Memory System Design for AI/ML Accelerators & ML/AI Techniques for Memory System Design

Onur Mutlu

omutlu@gmail.com

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

30 August 2022

SRC AIHW Annual Review

SAFARI ETH zürich Carnegie Mellon

Confidentiality

- By reviewing this presentation or participating in a SRC event, you are agreeing not to use the presented information for purposes unrelated to the event until approved by SRC;

- Material may be presented that represents current research, some of which has not been published or protected. This material is not for public disclosure and until potential IP rights have been protected, please treat all of the information presented as **confidential information** which is the property of the researcher and their university.

Agenda

- Problem and Background
- Task Overview
- Technical Challenges, Goals and Ideas
- Ideas, Results and Papers from the Past Year

The Problem

Computing

is Bottlenecked by Data

Data is Key for AI, ML, Genomics, …

- Important workloads are all data intensive

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

- Data is increasing
  - We can generate more than we can process

Data is Key for Future Workloads

In-memory Databases

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

Graph/Tree Processing

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

APACHE Spark™

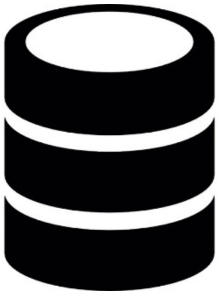
In-Memory Data Analytics

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

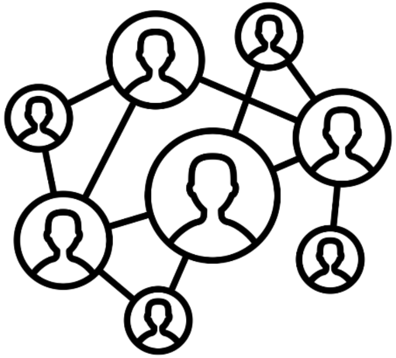
Datacenter Workloads

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

Data Overwhelms Modern Machines



In-memory Databases



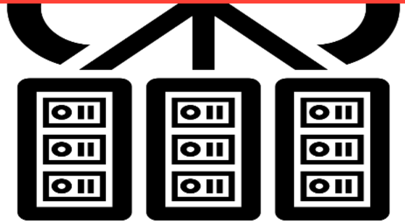
Graph/Tree Processing

Data → performance & energy bottleneck

APACHE Spark™

In-Memory Data Analytics

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



Datacenter Workloads

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

Data is Key for Future Workloads

Chrome

Google's web browser

TensorFlow Mobile

Google's machine learning framework

Video Playback

Google's **video codec**

Video Capture

Google's **video codec**

Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data → performance & energy bottleneck

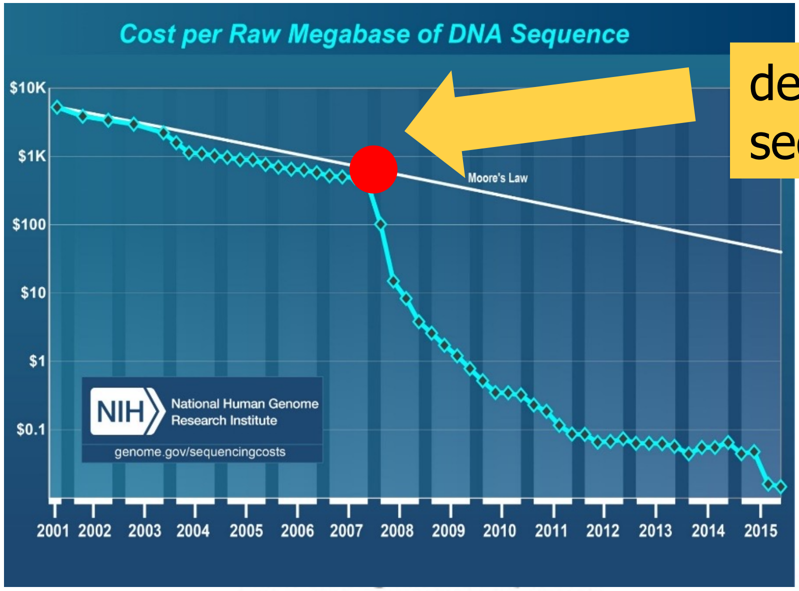
Video Playback

Google's **video codec**

Video Capture

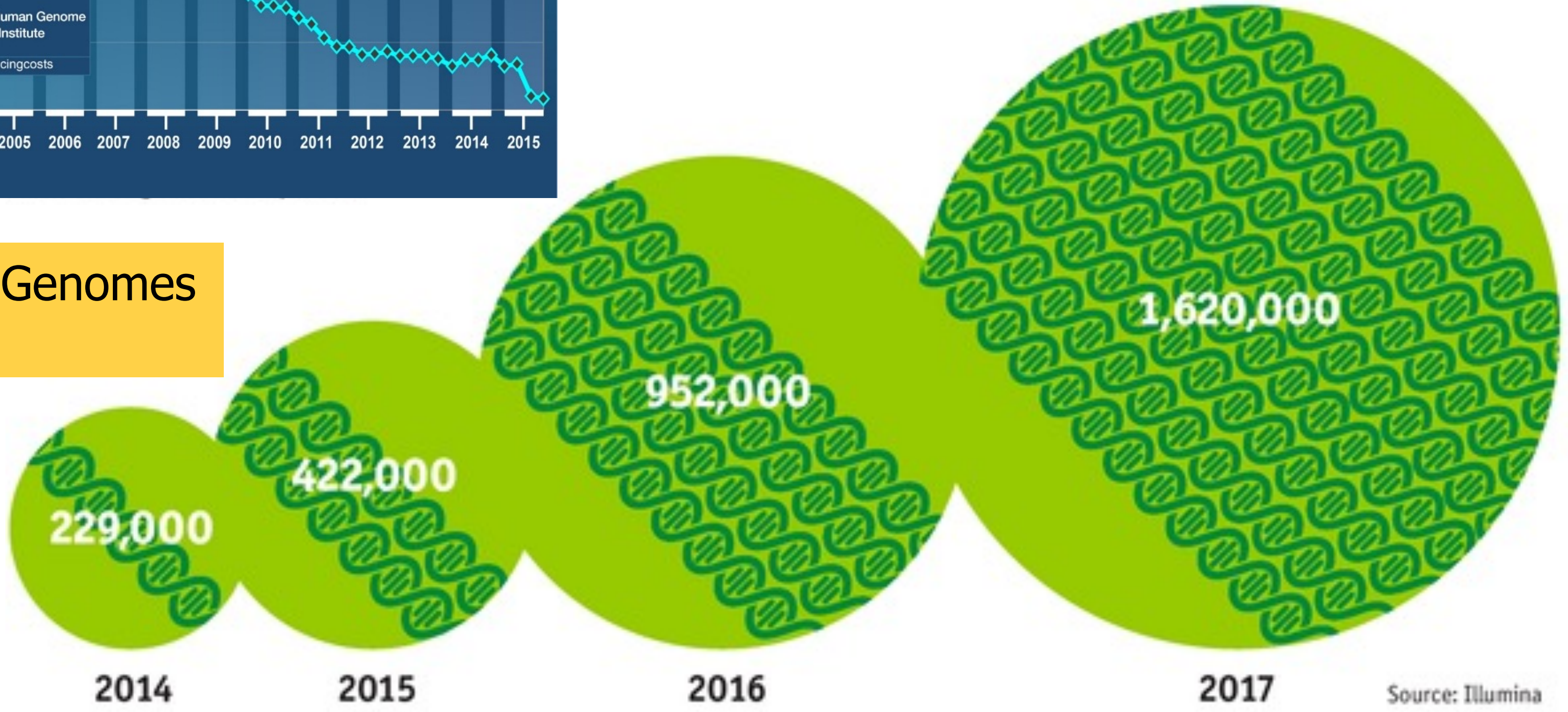
Google's **video codec**

Data is Key for Future Workloads



development of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced



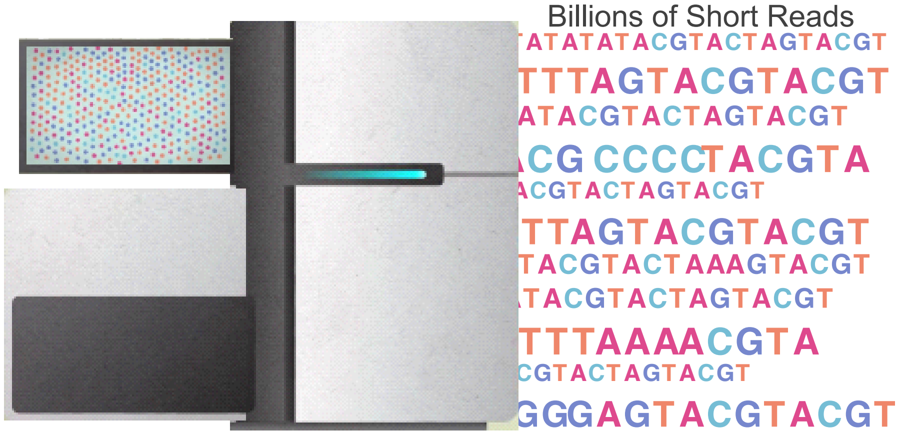
Source: Illumina

The Economist

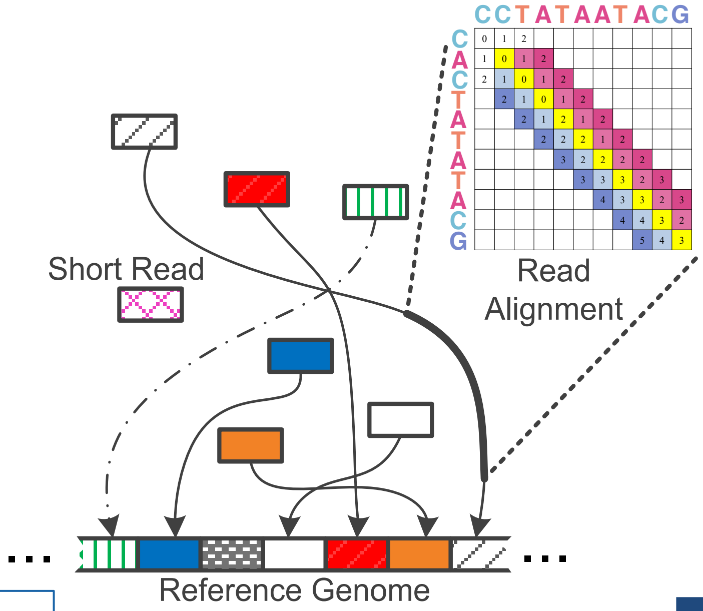
http://www.economist.com/news/21631808-so-much-genetic-data-so-many-uses-genes-unzipped

Genome Analysis

1 Sequencing



2 Read Mapping



Data → performance & energy bottleneck

3 Variant Calling

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

4 Scientific Discovery

Example Data Generator: Genome Sequencing

Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions

Damla Senol Cali ✉, Jeremie S Kim, Saugata Ghose, Can Alkan, Onur Mutlu

Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017

Published: 02 April 2018 **Article history ▾**

Oxford Nanopore MinION

Data → performance & energy bottleneck

Data Overwhelms Modern Machines …

- Storage/memory capability

- Communication capability

- Computation capability

- Greatly impacts robustness, energy, performance, cost

Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data → performance & energy bottleneck

Video Playback

Google's **video codec**

Video Capture

Google's **video codec**

Data Movement Overwhelms Modern Machines

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

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

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

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

Axiom

An Intelligent Architecture

Handles Data Well

How to Handle Data Well

- **Ensure data does not overwhelm** the components
  - via intelligent algorithms
  - via intelligent architectures
  - via whole system designs: algorithm-architecture-devices

- **Take advantage of** vast amounts of **data** and metadata
  - to improve architectural & system-level decisions

- **Understand and exploit** properties of (different) **data**
  - to improve algorithms & architectures in various metrics

Corollaries: Computing Systems Today …

- Are processor-centric vs. **data-centric**

- Make designer-dictated decisions vs. **data-driven**

- Make component-based myopic decisions vs. **data-aware**

Architectures for Intelligent Machines

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

Agenda

- Problem and Background
- Task Overview
- Technical Challenges, Goals and Ideas
- Ideas, Results and Papers from the Past Year

In This Task… (Task #2946.001)

- We focus on designing memory systems to handle data well

- We aim to solve two different yet related and synergistic problems, both focusing on ML/AI and memory system design

- We explore (and exploit the synergy between)
  - Memory system design for AI/ML workloads/accelerators
  - AI/ML techniques for improving memory system designs

- Task Name: Memory System Design for AI/ML Accelerators & ML/AI Techniques for Memory System Design

Our Goals in This Task

- Two Major Goals:

1. Memory system design for AI/ML workloads/accelerators

→ in-depth exploration of memory system designs for cutting-edge and emerging machine learning accelerators

→ more efficient on-chip and off-chip memory systems

2. AI/ML techniques for improving memory system designs

→ take a comprehensive look at memory system design and make it data driven, i.e., based on machine learning

→ more effective cache/memory/prefetch/thread controllers and data/resource management/mapping/scheduling policies

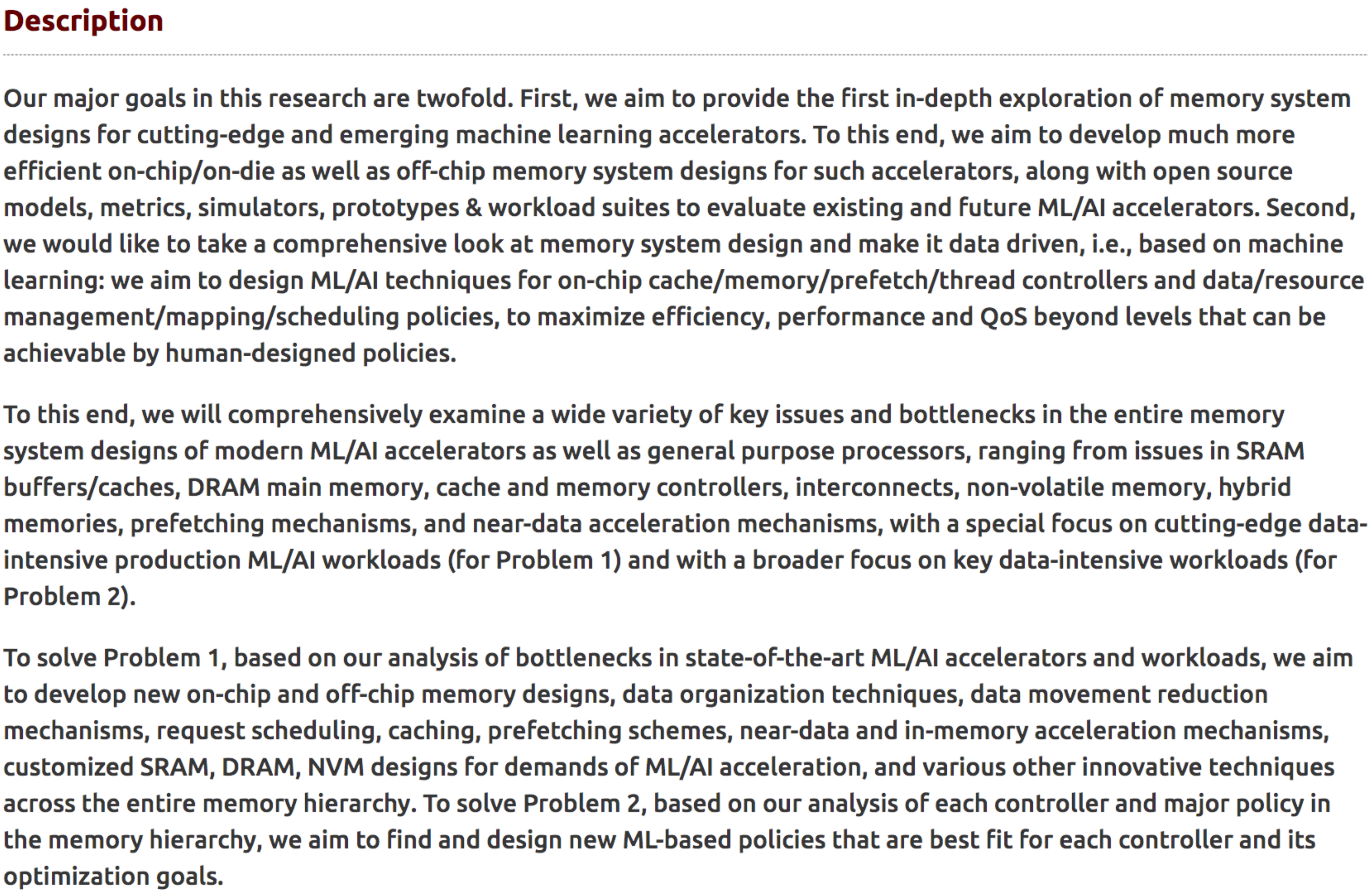
Anticipated Primary Results

- Realistic, practical and effective novel memory system designs for ML/AI accelerators

- New ML-based techniques to improve memory system efficiency and performance

- Open-source workloads, metrics, methodologies & infrastructures to analyze such designs and techniques.

Task Description



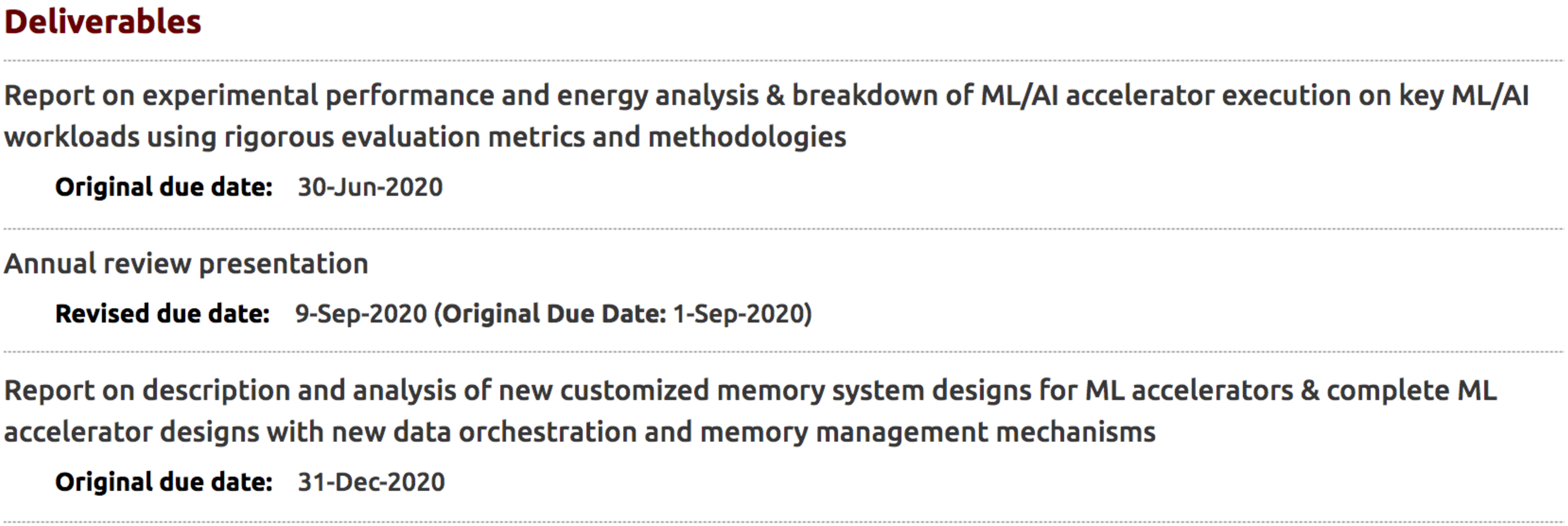
Description

Our major goals in this research are twofold. First, we aim to provide the first in-depth exploration of memory system designs for cutting-edge and emerging machine learning accelerators. To this end, we aim to develop much more efficient on-chip/on-die as well as off-chip memory system designs for such accelerators, along with open source models, metrics, simulators, prototypes & workload suites to evaluate existing and future ML/AI accelerators. Second, we would like to take a comprehensive look at memory system design and make it data driven, i.e., based on machine learning: we aim to design ML/AI techniques for on-chip cache/memory/prefetch/thread controllers and data/resource management/mapping/scheduling policies, to maximize efficiency, performance and QoS beyond levels that can be achievable by human-designed policies.

To this end, we will comprehensively examine a wide variety of key issues and bottlenecks in the entire memory system designs of modern ML/AI accelerators as well as general purpose processors, ranging from issues in SRAM buffers/caches, DRAM main memory, cache and memory controllers, interconnects, non-volatile memory, hybrid memories, prefetching mechanisms, and near-data acceleration mechanisms, with a special focus on cutting-edge data-intensive production ML/AI workloads (for Problem 1) and with a broader focus on key data-intensive workloads (for Problem 2).

To solve Problem 1, based on our analysis of bottlenecks in state-of-the-art ML/AI accelerators and workloads, we aim to develop new on-chip and off-chip memory designs, data organization techniques, data movement reduction mechanisms, request scheduling, caching, prefetching schemes, near-data and in-memory acceleration mechanisms, customized SRAM, DRAM, NVM designs for demands of ML/AI acceleration, and various other innovative techniques across the entire memory hierarchy. To solve Problem 2, based on our analysis of each controller and major policy in the memory hierarchy, we aim to find and design new ML-based policies that are best fit for each controller and its optimization goals.

Task Deliverables (2020)



Deliverables

Report on experimental performance and energy analysis & breakdown of ML/AI accelerator execution on key ML/AI workloads using rigorous evaluation metrics and methodologies

Original due date: 30-Jun-2020

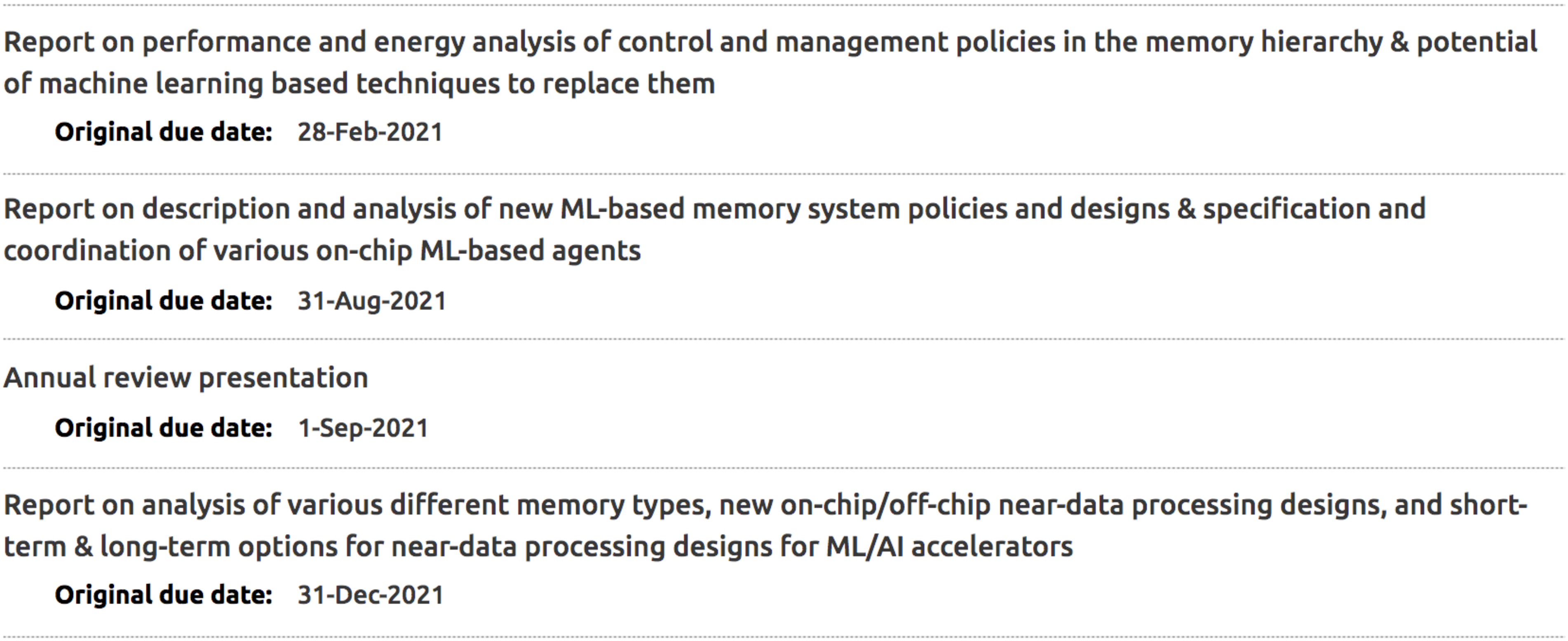
Annual review presentation

Revised due date: 9-Sep-2020 (**Original Due Date:** 1-Sep-2020)

Report on description and analysis of new customized memory system designs for ML accelerators & complete ML accelerator designs with new data orchestration and memory management mechanisms

Original due date: 31-Dec-2020

Task Deliverables (2021)



Report on performance and energy analysis of control and management policies in the memory hierarchy & potential of machine learning based techniques to replace them

Original due date: 28-Feb-2021

Report on description and analysis of new ML-based memory system policies and designs & specification and coordination of various on-chip ML-based agents

Original due date: 31-Aug-2021

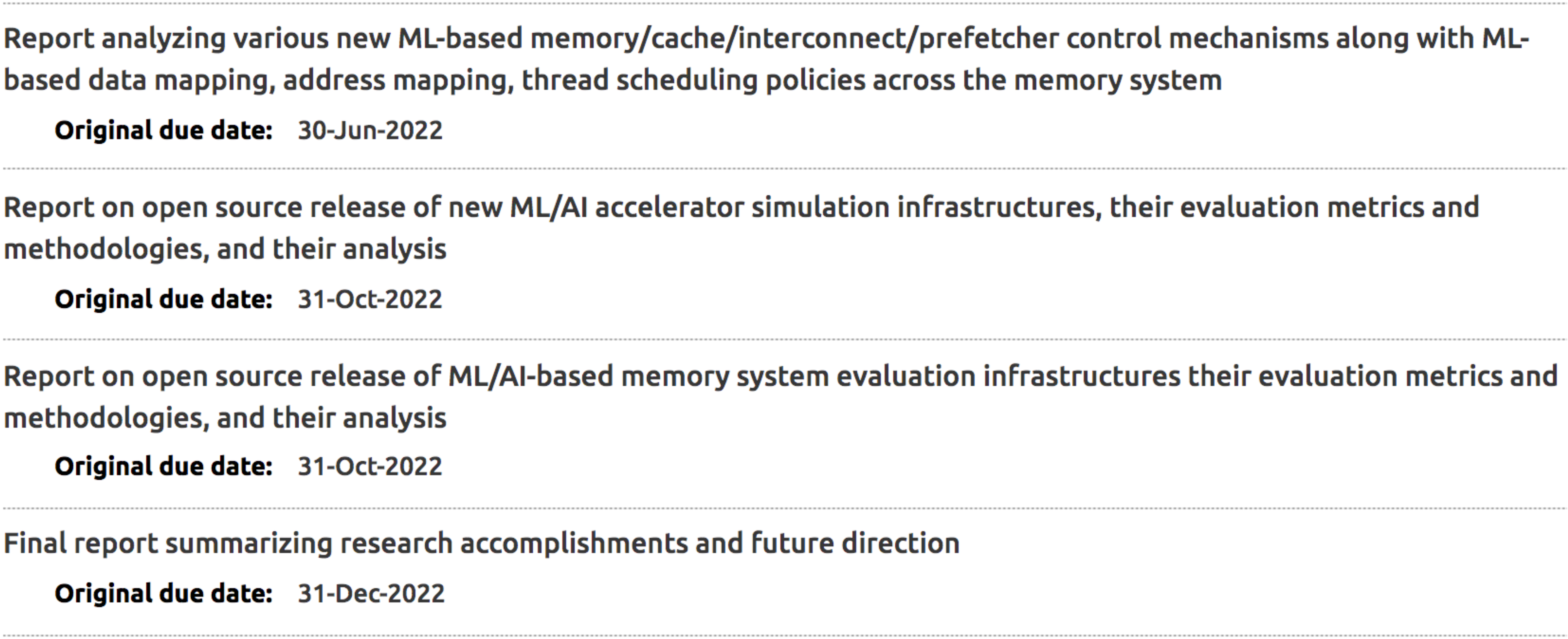
Annual review presentation

Original due date: 1-Sep-2021

Report on analysis of various different memory types, new on-chip/off-chip near-data processing designs, and short-term & long-term options for near-data processing designs for ML/AI accelerators

Original due date: 31-Dec-2021

Task Deliverables (2022)



Report analyzing various new ML-based memory/cache/interconnect/prefetcher control mechanisms along with ML-based data mapping, address mapping, thread scheduling policies across the memory system

Original due date: 30-Jun-2022

Report on open source release of new ML/AI accelerator simulation infrastructures, their evaluation metrics and methodologies, and their analysis

Original due date: 31-Oct-2022

Report on open source release of ML/AI-based memory system evaluation infrastructures their evaluation metrics and methodologies, and their analysis

Original due date: 31-Oct-2022

Final report summarizing research accomplishments and future direction

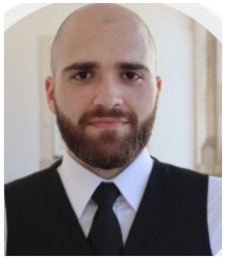
Original due date: 31-Dec-2022

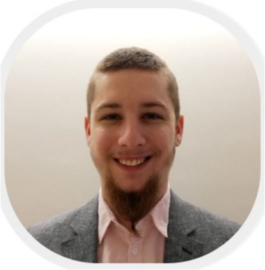
Task Information #2946.001 (1)

- Thrust: AI Hardware

- Task Leader: Onur Mutlu
  - https://people.inf.ethz.ch/omutlu/
  - onur.mutlu@inf.ethz.ch

- Students
  - Rahul Bera (ETH)
  - Joao Ferreira (ETH)
  - Geraldo Francisco de Oliveira Junior (ETH)
  - Konstantinos Kanellopoulos (ETH)
  - Joel Lindegger (ETH)
  - Aditya Manglik (ETH)
  - Rakesh Nadig (ETH)





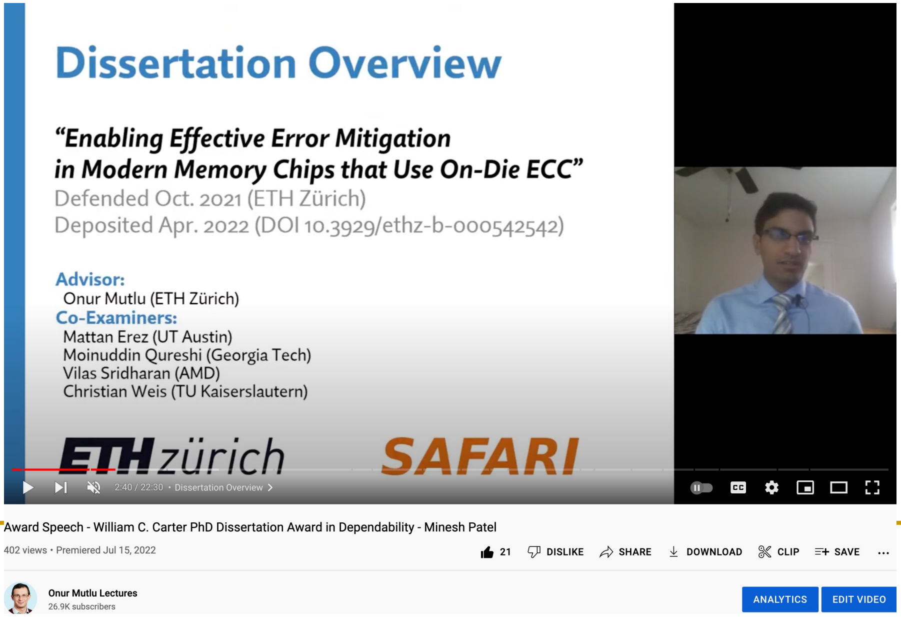
Task Information #2946.001 (2)

- Senior Researchers
  - Juan Gomez Luna (ETH)
  - Haiyu Mao (ETH)
  - Lois Orosa (ETH)
  - Jisung Park (ETH)
  - Gagandeep Singh (ETH)

- More students/postdocs to be added as the task evolves

Recent PhD Graduate

- Minesh Patel
  - October 2021
  - **Enabling Effective Error Mitigation in Memory Chips That Use On-Die Error-Correcting Codes**
  - **2022 William C. Carter PhD Dissertation Award in Dependability**
  - **Best Paper Awards at DSN 2019 & MICRO 2020**
  - https://www.youtube.com/watch?v=0c9bDr18jZE
  - https://arxiv.org/abs/2204.10387
  - https://www.mineshp.com/



Dissertation Overview

"Enabling Effective Error Mitigation in Modern Memory Chips that Use On-Die ECC"

Defended Oct. 2021 (ETH Zürich)

Deposited Apr. 2022 (DOI 10.3929/ethz-b-000542542)

Advisor:
Onur Mutlu (ETH Zürich)

Co-Examiners:
Mattan Erez (UT Austin)
Moinuddin Qureshi (Georgia Tech)
Vilas Sridharan (AMD)
Christian Weis (TU Kaiserslautern)

ETH zürich SAFARI

2:40 / 22:30 • Dissertation Overview

Award Speech - William C. Carter PhD Dissertation Award in Dependability - Minesh Patel

402 views • Premiered Jul 15, 2022

Onur Mutlu Lectures
26.9K subscribers

Recent PostDoc Alumni

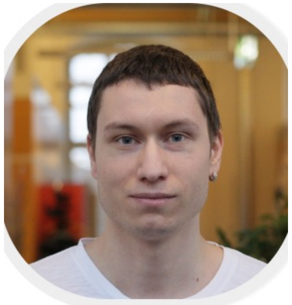
- Dr. Lois Orosa
  - March 2022
  - Director at the Galician Supercomputing Center

- Dr. Gagandeep Singh
  - September 2022
  - Joining AMD Research

- Dr. Jisung Park
  - September 2022
  - Joining POSTECH (South Korea) as Assistant Professor

Soon to Finish PhD

- Hasan Hassan
  - PhD Defense date: September 29, 2022
  - **Improving DRAM Performance, Reliability, and Security by Rigorously Understanding Intrinsic DRAM Operation**
  - https://drive.google.com/file/d/1E5mFYl_SMjCP-7TQ8qt6kRALROGhZs9K/view



Recent Internships

- Dr. Gagandeep Singh
  - February-June 2022
  - Visit to AMD Research

Upcoming TECHCON Presentation

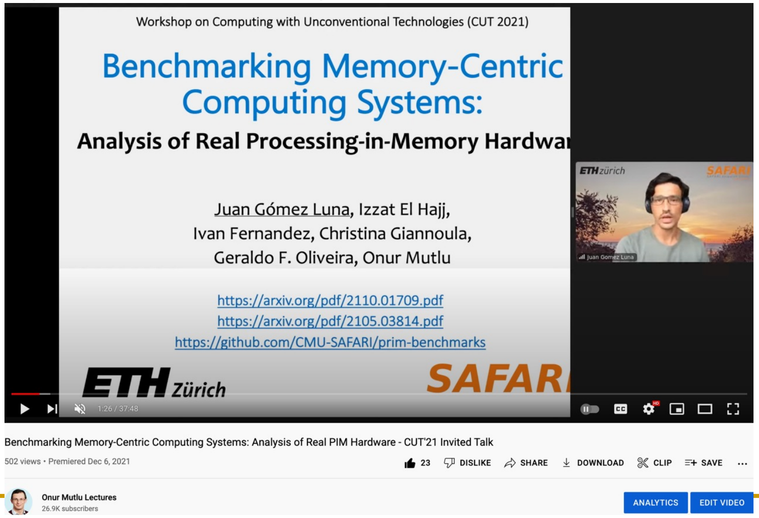
- Dr. Juan Gomez-Luna
  - Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware
  - Based on two major works
    - https://arxiv.org/pdf/2105.03814.pdf
    - https://arxiv.org/pdf/2207.07886.pdf

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

Year: 2021, Pages: 1-7
DOI Bookmark: 10.1109/IGSC54211.2021.9651614

Authors

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



Workshop on Computing with Unconventional Technologies (CUT 2021)

Benchmarking Memory-Centric Computing Systems:

Analysis of Real Processing-in-Memory Hardwar

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

https://arxiv.org/pdf/2110.01709.pdf
https://arxiv.org/pdf/2105.03814.pdf
https://github.com/CMU-SAFARI/prim-benchmarks

Benchmarking Memory-Centric Computing Systems: Analysis of Real PIM Hardware - CUT'21 Invited Talk

502 views • Premiered Dec 6, 2021

https://www.youtube.com/watch?v=nphV36SrysA

Industry Liaisons

- Charles Augustine, Intel
- Pradip Bose, IBM
- Alper Buyuktosunoglu, IBM
- Rosario Cammarota, Intel
- Ramesh Chauhan, Qualcomm
- Prokash Ghosh, NXP
- Jose Joao, ARM
- Arun Joseph, IBM
- Preetham Lobo, IBM
- Nithyakalyani Sampath, TI
- Willem Sanberg, NXP
- Pushkar Sareen, NXP
- Sreenivas Subramoney, Intel
- Xin Zhang, IBM

- We are having and will have regular and irregular meetings with all liaison companies
- Very open to other collaborators, feedback, internships, visits

Industry Interactions (This Year I)

- Intel: Collaborative papers with as part of this task
  - Sreenivas Subramoney, Gurpreet Kalsi, Anant Nori, Kamlesh Pillai, Shankar Balachandran, Bharathwaj Suresh
  - SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping [ISCA 2022]
  - pLUTo: Enabling Massively Parallel Computation In DRAM via Lookup Tables [MICRO 2022]
  - Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction [MICRO 2022]
  - ApHMM: Accelerating Profile Hidden Markov Models for Fast and Energy-Efficient Genome Analysis [arXiv 2022]

- IBM: Collaborative papers
  - Dionysios Diamantopoulos, Christoph Hagleitner
  - Accelerating Weather Prediction Using Near-Memory Reconfigurable Fabric [TRETS 2022]

Industry Interactions (This Year II)

- IBM: Collaborative EU Horizon Project BioPIM
  - Abu Sebastian, Irem Boybat (IBM Research Zurich)
  - http://www.biopim.eu/
  - **BioPIM** project aims to leverage the emerging processing-in-memory (PIM) technologies to enable powerful edge computing.
  - Synergistic with this task
  - We will focus on co-designing algorithms and data structures commonly used in bioinformatics together with several types of PIM architectures to obtain the highest benefit in cost, energy, and time savings.
  - BioPIM will also impact other fields that employ similar algorithms.
  - Our designs and algorithms will not be limited to cheap hardware, and they will impact computation efficiency on all forms of computing environments including cloud platforms.
  - The targeted breakthrough of **BioPIM** is to invent and leverage in-memory computing architectures to fundamentally improve the performance and energy efficiency of various important bioinformatics algorithms to make mobile genomics a reality

Industry Interactions (This Year III)

- Qualcomm: In-person Visit & Talk
  - Ramesh Chauhan
  - May 2022

- IBM Research: In-person Visit & Talk
  - Pradip Bose, Karthik Swaminathan, Alper Buyuktosunoglu, Krishnan Kailas
  - May 2022

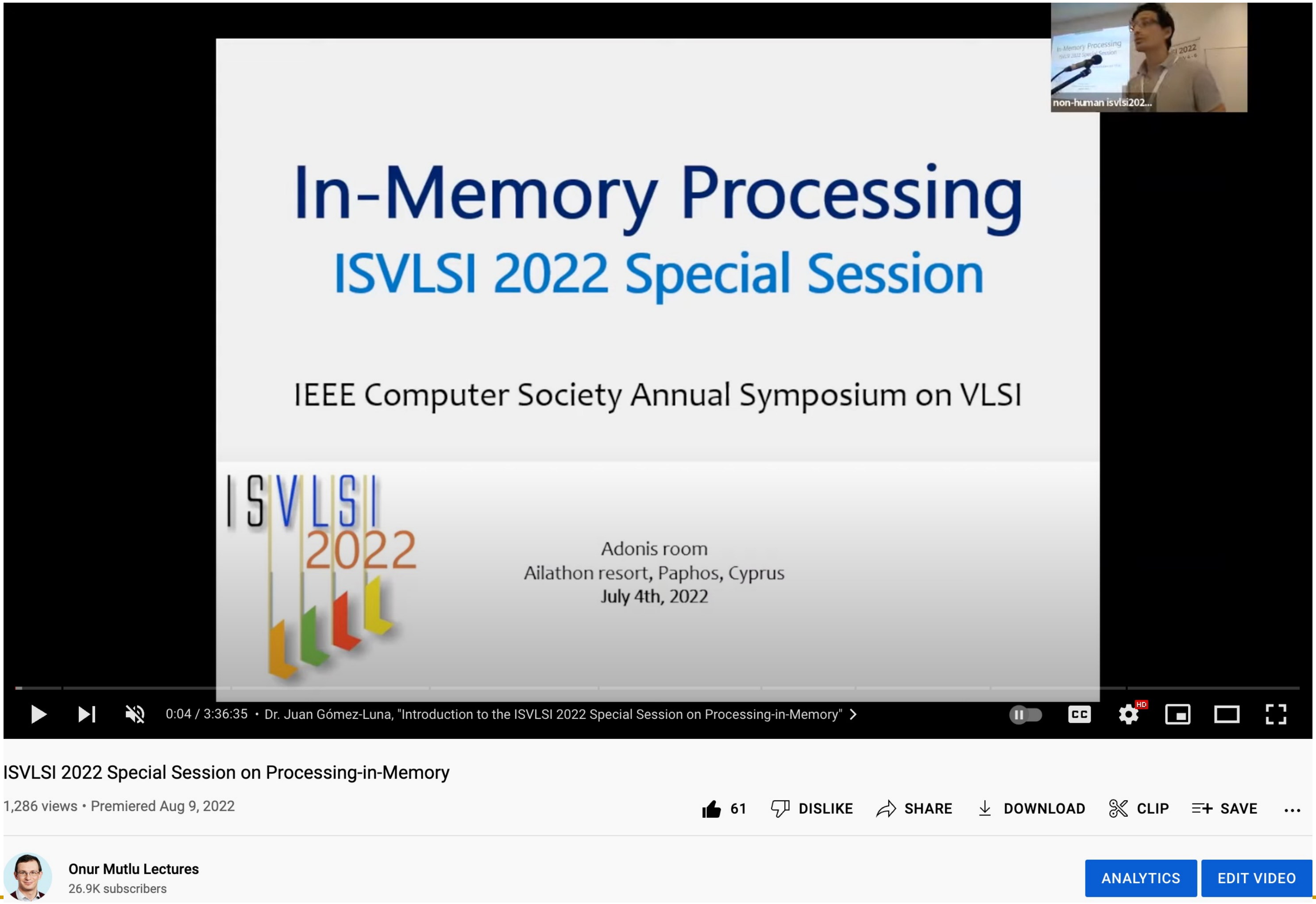
- Intel: Keynote Talk at the Intel Interconnect & Connectivity Summit
  - Debendra Das Sharma
  - **"Memory-Centric Computing"**
    *Keynote Talk at the Intel Interconnect & Connectivity Summit (**IICS**)*, Virtual, 9 February 2022.
    [Slides (pptx) (pdf)]

Posters for Annual Review 2022

- Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning [ISCA 2022]
  - Gagandeep Singh
- SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping [ISCA 2022]
  - Damla Senol Cali, Joel Lindegger
- Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction [MICRO 2022]
  - Rahul Bera
- Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design [ICDE 2022]
  - Geraldo Francisco de Oliveira Junior
- Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware [IEEE Access 2022]
  - Juan Gómez-Luna
- Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks [PACT 2021]
  - Geraldo Francisco de Oliveira Junior

Special Research Sessions & Courses

- Special Session at ISVLSI 2022: 9 cutting-edge talks



In-Memory Processing

ISVLSI 2022 Special Session

IEEE Computer Society Annual Symposium on VLSI

Adonis room
Ailathon resort, Paphos, Cyprus
July 4th, 2022

0:04 / 3:36:35 • Dr. Juan Gómez-Luna, "Introduction to the ISVLSI 2022 Special Session on Processing-in-Memory"

ISVLSI 2022 Special Session on Processing-in-Memory

1,286 views • Premiered Aug 9, 2022

Onur Mutlu Lectures
26.9K subscribers

SAFARI

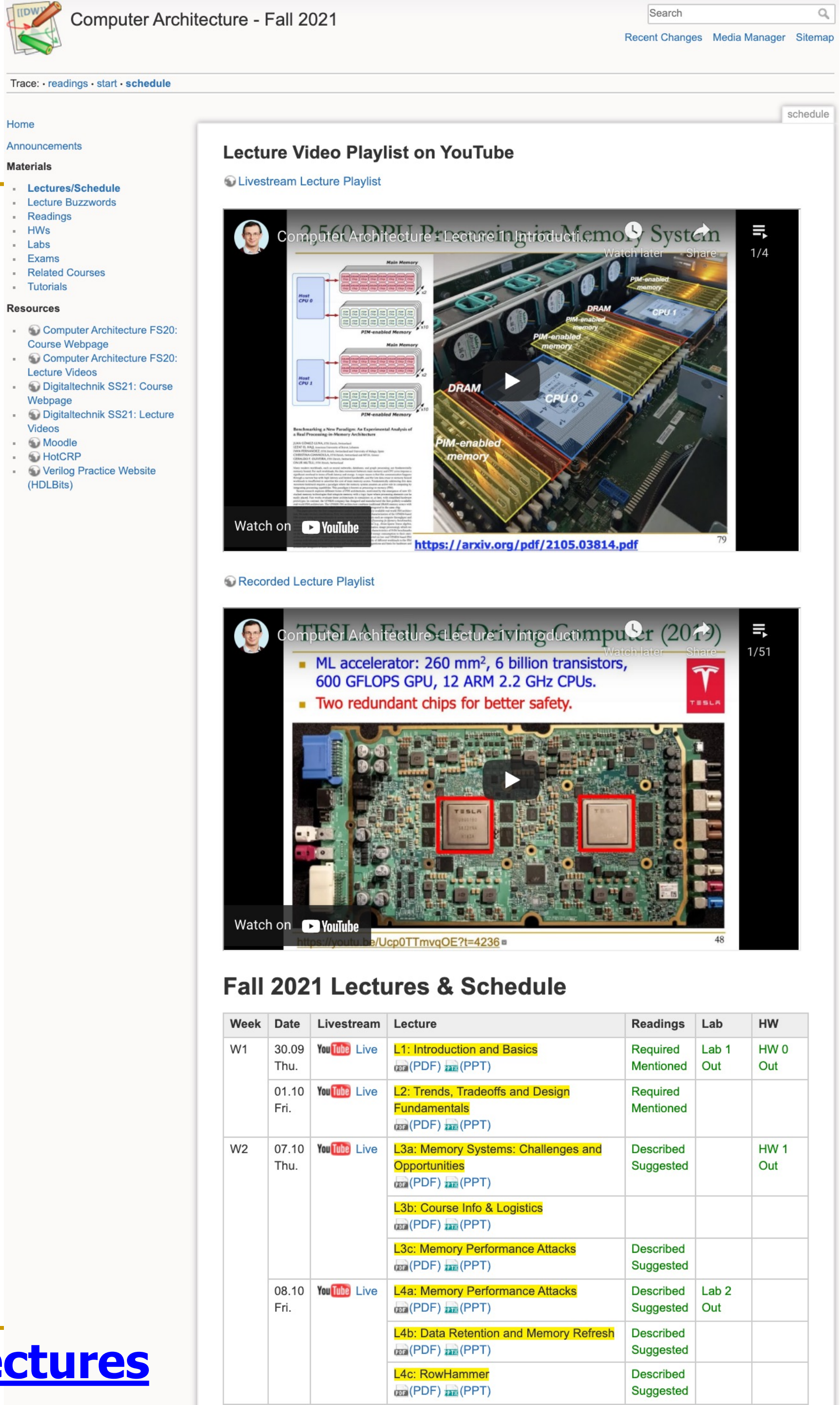
https://www.youtube.com/watch?v=qeukNs5XI3g

Comp Arch (Fall'21)

- **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_5EFgo&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



Computer Architecture - Fall 2021

Search

Recent Changes Media Manager Sitemap

Trace: · readings · start · **schedule**

schedule

Home

Announcements

Materials

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

Resources

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

Lecture Video Playlist on YouTube

Livestream Lecture Playlist

Computer Architecture - Lecture 1: Introducti...
Watch later
Share
1/4
Watch on YouTube
https://arxiv.org/pdf/2105.03814.pdf

Recoded Lecture Playlist

Computer Architecture - Lecture 1: Introducti...
Watch later
Share
1/51
ML accelerator: 260 mm², 6 billion transistors, 600 GFLOPS GPU, 12 ARM 2.2 GHz CPUs.
Two redundant chips for better safety.
Watch on YouTube
https://youtu.be/Ucp0TTmvqOE?t=4236

Fall 2021 Lectures & Schedule

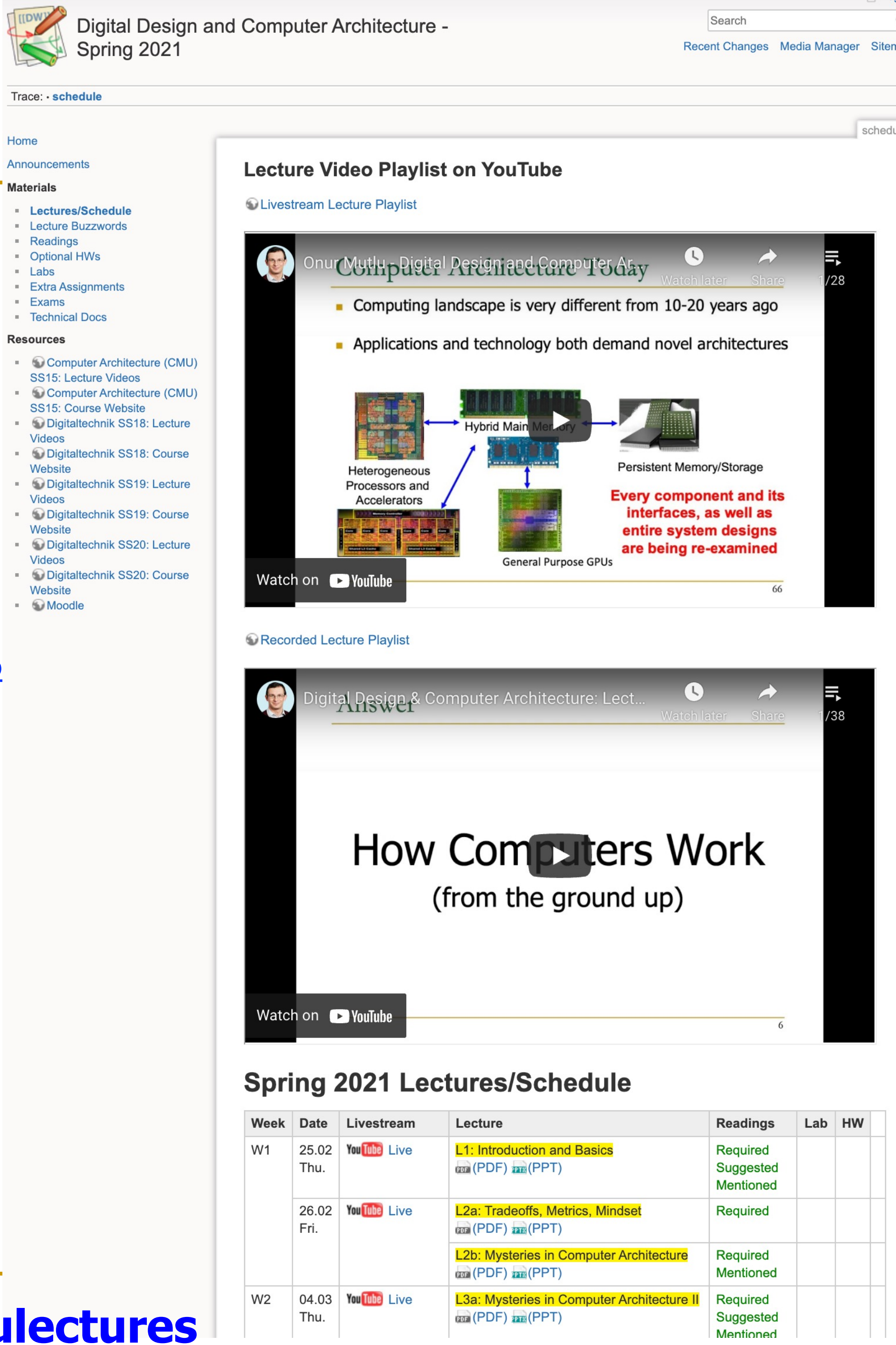
Week	Date	Livestream	Lecture	Readings	Lab	HW
W1	30.09 Thu.	YouTube Live	L1: Introduction and Basics (PDF) (PPT)	Required Mentioned	Lab 1 Out	HW 0 Out
	01.10 Fri.	YouTube Live	L2: Trends, Tradeoffs and Design Fundamentals (PDF) (PPT)	Required Mentioned		
W2	07.10 Thu.	YouTube Live	L3a: Memory Systems: 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		

https://www.youtube.com/onurmutlulectures

DDCA (Spring 2022)

- **Spring 2022 Edition:**
  - https://safari.ethz.ch/digitaltechnik/spring2022/doku.php?id=schedule
- **Spring 2021 Edition:**
  - https://safari.ethz.ch/digitaltechnik/spring2021/doku.php?id=schedule
- **Youtube Livestream (Spring 2022):**
  - https://www.youtube.com/watch?v=cpXdE3HwvK0&list=PL5Q2soXY2Zi97Ya5DEUpMpO2bbAoaG7c6
- **Youtube Livestream (Spring 2021):**
  - https://www.youtube.com/watch?v=LbC0EZY8yw4&list=PL5Q2soXY2Zi_uej3aY39YB5pfW4SJ7LlN

- Bachelor's course
  - 2nd semester at ETH Zurich
  - Rigorous introduction into "How Computers Work"
  - Digital Design/Logic
  - Computer Architecture
  - 10 FPGA Lab Assignments



Digital Design and Computer Architecture - Spring 2021

Trace: · schedule

Home

Announcements

Materials

- Lectures/Schedule
- Lecture Buzzwords
- Readings
- Optional HWs
- Labs
- Extra Assignments
- Exams
- Technical Docs

Resources

- Computer Architecture (CMU) SS15: Lecture Videos
- Computer Architecture (CMU) SS15: Course Website
- Digitaltechnik SS18: Lecture Videos
- Digitaltechnik SS18: Course Website
- Digitaltechnik SS19: Lecture Videos
- Digitaltechnik SS19: Course Website
- Digitaltechnik SS20: Lecture Videos
- Digitaltechnik SS20: Course Website
- Moodle

Lecture Video Playlist on YouTube

Livestream Lecture Playlist

Onur Mutlu - Digital Design and Computer Ar...
Computer Architecture Today
Watch later
Share
1/28
Computing landscape is very different from 10-20 years ago
Applications and technology both demand novel architectures
Hybrid Main Memory
Heterogeneous Processors and Accelerators
Persistent Memory/Storage
Every component and its interfaces, as well as entire system designs are being re-examined
General Purpose GPUs
Watch on YouTube
66

Recorded Lecture Playlist

Spring 2021 Lectures/Schedule

Week	Date	Livestream	Lecture	Readings	Lab	HW
W1	25.02 Thu.	YouTube Live	L1: Introduction and Basics (PDF) (PPT)	Required Suggested Mentioned		
	26.02 Fri.	YouTube Live	L2a: Tradeoffs, Metrics, Mindset (PDF) (PPT)	Required		
			L2b: Mysteries in Computer Architecture (PDF) (PPT)	Required Mentioned		
W2	04.03 Thu.	YouTube Live	L3a: Mysteries in Computer Architecture II (PDF) (PPT)	Required Suggested Mentioned		

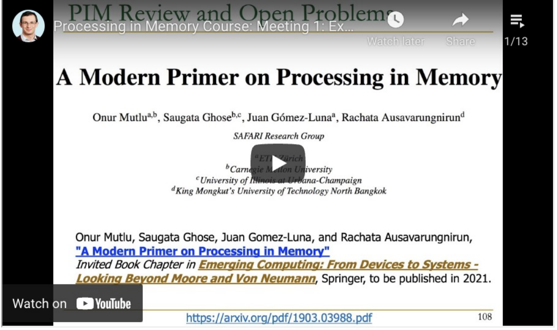
https://www.youtube.com/onurmutlulectures

PIM Course (Spring 2022)

- **[Spring 2022 Edition:](https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=processing_in_memory)**
  - 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 Review and Open Problems
Processing in Memory Course: Meeting 1: Ex...
Watch later
Share
1/13
Watch on YouTube
108

A Modern Primer on Processing in Memory

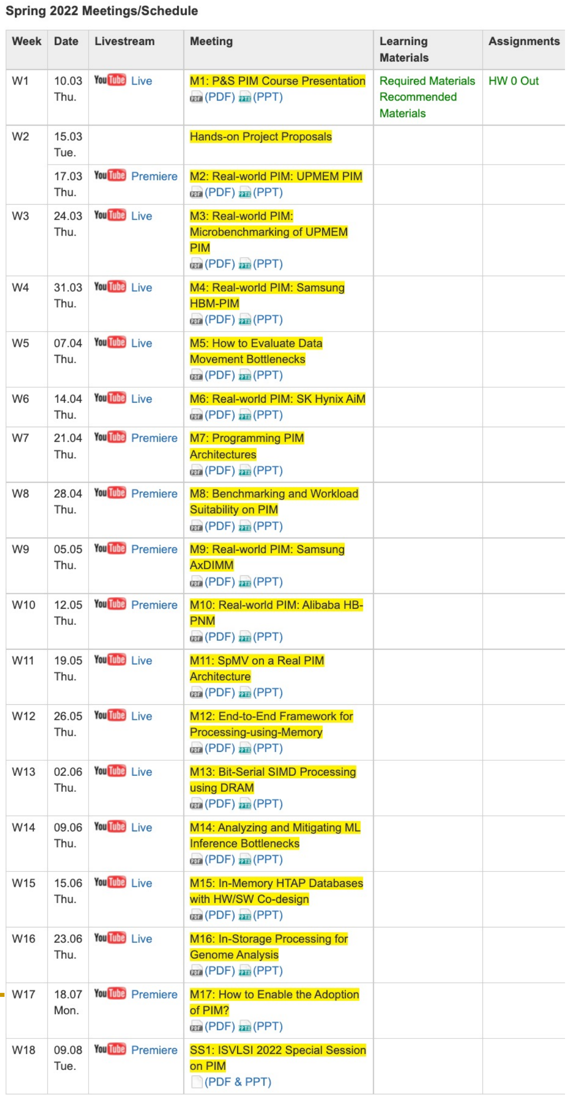
Onur Mutlu[a,b], Saugata Ghose[b,c], Juan Gómez-Luna[a], Rachata Ausavarungnirun[d]

SAFARI Research Group

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

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

https://arxiv.org/pdf/1903.03988.pdf

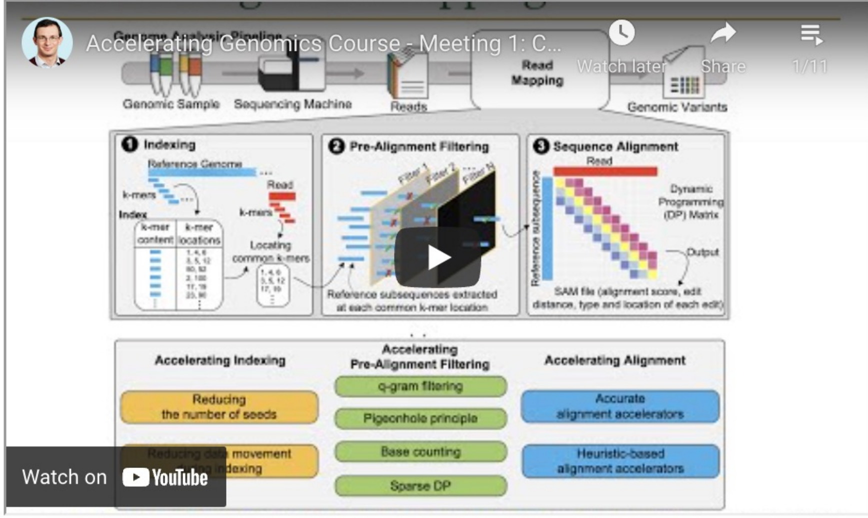


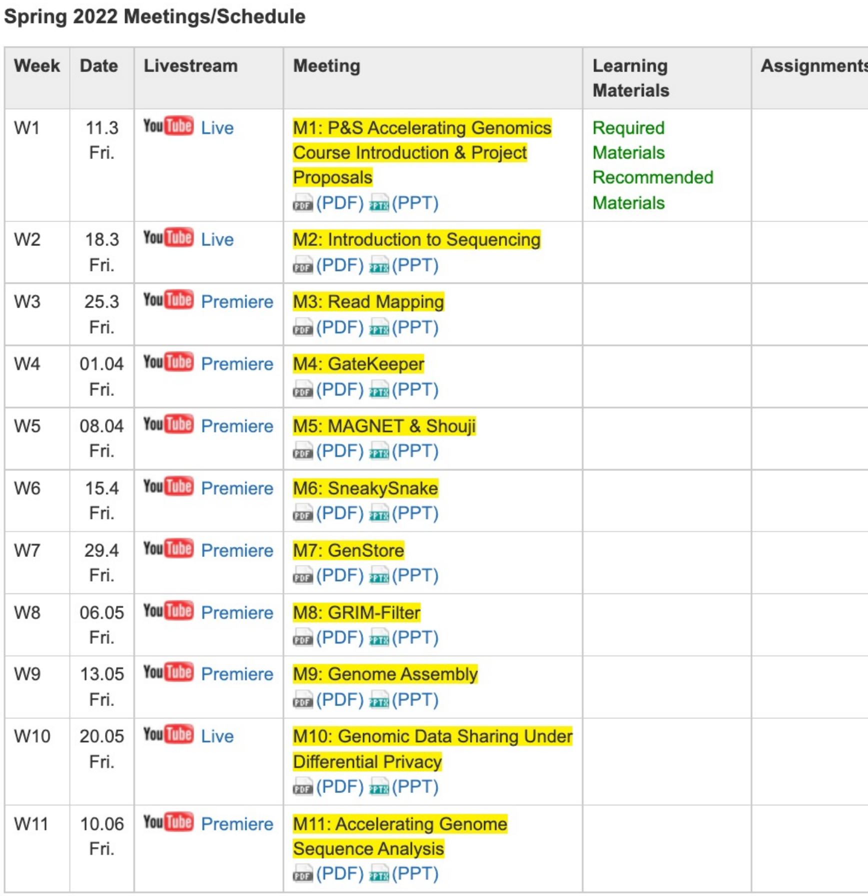
Spring 2022 Meetings/Schedule

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

Genomics (Spring 2022)

- **Spring 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=bioinformatics
- **Youtube Livestream:**
  - https://www.youtube.com/watch?v=DEL_5A_Y3TI&list=PL5Q2soXY2Zi8NrPDgOR1yRU_Cxxjw-u18
- Project course
  - Taken by Bachelor's/Master's students
  - Genomics lectures
  - Hands-on research exploration
  - Many research readings


Accelerating Genomics Course - Meeting 1: C...
Watch later
Share
1/11
Genome Analysis Pipeline
Genomic Sample
Sequencing Machine
Reads
Read Mapping
Genomic Variants
1 Indexing
Reference Genome
Read
k-mers
Index
k-mer content
k-mer locations
Locating common k-mers
2 Pre-Alignment Filtering
Filter 1
Filter 2
Filter N
Reference subsequences extracted at each common k-mer location
3 Sequence Alignment
Read
Reference subsequence
Dynamic Programming (DP) Matrix
Output
SAM file (alignment score, edit distance, type and location of each edit)
Accelerating Indexing
Reducing the number of seeds
Reducing data movement during indexing
Accelerating Pre-Alignment Filtering
q-gram filtering
Pigeonhole principle
Base counting
Sparse DP
Accelerating Alignment
Accurate alignment accelerators
Heuristic-based alignment accelerators
Watch on YouTube



Spring 2022 Meetings/Schedule

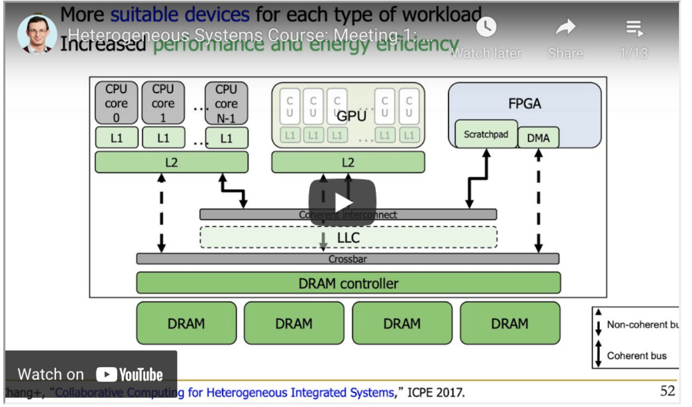
Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	11.3 Fri.	YouTube Live	M1: P&S Accelerating Genomics Course Introduction & Project Proposals (PDF) (PPT)	Required Materials Recommended Materials	
W2	18.3 Fri.	YouTube Live	M2: Introduction to Sequencing (PDF) (PPT)		
W3	25.3 Fri.	YouTube Premiere	M3: Read Mapping (PDF) (PPT)		
W4	01.04 Fri.	YouTube Premiere	M4: GateKeeper (PDF) (PPT)		
W5	08.04 Fri.	YouTube Premiere	M5: MAGNET & Shouji (PDF) (PPT)		
W6	15.4 Fri.	YouTube Premiere	M6: SneakySnake (PDF) (PPT)		
W7	29.4 Fri.	YouTube Premiere	M7: GenStore (PDF) (PPT)		
W8	06.05 Fri.	YouTube Premiere	M8: GRIM-Filter (PDF) (PPT)		
W9	13.05 Fri.	YouTube Premiere	M9: Genome Assembly (PDF) (PPT)		
W10	20.05 Fri.	YouTube Live	M10: Genomic Data Sharing Under Differential Privacy (PDF) (PPT)		
W11	10.06 Fri.	YouTube Premiere	M11: Accelerating Genome Sequence Analysis (PDF) (PPT)		

Hetero. Systems (Spring'22)

- **Spring 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=heterogeneous_systems

- **Youtube Livestream:**
  - https://www.youtube.com/watch?v=oFO5fTrgFIY&list=PL5Q2soXY2Zi9XrgXR38IM_FTjmY6h7Gzm

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

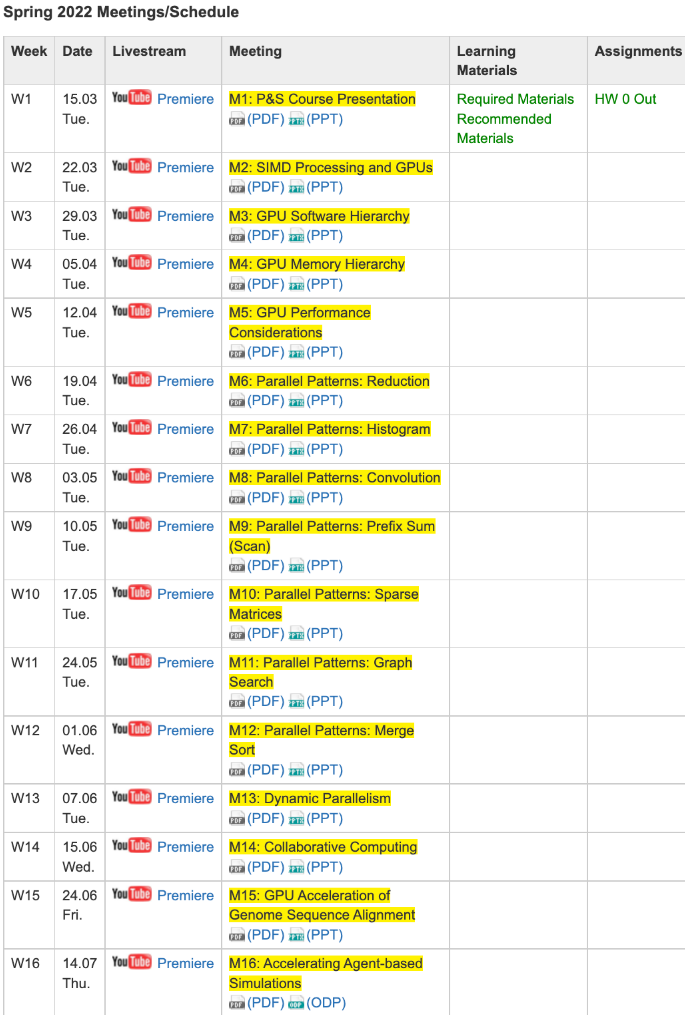


More suitable devices for each type of workload

Increased performance and energy efficiency

Heterogeneous Systems Course: Meeting 1: ...

hang+, "Collaborative Computing for Heterogeneous Integrated Systems," ICPE 2017.

