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

29 September 2021

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

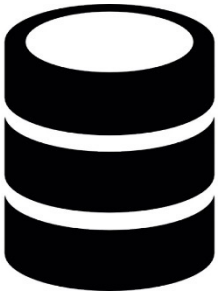
Computing

is Bottlenecked by Data

Data is Key for AI, ML, Genomics, …

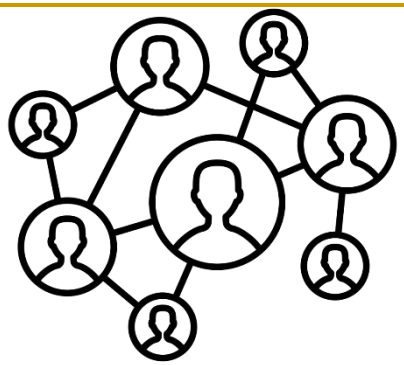
- Important workloads are all data intensive
- They require rapid and efficient processing of large amounts of data
- Data is increasing
  - We can generate more than we can process

Data is Key for Future Workloads



In-memory Databases

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

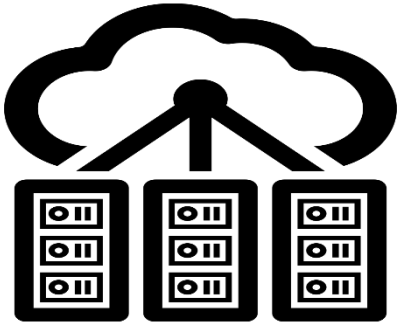


Graph/Tree Processing

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

In-Memory Data Analytics

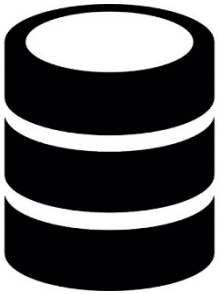
[Clapp+ (**Intel**), IISWC'15;
Awan+, BDCloud'15]



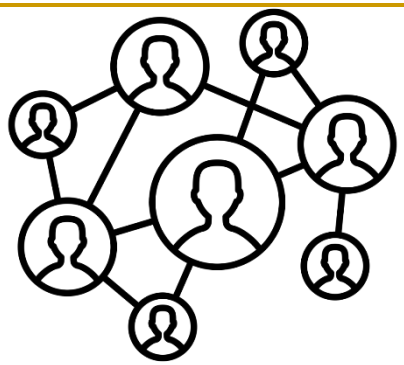
Datacenter Workloads

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

Data Overwhelms Modern Machines



In-memory Databases

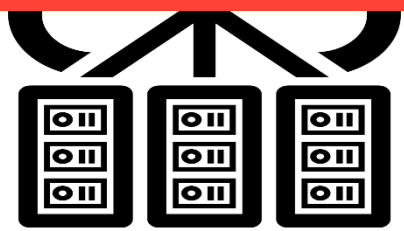


Graph/Tree Processing

Data → performance & energy bottleneck

In-Memory Data Analytics

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



Datacenter Workloads

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

Data is Key for Future Workloads

Chrome

Google's web browser

TensorFlow Mobile

Google's machine learning framework

VP9

YouTube

Video Playback

Google's video codec

VP9

YouTube

Video Capture

Google's video codec

Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data → performance & energy bottleneck

VP9

YouTube

Video Playback

Google's **video codec**

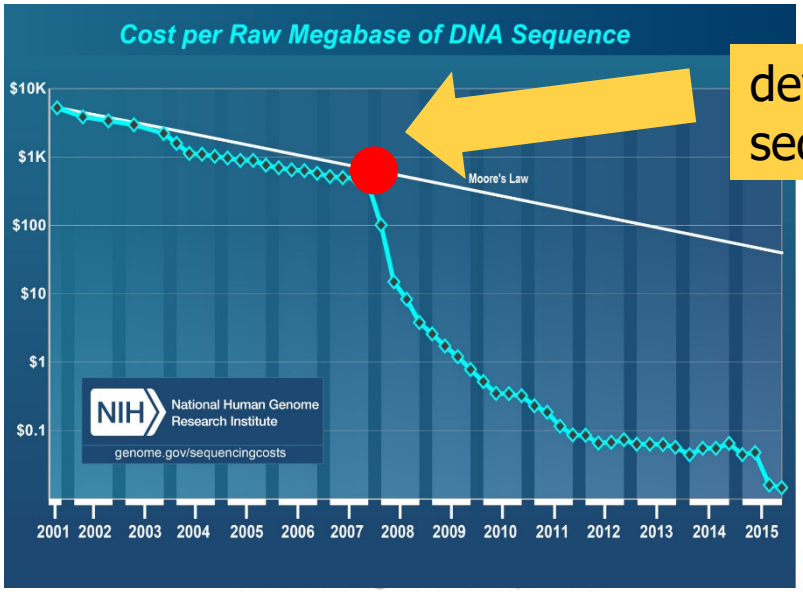
VP9

YouTube

Video Capture

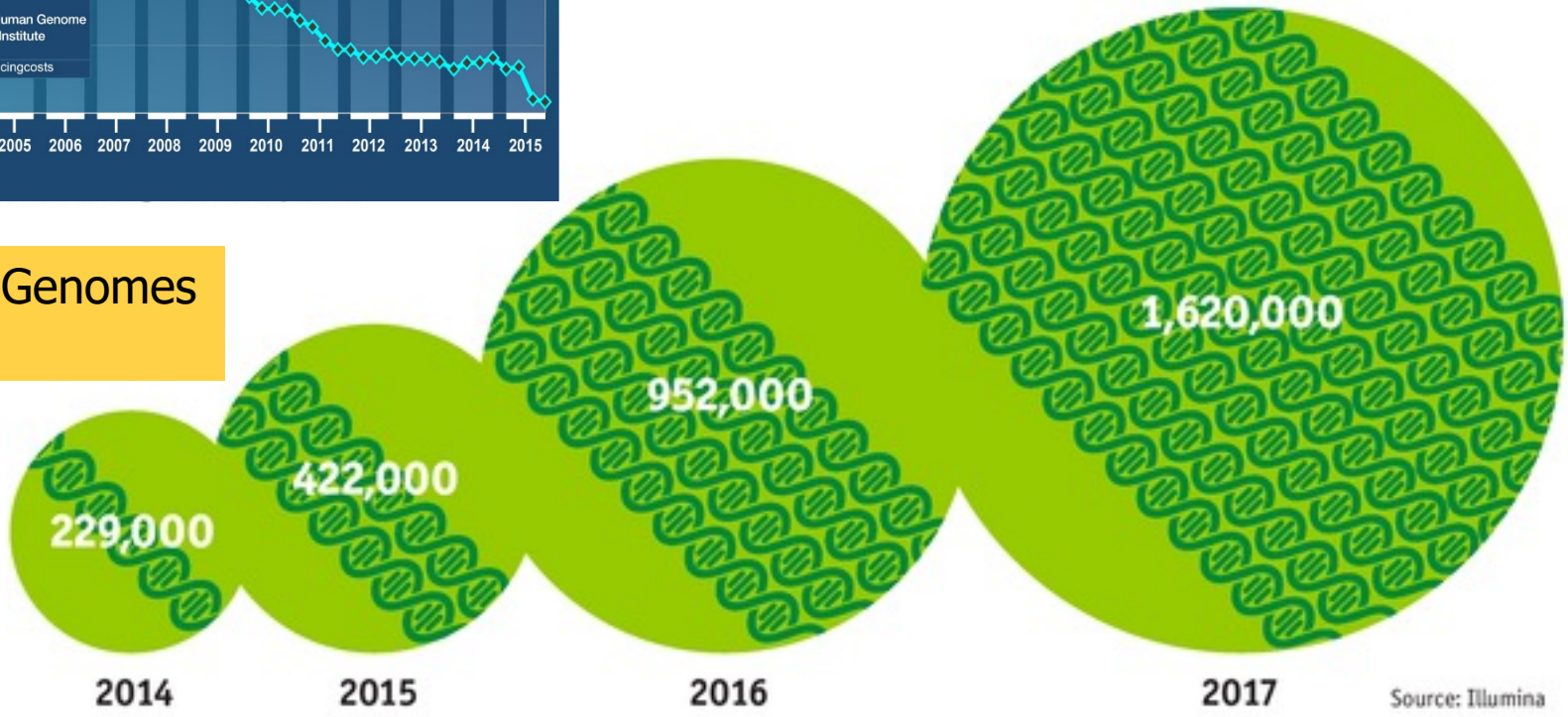
Google's **video codec**

Data is Key for Future Workloads



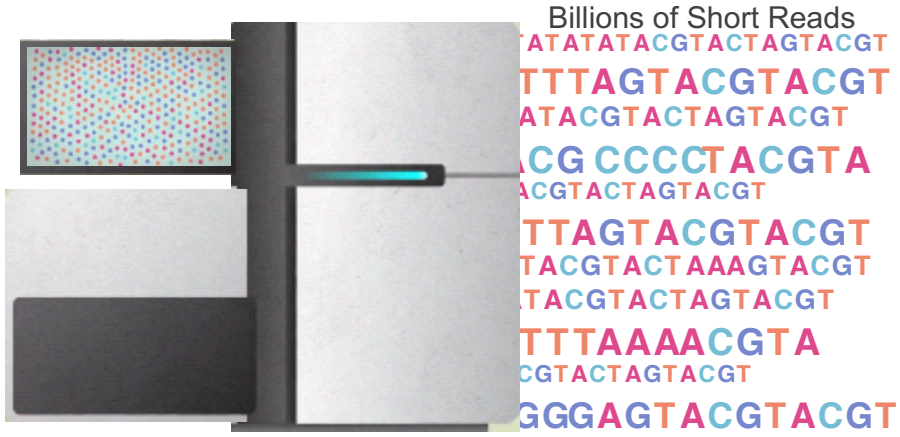
development of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced

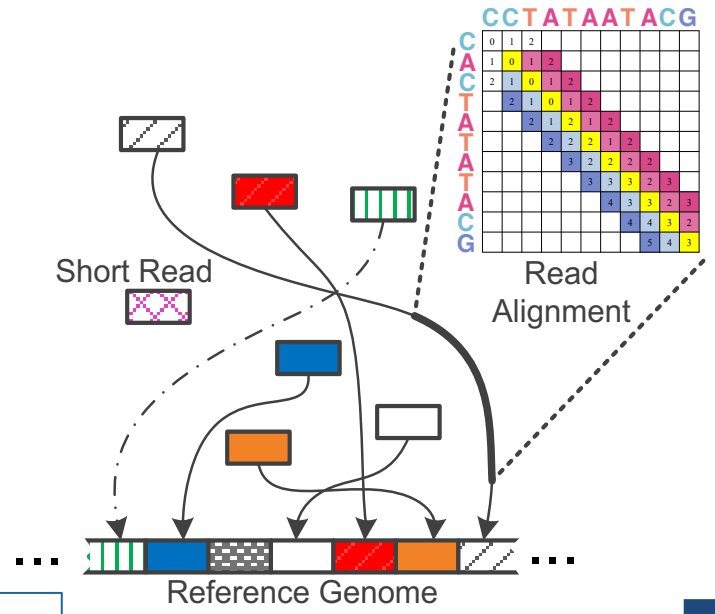


The Economist

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



1 Sequencing



Read Mapping 2

Genome Analysis

Data → performance & energy bottleneck

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

3 Variant Calling

Scientific Discovery 4

New Genome Sequencing Technologies

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

VP9

YouTube

Video Playback

Google's **video codec**

VP9

YouTube

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]

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: Architectures Today …

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

Architectures for Intelligent Machines

Data-centric

Data-driven

Data-aware

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

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

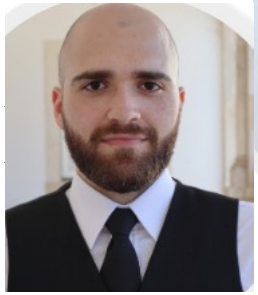
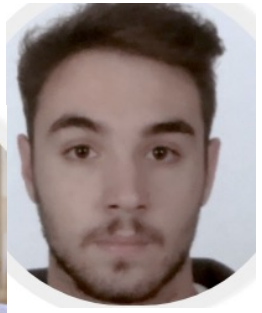
Final report summarizing research accomplishments and future direction

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)
  - Amirali Boroumand (CMU)
  - Geraldo Francisco de Oliveira Junior (ETH)
  - Joao Ferreira (ETH)
  - Konstantinos Kanellopoulos (ETH
  - Damla Senol Cali (CMU)




Task Information #2946.001 (2)

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

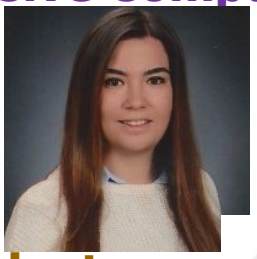



- More students/postdocs to be added as the task evolves

Recent PhD Graduates (I)

- Amirali Boroumand
  - December 2020
  - **Practical Mechanisms for Reducing Processor-Memory Data Movement in Modern Workloads**

- Gagandeep Singh
  - April 2021
  - **Designing, Modeling, and Optimizing Data-Intensive Computing Systems**



- Damla Senol Cali
  - August 2021
  - **Accelerating Genome Sequence Analysis via Efficient Hardware/Algorithm Co-Design**

- Nastaran Hajinazar
  - August 2021
  - **Data-Centric and Data-Aware Frameworks for Fundamentally Efficient Data Handling in Modern Computing Systems**

Recent PhD Graduates (II)

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

- Dr. Saugata Ghose
  - Started @ UIUC as Assistant Professor, Fall 2020

Soon to Finish PhD

- Minesh Patel
  - Defense date: October 1, 2021
  - **Enabling Effective Error Mitigation in Memory Chips That Use On-Die Error-Correcting Codes**

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**
- **Anurag Kar, ARM**
- **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

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

Memory System Design for AI/ML

- Some background works from the past

- **"EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM", MICRO 2019**

- **"SMASH: Co-designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations", MICRO 2019**

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

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

AI/ML for Memory System Design

- Some background works from the past

- **"Self Optimizing Memory Controllers: A Reinforcement Learning Approach", ISCA 2008**

- **"NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning", DAC 2019**

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]

Second Year Results: More Detail

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]

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

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

PACT 2021

SAFARI

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

Executive Summary

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

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

Problem: The Edge TPU accelerator suffers from three challenges:

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

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

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

Key Mechanism: A new framework called Mensa

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

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

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

Google Edge NN Models

We analyze inference execution using 24 edge NN models

Speech Recognition

6 RNN Transducers

Google Edge TPU

2 LSTMs

Language Translation

Face Detection

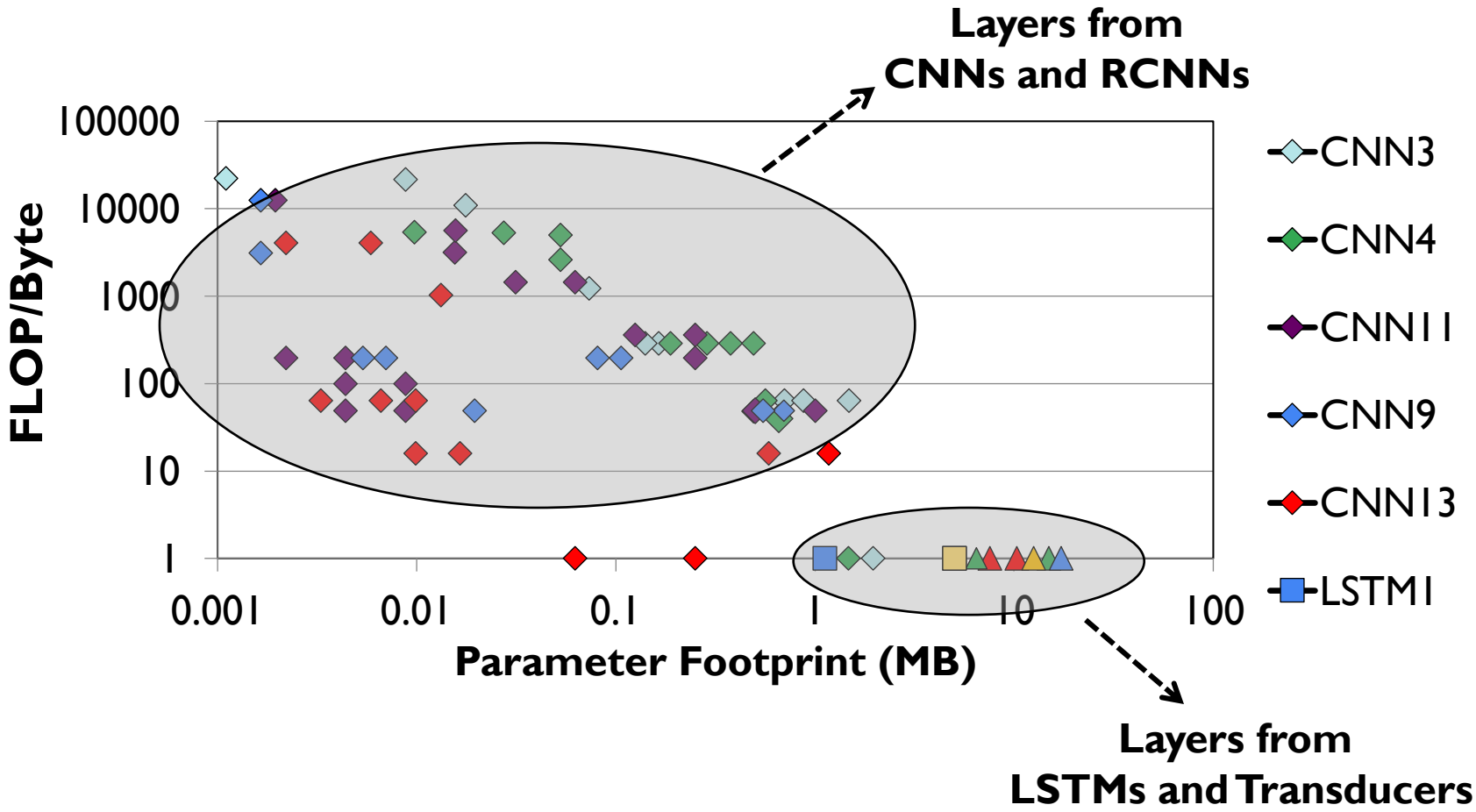
13 CNN

3 RCNN

Image Captioning

Diversity Across the Models

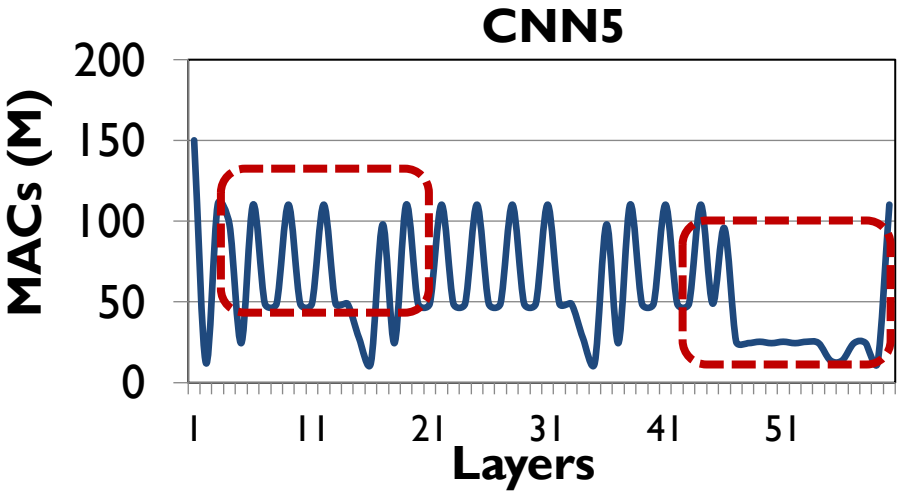
Insight 1: there is significant variation in terms of layer characteristics across the models

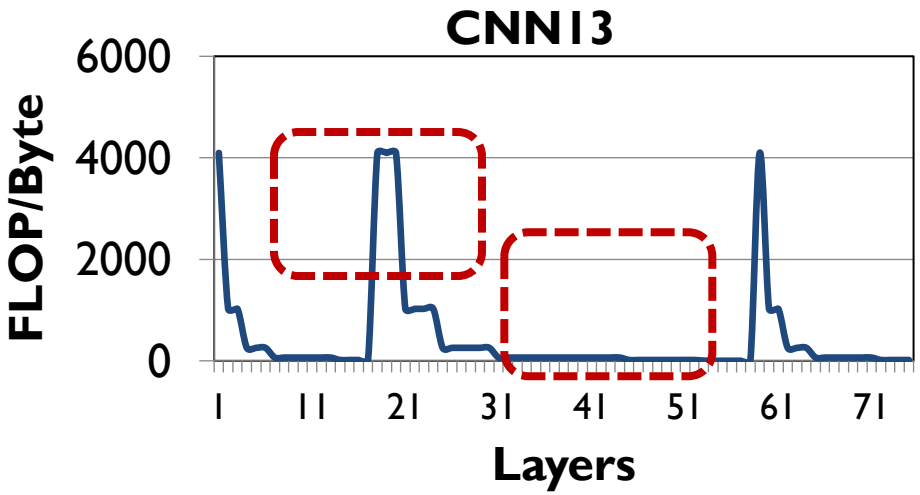


Diversity Within the Models

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

For example, our analysis of edge CNN models shows:



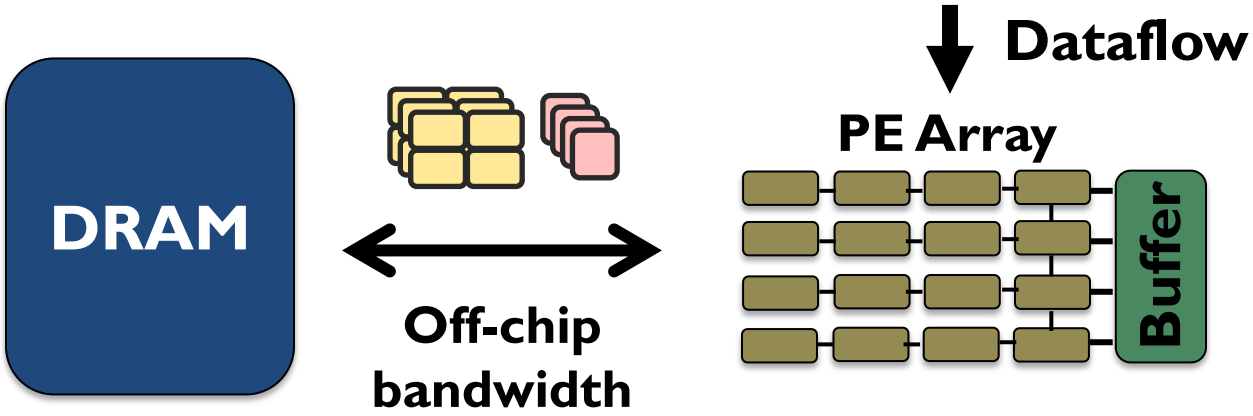


Variation in MAC intensity: up to 200x across layers

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

Root Cause of Accelerator Challenges

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



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

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

Mensa Framework

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

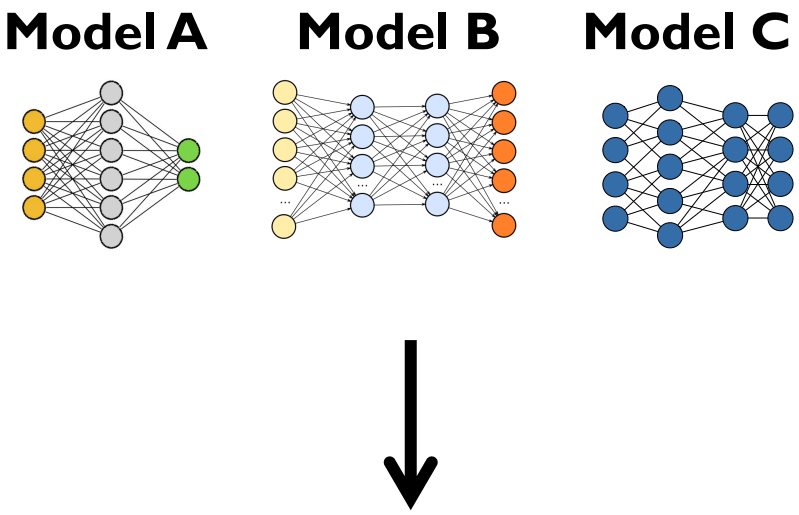
Instead of running the entire NN model on a monolithic accelerator:

