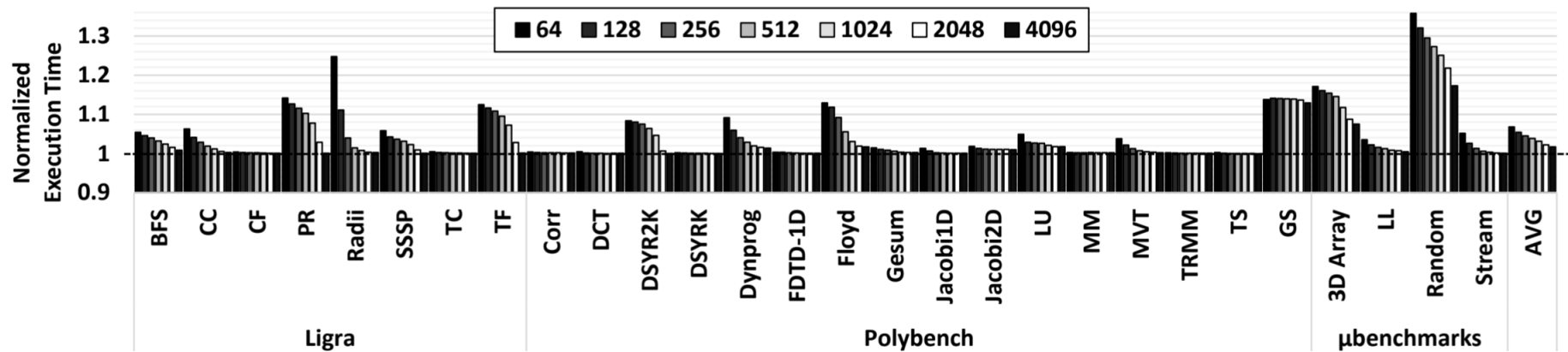
MMC hit rate correlates with locality of application requests

Impact of MMC size



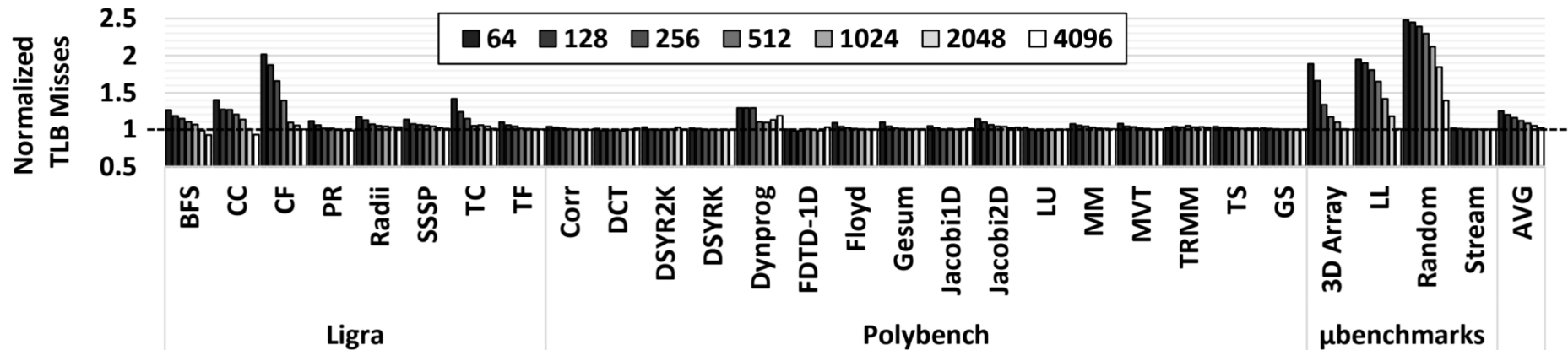
Workloads with low temporal and spatial locality
are not sensitive to MMC size

Impact of Tagging Granularity



Performance impact increases with finer granularity

Impact of Tagging Granularity on TLB misses

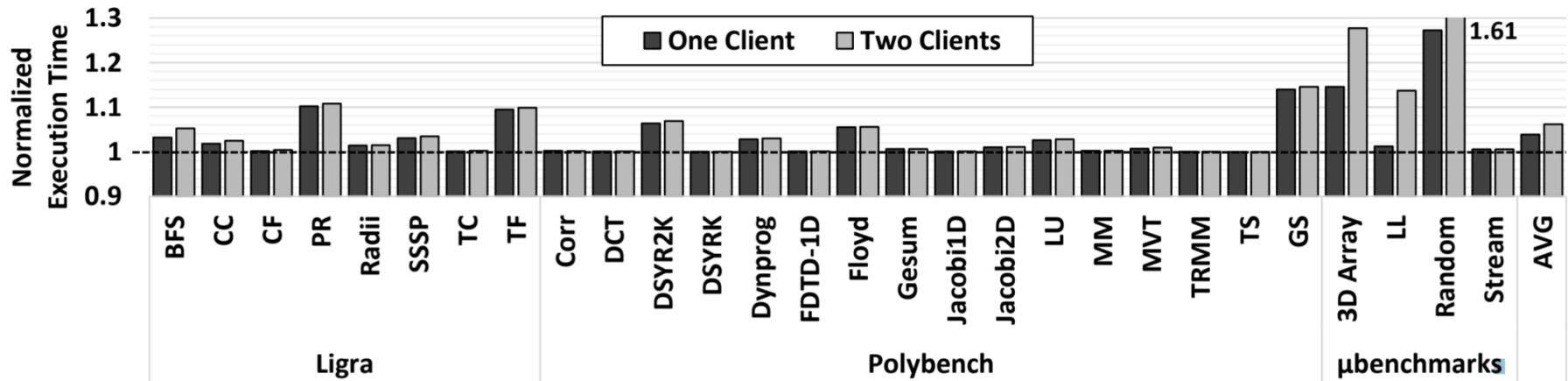


Fine tagging granularities increase TLB misses

Effect of Contention

One Client: All memory requests originating from rocket core

Two Clients: One client + all memory requests originating from the page table walker



Multiple clients do not significantly affect performance
(0.3% overhead on average)

Executive Summary

Problem

- Cross-layer techniques are **challenging** to **implement** because they require **full-stack changes**
- **Existing open-source infrastructures** for implementing cross-layer techniques are **not** designed to provide **key features**:

Key Idea – Provide:

- Rich dynamic HW/SW interfaces
- Low-overhead metadata management
- Interfaces to key hardware components (e.g., prefetcher)

Our **goal** is twofold:

1. Develop an **efficient** and **flexible** framework to enable **rapid implementation** of new cross-layer techniques
2. Perform a detailed limit study to quantify the overheads associated with **general** metadata systems

MetaSys

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

Nandita Vijaykumar

Ataberk Olgun, Konstantinos Kanellopoulos, F. Nisa Bostanci

Hasan Hassan, Mehrshad Lotfi, Phillip B. Gibbons, Onur Mutlu

SAFARI



UNIVERSITY OF
TORONTO

ETH *Zürich*



TOBB ETÜ
University of Economics & Technology