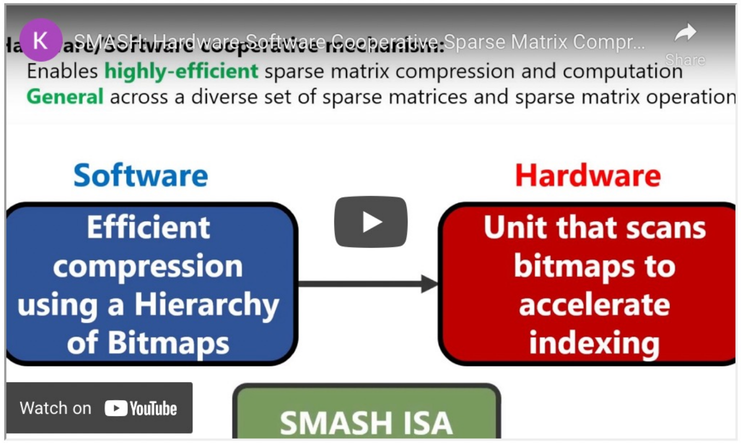


Spring 2022 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	15.03 Tue.	YouTube Premiere	M1: P&S Course Presentation (PDF) (PPT)	Required Materials Recommended Materials	HW 0 Out
W2	22.03 Tue.	YouTube Premiere	M2: SIMD Processing and GPUs (PDF) (PPT)		
W3	29.03 Tue.	YouTube Premiere	M3: GPU Software Hierarchy (PDF) (PPT)		
W4	05.04 Tue.	YouTube Premiere	M4: GPU Memory Hierarchy (PDF) (PPT)		
W5	12.04 Tue.	YouTube Premiere	M5: GPU Performance Considerations (PDF) (PPT)		
W6	19.04 Tue.	YouTube Premiere	M6: Parallel Patterns: Reduction (PDF) (PPT)		
W7	26.04 Tue.	YouTube Premiere	M7: Parallel Patterns: Histogram (PDF) (PPT)		
W8	03.05 Tue.	YouTube Premiere	M8: Parallel Patterns: Convolution (PDF) (PPT)		
W9	10.05 Tue.	YouTube Premiere	M9: Parallel Patterns: Prefix Sum (Scan) (PDF) (PPT)		
W10	17.05 Tue.	YouTube Premiere	M10: Parallel Patterns: Sparse Matrices (PDF) (PPT)		
W11	24.05 Tue.	YouTube Premiere	M11: Parallel Patterns: Graph Search (PDF) (PPT)		
W12	01.06 Wed.	YouTube Premiere	M12: Parallel Patterns: Merge Sort (PDF) (PPT)		
W13	07.06 Tue.	YouTube Premiere	M13: Dynamic Parallelism (PDF) (PPT)		
W14	15.06 Wed.	YouTube Premiere	M14: Collaborative Computing (PDF) (PPT)		
W15	24.06 Fri.	YouTube Premiere	M15: GPU Acceleration of Genome Sequence Alignment (PDF) (PPT)		
W16	14.07 Thu.	YouTube Premiere	M16: Accelerating Agent-based Simulations (PDF) (ODP)		

HW/SW Co-Design (Spring 2022)

- **Spring 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=hw_sw_codesign
- **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

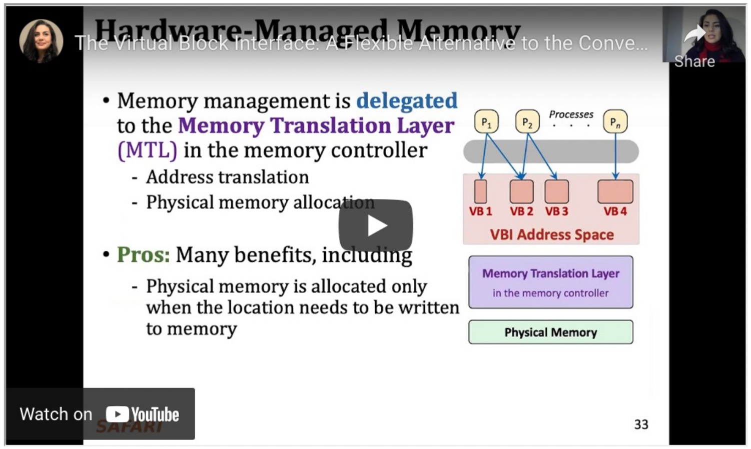

SMASH: Hardware-Software Cooperative Sparse Matrix Compr...

Hardware/Software cooperative mechanism:

Enables **highly-efficient** sparse matrix compression and computation

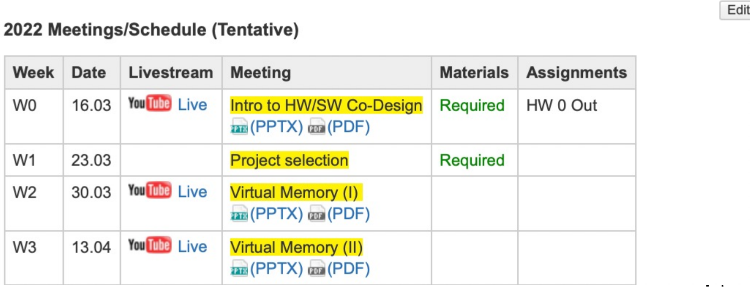
General across a diverse set of sparse matrices and sparse matrix operation

K
Share
Software
Efficient compression using a Hierarchy of Bitmaps
Hardware
Unit that scans bitmaps to accelerate indexing
SMASH ISA
Watch on YouTube


The Virtual Block Interface: A Flexible Alternative to the Conve...

Hardware-Managed Memory

- Memory management is **delegated** to the **Memory Translation Layer (MTL)** in the memory controller
  - Address translation
  - Physical memory allocation
- **Pros:** Many benefits, including
  - Physical memory is allocated only when the location needs to be written to memory



2022 Meetings/Schedule (Tentative)

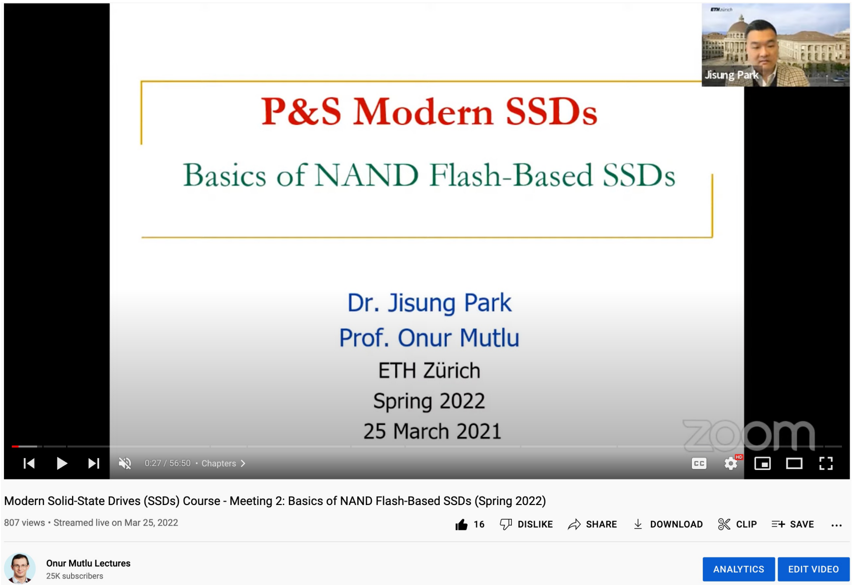
Week	Date	Livestream	Meeting	Materials	Assignments
W0	16.03	YouTube Live	Intro to HW/SW Co-Design (PPTX) (PDF)	Required	HW 0 Out
W1	23.03		Project selection	Required	
W2	30.03	YouTube Live	Virtual Memory (I) (PPTX) (PDF)		
W3	13.04	YouTube Live	Virtual Memory (II) (PPTX) (PDF)		

SSD Course (Spring 2022)

- **Spring 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=modern_ssds

- **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



P&S Modern SSDs

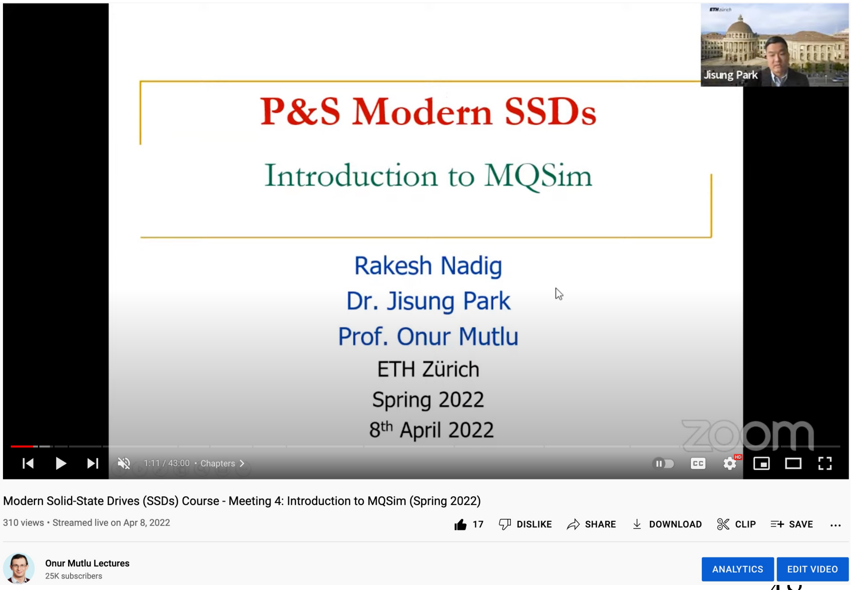
Basics of NAND Flash-Based SSDs

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

Onur Mutlu Lectures
25K subscribers



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

Onur Mutlu Lectures
25K subscribers

Agenda

- Problem and Background
- Task Overview
- Technical Challenges, Goals and Ideas
- Ideas, Results and Papers from the Past Year

Two Major Thrusts

1. Memory system design for AI/ML workloads/accelerators

2. AI/ML techniques for improving memory system designs

Thrust 1 Exploration Ideas

1.1. Comprehensive Energy and Performance Analysis of ML/AI Accelerator Execution on Key ML/AI Workloads

1.2. Cache/Buffer, On-Chip Memory, Interconnect, Memory Controller Designs for ML Accelerators and Their Interfaces

1.3. Complete on-chip ML/AI accelerator designs with careful data orchestration and on-chip memory management.

1.4. On-chip & off-chip near-data processing (NDP) designs, interfaces, evaluation, programming for AI/ML workloads

1.5. Evaluation and understanding of both short-term and long-term options for NDP for AI/ML Workloads

1.6. Use of NVM devices, simple customized DRAM and 3D-stacked Memory+Logic for AI/ML Acceleration

1.7. High-Fidelity and Highly-Flexible Open Source Simulation & Modeling Infrastructures for ML/AI Memory Systems

This talk

Two Major Thrusts

1. Memory system design for AI/ML workloads/accelerators

2. AI/ML techniques for improving memory system designs

Thrust 2 Exploration Ideas

2.1. Comprehensive performance and energy analysis of rigid policies in the memory hierarchy – how far are they from the ideal policies? What is the maximum potential ML techniques can achieve?

2.2. New caching, prefetching, mem. controller, runahead, compression policies that are directed with appropriate ML techniques

2.3. Rigorous specification and coordination of ML-based on-chip cache, prefetch, DRAM, NVM, hybrid mem. Controllers

2.4. Design and evaluation of new ML-based techniques to manage hybrid memories consisting of multiple different technologies

This talk

2.5. Design and evaluation of new ML-based data mapping policies across on-chip caches and memory controllers

2.6. Design and evaluation of new ML-based thread scheduling policies in both SMT and memory controllers

2.7. High-Fidelity and Highly-Flexible Open Source Simulation & Modeling Infrastructures for ML-Based Controllers

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?

Two Major Thrusts & Their Synergies

1. Memory system design for AI/ML workloads/accelerators

2. AI/ML techniques for improving memory system designs

Agenda

- Problem and Background
- Task Overview
- Technical Challenges, Goals and Ideas
- Ideas, Results and Papers from the Past Year

Initial Results in Year I (2020 Review)

- GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis **[MICRO 2020]**
- NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling **[FPL 2020]**
- An Experimental Study of Reduced-Voltage Operation in Modern FPGAs for Neural Network Acceleration **[DSN 2020]**
- NATSA: A Near-Data Processing Accelerator for Time Series Analysis **[ICCD 2020]**
- Robust Machine Learning Systems: Challenges, Current Trends, Perspectives, and the Road Ahead **[IEEE D&T 2020]**
- Accelerating Genome Analysis: A Primer on an Ongoing Journey **[IEEE Micro 2020]**
- SMASH Open Source Software Code Release **[GitHub]**

Initial Results in Year I (2020 Ongoing)

- Efficiently Accelerating Edge ML Inference by Exploiting Layer Heterogeneity: An Empirical Study with Google Edge Models [Ongoing]
- A New Methodology and Open-Source Benchmark Suite for Evaluating Data Movement Bottlenecks: A Near-Data Processing Case Study [Ongoing]
- Accelerating Profile Hidden Markov Models in Computational Biology Applications [Ongoing]
- StenCache: A Near-Cache Accelerator for Stencil Computations [Ongoing]
- SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM [Ongoing]
- Polynesia: Enabling Effective Hybrid Transactional/Analytical Databases with Specialized Hardware/Software Co-Design [Ongoing]
- Reinforcement Learning based Prefetch Generation [Ongoing]
- Benchmarking a New Paradigm: Understanding a Modern Processing-in-Memory Architecture [Ongoing]

Year II Results (2021 Annual Review I)

- Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks [PACT 2021]
- Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning [MICRO 2021]
- Refresh Triggered Computation: Improving the Energy Efficiency of Convolutional Neural Network Accelerators [TACO 2020]
- SynCron: Efficient Synchronization Support for Near-Data-Processing Architectures [HPCA 2021]
- SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM [ASPLOS 2021]

Year II Results (2021 Annual Review II)

- DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks [IEEE Access 2021]
- Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture [Arxiv, 2021]
- FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications [IEEE Micro 2021]
- A Modern Primer on Processing in Memory [Arxiv, 2020]
- Sibyl: A Reinforcement Learning Approach to Data Placement in Hybrid Storage Systems [Ongoing]

Year III Results (2022 Annual Review 1)

- Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System [IEEE Access’22]
- Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware [CUT 2021]
- An Experimental Evaluation of Machine Learning Training on a Real Processing-in-Memory System [arXiv 2022]
- SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures [SIGMETRICS 2022]
- High-throughput Pairwise Alignment with the Wavefront Algorithm using Processing-in-Memory [HICOMB 2022]
- PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM [arXiv 2021]

Part of Thrust 1:
Real PIM Systems

Year III Results (2022 Annual Review 2)

- SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping [ISCA 2022]

- GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis [ASPLOS 2022]

- Algorithmic Improvement and GPU Acceleration of the GenASM Algorithm [HICOMB 2022]

- Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design [ICDE 2022]

- Flash-Cosmos: In-Flash Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory [MICRO 2022]

Part of Thrust 1

Year III Results (2022 Annual Review 3)

- Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning [ISCA 2022]
- Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction [MICRO 2022]

Part of Thrust 2

- GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping [MICRO 2022]
- pLUTo: Enabling Massively Parallel Computation via In DRAM via Lookup Tables [MICRO 2022]

Part of Thrust 1

- DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression [FAST 2022]

- A Modern Primer on Processing in Memory [Arxiv, Updated 2022]

Year III Results (2022 Annual Review 4)

- EcoFlow: Efficient Convolutional Dataflows for Low-Power Neural Network Accelerators [arXiv 2022] https://arxiv.org/abs/2202.02310

- ApHMM: Accelerating Profile Hidden Markov Models for Fast and Energy-Efficient Genome Analysis [arXiv 2022] https://arxiv.org/abs/2207.09765

- Accelerating Weather Prediction Using Near-Memory Reconfigurable Fabric [TRETS 2022] https://arxiv.org/abs/2107.08716

Third Year Results: More Detail

Year III Results (2022 Annual Review 1)

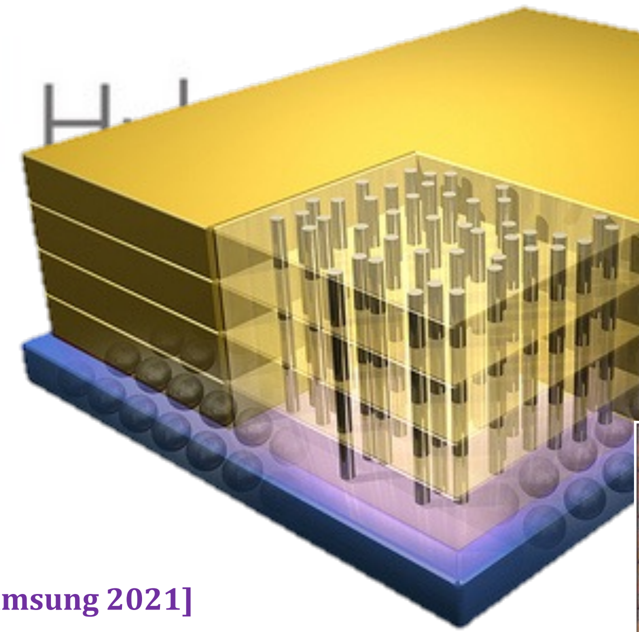
- Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System [IEEE Access'22]
- Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware [CUT 2021]
- An Experimental Evaluation of Machine Learning Training on a Real Processing-in-Memory System [arXiv 2022]
- SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures [SIGMETRICS 2022]
- High-throughput Pairwise Alignment with the Wavefront Algorithm using Processing-in-Memory [HICOMB 2022]
- PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM [arXiv 2021]

Eliminating the Adoption Barriers

Processing-in-Memory

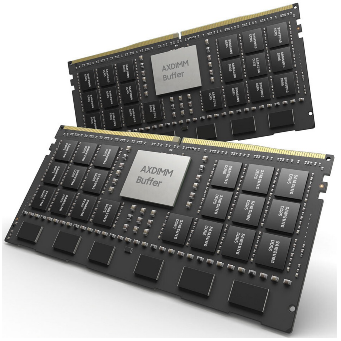
in the Real World

Processing-in-Memory Landscape Today

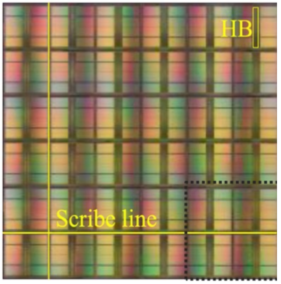


Micron

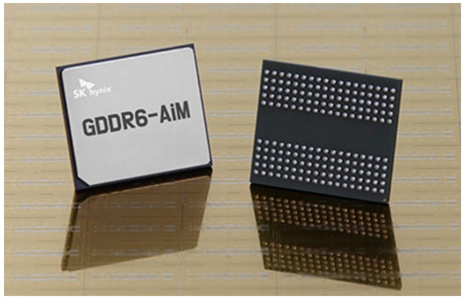
AUTOMATA PROCESSING



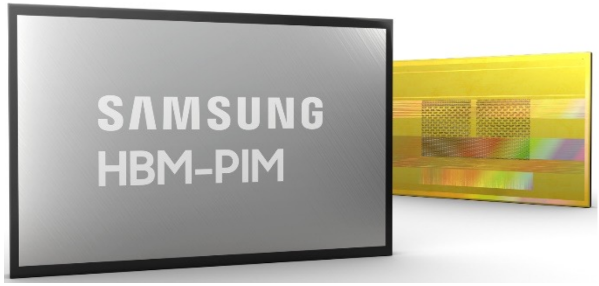
[Samsung 2021]



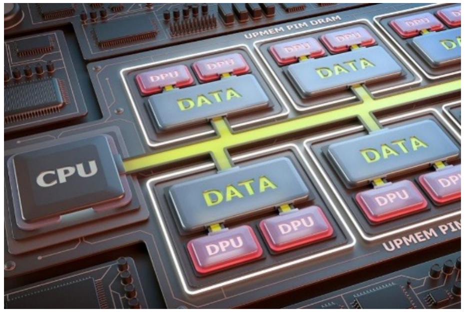
[Alibaba 2022]



[SK Hynix 2022]



[Samsung 2021]

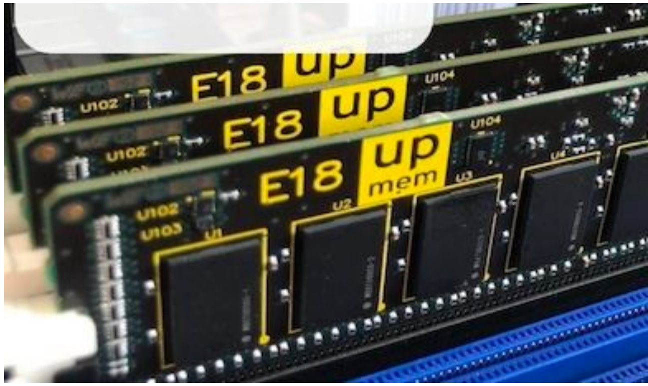


[UPMEM 2019]

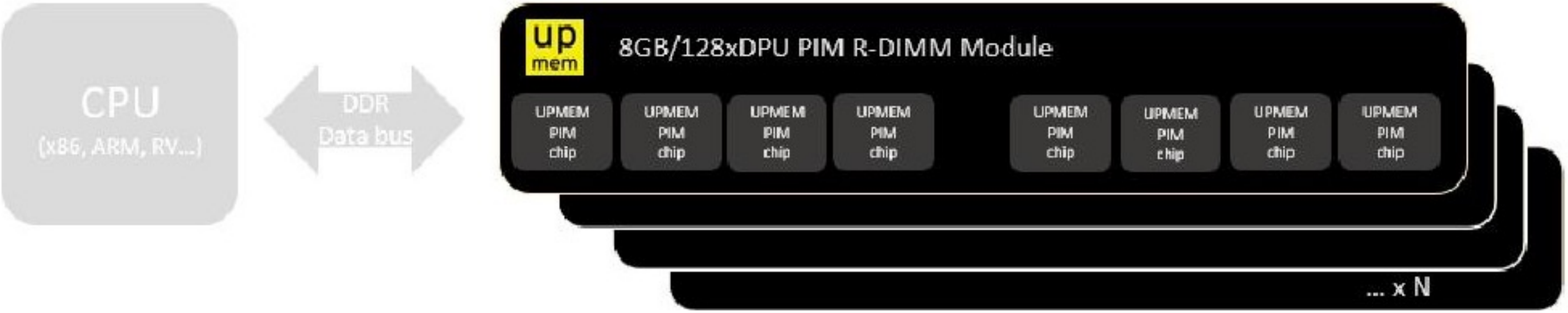
This does not include many experimental chips and startups

UPMEM Processing-in-DRAM Engine (2019)

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



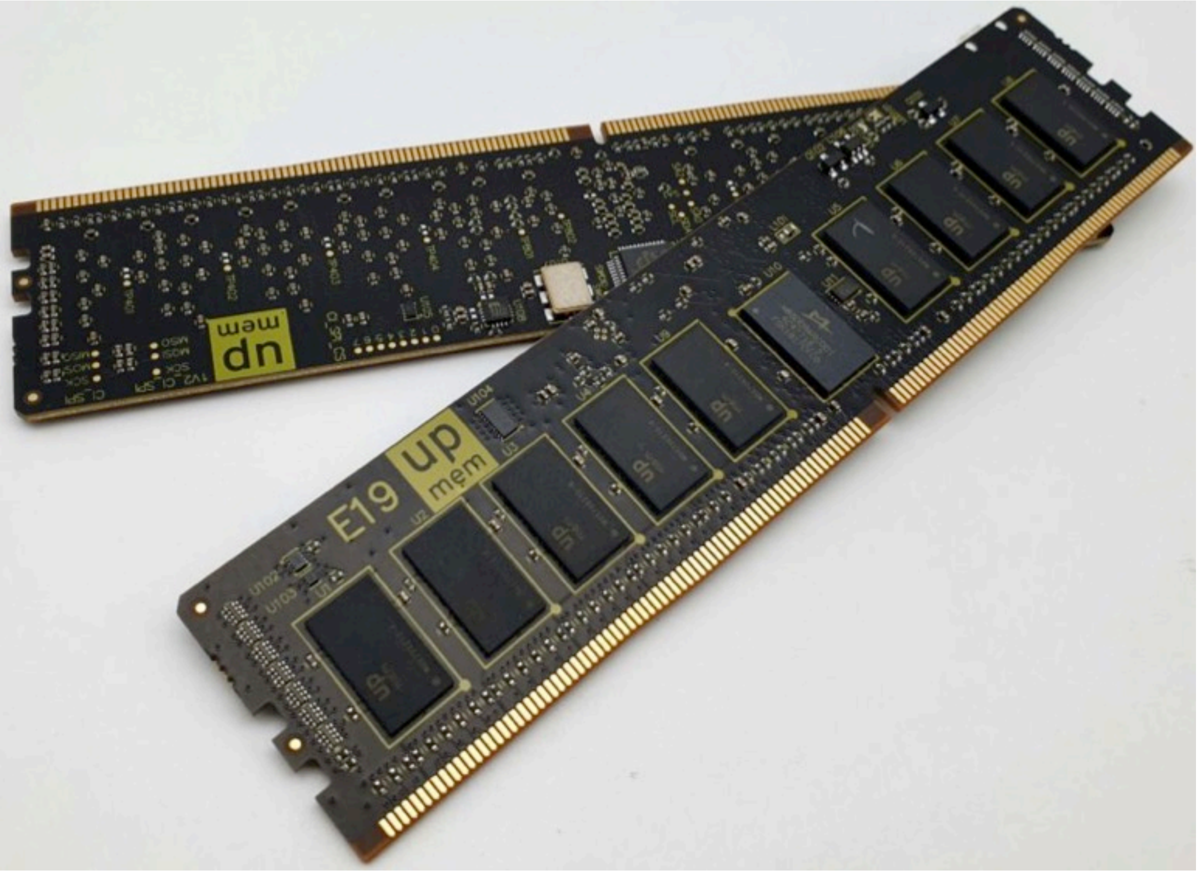
- Replaces **standard** DIMMs
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth



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

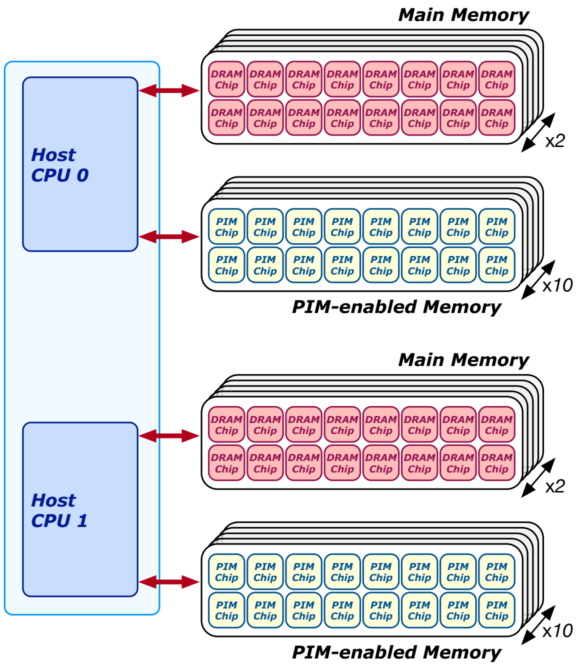
UPMEM Memory Modules

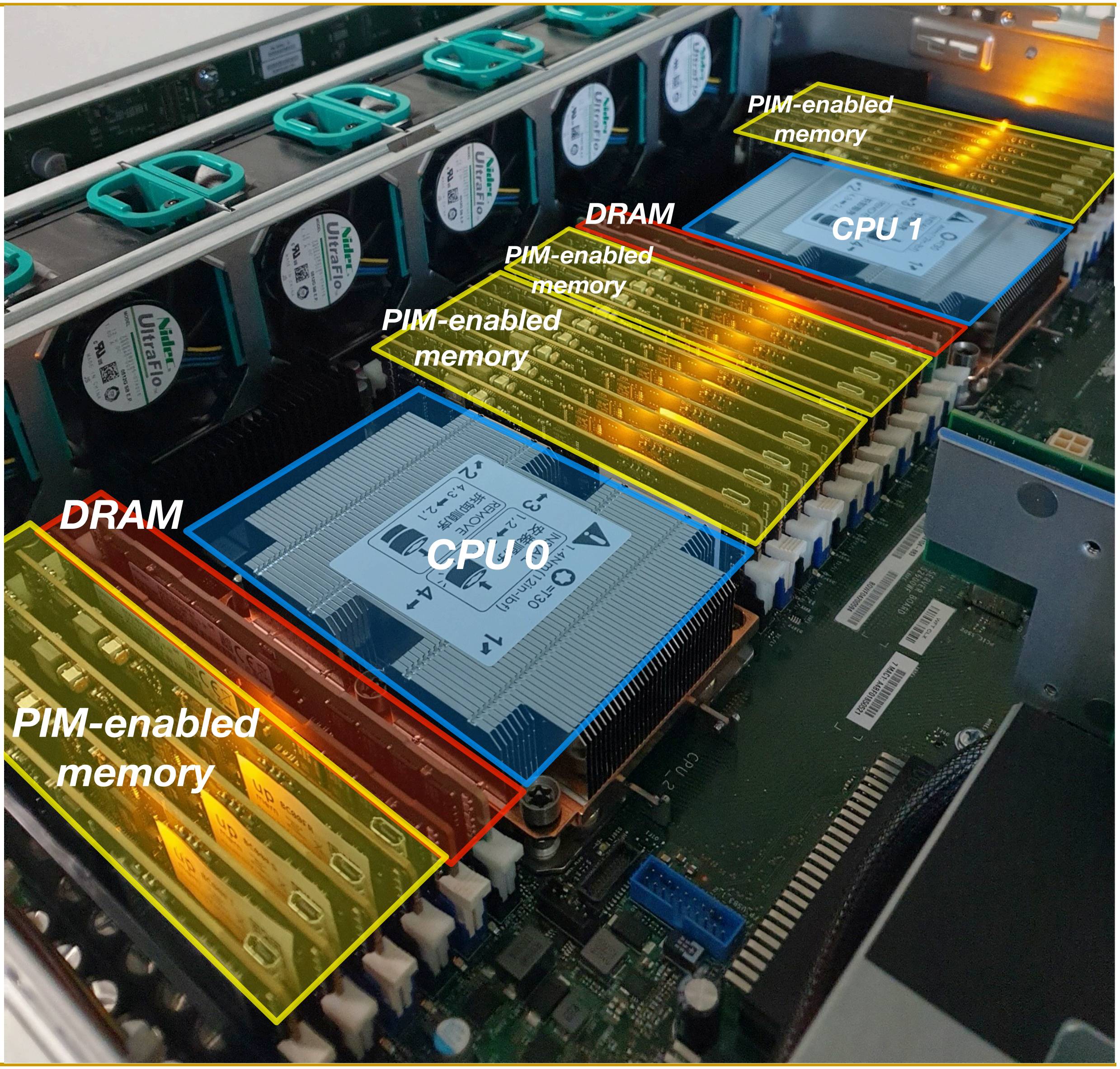
- E19: 8 chips DIMM (1 rank). DPUs @ 267 MHz
- P21: 16 chips DIMM (2 ranks). DPUs @ 350 MHz



www.upmem.com

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

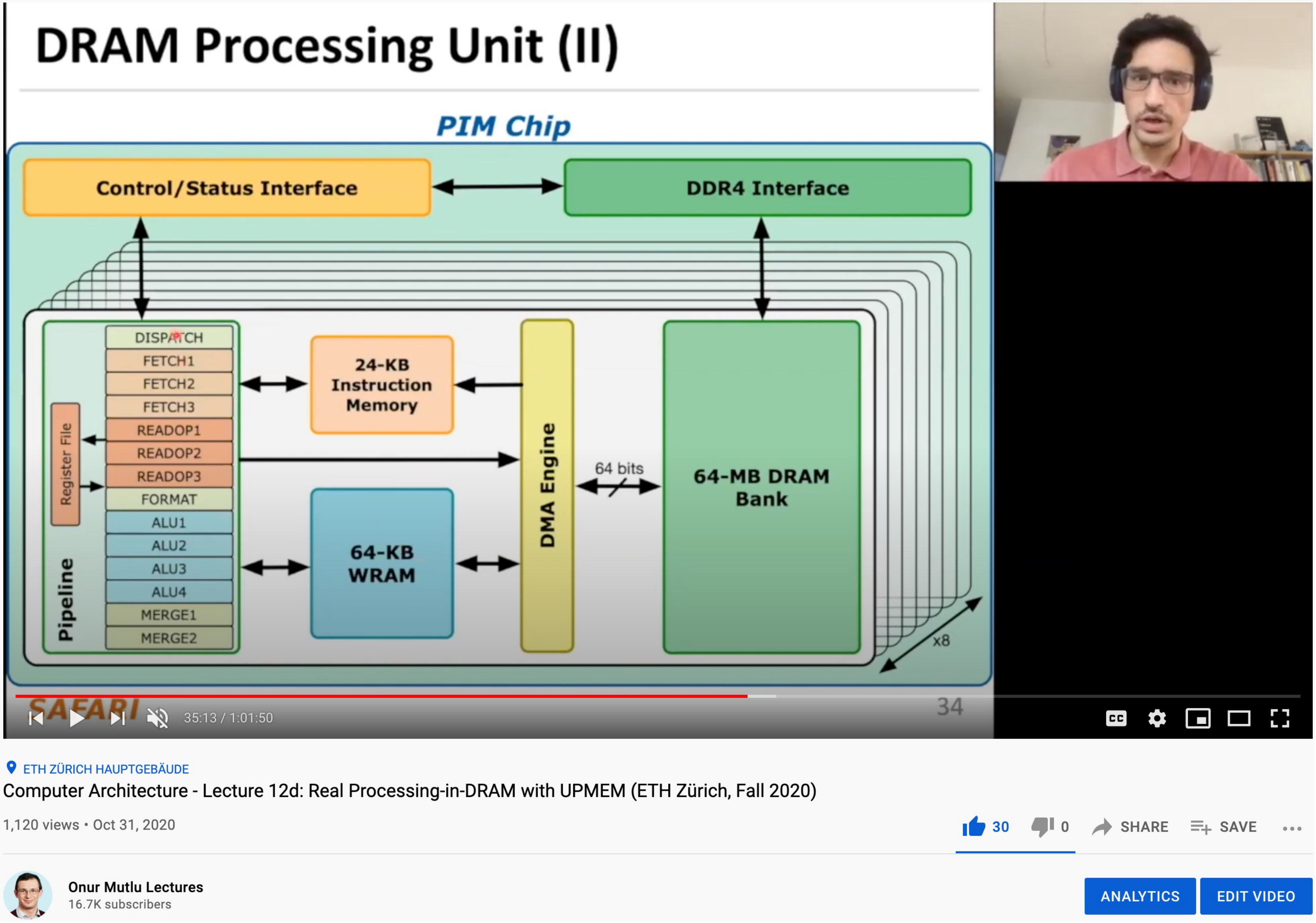
Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens 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 2,556 DPUs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.

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

More on the UPMEM PIM System



ETH ZÜRICH HAUPTGEBÄUDE

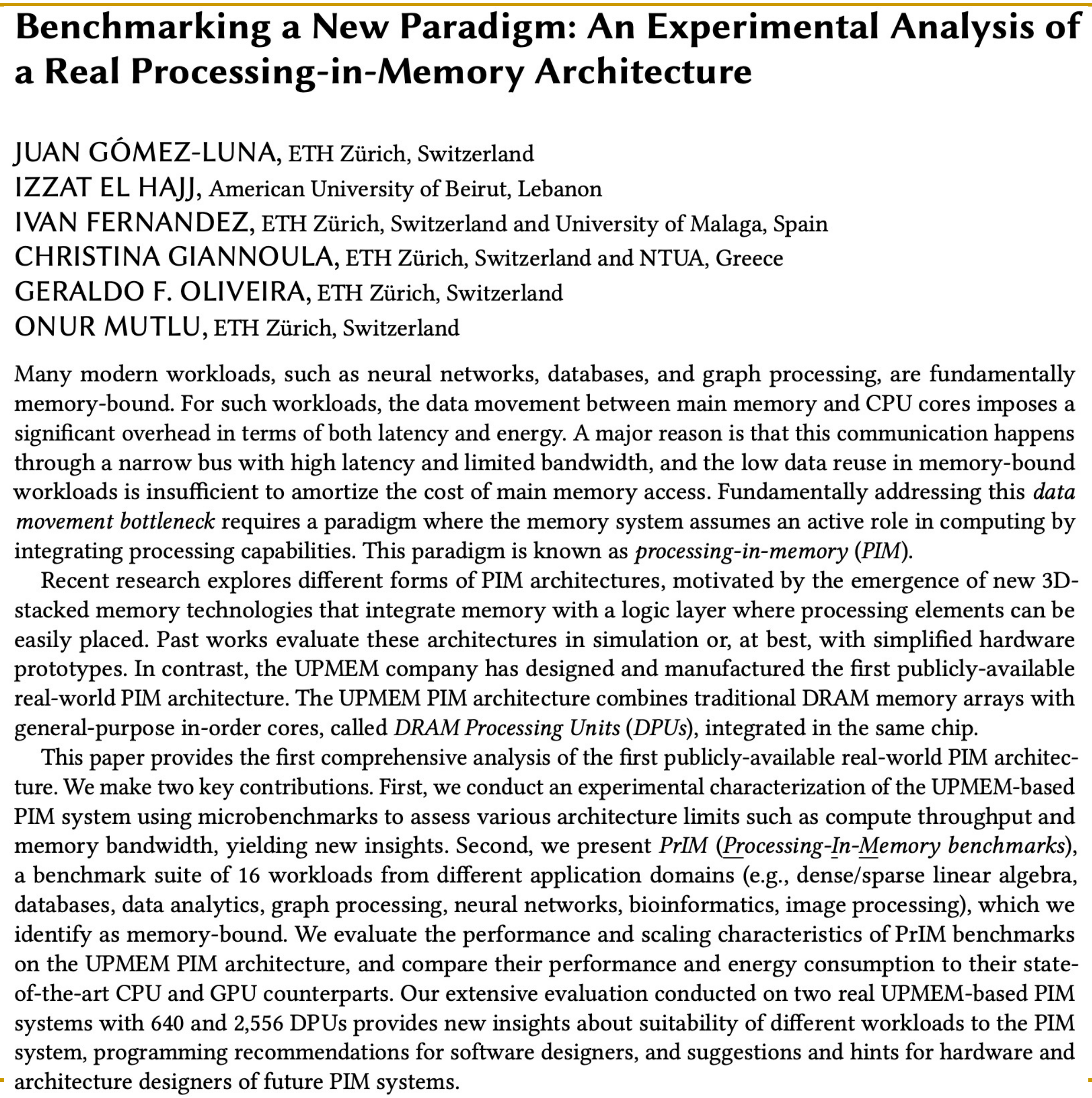
Computer Architecture - Lecture 12d: Real Processing-in-DRAM with UPMEM (ETH Zürich, Fall 2020)

1,120 views • Oct 31, 2020

Onur Mutlu Lectures
16.7K subscribers

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

Experimental Analysis of the UPMEM PIM Engine



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

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

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

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

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

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

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** (Processing-In-Memory) 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

Upcoming TECHCON Presentation

- Dr. Juan Gomez-Luna
  - Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware
  - Based on two major works
    - https://arxiv.org/pdf/2105.03814.pdf
    - https://arxiv.org/pdf/2207.07886.pdf

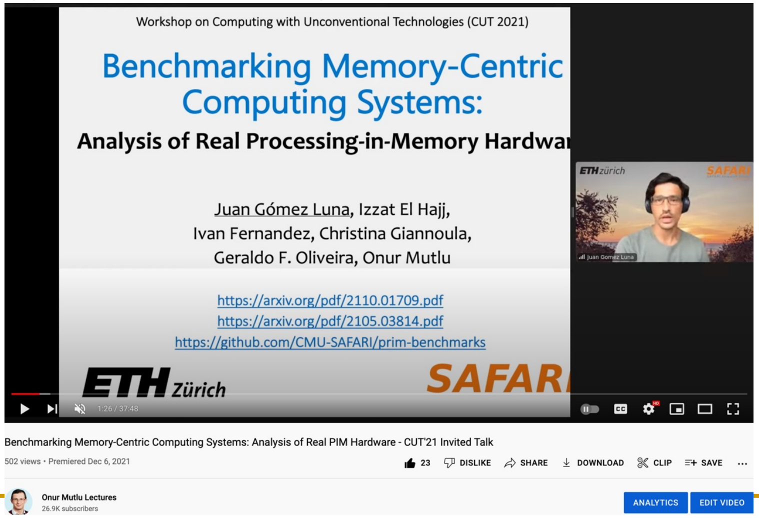
Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-In-Memory Hardware

Year: 2021, Pages: 1-7

DOI Bookmark: 10.1109/IGSC54211.2021.9651614

Authors

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



Workshop on Computing with Unconventional Technologies (CUT 2021)

Benchmarking Memory-Centric Computing Systems:

Analysis of Real Processing-in-Memory Hardwar

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

https://arxiv.org/pdf/2110.01709.pdf
https://arxiv.org/pdf/2105.03814.pdf
https://github.com/CMU-SAFARI/prim-benchmarks

Benchmarking Memory-Centric Computing Systems: Analysis of Real PIM Hardware - CUT'21 Invited Talk

502 views • Premiered Dec 6, 2021

Onur Mutlu Lectures
26.9K subscribers

https://www.youtube.com/watch?v=nphV36SrysA

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**.

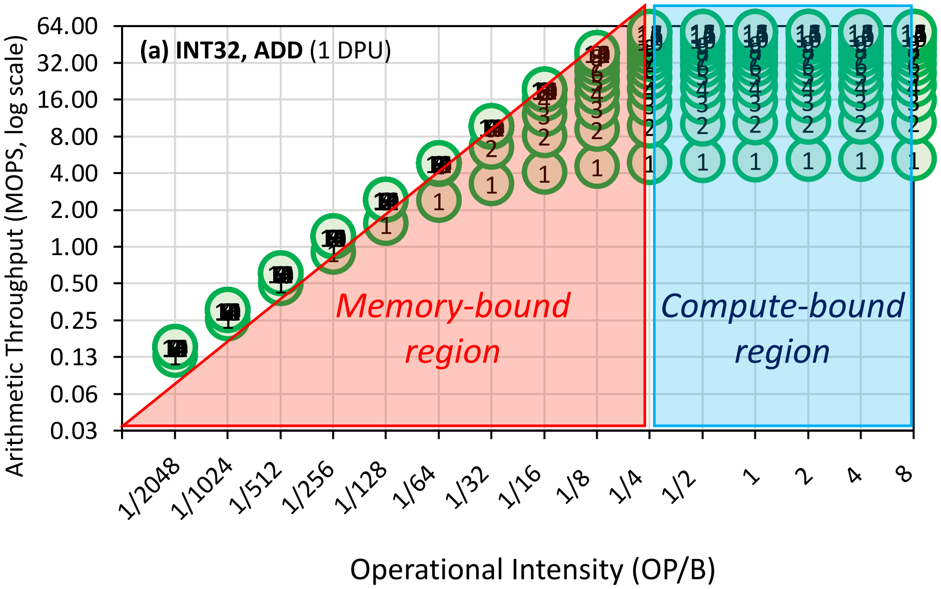
KEY TAKEAWAY 1

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 ¼ OP/B, i.e., 1 integer addition per every 32-bit element fetched

KEY TAKEAWAY 1

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

Key Takeaway 2

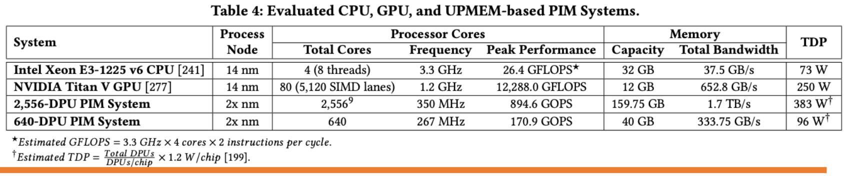
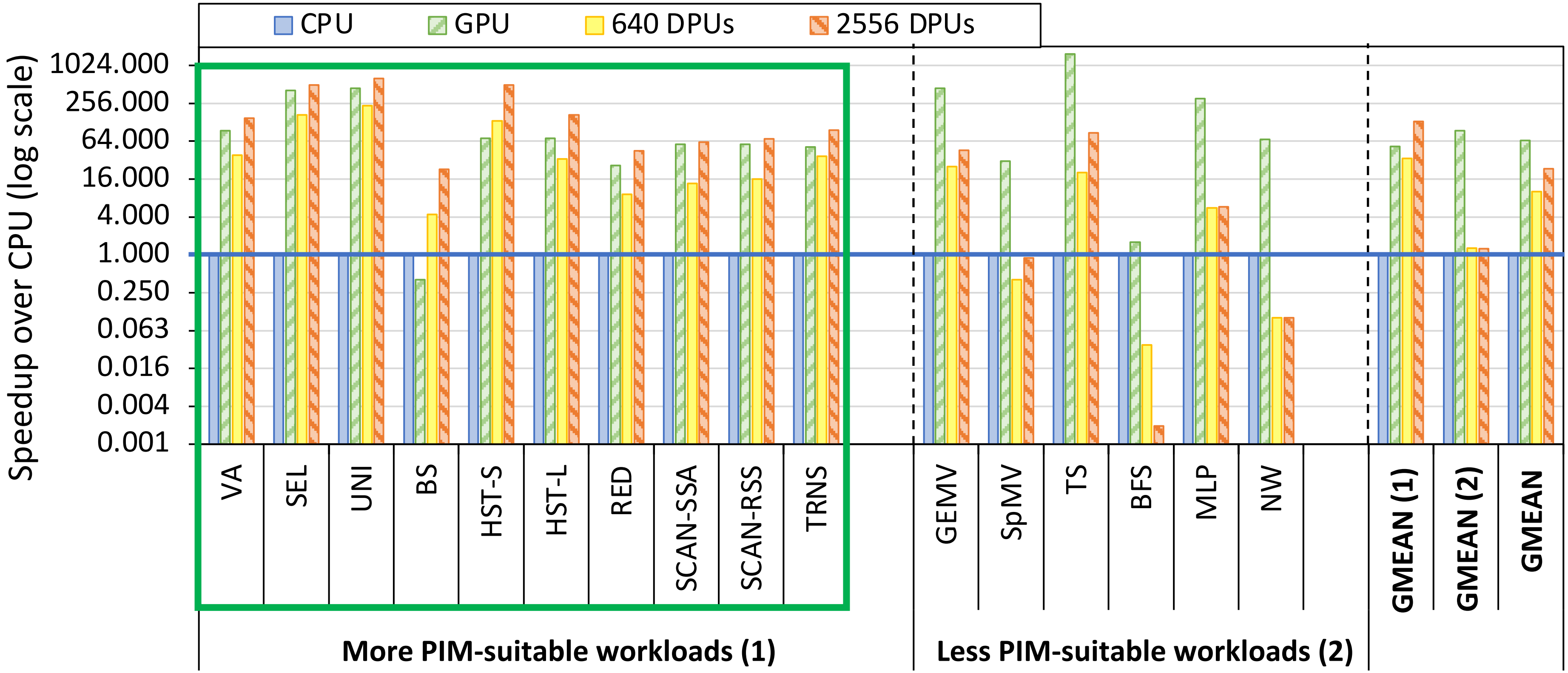


Table 4: Evaluated CPU, GPU, and UPMEM-based PIM Systems.

System	Process Node	Processor Cores			Memory		TDP
		Total Cores	Frequency	Peak Performance	Capacity	Total Bandwidth	
Intel Xeon E3-1225 v6 CPU [241]	14 nm	4 (8 threads)	3.3 GHz	26.4 GFLOPS★	32 GB	37.5 GB/s	73 W
NVIDIA Titan V GPU [277]	14 nm	80 (5,120 SIMD lanes)	1.2 GHz	12,288.0 GFLOPS	12 GB	652.8 GB/s	250 W
2,556-DPU PIM System	2x nm	2,556[9]	350 MHz	894.6 GOPS	159.75 GB	1.7 TB/s	383 W†
640-DPU PIM System	2x nm	640	267 MHz	170.9 GOPS	40 GB	333.75 GB/s	96 W†

★*Estimated GFLOPS = 3.3 GHz × 4 cores × 2 instructions per cycle.*

†*Estimated TDP* = $\frac{Total\ DPUs}{DPUs/chip} \times 1.2\ W/chip$ [199].



KEY TAKEAWAY 2

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

Key Takeaway 3

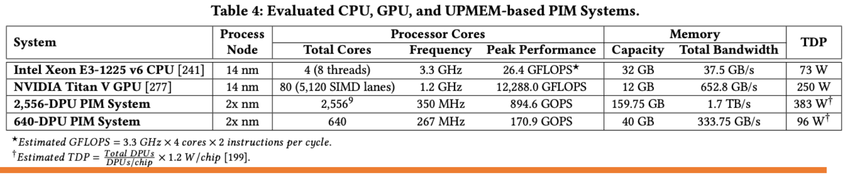
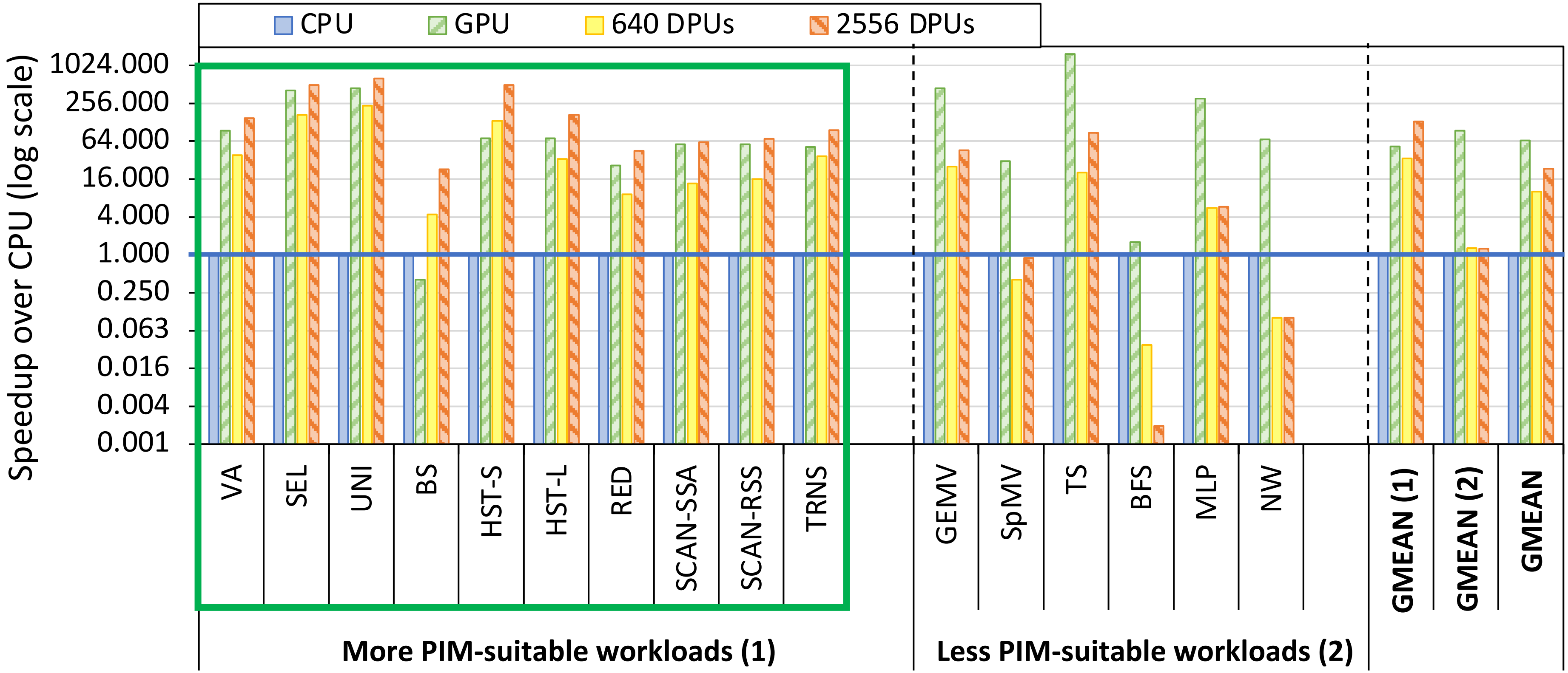


Table 4: Evaluated CPU, GPU, and UPMEM-based PIM Systems.

System	Process Node	Processor Cores			Memory		TDP
		Total Cores	Frequency	Peak Performance	Capacity	Total Bandwidth	
Intel Xeon E3-1225 v6 CPU [241]	14 nm	4 (8 threads)	3.3 GHz	26.4 GFLOPS★	32 GB	37.5 GB/s	73 W
NVIDIA Titan V GPU [277]	14 nm	80 (5,120 SIMD lanes)	1.2 GHz	12,288.0 GFLOPS	12 GB	652.8 GB/s	250 W
2,556-DPU PIM System	2x nm	$2{,}556^9$	350 MHz	894.6 GOPS	159.75 GB	1.7 TB/s	383 W†
640-DPU PIM System	2x nm	640	267 MHz	170.9 GOPS	40 GB	333.75 GB/s	96 W†

★*Estimated GFLOPS = 3.3 GHz × 4 cores × 2 instructions per cycle.*

†*Estimated TDP =* $\frac{Total\ DPUs}{DPUs/chip} \times 1.2\ W/chip$ [199].