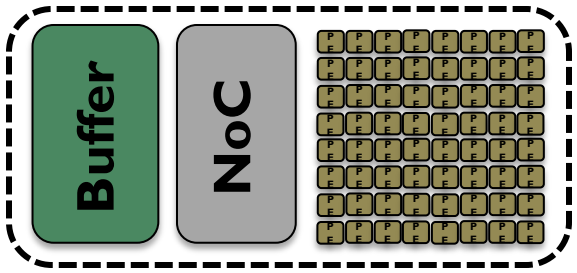
↓

Mensa: a new acceleration framework for edge NN inference

Mensa High-Level Overview

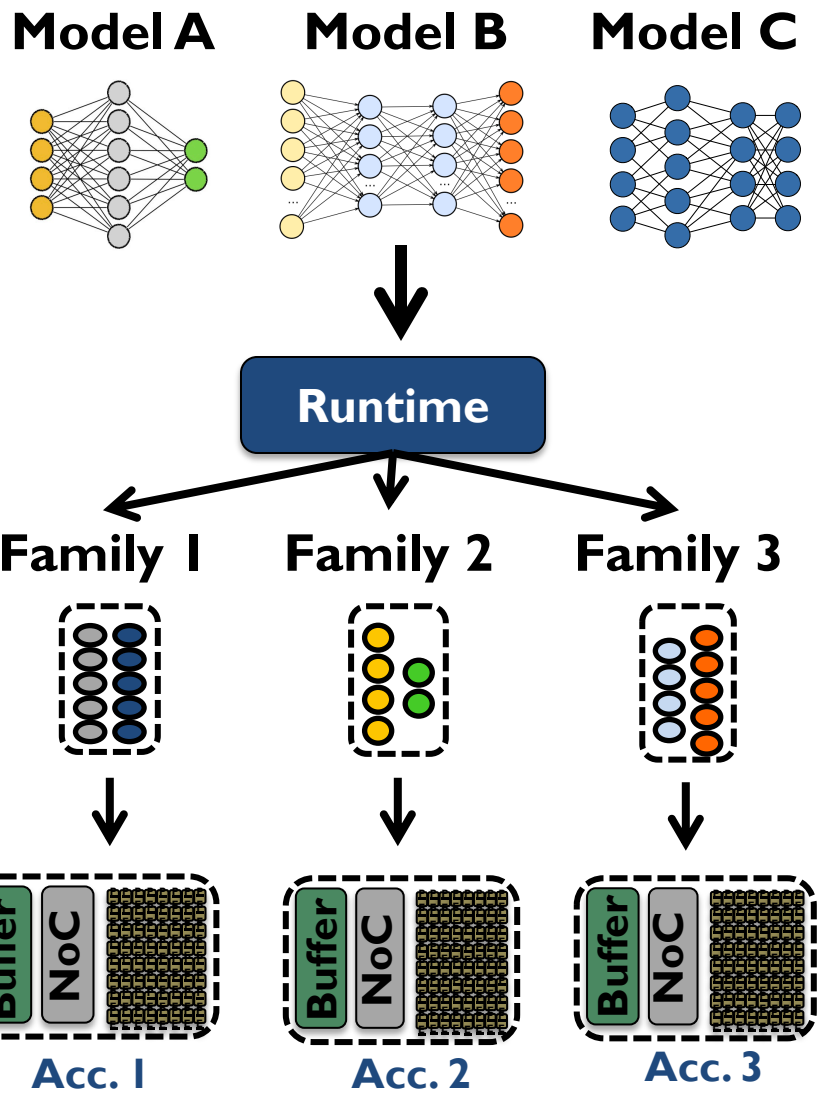
Edge TPU Accelerator





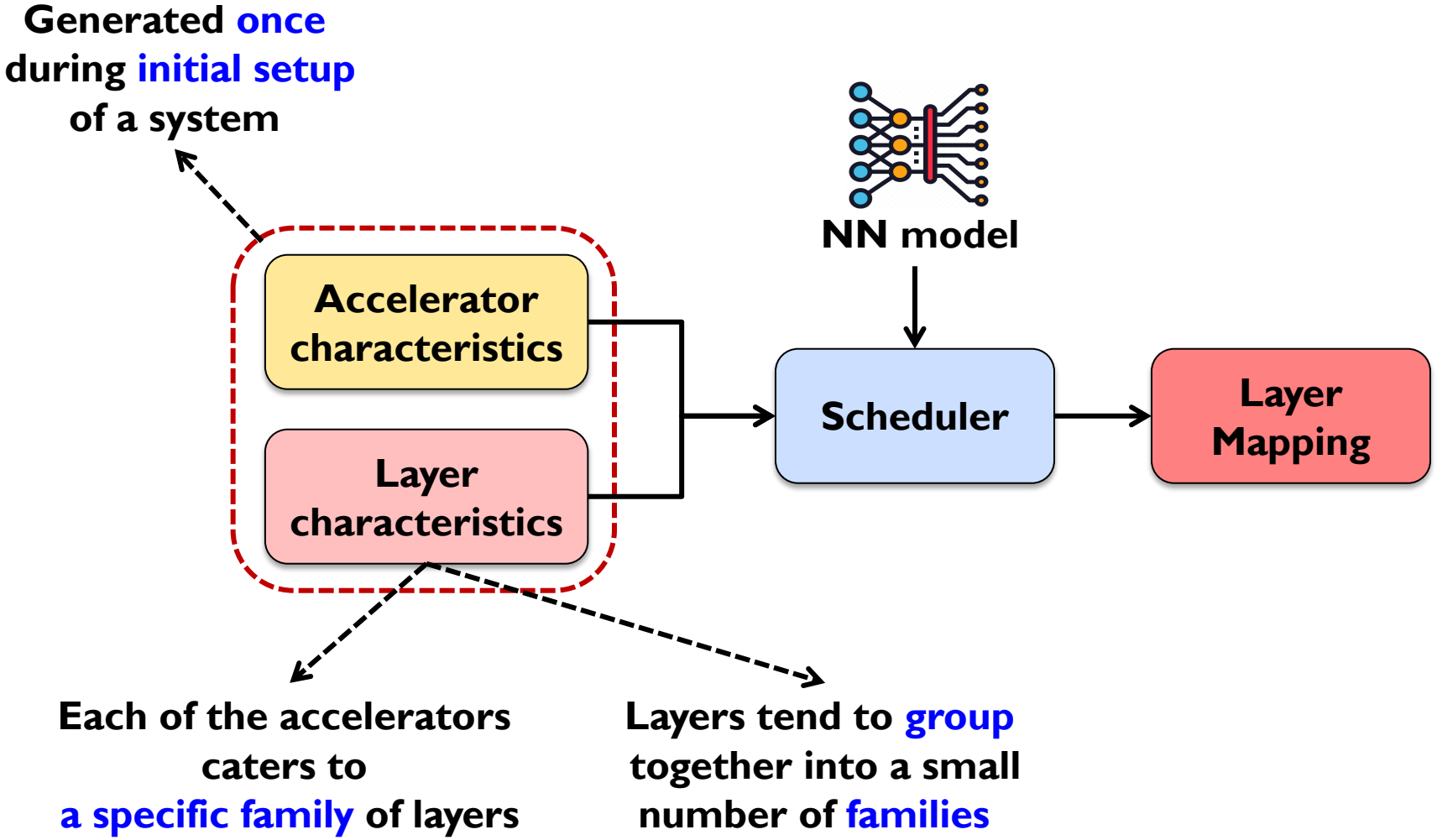
Monolithic Accelerator

Mensa



Mensa Runtime Scheduler

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



Mensa Runtime Scheduler

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

Generated once during initial setup

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

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

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

Layer characteristics

Each of the accelerators caters to a specific family of layers

Layers tend to group together into a small number of families

For More on Mensa [PACT 2021]

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

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

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

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]

Pythia

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

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

Executive Summary

- **Background**: Prefetchers learns to predict future addresses by associating patterns with program context (called **feature**)
- **Problem**: Three key shortcomings in prior prefetchers:
  - Predicts mainly using a **single program feature**
  - Lacks **inherent system awareness**
  - Lacks **online customization** ability
- **Goal**: Design a prefetching framework that:
  - Learns from **multiple features** and **inherent system-level feedback**
  - Can be **customized online** to change features and/or objective
- **Contribution**: **Pythia**, that formulates prefetching as reinforcement learning
  - **Adaptive, autonomous** learning using multiple features and system-level feedback
  - **Realistic, practical** implementation without any changes to software
- **Key Results**:
  - Evaluated using a wide range of workloads from SPEC CPU, PARSEC, Ligra, Cloudsuite
  - Outperforms prior SOTA by **3.4%, 7.7% and 16.9%** in 1/4/bandwidth-constrained cores
- **Open sourced**: https://github.com/CMU-SAFARI/Pythia

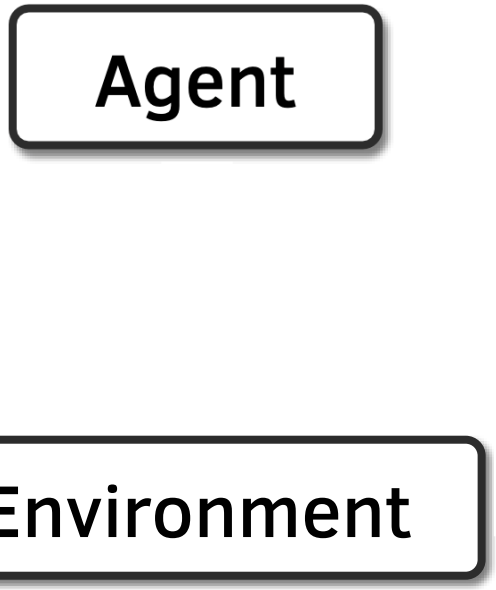
Our Goal

A prefetching **framework** that:

1. Can learn to prefetch using **multiple features** and **inherent system-level feedback** information
2. Can be **easily customized in silicon** to change feature type and/or prefetcher's objective

Basics of Reinforcement Learning (RL)

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

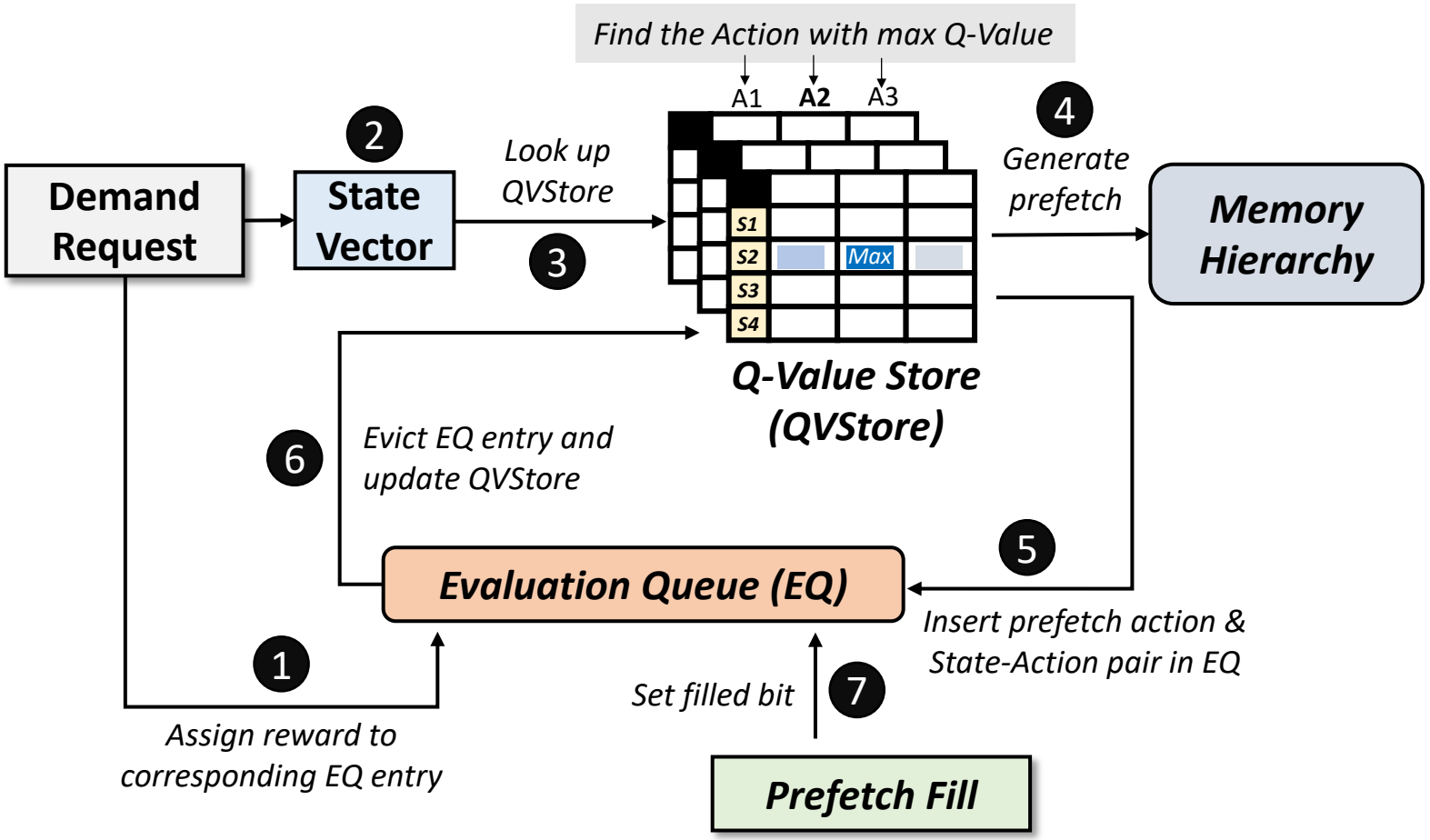


- Agent stores **Q-values** for *every* state-action pairs
  - Given a state, selects action that provides **highest** Q-value

Formulating Prefetching as RL

Pythia Overview

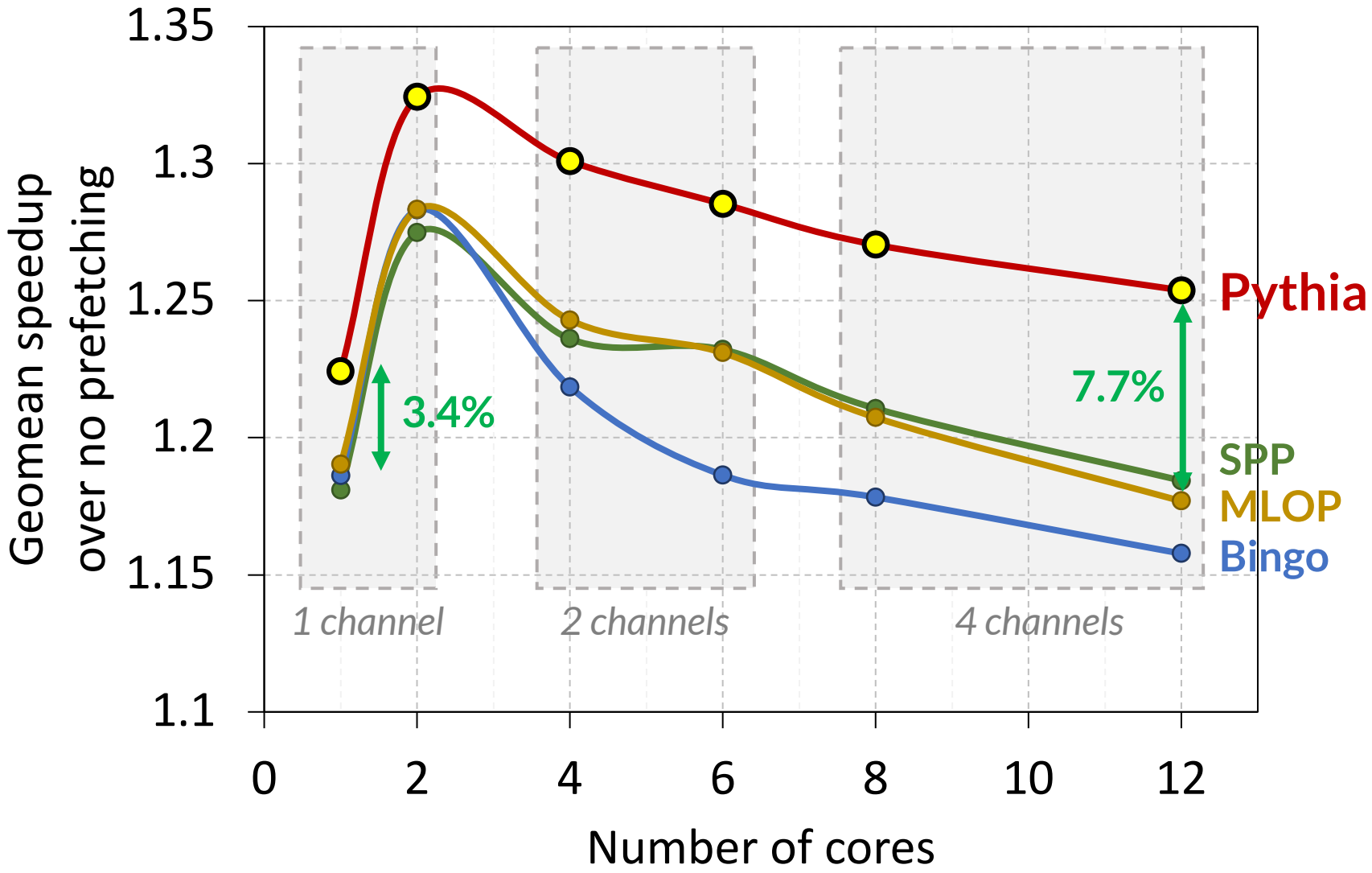
- **Q-Value Store**: Records Q-values for *all* state-action pairs
- **Evaluation Queue**: A FIFO queue of recently-taken actions



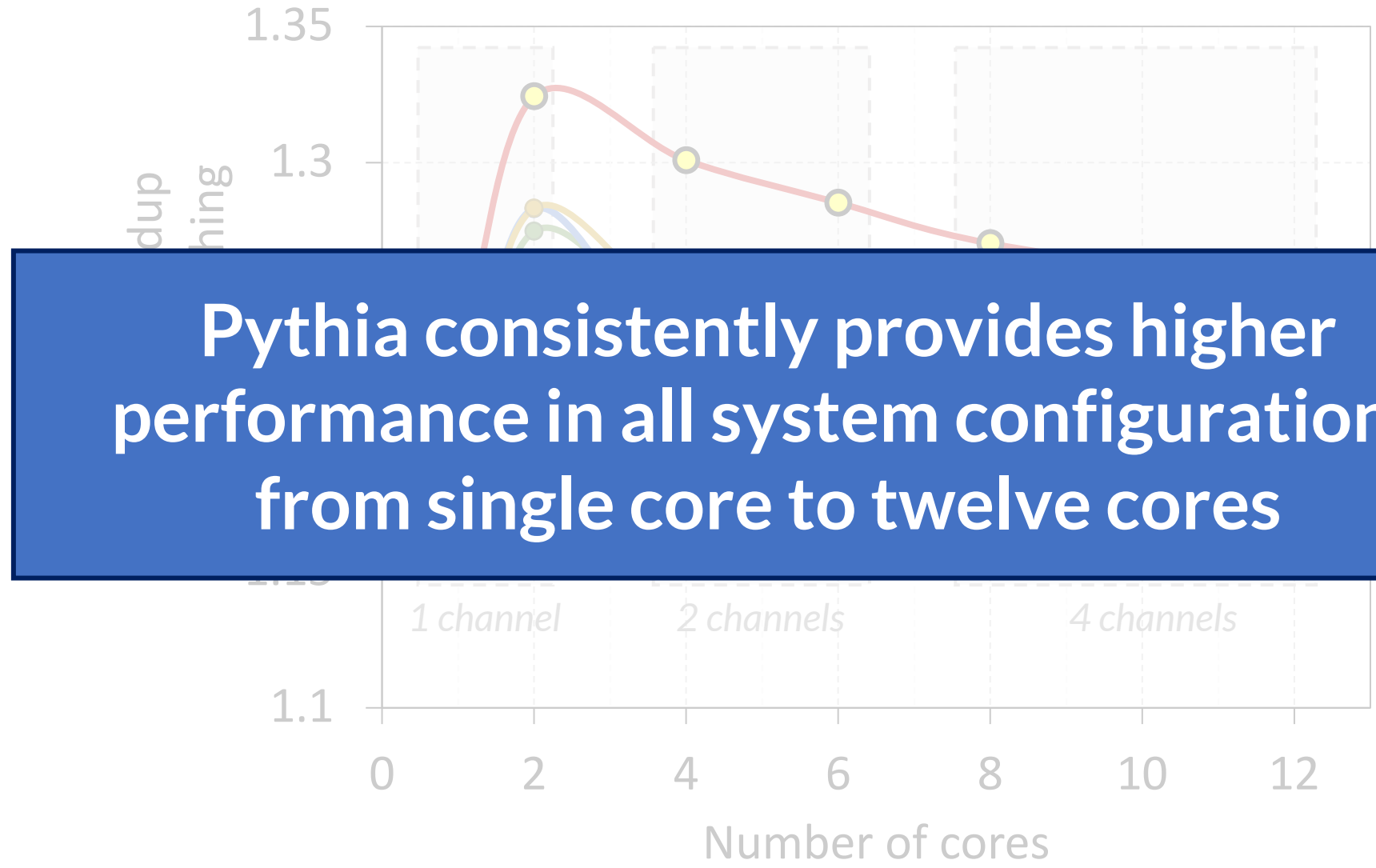
Simulation Methodology

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

Performance with Varying Core Count



Performance with Varying Core Count

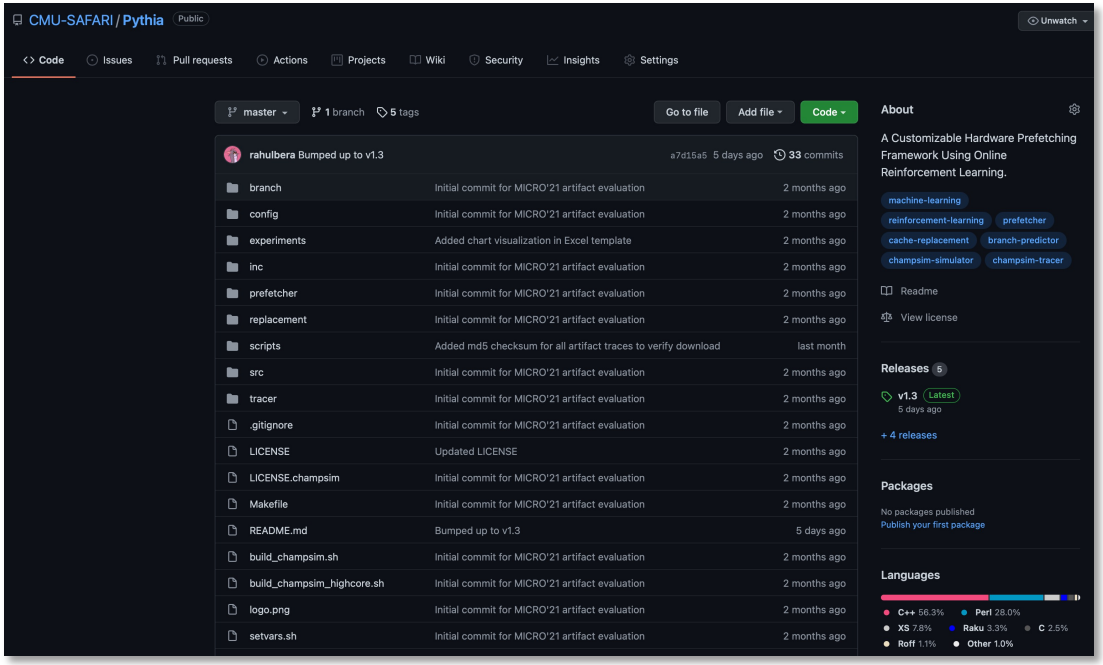


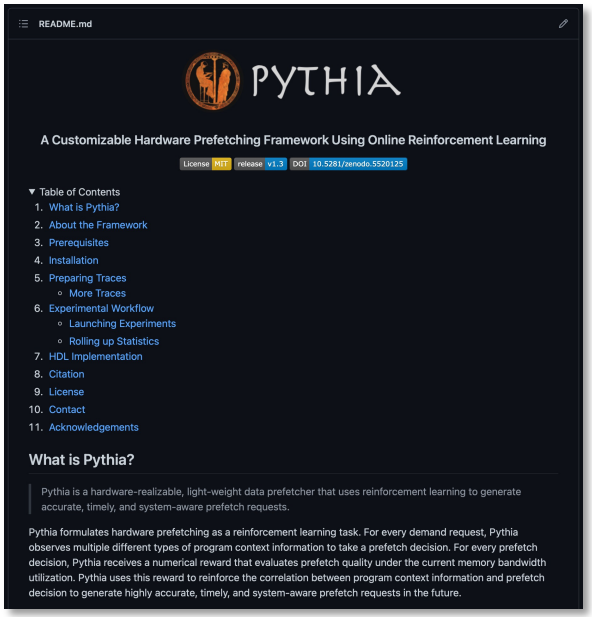
Pythia consistently provides higher performance in all system configurations from single core to twelve cores

Pythia is Completely Open Source

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

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





More in the Paper

- Automatic design-space exploration for Pythia
- Details about reward assignment and QVStore update

Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera[1] Konstantinos Kanellopoulos[1] Anant V. Nori[2] Taha Shahroodi[3,1]
Sreenivas Subramoney[2] Onur Mutlu[1]
[1]ETH Zürich [2]Processor Architecture Research Labs, Intel Labs [3]TU Delft

- ✓Performance comparison with unseen traces
- ✓Understanding Pythia's learning with a case study
- ✓Performance benefits via customization

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]

Refresh Triggered Computation

Refresh Triggered Computation: Improving the Energy Efficiency of Convolutional Neural Network Accelerators

SYED M. A. H. JAFRI, KTH Royal Institute of Technology
HASAN HASSAN, ETH Zürich
AHMED HEMANI, KTH Royal Institute of Technology
ONUR MUTLU, ETH Zürich

To employ a Convolutional Neural Network (CNN) in an energy-constrained embedded system, it is critical for the CNN implementation to be highly energy efficient. Many recent studies propose CNN accelerator architectures with custom computation units that try to improve energy-efficiency and performance of CNNs by minimizing data transfers from DRAM-based main memory. However, in these architectures, DRAM is still responsible for half of the overall energy consumption of the system, on average. A key factor of the high energy consumption of DRAM is the *refresh overhead*, which is estimated to consume 40% of the total DRAM energy.

In this paper, we propose a new mechanism, *Refresh Triggered Computation (RTC)*, that exploits the memory access patterns of CNN applications to reduce the number of *refresh operations*. RTC uses two major techniques to mitigate the refresh overhead. First, *Refresh Triggered Transfer (RTT)* is based on our *new* observation that a CNN application accesses a large portion of the DRAM in a predictable and recurring manner. Thus, the read/write accesses of the application inherently refresh the DRAM, and therefore a significant fraction of refresh operations can be skipped. Second, *Partial Array Auto-Refresh (PAAR)* eliminates the refresh operations to DRAM regions that do not store any data.

We propose three RTC designs (min-RTC, mid-RTC, and full-RTC), each of which requires a different level of aggressiveness in terms of customization to the DRAM subsystem. All of our designs have small overhead. Even the most aggressive RTC design (i.e., full-RTC) imposes an area overhead of only 0.18% in a 16 *Gb* DRAM chip and can have less overhead for denser chips. Our experimental evaluation on six well-known CNNs show that RTC reduces average DRAM energy consumption by 24.4% and 61.3%, for the least aggressive and the most aggressive RTC implementations, respectively. Besides CNNs, we also evaluate our RTC mechanism on three workloads from other domains. We show that RTC saves 31.9% and 16.9% DRAM energy for *Face Recognition* and *Bayesian Confidence Propagation Neural Network (BCPNN)*, respectively. We believe RTC can be applied to other applications whose memory access patterns remain predictable for a sufficiently long time.

Refresh Triggered Computation

- Syed M. A. H. Jafri, Hasan Hassan, Ahmed Hemani, and Onur Mutlu, **"Refresh Triggered Computation: Improving the Energy Efficiency of Convolutional Neural Network Accelerators"** *ACM Transactions on Architecture and Code Optimization* (***TACO***), December 2020.