KEY TAKEAWAY 3

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

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 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

ETH Zürich

SAFARI

UPMEM PIM System Summary & Analysis

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

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

Juan Gómez-Luna
ETH Zürich

Izzat El Hajj
American University of Beirut

Ivan Fernandez
University of Malaga

Christina Giannoula
National Technical University of Athens

Geraldo F. Oliveira
ETH Zürich

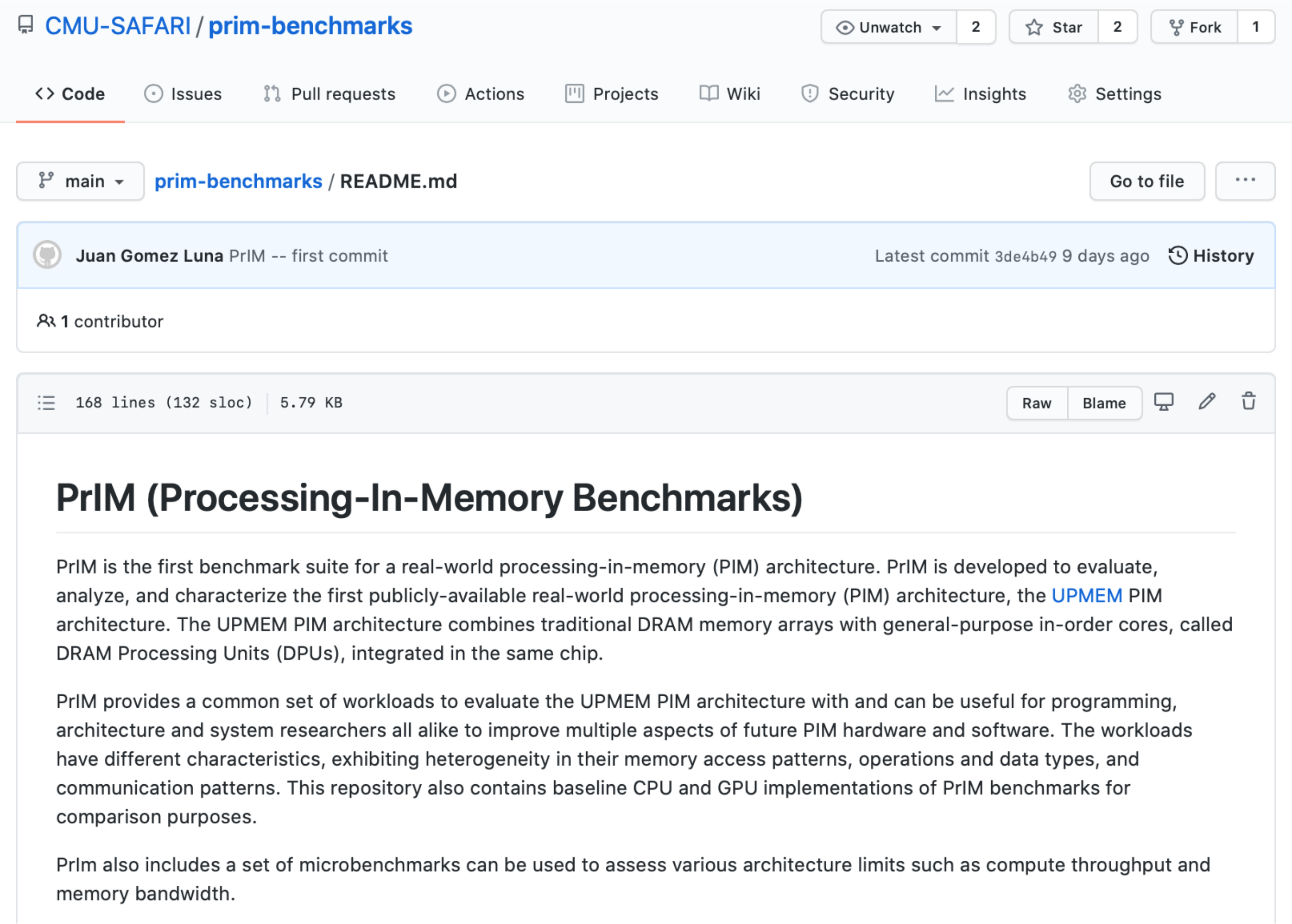
Onur Mutlu
ETH Zürich

PrIM Benchmarks: Application Domains

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

PrIM Benchmarks are Open Source

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



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

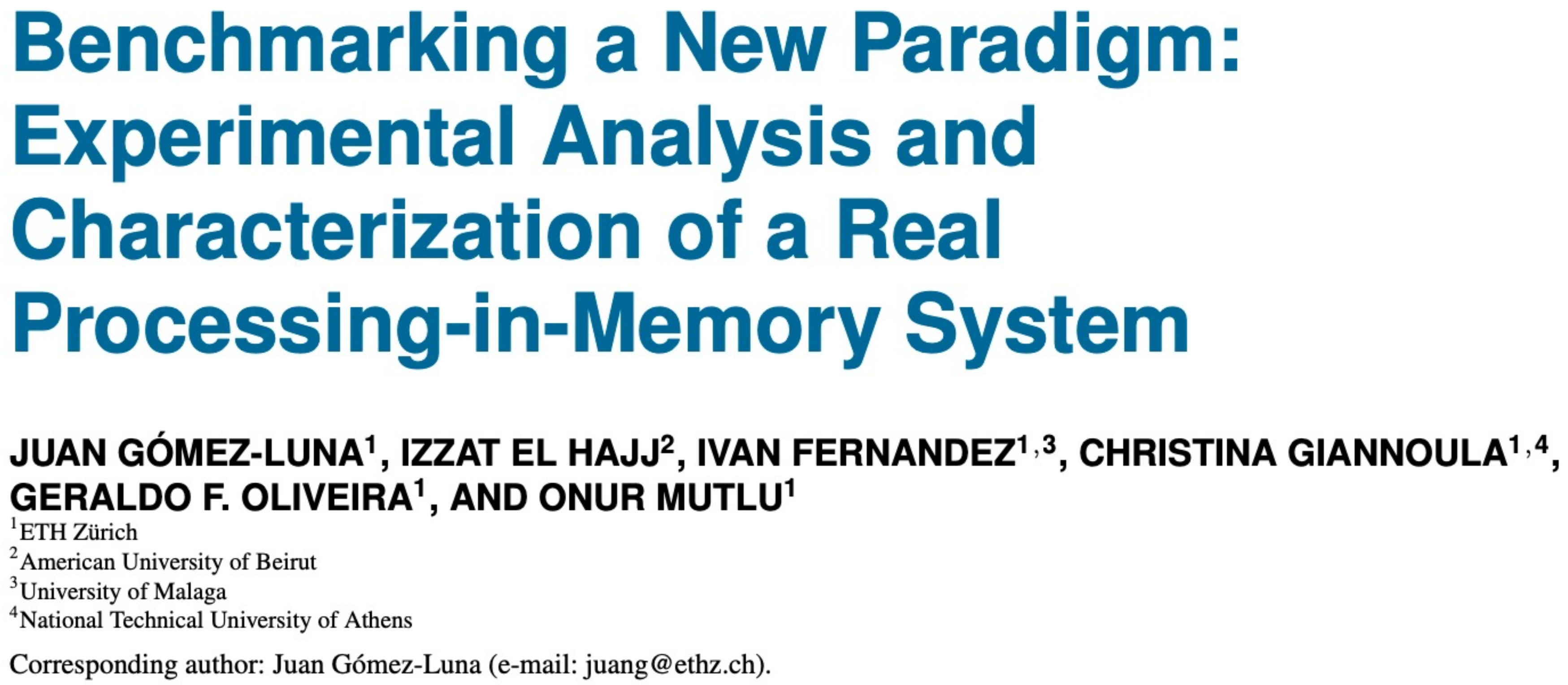
PrIM (Processing-In-Memory Benchmarks)

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.

Prlm 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-LUNA[1], IZZAT EL HAJJ[2], IVAN FERNANDEZ[1,3], CHRISTINA GIANNOULA[1,4], GERALDO F. OLIVEIRA[1], AND ONUR MUTLU[1]

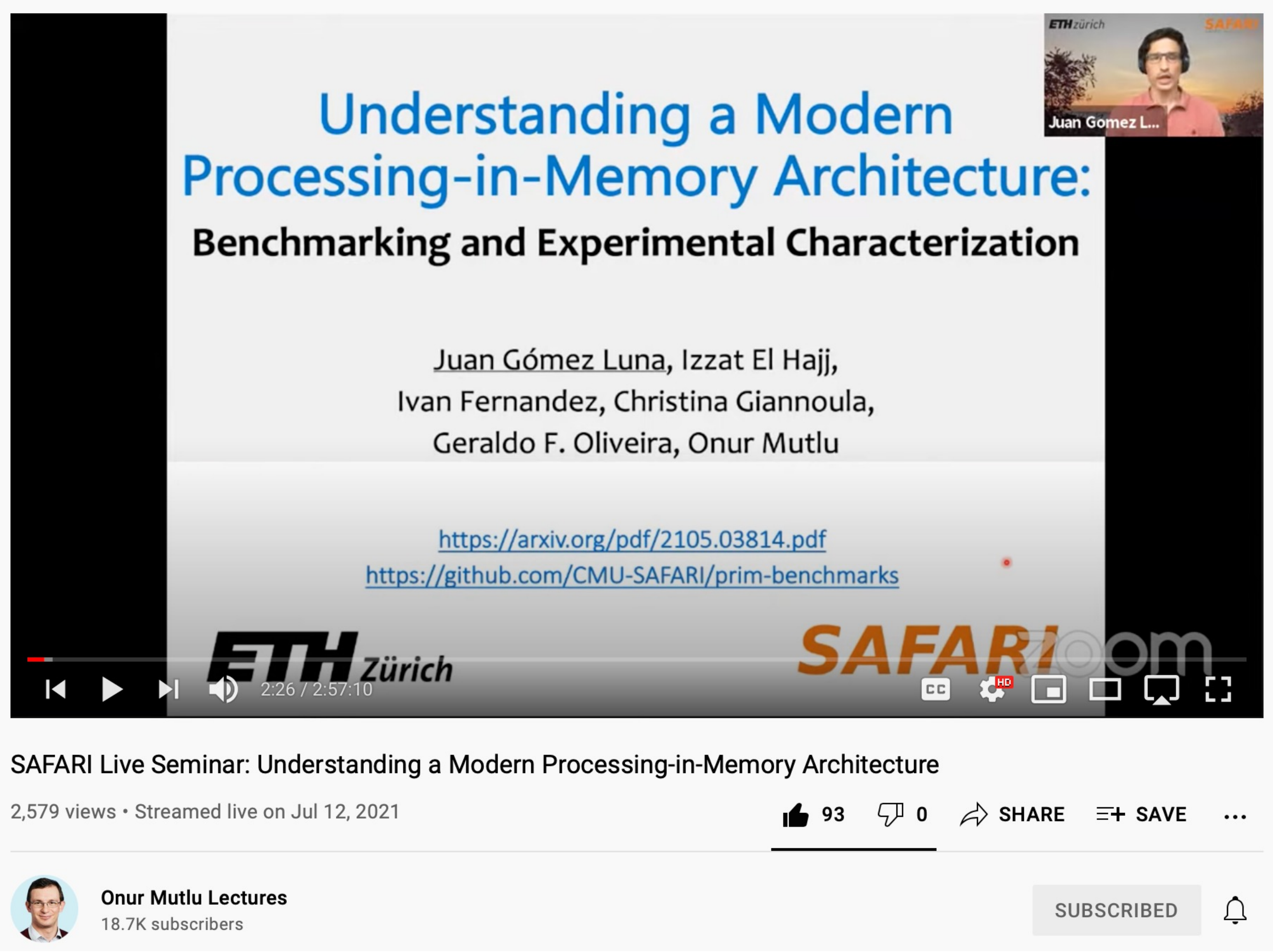
[1]ETH Zürich
[2]American University of Beirut
[3]University of Malaga
[4]National Technical University of Athens

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

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

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

Understanding a Modern PIM Architecture



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

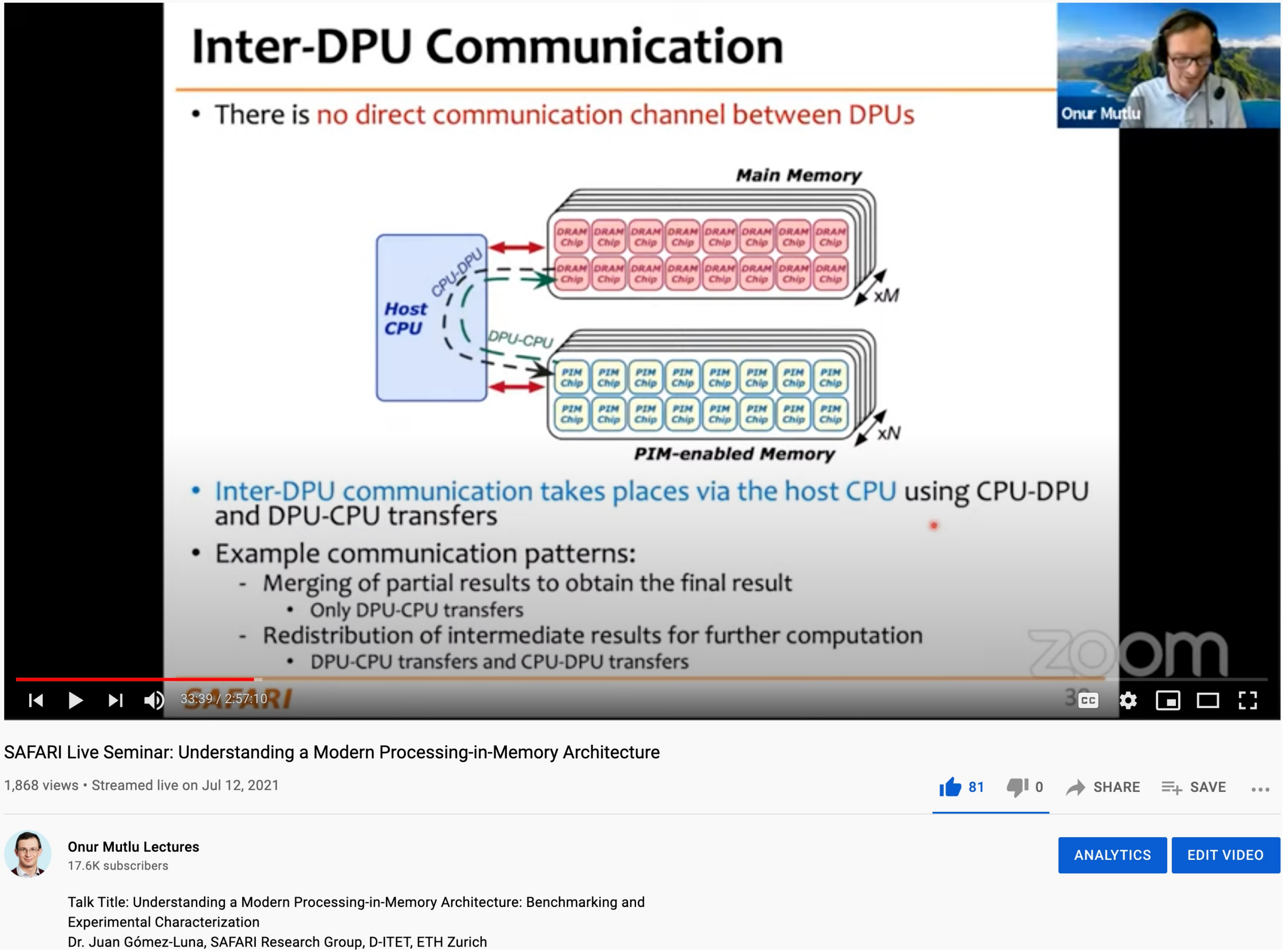
SAFARI Live Seminar: Understanding a Modern Processing-in-Memory Architecture

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

Onur Mutlu Lectures
18.7K subscribers

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

More on Analysis of the UPMEM PIM Engine



Inter-DPU Communication

- There is no direct communication channel between DPUs
- Inter-DPU communication takes places 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

Onur Mutlu Lectures

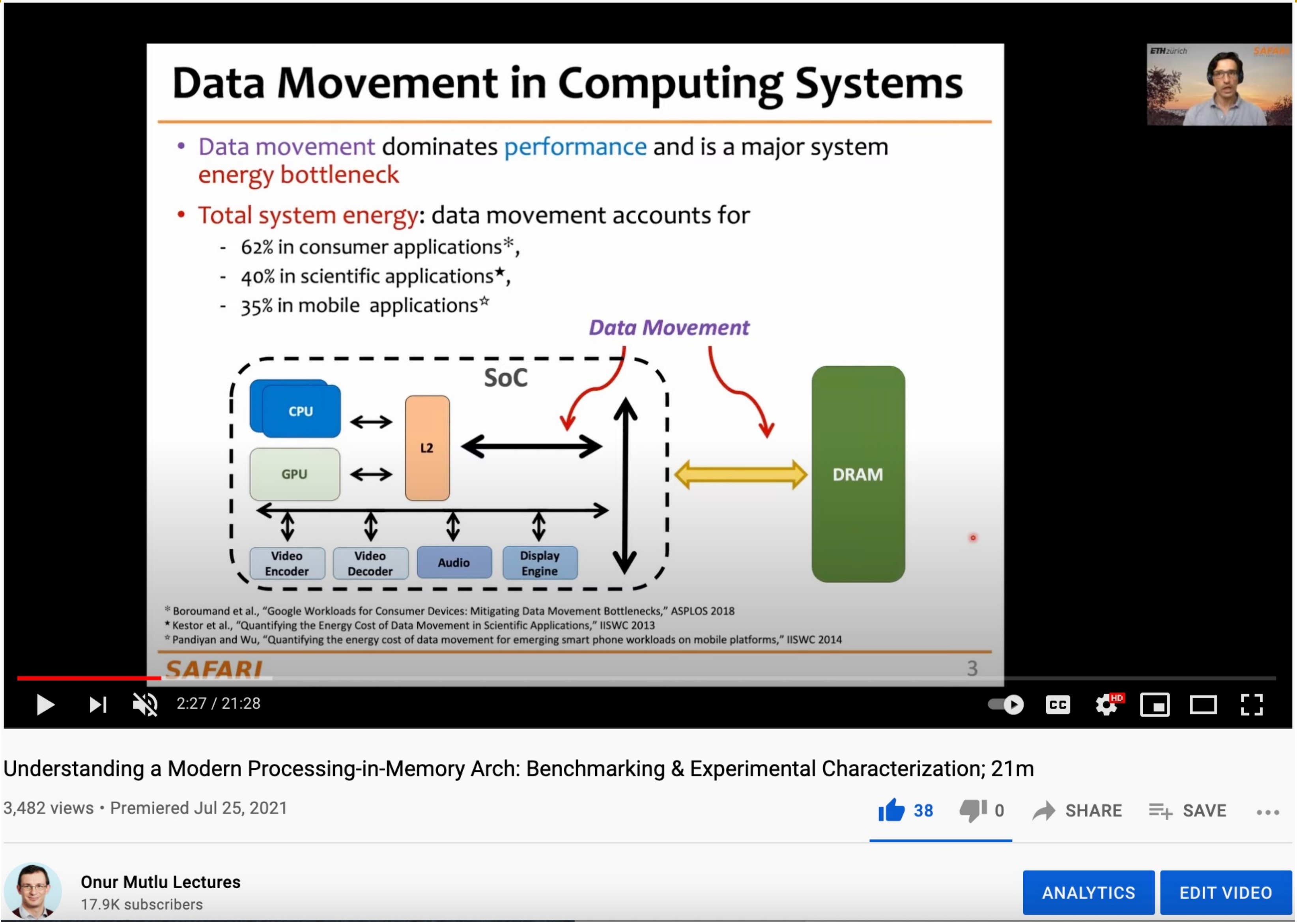
17.6K subscribers

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

Dr. Juan Gómez-Luna, SAFARI Research Group, D-ITET, ETH Zurich

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☆

Data Movement

SoC

CPU

GPU

L2

Video Encoder

Video Decoder

Audio

Display Engine

DRAM

* 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

Onur Mutlu Lectures
17.9K subscribers

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)]

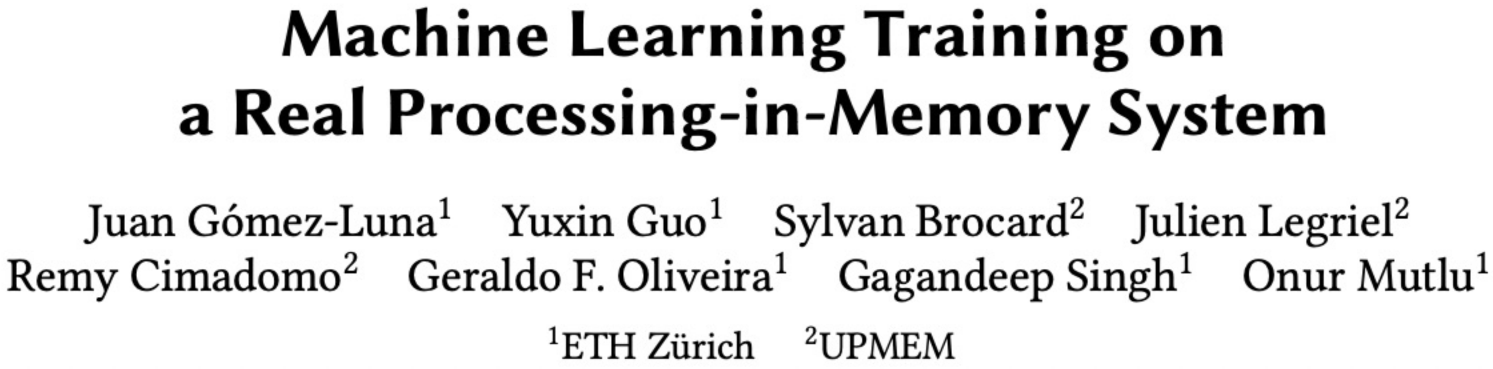
Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware

Juan Gómez-Luna, *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*

Year III Results (2022 Annual Review 1)

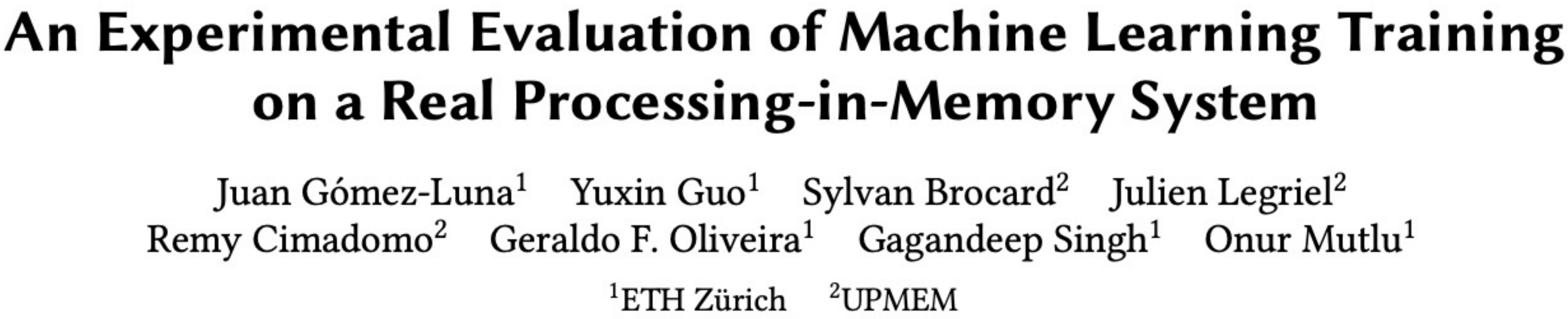
- Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System [IEEE Access'22]
- Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware [CUT 2021]
- An Experimental Evaluation of Machine Learning Training on a Real Processing-in-Memory System [arXiv 2022]
- SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures [SIGMETRICS 2022]
- High-throughput Pairwise Alignment with the Wavefront Algorithm using Processing-in-Memory [HICOMB 2022]
- PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM [arXiv 2021]

ML Training on a Real PIM System



Machine Learning Training on a Real Processing-in-Memory System

Juan Gómez-Luna[1] Yuxin Guo[1] Sylvan Brocard[2] Julien Legriel[2]
Remy Cimadomo[2] Geraldo F. Oliveira[1] Gagandeep Singh[1] Onur Mutlu[1]
[1]ETH Zürich [2]UPMEM



An Experimental Evaluation of Machine Learning Training on a Real Processing-in-Memory System

Juan Gómez-Luna[1] Yuxin Guo[1] Sylvan Brocard[2] Julien Legriel[2]
Remy Cimadomo[2] Geraldo F. Oliveira[1] Gagandeep Singh[1] Onur Mutlu[1]
[1]ETH Zürich [2]UPMEM

Short version: https://arxiv.org/pdf/2206.06022.pdf

Long version: https://arxiv.org/pdf/2207.07886.pdf

https://www.youtube.com/watch?v=qeukNs5XI3g&t=11226s

Machine Learning Training

on a Real Processing-in-Memory System

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

Short version: https://arxiv.org/pdf/2206.06022.pdf

Long version: https://arxiv.org/pdf/2207.07886.pdf

https://www.youtube.com/watch?v=qeukNs5XI3g&t=11226s

ETH Zürich

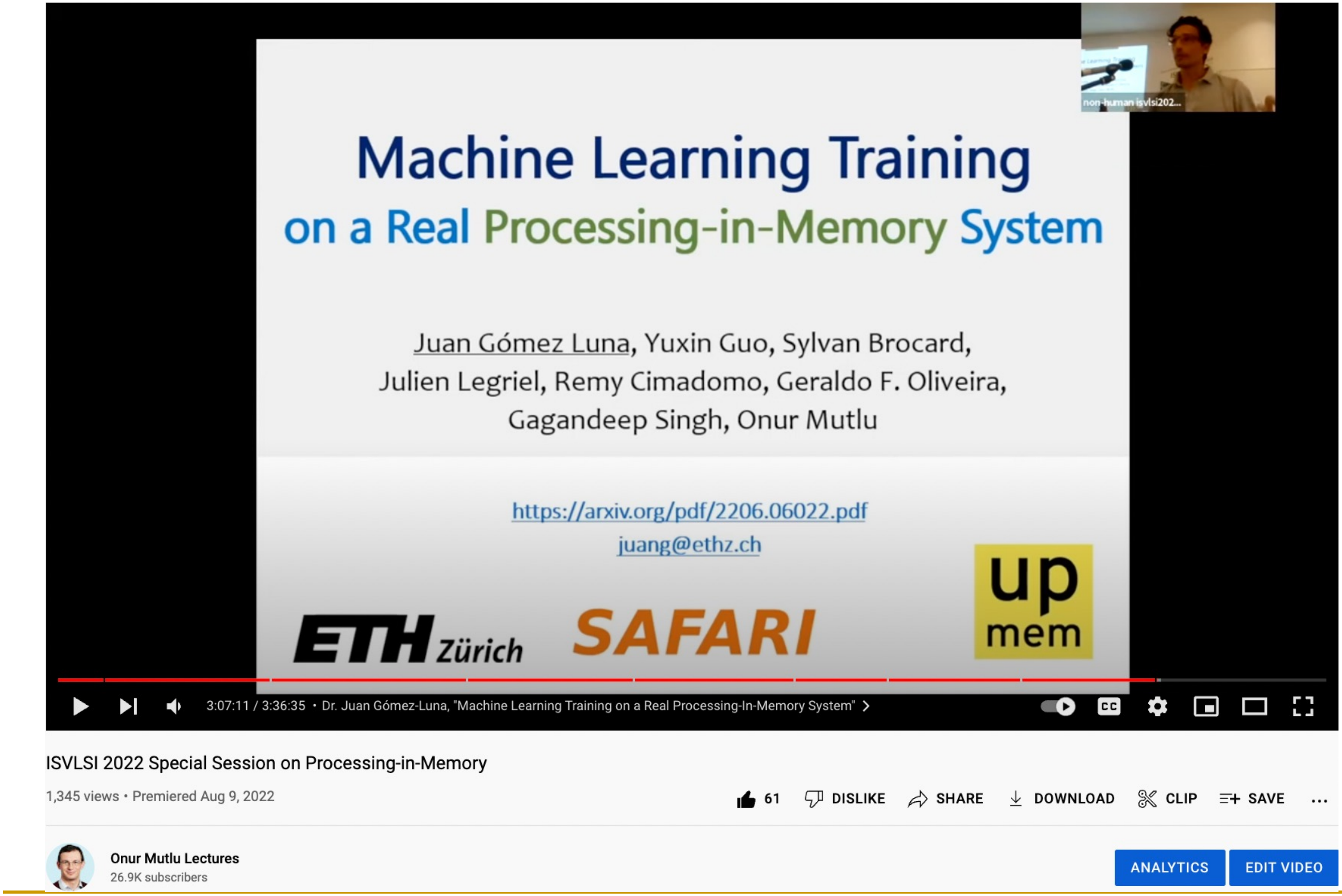
SAFARI

up
mem

Executive Summary

- Training machine learning (ML) algorithms is a computationally expensive process, frequently memory-bound due to repeatedly accessing large training datasets
- Memory-centric computing systems, i.e., with Processing-in-Memory (PIM) capabilities, can alleviate this *data movement bottleneck*
- Real-world PIM systems have only recently been manufactured and commercialized
  - UPMEM has designed and fabricated the first publicly-available real-world PIM architecture
- Our goal is to understand the potential of modern general-purpose PIM architectures to accelerate machine learning training
- Our main contributions:
  - PIM implementation of several classical machine learning algorithms: linear regression, logistic regression, decision tree, K-means clustering
  - Workload characterization in terms of accuracy, performance, and scaling
  - Comparison to their counterpart implementations on processor-centric systems (CPU and GPU)
- Experimental evaluation on a real-world PIM system with 2,524 PIM cores @ 425 MHz and 158 GB of DRAM memory
- New observations and insights:
  - ML training in PIM systems benefits from (1) fixed-point representation, (2) quantization, and (3) hybrid precision implementations
  - Complex activation functions (e.g., sigmoid) can take advantage of LUTs in PIM systems without native support for those activation functions
  - Data can be placed and laid out for PIM cores to access nearby memory banks in streaming, thus maximizing PIM memory bandwidth
  - ML training benefits from scaling the size of PIM-enabled memory with PIM cores attached to memory banks

ML Training on Real PIM Talk Video



Machine Learning Training

on a Real Processing-in-Memory System

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

https://arxiv.org/pdf/2206.06022.pdf

juang@ethz.ch

non-human isvlsi202...
ETH Zürich
SAFARI
up mem
3:07:11 / 3:36:35 • Dr. Juan Gómez-Luna, "Machine Learning Training on a Real Processing-In-Memory System" >
CC

ISVLSI 2022 Special Session on Processing-in-Memory

1,345 views • Premiered Aug 9, 2022

https://www.youtube.com/watch?v=qeukNs5XI3g&t=11226s

Outline

Machine learning workloads

Processing-in-memory

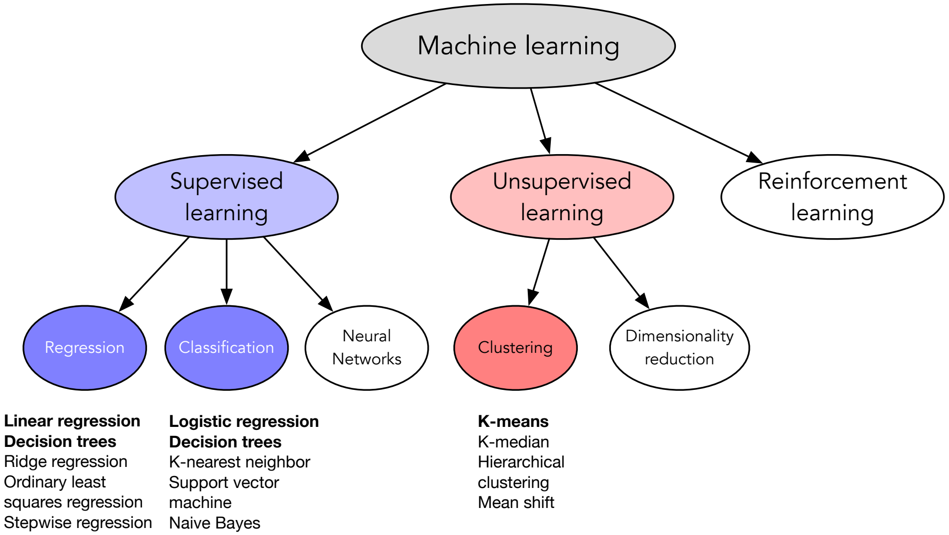
PIM implementation of ML workloads

Evaluation

Key observations and insights

Machine Learning Workloads

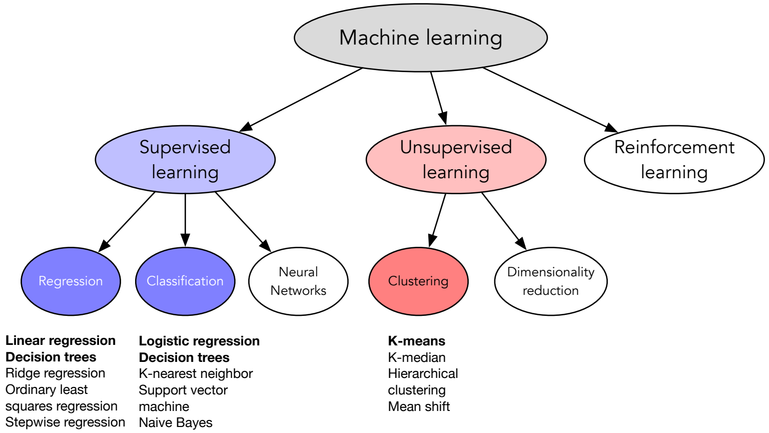
- Machine learning training with large amounts of data is a computationally expensive process, which requires many iterations to update an ML model's parameters



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

Machine Learning Workloads: Our Goal

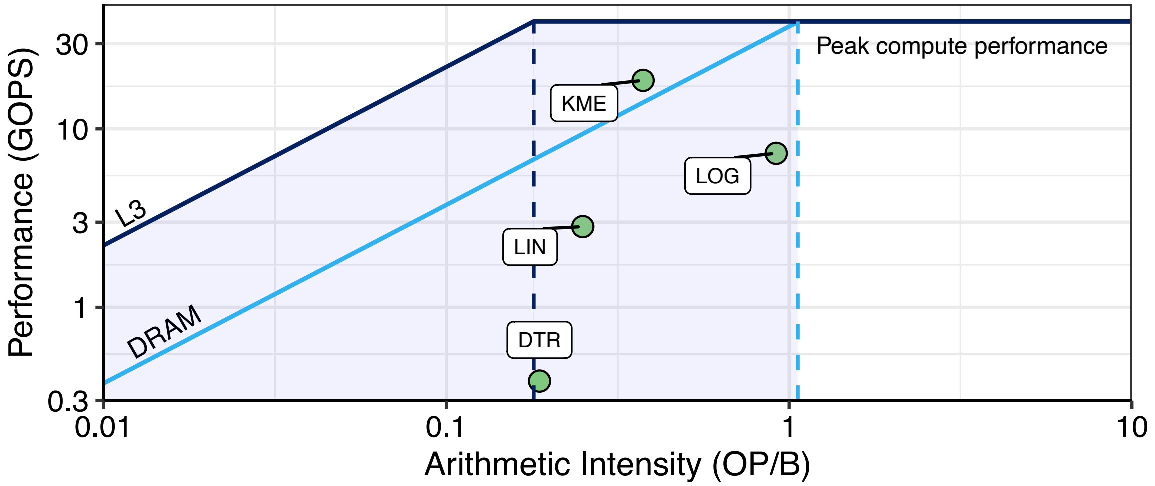
- Our goal is to study and analyze how real-world general-purpose PIM can accelerate ML training
- Four representative ML algorithms: linear regression, logistic regression, decision tree, K-means
- Roofline model to quantify the memory boundedness of CPU versions of the four workloads



Linear regression
Decision trees
Ridge regression
Ordinary least squares regression
Stepwise regression

Logistic regression
Decision trees
K-nearest neighbor
Support vector machine
Naive Bayes

K-means
K-median
Hierarchical clustering
Mean shift

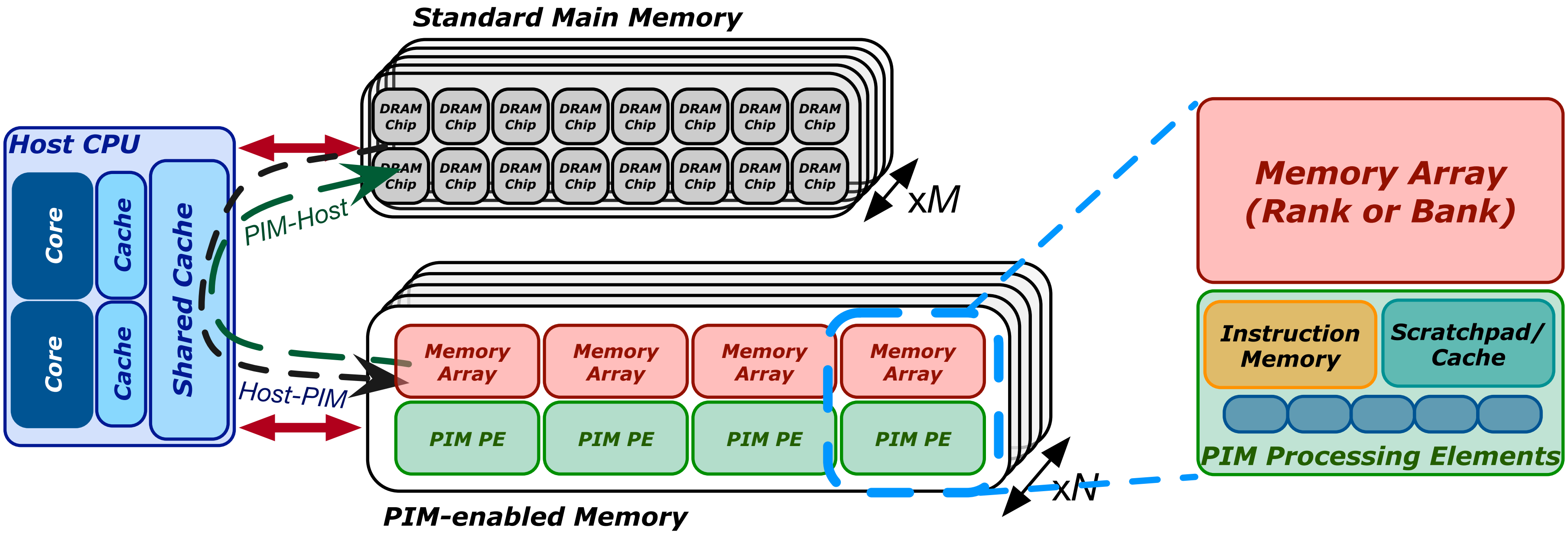


All workloads fall in the memory-bound area of the Roofline

Processing-in-Memory (PIM)

- PIM is a computing paradigm that advocates for memory-centric computing systems, where processing elements are placed near or inside the memory arrays
- Real-world PIM architectures are becoming a reality
  - UPMEM PIM, Samsung HBM-PIM, Samsung AxDIMM, SK Hynix AiM, Alibaba HB-PNM
- These PIM systems have some common characteristics:
  1. There is a host processor (CPU or GPU) with access to (1) standard main memory, and (2) PIM-enabled memory
  2. PIM-enabled memory contains multiple PIM processing elements (PEs) with high bandwidth and low latency memory access
  3. PIM PEs run only at a few hundred MHz and have a small number of registers and small (or no) cache/scratchpad
  4. PEs may need to communicate via the host processor

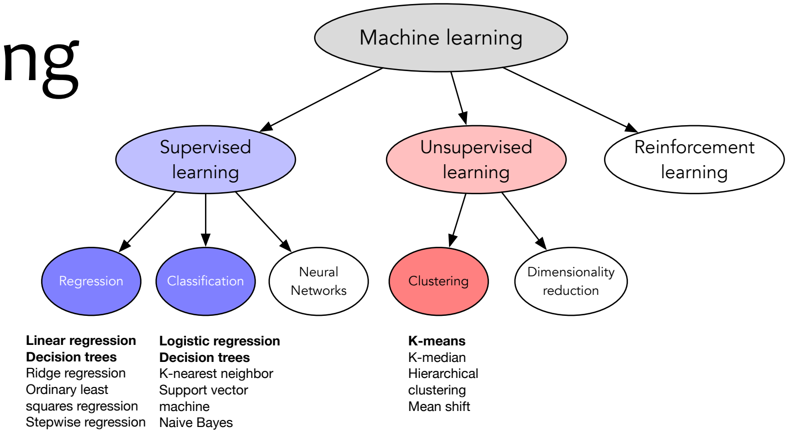
A State-of-the-Art PIM System



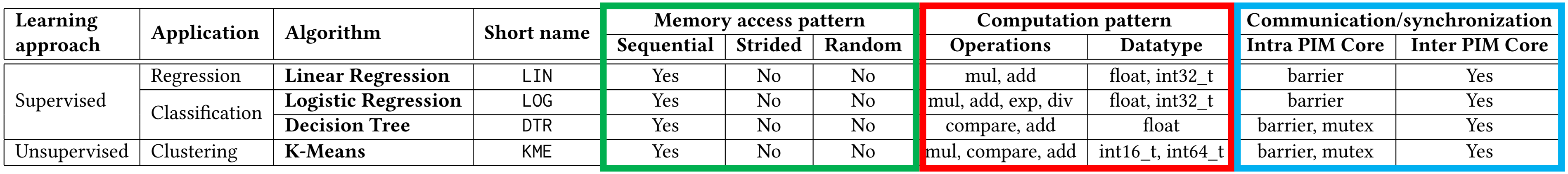
- In our work, we use the UPMEM PIM architecture
  - General-purpose processing cores called *DRAM Processing Units (DPUs)*
    - Up to 24 PIM threads, called *tasklets*
    - 32-bit integer arithmetic, but multiplication/division are emulated, as well as floating-point operations
  - 64-MB DRAM bank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

ML Training Workloads

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



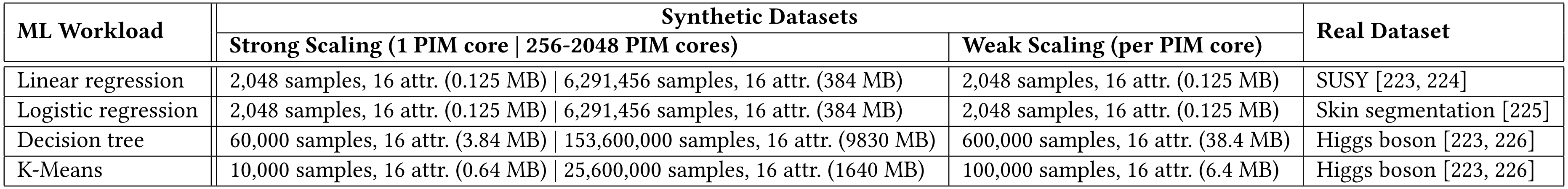
- Diversity of our ML training workloads:
  - Memory access patterns
  - Operations and datatypes
  - Communication/synchronization



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

Evaluation Methodology

- Synthetic and real datasets



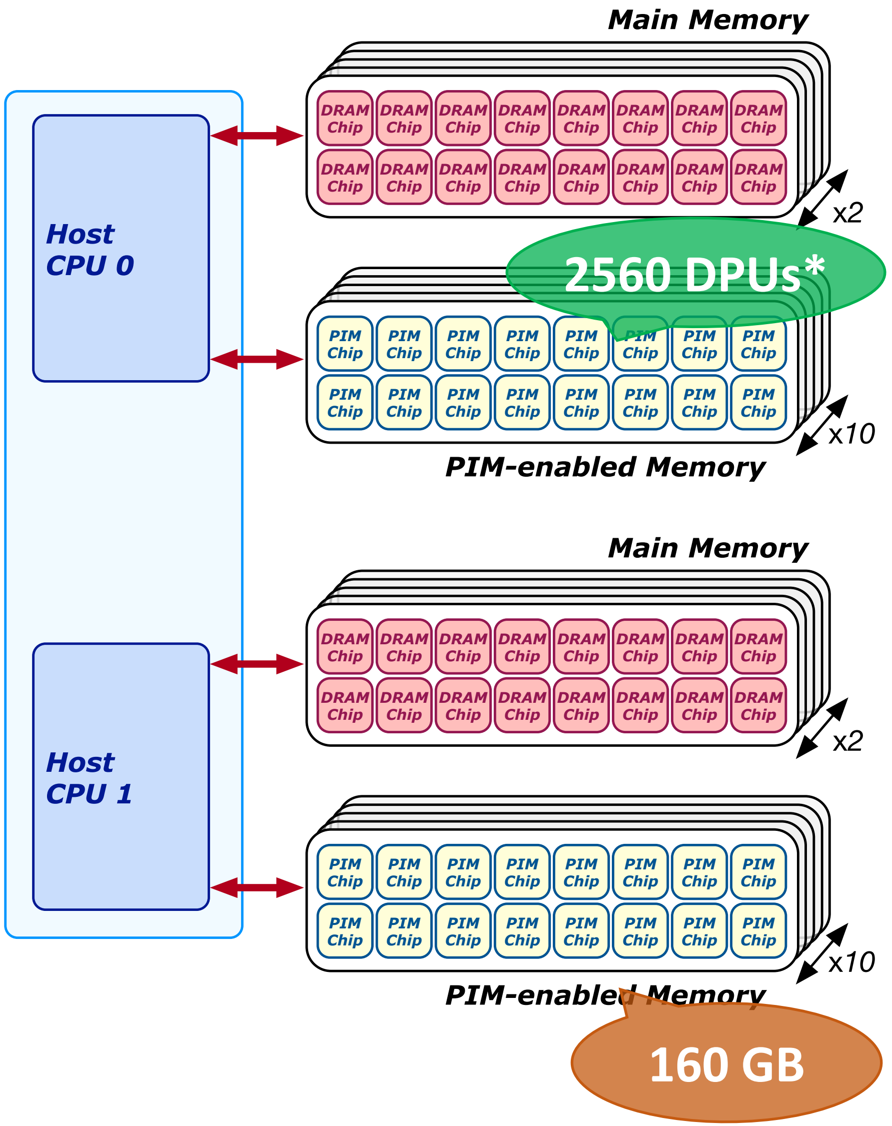
ML Workload	Synthetic Datasets		Real Dataset
	Strong Scaling (1 PIM core | 256-2048 PIM cores)	**Weak Scaling (per PIM core)**	
Linear regression	2,048 samples, 16 attr. (0.125 MB) | 6,291,456 samples, 16 attr. (384 MB)	2,048 samples, 16 attr. (0.125 MB)	SUSY [223, 224]
Logistic regression	2,048 samples, 16 attr. (0.125 MB) | 6,291,456 samples, 16 attr. (384 MB)	2,048 samples, 16 attr. (0.125 MB)	Skin segmentation [225]
Decision tree	60,000 samples, 16 attr. (3.84 MB) | 153,600,000 samples, 16 attr. (9830 MB)	600,000 samples, 16 attr. (38.4 MB)	Higgs boson [223, 226]
K-Means	10,000 samples, 16 attr. (0.64 MB) | 25,600,000 samples, 16 attr. (1640 MB)	100,000 samples, 16 attr. (6.4 MB)	Higgs boson [223, 226]

- Evaluated systems
  - UPMEM PIM system with 2,524 PIM cores @ 425 MHz and 158 GB of DRAM
  - Intel Xeon Silver 4215 CPU (16 hardware threads)
  - NVIDIA A100 GPU

- We evaluate:
  - Metrics
  - Performance of PIM kernels
  - Performance scaling
  - Comparison to CPU and GPU

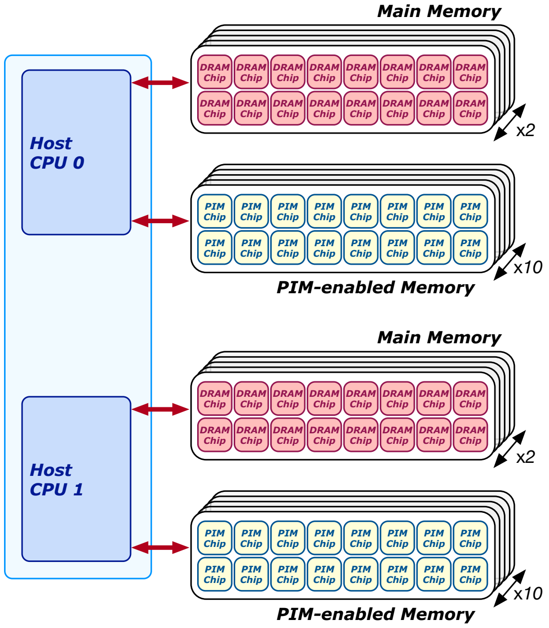
2,560-DPU System (I)

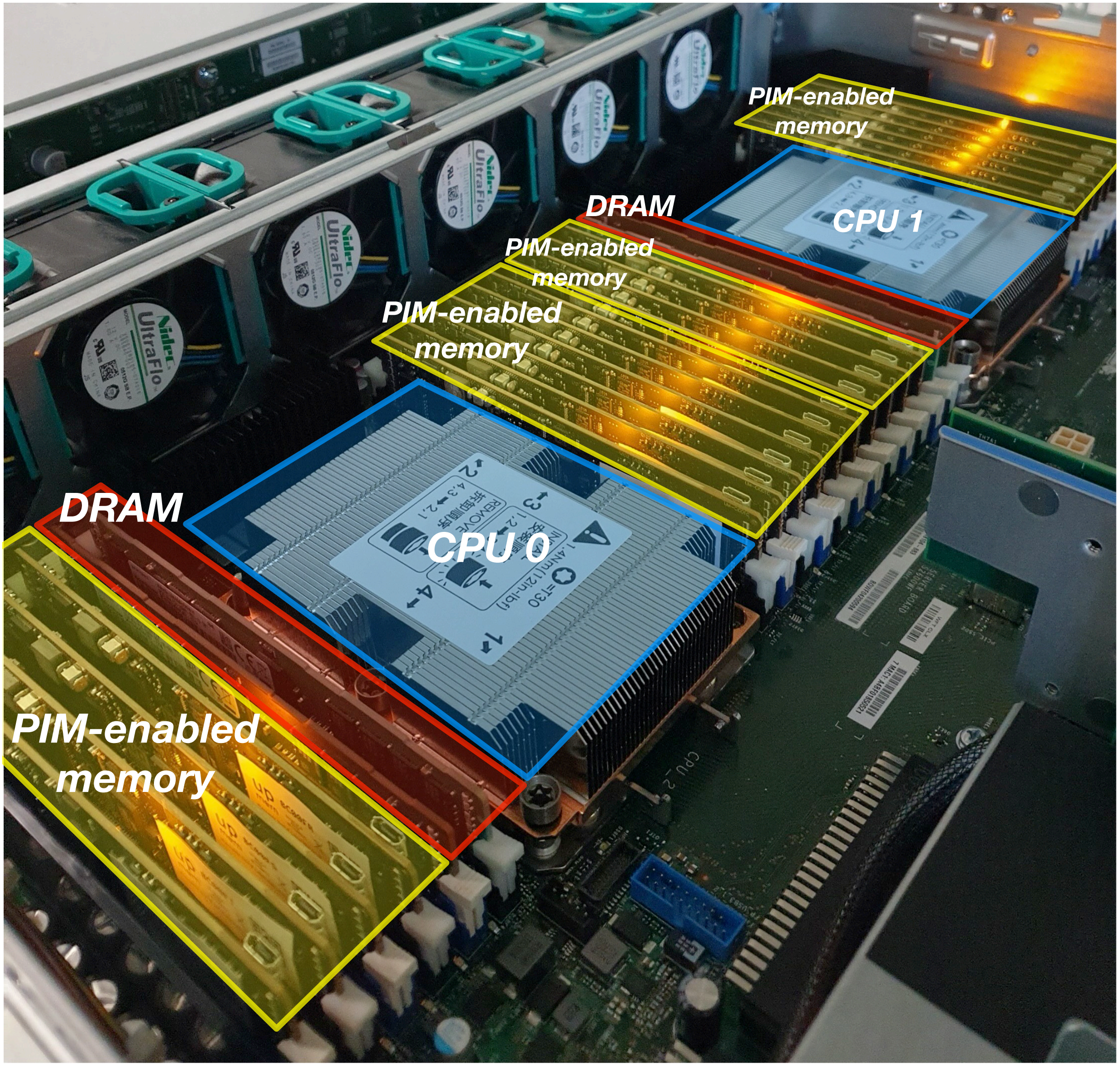
- UPMEM-based PIM system with 20 UPMEM DIMMs of 16 chips each (40 ranks)
  - P21 DIMMs
  - Dual x86 socket
    - UPMEM DIMMs coexist with regular DDR4 DIMMs
    - 2 memory controllers/socket (3 channels each)
    - 2 conventional DDR4 DIMMs on one channel of one controller



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

2,560-DPU System (II)





Evaluation: Metrics

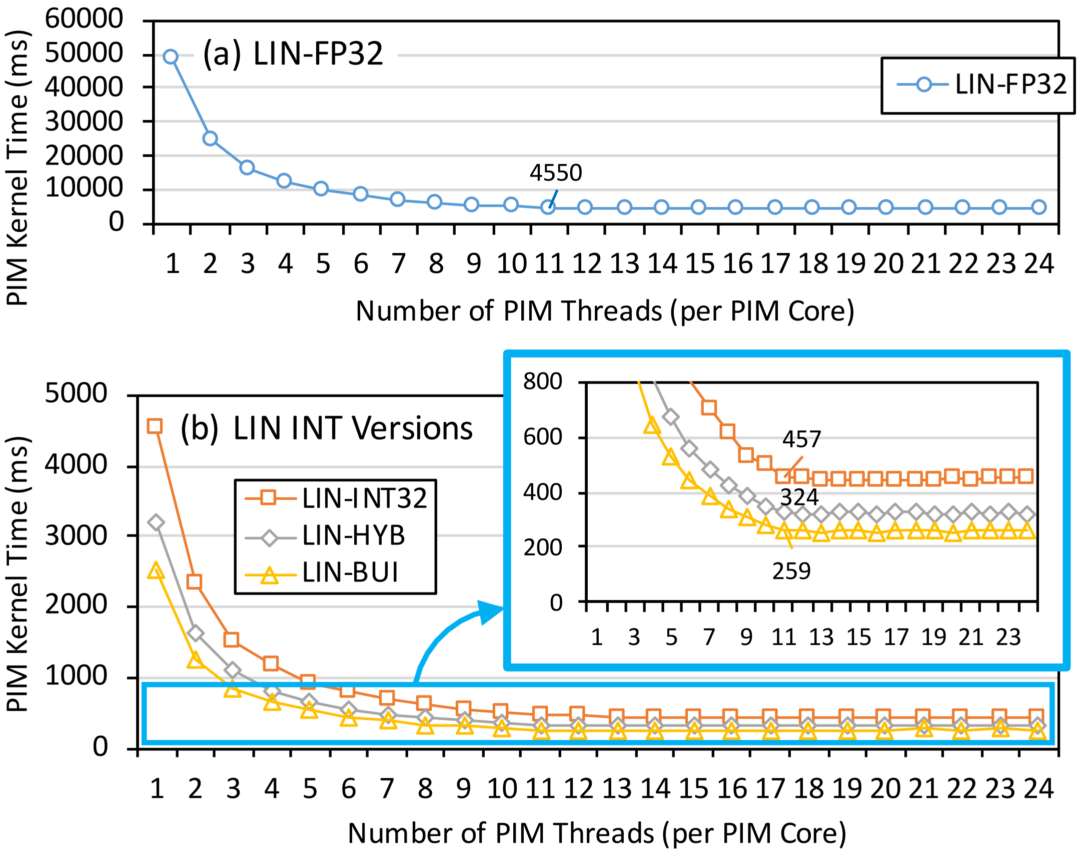
- Linear regression
  - Training error rate of `LIN-FP32` is the same as the CPU version
  - For integer versions, it remains low and close to that of `LIN-FP32`
- Logistic regression
  - LUT-based versions obtain lower training error rates that `LOG-INT32`, since they use exact values, not approximations
- Decision tree
  - Training accuracy only slightly lower than that of the CPU version
- K-means
  - Same *Calinski-Harabasz score* and *adjusted Rand index* of PIM and CPU versions

Evaluation: Analysis of PIM Kernels (I)

- Linear regression

All versions saturate at 11 or more PIM threads

Fixed point accelerates the kernel by an order of magnitude



`LIN-HYB` is 41% faster than `LIN-INT32`

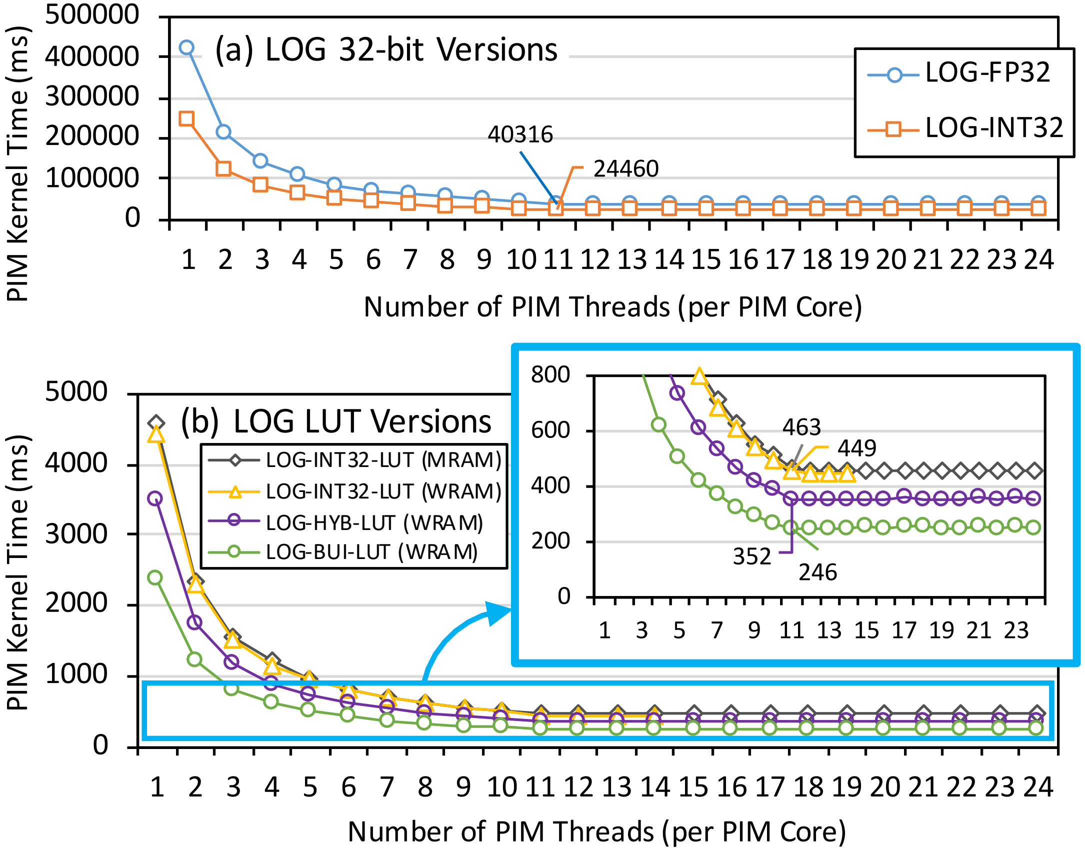
`LIN-BUI` provides an additional 25% speedup

Evaluation: Analysis of PIM Kernels (II)

- Logistic regression

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

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



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

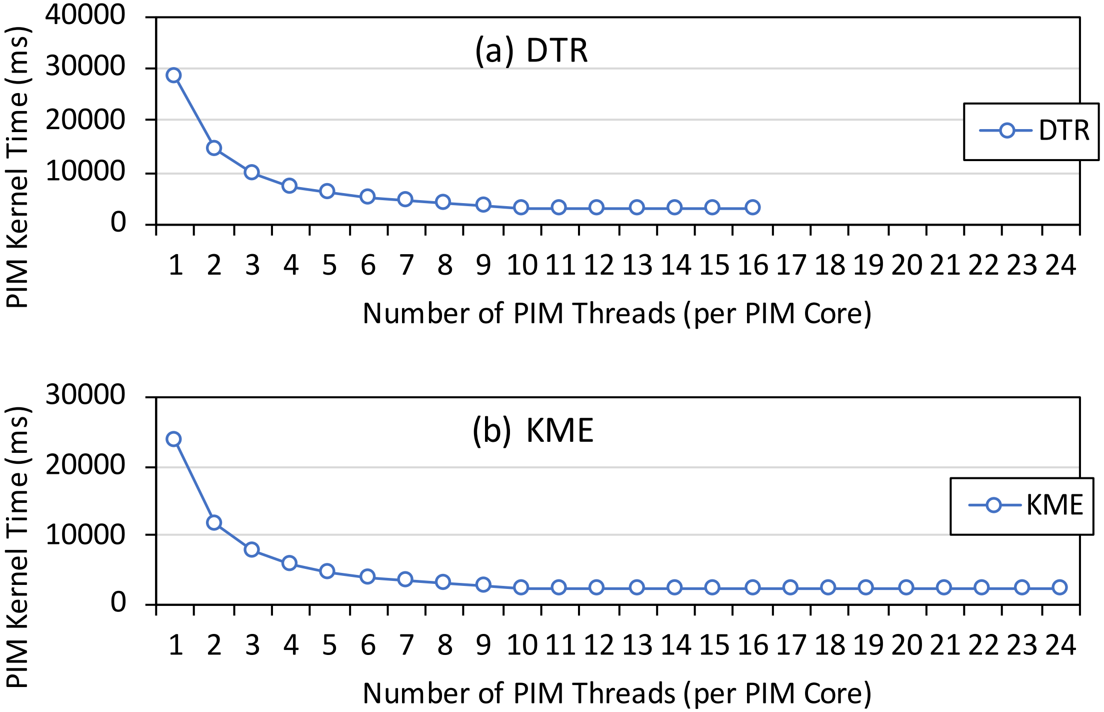
LOG-BUI-LUT provides an additional 43% speedup

Evaluation: Analysis of PIM Kernels (III)

- Decision tree & K-means

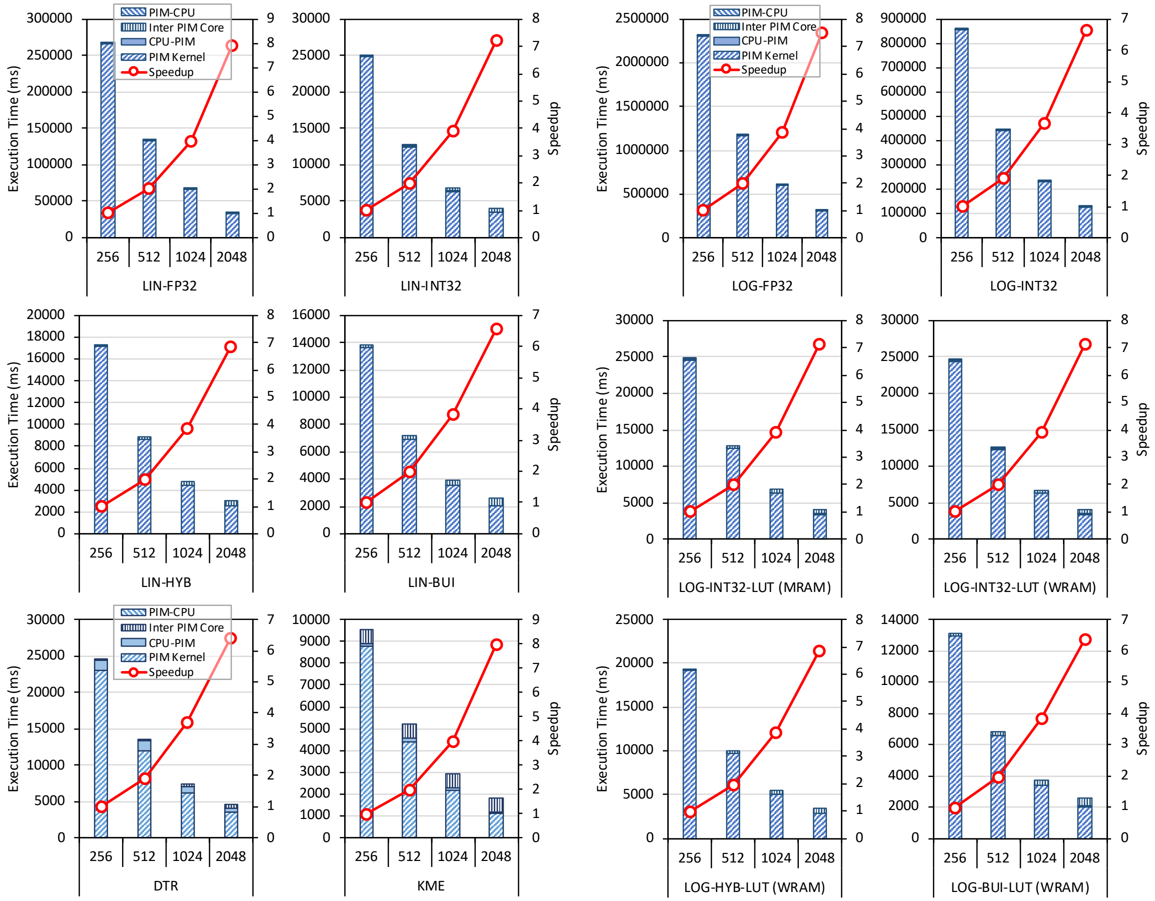
Both workloads saturate at 11 or more PIM threads

Maximum number of PIM threads in DTR is 16 due to the usage of local scratchpad memory



Evaluation: Performance Scaling

- Strong scaling: 256 to 2,048 PIM cores



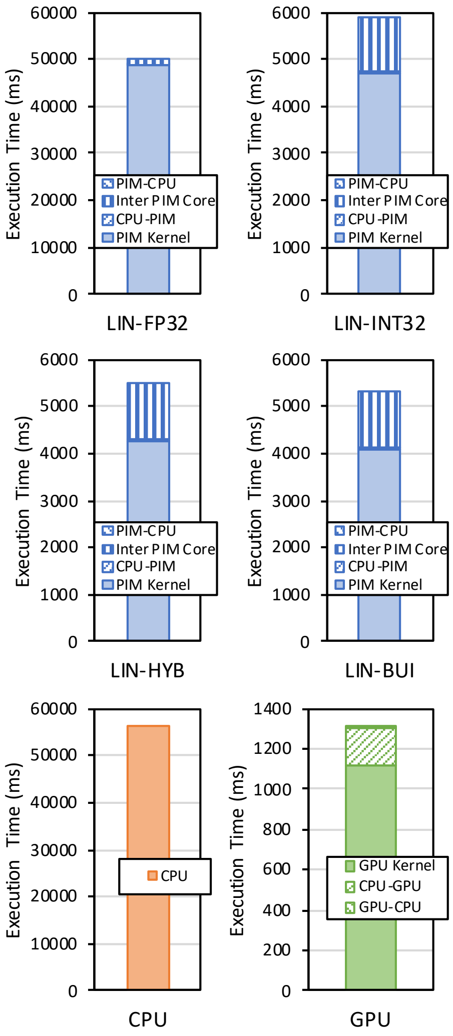
PIM kernel time scales linearly with the number of PIM cores