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]

SynCron

Efficient Synchronization Support for Near-Data-Processing Architectures

Christina Giannoula
christina.giann@gmail.com

Nandita Vijaykumar, Nikela Papadopoulou, Vasileios Karakostas
Ivan Fernandez, Juan Gómez Luna, Lois Orosa
Nectarios Koziris, Georgios Goumas, Onur Mutlu

SAFARI

ETH zürich

National Technical University of Athens
CSLab

UNIVERSITY OF TORONTO

UNIVERSIDAD DE MÁLAGA

Executive Summary

Problem:

Synchronization support is **challenging** for NDP systems

Prior schemes are **not suitable** or **efficient** for NDP systems

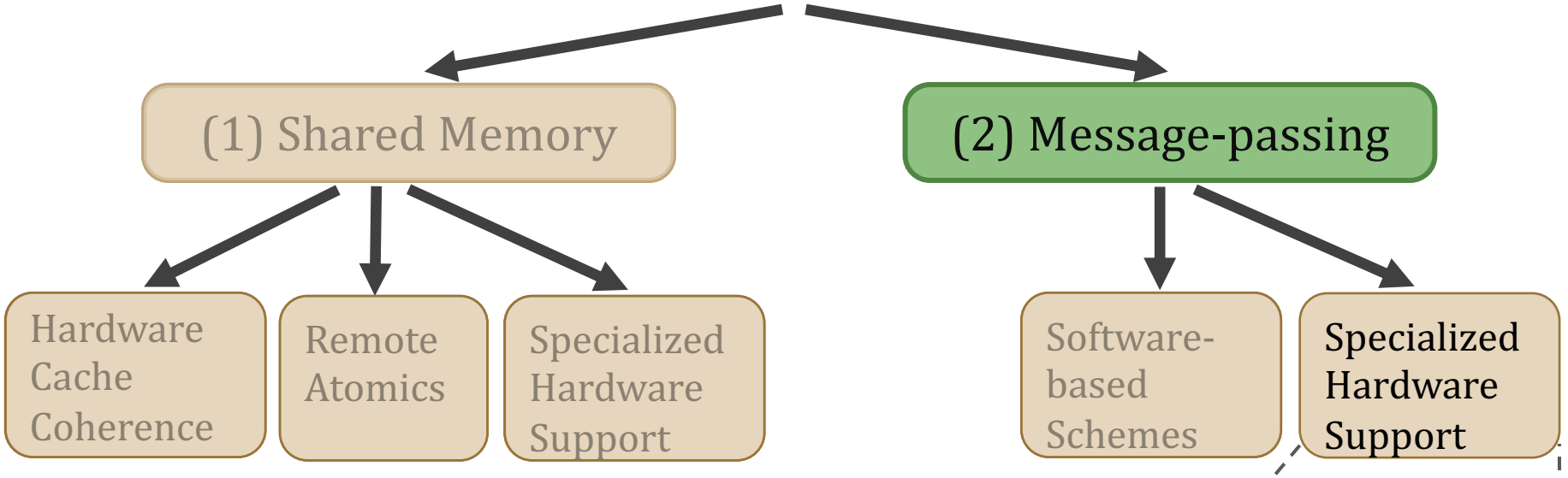
Contribution:

SynCron: the **first end-to-end** synchronization solution for NDP architectures

Key Results:

SynCron comes within **9.5%** and **6.2%** of performance and energy of an **Ideal** zero-overhead synchronization scheme

NDP Synchronization Solution Space

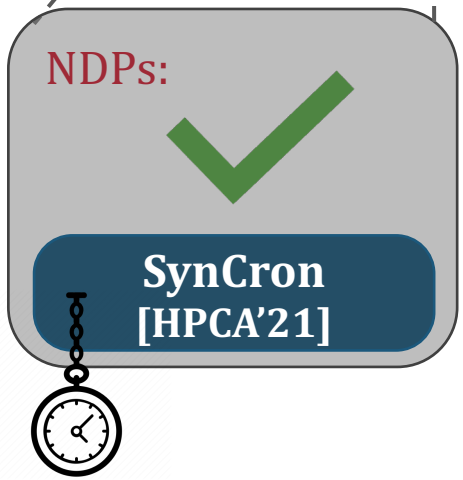


SynCron's Design Choices

Hardware Message-passing
to Avoid Synchronization via Shared Memory

Hierarchical Communication
to Eliminate Expensive Network Traffic

Specialized Cache Structure
to Minimize Latency Costs



Summary & Conclusion

- Synchronization is a **major system challenge** for NDP systems
- **Prior** schemes are **not suitable** or **efficient** for NDP systems
- **SynCron** is the **first end-to-end** synchronization solution for NDP architectures
- Syncron consists of **four** key techniques:
  i. **Hardware support** for synchronization acceleration
  ii. **Direct buffering** of synchronization variables
  iii. **Hierarchical** message-passing **communication**
  iv. Integrated hardware-only **overflow management**
- SynCron's benefits: **90.5%** and **93.8%** of performance and energy of an **Ideal** zero-overhead scheme
- SynCron is **highly-efficient**, **low-cost**, **easy-to-use**, and **general** to support many synchronization primitives

More on SynCron

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

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

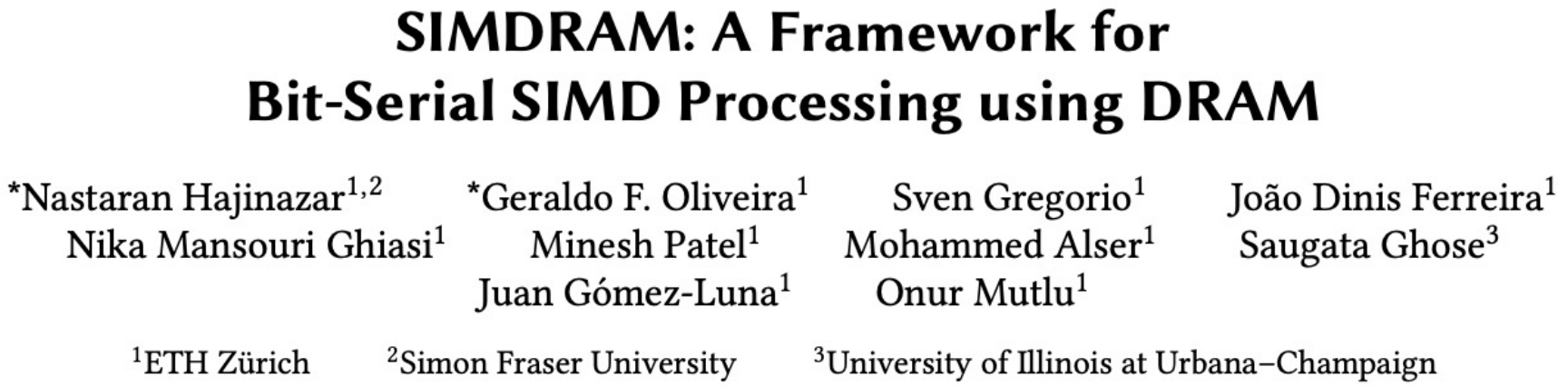
Christina Giannoula[†‡] Nandita Vijaykumar[*‡] Nikela Papadopoulou[†] Vasileios Karakostas[†] Ivan Fernandez[§‡]
Juan Gómez-Luna[‡] Lois Orosa[‡] Nectarios Koziris[†] Georgios Goumas[†] Onur Mutlu[‡]
[†]*National Technical University of Athens* [‡]*ETH Zürich* [*]*University of Toronto* [§]*University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

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]

SIMDRAM Framework

- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu, **"SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"** *Proceedings of the 26th International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Virtual, March-April 2021.
  [2-page Extended Abstract]
  [Short Talk Slides (pptx) (pdf)]
  [Talk Slides (pptx) (pdf)]
  [Short Talk Video (5 mins)]
  [Full Talk Video (27 mins)]



SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

*Nastaran Hajinazar[1,2] *Geraldo F. Oliveira[1] Sven Gregorio[1] João Dinis Ferreira[1]
Nika Mansouri Ghiasi[1] Minesh Patel[1] Mohammed Alser[1] Saugata Ghose[3]
Juan Gómez-Luna[1] Onur Mutlu[1]

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

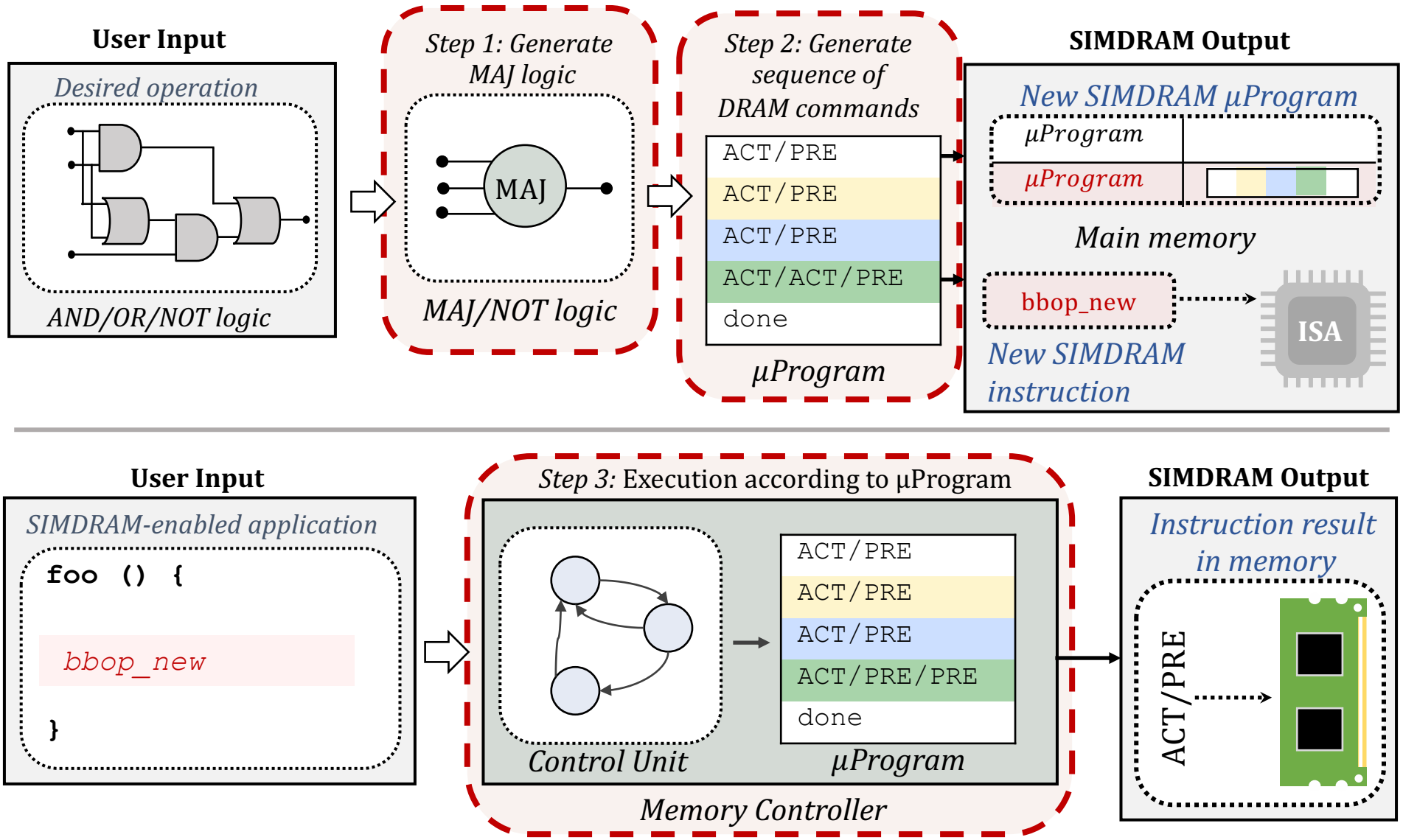
Executive Summary

- **Motivation:** Processing-using-Memory (PuM) architectures can efficiently perform bulk bitwise computation
- **Problem**: Existing PuM architectures are not widely applicable
  - Support only a limited and specific set of operations
  - Lack the flexibility to support new operations
  - Require significant changes to the DRAM subarray
- **Goals**: Design a processing-using-DRAM framework that:
  - Efficiently implements complex operations
  - Provides the flexibility to support new desired operations
  - Minimally changes the DRAM architecture
- **SIMDRAM**: An end-to-end processing-using-DRAM framework that provides the programming interface, the ISA, and the hardware support for:
  1. Efficiently computing complex operations
  2. Providing the ability to implement arbitrary operations as required
  3. Using a massively-parallel in-DRAM SIMD substrate that requires minimal changes to DRAM
- **Key Results**: SIMDRAM provides:
  - 88x and 5.8x the throughput and 257x and 31x the energy efficiency of a baseline CPU and a high-end GPU, respectively, for 16 in-DRAM operations
  - 21x and 2.1x the performance of the CPU and GPU for seven real-world applications

SIMDRAM Key Idea

- **SIMDRAM:** An end-to-end processing-using-DRAM framework that provides the programming interface, the ISA, and the hardware support for:
  - **Efficiently** computing **complex** operations in DRAM
  - Providing the ability to implement **arbitrary** operations as required
  - Using an **in-DRAM massively-parallel SIMD substrate** that requires **minimal** changes to DRAM architecture

SIMDRAM Framework: Overview



SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

Nastaran Hajinazar* Geraldo F. Oliveira*

Sven Gregorio Joao Ferreira Nika Mansouri Ghiasi

Minesh Patel Mohammed Alser Saugata Ghose

Juan Gómez–Luna Onur Mutlu

SAFARI

ETH zürich

SFU SIMON FRASER UNIVERSITY

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Processing-using-Memory (PuM)

- PuM: Exploits analog operation principles of the memory circuitry to perform computation
  - Leverages the large internal bandwidth and parallelism available inside the memory arrays
- A common approach for PuM architectures is to perform bulk bitwise operations
  - Simple logical operations (e.g., AND, OR, XOR)
  - More complex operations (e.g., addition, multiplication)

Motivation, Goal, and Key Idea

- Existing PuM mechanisms are not widely applicable
  - Support only a limited and mainly basic set of operations
  - Lack the flexibility to support new operations
  - Require significant changes to the DRAM subarray

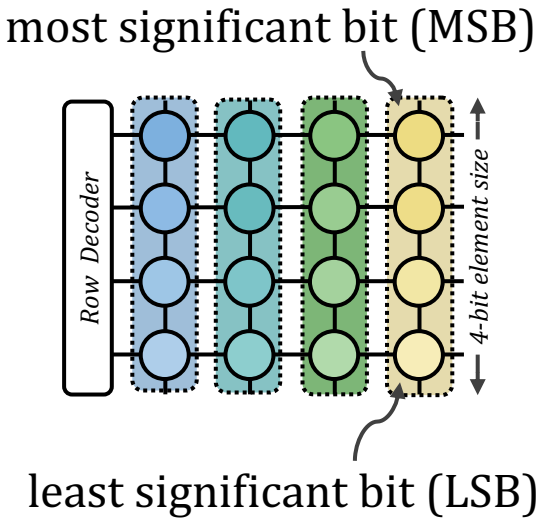
- **Goal:** Design a PuM framework that
  - Efficiently implements complex operations
  - Provides the flexibility to support new desired operations
  - Minimally changes the DRAM architecture

- **SIMDRAM:** An end-to-end processing-using-DRAM framework that provides the programming interface, the ISA, and the hardware support for:
  - Efficiently computing complex operations in DRAM
  - Providing the ability to implement arbitrary operations as required
  - Using an in-DRAM massively-parallel SIMD substrate that requires minimal changes to DRAM architecture

SIMDRAM: PuM Substrate

- SIMDRAM framework is built around a DRAM substrate that enables two techniques:

(1) Vertical data layout

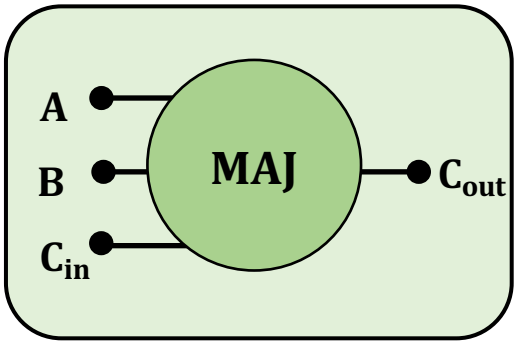


Pros compared to the conventional horizontal layout:

- Implicit shift operation
- Massive parallelism

(2) Majority-based computation

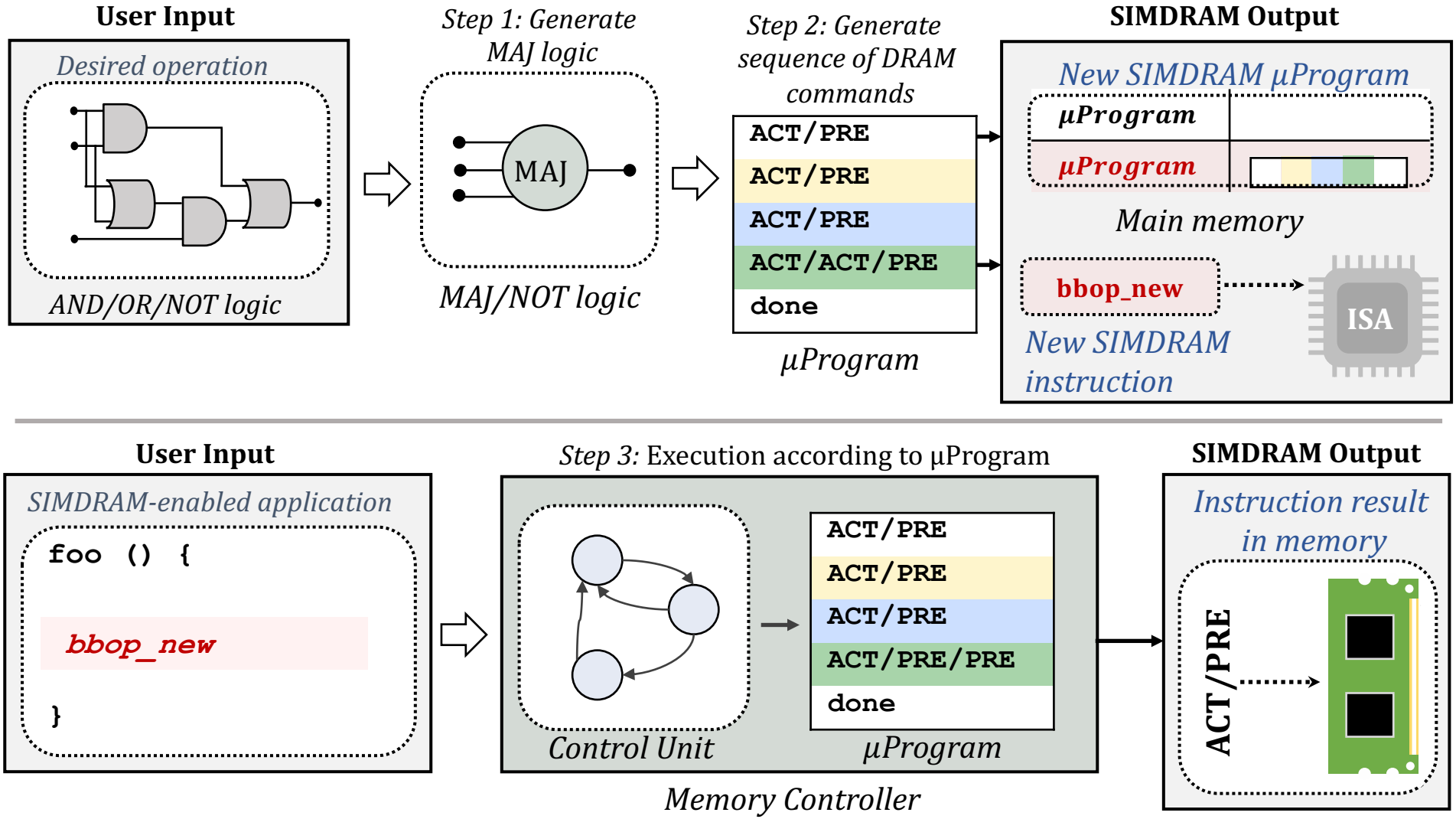
$C_{out} = AB + AC_{in} + BC_{in}$



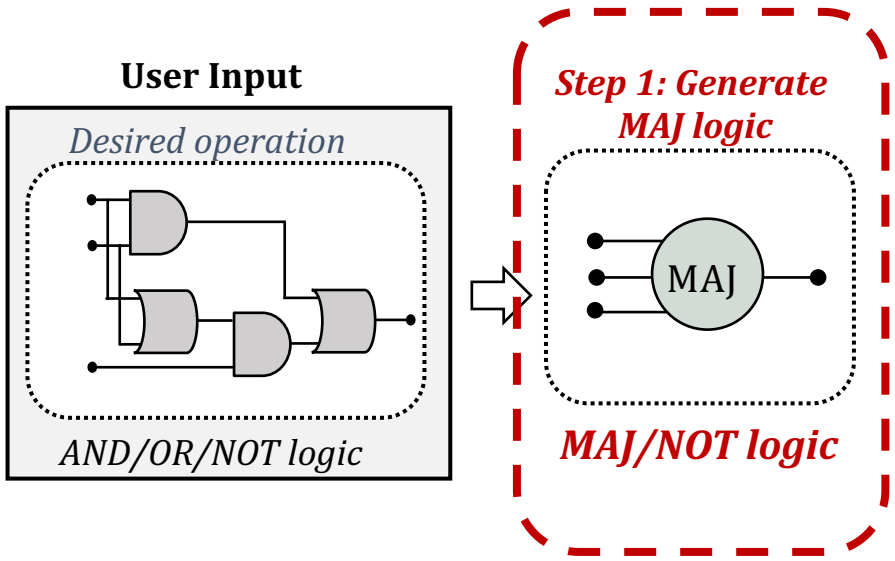
Pros compared to AND/OR/NOT-based computation:

- Higher performance
- Higher throughput
- Lower energy consumption

SIMDRAM Framework: Overview



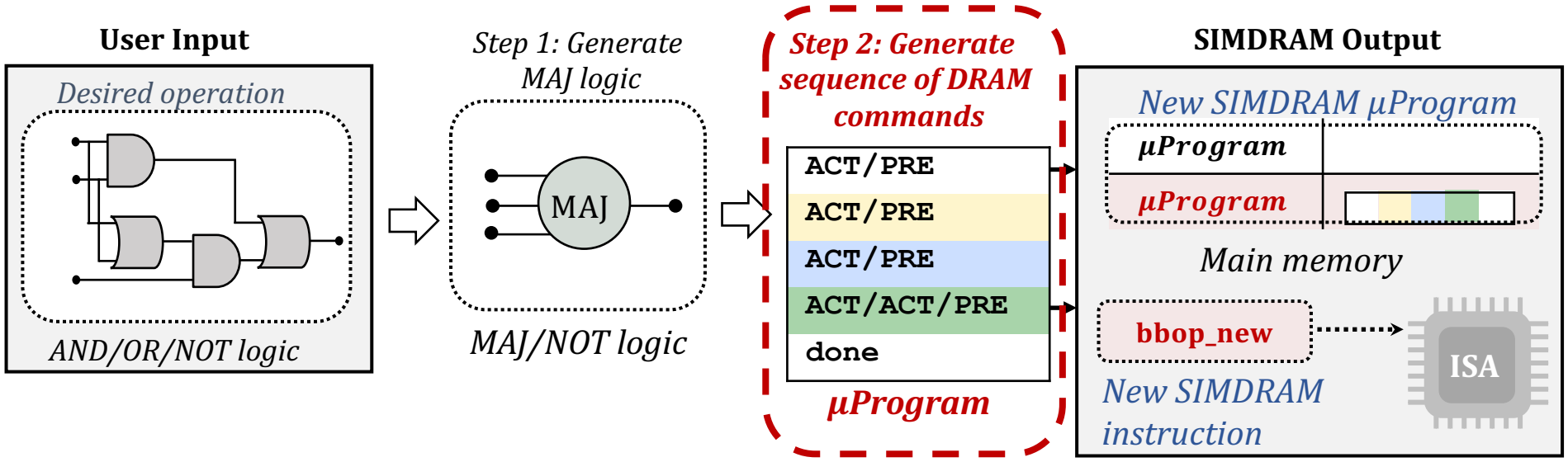
SIMDRAM Framework: Overview



Step 1:

- Builds an **efficient MAJ/NOT representation** of a given desired operation from its AND/OR/NOT-based implementation

SIMDRAM Framework: Overview



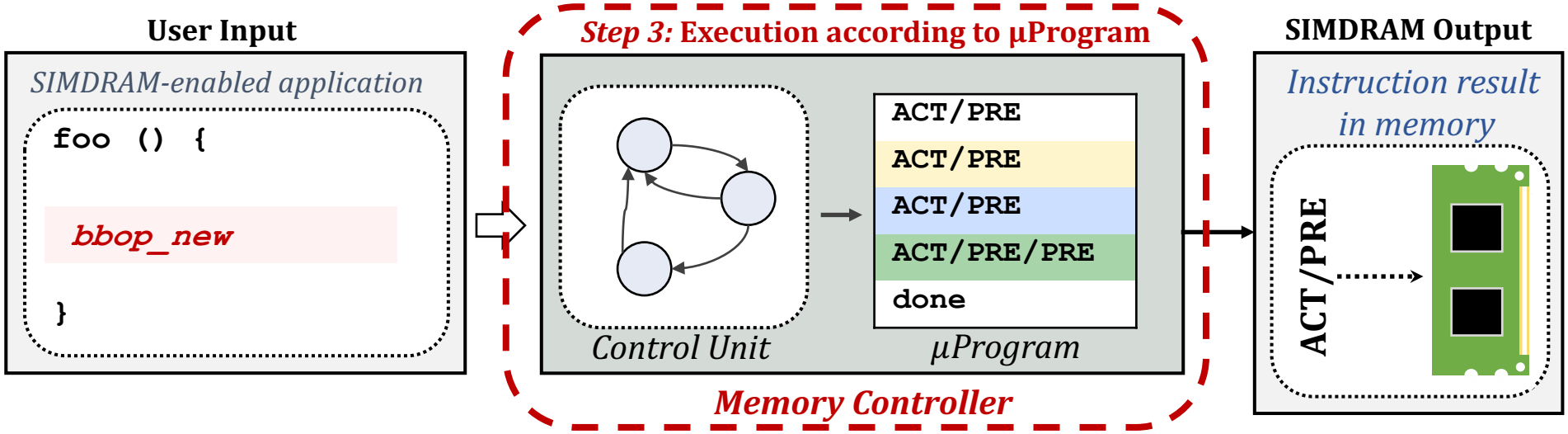
Step 2:

- **Allocates DRAM** rows to the operation's inputs and outputs
- Generates the **sequence of DRAM commands** (*μProgram*) to execute the desired operation

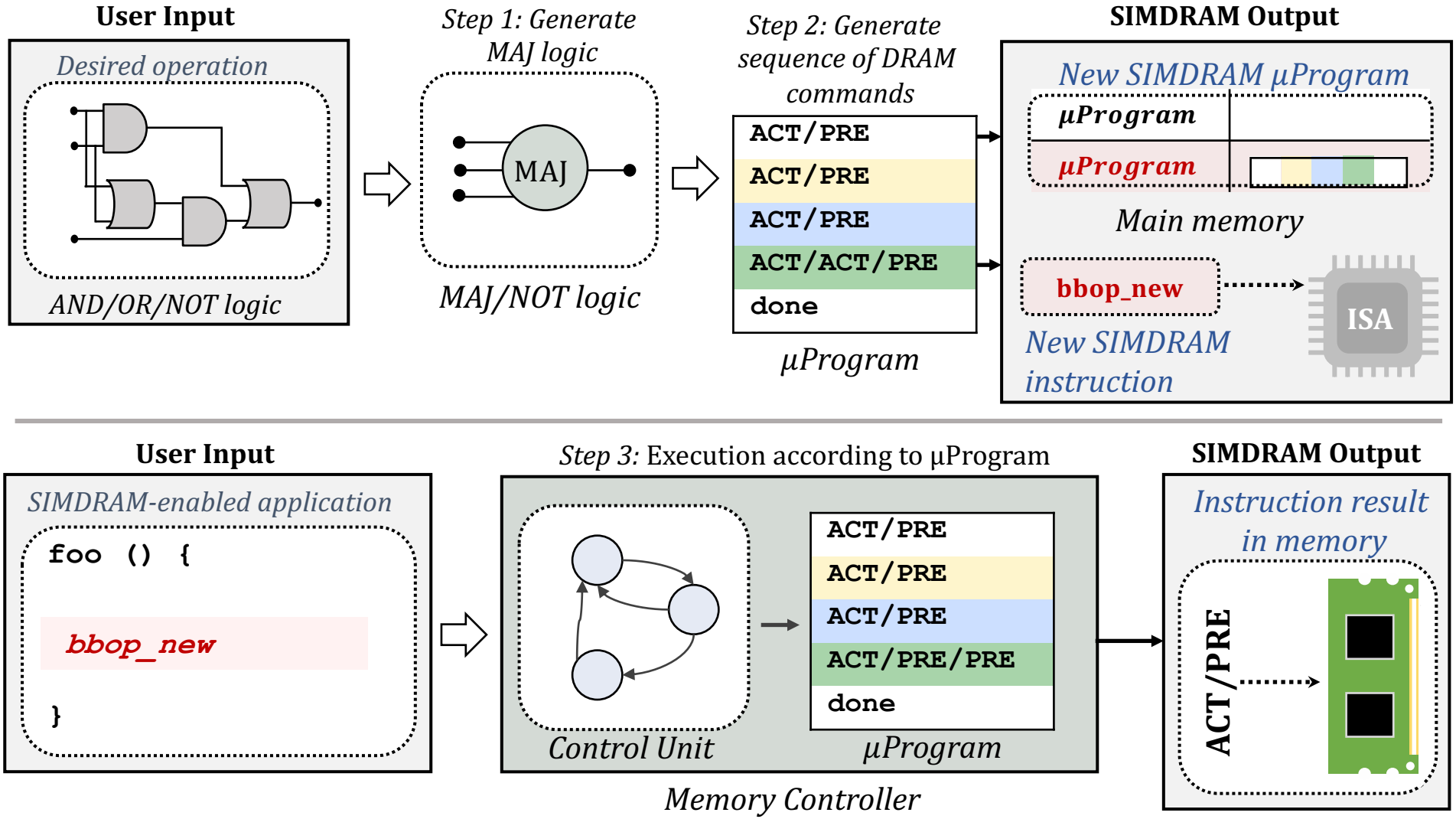
SIMDRAM Framework: Overview

Step 3:

- **Executes the μProgram** to perform the operation
- Uses a **control unit** in the memory controller



SIMDRAM Framework: Overview



Key Results

Evaluated on:

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

SIMDRAM provides:

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

Conclusion

- **SIMDRAM:**
  - Enables efficient computation of a flexible set and wide range of operations in a PuM massively parallel SIMD substrate
  - Provides the hardware, programming, and ISA support, to:
    - Address key system integration challenges
    - Allow programmers to define and employ new operations without hardware changes

SIMDRAM is a promising PuM framework

- Can ease the adoption of processing-using-DRAM architectures
- Improve the performance and efficiency of processing-using-DRAM architectures

SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

Nastaran Hajinazar* Geraldo F. Oliveira*

Sven Gregorio Joao Ferreira Nika Mansouri Ghiasi

Minesh Patel Mohammed Alser Saugata Ghose

Juan Gómez–Luna Onur Mutlu

SAFARI

ETH zürich

SFU SIMON FRASER UNIVERSITY

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

More on the SIMDRAM Framework

- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu, **"SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"** *Proceedings of the 26th International Conference on Architectural Support for Programming Languages and Operating Systems* (***ASPLOS***), Virtual, March-April 2021.
  [2-page Extended Abstract]
  [Short Talk Slides (pptx) (pdf)]
  [Talk Slides (pptx) (pdf)]
  [Short Talk Video (5 mins)]
  [Full Talk Video (27 mins)]

SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

*Nastaran Hajinazar[1,2] *Geraldo F. Oliveira[1] Sven Gregorio[1] João Dinis Ferreira[1]
Nika Mansouri Ghiasi[1] Minesh Patel[1] Mohammed Alser[1] Saugata Ghose[3]
Juan Gómez-Luna[1] Onur Mutlu[1]

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

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]

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

Geraldo F. Oliveira

Juan Gómez-Luna Lois Orosa Saugata Ghose

Nandita Vijaykumar Ivan Fernandez Mohammad Sadrosadati

Onur Mutlu

SAFARI

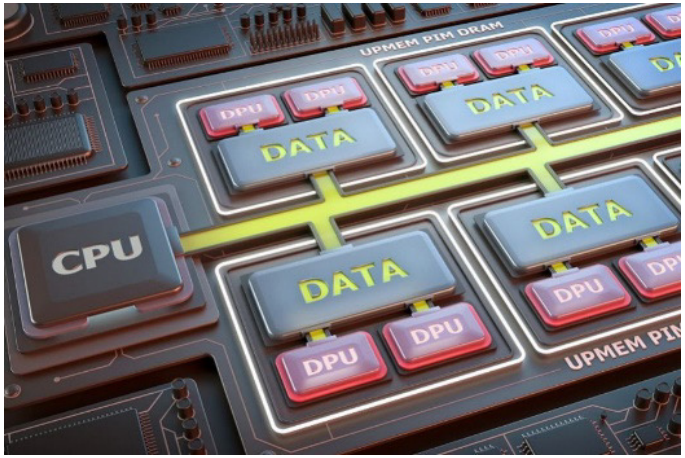
ETH Zürich UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN UNIVERSITY OF TORONTO UNIVERSITAS MALACITANA UNIVERSIDAD DE MÁLAGA

Executive Summary

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

Near-Data Processing (2/2)

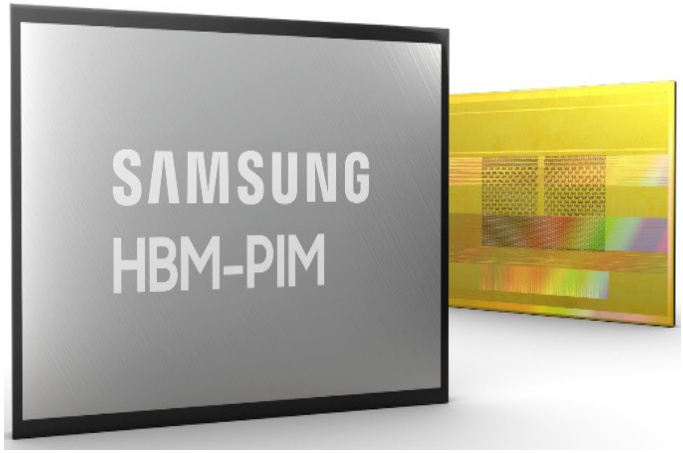
UPMEM (2019)



Near-DRAM-banks processing
for general-purpose computing

0.9 TOPS compute throughput[1]

Samsung FIMDRAM (2021)



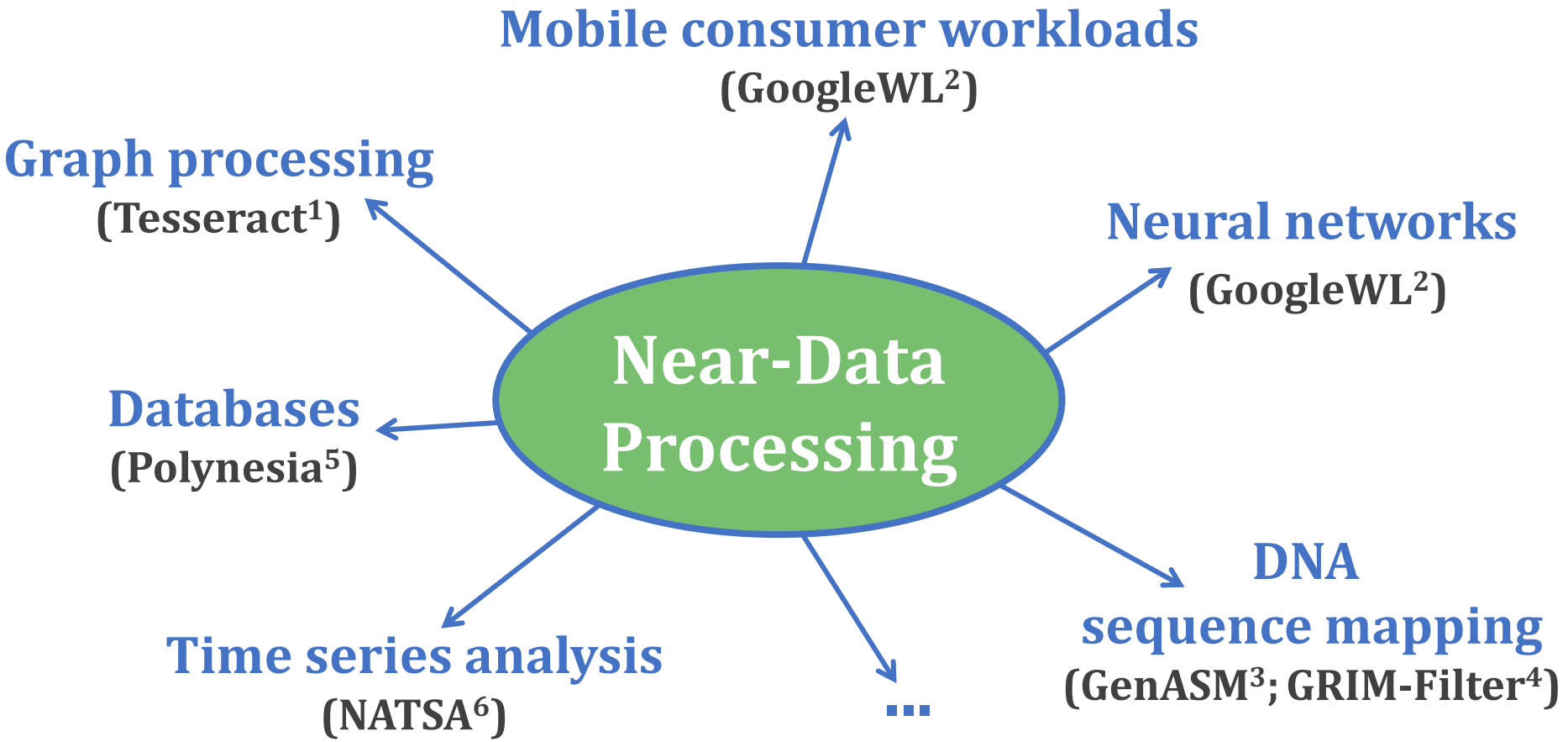
Near-DRAM-banks processing
for neural networks

1.2 TFLOPS compute throughput[2]

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

[1] Devaux, "The True Processing In Memory Accelerator," HCS, 2019
[2] Kwon+, "A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications," ISSCC, 2021

When to Employ Near-Data Processing?



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

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

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

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

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

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

Key Approach

- New workload characterization methodology to analyze:
  - data movement bottlenecks
  - suitability of different data movement mitigation mechanisms

- Two main profiling strategies:

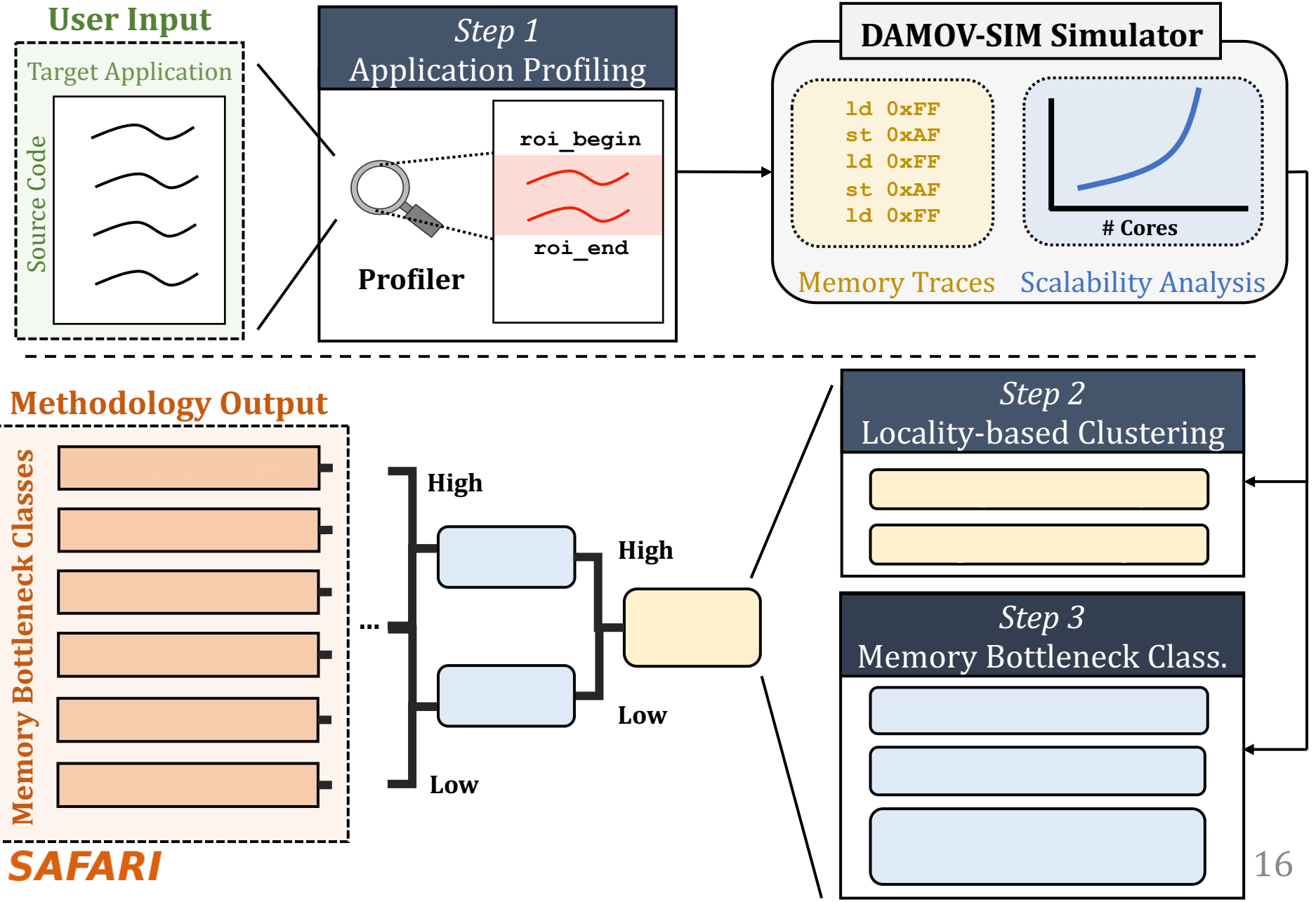
Architecture-independent profiling:

characterizes the memory behavior independently of the underlying hardware

Architecture-dependent profiling:

evaluates the impact of the system configuration on the memory behavior

Methodology Overview

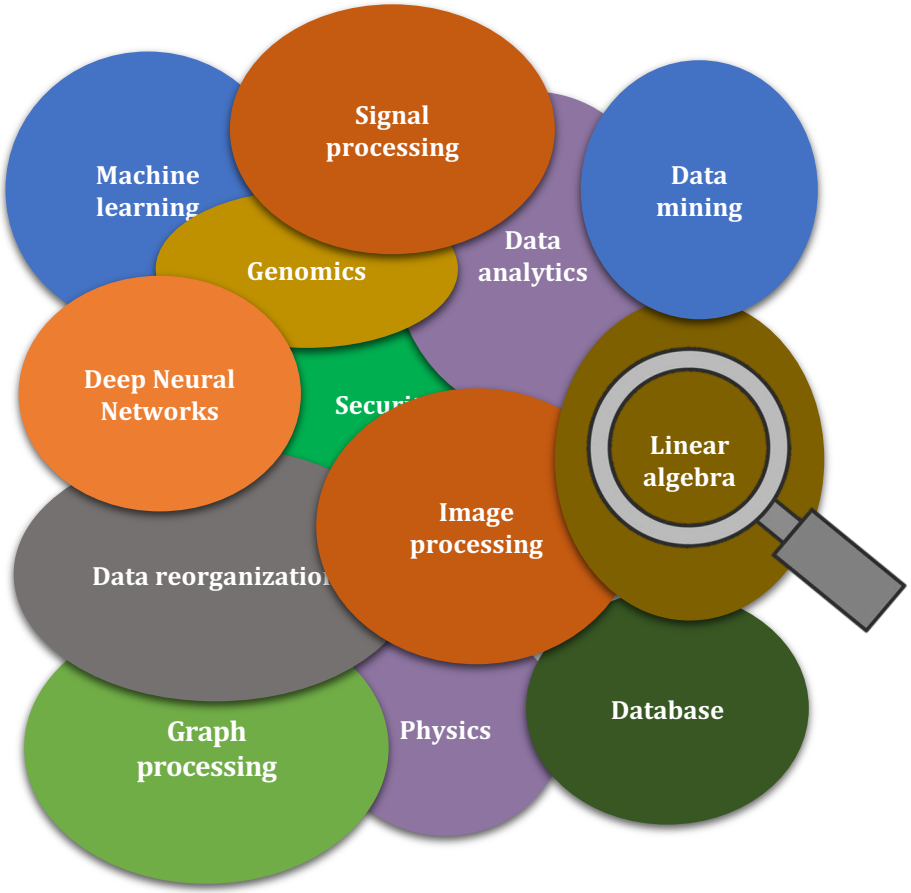


Step 1: Application Profiling

- We analyze 345 applications from distinct domains:

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

...



Step 3: Memory Bottleneck Analysis

Six classes of data movement bottlenecks:

each class ↔ data movement mitigation mechanism

Memory Bottleneck Class

- **1a:** ***DRAM Bandwidth***
- **1b:** ***DRAM Latency***
- **1c: L1/L2** ***Cache Capacity***
- **2a: L3** ***Cache Contention***
- **2b:** ***L1 Cache Capacity***
- **2c:** ***Compute-Bound***

DAMOV is Open-Source

- We open-source our benchmark suite and our toolchain

DAMOV-SIM

DAMOV Benchmarks

CMU-SAFARI / **DAMOV**

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

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

omutlu Update README.md — ce1b4ea 17 days ago — 5 commits

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

README.md

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

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

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

About

DAMOV is a benchmark suite and a methodical framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processing. Described by Oliveira et al. (preliminary version at https://arxiv.org/pdf/2105.03725.pdf)

Readme

Releases

No releases published
Create a new release

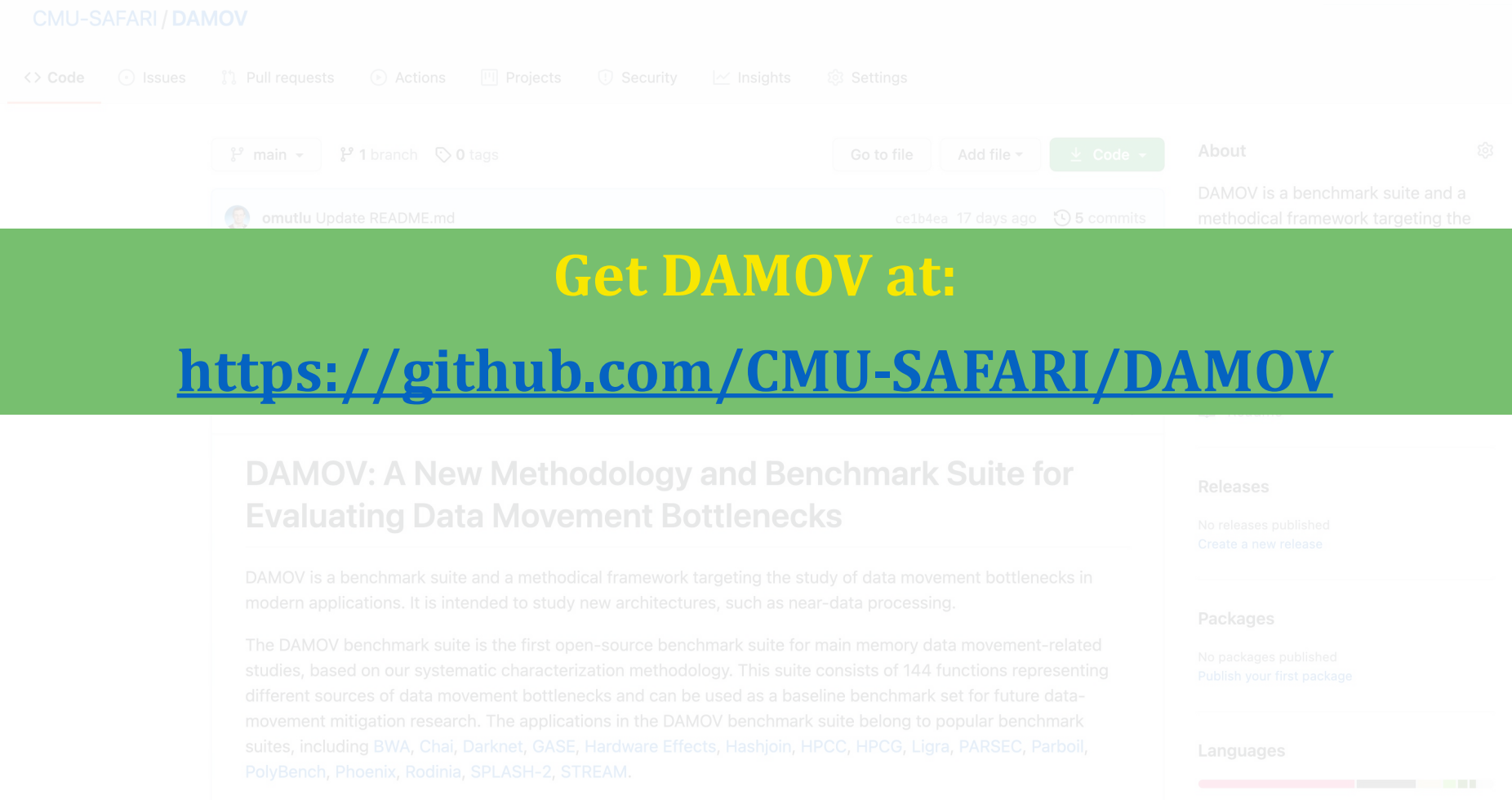
Packages

No packages published
Publish your first package

Languages

DAMOV is Open-Source

- We open-source our benchmark suite and our toolchain



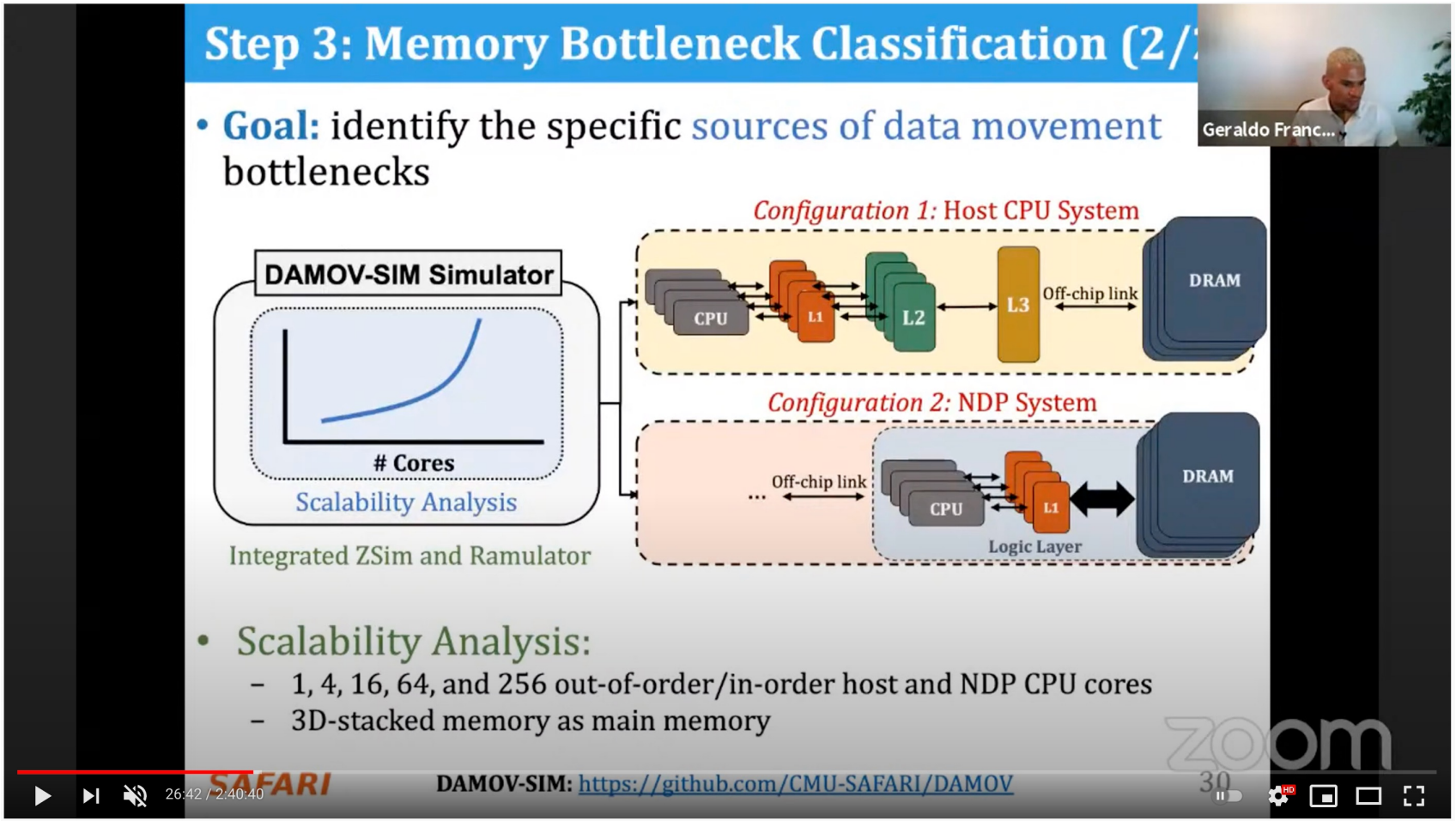
Get DAMOV at:

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

More on DAMOV

- Geraldo F. Oliveira, Juan Gomez-Luna, Lois Orosa, Saugata Ghose, Nandita Vijaykumar, Ivan fernandez, Mohammad Sadrosadati, and Onur Mutlu,
  "DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks"
  *Preprint in **arXiv***, 8 May 2021.
  [arXiv preprint]
  [DAMOV Suite and Simulator Source Code]

More on DAMOV Analysis Methodology & Workloads



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

352 views • Streamed live on Jul 22, 2021

18 0 SHARE SAVE ...

Onur Mutlu Lectures
17.7K subscribers

ANALYTICS EDIT VIDEO

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

DAMOV Analysis Methodology & Workloads

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

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

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

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

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

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]

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]

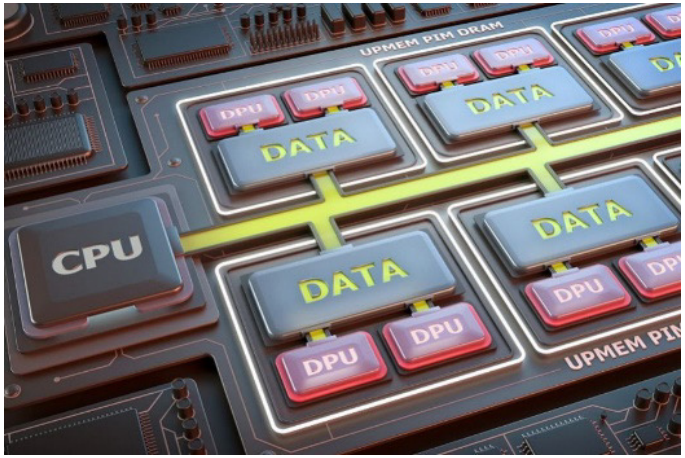
https://arxiv.org/pdf/1907.12947.pdf

Year 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]

Near-Data Processing

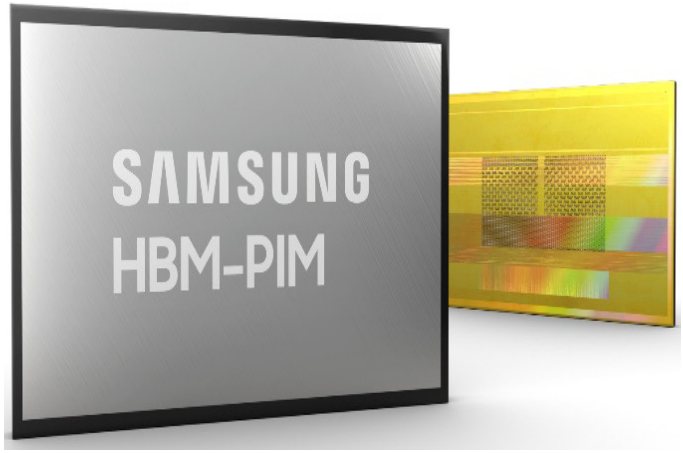
UPMEM (2019)



Near-DRAM-banks processing
for general-purpose computing

0.9 TOPS compute throughput[1]

Samsung FIMDRAM (2021)



Near-DRAM-banks processing
for neural networks

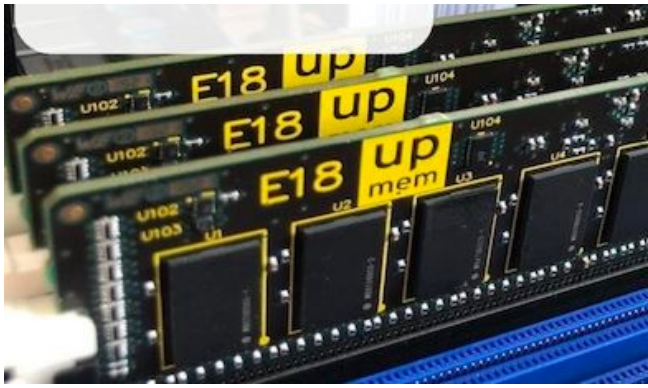
1.2 TFLOPS compute throughput[2]

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

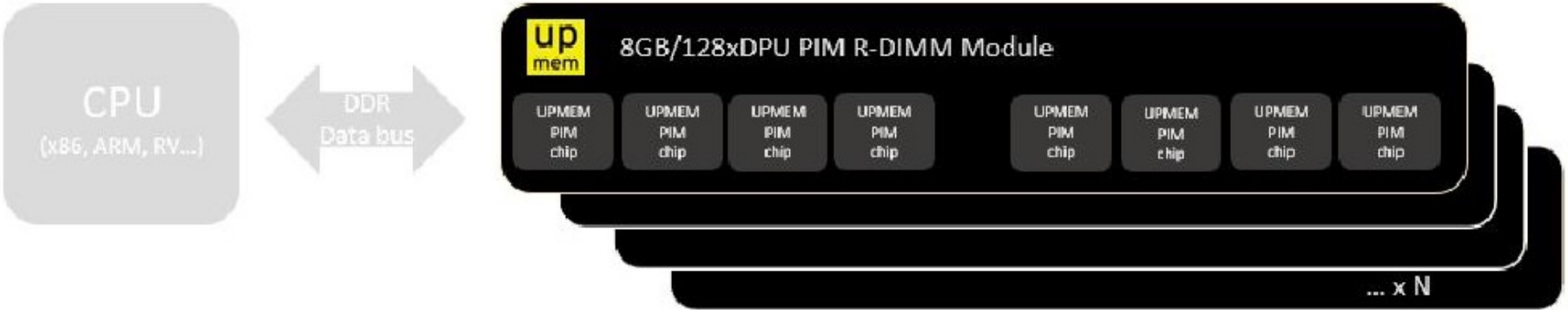
[1] Devaux, "The True Processing In Memory Accelerator," HCS, 2019
[2] Kwon+, "A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications," ISSCC, 2021

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.