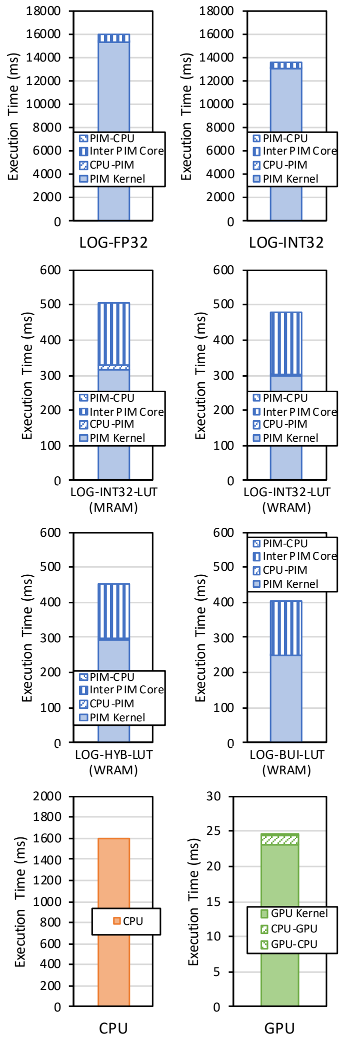
Little overhead from inter PIM core communication and communication between host and PIM cores

SAFARI

Comparison to CPU and GPU (I)

- Linear regression and logistic regression



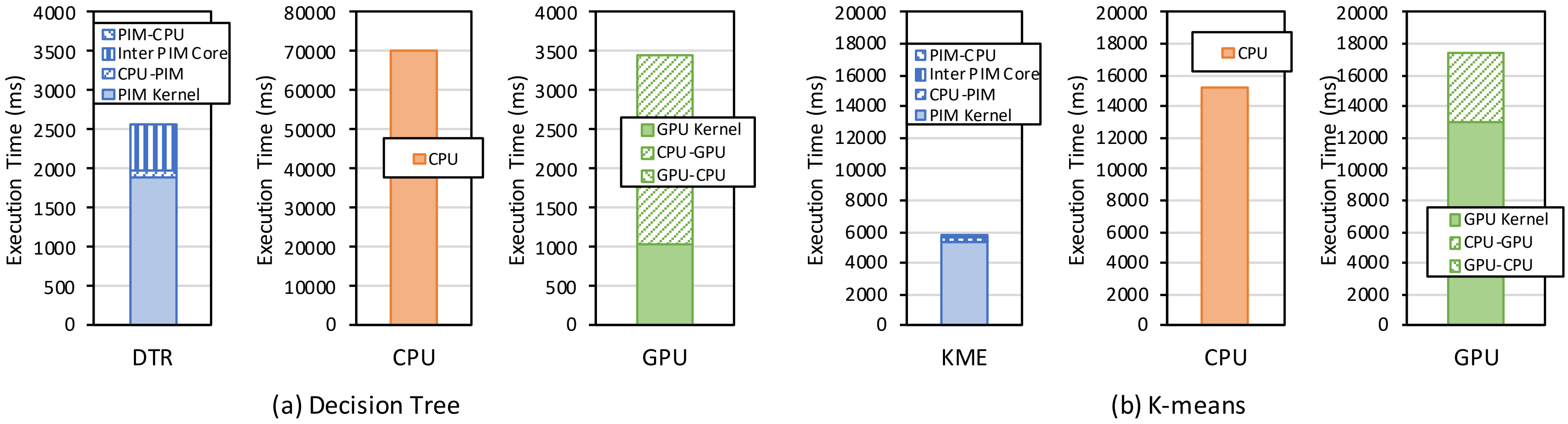


PIM versions are heavily burdened when they use operations that are not natively supported by the hardware

Several optimizations reduce the execution time considerably and close the gap with GPU performance

Comparison to CPU and GPU (II)

- Decision tree and K-means



(a) Decision Tree

(b) K-means

PIM version of DTR is 27x faster than the CPU version and 1.34x faster than the GPU version

PIM version of KME is 2.8x faster than the CPU version and 3.2x faster than the GPU version

Key Observations and Insights

- ML training workloads can greatly benefit from (1) fixed-point data representation, (2) quantization, and (3) hybrid precision implementation in PIM systems
- ML training workloads that require complex activation functions (e.g., sigmoid) can take advantage of lookup tables (LUTs) in PIM systems instead of function approximation
- Data can be placed and laid out such that memory accesses of PIM cores are streaming
- ML training workloads with large training datasets benefit from scaling the size of PIM-enabled memory with PIM cores attached to memory arrays

Machine Learning Training on a Real Processing-in-Memory System

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

Short version: https://arxiv.org/pdf/2206.06022.pdf

Long version: https://arxiv.org/pdf/2207.07886.pdf

https://www.youtube.com/watch?v=qeukNs5XI3g&t=11226s

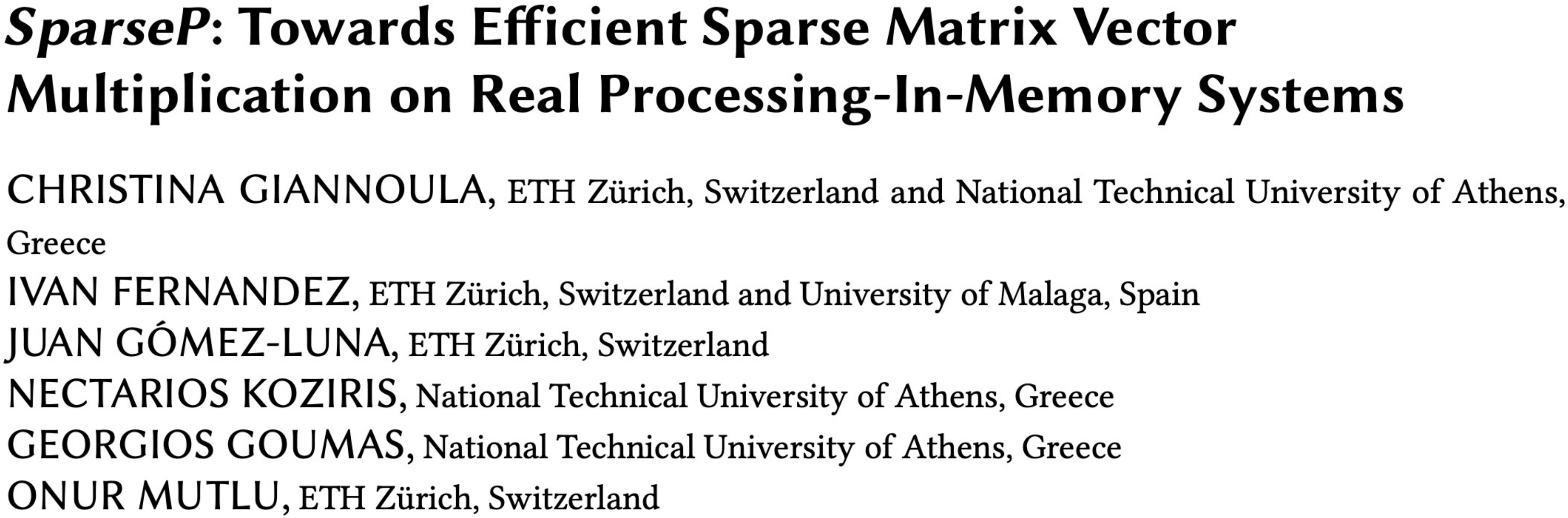
ETH Zürich SAFARI upmem

Year III Results (2022 Annual Review 1)

- Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System [IEEE Access'22]
- Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware [CUT 2021]
- An Experimental Evaluation of Machine Learning Training on a Real Processing-in-Memory System [arXiv 2022]
- SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures [SIGMETRICS 2022]
- High-throughput Pairwise Alignment with the Wavefront Algorithm using Processing-in-Memory [HICOMB 2022]
- PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM [arXiv 2021]

SpMV Multiplication on Real PIM Systems

- Appears 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

https://www.youtube.com/watch?v=5kaOsJKlGrE

SparseP

Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures

Christina Giannoula

Ivan Fernandez, Juan Gomez-Luna,

Nectarios Koziris, Georgios Goumas, Onur Mutlu

SAFARI ETHzürich National Technical University of Athens CSLab UNIVERSIDAD DE MÁLAGA

SparseP Summary

Efficient Algorithmic Designs

> The first open-source Sparse Matrix Vector Multiplication (SpMV) software package, SparseP, for real Processing-In-Memory (PIM) systems

SparseP is Open-Source

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

Extensive Characterization

> The first comprehensive analysis of SpMV on the first real commercial PIM architecture

Recommendations for Architects and Programmers

Full Paper: https://arxiv.org/pdf/2201.05072.pdf

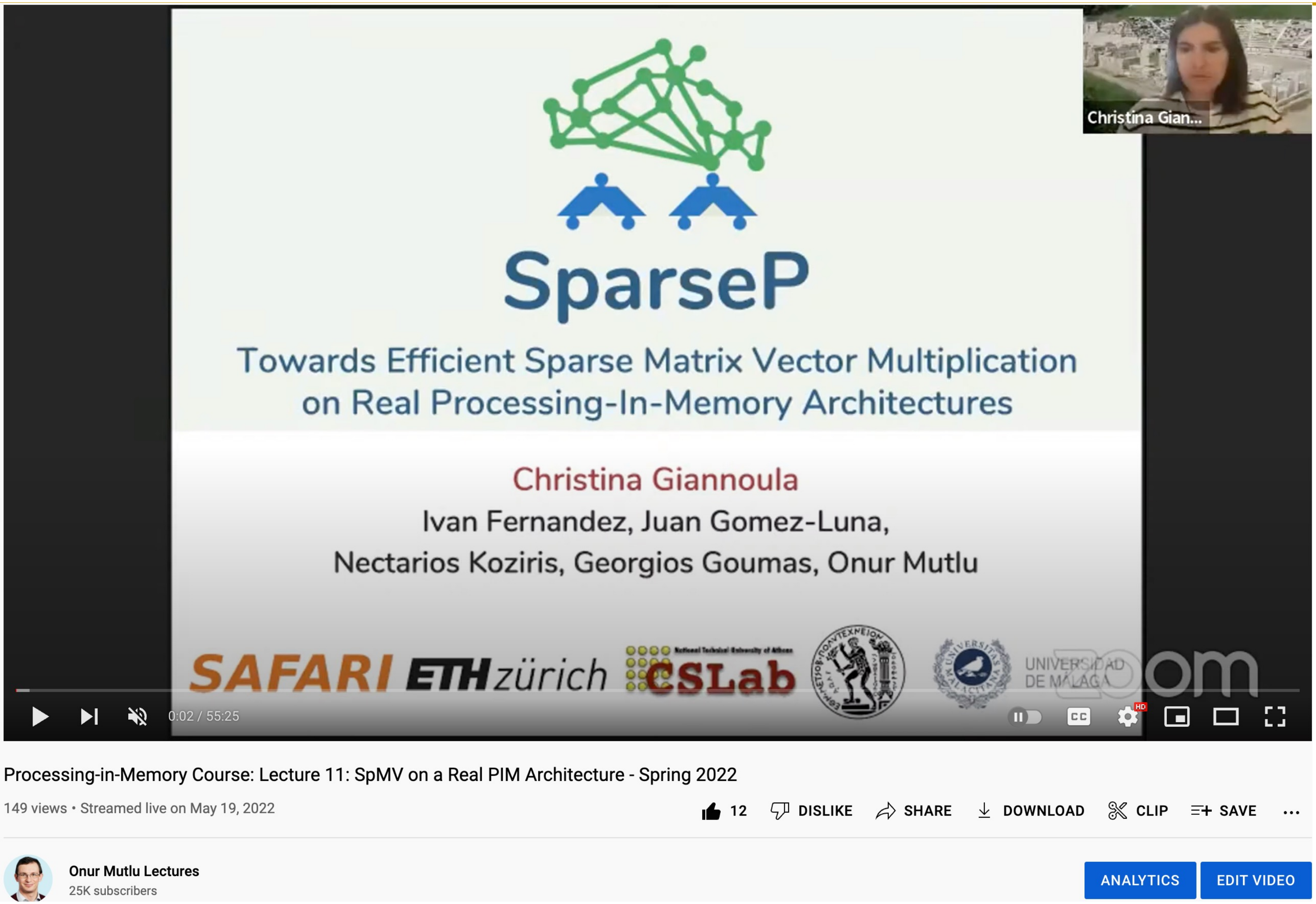
SparseP: SpMV Library for Real PIMs

Our Contributions:

1. Design efficient SpMV kernels for current and future PIM systems
   - 25 SpMV kernels
     - 4 compressed matrix formats (CSR, COO, BCSR, BCOO)
     - 6 data types
     - 4 data partitioning techniques
     - Various load balancing schemes among PIM cores/threads
     - 3 synchronization approaches

2. Provide a comprehensive analysis of SpMV on the first commercially-available real PIM system
   - 26 sparse matrices
   - Comparisons to state-of-the-art CPU and GPU systems
   - Recommendations for software, system and hardware designers

SparseP Talk Video



SparseP

Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures

Christina Giannoula

Ivan Fernandez, Juan Gomez-Luna, Nectarios Koziris, Georgios Goumas, Onur Mutlu

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

149 views • Streamed live on May 19, 2022

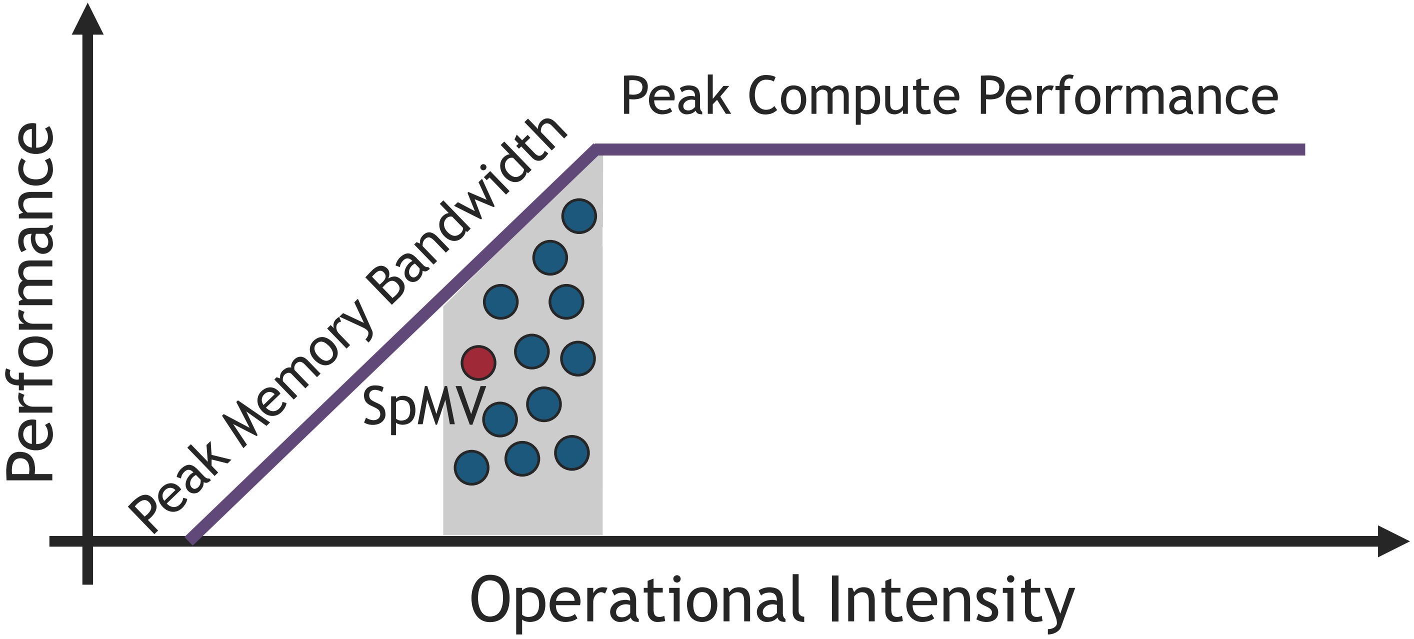
https://www.youtube.com/watch?v=5kaOsJKlGrE

Sparse Matrix Vector Multiplication

Sparse Matrix Vector Multiplication (SpMV):

- Widely-used kernel in graph processing, machine learning, scientific computing …
- A highly memory-bound kernel

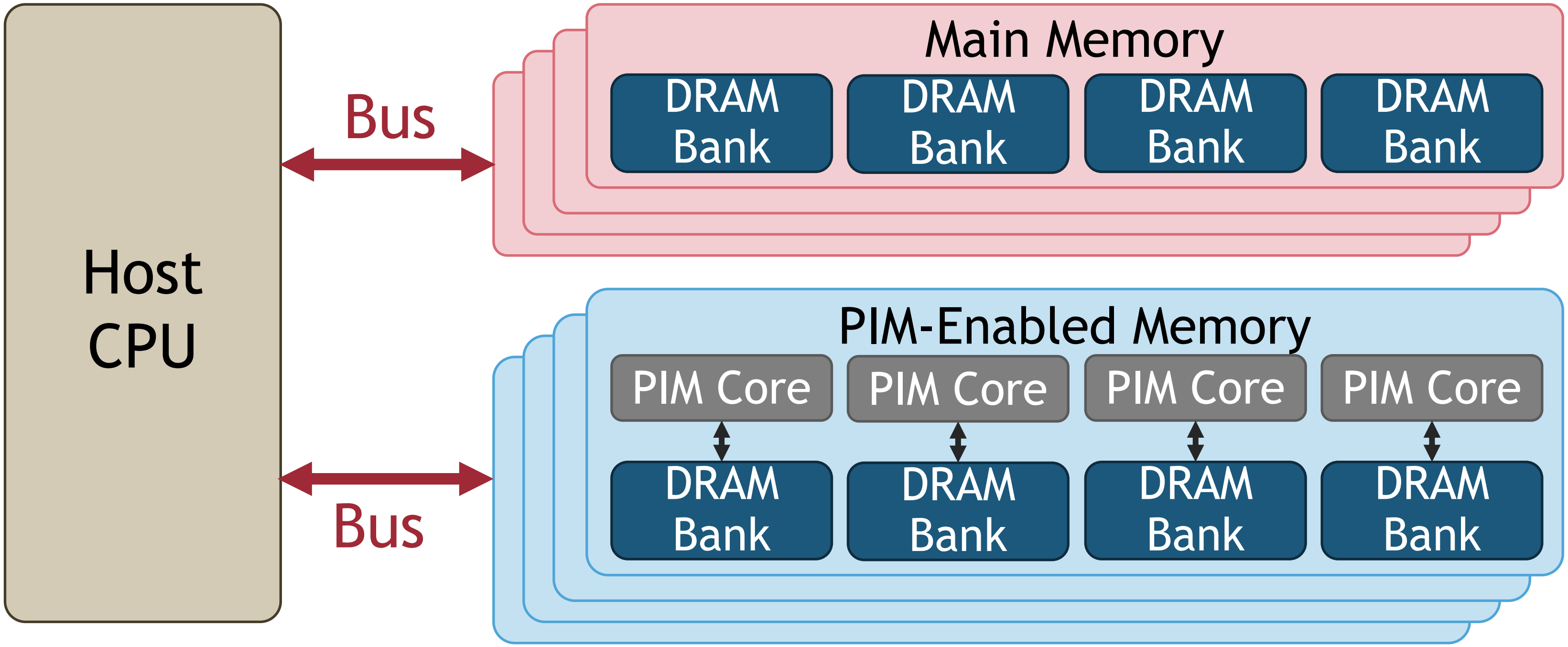
Roofline Model



Real Processing-In-Memory Systems

Real Near-Bank Processing-In-Memory (PIM) Systems:

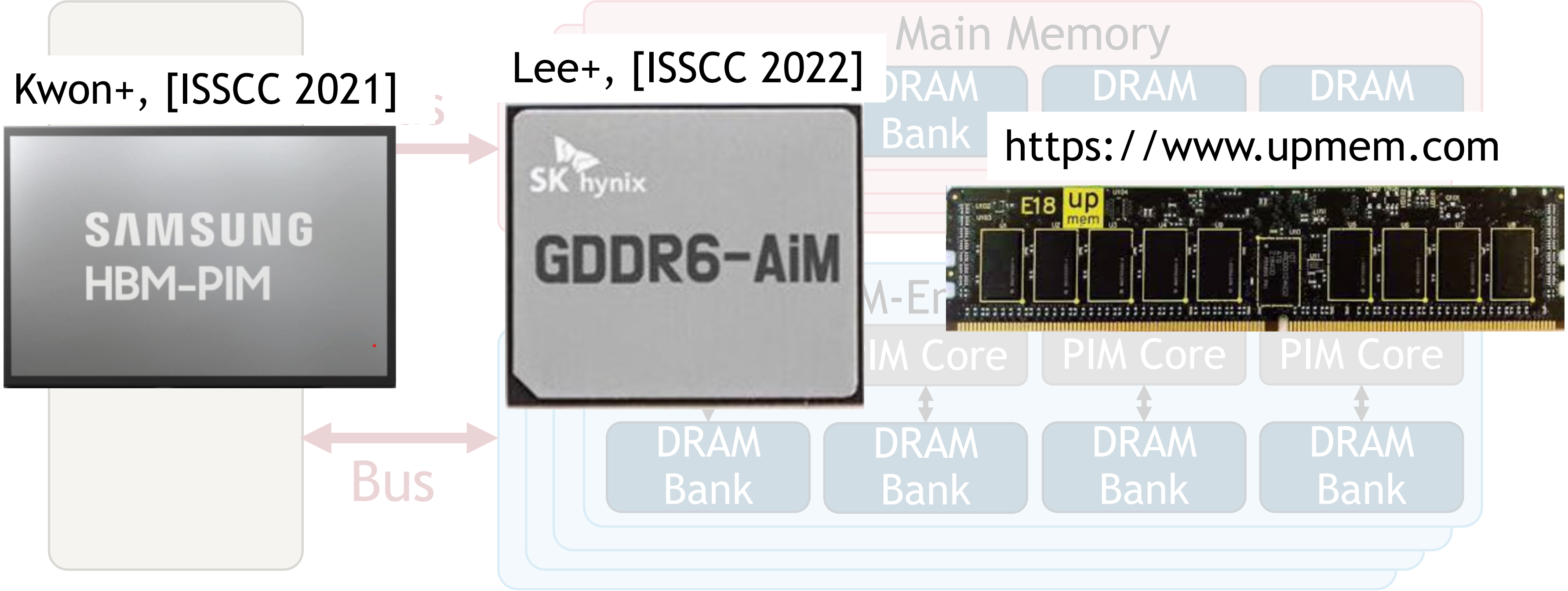
- High levels of parallelism
- Low memory access latency
- Large aggregate memory bandwidth



Real Processing-In-Memory Systems

Real Near-Bank Processing-In-Memory (PIM) Systems:

- High levels of parallelism
- Low memory access latency
- Large aggregate memory bandwidth



Kwon+, [ISSCC 2021]

Lee+, [ISSCC 2022]

https://www.upmem.com

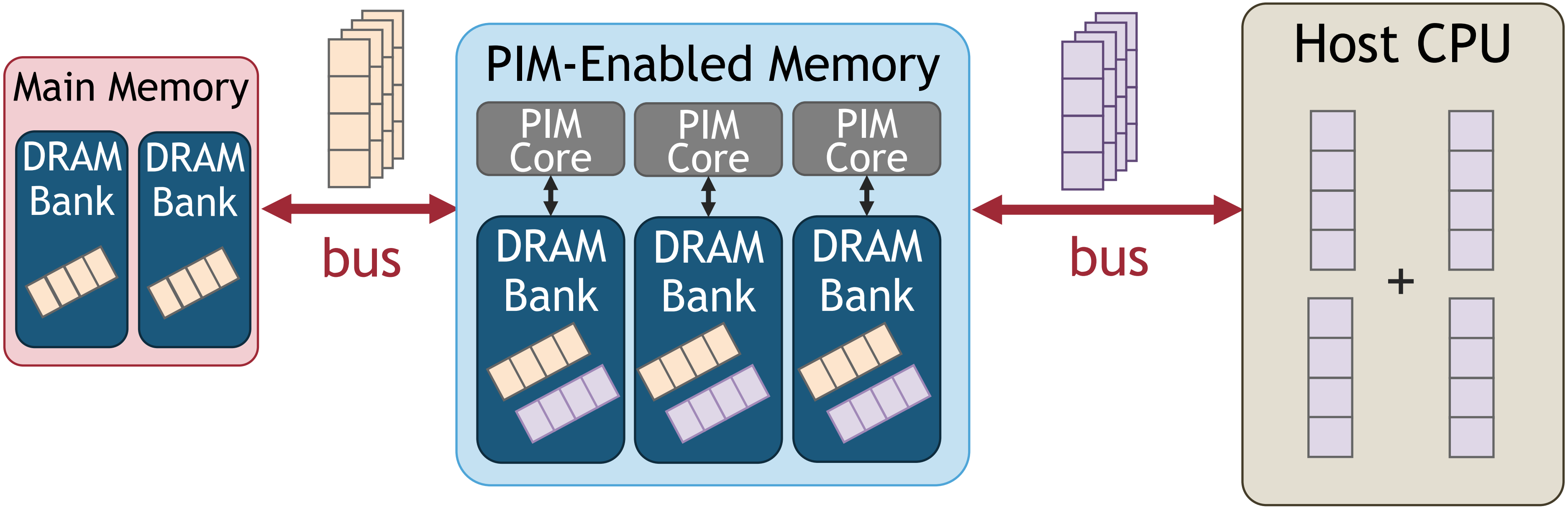
SpMV Execution on a PIM System

1 Load the input vector

2 Execute the kernel

3 Retrieve the partial results

4 Merge the partial results



SparseP Software Package

25 SpMV kernels for PIM Systems → https://github.com/CMU-SAFARI/SparseP

Partitioning	Matrix Format	Load-Balancing
9x 1D Kernels	CSR	rows, nnzs *
	COO ▲	rows, nnzs *, nnzs
	BCSR	blocks ^, nnzs ^
	BCOO ▲	blocks, nnzs
4x 2D Equally-Sized Tiles	CSR	--
	COO ▲	--
	BCSR	--
	BCOO ▲	--
6x 2D Equally-Wide Tiles	CSR	nnzs *
	COO ▲	nnzs
	BCSR	blocks ^, nnzs ^
	BCOO ▲	blocks, nnzs
6x 2D Variable-Sized Tiles	CSR	nnzs *
	COO ▲	nnzs
	BCSR	blocks ^, nnzs ^
	BCOO ▲	blocks, nnz

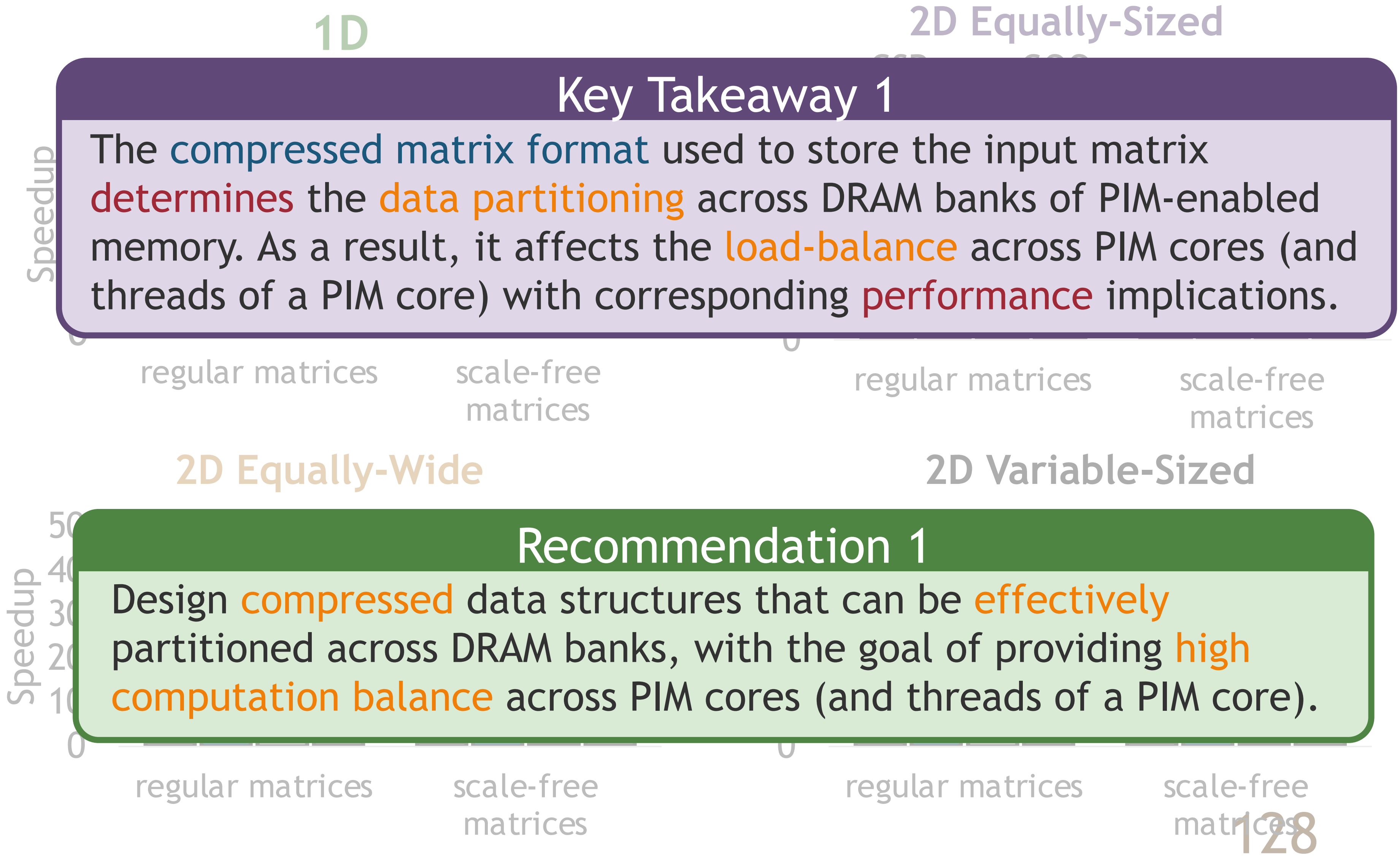
Load-balance
across PIM cores/threads:
* row-granularity (CSR)
^ block-row-granularity (BCSR)

Synchronization
among threads of a PIM core:
▲ lb-cg, lb-fb, lf (COO, BCOO)

Data Types:
- 8-bit integer
- 16-bit integer
- 32-bit integer
- 64-bit integer
- 32-bit float
- 64-bit float

Comparison of Compressed Formats

2048 PIM Cores, 32-bit integer



Key Takeaway 1

The compressed matrix format used to store the input matrix determines the data partitioning across DRAM banks of PIM-enabled memory. As a result, it affects the load-balance across PIM cores (and threads of a PIM core) with corresponding performance implications.

Recommendation 1

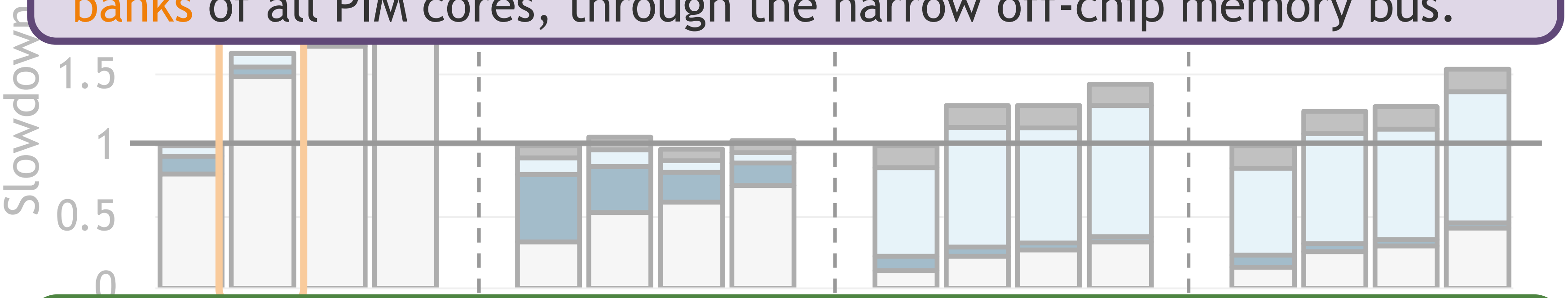
Design compressed data structures that can be effectively partitioned across DRAM banks, with the goal of providing high computation balance across PIM cores (and threads of a PIM core).

Scalability

COO format, 32-bit integer

Key Takeaway 2

The 1D-partitioned kernels are severely bottlenecked by the high data transfer costs to broadcast the whole input vector into DRAM banks of all PIM cores, through the narrow off-chip memory bus.



Recommendation 2

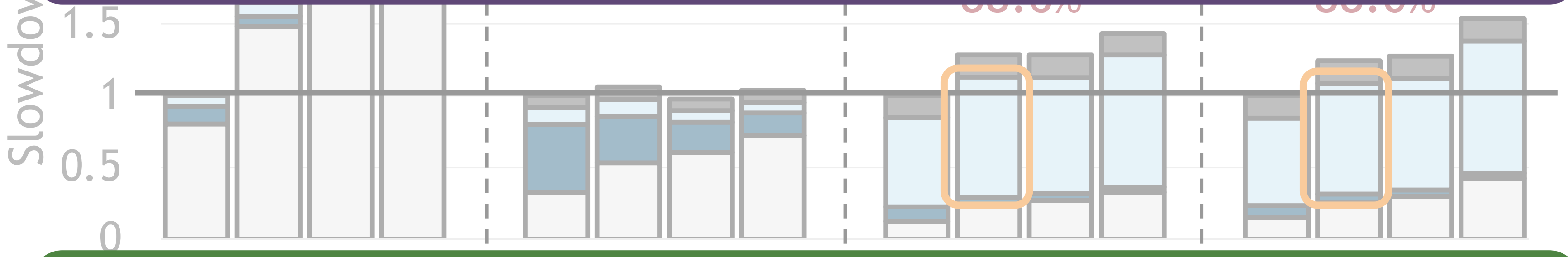
Optimize the broadcast collective collective in data transfers to PIM-enabled memory to efficiently copy the input data into DRAM banks in the PIM system.

Scalability

COO format, 32-bit integer

Key Takeaway 3

The 2D equally-wide and variable-sized kernels need fine-grained parallel data transfers at DRAM bank granularity (zero padding) to be supported by the PIM system to achieve high performance.



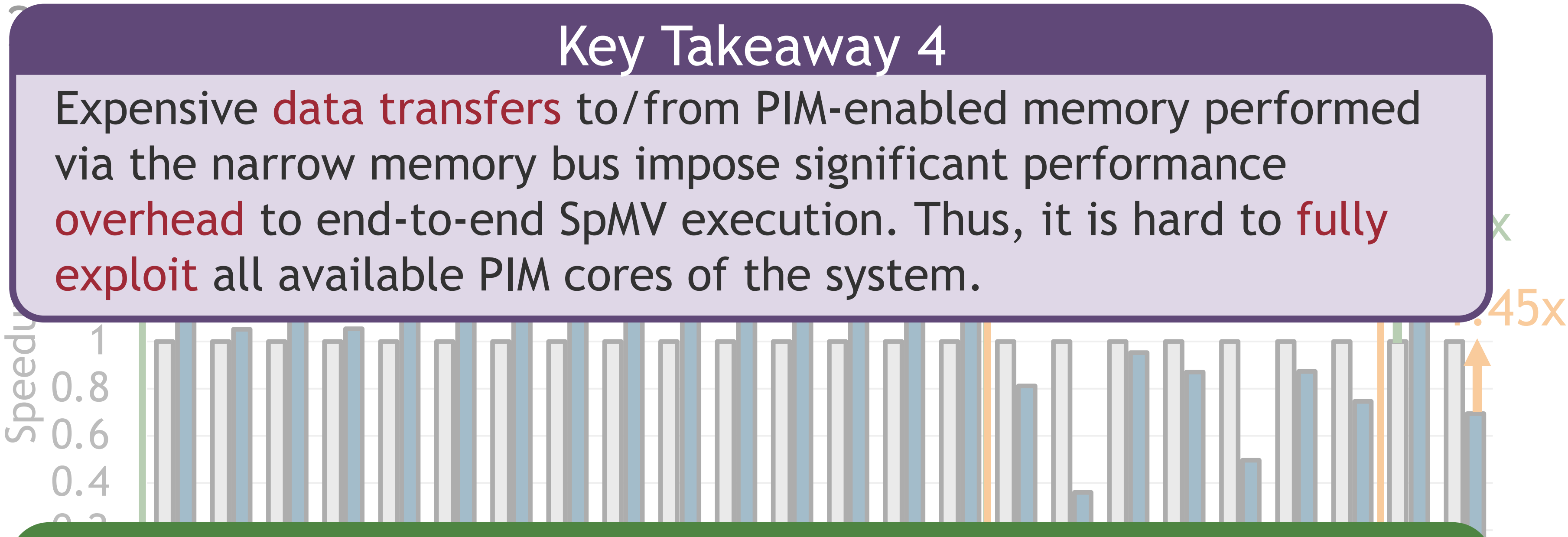
Recommendation 3

Optimize the gather collective operation at DRAM bank granularity in data transfers from PIM-enabled memory to efficiently retrieve the output results to the host CPU.

1D vs 2D

Key Takeaway 4

Expensive data transfers to/from PIM-enabled memory performed via the narrow memory bus impose significant performance overhead to end-to-end SpMV execution. Thus, it is hard to fully exploit all available PIM cores of the system.



Recommendation 4

Design high-speed communication channels and optimized libraries in data transfers to/from PIM-enabled memory, provide hardware support to effectively overlap computation with data transfers in the PIM system, and/or integrate PIM-enabled memory as the main memory of the system.

CPU/GPU Comparisons

- Kernel-Only (COO, 32-bit float):
  - CPU = 0.51% of Peak Perf.
  - GPU = 0.21% of Peak Perf.
  - PIM (1D) = 50.7% of Peak Perf.

- End-to-End (COO, 32-bit float):
  - CPU = 4.08 GFlop/s
  - GPU = 1.92 GFlop/s
  - PIM (1D) = 0.11 GFlop/s

System		Peak Performance	Bandwidth	TDP
CPU	Intel Xeon Silver 4110	660 GFlops	23.1 GB/s	2x85 W
GPU	NVIDIA Tesla V100	14.13 TFlops	897 GB/s	300 W
PIM	UPMEM 1st Gen.	4.66 GFlops	1.77 TB/s	379 W

Processor-Centric

Memory-Centric

CPU/GPU Comparisons

- Kernel-Only (COO, 32-bit float):
  - CPU = 0.51% of Peak Perf.
  - GPU = 0.21% of Peak Perf.
  - PIM (1D) = **50.7%** of Peak Perf.

- End-to-End (COO, 32-bit float):
  - CPU = **4.08 GFlop/s**
  - GPU = 1.92 GFlop/s
  - PIM (1D) = 0.11 GFlop/s

Many more results in the full paper:
https://arxiv.org/pdf/2201.05072.pdf

SparseP

Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures

Christina Giannoula

Ivan Fernandez, Juan Gomez-Luna,

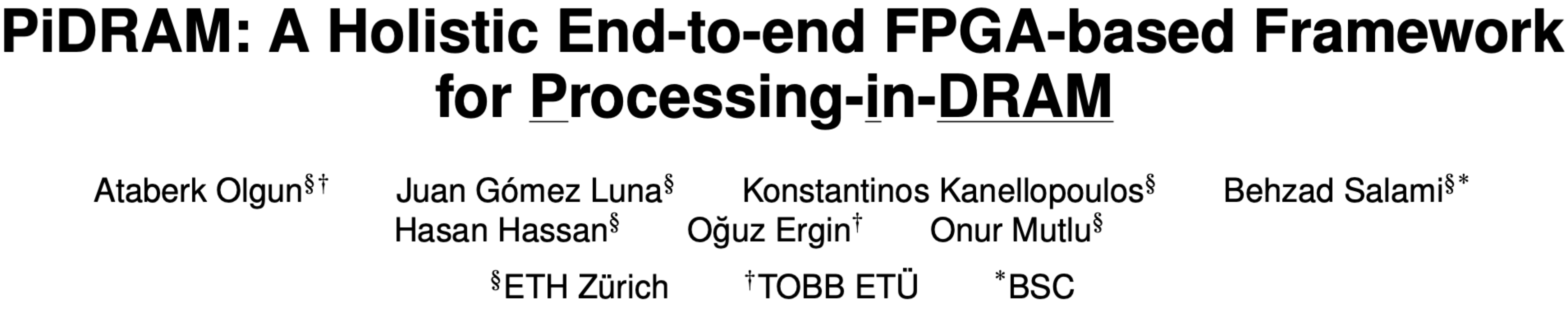
Nectarios Koziris, Georgios Goumas, Onur Mutlu

SAFARI ETH zürich National Technical University of Athens CSLab UNIVERSIDAD DE MÁLAGA

Year III Results (2022 Annual Review 1)

- Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System [IEEE Access'22]
- Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware [CUT 2021]
- An Experimental Evaluation of Machine Learning Training on a Real Processing-in-Memory System [arXiv 2022]
- SparseP: Towards Efficient Sparse Matrix Vector Multiplication on Real Processing-In-Memory Architectures [SIGMETRICS 2022]
- High-throughput Pairwise Alignment with the Wavefront Algorithm using Processing-in-Memory [HICOMB 2022]
- PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM [arXiv 2021]

Real Processing Using Memory Prototype



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

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

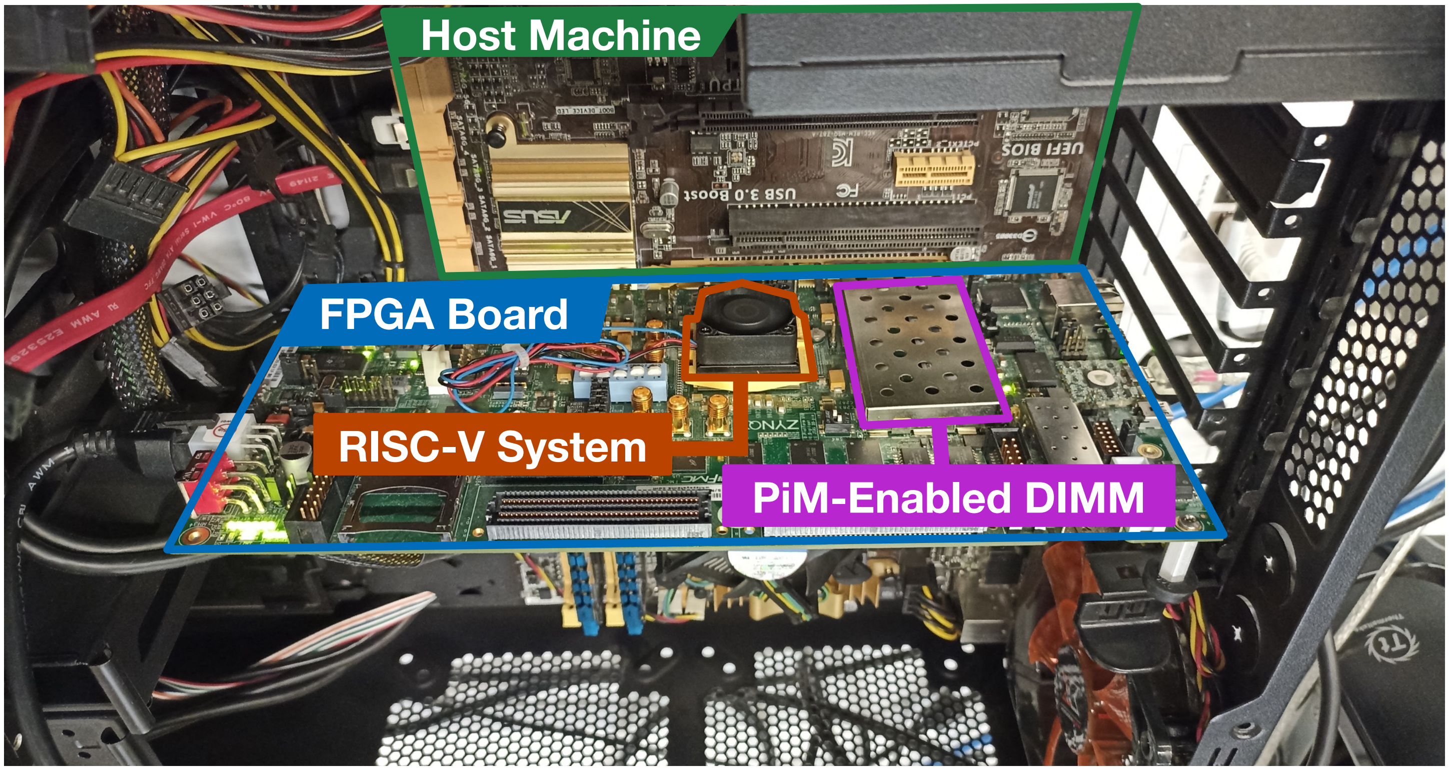
[§]ETH Zürich [†]TOBB ETÜ [*]BSC

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

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

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

Real Processing Using Memory Prototype

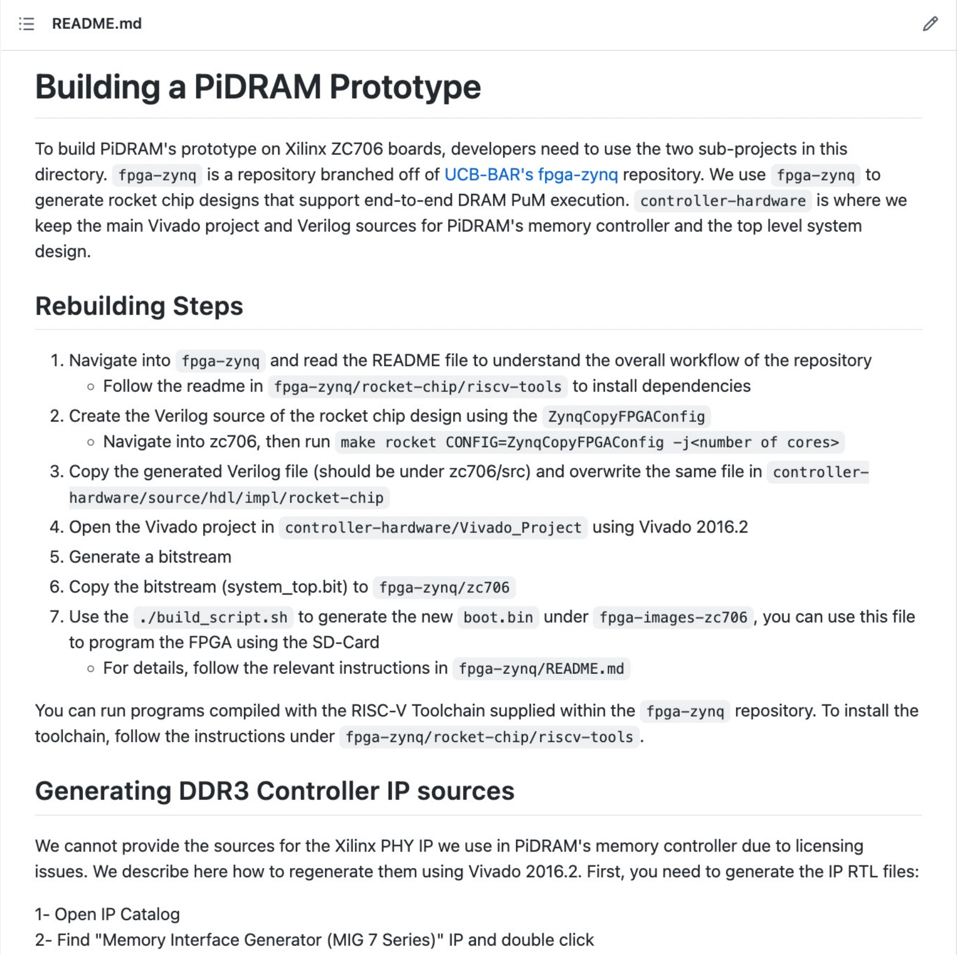


https://arxiv.org/pdf/2111.00082.pdf

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

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

Real Processing Using Memory Prototype



README.md

Building a PiDRAM Prototype

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

Rebuilding Steps

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

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

Generating DDR3 Controller IP sources

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

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

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

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

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

PiDRAM
An FPGA-based Framework for End-to-end Evaluation of Processing-in-DRAM Techniques

Ataberk Olgun

Juan Gomez Luna Konstantinos Kanellopoulos Behzad Salami

Hasan Hassan Oğuz Ergin Onur Mutlu

SAFARI kasırga

ETH zürich TOBB ETÜ University of Economics & Technology

Executive Summary

Motivation: Commodity DRAM based PiM techniques improve the performance and energy efficiency of computing systems at no additional DRAM hardware cost

Problem: Challenges of integrating these PiM techniques into real systems are not solved
General-purpose computing systems, special-purpose testing platforms, and system simulators *cannot* be used to efficiently study system integration challenges

Goal: Design and implement a flexible framework that can be used to:

- Solve system integration challenges
- Analyze trade-offs of end-to-end implementations

of commodity DRAM based PiM techniques

Key idea: PiDRAM, an FPGA-based framework that enables:

- System integration studies
- End-to-end evaluations

of commodity DRAM based PiM techniques using real unmodified DRAM chips

Evaluation: End-to-end integration of two PiM techniques on PiDRAM's FPGA prototype

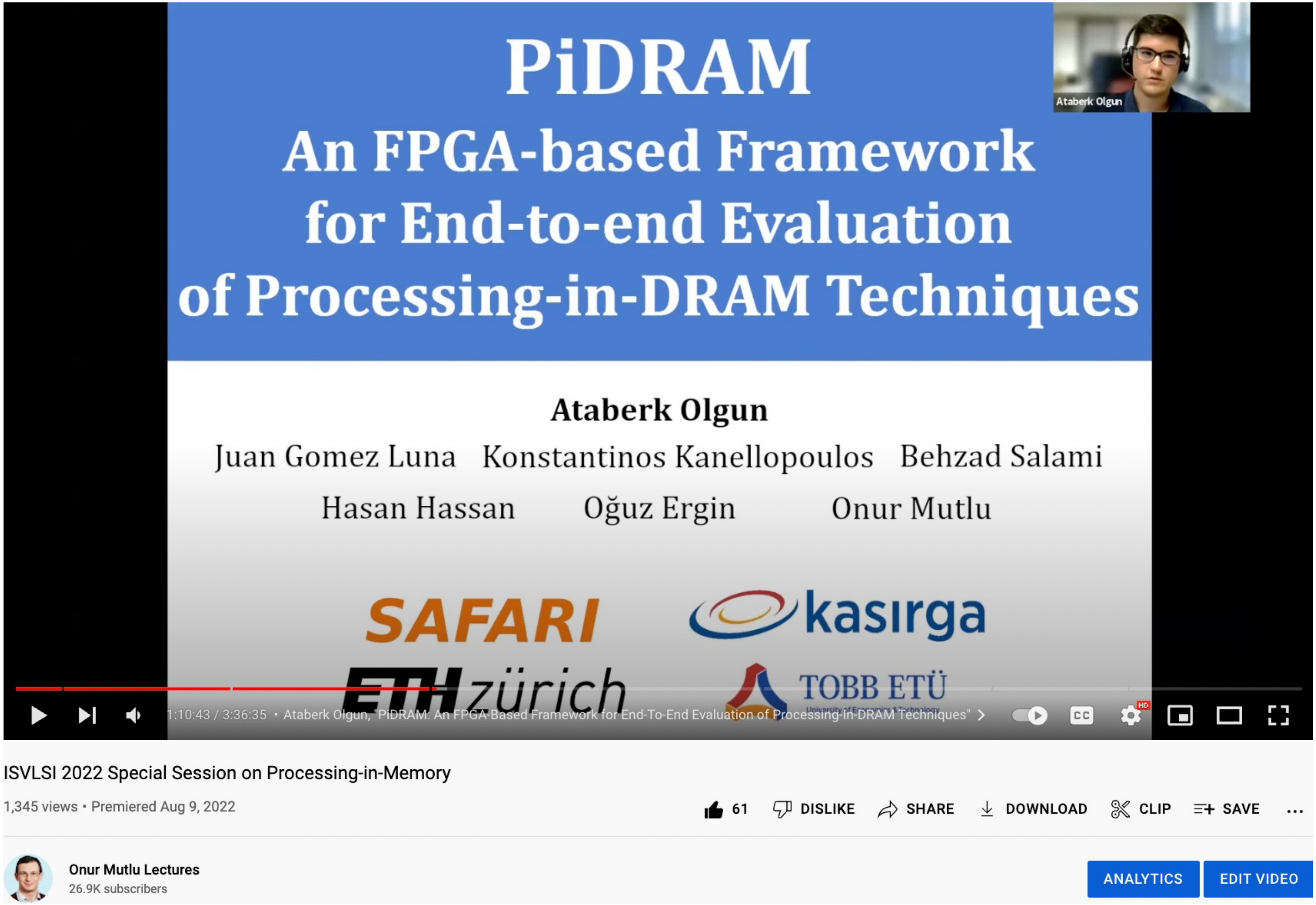
Case Study #1 – RowClone: In-DRAM bulk data copy operations

- 119x speedup for copy operations compared to CPU-copy with system support
- 198 lines of Verilog and 565 lines of C++ code over PiDRAM's flexible codebase

Case Study #2 – D-RaNGe: DRAM-based random number generation technique

- 8.30 Mb/s true random number generator (TRNG) throughput, 220 ns TRNG latency
- 190 lines of Verilog and 78 lines of C++ code over PiDRAM's flexible codebase

PiDRAM Talk Video



PiDRAM

An FPGA-based Framework for End-to-end Evaluation of Processing-in-DRAM Techniques

Ataberk Olgun

Juan Gomez Luna   Konstantinos Kanellopoulos   Behzad Salami

Hasan Hassan   Oğuz Ergin   Onur Mutlu

1:10:43 / 3:36:35 • Ataberk Olgun, "PiDRAM: An FPGA-Based Framework for End-To-End Evaluation of Processing-In-DRAM Techniques" >

ISVLSI 2022 Special Session on Processing-in-Memory

1,345 views • Premiered Aug 9, 2022

Onur Mutlu Lectures

26.9K subscribers

SAFARI https://www.youtube.com/watch?v=qeukNs5XI3g&t=4243s

PiDRAM: Overview (I)

A flexible framework that can be used to:

- Solve system integration challenges
- Analyze trade-offs of end-to-end implementations

of commodity DRAM based PiM techniques

Identify key components shared across PiM techniques

Implement customizable key components:

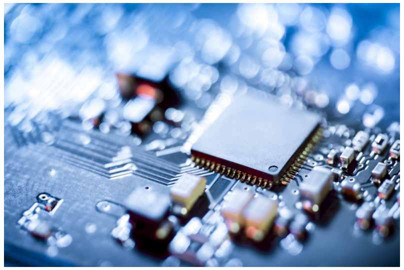
- Provide modularity, enhance extensibility of the framework

Common basis to enable system support for PiM techniques

PiDRAM: Overview (II)

Identify and develop four key hardware and software components

Hardware



❶ Flexible PiM Ops. Controller

❷ Easy-to-extend Memory Controller

Software

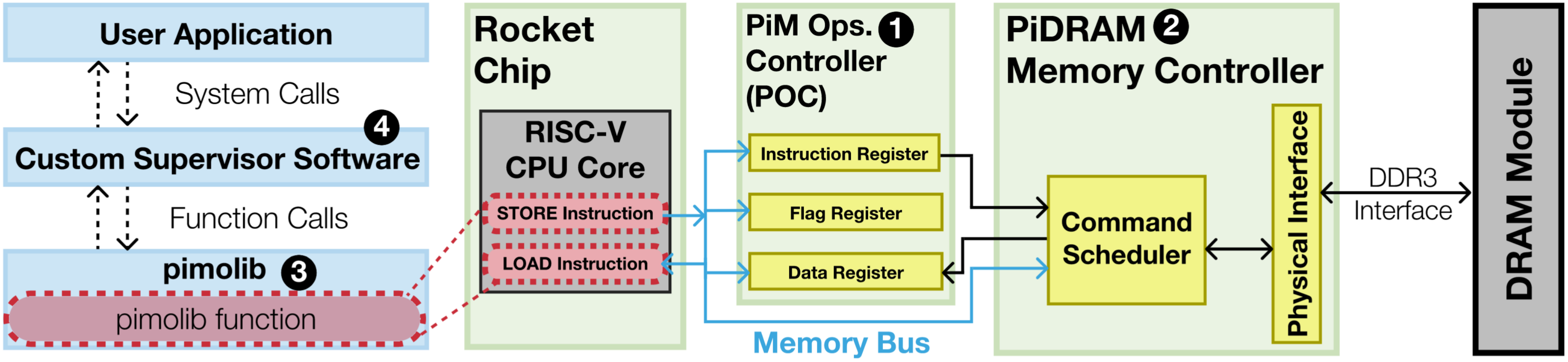
❸ Extensible Software Library

❹ Custom Supervisor Software

PiDRAM: System Design

Key components are attached to a real computing system

- PiM Ops. Controller and PiDRAM Memory Controller is implemented within the hardware system
- Custom supervisor software runs on the hardware system
- Extensible software library is used by the supervisor software

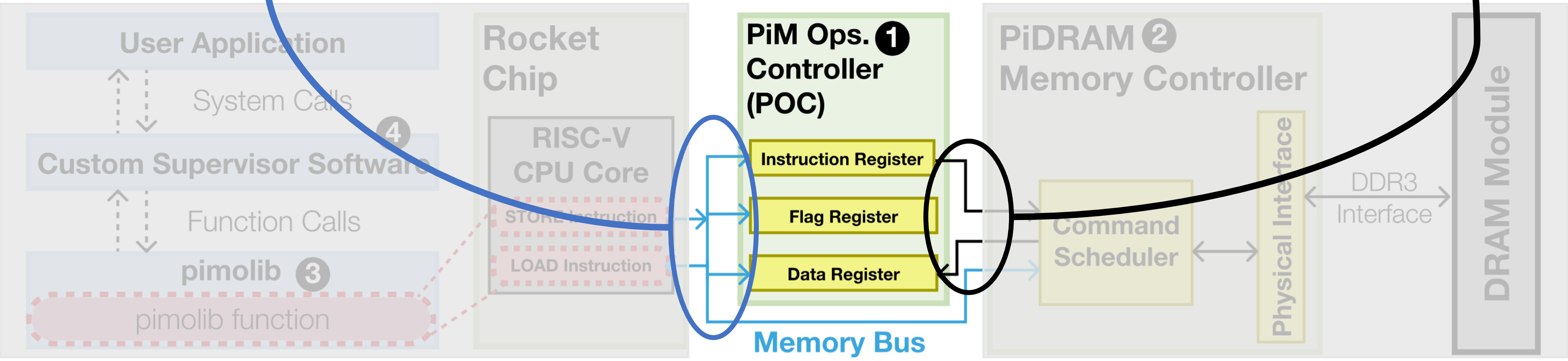


PiM Operations Controller (POC)

Decode & execute PiDRAM instructions (e.g., in-DRAM copy)

Receive instructions over memory-mapped interface
(portable to other systems with different CPU ISAs)

Simple interface to the PiDRAM memory controller
(i) send request, (ii) wait until completion, (iii) read results



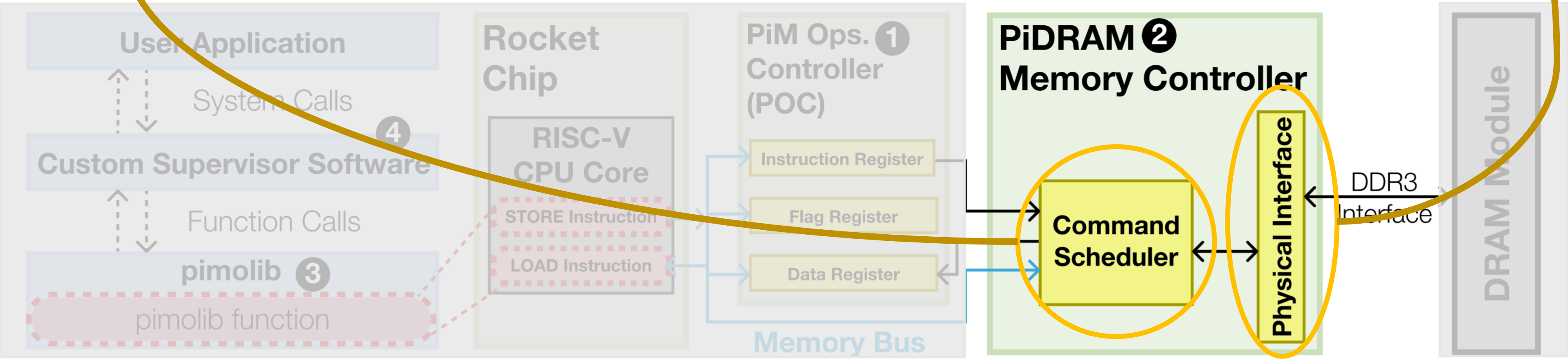
PiDRAM Memory Controller

Perform PiM operations by violating DRAM timing parameters

Support conventional memory operations (e.g., LOAD/STORE)
One state machine per operation (e.g., LOAD/STORE, in-DRAM copy)

✦ Easily replicate a state machine to implement a new operation

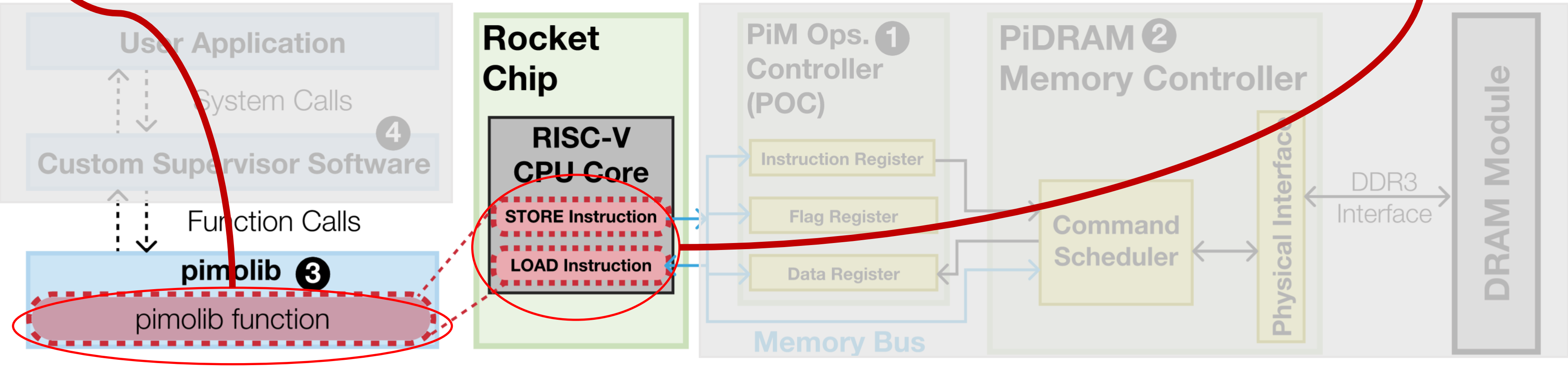
Controls the physical DDR3 interface
Receives commands from command scheduler & operates DDR3 pins



PiM Operations Library (pimolib)

Contains customizable functions that interface with the POC
Software interface for performing PiM operations

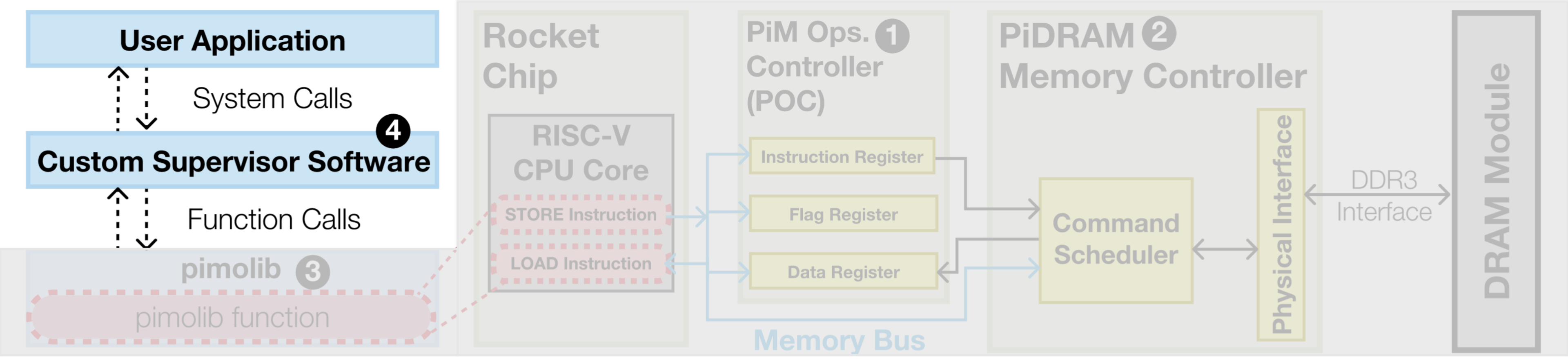
Executes LOAD & STORE requests to communicate with the POC



Custom Supervisor Software

Exposes PiM operations to the user application via system calls

Contains the necessary OS primitives to develop end-to-end PiM techniques (e.g., memory management and allocation for RowClone)



PiM Operation Execution Flow

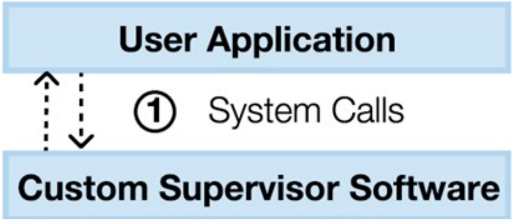
`copy()` function called by the user to perform a RowClone-Copy operation in DRAM

1. Application makes a system call: `copy(A, B, N bytes)`

2. Custom Supervisor Software calls the `copy()` pimolib function

`Copy (S, D)`

`S`: source DRAM row
`D`: destination DRAM row



PiM Operation Execution Flow

3 `Copy(S, D)` executes two store instructions in the CPU

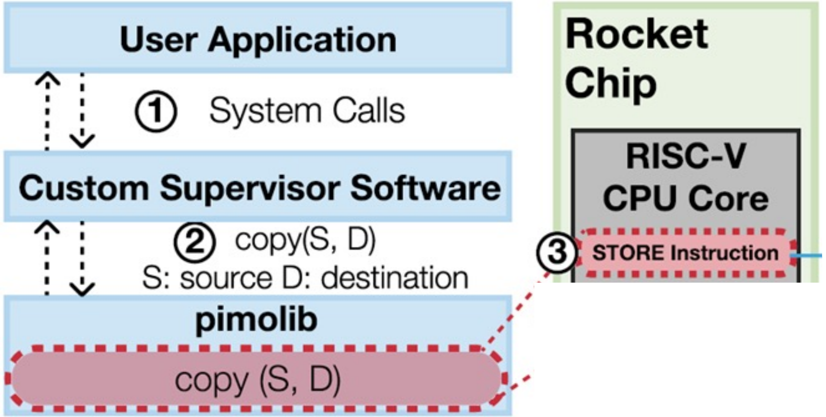
4 The first store updates the *instruction* register with `Copy(S, D)`

5 The second store sets the “Start” flag in the *flag* register

Start (S)

1

Start the execution of PiM operation

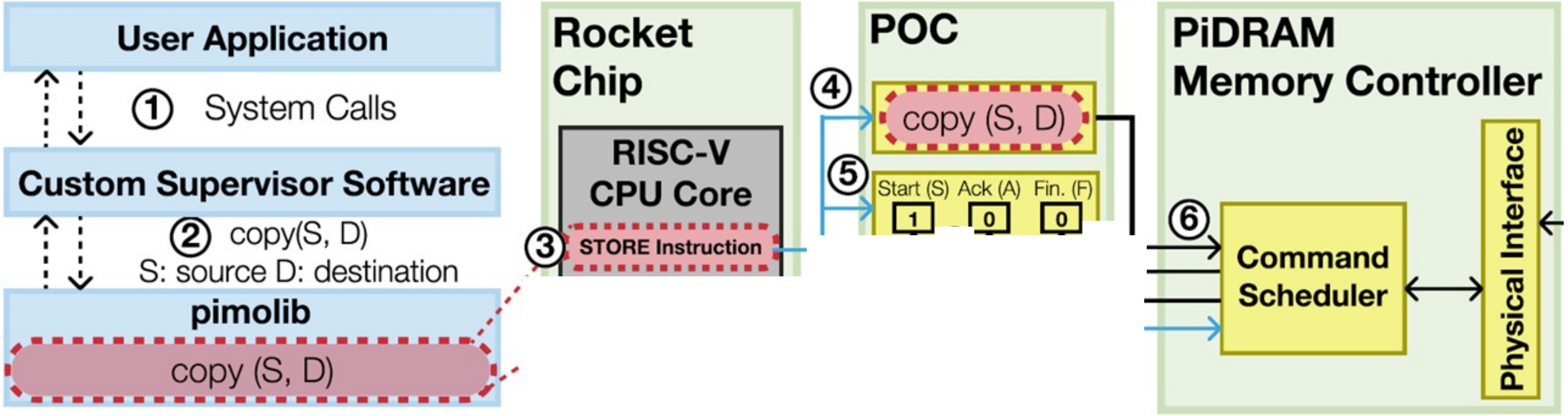


PiM Operation Execution Flow

6 POC instructs the memory controller to perform RowClone

7 POC resets the “Start” flag, and sets the “Ack” flag

8 PiDRAM memory controller issues commands with violated timing parameters to the DDR3 module

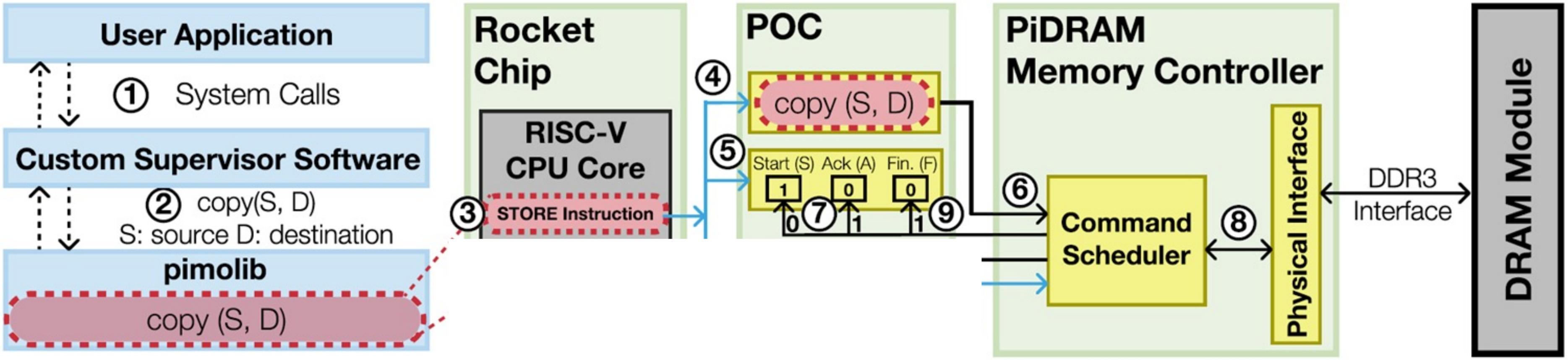


PiM Operation Execution Flow

9 The memory controller sets the “Fin.” flag

10 `Copy(S, D)` periodically checks either “Ack” or “Fin.” flags using LOAD instructions

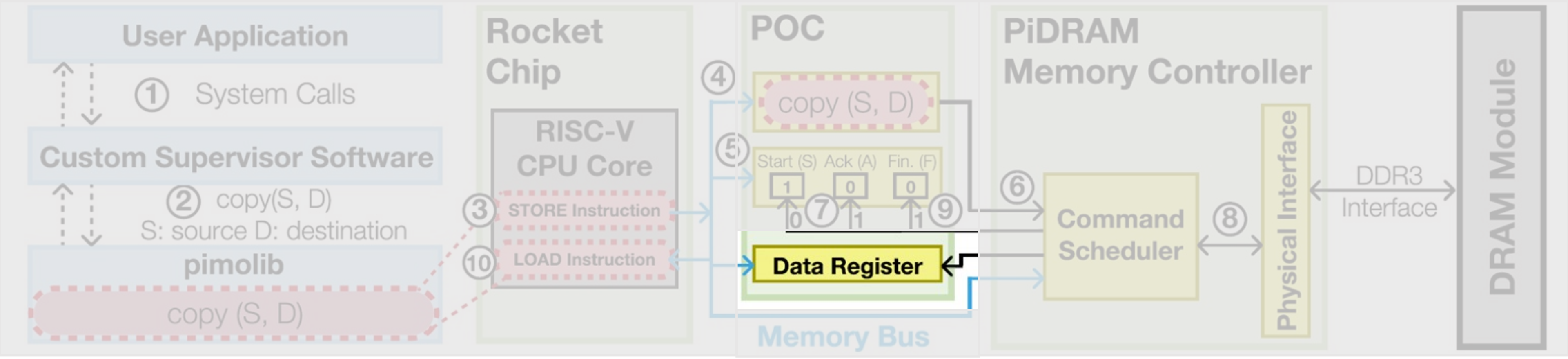
`Copy(S, D)` returns when the periodically checked flag is set



PiM Operation Execution Flow

Data Register is not used in RowClone operations because the result is stored *in memory*

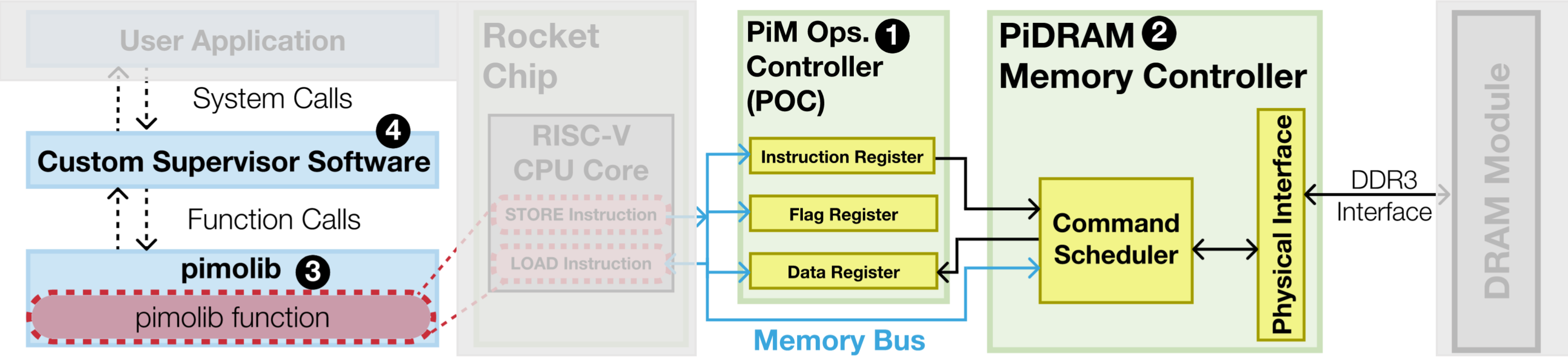
It is used to read true random numbers generated by D-RaNGe



PiDRAM Components Summary

Four key components orchestrate PiM operation execution

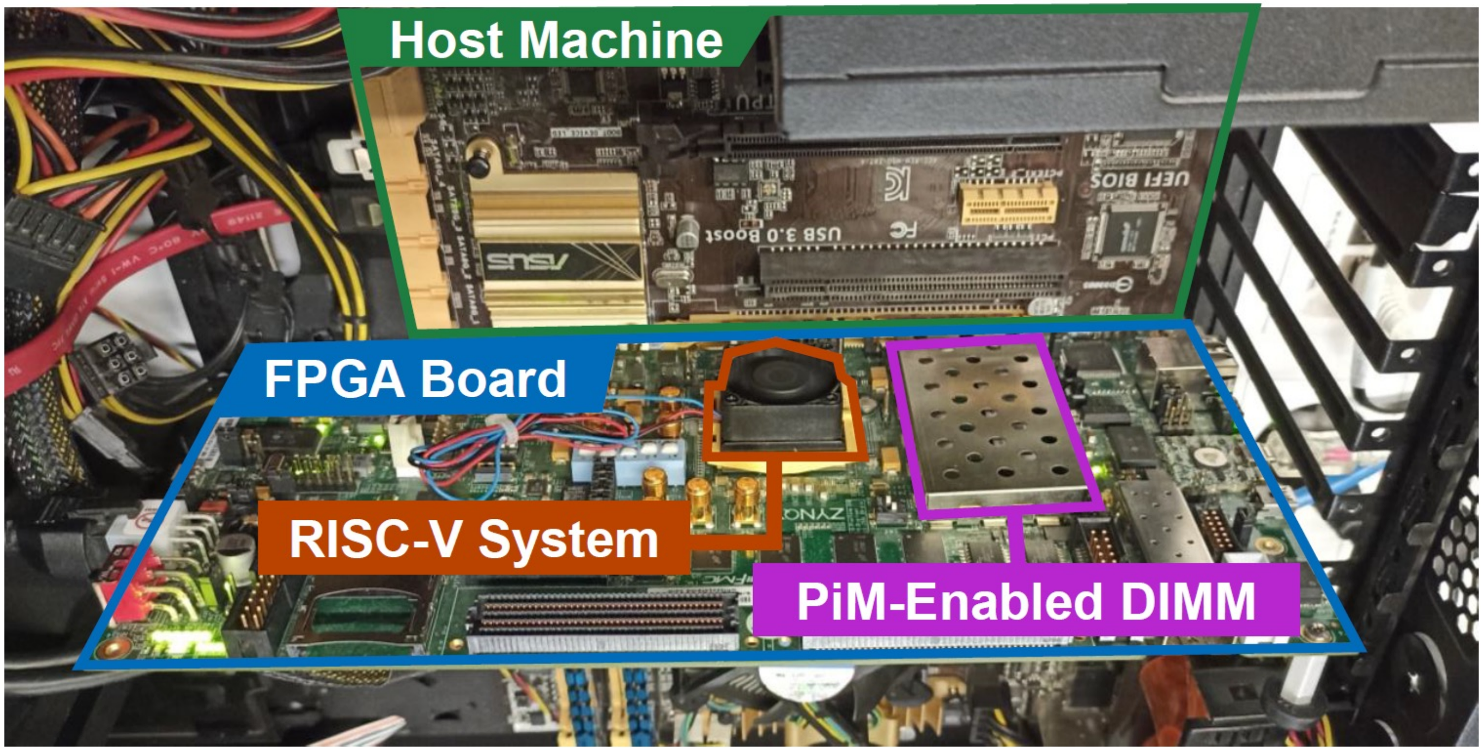
Four key components provide an extensible basis for end-to-end integration of PiM techniques



PiDRAM's FPGA Prototype

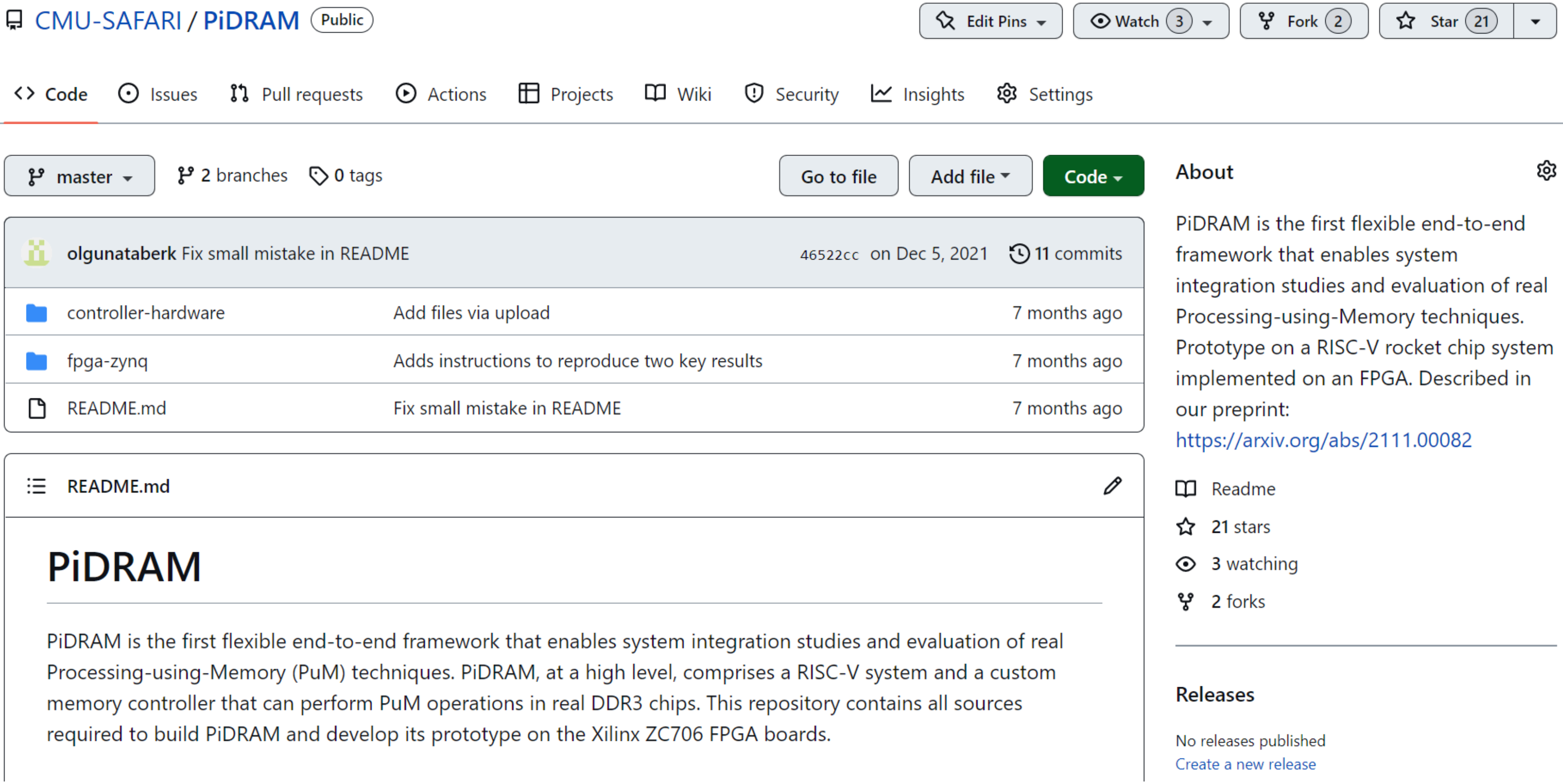
Full system prototype on Xilinx ZC706 FPGA board

- **RISC-V System:** In-order, pipelined RISC-V Rocket CPU core, L1D/I$, TLB
- **PiM-Enabled DIMM:** Micron MT8JTF12864, 1 GiB, 8 banks



PiDRAM is Open Source

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



olgunataberk Fix small mistake in README 46522cc on Dec 5, 2021 11 commits

controller-hardware	Add files via upload	7 months ago
fpga-zynq	Adds instructions to reproduce two key results	7 months ago
README.md	Fix small mistake in README	7 months ago

README.md

PiDRAM

PiDRAM is the first flexible end-to-end framework that enables system integration studies and evaluation of real Processing-using-Memory (PuM) techniques. PiDRAM, at a high level, comprises a RISC-V system and a custom memory controller that can perform PuM operations in real DDR3 chips. This repository contains all sources required to build PiDRAM and develop its prototype on the Xilinx ZC706 FPGA boards.

About

PiDRAM is the first flexible end-to-end framework that enables system integration studies and evaluation of real Processing-using-Memory techniques. Prototype on a RISC-V rocket chip system implemented on an FPGA. Described in our preprint: https://arxiv.org/abs/2111.00082

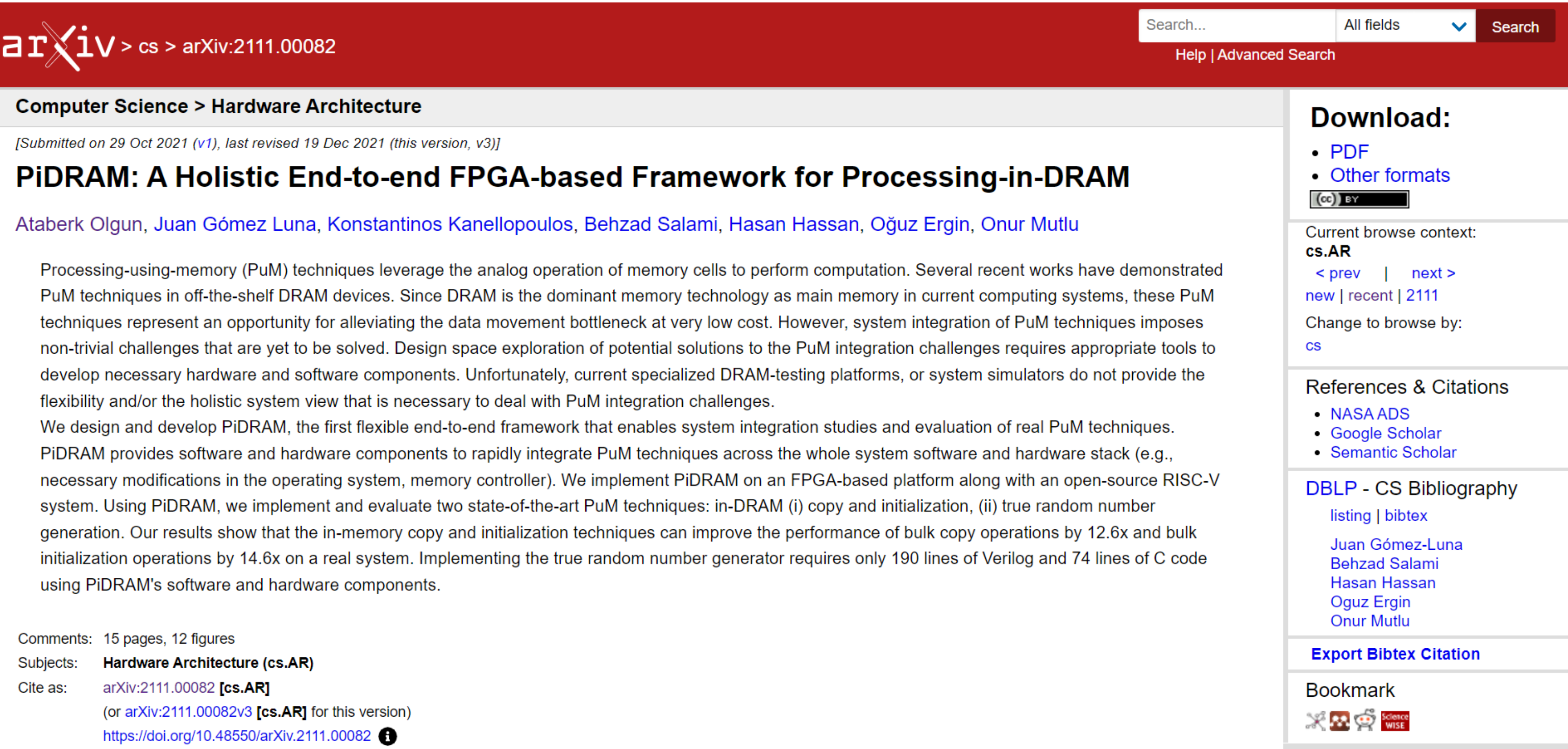
Readme
21 stars
3 watching
2 forks

Releases

No releases published
Create a new release

Extended Version on ArXiv

https://arxiv.org/abs/2111.00082



Computer Science > Hardware Architecture

[Submitted on 29 Oct 2021 (v1), last revised 19 Dec 2021 (this version, v3)]

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

Processing-using-memory (PuM) techniques leverage the analog operation of memory cells to perform computation. Several recent works have demonstrated PuM techniques in off-the-shelf DRAM devices. Since DRAM is the dominant memory technology as main memory in current computing systems, these PuM techniques represent an opportunity for alleviating the data movement bottleneck at very low cost. However, system integration of PuM techniques imposes non-trivial challenges that are yet to be solved. Design space exploration of potential solutions to the PuM integration challenges requires appropriate tools to develop necessary hardware and software components. Unfortunately, current specialized DRAM-testing platforms, or system simulators do not provide the flexibility and/or the holistic system view that is necessary to deal with PuM integration challenges.
We design and develop PiDRAM, the first flexible end-to-end framework that enables system integration studies and evaluation of real PuM techniques. PiDRAM provides software and hardware components to rapidly integrate PuM techniques across the whole system software and hardware stack (e.g., necessary modifications in the operating system, memory controller). We implement PiDRAM on an FPGA-based platform along with an open-source RISC-V system. Using PiDRAM, we implement and evaluate two state-of-the-art PuM techniques: in-DRAM (i) copy and initialization, (ii) true random number generation. Our results show that the in-memory copy and initialization techniques can improve the performance of bulk copy operations by 12.6x and bulk initialization operations by 14.6x on a real system. Implementing the true random number generator requires only 190 lines of Verilog and 74 lines of C code using PiDRAM's software and hardware components.

Comments:	15 pages, 12 figures
Subjects:	**Hardware Architecture (cs.AR)**
Cite as:	arXiv:2111.00082 **[cs.AR]**
	(or arXiv:2111.00082v3 **[cs.AR]** for this version)
	https://doi.org/10.48550/arXiv.2111.00082 ℹ

Download:

- PDF
- Other formats

(cc) BY

Current browse context:
cs.AR
< prev | next >
new | recent | 2111
Change to browse by:
cs

References & Citations

- NASA ADS
- Google Scholar
- Semantic Scholar

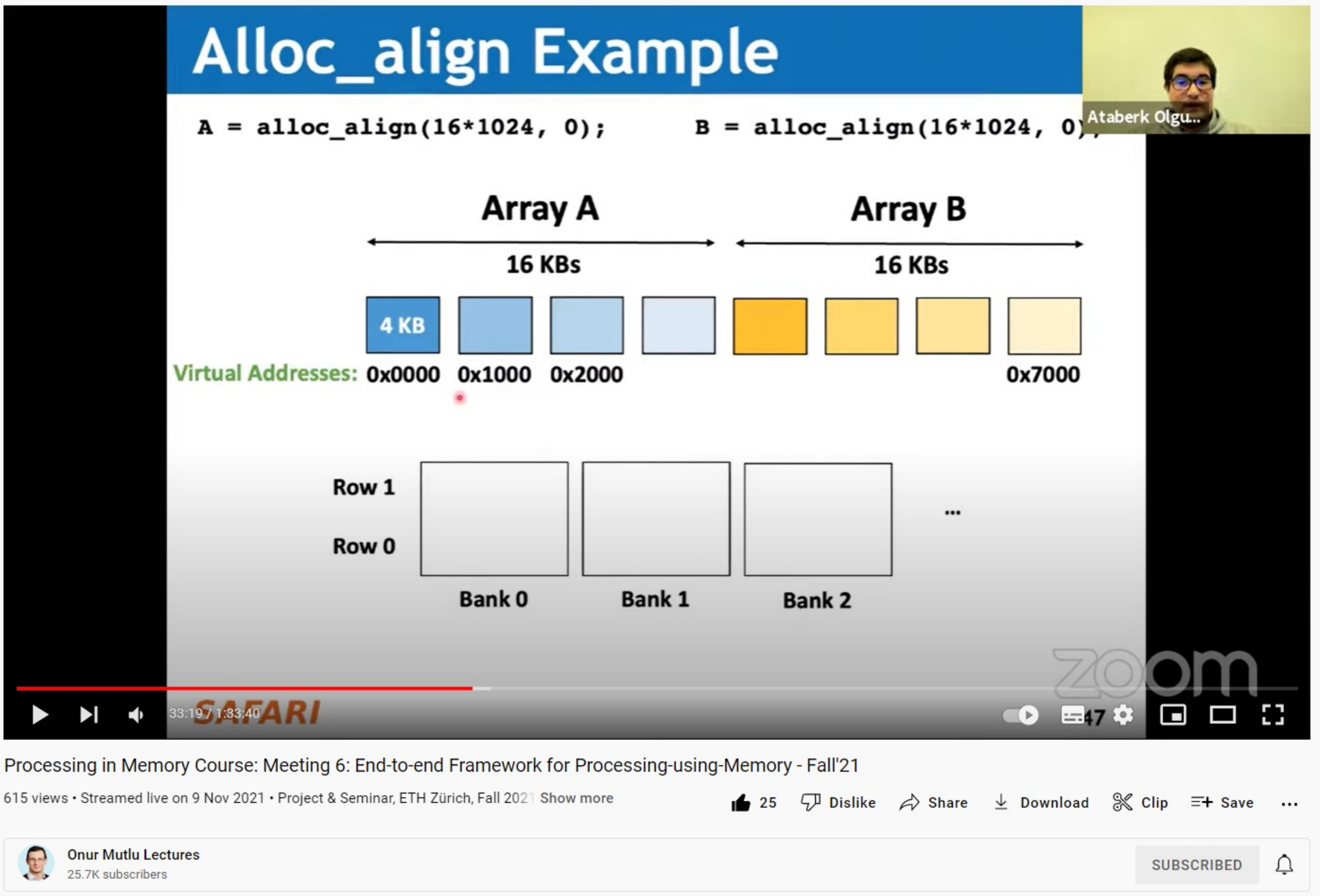
DBLP - CS Bibliography
listing | bibtex
Juan Gómez-Luna
Behzad Salami
Hasan Hassan
Oguz Ergin
Onur Mutlu

Export Bibtex Citation

Bookmark

Longer Talk + Tutorial on Youtube

https://youtu.be/s_z_S6FYpC8



Alloc_align Example

```
A = alloc_align(16*1024, 0);        B = alloc_align(16*1024, 0);
```

Processing in Memory Course: Meeting 6: End-to-end Framework for Processing-using-Memory - Fall'21

615 views • Streamed live on 9 Nov 2021 • Project & Seminar, ETH Zürich, Fall 2021 Show more

Year III Results (2022 Annual Review 2)

- SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping [ISCA 2022]

- GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis [ASPLOS 2022]

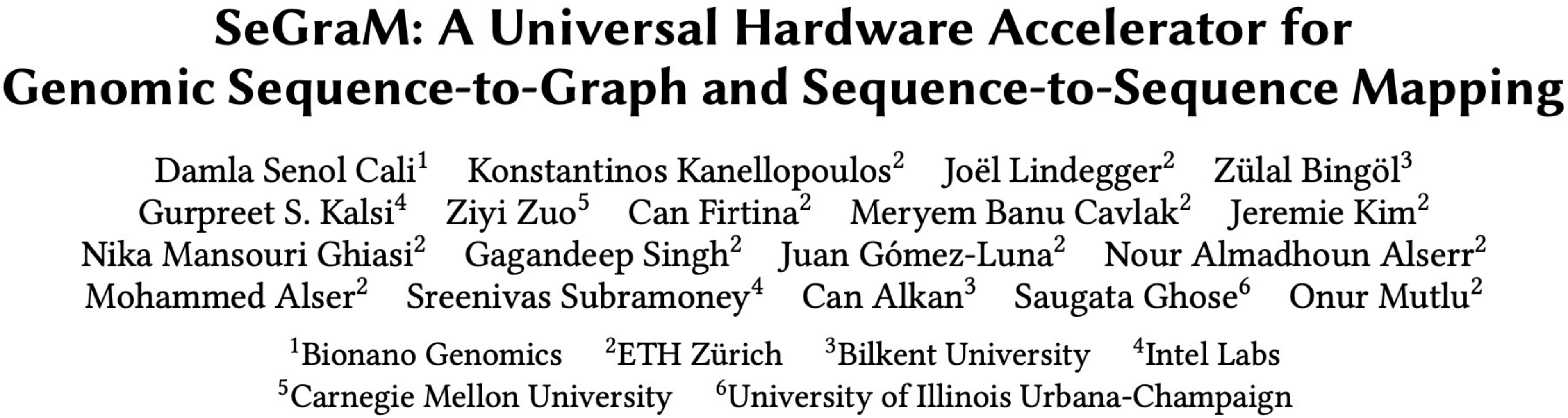
- Algorithmic Improvement and GPU Acceleration of the GenASM Algorithm [HICOMB 2022]

- Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design [ICDE 2022]

- Flash-Cosmos: In-Flash Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory [MICRO 2022]

Accelerating Sequence-to-Graph Mapping

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



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

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

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

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

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

Damla Senol Cali, Ph.D.

damlasenolcali@gmail.com
https://damlasenolcali.github.io/

Konstantinos Kanellopoulos, Joel Lindegger, Zulal Bingol, Gurpreet S. Kalsi, Ziyi Zuo, Can Firtina, Meryem Banu Cavlak, Jeremie S. Kim, Nika Mansouri Ghiasi, Gagandeep Singh, Juan Gomez-Luna, Nour Almadhoun Alserr, Mohammed Alser, Sreenivas Subramoney, Can Alkan, Saugata Ghose, Onur Mutlu

Carnegie Mellon

ETH zürich

Bilkent University

intel

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

SAFARI

Genome Sequence Analysis

- Mapping the reads to a reference genome (i.e., ***read mapping***) is a *critical step* in genome sequence analysis

Linear Reference: ACGTACGT

Read: ACG**G**

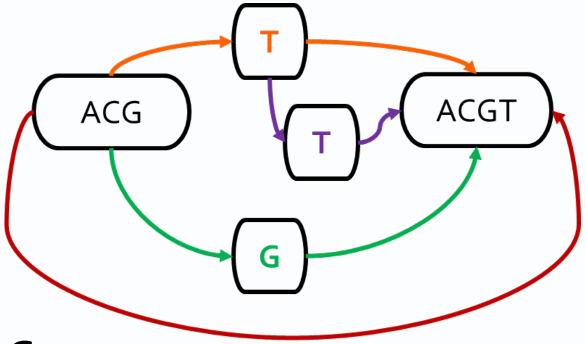
Alternative Sequence: ACG**G**ACGT

Alternative Sequence: ACG**TT**ACGT

Alternative Sequence: ACG–ACGT

Sequence-to-Sequence (S2S) Mapping

Graph-based Reference:



Read: ACGG

Sequence-to-Graph (S2G) Mapping

Sequence-to-graph mapping results in **notable quality improvements.**

However, it is a **more difficult** computational problem,

with **no prior hardware design.**

SeGraM: First Graph Mapping Accelerator

Our Goal:

Specialized, high-performance, scalable, and low-cost algorithm/hardware co-design that alleviates bottlenecks in **multiple steps** of sequence-to-graph mapping

SeGraM: *First universal algorithm/hardware co-designed genomic mapping accelerator* that can effectively and efficiently support:

- ❑ Sequence-to-graph mapping
- ❑ Sequence-to-sequence mapping
- ❑ Both short and long reads

Use Cases & Key Results

(1) Sequence-to-Graph (S2G) Mapping

- ❑ **5.9×/106×** speedup, **4.1×/3.0×** less power than **GraphAligner** for long and short reads, respectively (state-of-the-art **SW**)
- ❑ **3.9×/742×** speedup, **4.4×/3.2×** less power than **vg** for long and short reads, respectively (state-of-the-art **SW**)

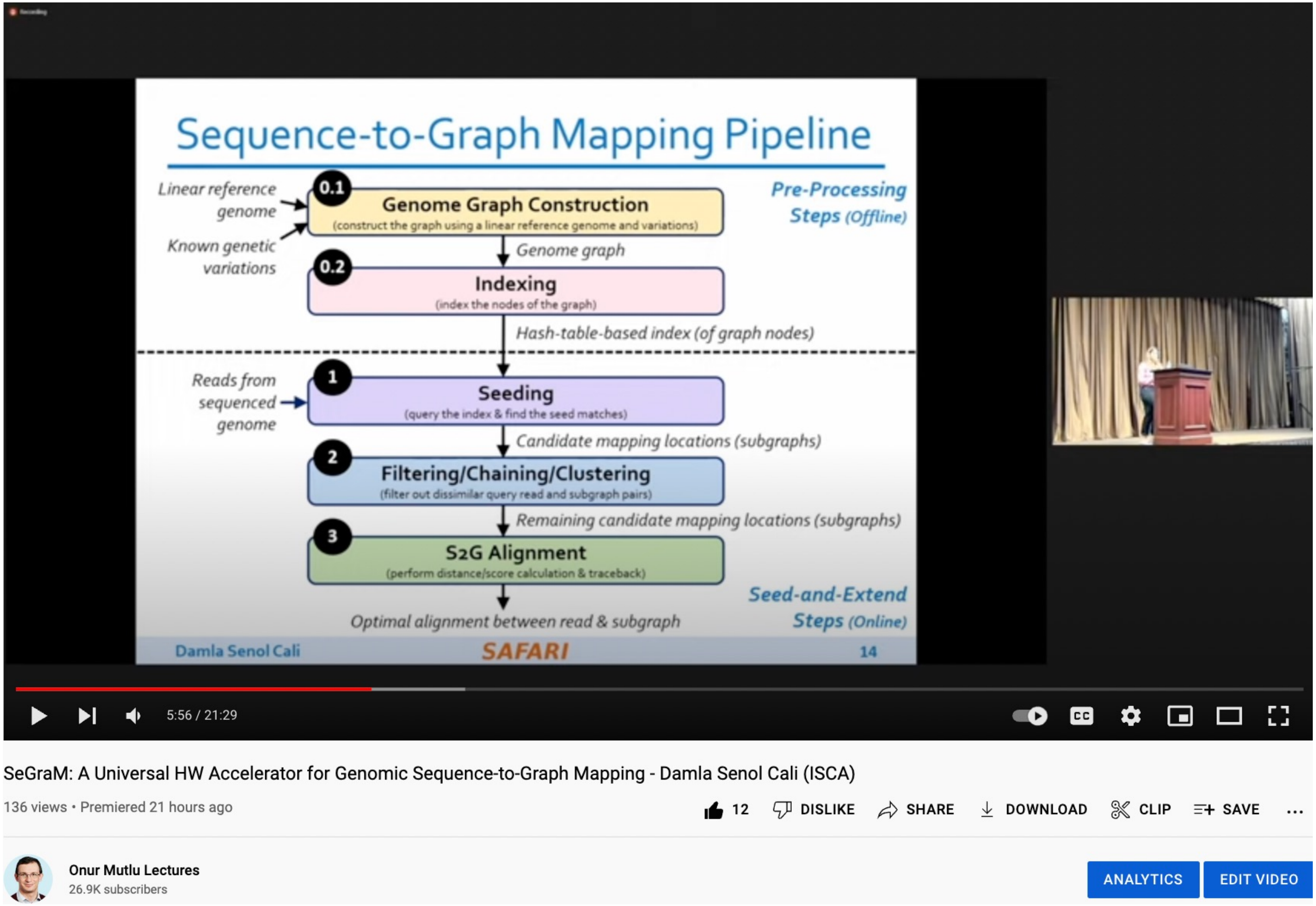
(2) Sequence-to-Graph (S2G) Alignment

- ❑ **41×–539×** speedup over **PaSGAL** with AVX-512 support (state-of-the-art **SW**)

(3) Sequence-to-Sequence (S2S) Alignment

- ❑ **1.2×/4.8×** higher throughput than **GenASM** and **GACT of Darwin** for long reads (state-of-the-art **HW**)
- ❑ **1.3×/2.4×** higher throughput than **GenASM** and **SillaX of GenAX** for short reads (state-of-the-art **HW**)

SeGraM Talk Video



Sequence-to-Graph Mapping Pipeline

Linear reference genome → **0.1** **Genome Graph Construction** (construct the graph using a linear reference genome and variations) ← Known genetic variations

Genome graph

0.2 **Indexing** (index the nodes of the graph)

Hash-table-based index (of graph nodes)

Pre-Processing Steps (Offline)

Reads from sequenced genome → **1** **Seeding** (query the index & find the seed matches)

Candidate mapping locations (subgraphs)

2 **Filtering/Chaining/Clustering** (filter out dissimilar query read and subgraph pairs)

Remaining candidate mapping locations (subgraphs)

3 **S2G Alignment** (perform distance/score calculation & traceback)

Optimal alignment between read & subgraph

Seed-and-Extend Steps (Online)

Damla Senol Cali SAFARI 14

5:56 / 21:29

SeGraM: A Universal HW Accelerator for Genomic Sequence-to-Graph Mapping - Damla Senol Cali (ISCA)

136 views • Premiered 21 hours ago

Onur Mutlu Lectures

26.9K subscribers

https://www.youtube.com/watch?v=gyjqYoyDP9s

Genome Graphs

Genome graphs:

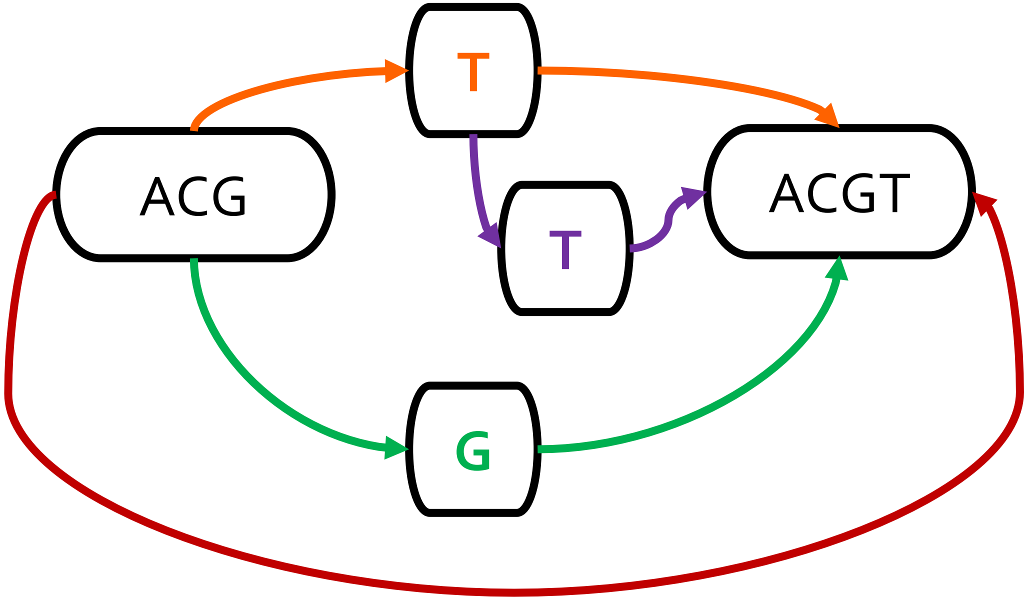
- ❑ Combine the linear reference genome with the known genetic variations in the entire population as a graph-based data structure
- ❑ Enable us to move away from aligning with a single linear reference genome (reference bias) and more accurately express the genetic diversity in a population

Sequence #1: ACGTACGT

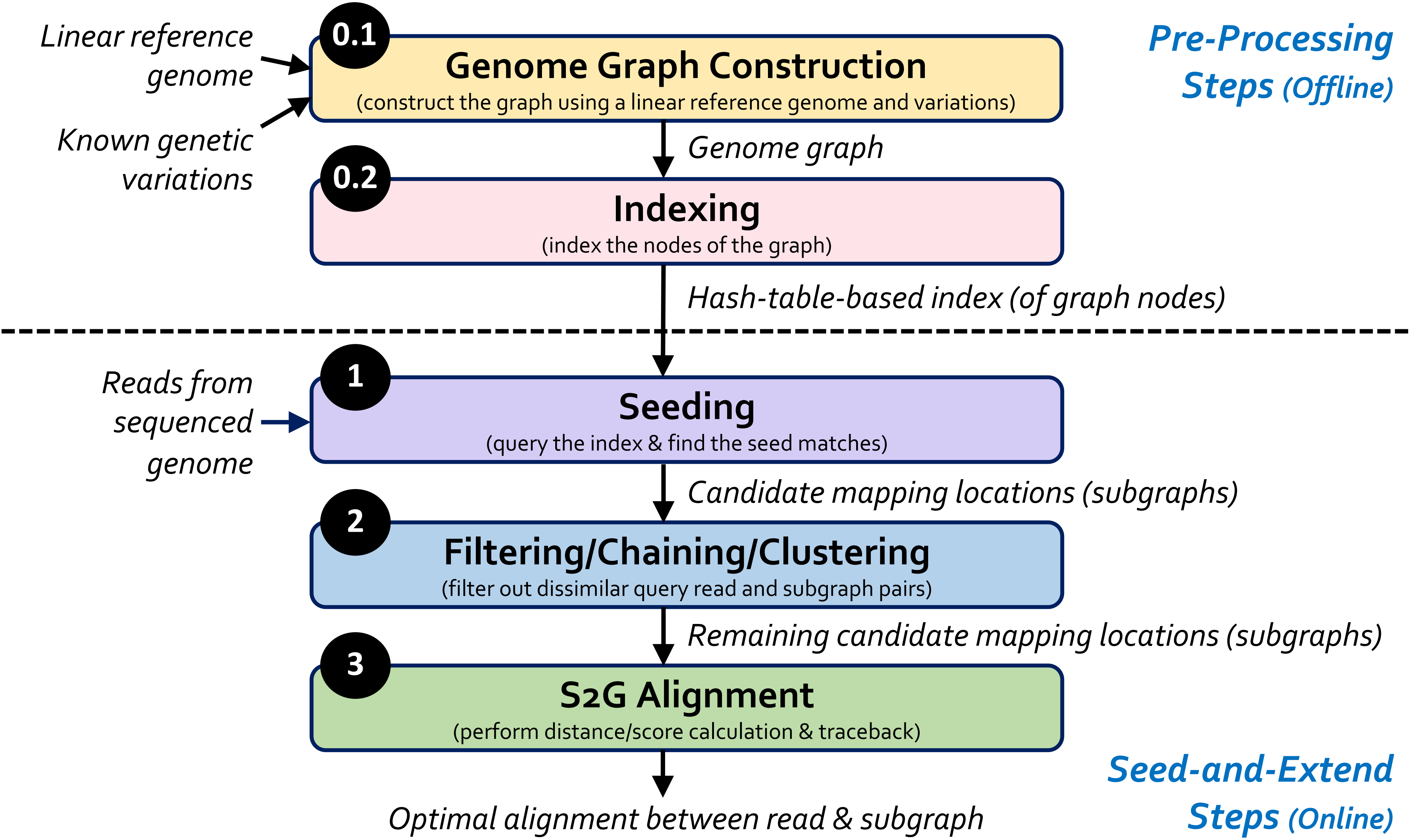
Sequence #2: ACGGACGT

Sequence #3: ACGTTACGT

Sequence #4: ACGACGT



Sequence-to-Graph Mapping Pipeline



Pre-Processing Steps (Offline)

Linear reference genome

Known genetic variations

0.1 **Genome Graph Construction**
(construct the graph using a linear reference genome and variations)

Genome graph

0.2 **Indexing**
(index the nodes of the graph)

Hash-table-based index (of graph nodes)

Reads from sequenced genome

1 **Seeding**
(query the index & find the seed matches)

Candidate mapping locations (subgraphs)

2 **Filtering/Chaining/Clustering**
(filter out dissimilar query read and subgraph pairs)

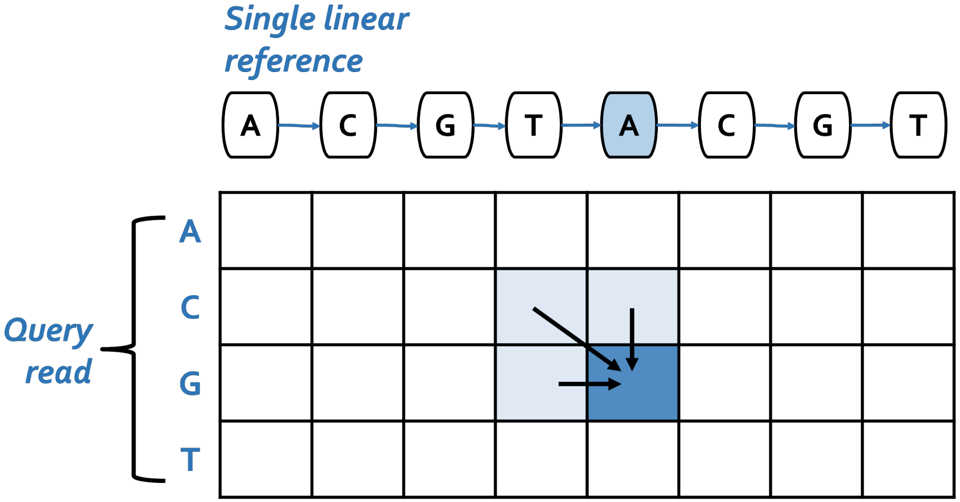
Remaining candidate mapping locations (subgraphs)

3 **S2G Alignment**
(perform distance/score calculation & traceback)

Optimal alignment between read & subgraph

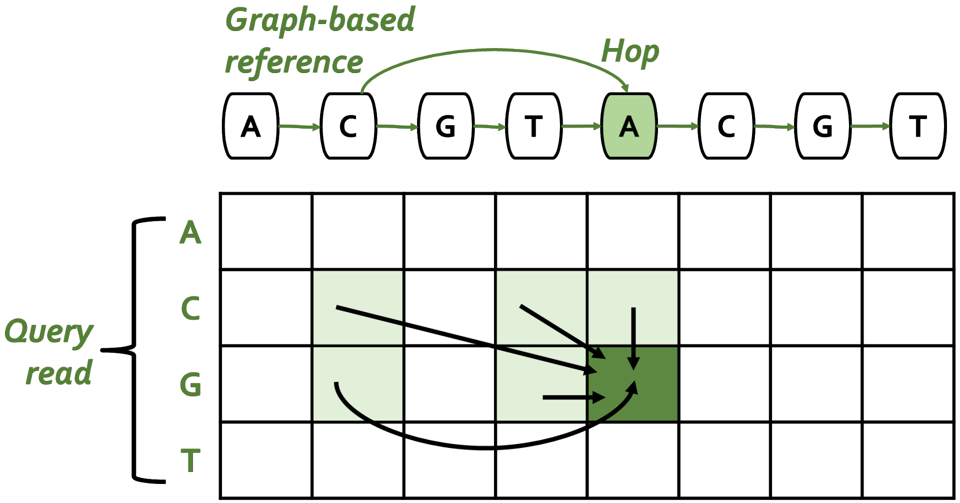
Seed-and-Extend Steps (Online)

S2S vs. S2G Alignment



Sequence-to-Sequence (S2S) Alignment

S2S vs. S2G Alignment



Sequence-to-Graph (S2G) Alignment

In contrast to S2S alignment,

S2G alignment must incorporate **non-neighboring characters**

as well whenever there is an edge (i.e., ***hop***)

from the non-neighboring character to the current character

Analysis of State-of-the-Art Tools

SW

Based on our analysis with **GraphAligner** and **vg:**

Observation 1: Alignment step is the bottleneck

Observation 2: Alignment suffers from high cache miss rates

Observation 3: Seeding suffers from the DRAM latency bottleneck

Observation 4: Baseline tools scale sublinearly

HW

Observation 5: Existing S2S mapping accelerators are unsuitable for the S2G mapping problem

Observation 6: Existing graph accelerators are unable to handle S2G alignment

SeGraM: Universal Genomic Mapping Accelerator

- ***First universal genomic mapping accelerator*** that can support *both* sequence-to-graph mapping and sequence-to-sequence mapping, for *both* short and long reads

- ***First algorithm/hardware co-design*** for accelerating sequence-to-graph mapping

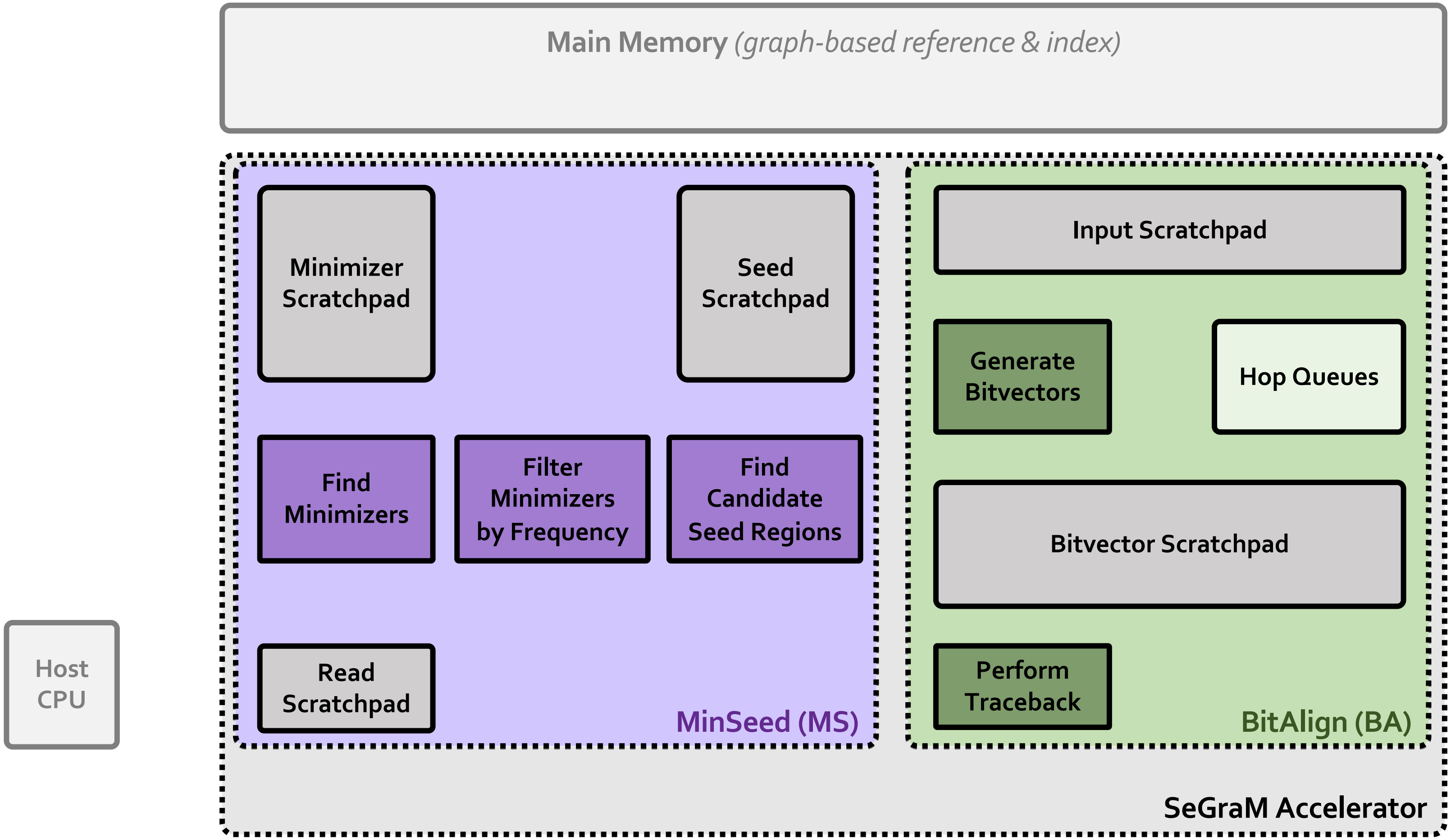
- We base SeGraM upon a minimizer-based seeding algorithm
- We propose a novel bitvector-based alignment algorithm to perform approximate string matching between a read and a graph-based reference genome

SW

- We co-design both algorithms with high-performance, scalable, and efficient hardware accelerators

HW

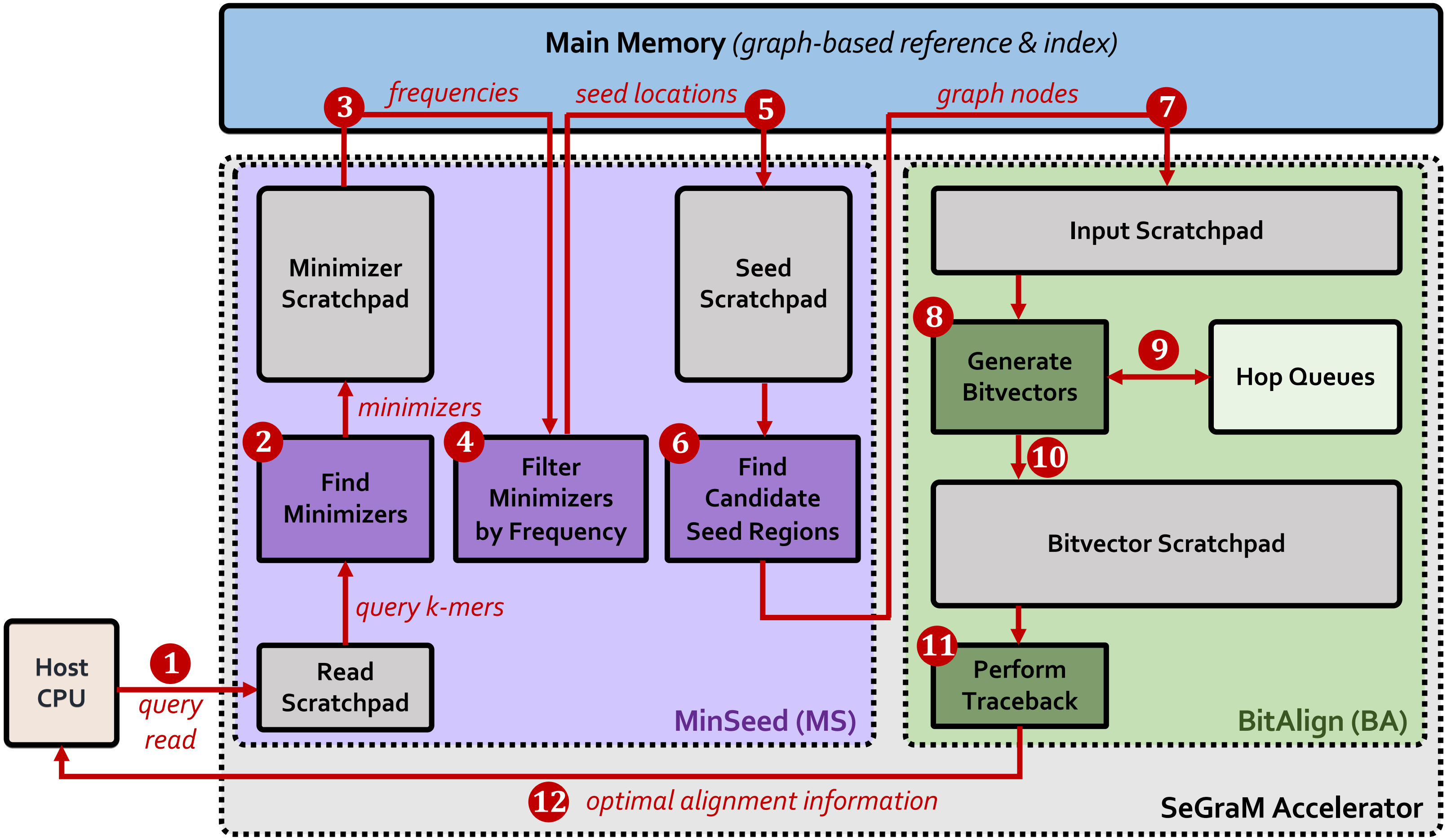
SeGraM Hardware Design



MinSeed: *first* hardware accelerator for **Min**imizer-based **Seed**ing

BitAlign: *first* hardware accelerator for (**Bit**vector-based) sequence-to-graph **Align**ment

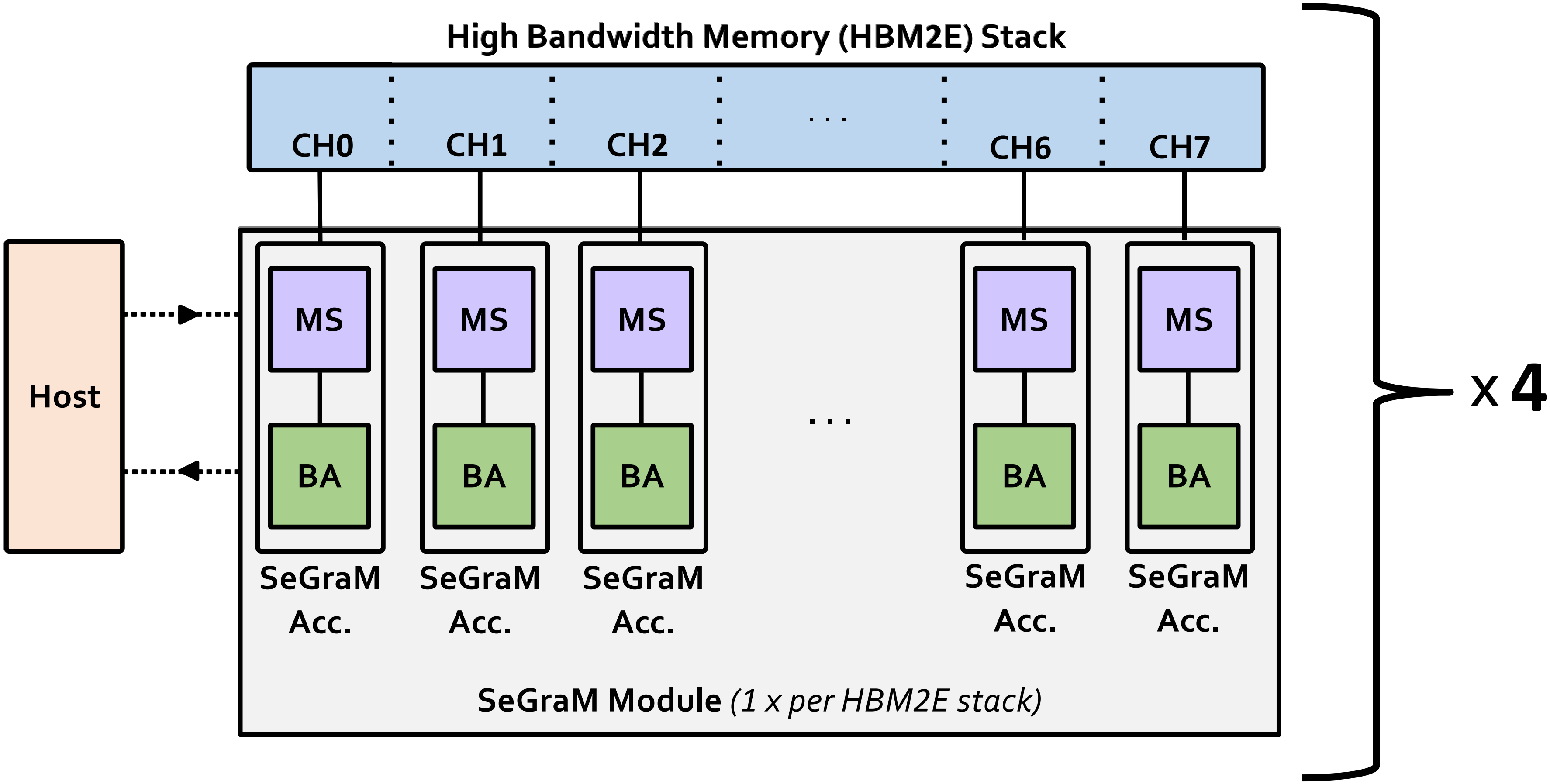
SeGraM Hardware Design



MinSeed: *first* hardware accelerator for **Min**imizer-based **Seed**ing

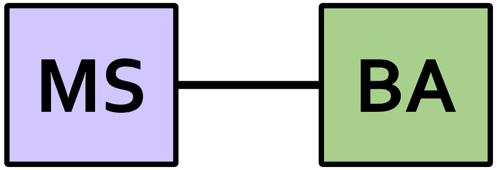
BitAlign: *first* hardware accelerator for (**Bit**vector-based) sequence-to-graph **Align**ment