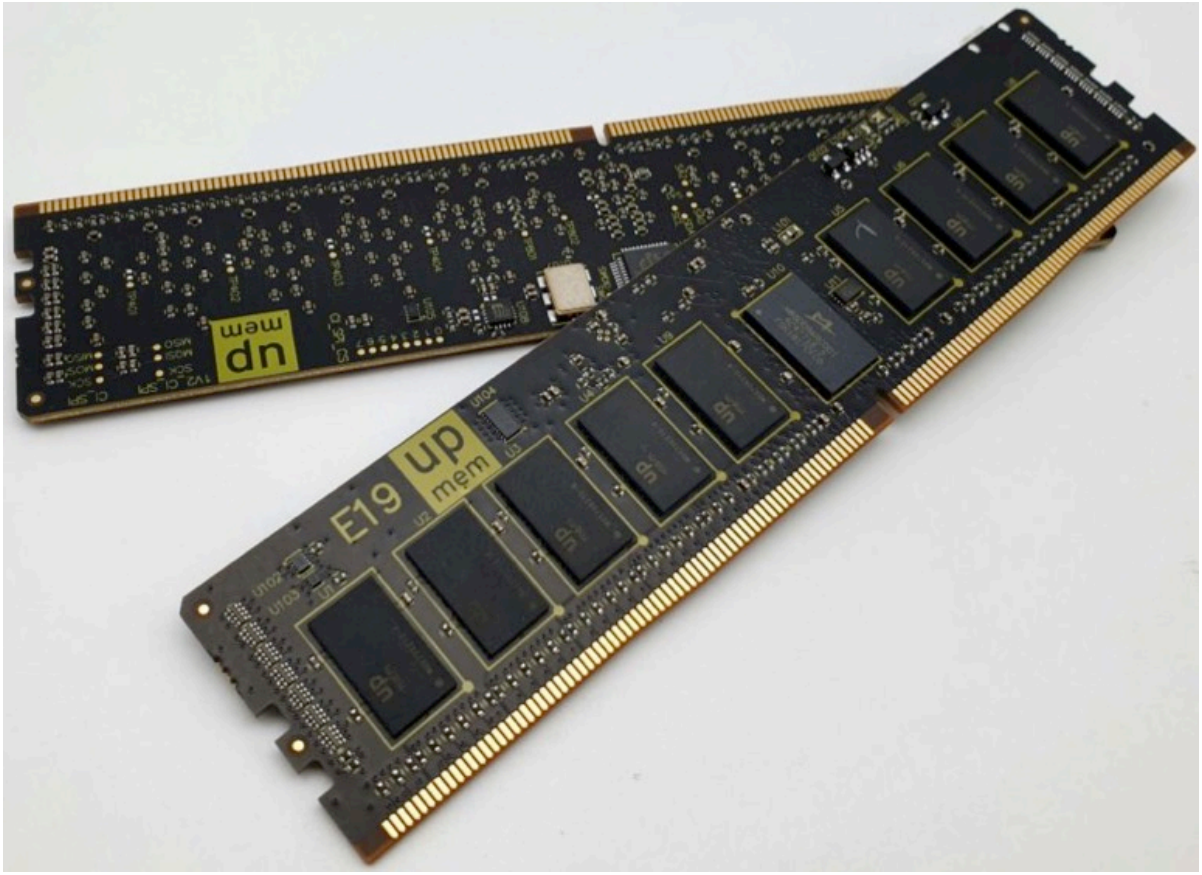
- Replaces **standard** DIMMs
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth



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/

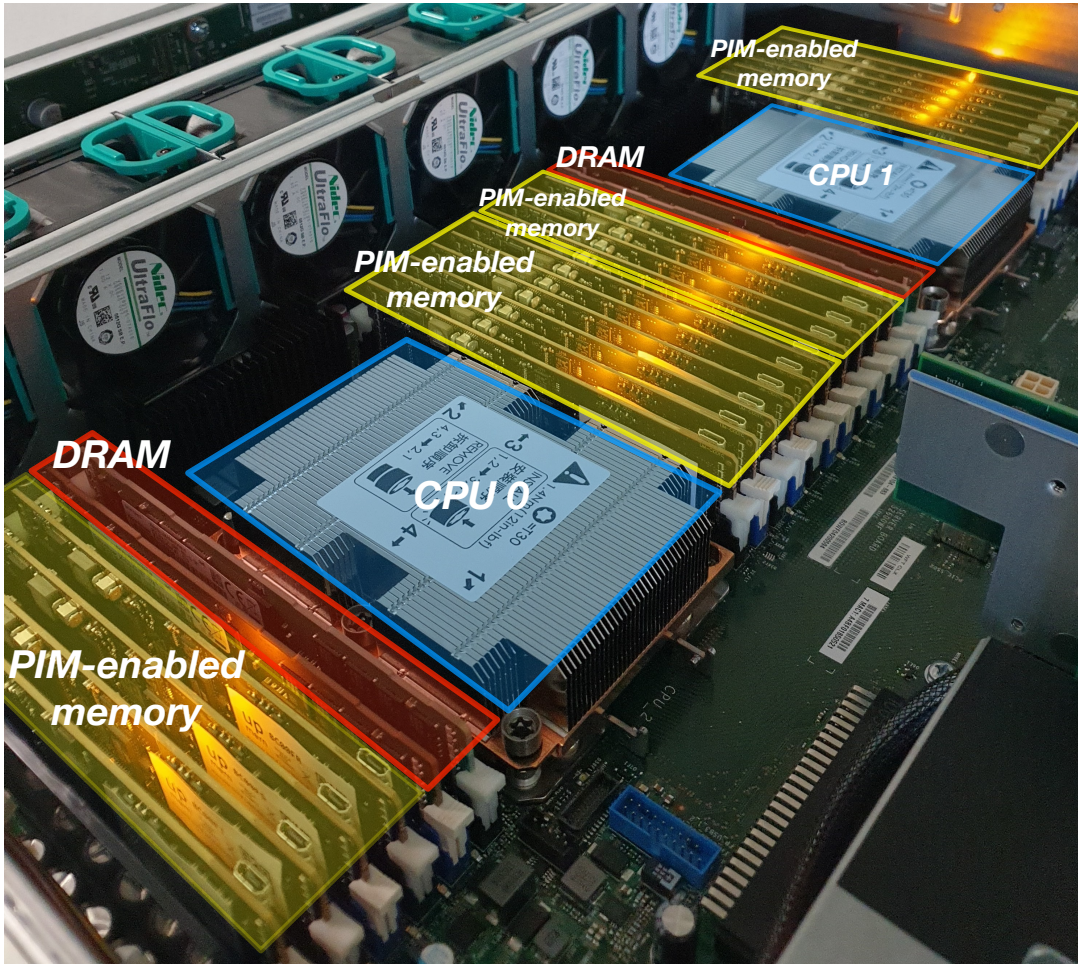
UPMEM Memory Modules

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

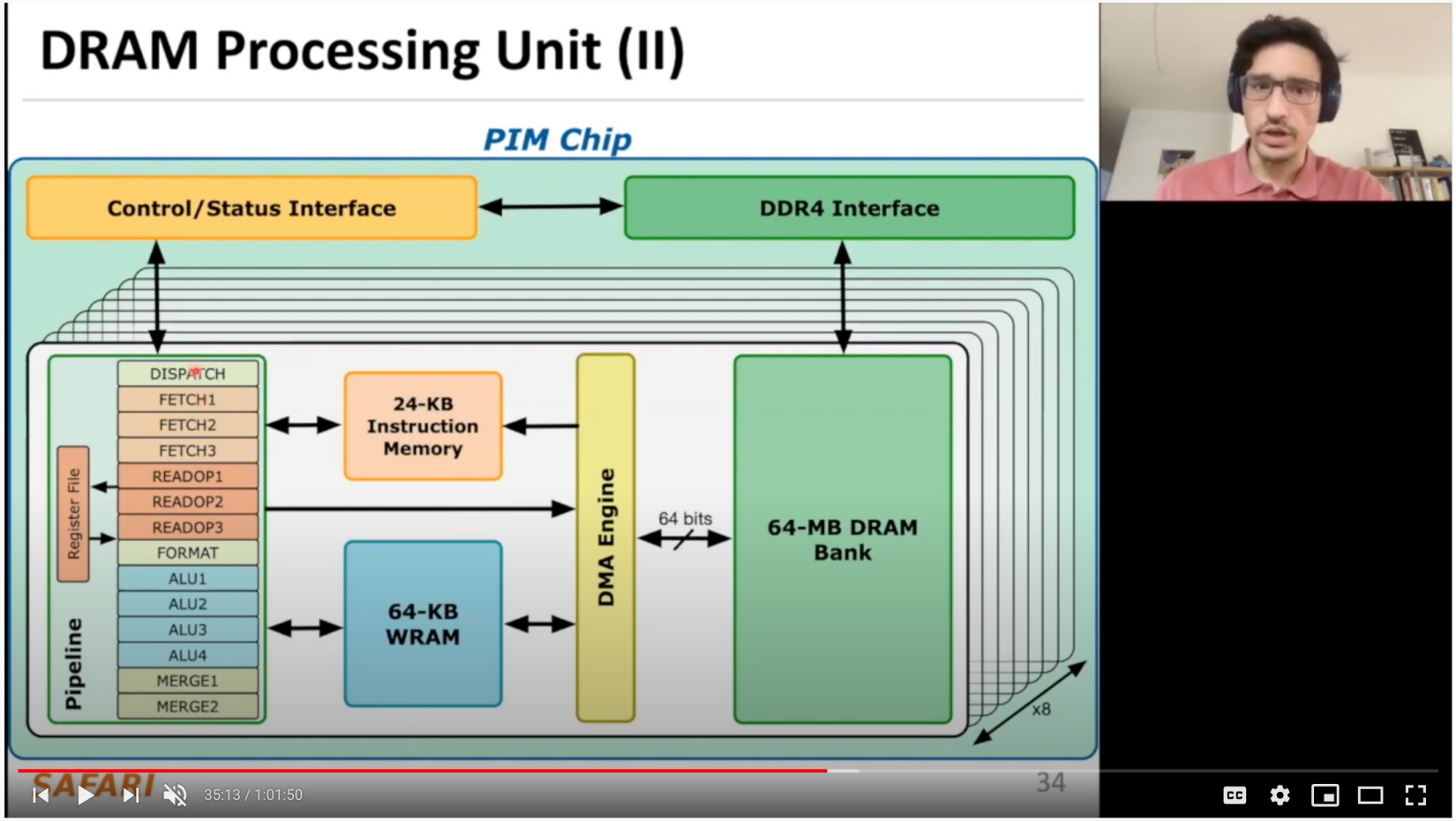


PIM System Organization

- UPMEM-based PIM system with 20 UPMEM memory modules of 16 chips each (40 ranks) → 2560 DPUs



More on the UPMEM PIM System



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

ETH Zürich
SAFARI

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

Understanding a Modern PIM Architecture

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

Juan Gómez-Luna[1] Izzat El Hajj[2] Ivan Fernandez[1,3] Christina Giannoula[1,4]
Geraldo F. Oliveira[1] Onur Mutlu[1]

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

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

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

Observations, Recommendations, Takeaways

GENERAL PROGRAMMING RECOMMENDATIONS

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

PROGRAMMING RECOMMENDATION 1

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

KEY OBSERVATION 7

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

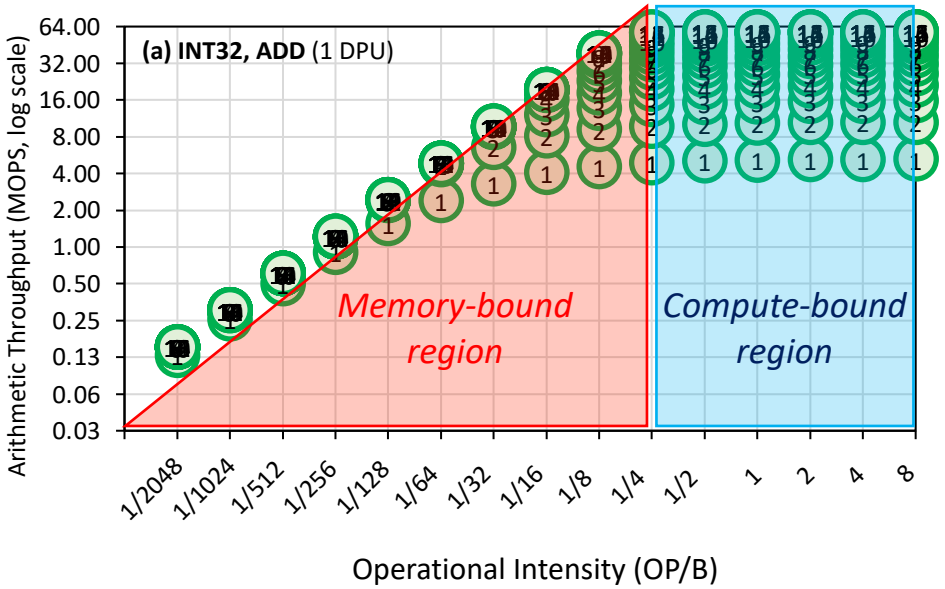
KEY TAKEAWAY 1

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

Outline

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

Key Takeaway 1

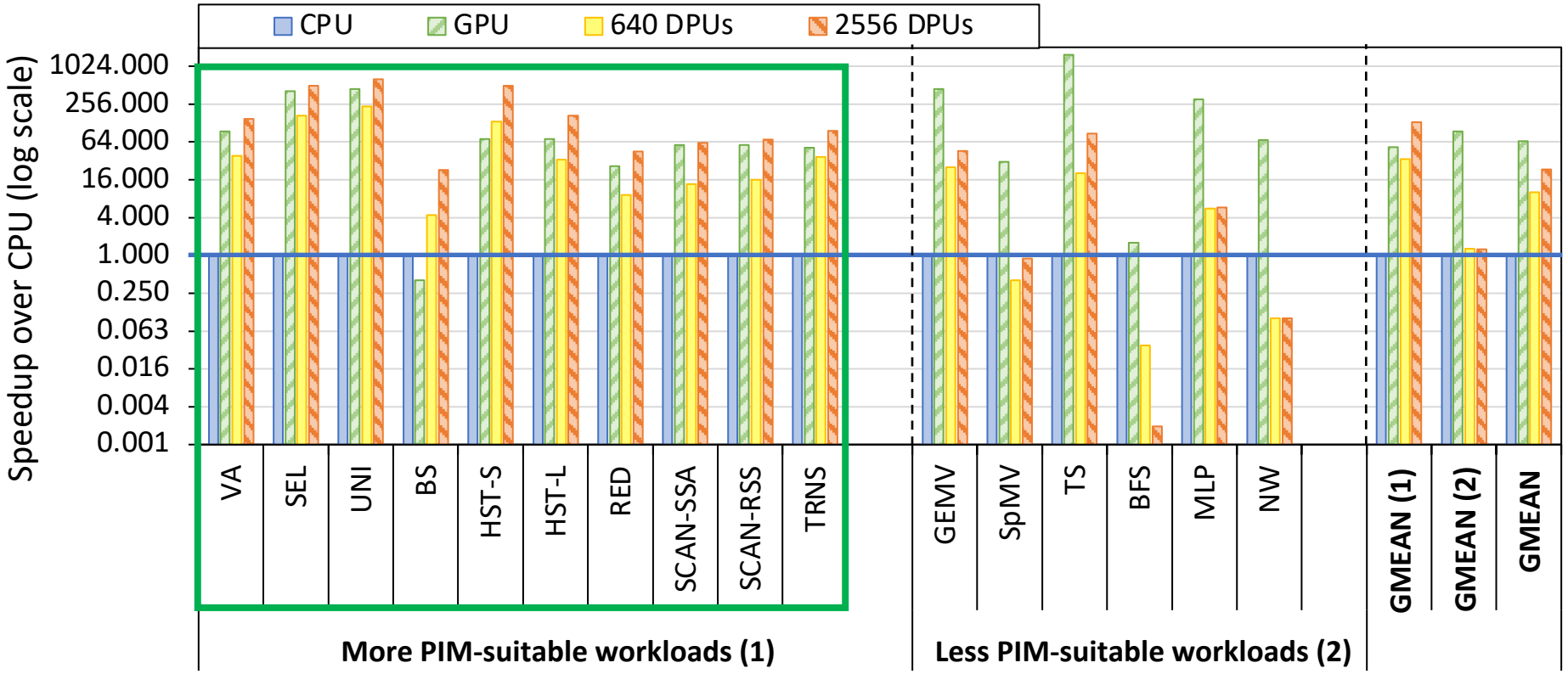


The throughput saturation point is as low as ¼ OP/B, i.e., 1 integer addition per every 32-bit element fetched

KEY TAKEAWAY 1

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

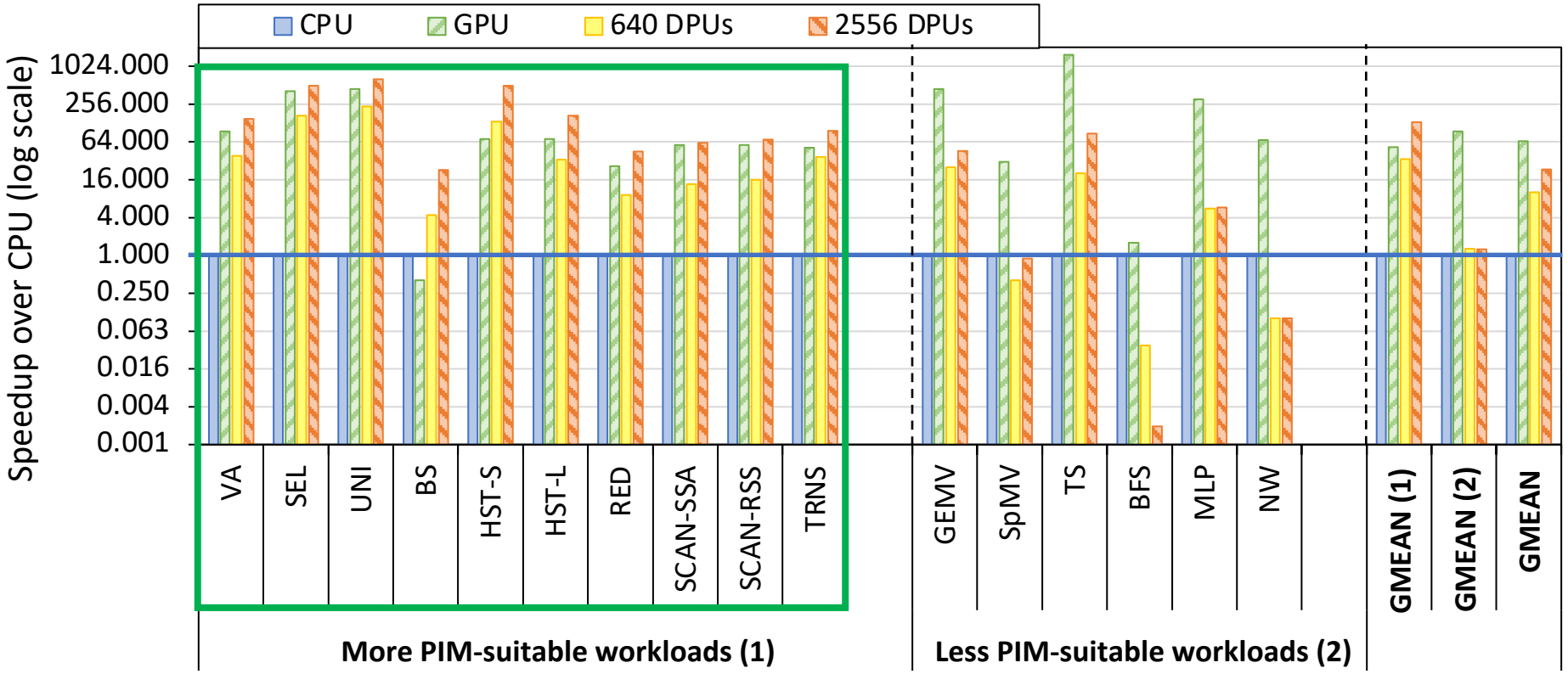
Key Takeaway 2



KEY TAKEAWAY 2

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

Key Takeaway 3



KEY TAKEAWAY 3

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

Key Takeaway 4

KEY TAKEAWAY 4

- UPMEM-based PIM systems **outperform state-of-the-art CPUs in terms of performance and energy efficiency on most of PrIM benchmarks.**

- UPMEM-based PIM systems **outperform state-of-the-art GPUs on a majority of PrIM benchmarks**, and the outlook is even more positive for future PIM systems.

- UPMEM-based PIM systems are **more energy-efficient than state-of-the-art CPUs and GPUs on workloads that they provide performance improvements** over the CPUs and the GPUs.

Understanding a Modern PIM Architecture

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

Juan Gómez-Luna[1] Izzat El Hajj[2] Ivan Fernandez[1,3] Christina Giannoula[1,4]
Geraldo F. Oliveira[1] Onur Mutlu[1]

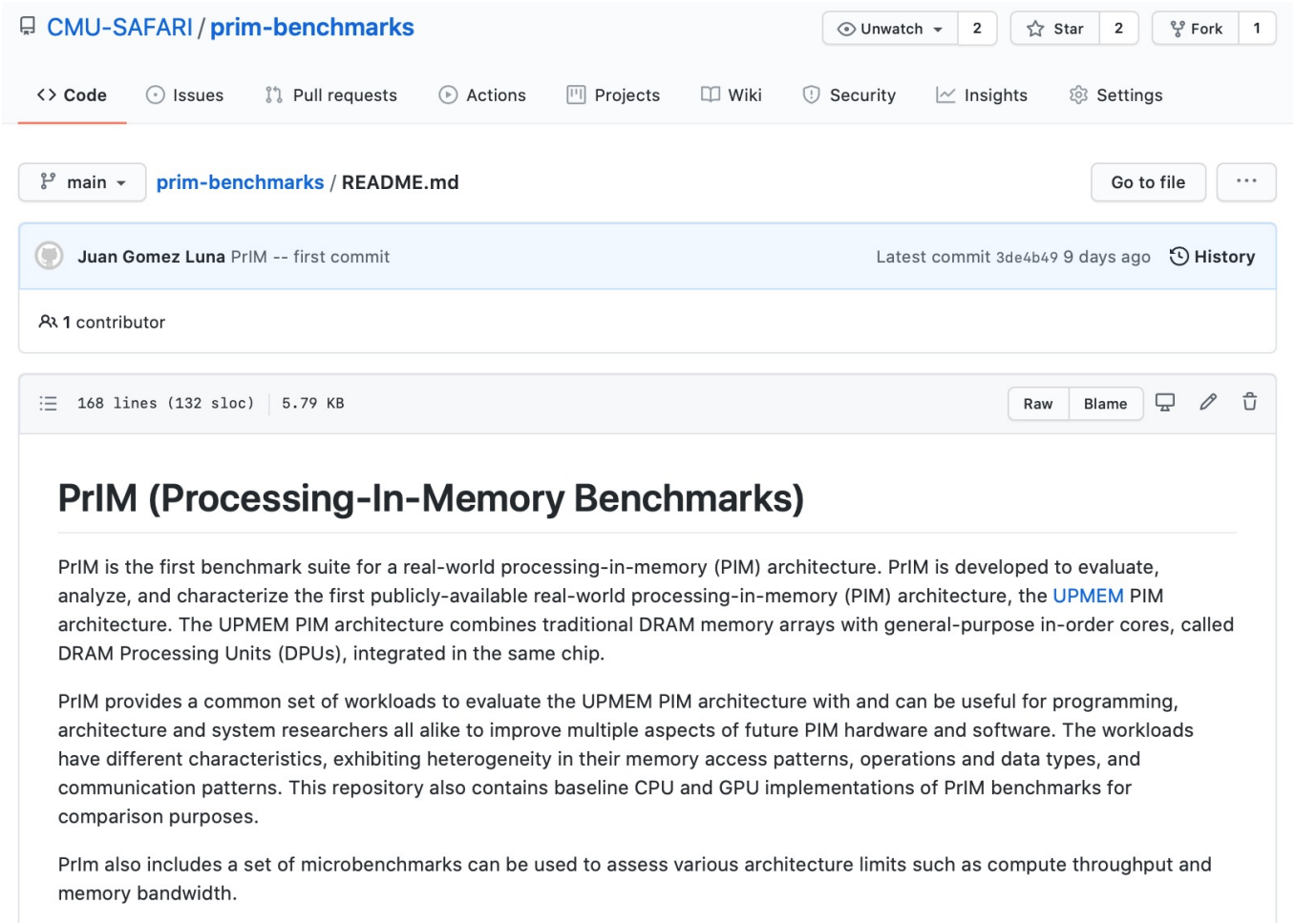
[1]ETH Zürich [2]American University of Beirut [3]University of Malaga [4]National Technical University of Athens

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

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

PrIM Repository

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



Understanding a Modern Processing-in-Memory Architecture:

Benchmarking and Experimental Characterization

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

el1goluj@gmail.com

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

ETH Zürich

SAFARI

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

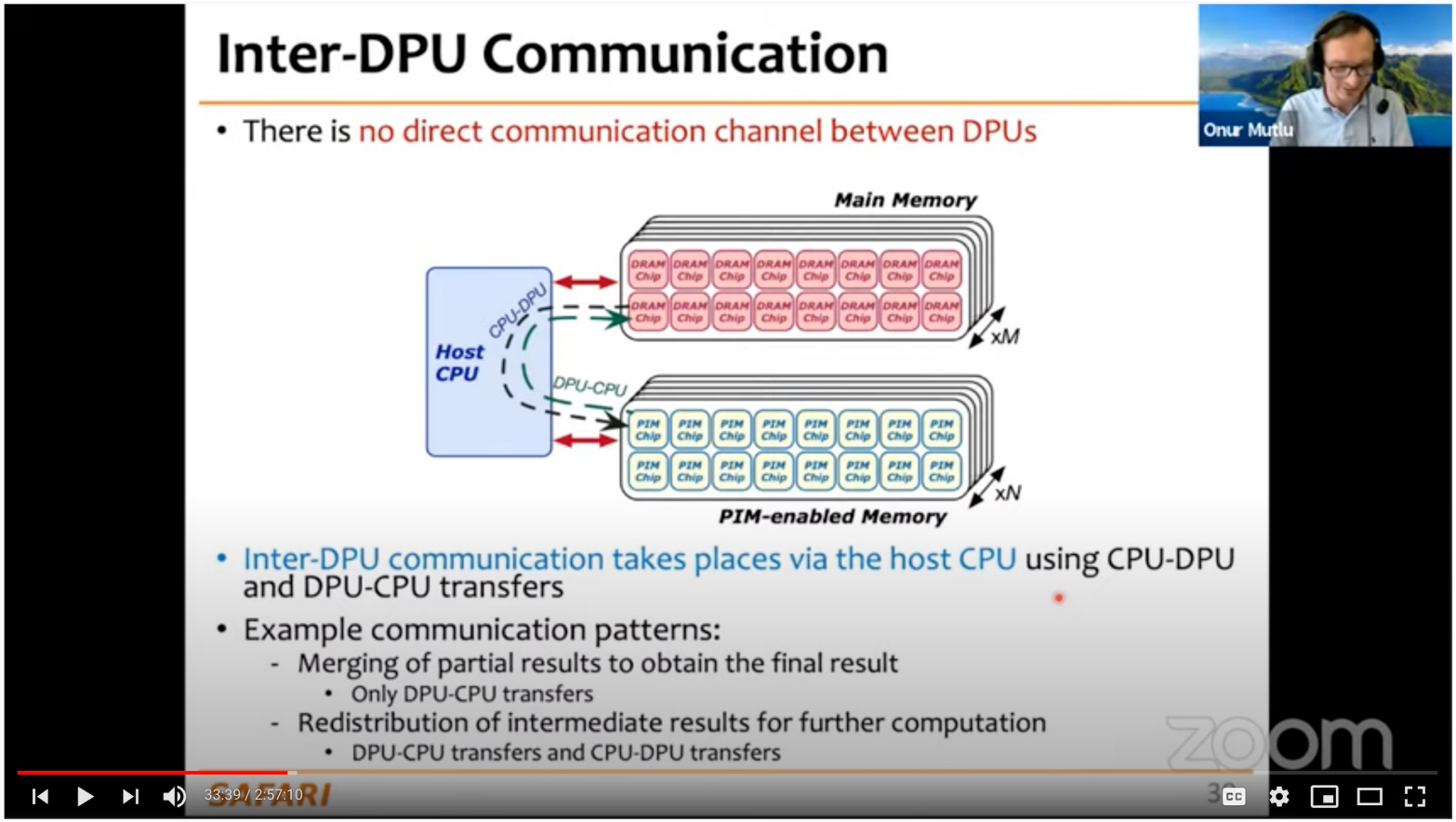
Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens 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

More on Analysis of the UPMEM PIM Engine



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

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

81 0 SHARE SAVE ...