Overall System Design of SeGraM

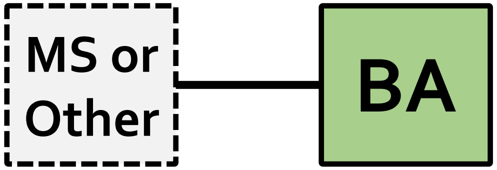


Use Cases of SeGraM

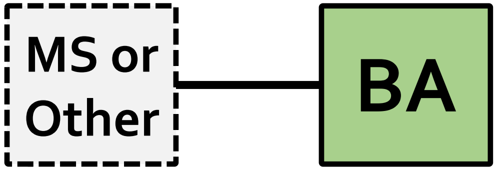
(1) Sequence-to-Graph Mapping



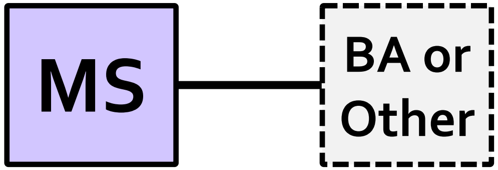
(2) Sequence-to-Graph Alignment



(3) Sequence-to-Sequence Alignment



(4) Seeding

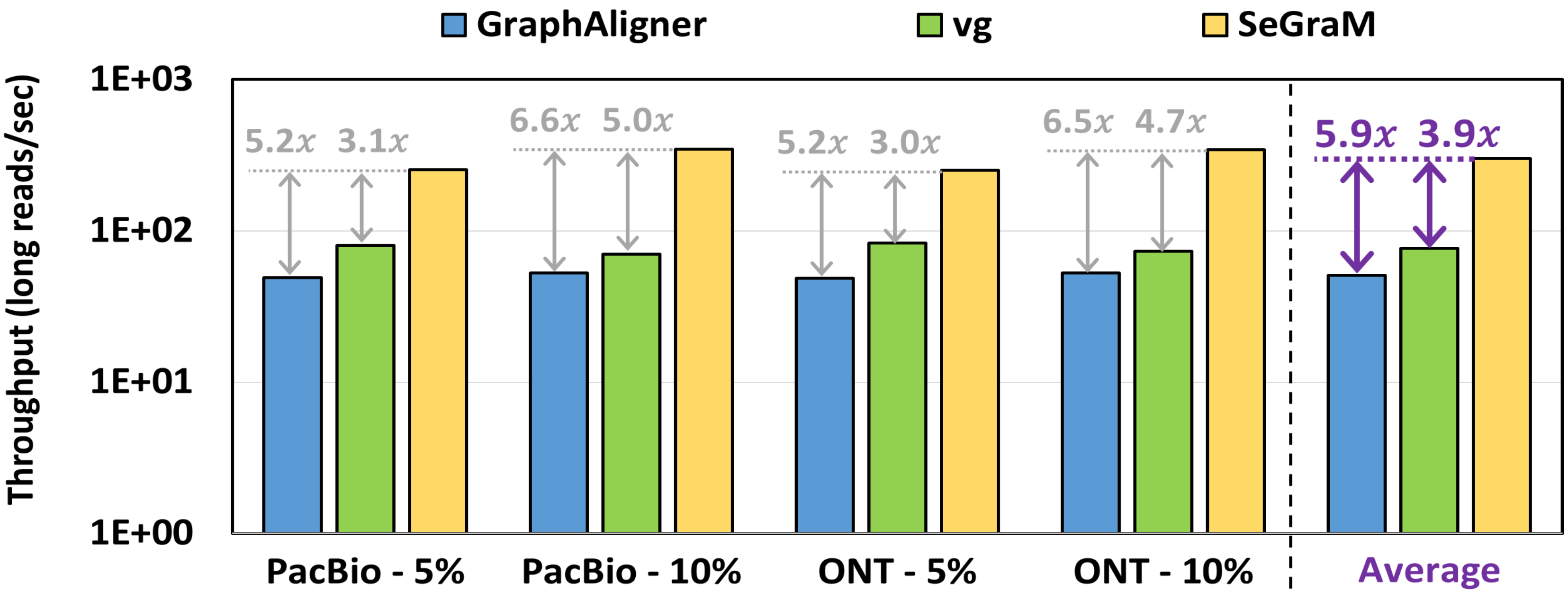


Key Results – Area & Power

- Based on our **synthesis** of **MinSeed** and **BitAlign** accelerator datapaths using the Synopsys Design Compiler with a **28nm** process (**@ 1GHz):**

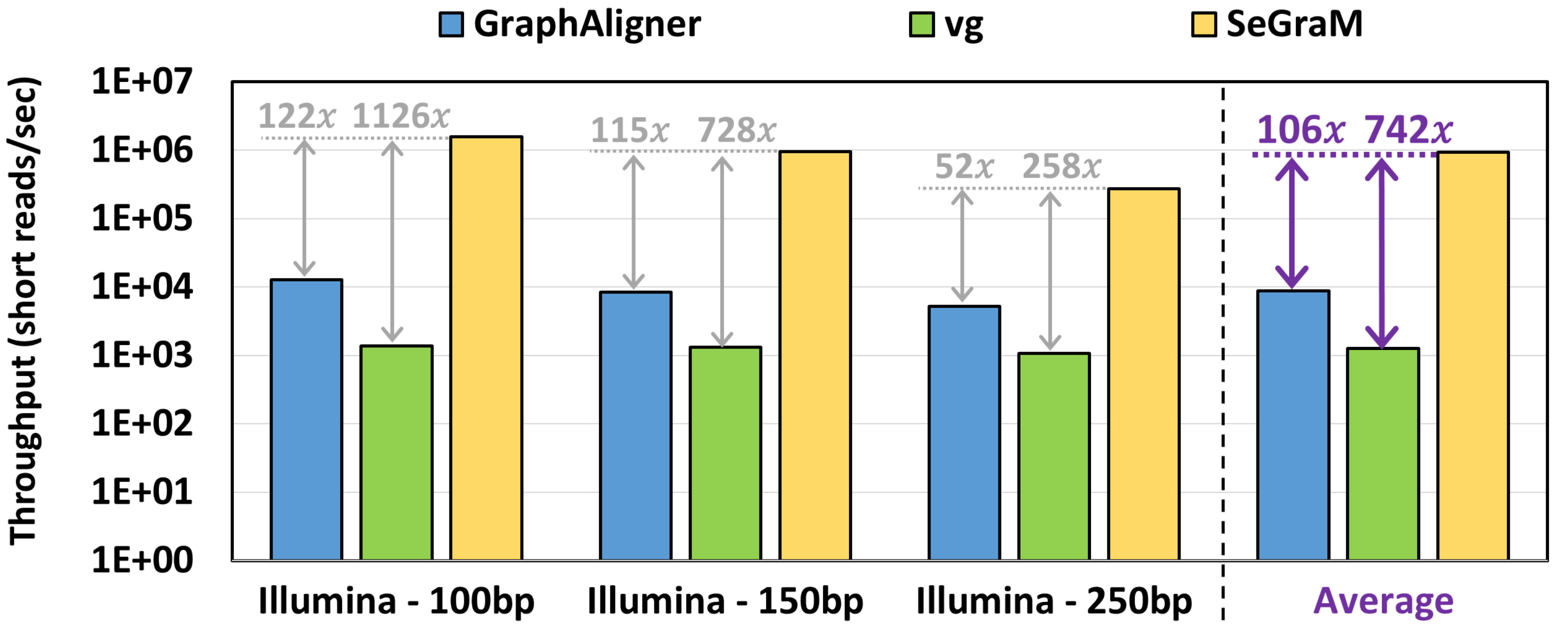
Component	Area (mm^2)	Power (mW)
MinSeed – Logic	0.017	10.8
Read Scratchpad (6 kB)	0.012	7.9
Minimizer Scratchpad (40 kB)	0.055	22.7
Seed Scratchpad (4 kB)	0.008	6.4
BitAlign – Edit Distance Calculation Logic with Hop Queue Registers (64 PEs)	0.393	378.0
BitAlign – Traceback Logic	0.020	2.7
Input Scratchpad (24 kB)	0.033	13.3
Bitvector Scratchpads (128 kB)	0.329	316.2
Total – 1 SeGraM Accelerator	**0.867**	**758.0 (0.8 W)**
Total – 4 SeGraM Modules (32 SeGraM Accelerators)	**27.744**	**24.3 W**
HBM2E (4 stacks)	**--**	**3.8 W**

Key Results – SeGraM with Long Reads



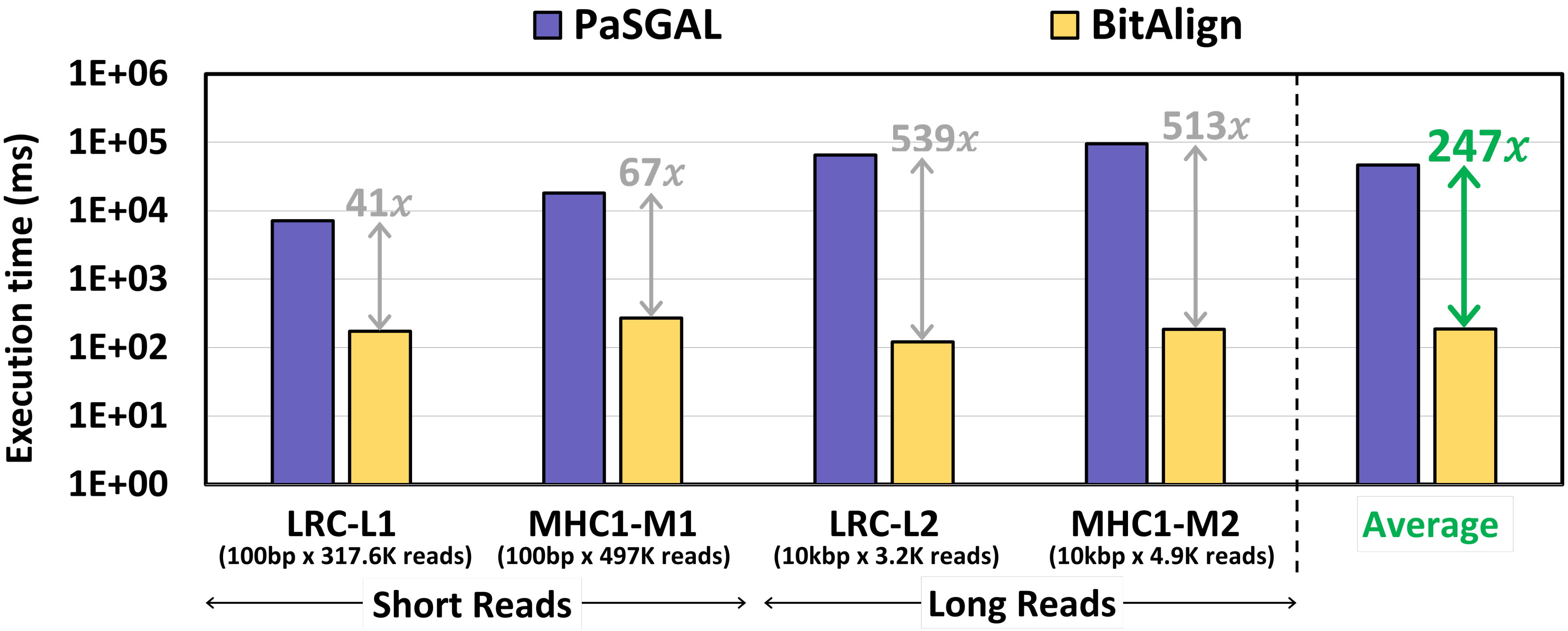
SeGraM provides **5.9× and 3.9× throughput improvement** over GraphAligner and vg, while **reducing the power consumption by 4.1× and 4.4×**

Key Results – SeGraM with Short Reads



SeGraM provides **106× and 742× throughput improvement** over GraphAligner and vg, while **reducing the power consumption by 3.0× and 3.2×**

Key Results – BitAlign (S2G Alignment)



BitAlign provides **41×-539× speedup** over PaSGAL

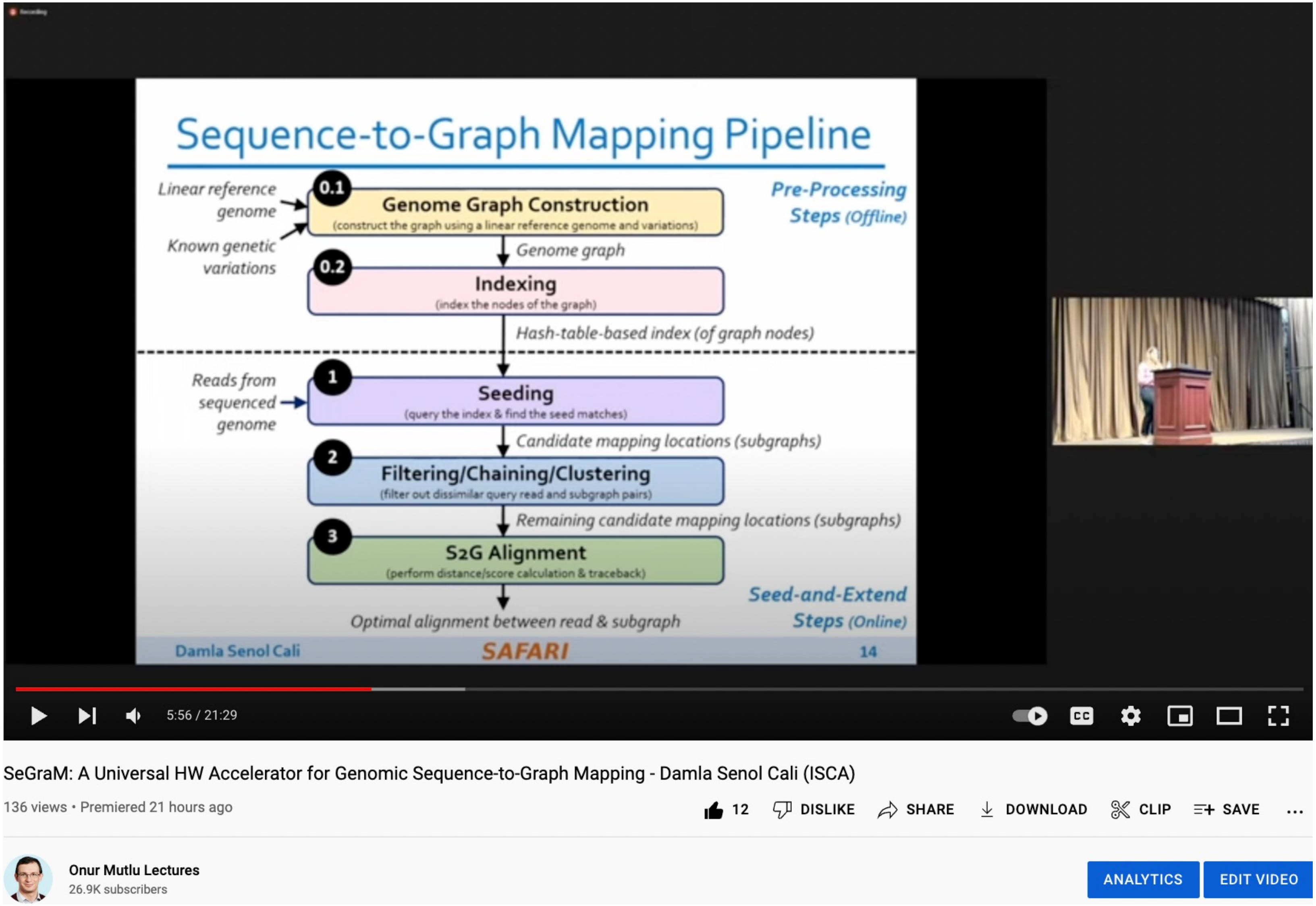
Key Results – BitAlign (S2S Alignment)

- ❑ BitAlign can also be used for sequence-to-sequence alignment
  - ○ The cost of more functionality: extra hop queue registers
  - ○ We do *not* sacrifice any performance
- ❑ **For long reads (over GACT of Darwin and GenASM):**
  - ○ **4.8× and 1.2×** throughput improvement,
  - ○ **2.7× and 7.5×** higher power consumption, and
  - ○ **1.5× and 2.6×** higher area overhead
- ❑ **For short reads (over SillaX of GenAx and GenASM):**
  - ○ **2.4× and 1.3×** throughput improvement

Conclusion

- **SeGraM:** *First universal algorithm/hardware co-designed genomic mapping accelerator that supports:*
  - Sequence-to-graph (S2G) & sequence-to-sequence (S2S) mapping
  - Short & long reads
  - **MinSeed:** *First minimizer-based seeding accelerator*
  - **BitAlign:** *First (bitvector-based) S2G alignment accelerator*
- SeGraM **supports multiple use cases:**
  - End-to-end S2G mapping
  - S2G alignment
  - S2S alignment
  - Seeding
- SeGraM outperforms state-of-the-art software & hardware solutions

SeGraM Talk Video



Sequence-to-Graph Mapping Pipeline

Linear reference genome → **0.1 Genome Graph Construction** (construct the graph using a linear reference genome and variations) ← Known genetic variations

Genome graph

0.2 Indexing (index the nodes of the graph)

Hash-table-based index (of graph nodes)

Pre-Processing Steps (Offline)

Reads from sequenced genome → **1 Seeding** (query the index & find the seed matches)

Candidate mapping locations (subgraphs)

2 Filtering/Chaining/Clustering (filter out dissimilar query read and subgraph pairs)

Remaining candidate mapping locations (subgraphs)

3 S2G Alignment (perform distance/score calculation & traceback)

Optimal alignment between read & subgraph

Seed-and-Extend Steps (Online)

Damla Senol Cali SAFARI 14

5:56 / 21:29

SeGraM: A Universal HW Accelerator for Genomic Sequence-to-Graph Mapping - Damla Senol Cali (ISCA)

136 views • Premiered 21 hours ago

Onur Mutlu Lectures

26.9K subscribers

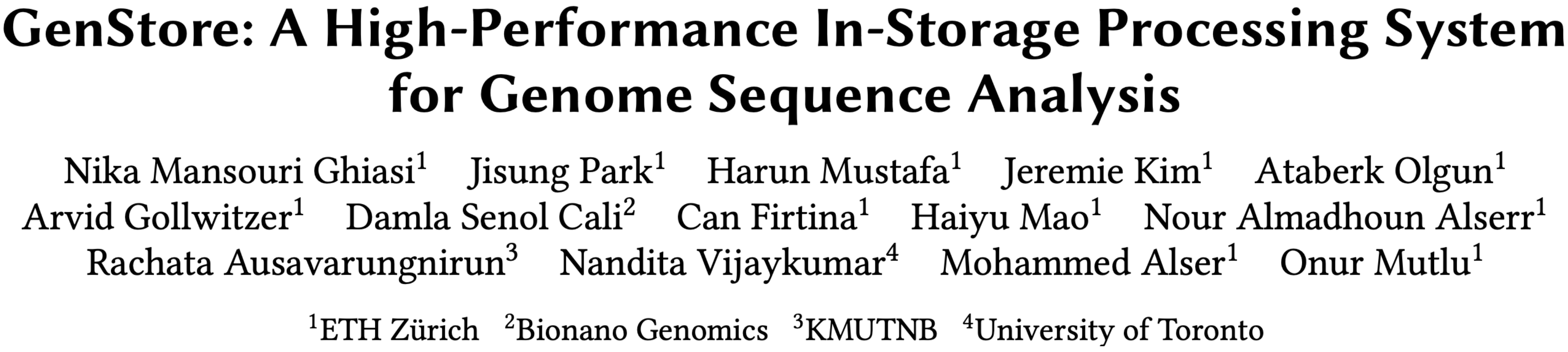
https://www.youtube.com/watch?v=gyjqYoyDP9s

Year III Results (2022 Annual Review 2)

- SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping [ISCA 2022]
- GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis [ASPLOS 2022]
- Algorithmic Improvement and GPU Acceleration of the GenASM Algorithm [HICOMB 2022]
- Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design [ICDE 2022]
- Flash-Cosmos: In-Flash Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory [MICRO 2022]

In-Storage Genomic Data Filtering [ASPLOS 2022]

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

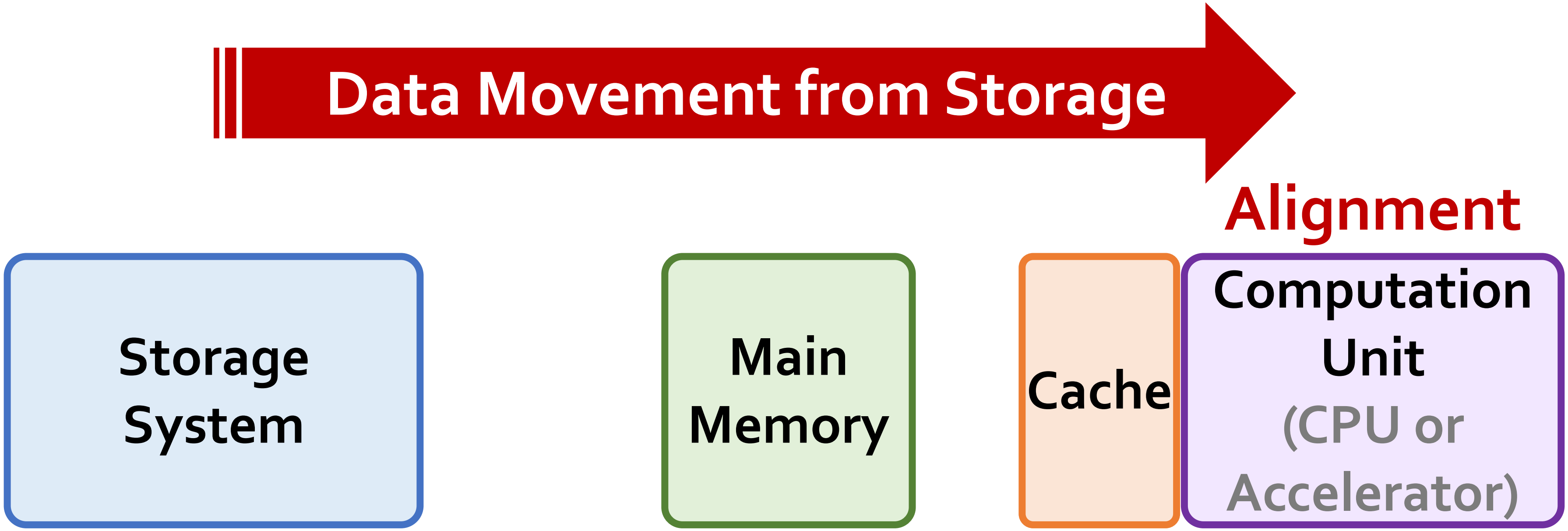


GenStore: A High-Performance In-Storage Processing System for Genome Sequence Analysis

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

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

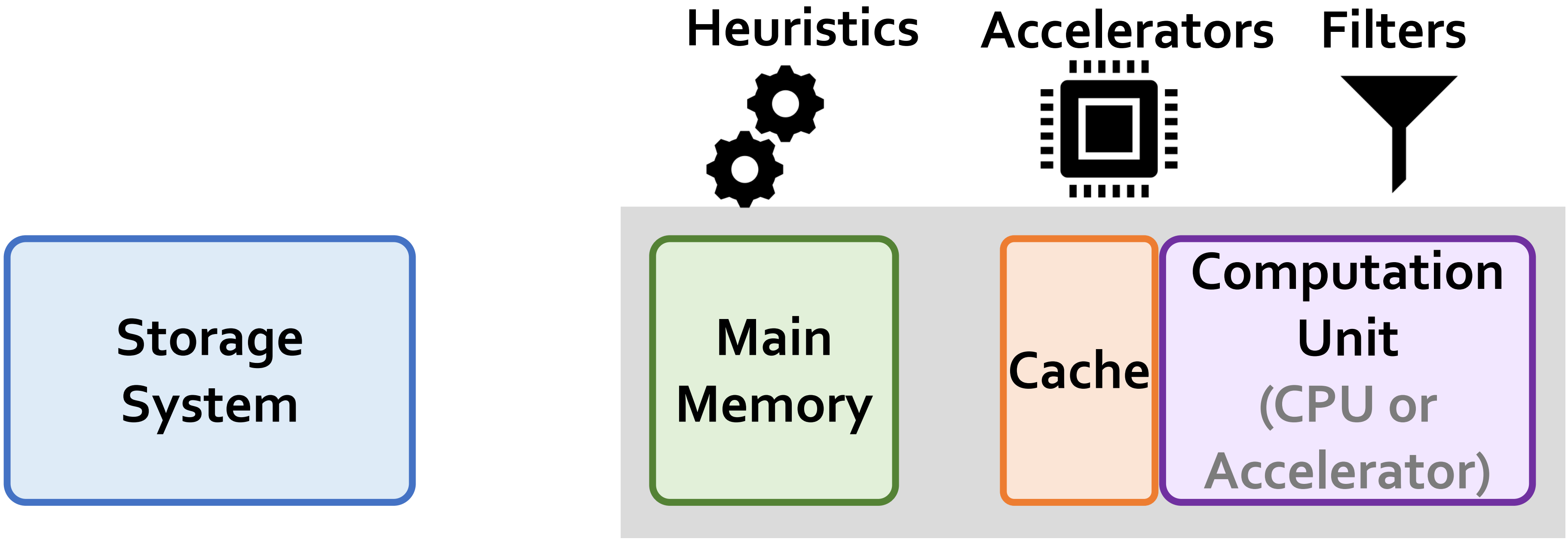
Genome Sequence Analysis



✗ Computation overhead

✗ Data movement overhead

Accelerating Genome Sequence Analysis



Heuristics

Accelerators

Filters

Storage System

Main Memory

Cache

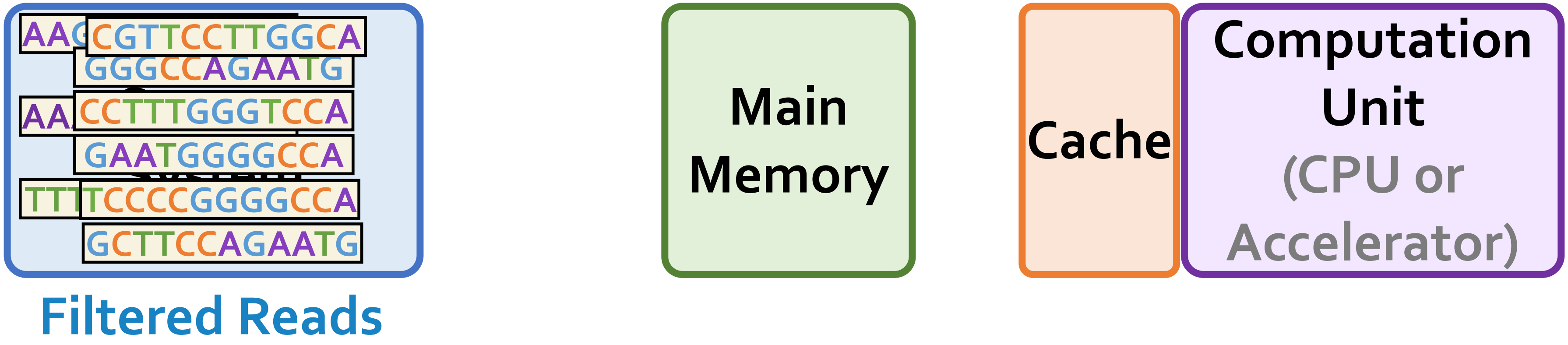
Computation Unit (CPU or Accelerator)

✓ Computation overhead

✗ Data movement overhead

Key Idea

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



Exactly-matching reads

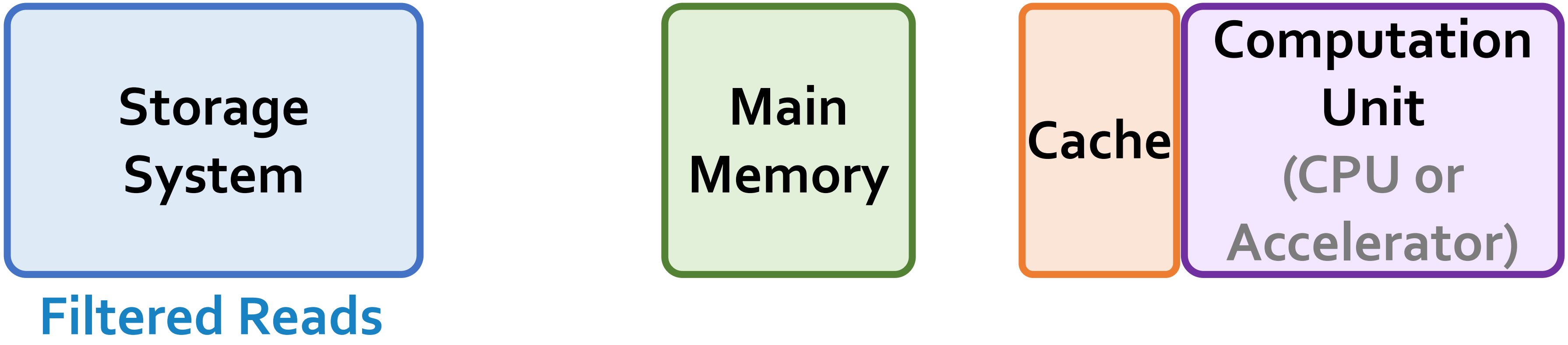
Do not need expensive approximate string matching during alignment

Non-matching reads

Do not have potential matching locations and can skip alignment

Challenges

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

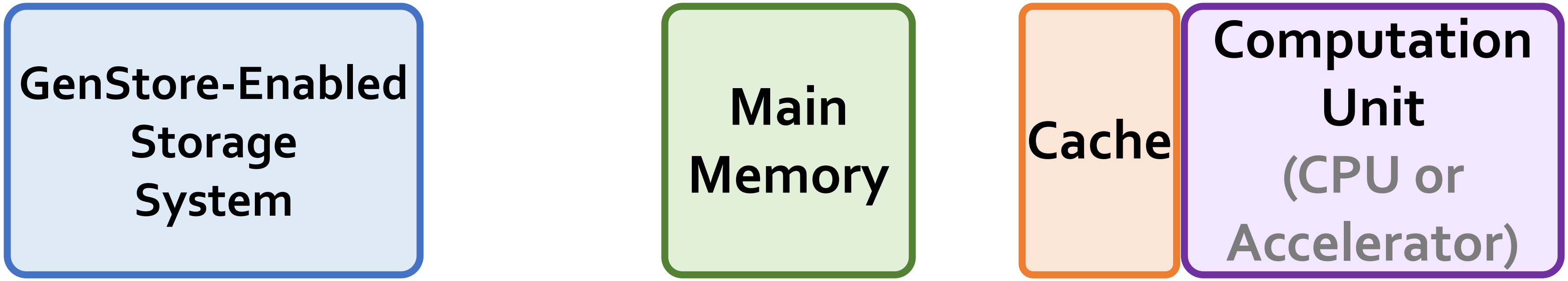


Read mapping workloads can exhibit different behavior

**There are limited hardware resources
in the storage system**

GenStore

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



GenStore-Enabled Storage System

Main Memory

Cache

Computation Unit (CPU or Accelerator)

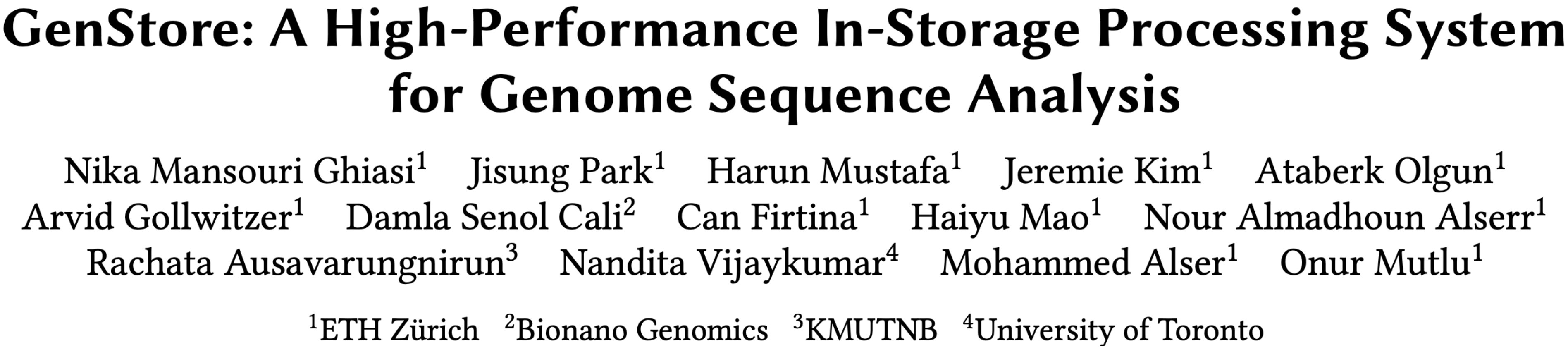
✓ Computation overhead

✓ Data movement overhead

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

In-Storage Genomic Data Filtering [ASPLOS 2022]

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

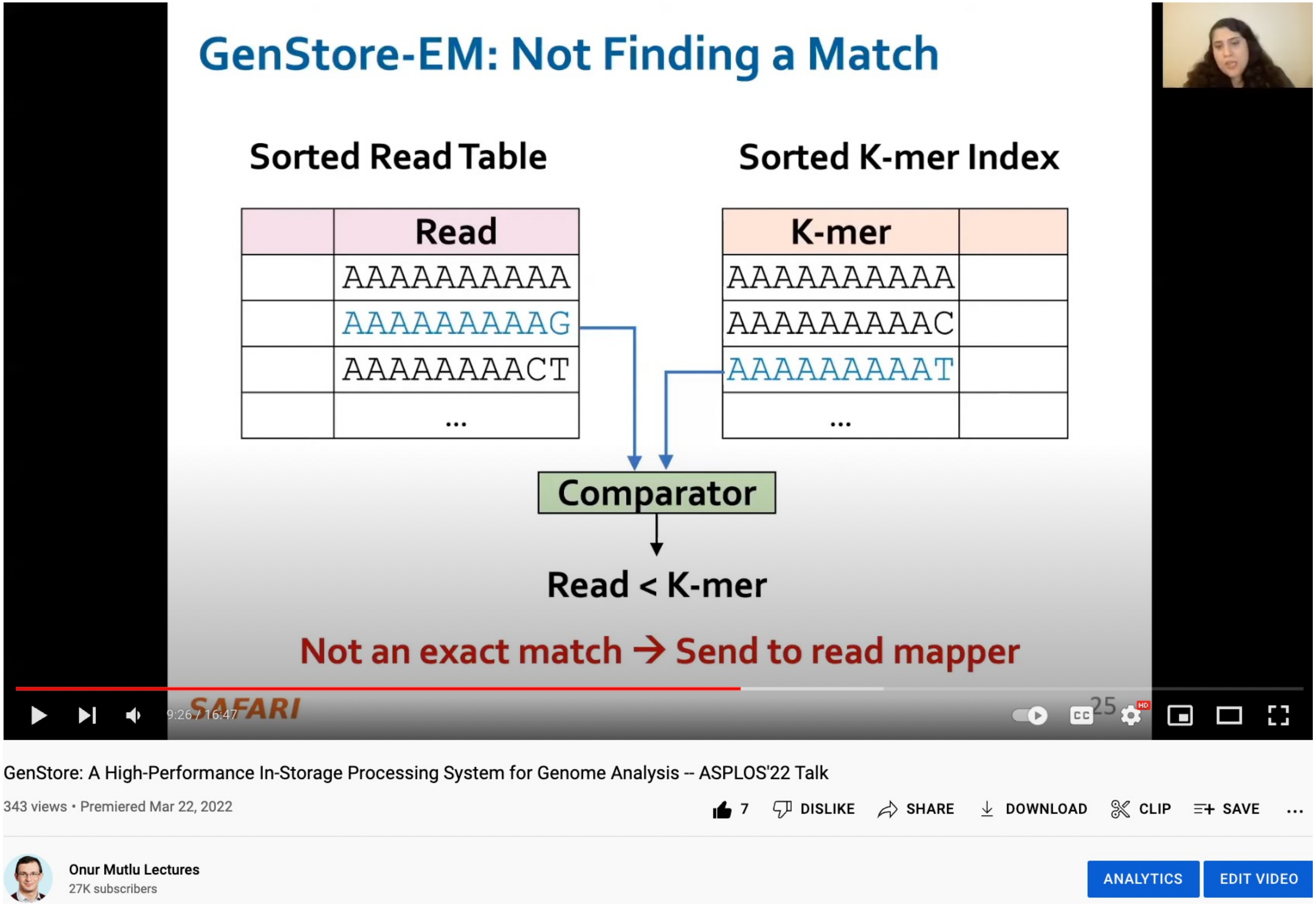


GenStore: A High-Performance In-Storage Processing System for Genome Sequence Analysis

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

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

GenStore Talk Video



GenStore-EM: Not Finding a Match

Sorted Read Table

	Read
	AAAAAAAAAA
	AAAAAAAAAG
	AAAAAAAACT
	...

Sorted K-mer Index

K-mer	
AAAAAAAAAA	
AAAAAAAAAC	
AAAAAAAAAT	
...	

Comparator

Read < K-mer

Not an exact match → Send to read mapper

GenStore: A High-Performance In-Storage Processing System for Genome Analysis -- ASPLOS'22 Talk

343 views • Premiered Mar 22, 2022

Onur Mutlu Lectures

27K subscribers

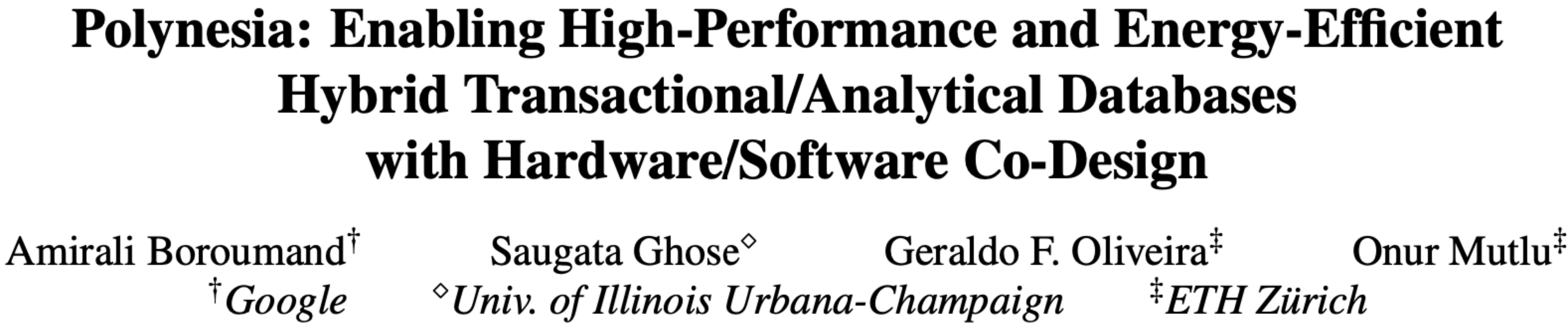
https://www.youtube.com/watch?v=bv7hgXOOMjk

Year III Results (2022 Annual Review 2)

- SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping [ISCA 2022]
- GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis [ASPLOS 2022]
- Algorithmic Improvement and GPU Acceleration of the GenASM Algorithm [HICOMB 2022]
- Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design [ICDE 2022]
- Flash-Cosmos: In-Flash Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory [MICRO 2022]

Accelerating HTAP Database Systems

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



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

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

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

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

Amirali Boroumand
Saugata Ghose
Geraldo F. Oliveira
Onur Mutlu

ICDE
2022

Executive Summary

- **Context:** Many applications need to perform real-time data analysis using an Hybrid Transactional/Analytical Processing (HTAP) system
  - An ideal HTAP system should have **three properties**: (1) **data freshness** and **consistency**, (2) **workload-specific optimization**, (3) **performance isolation**

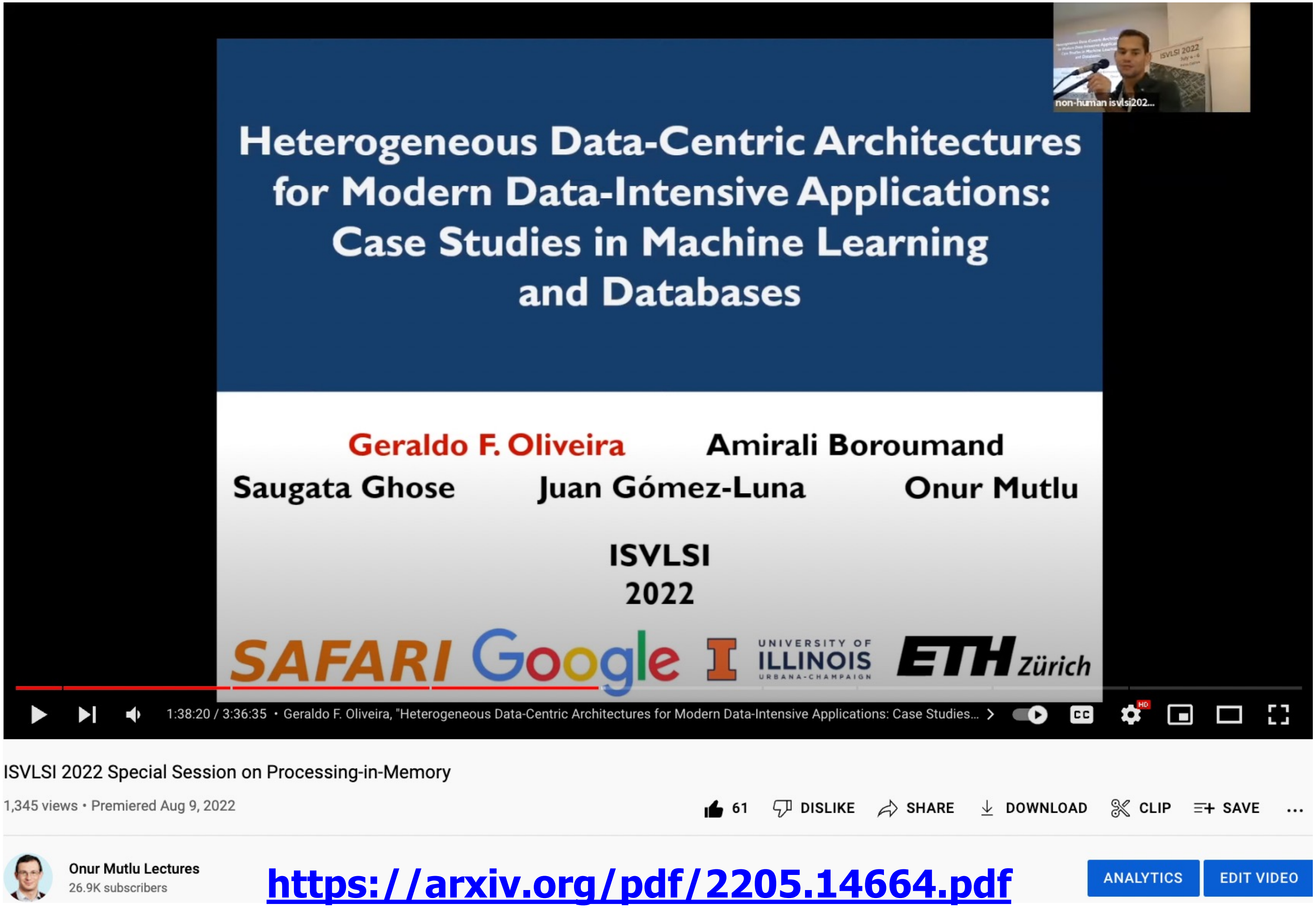
- **Problem:** Prior works **cannot achieve all properties** of an ideal HTAP system

- **Key Idea:** Divide the system into transactional and analytical **processing islands**
  - Enables **workload-specific optimizations** and **performance isolation**

- **Key Mechanism:** Polynesia, a novel hardware/software cooperative design for in-memory HTAP databases
  - Implements **custom algorithms and hardware** to reduce the costs of **data freshness** and **consistency**
  - Exploits **PIM** for analytical processing to alleviate **data movement**

- **Key Results:** Polynesia outperforms three state-of-the-art HTAP systems
  - Average transactional/analytical throughput improvements of **1.7x/3.7x**
  - **48%** reduction on energy consumption

Polynesia Talk Video (I)



Heterogeneous Data-Centric Architectures for Modern Data-Intensive Applications: Case Studies in Machine Learning and Databases

Geraldo F. Oliveira   Amirali Boroumand
Saugata Ghose   Juan Gómez-Luna   Onur Mutlu

ISVLSI
2022

SAFARI Google UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN ETH Zürich

1:38:20 / 3:36:35 • Geraldo F. Oliveira, "Heterogeneous Data-Centric Architectures for Modern Data-Intensive Applications: Case Studies...

ISVLSI 2022 Special Session on Processing-in-Memory

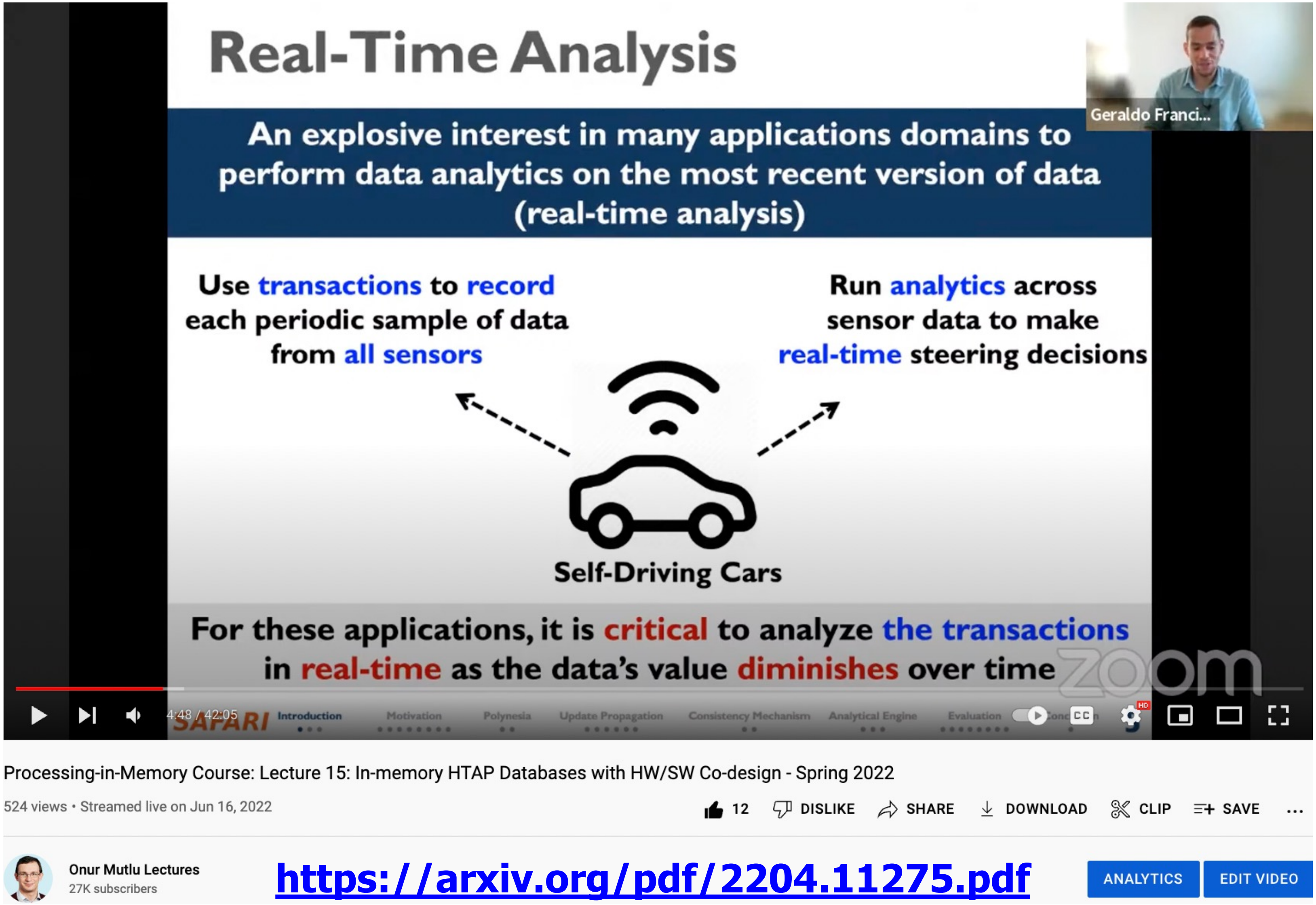
1,345 views • Premiered Aug 9, 2022

Onur Mutlu Lectures
26.9K subscribers

https://arxiv.org/pdf/2205.14664.pdf

https://www.youtube.com/watch?v=qeukNs5XI3g&t=5897s

Polynesia Talk Video (II)



Real-Time Analysis

An explosive interest in many applications domains to perform data analytics on the most recent version of data (real-time analysis)

Use transactions to record each periodic sample of data from all sensors

Run analytics across sensor data to make real-time steering decisions

Self-Driving Cars

For these applications, it is critical to analyze the transactions in real-time as the data's value diminishes over time

Processing-in-Memory Course: Lecture 15: In-memory HTAP Databases with HW/SW Co-design - Spring 2022

524 views • Streamed live on Jun 16, 2022

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

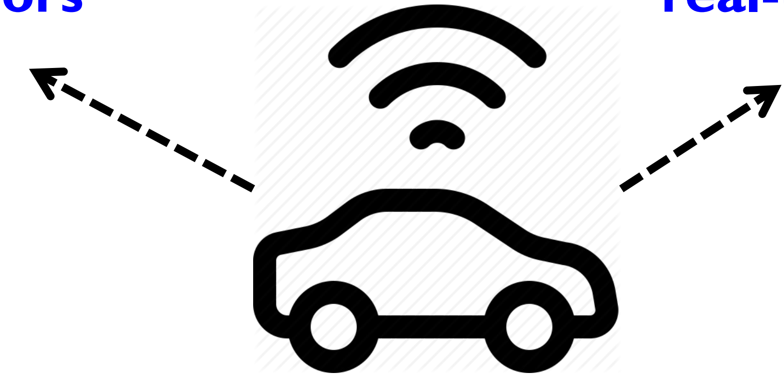
https://www.youtube.com/watch?v=1HkXy3g6FF4

Real-Time Analysis

Increasing interest in many applications domains to perform data analytics on the most recent version of data (real-time analysis)

Use transactions to record each periodic sample of data from all sensors

Run analytics across sensor data to make real-time steering decisions

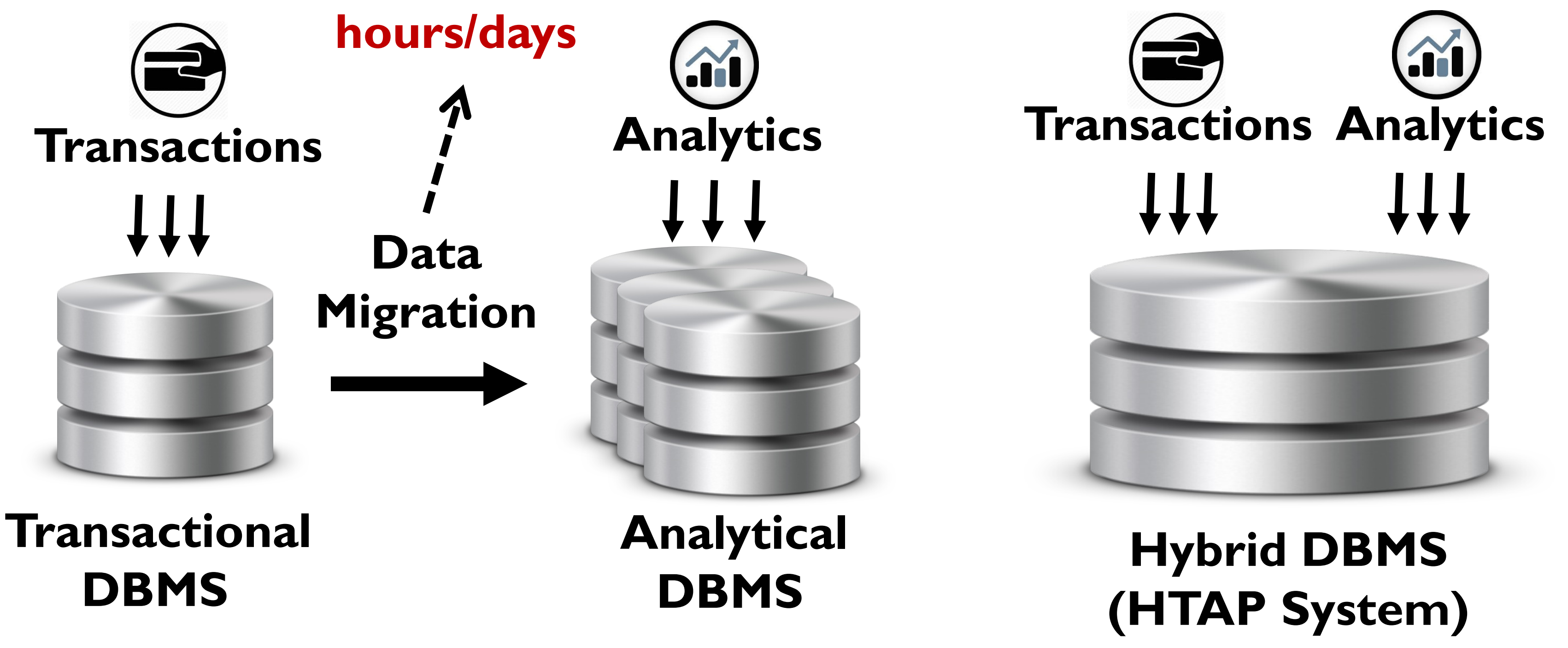


Self-Driving Cars

For these applications, it is critical to analyze the transactions in real-time as the data's value diminishes over time

HTAP: Supporting Real-Time Analysis

Traditionally, new transactions (updates) are propagated to the analytical database using a periodic and costly process



To support real-time analysis: a single hybrid DBMS is used to execute both transactional and analytical workloads

Ideal HTAP System Properties

An ideal HTAP system should have three properties:

1 **Workload-Specific Optimizations**

- Transactional and analytical workloads must benefit from their own specific optimizations

2 **Data Freshness and Consistency Guarantees**

- Guarantee access to the most recent version of data for analytics while ensuring that transactional and analytical workloads have a consistent view of data

3 **Performance Isolation**

- Latency and throughput of transactional and analytical workloads are the same as if they were run in isolation

Achieving all three properties at the same time is very challenging

Problem and Goal

Problems:

1 State-of-the-art HTAP systems do not achieve all of the desired HTAP properties

2 Data freshness and consistency mechanisms are data-intensive and cause a drastic reduction in throughput

3 These systems fail to provide performance isolation because of high resource contention

Goal:

4 Take advantage of custom algorithm and processing-in-memory (PIM) to address these challenges

Polynesia

Key idea: **partition computing resources into two types of isolated and specialized processing islands**

Isolating transactional islands from analytical islands allows us to:

1 **Apply workload-specific optimizations to each island**

2 **Avoid high resource contention**

3 **Design efficient data freshness and consistency mechanisms without incurring high data movement costs**

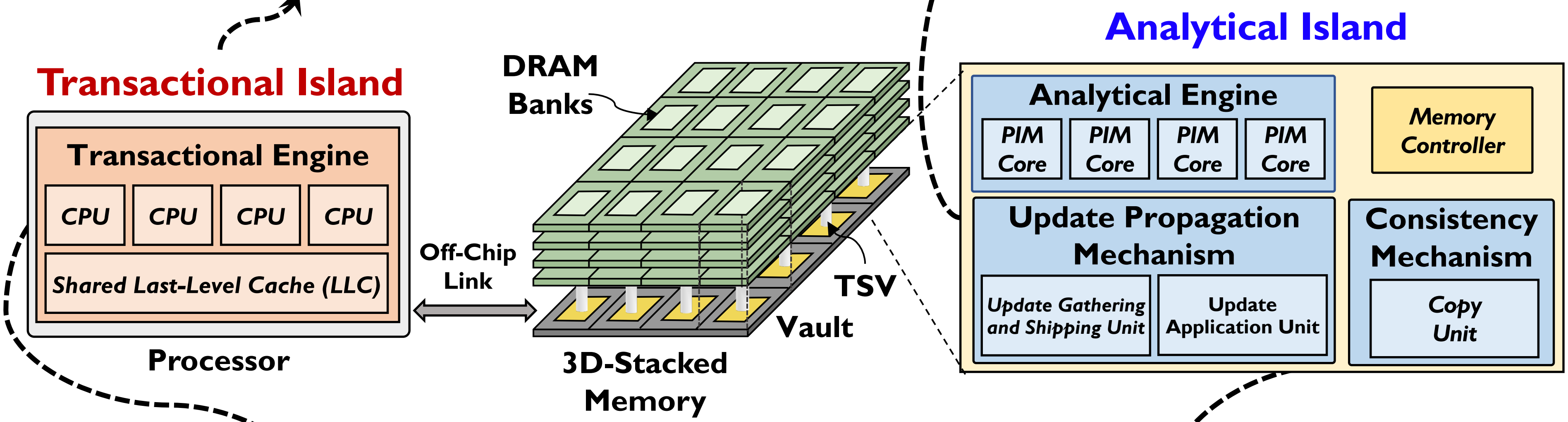
- **Leverage processing-in-memory (PIM) to reduce data movement**
- **PIM mitigates data movement overheads by placing computation units nearby or inside memory**

Polynesia: High-Level Overview

Each island includes (1) a replica of data, (2) an optimized execution engine, and (3) a set of hardware resources

Designed to provide high read throughput

Designed to sustain bursts of updates



Conventional multicore CPUs with multi-level caches

Take advantage of PIM to mitigate data movement bottleneck

Key Results

Polynesia achieves 91.6% the transactional throughput of an ideal system by employing custom PIM logic for data freshness/consistency, which significantly reduces resource contention and data movement

Polynesia improves analytical throughput by 63.8% over an optimized multiple-instance system, by eliminating data movement, and using custom logic for update propagation and consistency

Overall, Polynesia achieves all three properties of HTAP system and has a higher transactional/analytical throughput (1.7x/3.74x) over prior HTAP systems

Conclusion

- Context: Many applications need to perform real-time data analysis using an Hybrid Transactional/Analytical Processing (HTAP) system
  - An ideal HTAP system should have **three properties**:
    (1) **data freshness** and **consistency**, (2) **workload-specific optimization**,
    (3) **performance isolation**

- Problem: Prior works **cannot achieve all properties** of an ideal HTAP system

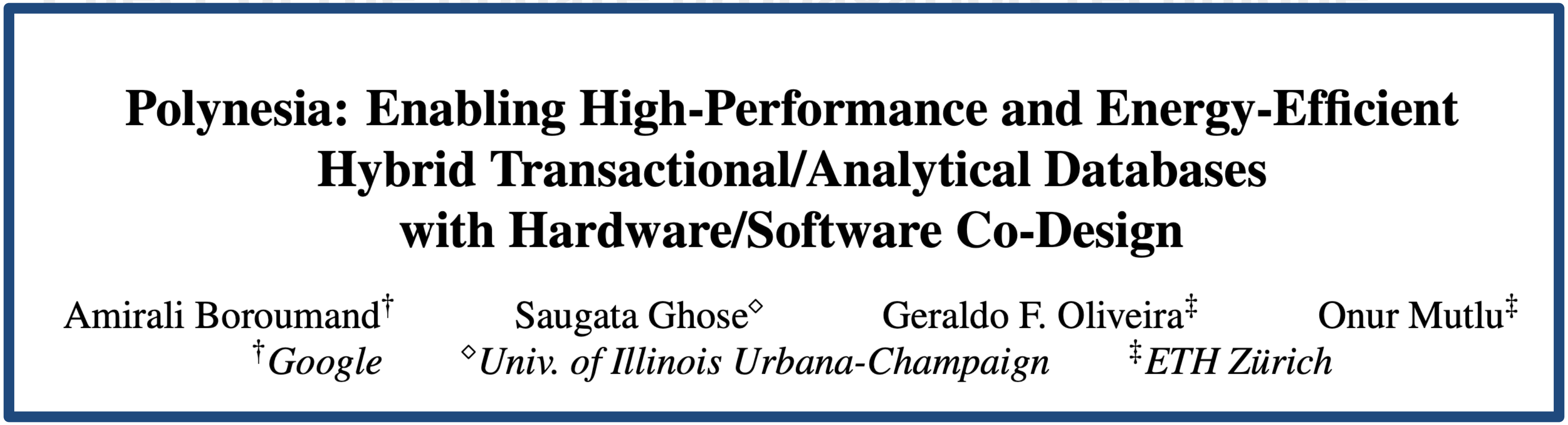
- Key Idea: Divide the system into transactional and analytical **processing islands**
  - Enables **workload-specific optimizations** and **performance isolation**

- Key Mechanism: Polynesia, a novel hardware/software cooperative design for in-memory HTAP databases
  - Implements **custom algorithms and hardware** to reduce the costs of **data freshness** and **consistency**
  - Exploits **PIM** for analytical processing to alleviate **data movement**

- Key Results: Polynesia outperforms three state-of-the-art HTAP systems
  - Average transactional/analytical throughput improvements of **1.7x/3.7x**
  - **48%** reduction on energy consumption

More in the Paper

- Real workload analysis
- Effect of the update propagation technique
- Effect of the dataset size
- Area Analysis