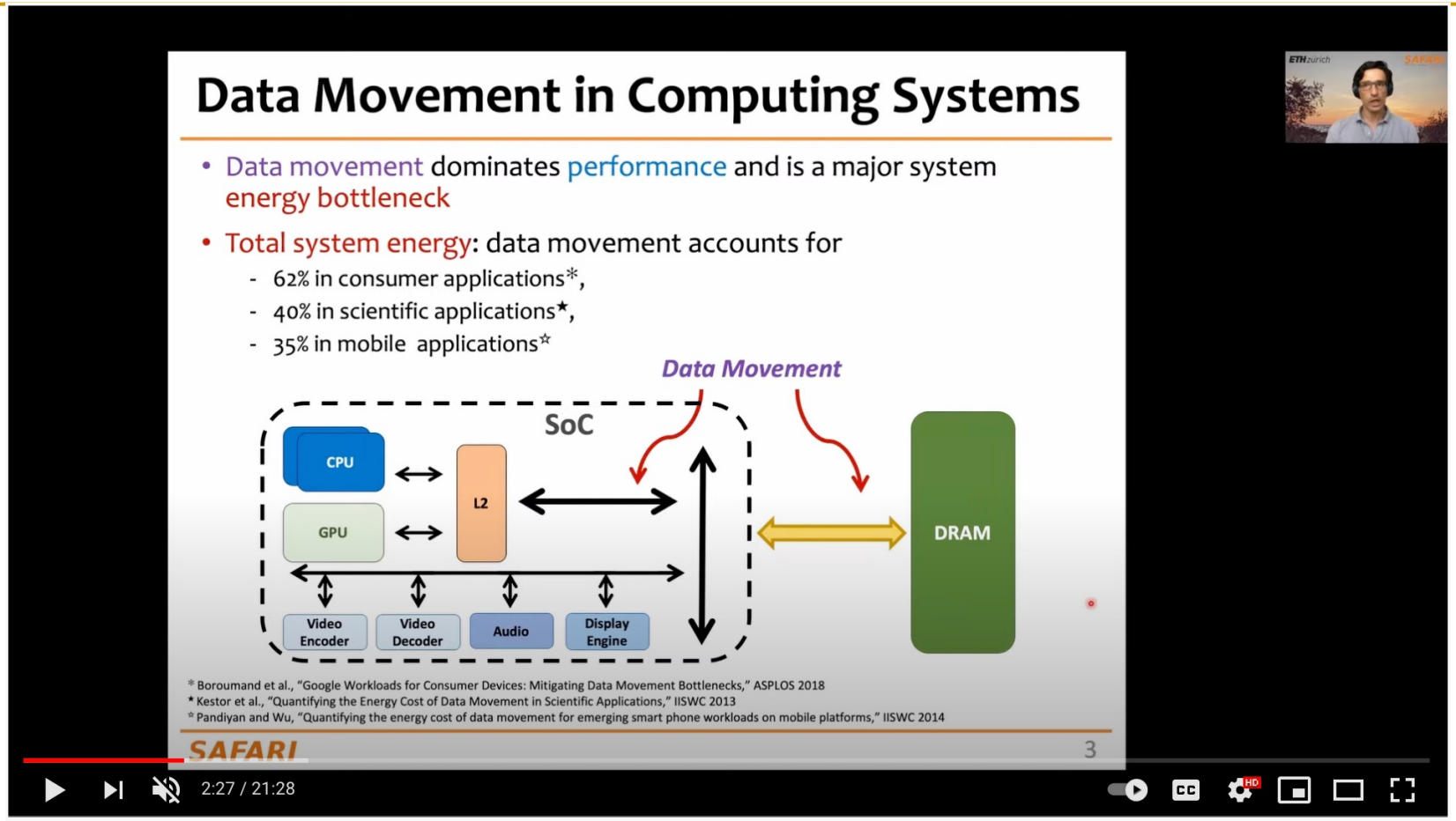
Onur Mutlu Lectures
17.6K subscribers

ANALYTICS EDIT VIDEO

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



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

3,482 views • Premiered Jul 25, 2021 38 0 SHARE SAVE

Onur Mutlu Lectures
17.9K subscribers

ANALYTICS EDIT VIDEO

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

More on PRIM Benchmarks

- Juan Gomez-Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu, **"Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture"**
  *Preprint in **arXiv***, 9 May 2021.
  [arXiv preprint]
  [PrIM Benchmarks Source Code]
  [Slides (pptx) (pdf)]
  [Long Talk Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]
  [SAFARI Live Seminar Slides (pptx) (pdf)]
  [SAFARI Live Seminar Video (2 hrs 57 mins)]
  [Lightning Talk Video (3 minutes)]

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]

FPGA-based Processing Near Memory

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

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

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

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

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

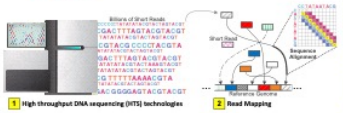
Gagandeep Singh[1] Mohammed Alser[1] Damla Senol Cali[4] Dionysios Diamantopoulos[2]
Juan Gomez-Luna[1] Henk Corporaal[3] Onur Mutlu[1]

[1] ETH zürich [2] IBM Research | Zurich [3] TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY [4] Carnegie Mellon

1: Modern Data-Intensive Applications

Genome Sequencing

- **High throughput DNA sequencing** (HTS) technologies can sequence only segments of the original molecule.
- We need ***read mapping*** to link the reads together and construct back the donor's complete genome by 1) determining the *location* of each read within the reference genome and 2) calculating its **optimal sequence alignment**.
- SneakySnake is the state-of-the-art pre-alignment filtering algorithm

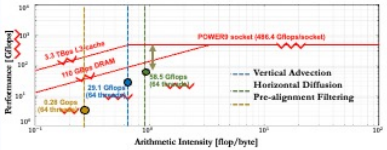


Weather Modeling

- **COSMO (Consortium for Small-Scale Modeling)** weather prediction application is used by a dozen nations and research institutes
- Compound stencil kernel consists of a **collection** of elementary stencil kernels
- Horizontal diffusion (hdiff) and vertical advection (vadvc) are representative stencils

2: Motivation and Goal

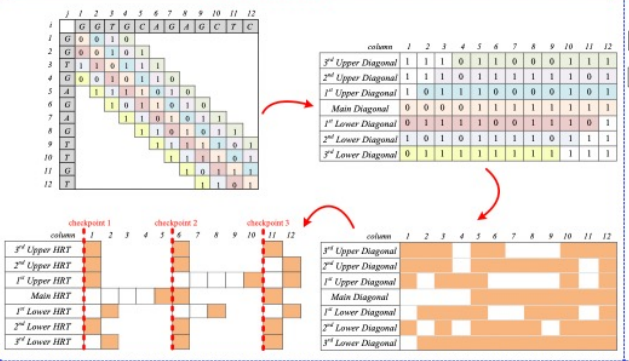
Modern data-intensive applications are **memory-bound** with **limited performance** and **high energy consumption** on **multi-core architectures**



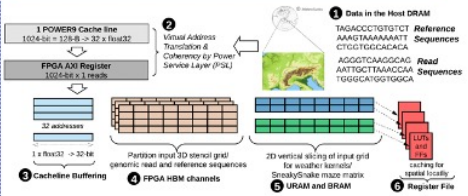
Mitigate the **performance bottleneck** of modern data-intensive applications in an **energy-efficient way**

Evaluate the use of **near-memory acceleration** using a **FPGA+HBM** connected through **IBM CAPI2** (Coherent Accelerator Processor Interface) and **OCAPI** (OpenCAPI)

3: SneakySnake Walkthrough



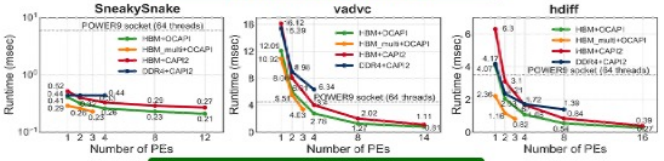
4: Data Mapping onto Heterogeneous Memories



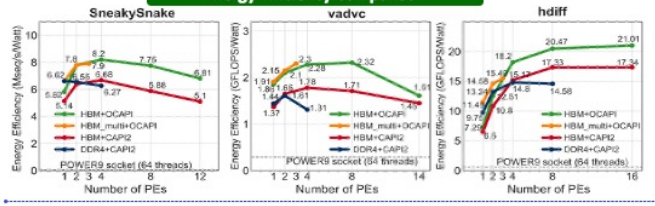
- Create a **specialized memory hierarchy** from the pool of **heterogeneous FPGA memories** (i.e., on-chip BRAM and URAM, and in-package HBM)
- We partition the data among HBM channels to **exploit data-level parallelism** and to scale our design
- Each processing engine (PE) fetches from an independent 256-bit channel, which provides **low-latency** and **high-bandwidth** data access to each PE
- All the tasks execute in a **dataflow manner** that enables **task-level parallelism**

5: Evaluation & Key Takeaways

Performance Comparison



Energy Efficiency Comparison



Key Results

- **27.4×, 5.3×, and 12.7× speedup** compared to a complete POWER9 socket for SneakySnake, hdiff, and vadvc, respectively
- **1.4×, 1.2×, and 1.8× performance improvement** while using multiple HBM channels compared to a single HBM channel for SneakySnake, hdiff, and vadvc, respectively
- **133×, 12×, and 35× higher energy efficiency** than a complete POWER9 socket for SneakySnake, hdiff, and vadvc, respectively
- **HBM design avoids memory access congestion**, which is typical in DDR4-based FPGA designs, and ensures memory bandwidth scaling with the number of PEs
- **Single channel & multiple channel HBM designs**

Open-source:
https://github.com/CMU-SAFARI/SneakySnake/tree/master/SneakySnake-HLS-HBM

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]

Sibyl:
A Reinforcement Learning Approach to Data Placement in Hybrid Storage Systems

Gagandeep Singh, Rakesh Nadig, Jisung Park, Rahul Bera,
Nastaran Hajinazar, Juan Gómez-Luna, Onur Mutlu

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

ETH Zürich

Executive Summary

Motivation: Complement different storage technologies to **extend the overall capacity** and **reduce the system cost** with **minimal effect on the application performance**

Problem: Data allocation and movement between the heterogeneous devices to achieve **optimal (near-optimal) performance** of the storage system is challenging

Goal: Develop an **efficient, high-performant** data-placement mechanism for hybrid storage systems that can **flexibly adapt** to the **behavior of the workload** as well as the storage **device characteristics**

Sibyl

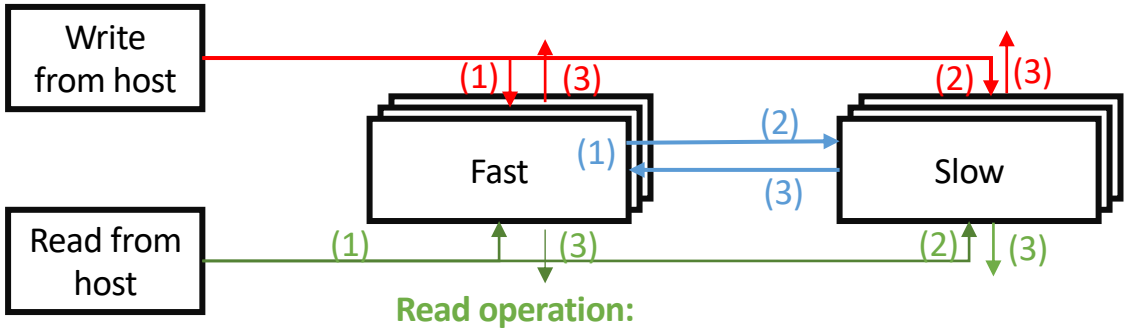
- Uses **reinforcement learning (RL)** to develop a **data-placement policy** that **decides which data** should be stored in the fast storage while **minimizing the migration overhead**
- Performs **dynamic data-placement decision** by learning **device characteristics** while taking into account **workload's inherent behavior**
- QRator improves I/O performance by **24.1%, 34.7%, and 27.9%** on average compared to **two state-of-the-art heuristic-based baselines and a supervised learning-based baseline, respectively,** and achieves **80%** performance of the oracle policy that has knowledge of future access patterns

Background: Hybrid Storage Systems

Write operation:

(1) Write **hot and random** data to the fast device

(2) Cold and sequential data to the slow device



Read operation:

(1) Read **hot and random** data from the fast device

(2) Read **cold and sequential** from the slow device

Frozen data eviction: (Migration)

(1) Evict **cold data** to the slow device

(2) In case of many reads to slow memory (**hot data**) move data to the fast device. Cold and sequential is read directly from the slow device

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

29 September 2021

SRC AIHW Annual Review

SAFARI ETH zürich Carnegie Mellon

Pythia

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

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

Executive Summary

- **Background**: Prefetchers learns to predict addresses of future memory requests by associating patterns with program context (called **feature**)
- **Problem**: Three key shortcomings in prior prefetchers:
  - Predicts mainly using a **single program feature**
  - Lacks **inherent system awareness**
  - Lacks **in-silicon customization** ability
- **Goal**: Design a prefetching framework that:
  - Learns from **multiple features** and **inherent system-level feedback**
  - Can be **customized in silicon** to change features and/or objective
- **Contribution**: **Pythia**, that formulates prefetching as reinforcement learning
  - **Adaptive, autonomous** learning using multiple features and system-level feedback
  - **Realistic, practical** implementation without any changes to software
- **Key Results**:
  - Evaluated using a wide range of workloads from SPEC CPU, PARSEC, Ligra, Cloudsuite
  - Outperforms prior SOTA by **3.4%, 7.7% and 16.9%** in 1/4/bandwidth-constrained cores
- **Open sourced**: https://github.com/CMU-SAFARI/Pythia

Talk Outline

Key Shortcomings of Prior Prefetchers

Formulating Prefetching as Reinforcement Learning

Pythia Overview

Evaluation of Pythia and Key Results

Conclusion

Prefetching Basics

- Predicts address of **long-latency memory requests** and fetches data before the program demands

- Associates access patterns from past memory requests to program context information

Program Feature → Access Pattern

- PC, Page#, Page offset, Cacheline delta, …
  - Any combination of these

Key Shortcomings in Prior Prefetchers

- We observe **three key shortcomings** that significantly limits performance benefits

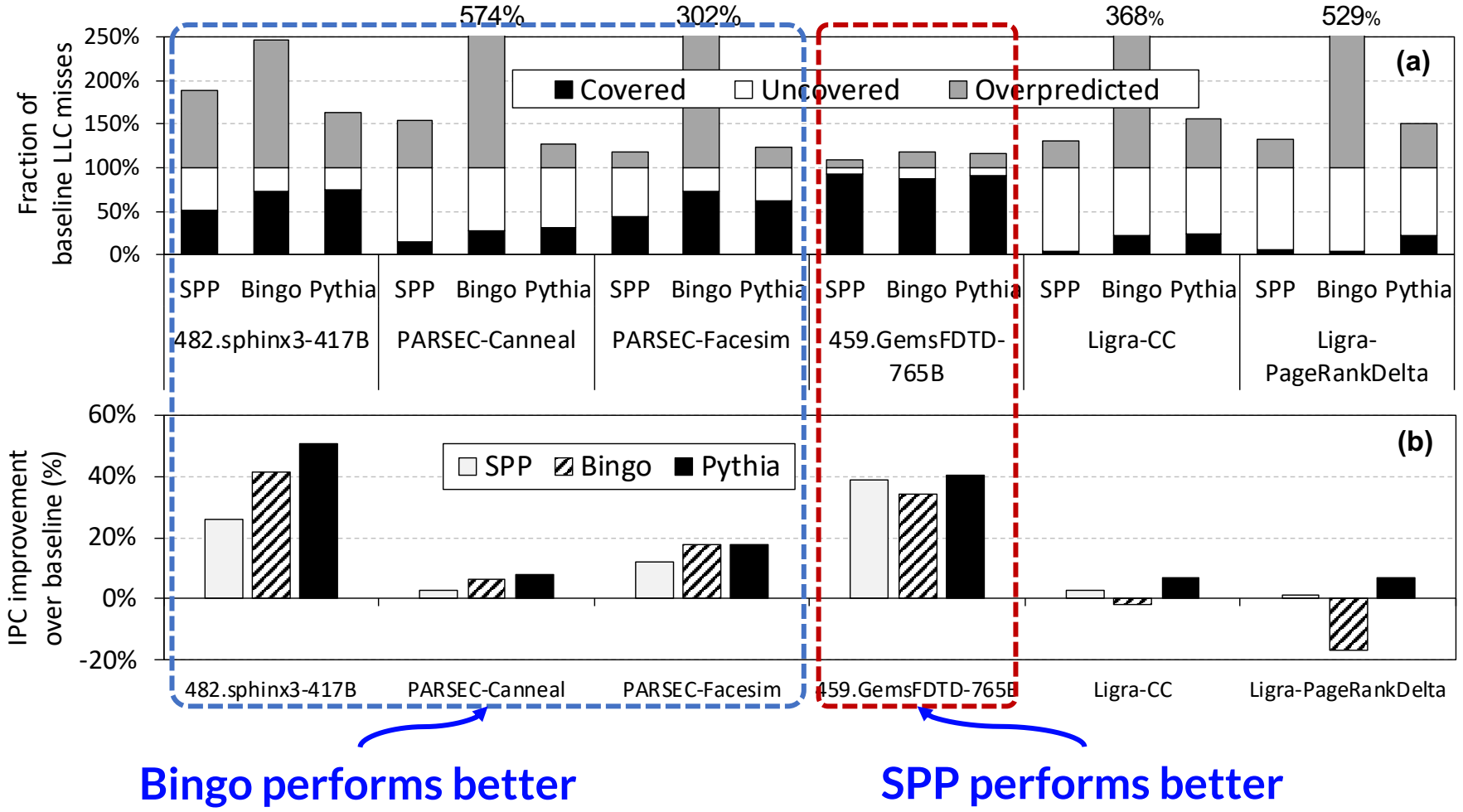
1 Predicts mainly using a single program feature

2 Lacks inherent system awareness

3 Lacks in-silicon customization ability

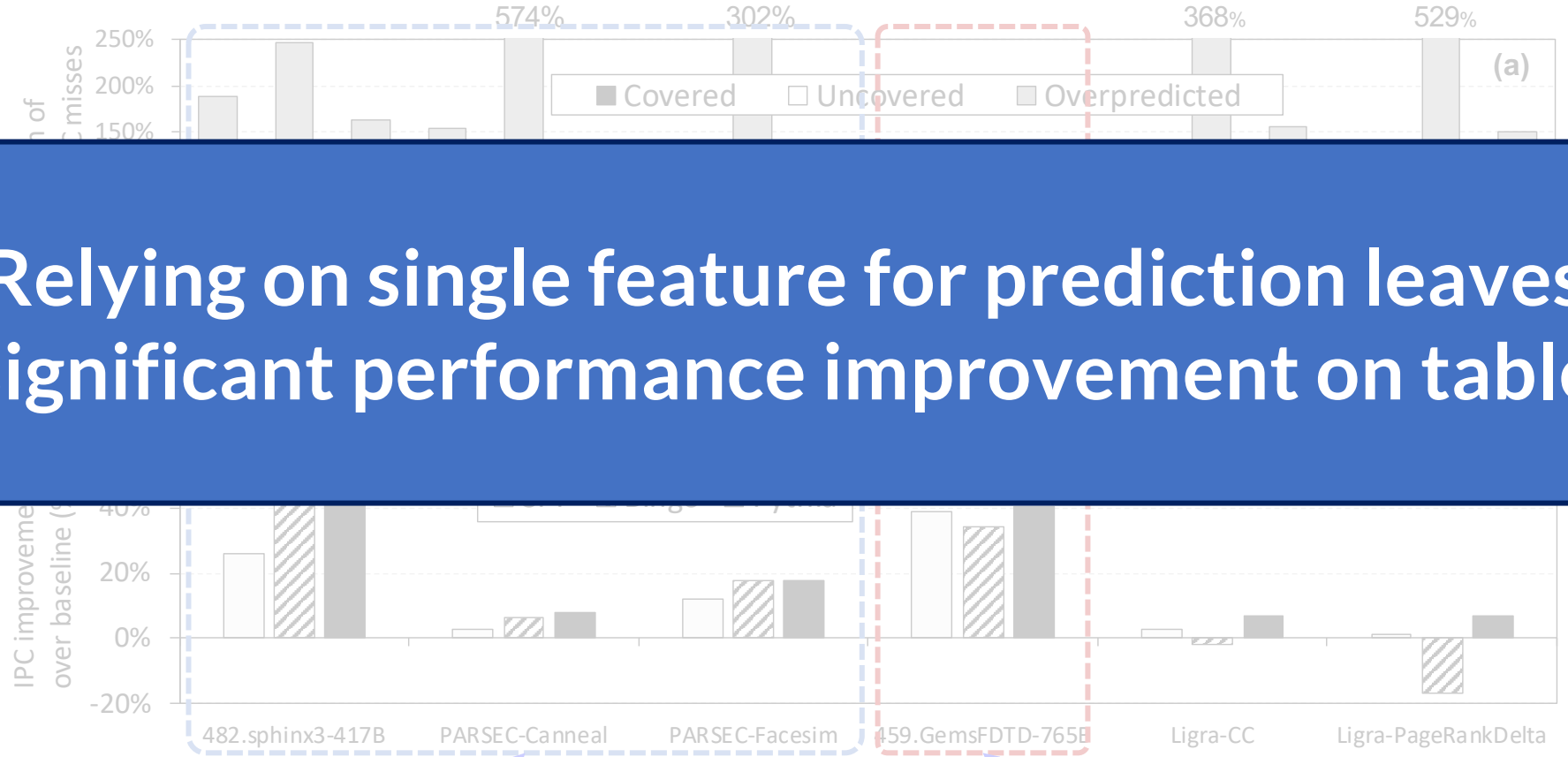
(1) Single-Feature Prefetch Prediction

- Provides benefits mainly on those workloads where the feature to pattern correlation exists



(1) Single-Feature Prefetch Prediction

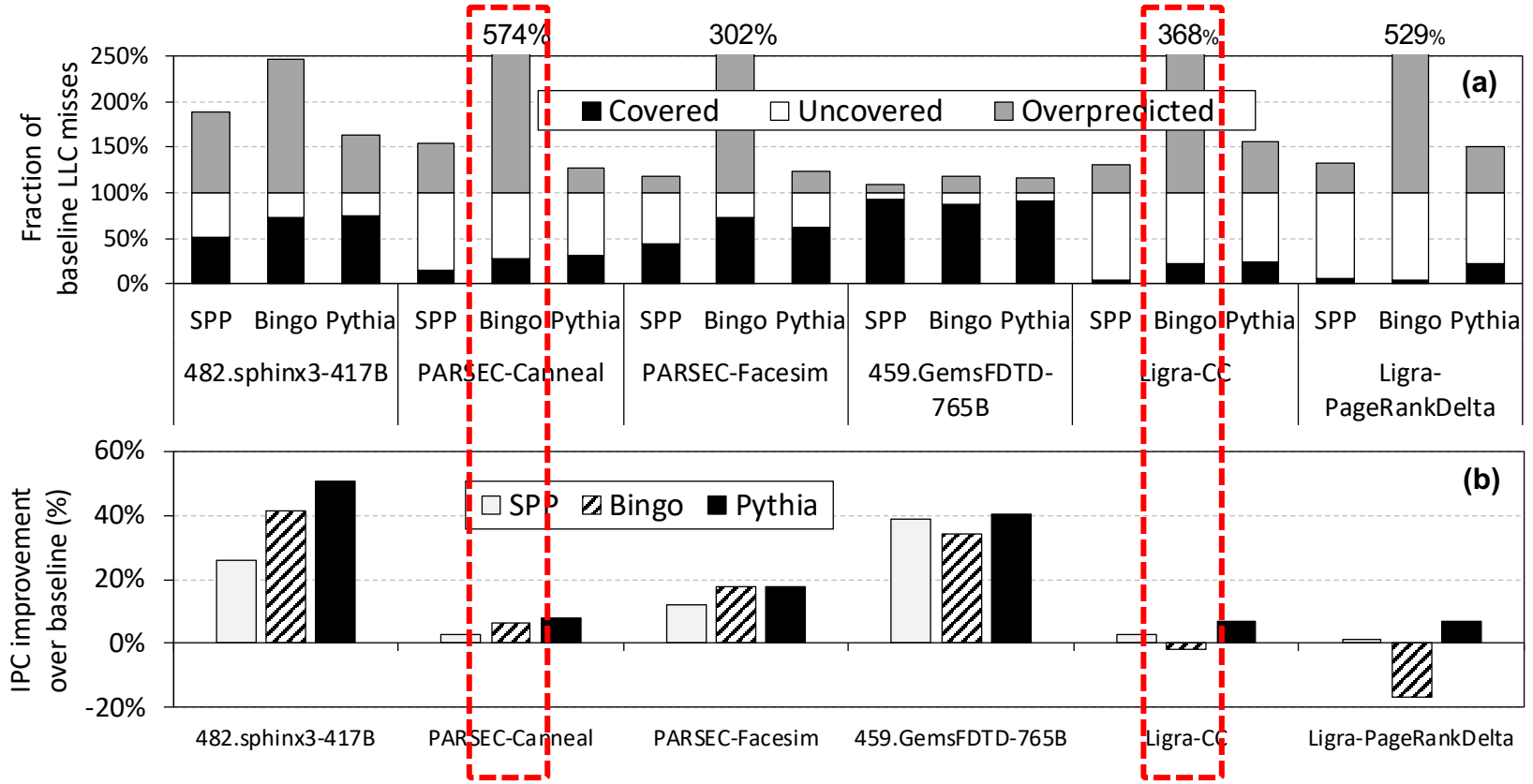
- Provides benefits mainly on those workloads where the feature to pattern correlation exists



Relying on single feature for prediction leaves significant performance improvement on table

(2) Lack of Inherent System Awareness

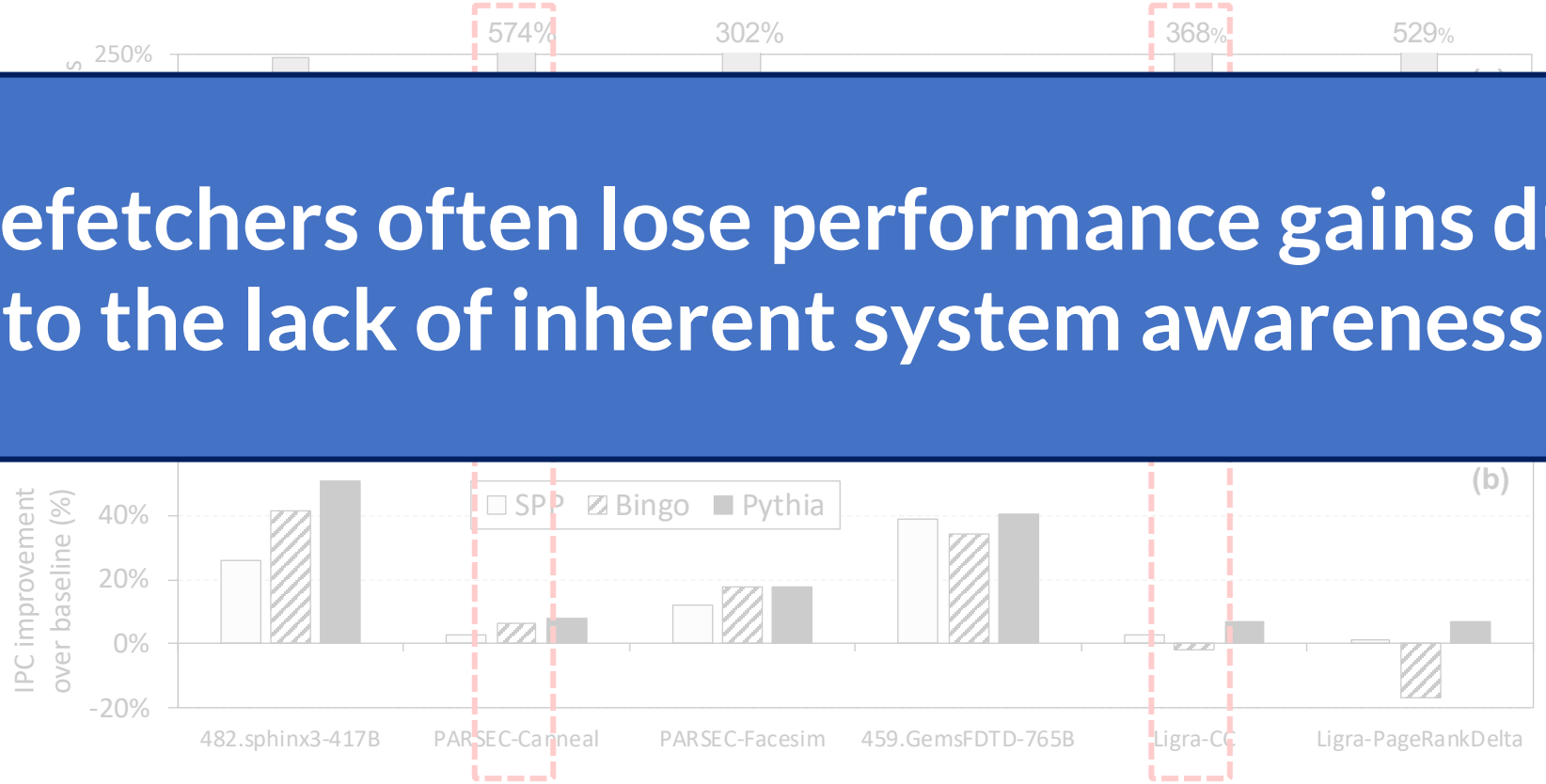
- Little understanding of **undesirable effects** (e.g., memory bandwidth usage, cache pollution, …)
  - Performance loss in **resource-constrained** configurations



Similar coverage, lower overpredictions
Yet lower performance

(2) Lack of Inherent System Awareness

- Little understanding of **undesirable effects** (e.g., memory bandwidth usage, cache pollution, …)
  - Performance loss in **resource-constrained** configurations

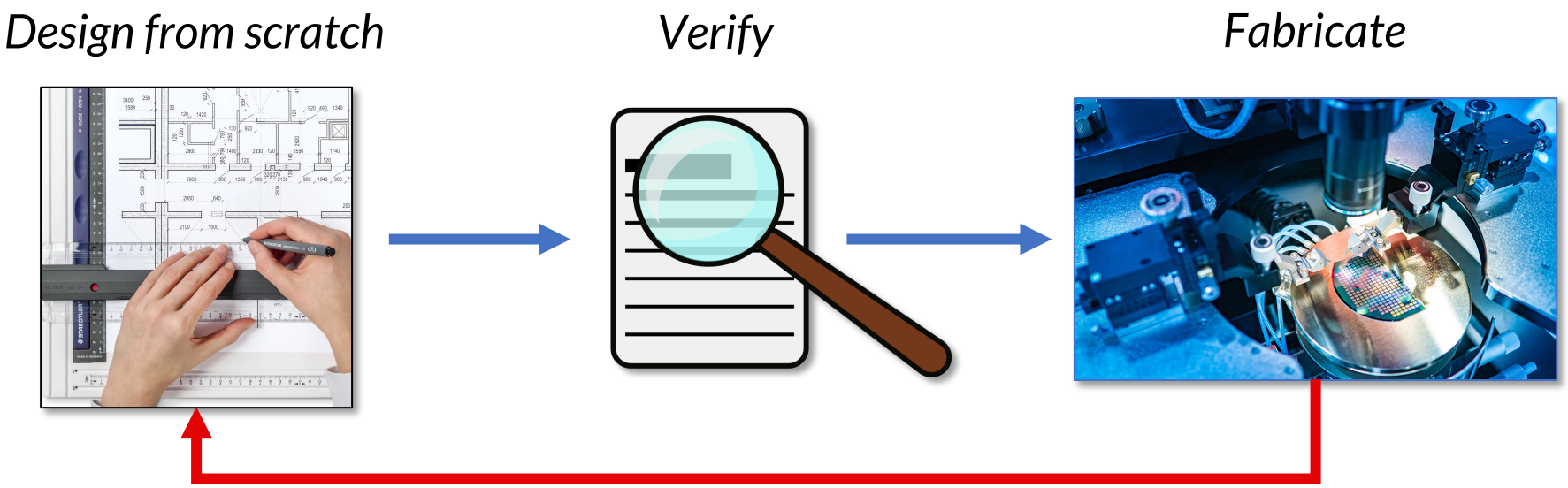


Prefetchers often lose performance gains due to the lack of inherent system awareness

Similar coverage, lower overpredictions
Yet lower performance

(3) Lack of In-silicon Customizability

- Feature **statically** selected at design time
  - **Rigid hardware** designed specifically to exploit that feature



Our Goal

A prefetching **framework** that:

1. Can learn to prefetch using **multiple features** and **inherent system-level feedback** information
2. Can be **easily customized in silicon** to change feature type and/or prefetcher's objective

Talk Outline

Key Shortcomings of Prior Prefetchers

Formulating Prefetching as Reinforcement Learning

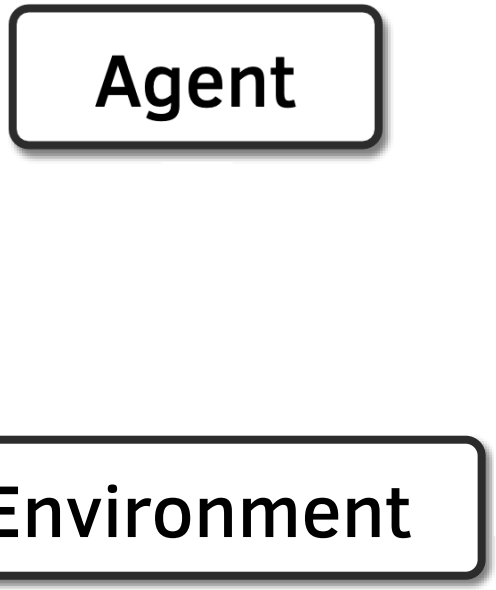
Pythia Overview

Evaluation of Pythia and Key Results

Conclusion

Basics of Reinforcement Learning (RL)

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



- Agent stores **Q-values** for *every* state-action pairs
  - Given a state, selects action that provides **highest** Q-value

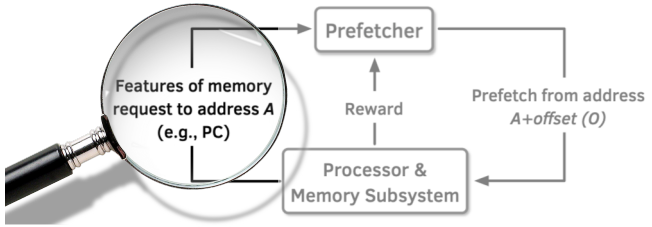
Formulating Prefetching as RL

What is State?

- ***k*-dimensional** vector of features

$$S \equiv \{\phi_S^1, \phi_S^2, \ldots, \phi_S^k\}$$

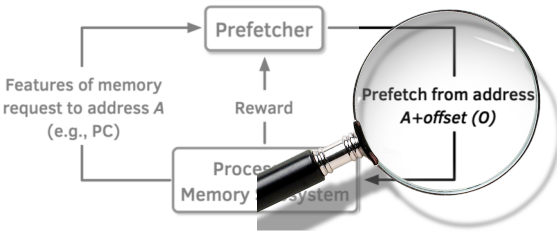
- Feature = control-flow + data-flow



- **Control-flow**
  - PC
  - Branch PC
  - Last-3 PCs, …
- **Data-flow**
  - Cacheline address
  - Page#
  - Delta between two cacheline address
  - Last 4 deltas, …

What is Action?

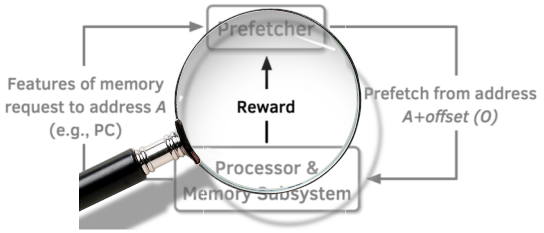
- Selection of a **prefetch offset**
  - Add to demanded cacheline to get prefetch cacheline address



- A **zero offset** means **no prefetch** is generated

What is Reward?

- Defines the **objective** of Pythia
- **Five** distinct levels
  - *Accurate and timely* (R_{AT})
  - *Accurate but late* (R_{AL})
  - *Loss of coverage* (R_{CL})
  - *Inaccurate*
    - When memory **b/w usage is low** (R_{IN}-L)
    - When memory **b/w usage is high** (R_{IN}-H)
  - *No-prefetch*
    - When memory **b/w usage is low** (R_{NP}-L)
    - When memory **b/w usage is high** (R_{NP}-H)
- Values are set at design time via **automatic design-space exploration**
  - Can be **customized** further in silicon for higher performance



Talk Outline

Key Shortcomings of Prior Prefetchers

Formulating Prefetching as Reinforcement Learning

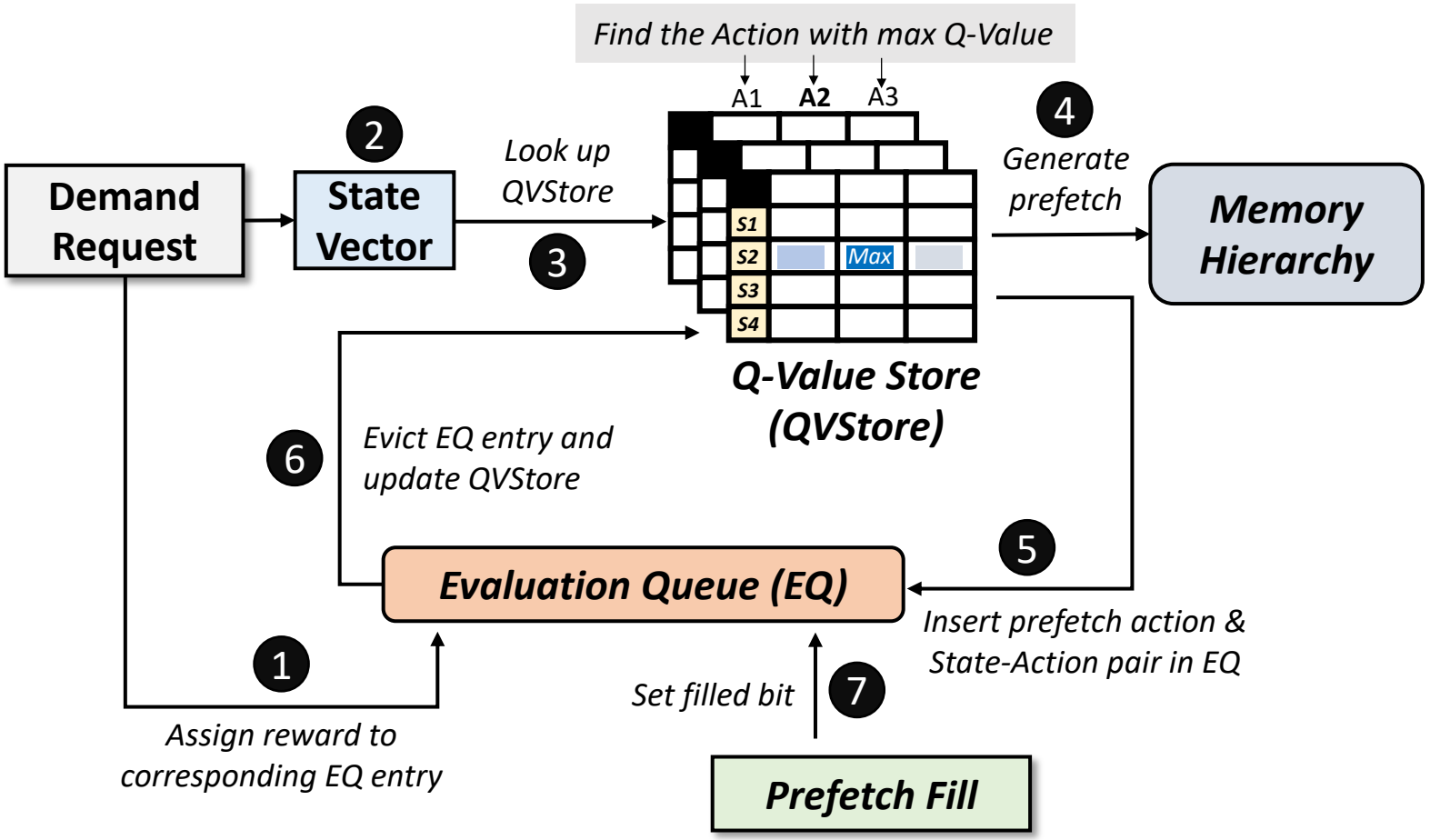
Pythia Overview

Evaluation of Pythia and Key Results

Conclusion

Pythia Overview

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

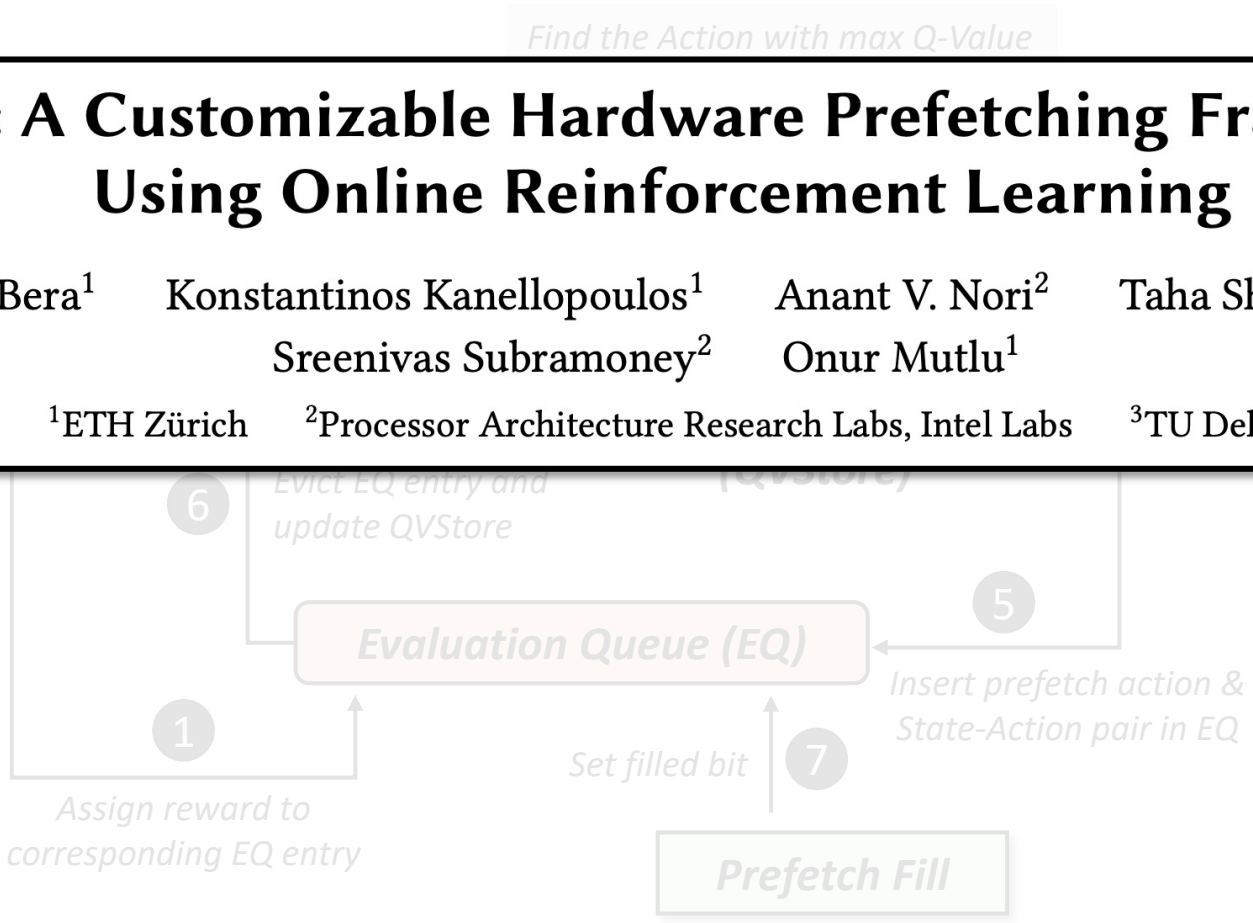


Pythia Overview

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

Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera[1] Konstantinos Kanellopoulos[1] Anant V. Nori[2] Taha Shahroodi[3,1]
Sreenivas Subramoney[2] Onur Mutlu[1]
[1]ETH Zürich [2]Processor Architecture Research Labs, Intel Labs [3]TU Delft