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

Amirali Boroumand[†] Saugata Ghose[◇] Geraldo F. Oliveira[‡] Onur Mutlu[‡]

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

Full Draft

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

Amirali Boroumand
Geraldo F. Oliveira
Saugata Ghose
Onur Mutlu

ICDE
2022

Full Draft

SAFARI Google UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN ETH Zürich

Year III Results (2022 Annual Review 3)

- Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning [ISCA 2022]

- Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction [MICRO 2022]

- GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping [MICRO 2022]

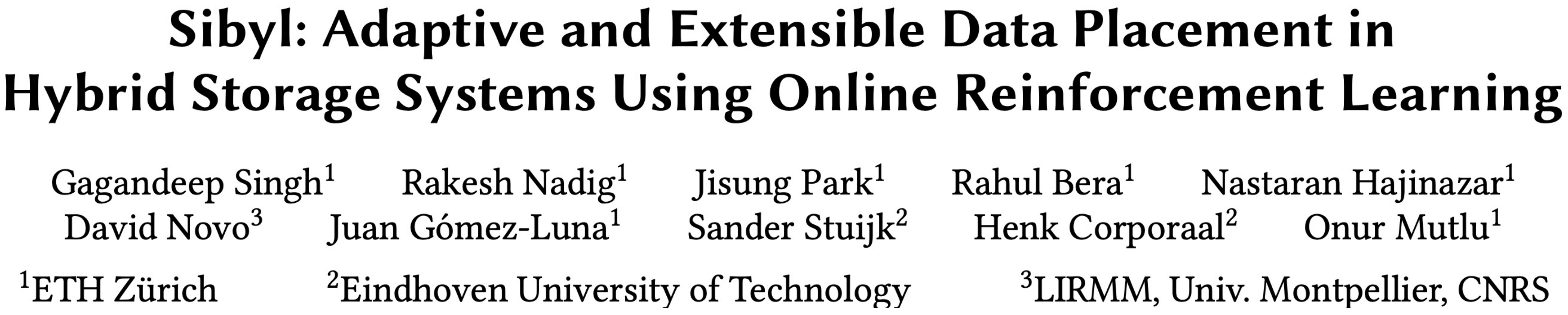
- pLUTo: Enabling Massively Parallel Computation via In DRAM via Lookup Tables [MICRO 2022]

- DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression [FAST 2022]

- A Modern Primer on Processing in Memory [Arxiv, Updated 2022]

Sibyl: Self-Optimizing Hybrid Storage Systems

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



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

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

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

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

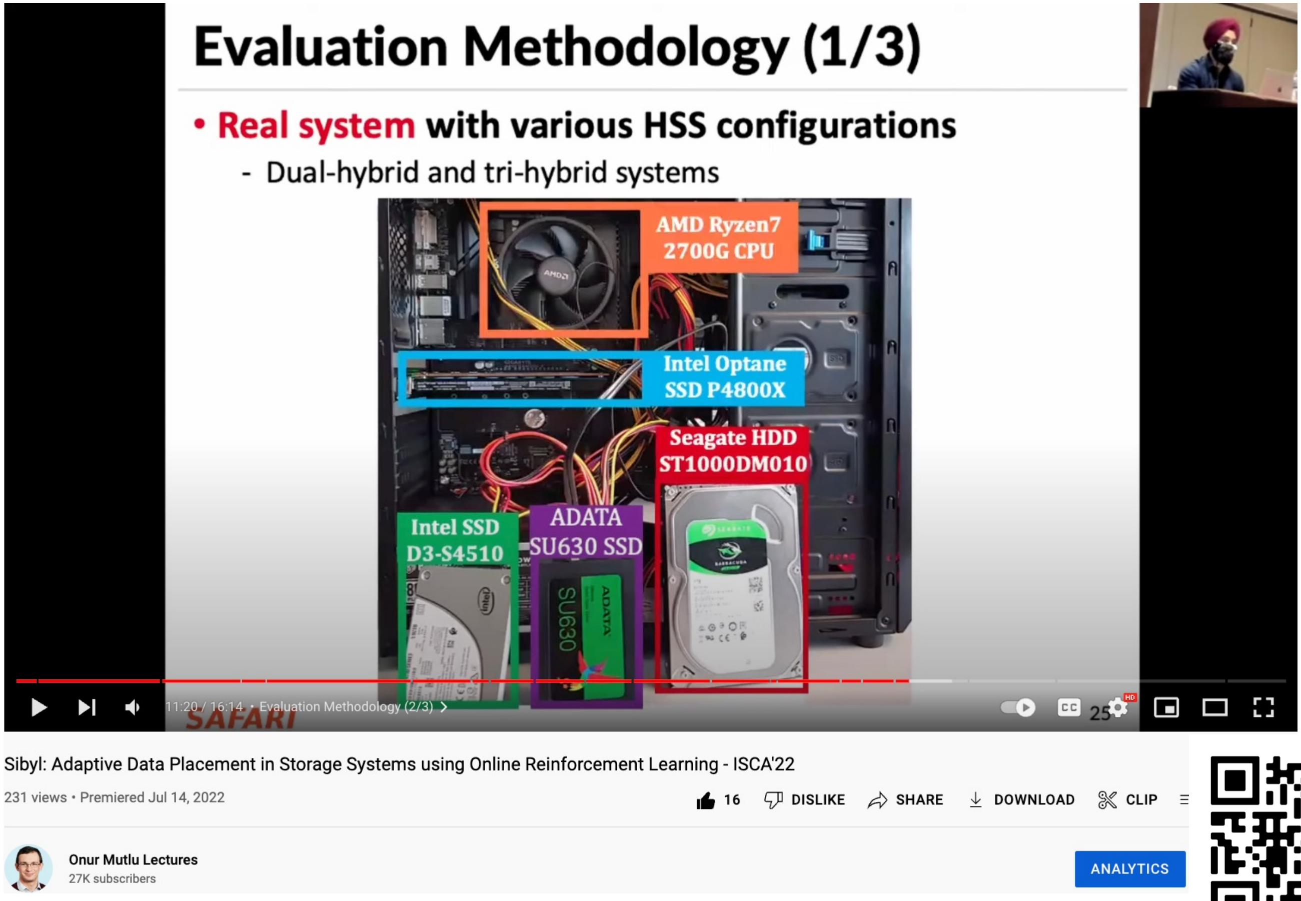
Sibyl:
Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning

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

SAFARI
SAFARI Research Group
safari.ethz.ch
ETH zürich
LIRMM
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Sibyl Talk Video [ISCA'22]

- **Gagandeep Singh**, Rakesh Nadig, Jisung Park, Rahul Bera, Nastaran Hajinazar, David Novo, Juan Gomez-Luna, Sander Stuijk, Henk Corporaal, and Onur Mutlu,
"Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning"
ISCA, New York, June 2022.
[Sibyl Source Code]



Evaluation Methodology (1/3)

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

Sibyl: Adaptive Data Placement in Storage Systems using Online Reinforcement Learning - ISCA'22

231 views • Premiered Jul 14, 2022

Onur Mutlu Lectures

27K subscribers

https://www.youtube.com/watch?v=5-WedkiB000

Executive Summary

- **Background**: A hybrid storage system (HSS) uses multiple different storage devices to provide high and scalable storage capacity at high performance

- **Problem**: Two key shortcomings of prior data placement policies:
  - Lack of **adaptivity to:**
    - **Workload changes**
    - **Changes in device types and configurations**
  - Lack of **extensibility** to more devices

- **Goal**: Design a data placement technique that provides:
  - ***Adaptivity***, by **continuously learning and adapting** to the **application and underlying device characteristics**
  - ***Easy extensibility*** to incorporate a wide range of hybrid storage configurations

- **Contribution:** Sibyl, the first reinforcement learning-based data placement technique in hybrid storage systems that:
  - Provides **adaptivity** to changing workload demands and underlying device characteristics
  - Can **easily extend** to any number of storage devices
  - Provides **ease of design and implementation** that requires only a small computation overhead

- **Key Results**: Evaluate on **real systems** using a wide range of workloads
  - Sibyl **improves performance by 21.6%** compared to the best previous data placement technique in dual-HSS configuration
  - In a tri-HSS configuration, Sibyl outperforms the state-of-the-art-policy policy by **48.2%**
  - Sibyl achieves **80% of the performance** of an oracle policy with storage overhead of only **124.4 KiB**

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

212

Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

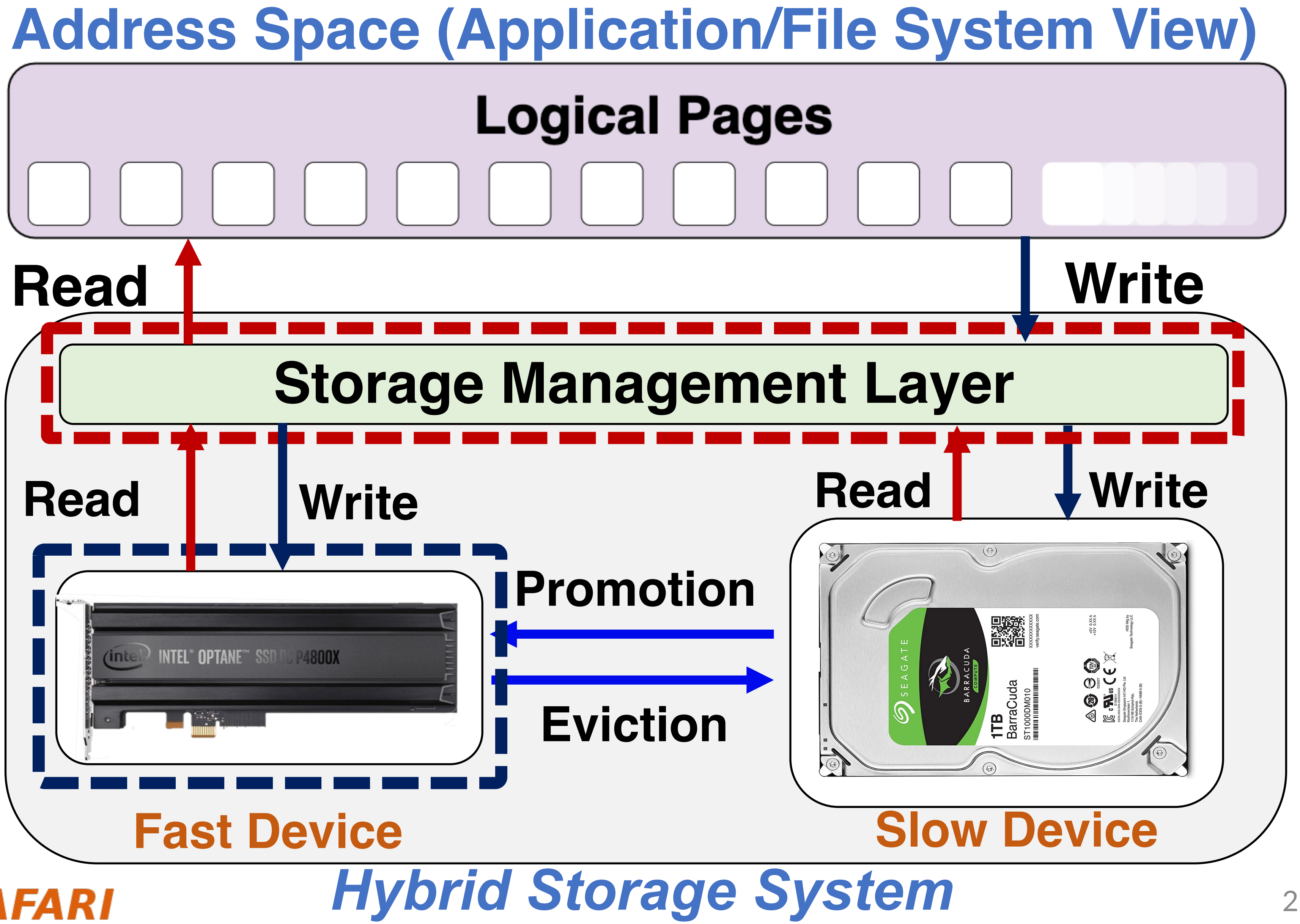
Sybil: Overview

Evaluation of Sybil and Key Results

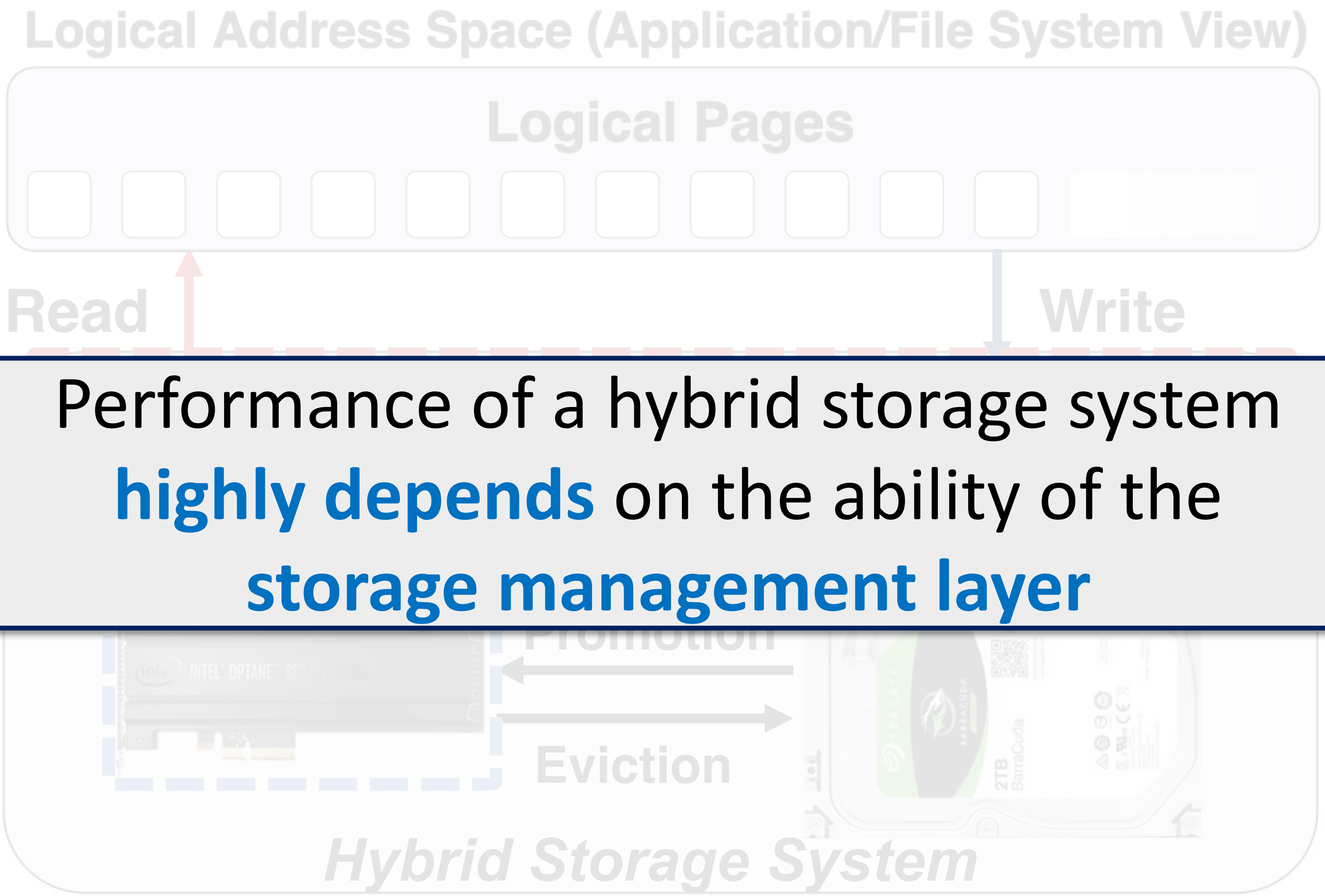
Conclusion

213

Hybrid Storage System Basics



Hybrid Storage System Basics



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

Key Shortcomings in Prior Techniques

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

1. Lack of **adaptivity to:**
    a) **Workload changes**
    b) **Changes in device types and configuration**

2. Lack of **extensibility** to more devices

216

Our Goal

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

1. **Adaptivity,** by **continuously learning** and **adapting** to the application and underlying device characteristics

2. **Easy extensibility** to incorporate a wide range of hybrid storage configurations

217

Our Proposal

Sibyl

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

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

218

Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

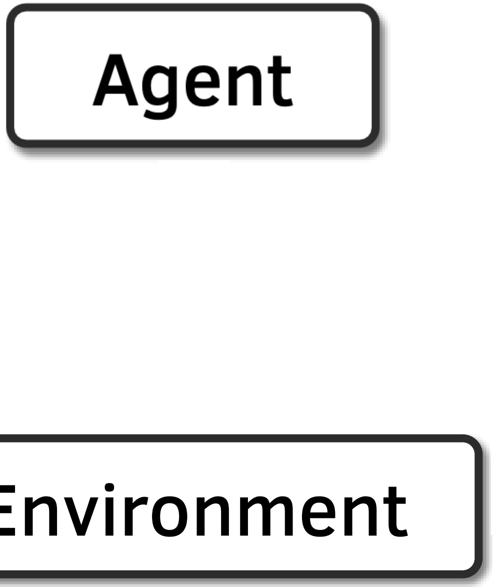
Sybil: Overview

Evaluation of Sybil and Key Results

Conclusion

219

Basics of Reinforcement Learning (RL)



Agent

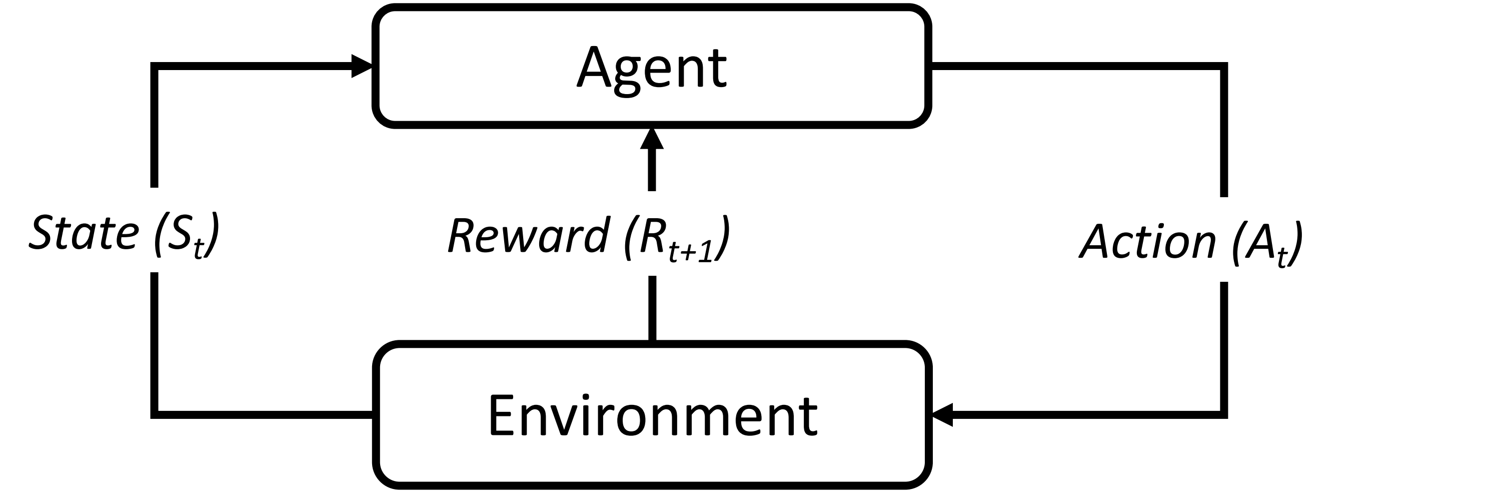
Environment

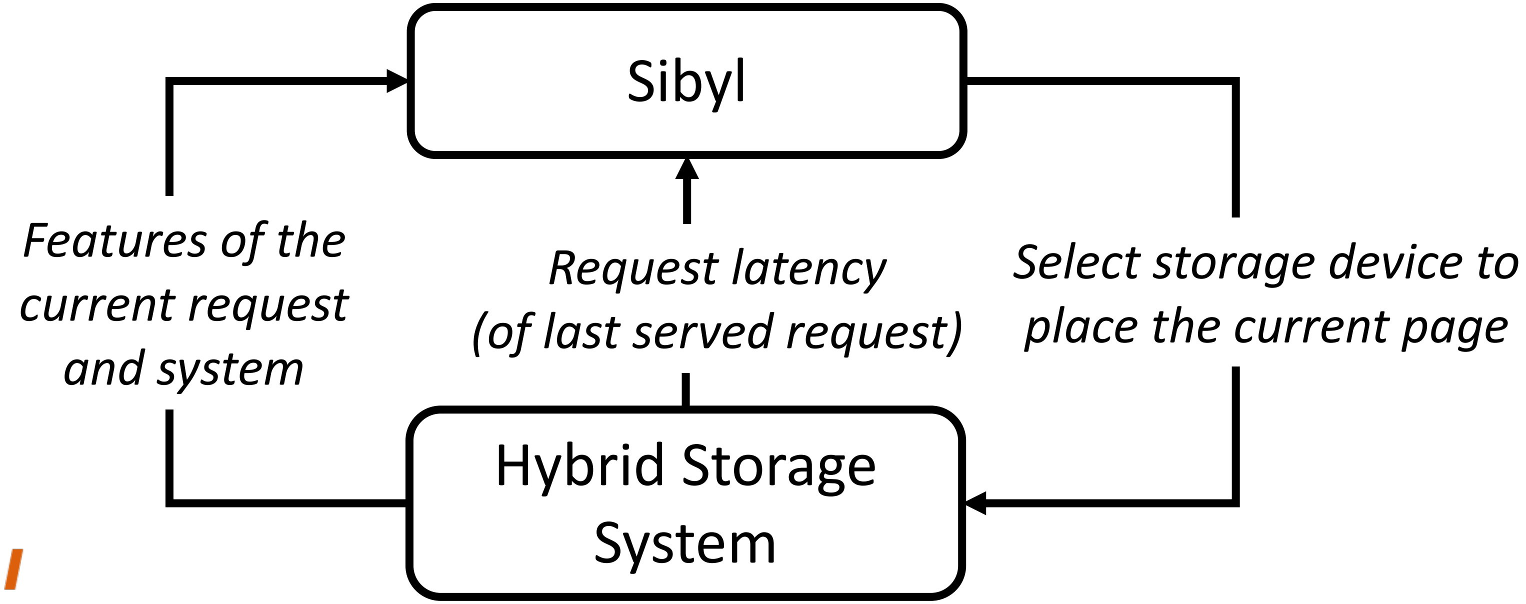
Agent learns to take an **action** in a given **state**

to maximize a numerical **reward**

220

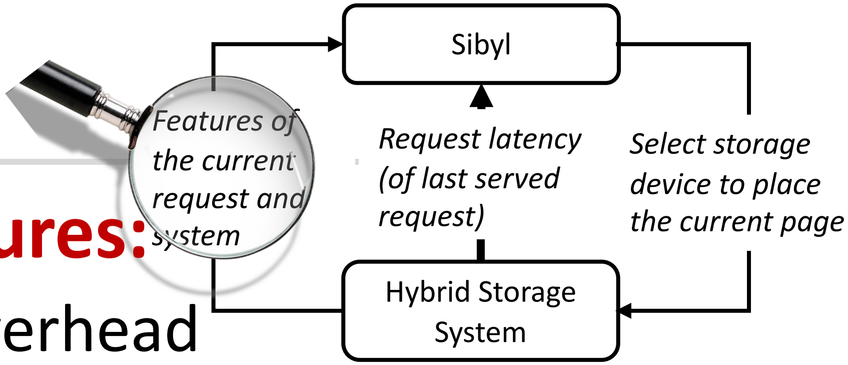
Formulating Data Placement as RL




Sibyl
Features of the current request and system
Request latency (of last served request)
Select storage device to place the current page
Hybrid Storage System

SAFARI

What is State?


Sibyl
Features of the current request and system
Request latency (of last served request)
Select storage device to place the current page
Hybrid Storage System

- **Limited number of state features:**
  - Reduce the implementation overhead
  - RL agent is more sensitive to reward

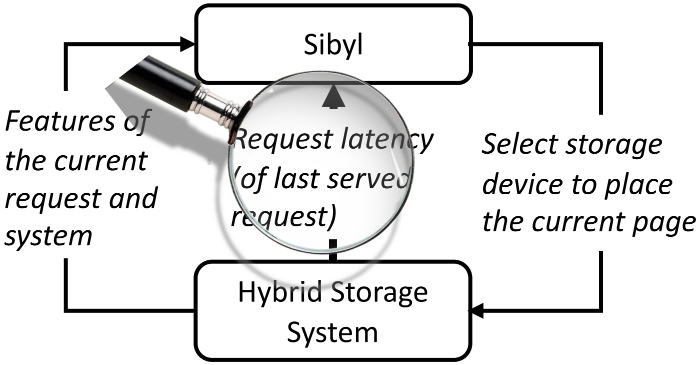
- ***6*-dimensional** vector of state features

$$O_t = (size_t, type_t, intr_t, cnt_t, cap_t, curr_t)$$

- We **quantize the state representation** into bins to reduce storage overhead

SAFARI

What is Reward?


Sibyl
Features of the current request and system
Request latency (of last served request)
Select storage device to place the current page
Hybrid Storage System

- Defines the **objective** of Sibyl

- We formulate the reward as a function of the **request latency**

- Encapsulates three key aspects:
  - **Internal state of the device** (e.g., read/write latencies, the latency of garbage collection, queuing delays, …)
  - **Throughput**
  - **Evictions**

- **More details in the paper**

SAFARI

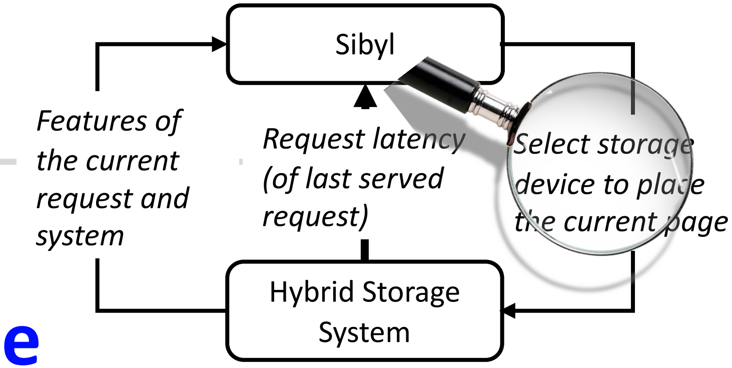
223

What is Action?

- At every new page request, the action is to **select a storage device**

- Action can be **easily extended** to any number of storage devices

- Sibyl learns to **proactively evict or promote** a page


Sibyl
Features of the current request and system
Request latency (of last served request)
Select storage device to place the current page
Hybrid Storage System

224

Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

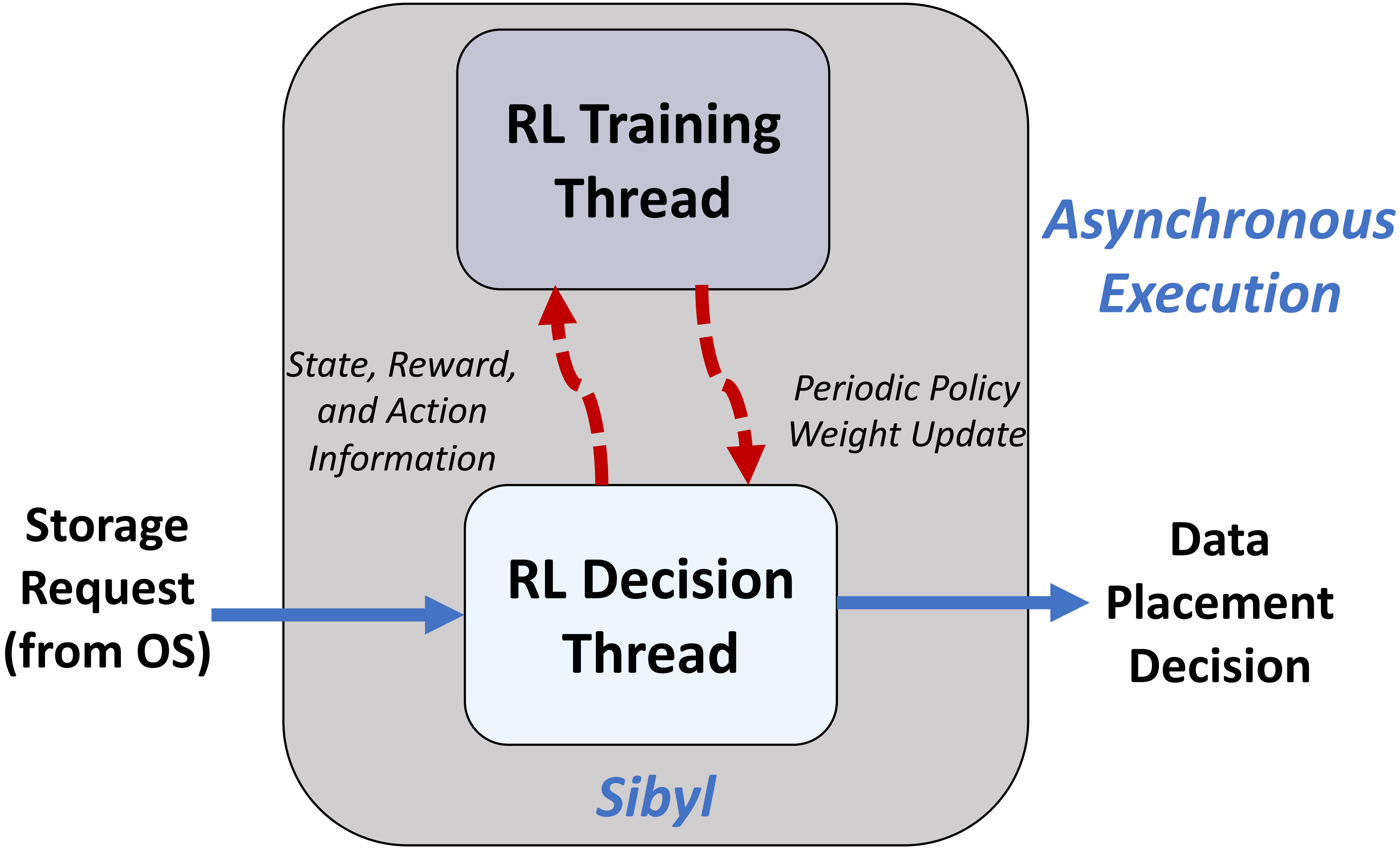
Sybil: Overview

Evaluation of Sybil and Key Results

Conclusion

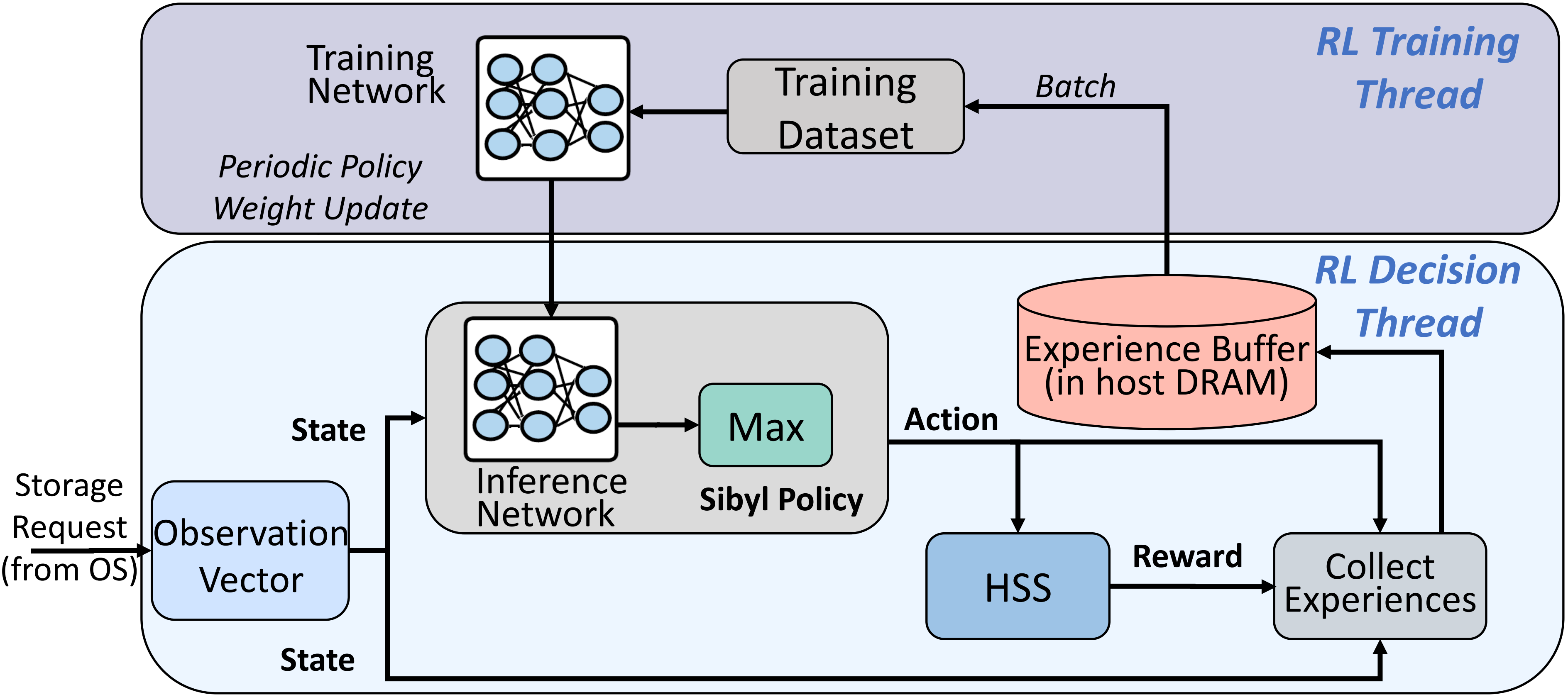
225

Sibyl Execution

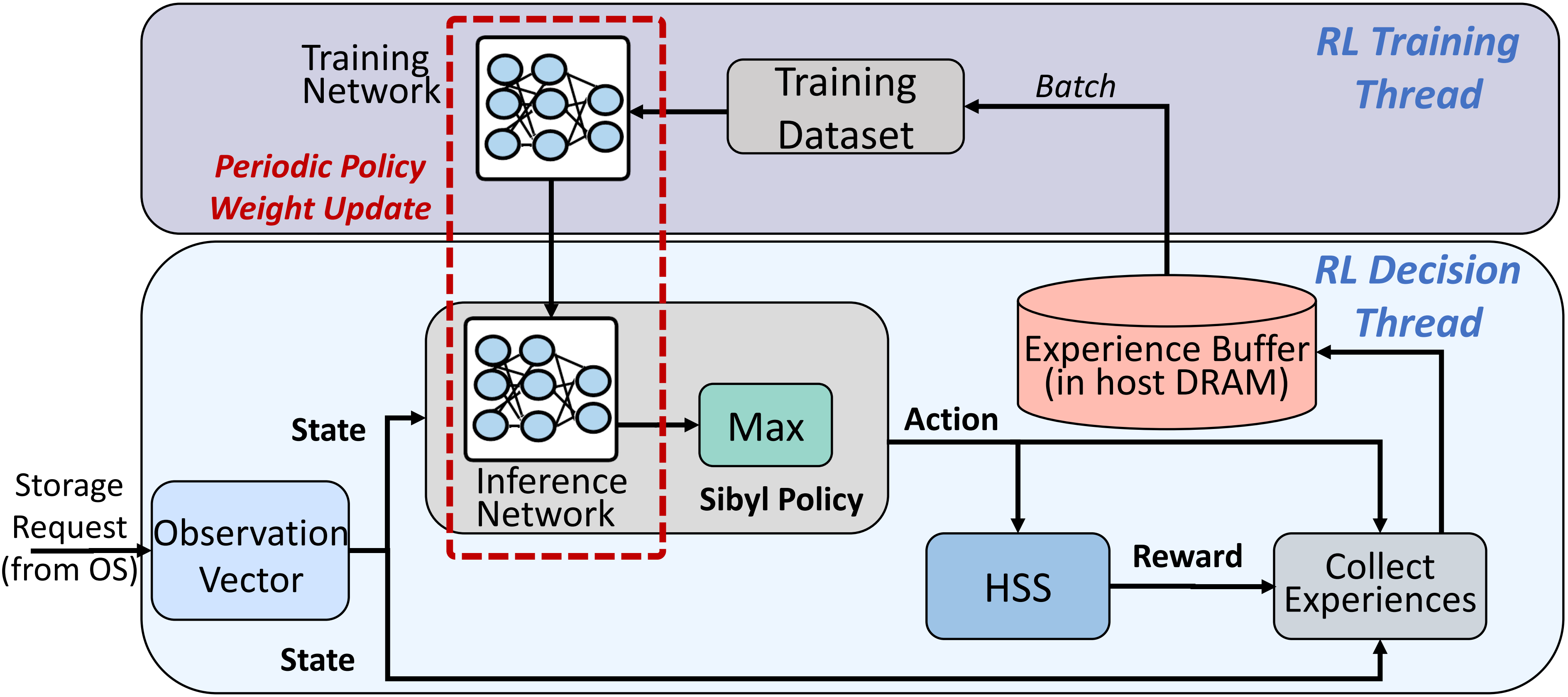

RL Training Thread
Asynchronous Execution
State, Reward, and Action Information
Periodic Policy Weight Update
Storage Request (from OS)
RL Decision Thread
Data Placement Decision
Sibyl

226

Sibyl Design: Overview



Sibyl Design: Overview



228

Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

Sybil: Overview

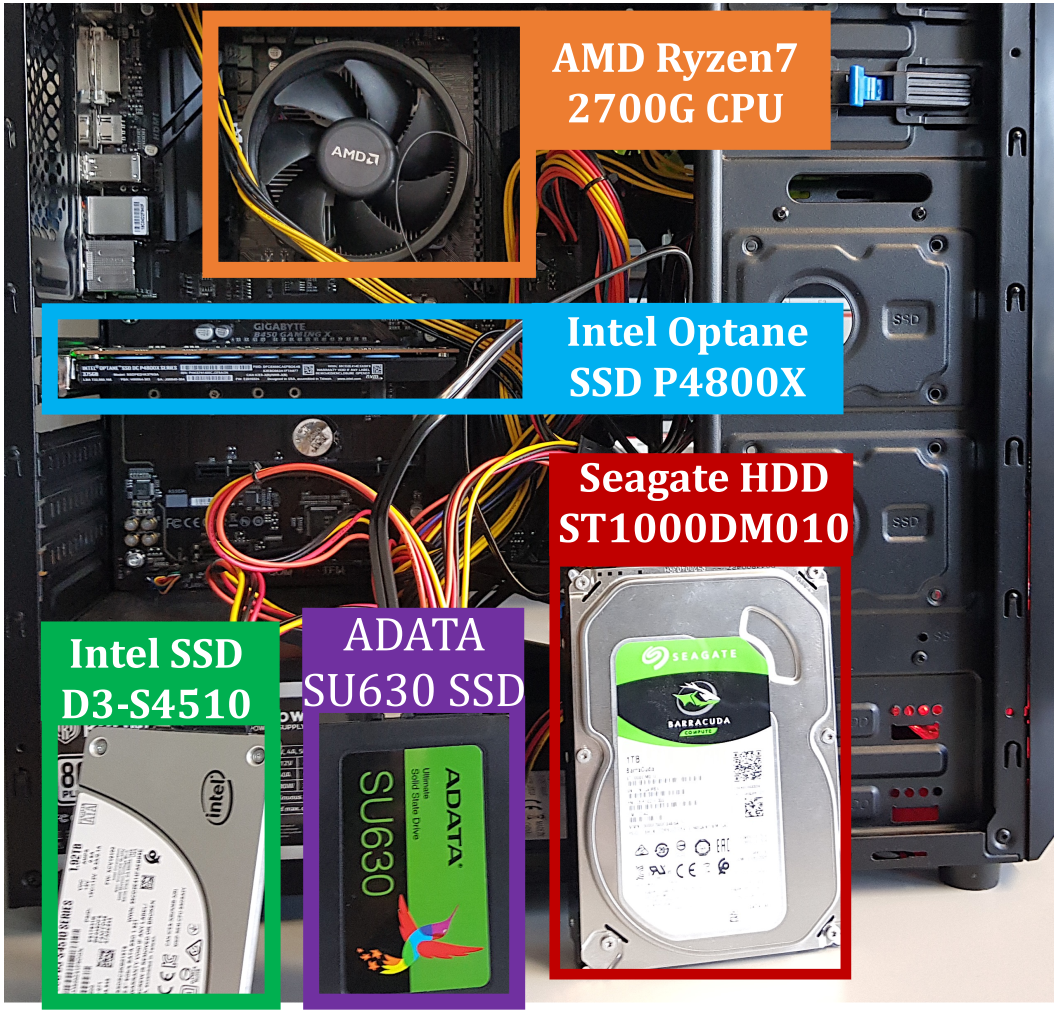
Evaluation of Sybil and Key Results

Conclusion

229

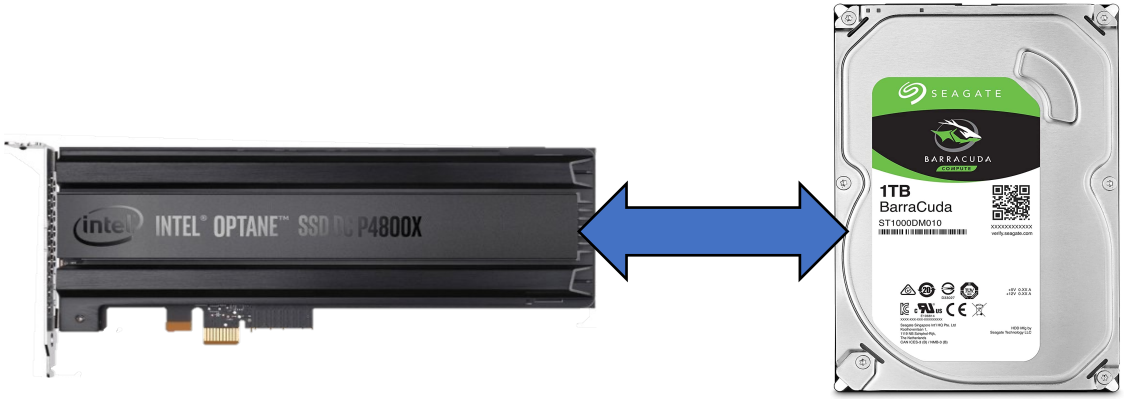
Evaluation Methodology (1/3)

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



Evaluation Methodology (2/3)

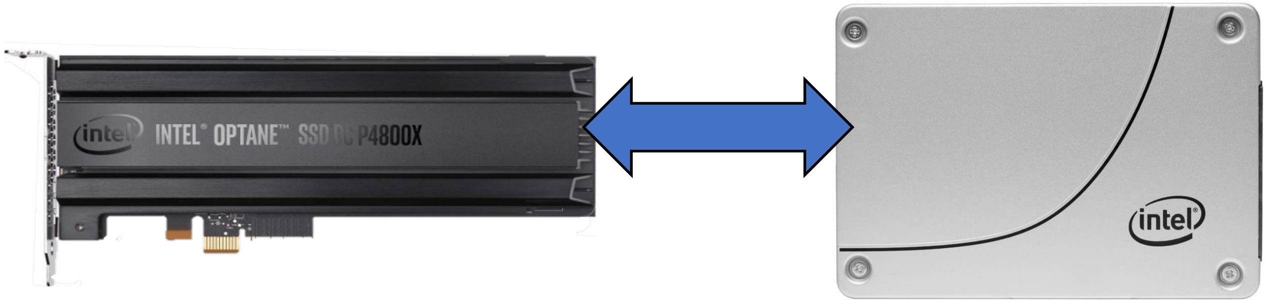
Cost-Oriented HSS Configuration



High-end SSD

Low-end HDD

Performance-Oriented HSS Configuration



High-end SSD

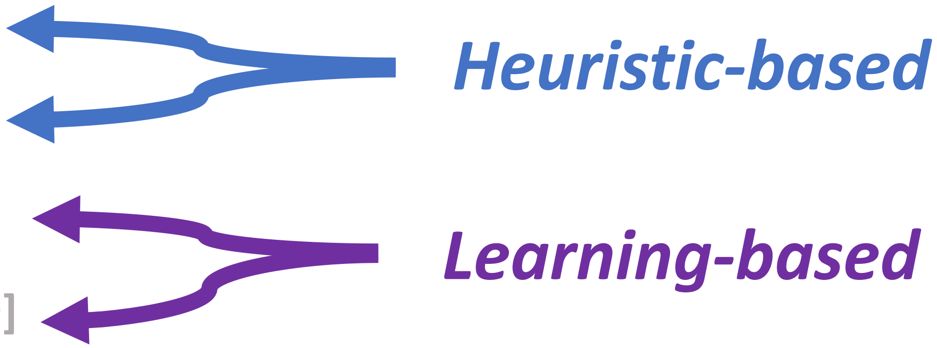
Middle-end SSD

231

Evaluation Methodology (3/3)

- **18 different workloads** from:
  - MSR Cambridge and Filebench Suites

- **Four** state-of-the-art data placement baselines:
  - CDE [Matsui+, Proc. IEEE'17]
  - HPS [Meswani+, HPCA'15]
  - Archivist [Ren+, ICCD'19]
  - RNN-HSS [Doudali+, HPDC'19]

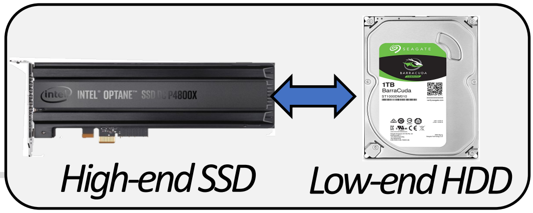


Heuristic-based

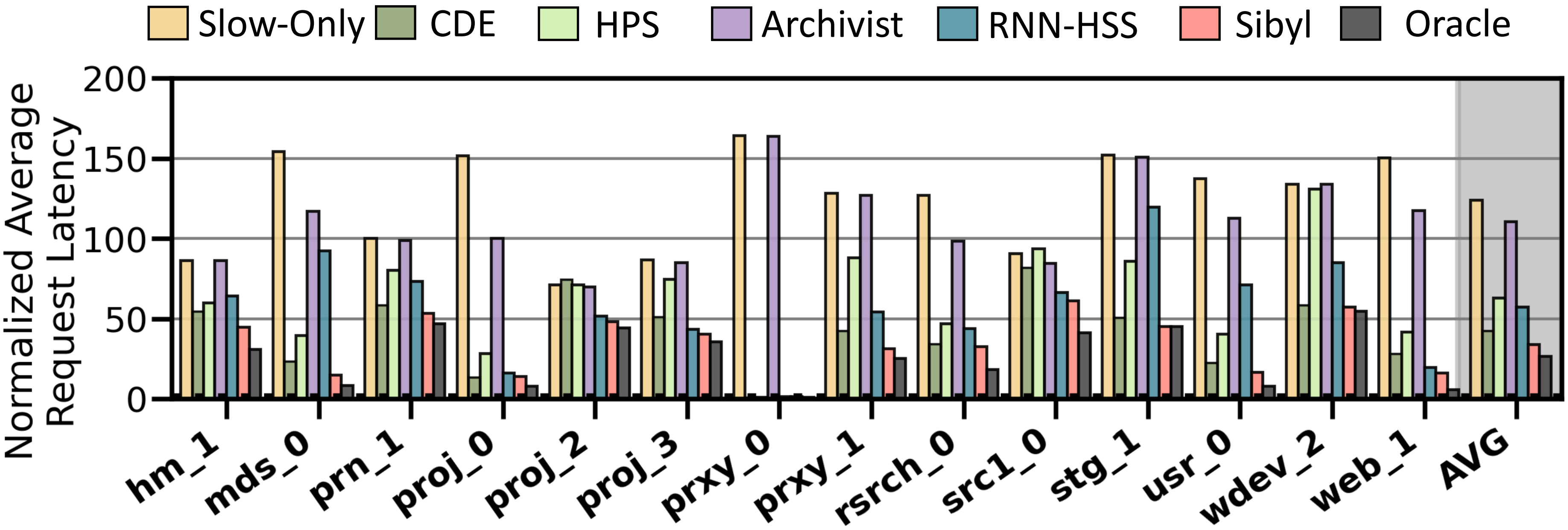
Learning-based

232

Performance Analysis

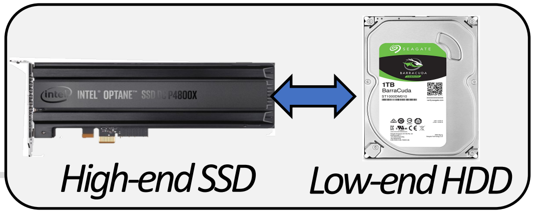


Cost-Oriented HSS Configuration

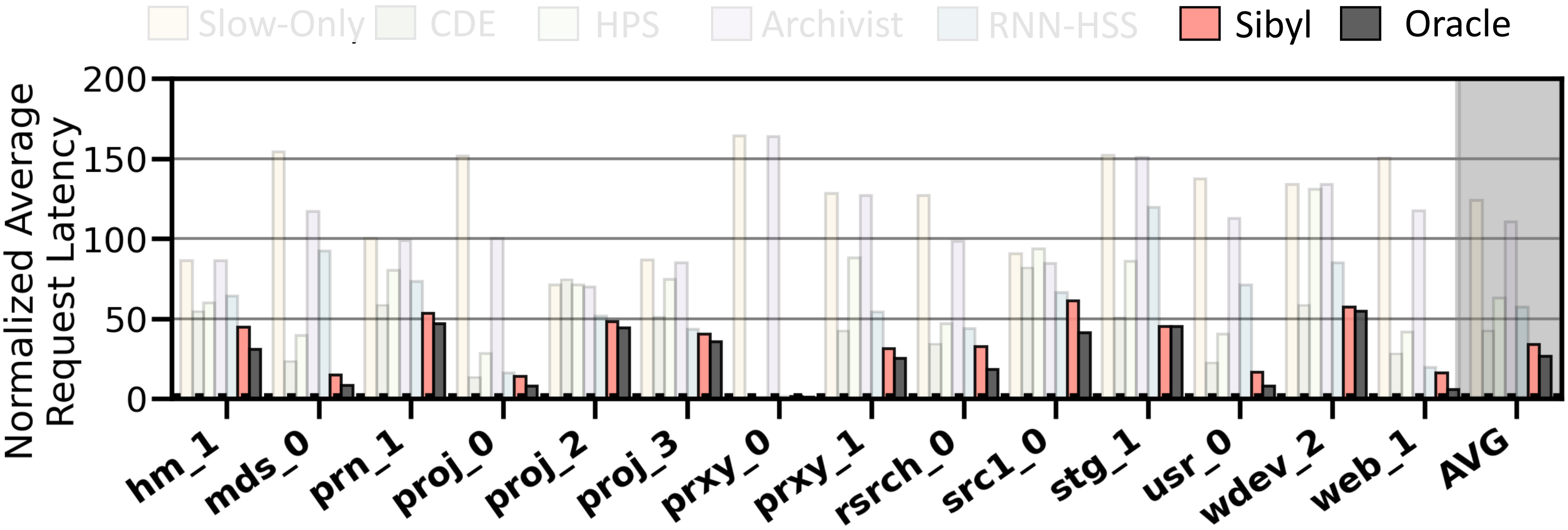


SAFARI

Performance Analysis

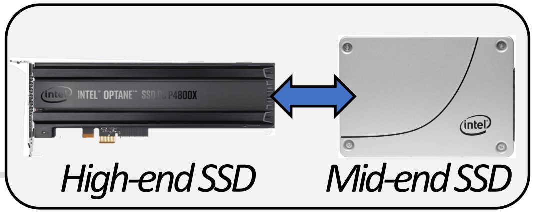


Cost-Oriented HSS Configuration

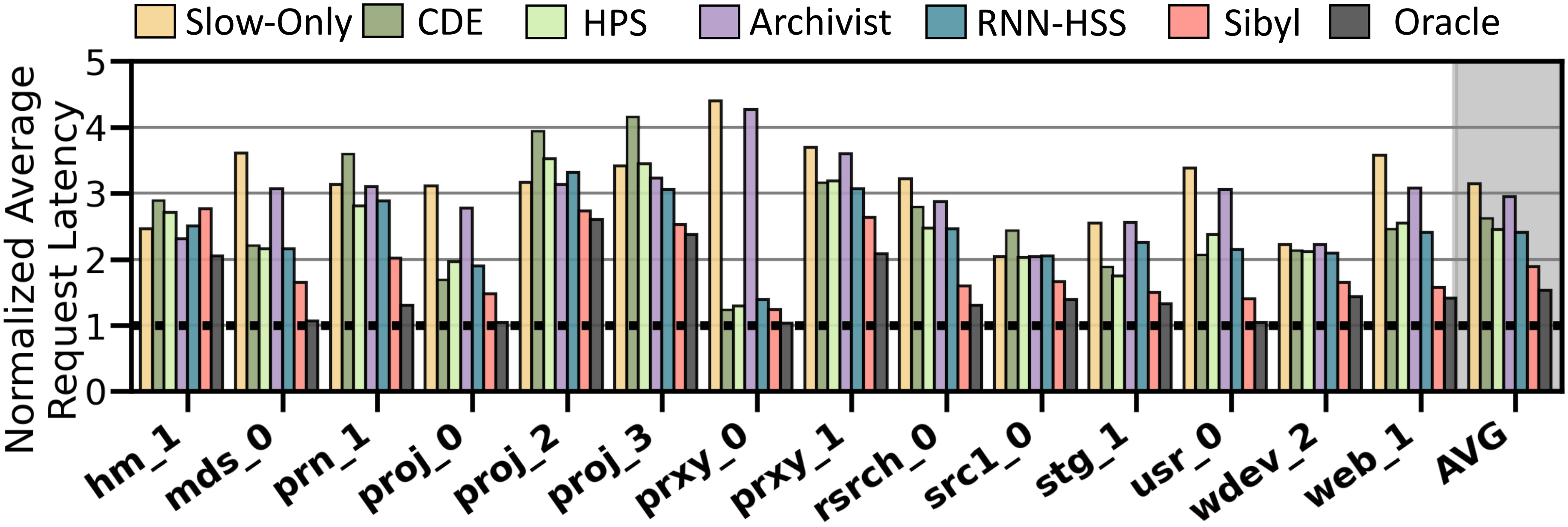


Sibyl consistently **outperforms all the baselines for all the workloads**

Performance Analysis



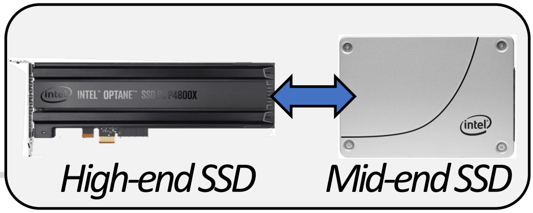
Performance-Oriented HSS Configuration



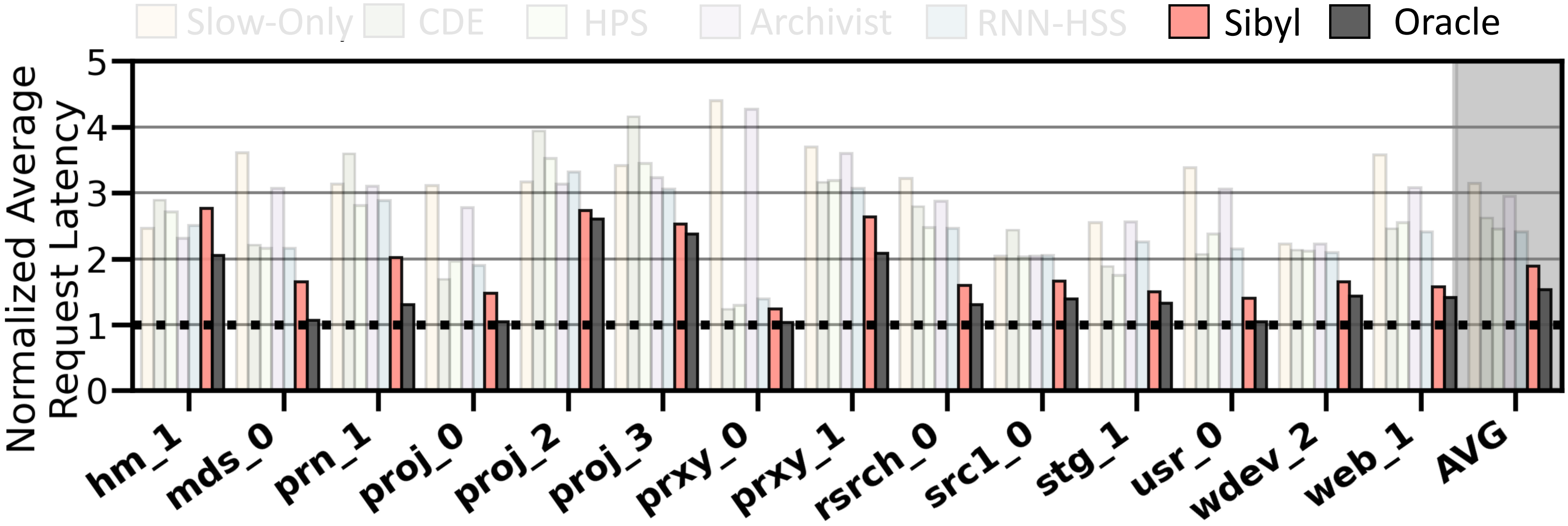
SAFARI

235

Performance Analysis



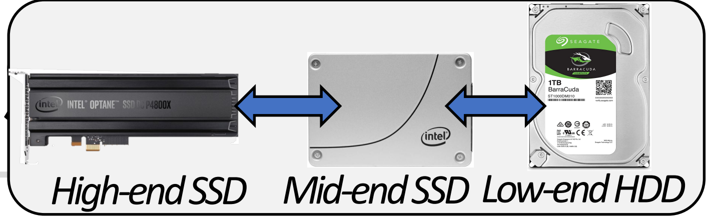
Performance-Oriented HSS Configuration



Sibyl provides **21.6% performance improvement** by **dynamically adapting its data placement policy**

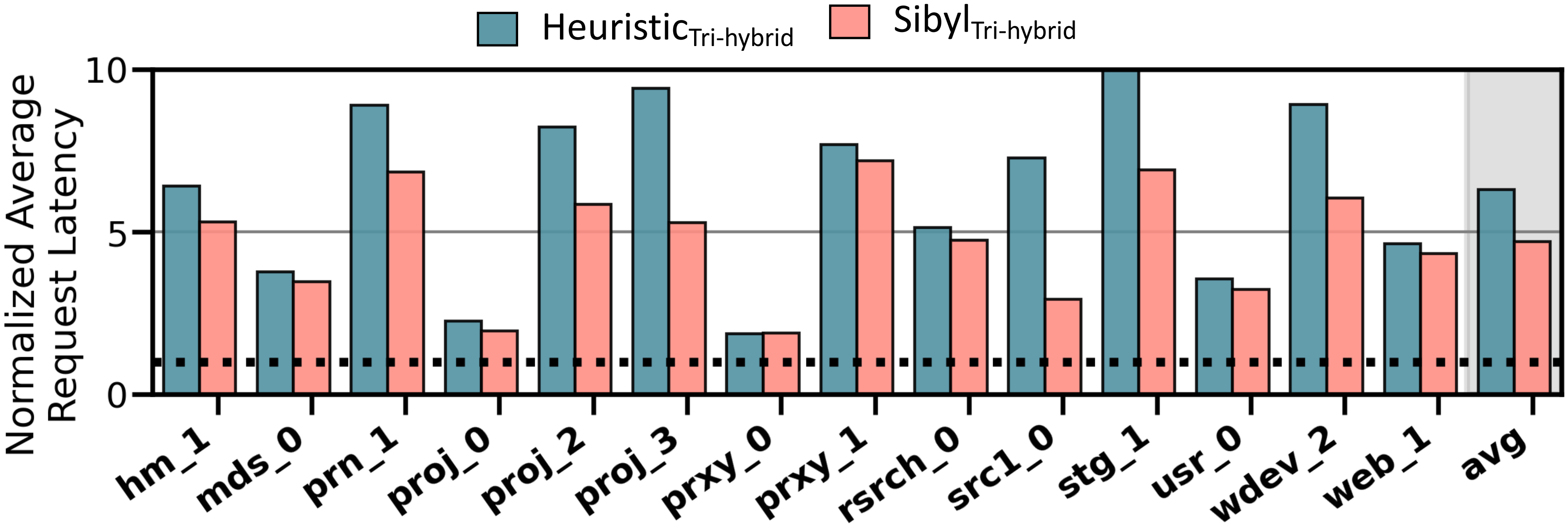
236

Performance on Tri-HS

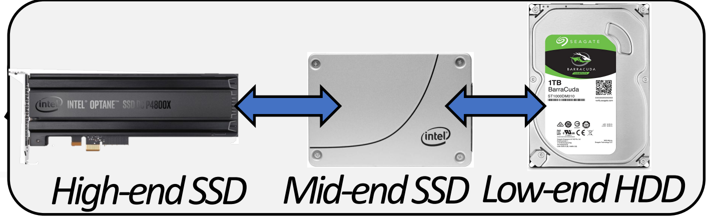


Extending Sibyl for **more devices**:

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



Performance on Tri-HS



Extending Sibyl for **more devices**:

1. **Add a new action**

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

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

Sibyl's Overhead

- **124.4 KiB** of total storage cost
  - Experience buffer, inference and training network
- **40-bit** metadata overhead per page for state features
- Inference latency of **~10ns**
- Training latency of **~2us**

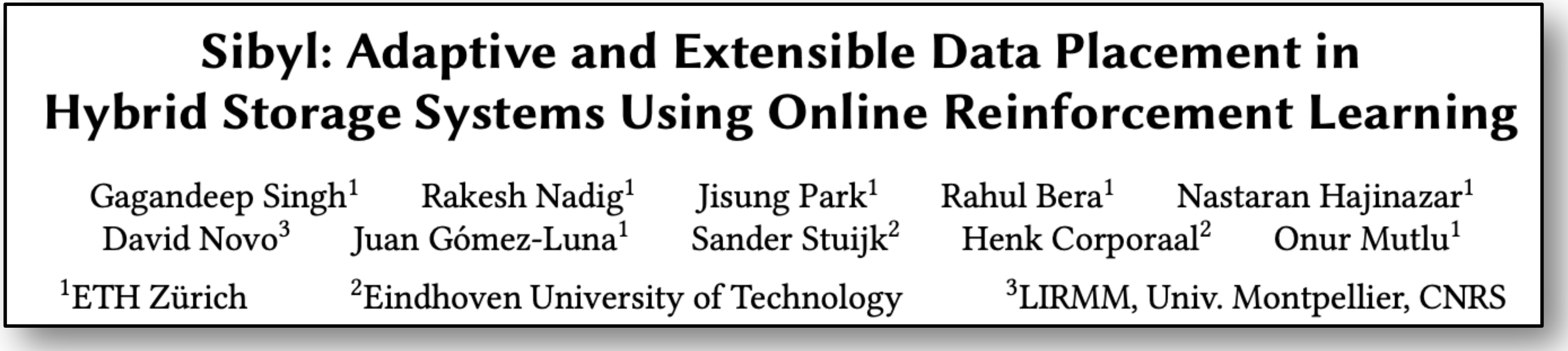
✓ **Small area overhead**

✓ **Small inference overhead**

✓ **Satisfies prediction latency**

239

More in the Paper



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

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

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

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

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

https://www.youtube.com/watch?v=5-WedkiB000

240

Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

Sybil: Overview

Evaluation of Sybil and Key Results

Conclusion

241

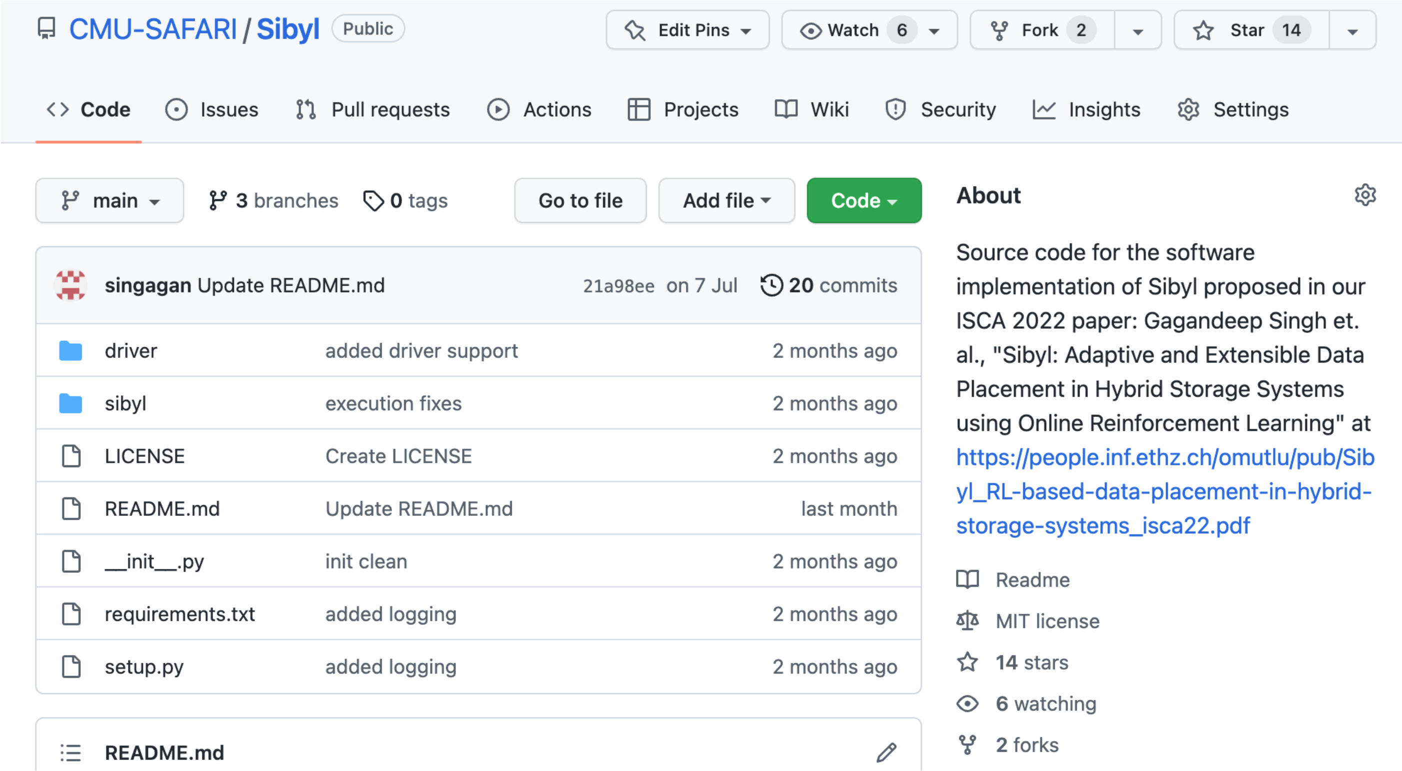
Conclusion

- **We introduced Sibyl,** the first reinforcement learning-based data placement technique in hybrid storage systems that provides
  - **Adaptivity**
  - **Easily extensibility**
  - **Ease of design and implementation**
- **We evaluated Sibyl** on **real systems** using many different workloads
  - Sibyl **improves performance by 21.6%** compared to the best prior data placement policy in a dual-HSS configuration
  - In a tri-HSS configuration, Sibyl **outperforms** the state-of-the-art-data placement policy by **48.2%**
  - Sibyl achieves **80% of the performance** of an oracle policy with a storage overhead of only **124.4 KiB**

242

Sibyl is Open-Source

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



singagan Update README.md 21a98ee on 7 Jul 20 commits

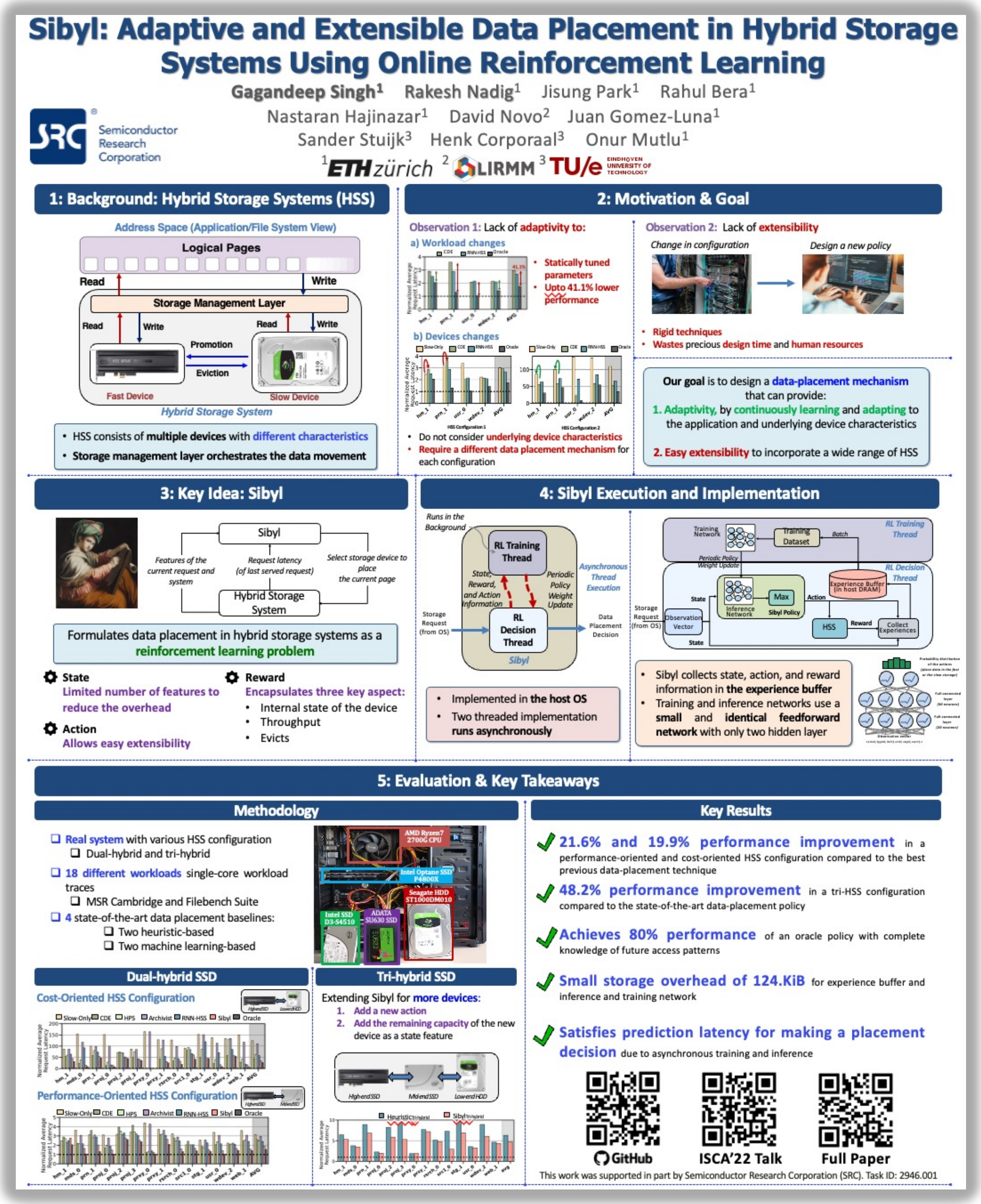
driver	added driver support	2 months ago
sibyl	execution fixes	2 months ago
LICENSE	Create LICENSE	2 months ago
README.md	Update README.md	last month
__init__.py	init clean	2 months ago
requirements.txt	added logging	2 months ago
setup.py	added logging	2 months ago

README.md

About

Source code for the software implementation of Sibyl proposed in our ISCA 2022 paper: Gagandeep Singh et. al., "Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems using Online Reinforcement Learning" at https://people.inf.ethz.ch/omutlu/pub/Sibyl_RL-based-data-placement-in-hybrid-storage-systems_isca22.pdf

- Readme
- MIT license
- 14 stars
- 6 watching
- 2 forks



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

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

[1] ETH zürich [2] LIRMM [3] TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Semiconductor Research Corporation

1: Background: Hybrid Storage Systems (HSS)

- HSS consists of **multiple devices** with different characteristics
- **Storage management layer orchestrates the data movement**

2: Motivation & Goal

Observation 1: Lack of adaptivity to:

a) Workload changes

- Statically tuned parameters
- Upto 41.1% lower performance

b) Devices changes

- Do not consider underlying device characteristics
- Require a different data placement mechanism for each configuration

Observation 2: Lack of extensibility

Change in configuration → *Design a new policy*

- Rigid techniques
- Wastes precious design time and human resources

Our goal is to design a data-placement mechanism that can provide:

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

3: Key Idea: Sibyl

Sibyl
Features of the current request and system
Request latency (of last served request)
Select storage device to place the current page
Hybrid Storage System

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

State
Limited number of features to reduce the overhead

Action
Allows easy extensibility

Reward
Encapsulates three key aspect:

- Internal state of the device
- Throughput
- Evicts

4: Sibyl Execution and Implementation

- Implemented in **the host OS**
- Two threaded implementation **runs asynchronously**

- Sibyl collects state, action, and reward information in **the experience buffer**
- Training and inference networks use a **small** and **identical feedforward network** with only two hidden layer

5: Evaluation & Key Takeaways

Methodology

- ❑ Real system with various HSS configuration
  - ❑ Dual-hybrid and tri-hybrid
- ❑ 18 different workloads single-core workload traces
  - ❑ MSR Cambridge and Filebench Suite
- ❑ 4 state-of-the-art data placement baselines:
  - ❑ Two heuristic-based
  - ❑ Two machine learning-based

Dual-hybrid SSD

Cost-Oriented HSS Configuration

Performance-Oriented HSS Configuration

Tri-hybrid SSD

Extending Sibyl for more devices:

1. Add a new action
2. Add the remaining capacity of the new device as a state feature

Key Results

- ✓ 21.6% and 19.9% performance improvement in a performance-oriented and cost-oriented HSS configuration compared to the best previous data-placement technique
- ✓ 48.2% performance improvement in a tri-HSS configuration compared to the state-of-the-art data-placement policy
- ✓ Achieves 80% performance of an oracle policy with complete knowledge of future access patterns
- ✓ Small storage overhead of 124.KiB for experience buffer and inference and training network
- ✓ Satisfies prediction latency for making a placement decision due to asynchronous training and inference

GitHub ISCA'22 Talk Full Paper

This work was supported in part by Semiconductor Research Corporation (SRC). Task ID: 2946.001

Please Check Out Our Poster!

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

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

SAFARI
SAFARI Research Group
safari.ethz.ch

ETH zürich

LIRMM

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Year III Results (2022 Annual Review 3)

- Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning [ISCA 2022]

- Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction [MICRO 2022]

- GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping [MICRO 2022]

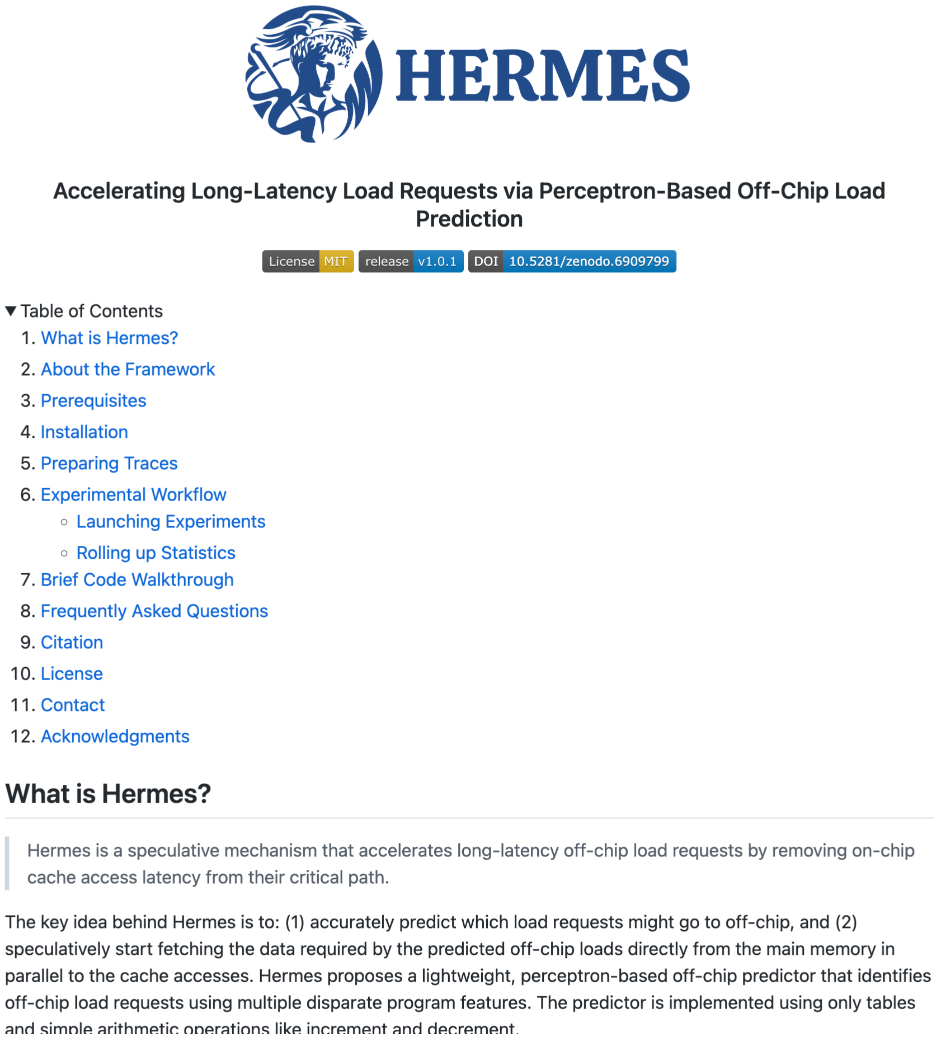
- pLUTo: Enabling Massively Parallel Computation via In DRAM via Lookup Tables [MICRO 2022]

- DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression [FAST 2022]

- A Modern Primer on Processing in Memory [Arxiv, Updated 2022]

Hermes

- To Appear in MICRO 2022



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

License MIT release v1.0.1 DOI 10.5281/zenodo.6909799

▼ Table of Contents

What is Hermes?

> Hermes is a speculative mechanism that accelerates long-latency off-chip load requests by removing on-chip cache access latency from their critical path.

The key idea behind Hermes is to: (1) accurately predict which load requests might go to off-chip, and (2) speculatively start fetching the data required by the predicted off-chip loads directly from the main memory in parallel to the cache accesses. Hermes proposes a lightweight, perceptron-based off-chip predictor that identifies off-chip load requests using multiple disparate program features. The predictor is implemented using only tables and simple arithmetic operations like increment and decrement.

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

HERMES

Accelerating Long-Latency Load Requests via Perceptron-based Off-chip Load Prediction

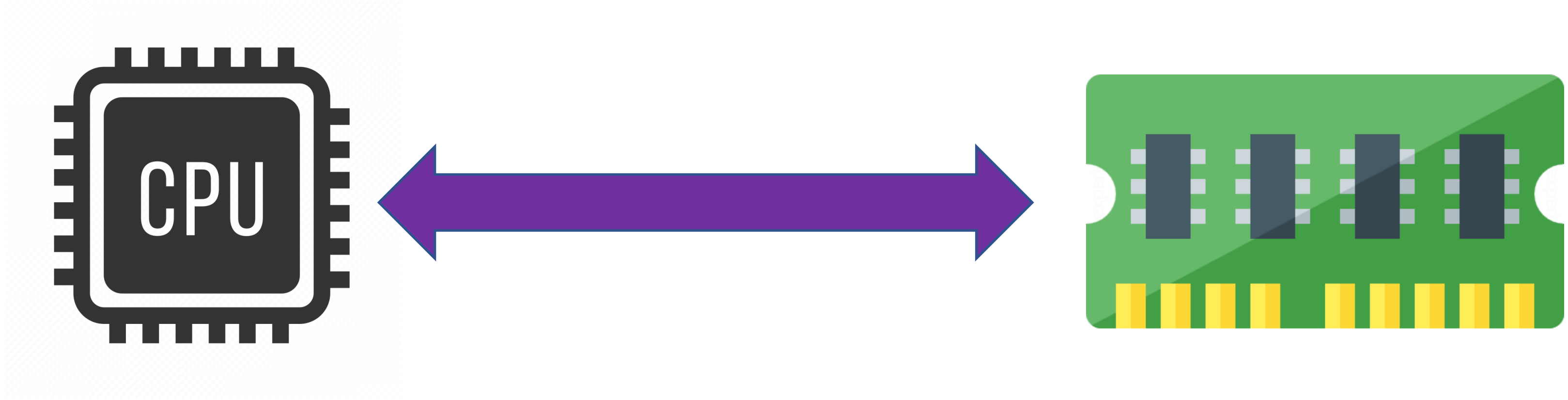
Rahul Bera, Konstantinos Kanellopoulos, Shankar Balachandran, David Novo, Ataberk Olgun, Mohammad Sadrosadati, Onur Mutlu

SAFARI
SAFARI Research Group
safari.ethz.ch

ETH zürich

intel

LIRMM



Long-latency off-chip requests

significantly limit performance of a processor

1 Deploy sophisticated **prefetchers**

2 **Increase size** of on-chip caches

Nearly **50%** of the off-chip requests

in a no-prefetching system

still go to the main memory

even in presence of state-of-the-art prefetcher

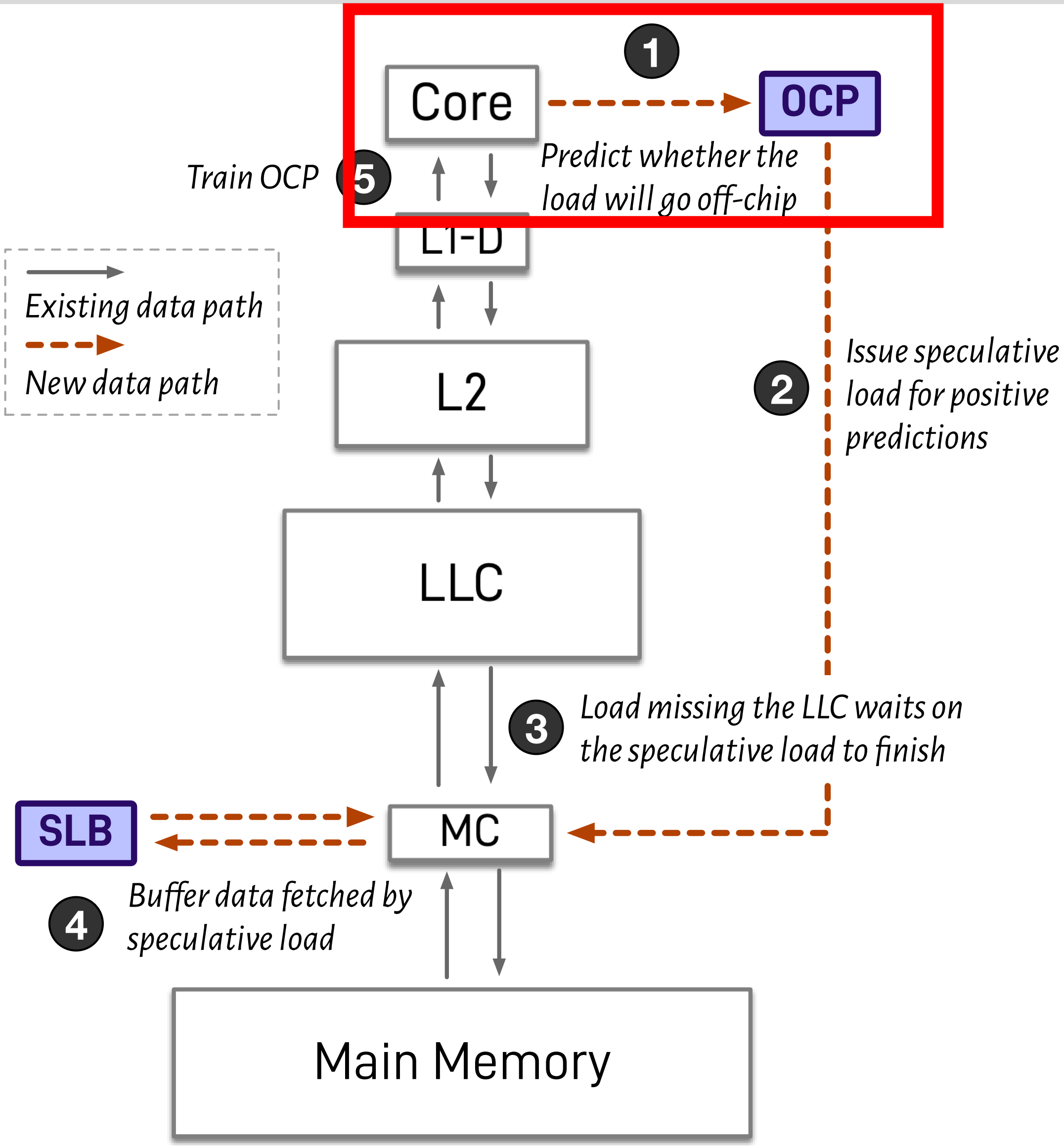
37.5% of the stall cycles caused by an off-chip load can be reduced by **removing on-chip cache access latency from its critical path**

HERMES

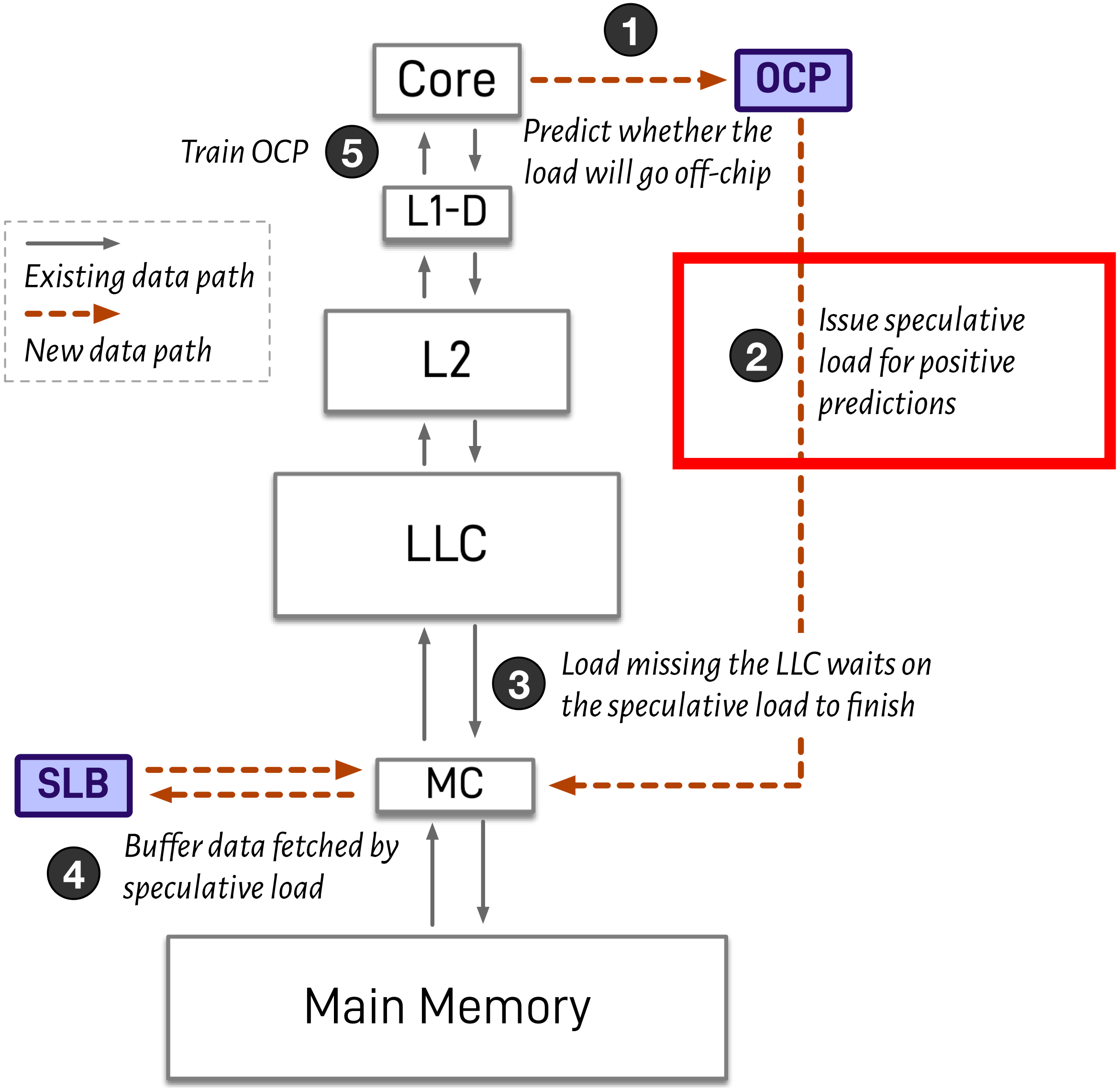
Predicts which load requests might go off-chip using multiple program features

Starts fetching data **directly** from main memory while concurrently accessing the cache hierarchy

Hermes: Overview

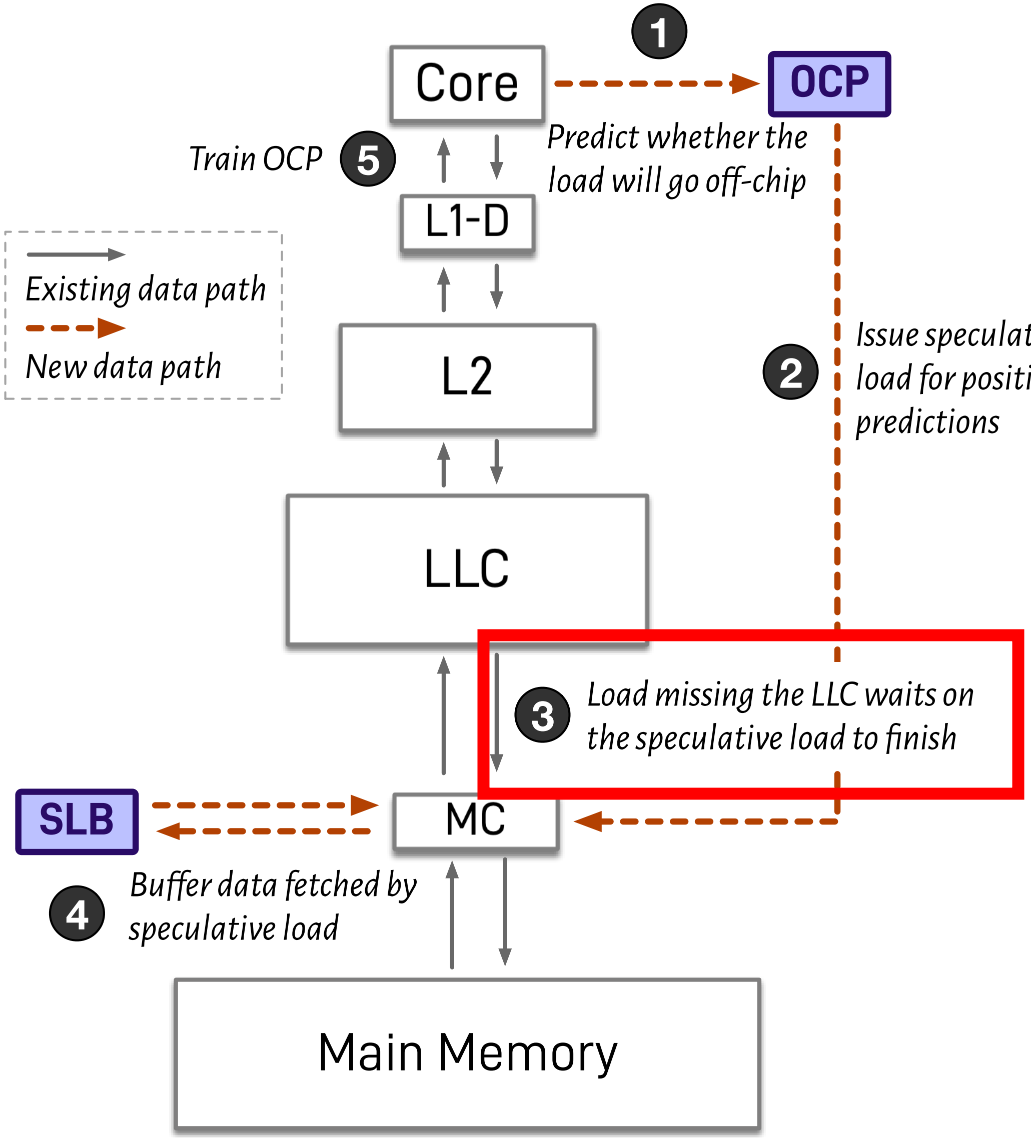

1
Core
OCP
Predict whether the load will go off-chip
Train OCP
5
L1-D
Existing data path
New data path
L2
2
Issue speculative load for positive predictions
LLC
3
Load missing the LLC waits on the speculative load to finish
SLB
MC
4
Buffer data fetched by speculative load
Main Memory

Hermes: Overview


1
Core
OCP
Predict whether the load will go off-chip
Train OCP
5
L1-D
Existing data path
New data path
L2
2
Issue speculative load for positive predictions
LLC
3
Load missing the LLC waits on the speculative load to finish
SLB
MC
4
Buffer data fetched by speculative load
Main Memory

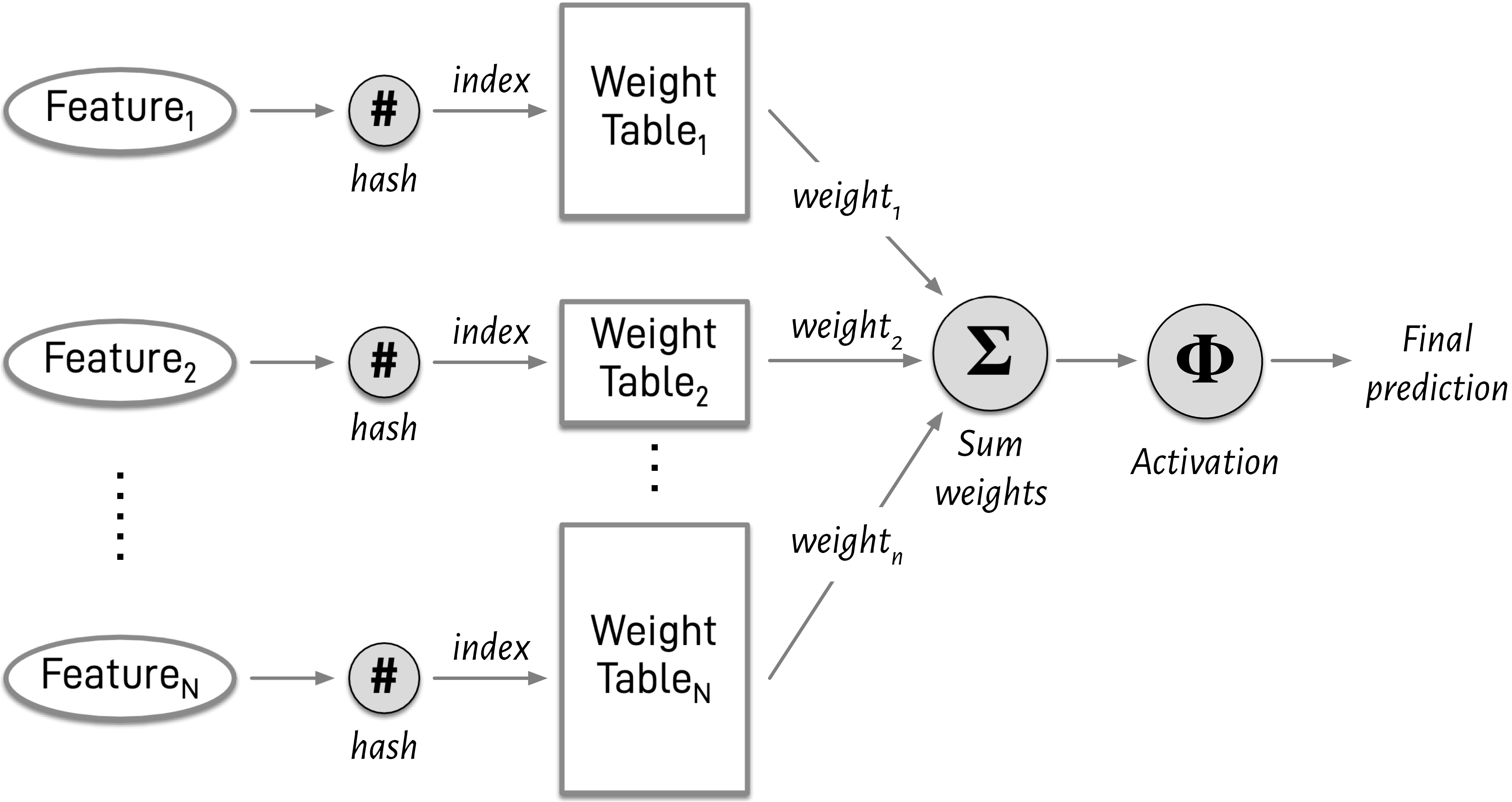
SAFARI

Hermes: Overview


1
Core
OCP
Predict whether the load will go off-chip
Train OCP
5
L1-D
Existing data path
New data path
L2
2
Issue speculative load for positive predictions
LLC
3
Load missing the LLC waits on the speculative load to finish
SLB
MC
4
Buffer data fetched by speculative load
Main Memory

SAFARI

Perceptron-based Off-chip Predictor



1

We evaluate Hermes using a wide-range of workloads

Hermes improves performance by

5.4% in single-core
5.1% in eight-core
6.2% in memory bandwidth-constrained core

over the baseline with the state-of-the-art prefetcher

2

Consistent performance improvement in a wide range of configurations with varying prefetchers and cache access latency

5.1%, 6.2%, 7.7% performance improvement
in single-core with SPP, Bingo, SMS prefetchers

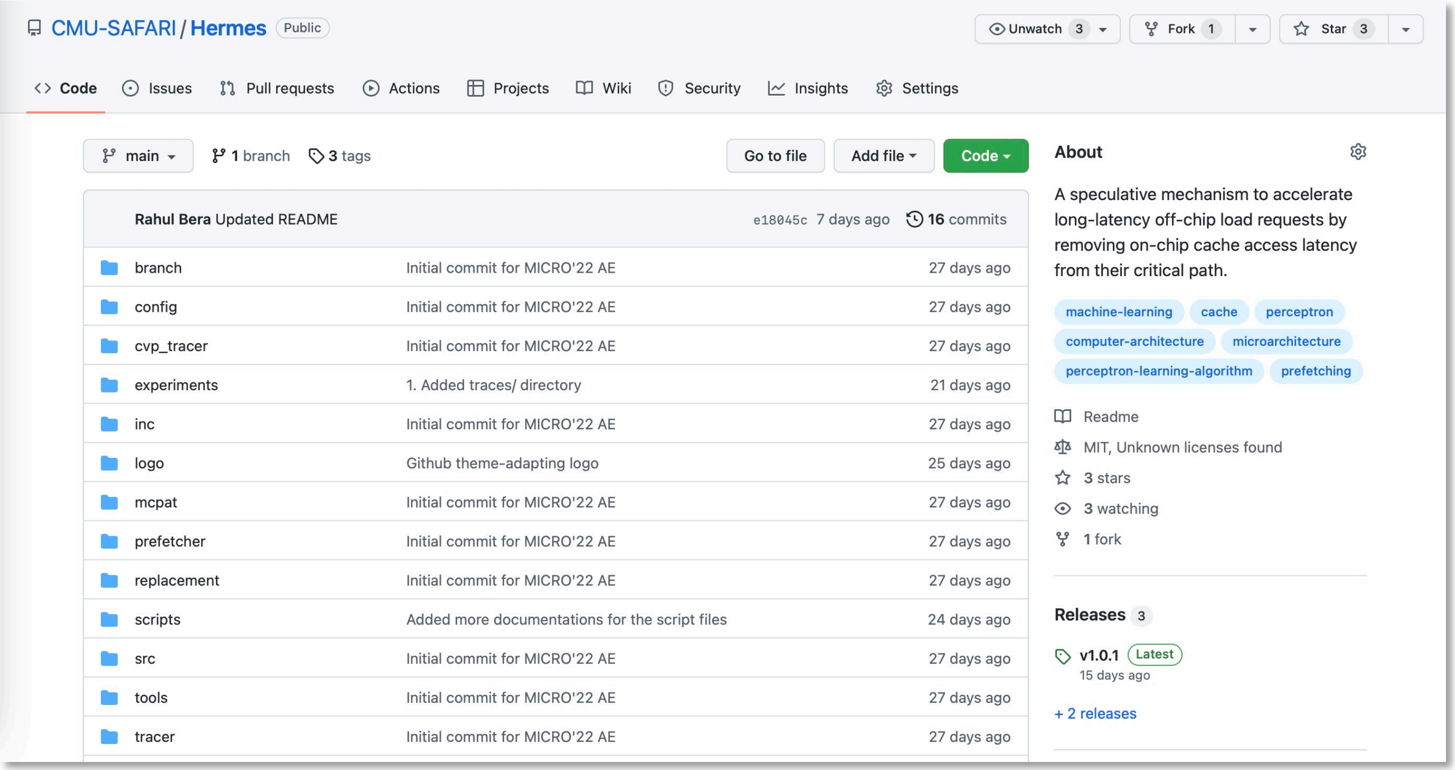
3

Realistic, practical implementation
Only 5.1 KB storage and 1.5% power overhead
of a desktop-class processor

Hermes is Open Source

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

- **All 3 badges from MICRO'22 artifact evaluation**
- **Champsim and McPAT source** code
- **All traces & scripts** used for evaluation



Rahul Bera Updated README e18045c 7 days ago 16 commits

branch	Initial commit for MICRO'22 AE	27 days ago
config	Initial commit for MICRO'22 AE	27 days ago
cvp_tracer	Initial commit for MICRO'22 AE	27 days ago
experiments	1. Added traces/ directory	21 days ago
inc	Initial commit for MICRO'22 AE	27 days ago
logo	Github theme-adapting logo	25 days ago
mcpat	Initial commit for MICRO'22 AE	27 days ago
prefetcher	Initial commit for MICRO'22 AE	27 days ago
replacement	Initial commit for MICRO'22 AE	27 days ago
scripts	Added more documentations for the script files	24 days ago
src	Initial commit for MICRO'22 AE	27 days ago
tools	Initial commit for MICRO'22 AE	27 days ago
tracer	Initial commit for MICRO'22 AE	27 days ago

About

A speculative mechanism to accelerate long-latency off-chip load requests by removing on-chip cache access latency from their critical path.

machine-learning cache perceptron computer-architecture microarchitecture perceptron-learning-algorithm prefetching

Readme
MIT, Unknown licenses found
3 stars
3 watching
1 fork

Releases 3

v1.0.1 Latest
15 days ago

+ 2 releases

HERMES

Accelerating Long-Latency Load Requests via Perceptron-based Off-chip Load Prediction

Rahul Bera, Konstantinos Kanellopoulos, Shankar Balachandran,
David Novo, Ataberk Olgun, Mohammad Sadrosadati, Onur Mutlu

SAFARI
SAFARI Research Group
safari.ethz.ch

ETH zürich

intel

LIRMM

Year III Results (2022 Annual Review 3)

- Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning [ISCA 2022]
- Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction [MICRO 2022]
- GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping [MICRO 2022]
- pLUTo: Enabling Massively Parallel Computation via In DRAM via Lookup Tables [MICRO 2022]
- DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression [FAST 2022]
- A Modern Primer on Processing in Memory [Arxiv, Updated 2022]

Year III Results (2022 Annual Review 3)

- Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning [ISCA 2022]

- Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction [MICRO 2022]

- GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping [MICRO 2022]

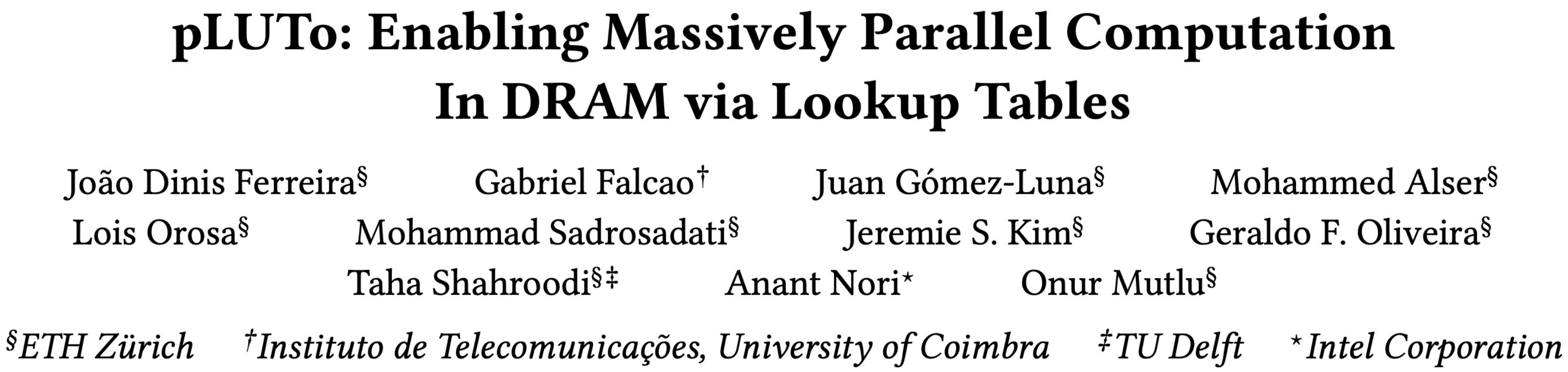
- pLUTo: Enabling Massively Parallel Computation via In DRAM via Lookup Tables [MICRO 2022]

- DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression [FAST 2022]

- A Modern Primer on Processing in Memory [Arxiv, Updated 2022]

pLUTo

- To Appear in MICRO 2022



pLUTo: Enabling Massively Parallel Computation In DRAM via Lookup Tables

João Dinis Ferreira[§] Gabriel Falcao[†] Juan Gómez-Luna[§] Mohammed Alser[§]
Lois Orosa[§] Mohammad Sadrosadati[§] Jeremie S. Kim[§] Geraldo F. Oliveira[§]
Taha Shahroodi[§‡] Anant Nori[⋆] Onur Mutlu[§]

[§]*ETH Zürich* [†]*Instituto de Telecomunicações, University of Coimbra* [‡]*TU Delft* [⋆]*Intel Corporation*

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

pLUTo: In-DRAM Lookup Tables to Enable General-Purpose Massively Parallel Computations

João Dinis Ferreira, Gabriel Falcao, Juan Gómez-Luna, Mohammed Alser, Geraldo F. Oliveira, Jeremie S. Kim, Mohammad Sadrosadati, Lois Orosa, Taha Shahroodi, Anant Nori, Onur Mutlu

SAFARI ETHzürich

August 2022

Executive Summary

- **Problem.** Many workloads require significant data movement. Existing Processing-using-Memory solutions mitigate this data movement but lack support for complex operations.
- **Key Idea.** LUTs enable general-purpose computation: perform LUT-based computation inside memory subarrays to perform complex operations.
- **Mechanism Overview.** With the LUT query operation, the elements in a source memory row are queried simultaneously in a LUT. In this way, it is possible to perform bulk LUT queries in-DRAM.
- **Key Contributions.**
  - Introduce support for bulk in-memory LUT querying for general-purpose in-memory computing.
  - Three implementations of pLUTo with varying area/performance/efficiency trade-offs.
- **Key Results.**
  - **Compared to CPU:** up to 33x faster and 110x more energy-efficient.
  - **Compared to GPU:** up to 8x faster and 80x more energy-efficient.

Year III Results (2022 Annual Review 3)

- Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning [ISCA 2022]

- Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction [MICRO 2022]

- GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping [MICRO 2022]

- pLUTo: Enabling Massively Parallel Computation via In DRAM via Lookup Tables [MICRO 2022]

- DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression [FAST 2022]

- A Modern Primer on Processing in Memory [Arxiv, Updated 2022]

DeepSketch

- Jisung Park, Jeonggyun Kim, Yeseong Kim, Sungjin Lee, and Onur Mutlu, **"DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression"** *Proceedings of the 20th USENIX Conference on File and Storage Technologies (**FAST**)*, Santa Clara, CA, USA, February 2022.
  [Slides (pptx) (pdf)]
  [Talk Video (15 minutes)]

DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression

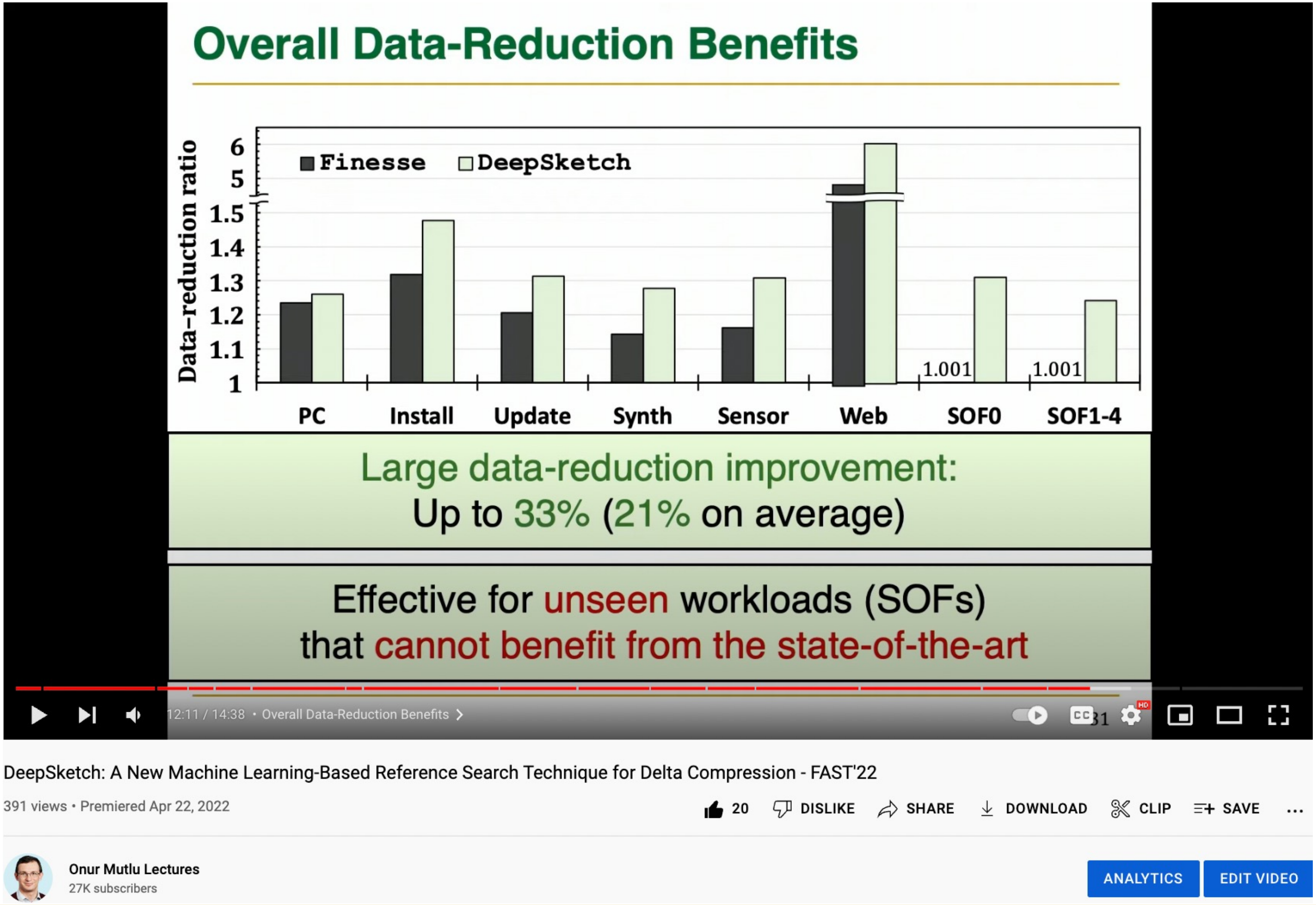
Jisung Park[1]* Jeonggyun Kim[2]* Yeseong Kim[2] Sungjin Lee[2] Onur Mutlu[1]
[1]*ETH Zürich* [2]*DGIST*

https://arxiv.org/pdf/2202.10584.pdf

Executive Summary

- **Motivation**
  - Data reduction: Effective at reducing the management cost of a data center by reducing the amount of data physically written to storage devices
  - Post-deduplication delta compression: Maximizes the data-reduction ratio by applying delta compression along with deduplication and lossless compression
- **Problem:** Existing post-deduplication delta-compression techniques provide significantly low data-reduction ratios compared to the optimal.
  - Due to the limited accuracy of reference search for delta compression
  - Cannot identify a good reference block for many incoming data blocks
- **Key Idea:** DeepSketch, a new machine learning-based reference search technique that uses the learning-to-hash method
  - Generates a given data block's signature (sketch) using a deep neural network
  - The higher the delta-compression benefit of two data blocks, the more similar the signatures of the two blocks to each other
- **Evaluation Results:** DeepSketch reduces the amount of physically-written data
  - Up to 33% (21% on average) compared to a state-of-the-art baseline

DeepSketch Talk Video



Overall Data-Reduction Benefits

Large data-reduction improvement:
Up to 33% (21% on average)

Effective for unseen workloads (SOFs)
that cannot benefit from the state-of-the-art

DeepSketch: A New Machine Learning-Based Reference Search Technique for Delta Compression - FAST'22

391 views • Premiered Apr 22, 2022

https://www.youtube.com/watch?v=RFdGyAJCk9M

Year III Results (2022 Annual Review 3)

- Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning [ISCA 2022]

- Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction [MICRO 2022]

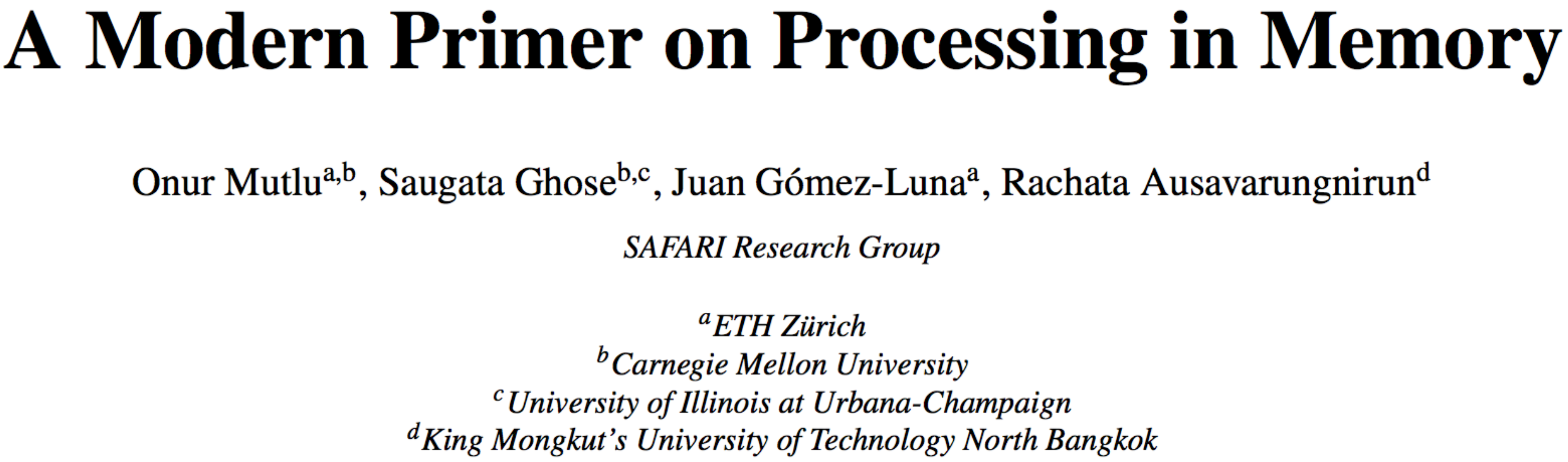
- GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping [MICRO 2022]

- pLUTo: Enabling Massively Parallel Computation via In DRAM via Lookup Tables [MICRO 2022]

- DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression [FAST 2022]

- A Modern Primer on Processing in Memory [Arxiv, Updated 2022]

PIM Review and Open Problems



A Modern Primer on Processing in Memory

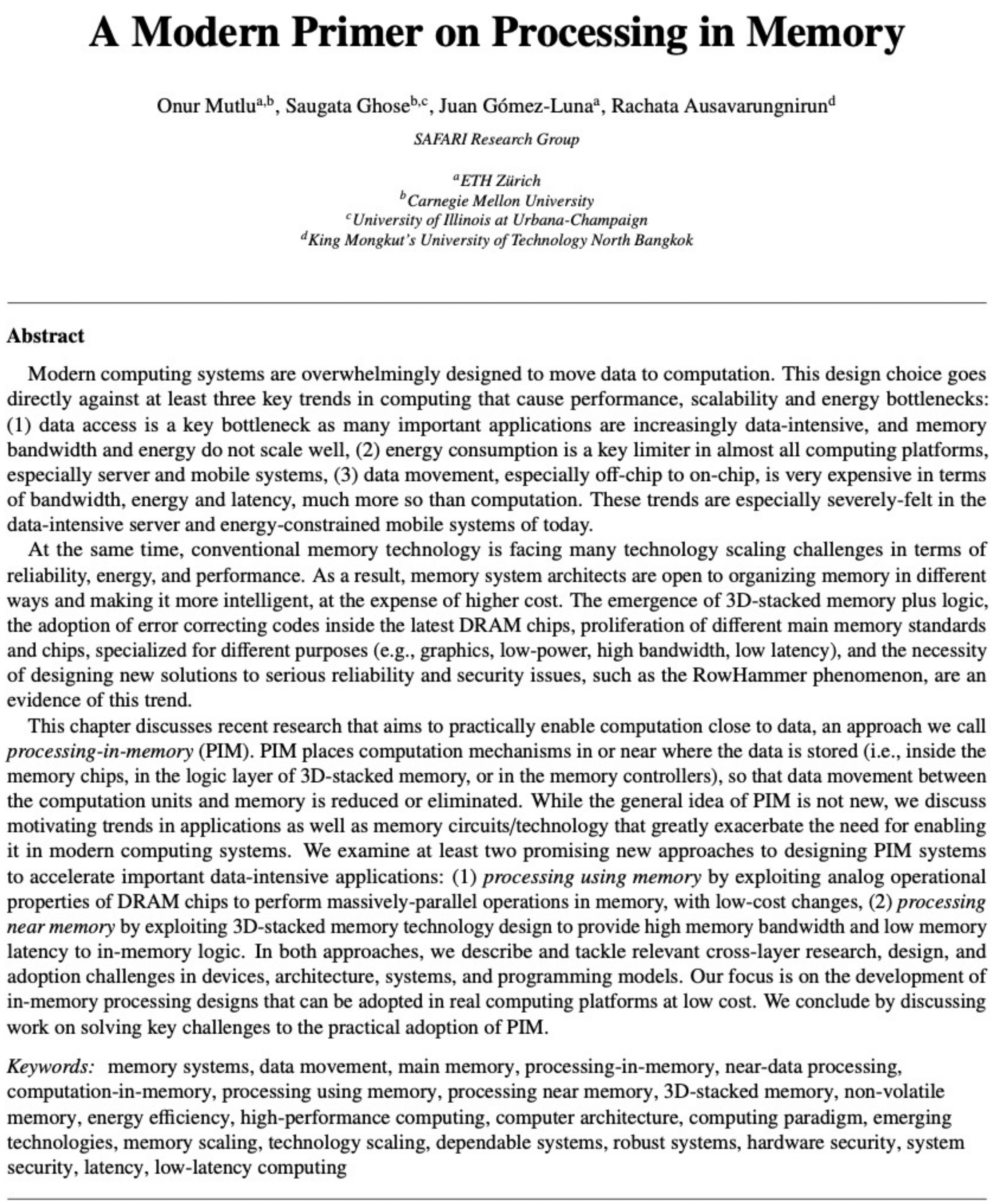
Onur Mutlu[a,b], Saugata Ghose[b,c], Juan Gómez-Luna[a], Rachata Ausavarungnirun[d]

SAFARI Research Group

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

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

https://arxiv.org/pdf/1903.03988.pdf



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

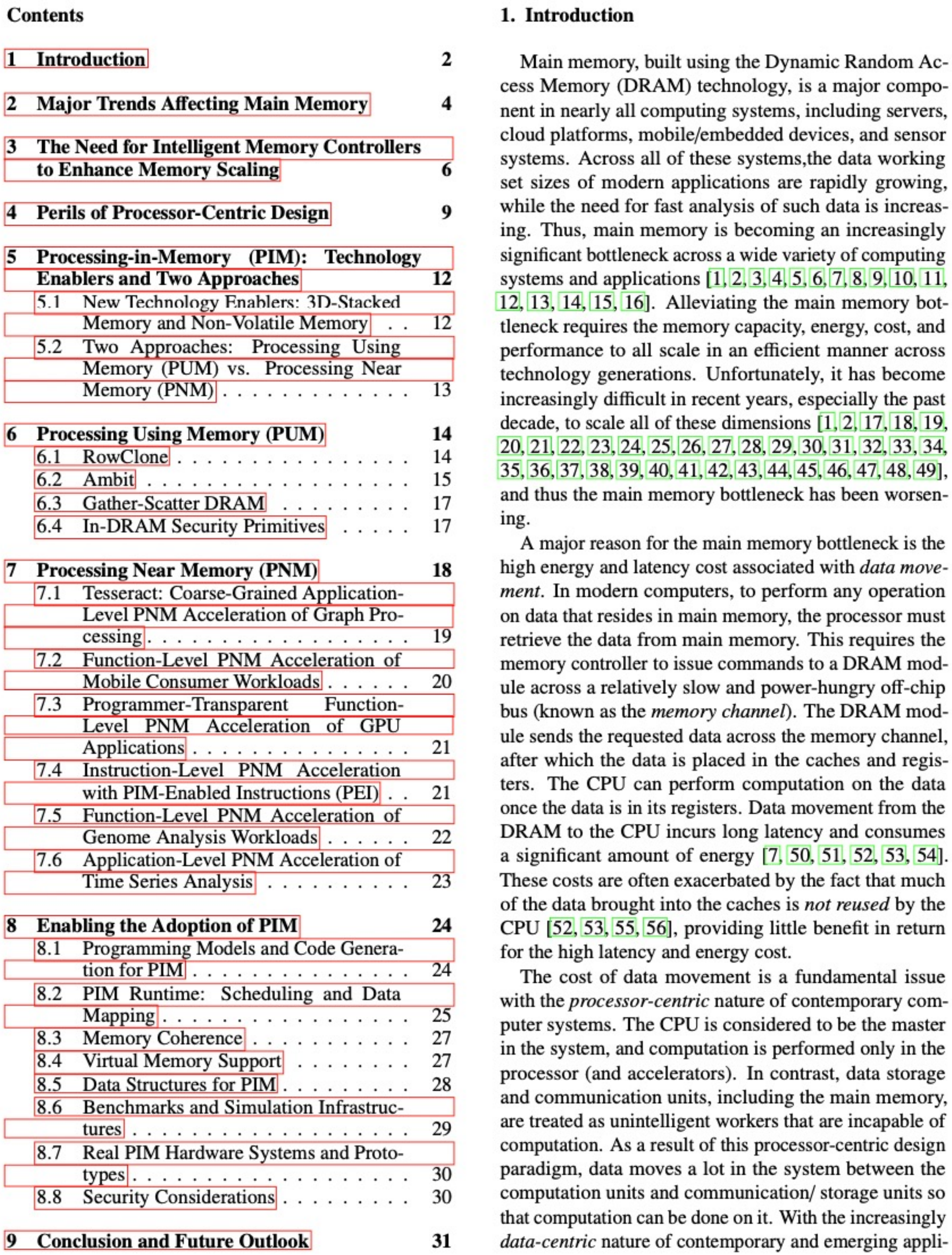
Abstract

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



Contents

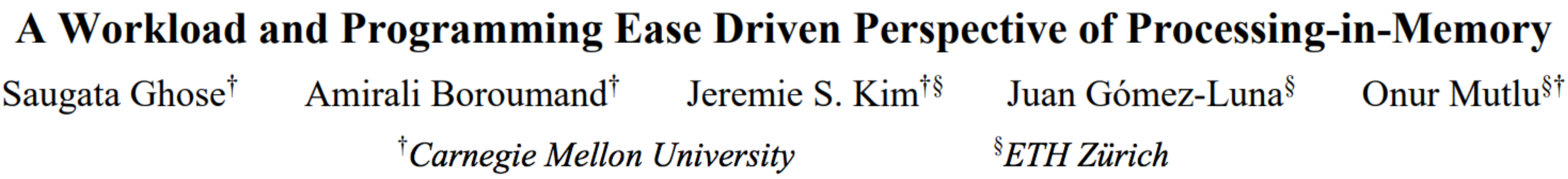
1. Introduction

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-

PIM Review and Open Problems (II)



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

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

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

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

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

Year III Results (2022 Annual Review 4)

- EcoFlow: Efficient Convolutional Dataflows for Low-Power Neural Network Accelerators [arXiv 2022] https://arxiv.org/abs/2202.02310

- ApHMM: Accelerating Profile Hidden Markov Models for Fast and Energy-Efficient Genome Analysis [arXiv 2022] https://arxiv.org/abs/2207.09765

- Accelerating Weather Prediction Using Near-Memory Reconfigurable Fabric [TRETS 2022] https://arxiv.org/abs/2107.08716

Memory System Design for AI/ML Accelerators & ML/AI Techniques for Memory System Design

Onur Mutlu

omutlu@gmail.com

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

30 August 2022

SRC AIHW Annual Review

SAFARI ETH zürich Carnegie Mellon