Talk Outline

Key Shortcomings of Prior Prefetchers

Formulating Prefetching as Reinforcement Learning

Pythia Overview

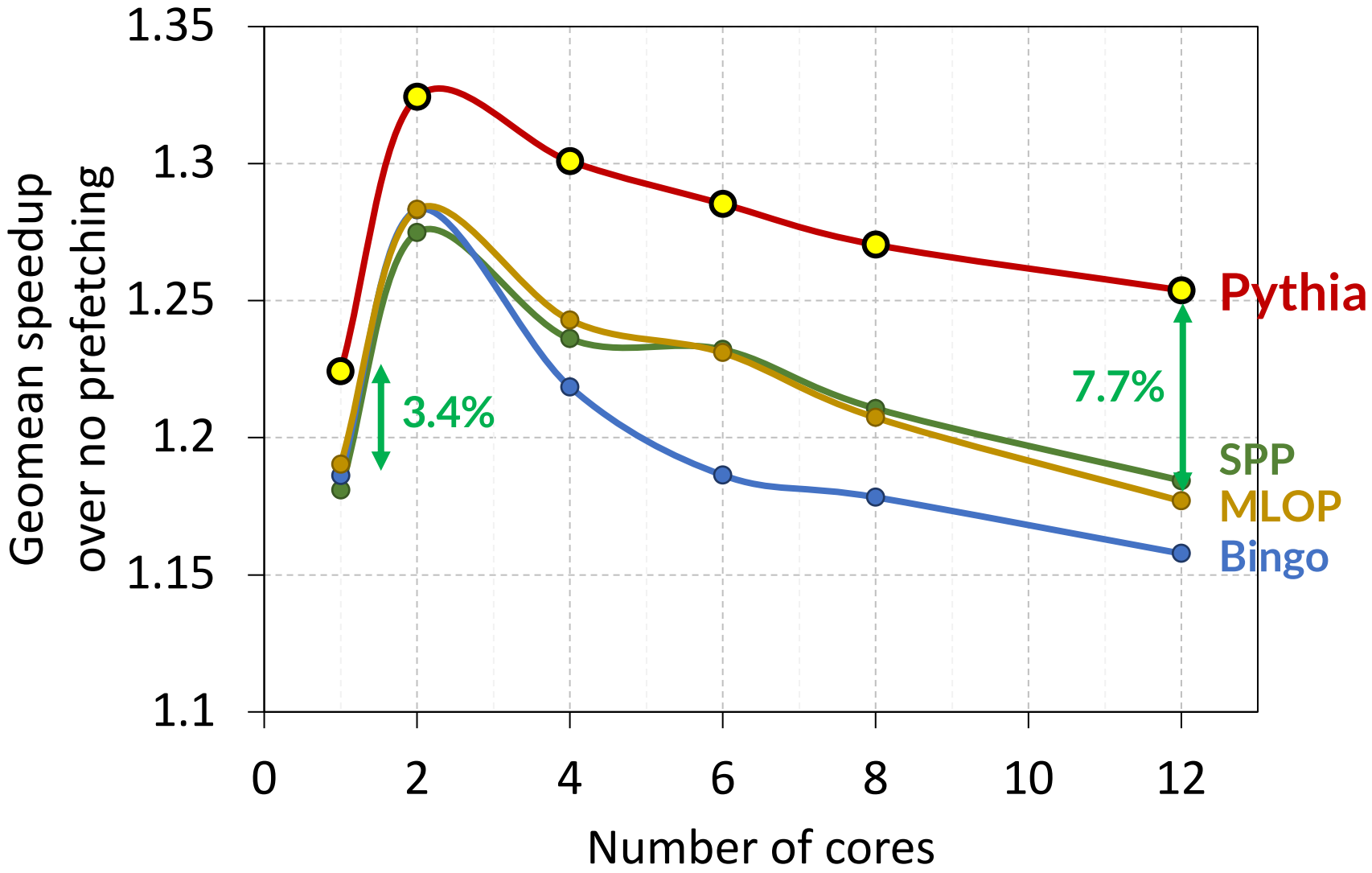
Evaluation of Pythia and Key Results

Conclusion

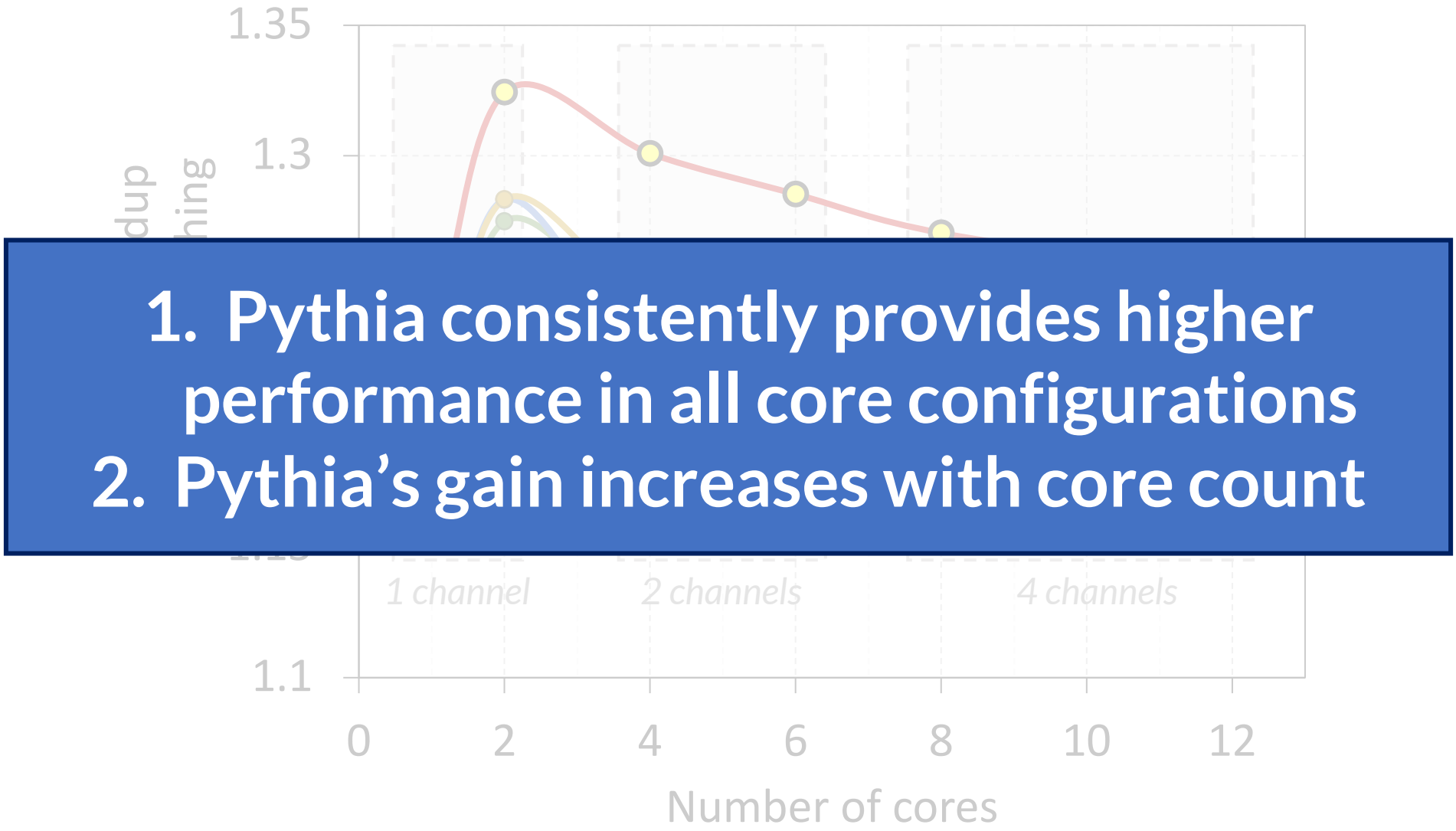
Simulation Methodology

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

Performance with Varying Core Count

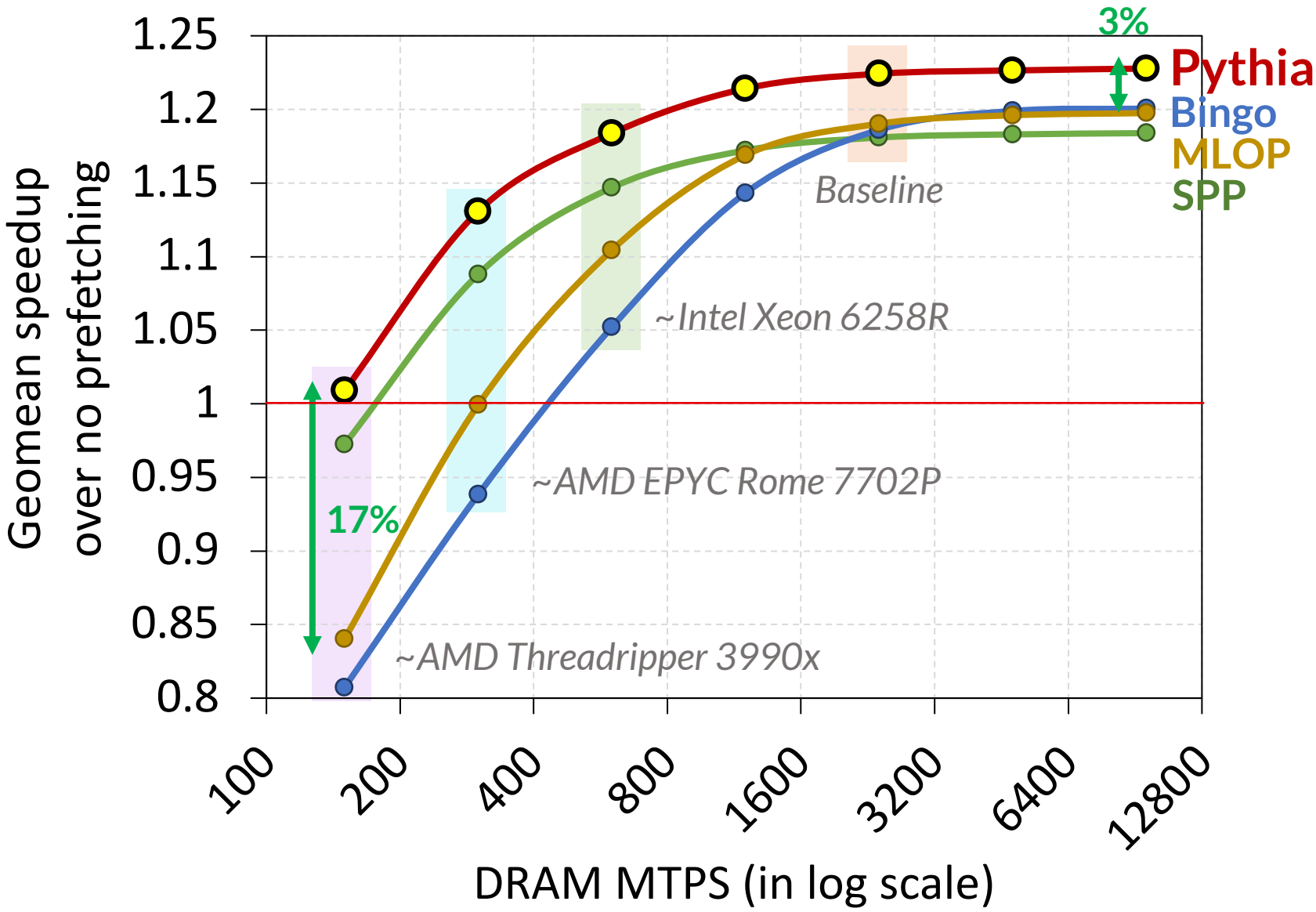


Performance with Varying Core Count

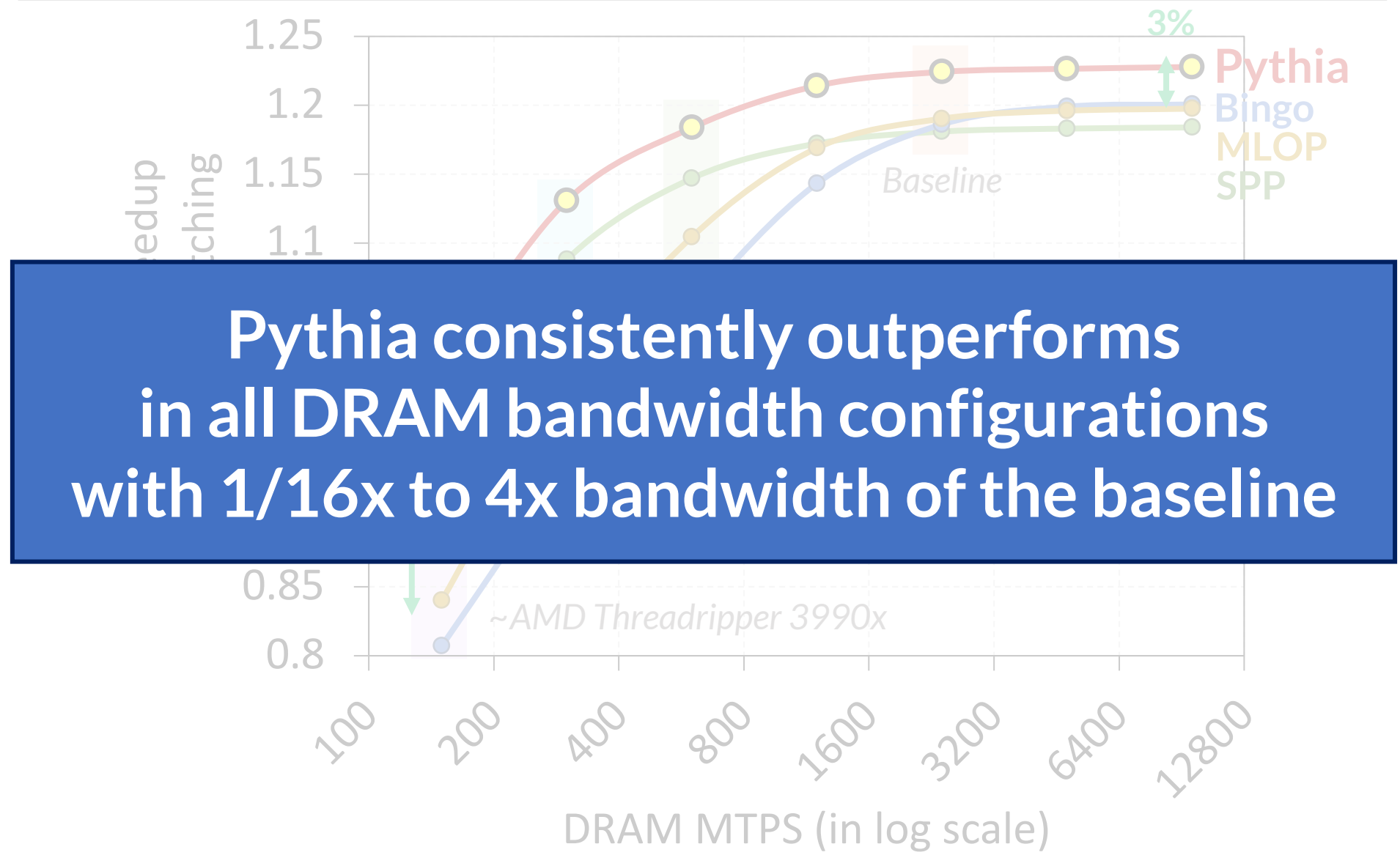


1. Pythia consistently provides higher performance in all core configurations
2. Pythia's gain increases with core count

Performance with Varying DRAM Bandwidth



Performance with Varying DRAM Bandwidth



Pythia consistently outperforms in all DRAM bandwidth configurations with 1/16x to 4x bandwidth of the baseline

More in the Paper

- Automatic **design-space exploration** for Pythia
- Details about **reward assignment** and QVStore update
- More results
  - ✓Performance comparison against **multi-level prefetchers**
  - ✓Detailed analysis of single-core and four-core performance
  - ✓Performance comparison with **unseen traces**
  - ✓Understanding Pythia's learning with **a case study**
  - ✓Performance benefits via **customization**

More in the Paper

- Automatic design-space exploration for Pythia
- Details about reward assignment and QVStore update

Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

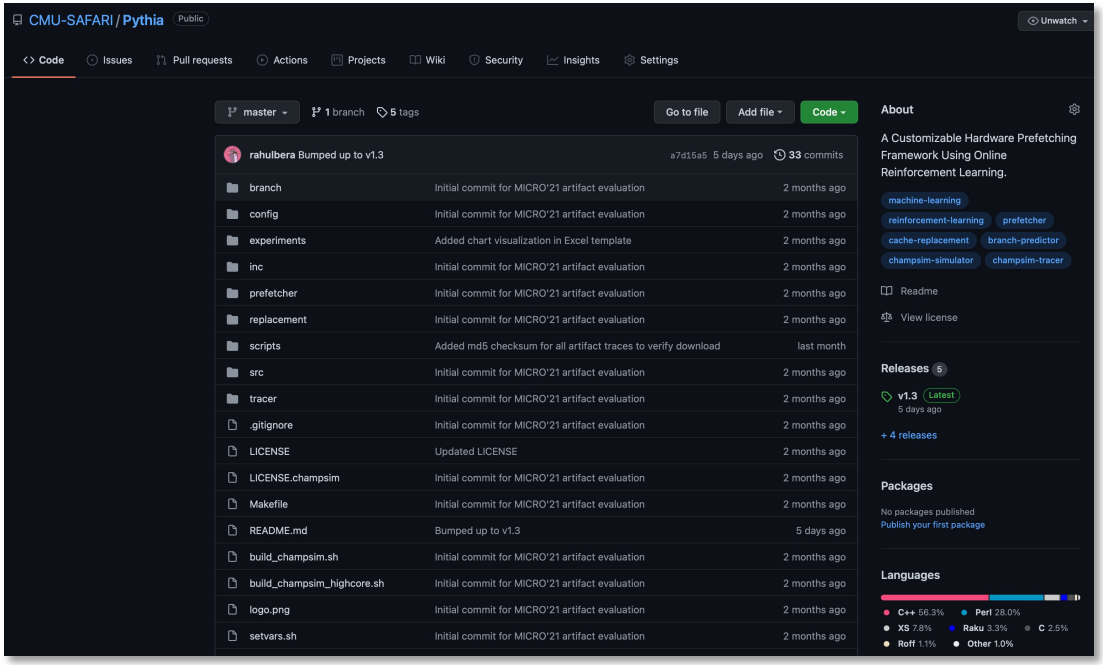
Rahul Bera[1] Konstantinos Kanellopoulos[1] Anant V. Nori[2] Taha Shahroodi[3,1]
Sreenivas Subramoney[2] Onur Mutlu[1]
[1]ETH Zürich [2]Processor Architecture Research Labs, Intel Labs [3]TU Delft

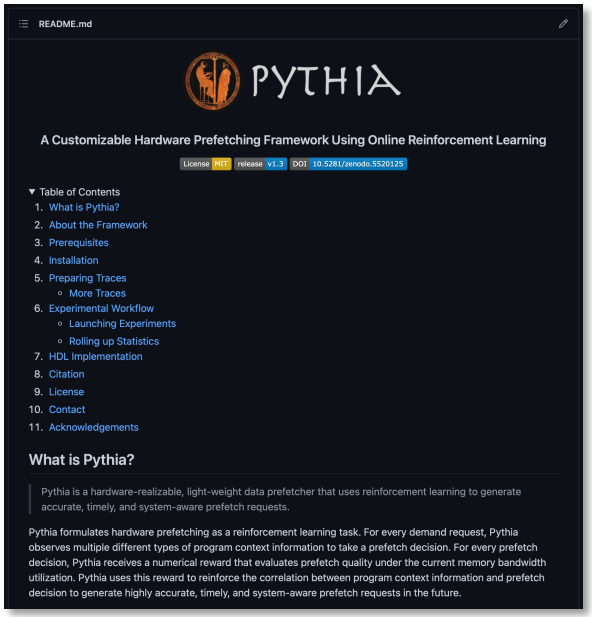
- ✓Performance comparison with unseen traces
- ✓Understanding Pythia's learning with a case study
- ✓Performance benefits via customization

Pythia is Open-sourced

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

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





Talk Outline

Key Shortcomings of Prior Prefetchers

Formulating Prefetching as Reinforcement Learning

Pythia Overview

Evaluation of Pythia and Key Results

Conclusion

Conclusion

- **Background**: Prefetchers learns to predict future addresses by associating patterns with program context (called **feature**)
- **Problem**: Three key shortcomings in prior prefetchers:
  - Predicts mainly using a **single program feature**
  - Lacks **inherent system awareness**
  - Lacks **online customization** ability
- **Goal**: Design a prefetching framework that:
  - Learns from **multiple features** and **inherent system-level feedback**
  - Can be **customized online** to change features and/or objective
- **Contribution**: **Pythia**, that formulates prefetching as reinforcement learning
  - **Adaptive, autonomous** learning using multiple features and system-level feedback
  - **Realistic, practical** implementation without any changes to software
- **Key Results**:
  - Evaluated using a wide range of workloads from SPEC CPU, PARSEC, Ligra, Cloudsuite
  - Outperforms prior SOTA by **3.4%, 7.7% and 16.9%** in 1/4/bandwidth-constrained cores
- **Open sourced**: https://github.com/CMU-SAFARI/Pythia

Pythia

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

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

Semiconductor Research Corporation

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

Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY
ETH Zürich
SAFARI
IBM Research | Zurich

Near-Memory Acceleration

Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gomez-Luna, Henk Corporaal, Onur Mutlu,
"FPGA-Based Near-Memory Acceleration of Modern Data-Intensive Applications"
IEEE Micro, 2021.
[Source Code]

Home / Magazines / IEEE Micro / 2021.04

IEEE Micro

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

July-Aug. 2021, pp. 39-48, vol. 41
DOI Bookmark: 10.1109/MM.2021.3088396

Authors

Gagandeep Singh, ETH Zürich, Zürich, Switzerland
Mohammed Alser, ETH Zürich, Zürich, Switzerland
Damla Senol Cali, Carnegie Mellon University, Pittsburgh, PA, USA
Dionysios Diamantopoulos, Zürich Lab, IBM Research Europe, Rüschlikon, Switzerland
Juan Gomez-Luna, ETH Zürich, Zürich, Switzerland
Henk Corporaal, Eindhoven University of Technology, Eindhoven, The Netherlands
Onur Mutlu, ETH Zürich, Zürich, Switzerland

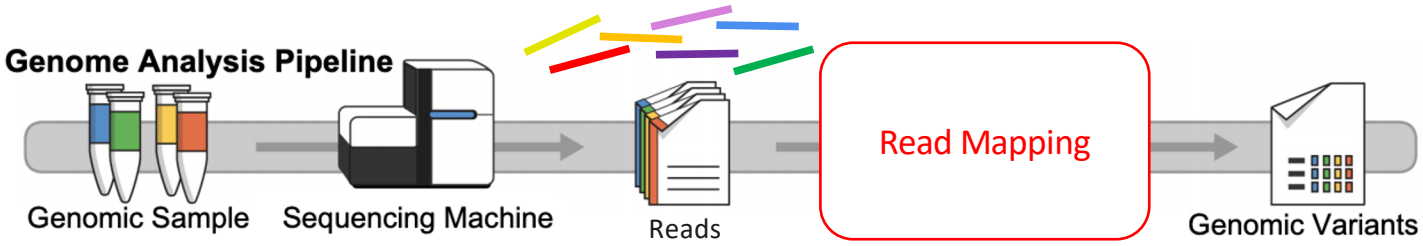
How to Analyze a Genome?

NO

machine gives the **complete sequence** of genome as output

>CCTCCTCAGTGCCACCCAGCCCACTGGCAGCTCCCAAACAGGCTCTTATTAAAACACCCTGTTCCCTGCCCCTTGGAGTGAGGTGTCAA
GGACCTAAACTAAAAAAAAAAAAAGAAAAAGAAAAGAAAAAGAATTTAAAATTTAAGTAATTCTTTGAAAAAAACTAATTTCTAAGCTT
CTTCATGTCAAGGACCTAATGTGCTAAACAGCACTTTTTTGACCATTATTTTGGATCTGAAAGAAATCAAGAATAAATGAAGGACTTGAT
ACATTGGAAGAGGAGAGTCAAGGACCTACAGAAAAAAAAAAAAAAGAAAAAGAAAAGAAAAAGA**A**TTTAAAATTTAAGTAATTCTTT
GAAAAAAACTAATTTCTAAGCTTCTT**C**ATGTCAAGGACCTAATGTCTGTGTTGCAGGTCTTCTTGCATTTCCCTGTCAAAAGAAAAAGAA
TTTAAAATTTAAGTAATTCTTTGAAAAAAACTAATTTCTAAGCTTCTTCATGTCAAGGACCTAATGTCAGGCCAAGAGTTGCAAAAAAAA
AAAAAGAAAAAGAAAAGAAAAAGAATTTAAAATTTA**AGTAATTCTTTGAAAAAAACTAATTTCTAAGCTTCTTCAT**GTCAAGGACCTAA
TGTAGCCAGAATGGTTGTGGGATGGGAGCCTCTGTGGACCGACCAGGTAGCTCTCTTTTCCACACTGTAGTCTCAAAGCTTCTTCATGT
GGTTTCTCTGAGTGAAAAAAAAAAAAAGAAAAAGAAAAGAAAAAGAATTTAAAATTTAAGTAATTCTTTGAAAAAAACTAATTTCTAA
GCTT**T**TTCATGTCAAGGACCTAATGTAGCTATACTGAACGTTATCTAGGGGAAAGATTGAAGGGGAGCTCTAAGGTCAACACACCACCA
CTTCCCAGAAAGCTTCTTCA……

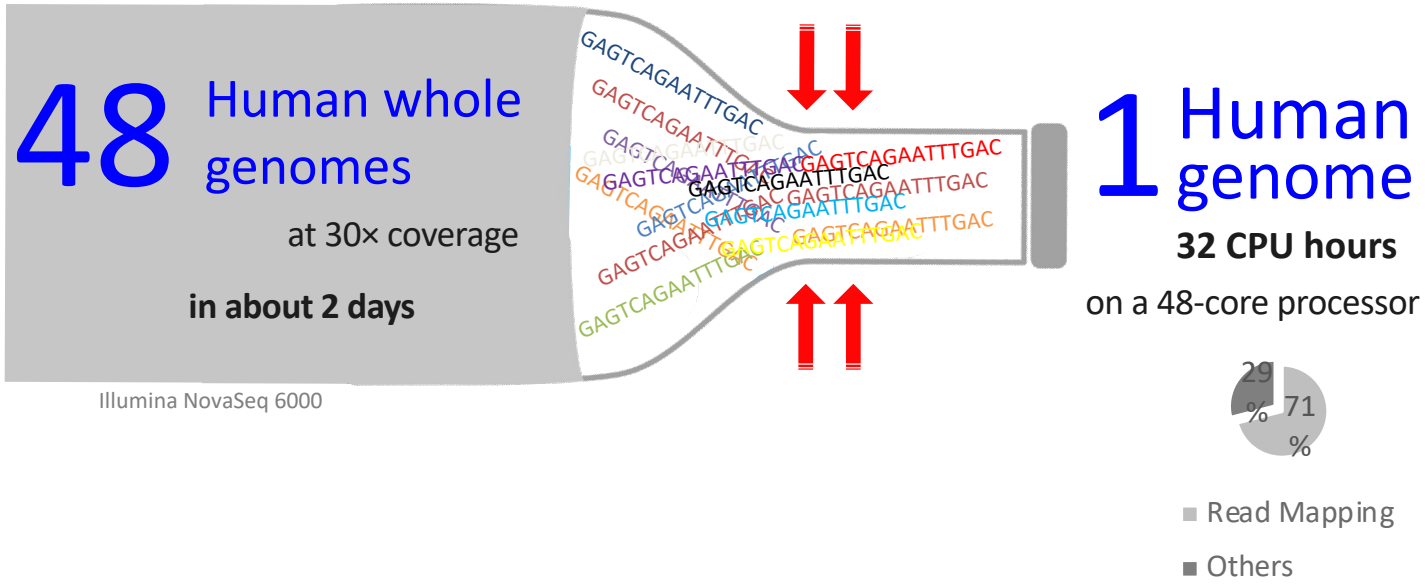
Genome Analysis in Real Life



Current sequencing machine provides
small randomized fragments
of the original DNA sequence

Alser+, "Technology dictates algorithms: Recent developments in read alignment", Genome Biology, 2021

Bottlenecked in Read Mapping!!



Goyal+, "Ultra-fast next generation human genome sequencing data processing using DRAGENTM bio-IT processor for precision medicine", *Open Journal of Genetics,* 2017.

Stencil Computation in Weather Modeling

COSMO (Consortium for Small-Scale Modeling)

- Around **80 complex stencils**

- Horizontal diffusion
  Vertical advection

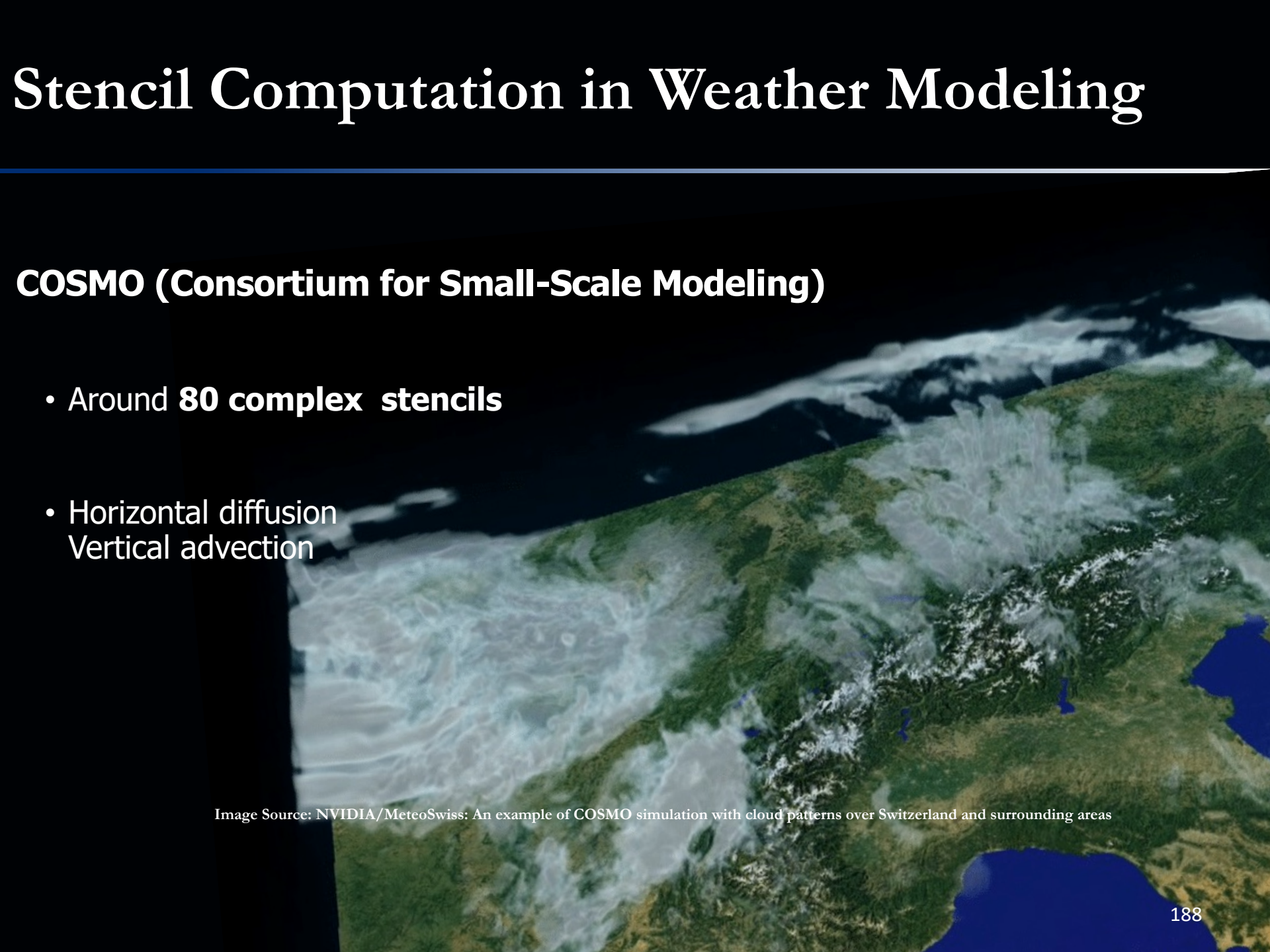


Image Source: NVIDIA/MeteoSwiss: An example of COSMO simulation with cloud patterns over Switzerland and surrounding areas

NERO: Weather Prediction Accelerator [FPL 2020]

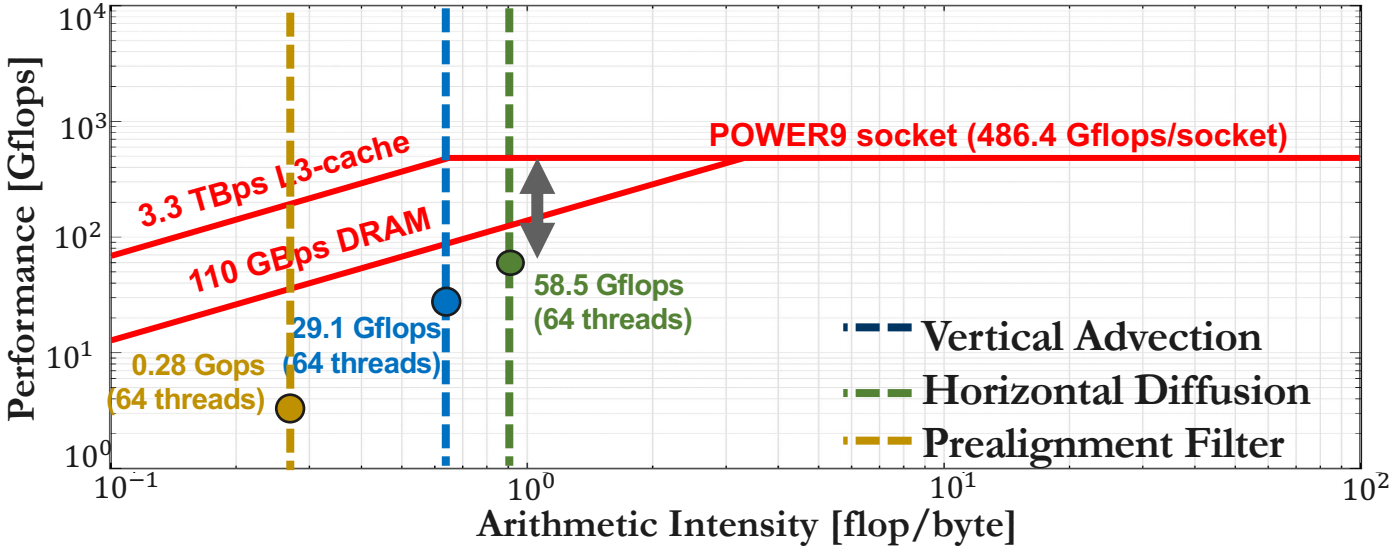
- **Gagandeep Singh**, Dionysios Diamantopoulos, Christoph Hagleitner, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal,
  "NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling"
  *Proceedings of the 30th International Conference on Field-Programmable Logic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

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

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

Motivation and Goal

Memory bound with **limited performance** and **high energy consumption** on **IBM POWER9 CPU**



Goal:

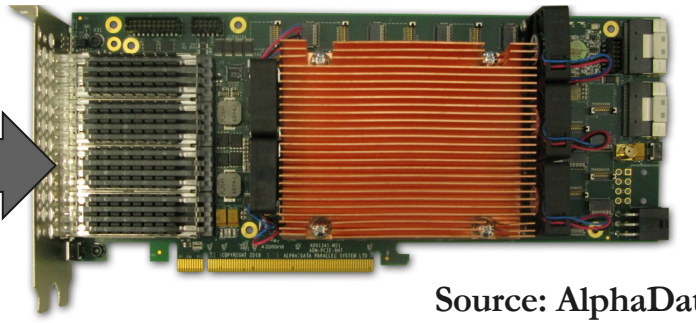
- **Mitigate** the **performance bottleneck** of modern data-intensive applications in an **energy-efficient way**

Near-Memory Acceleration

Source: IBM

IBM POWER9 CPU

OCAPI

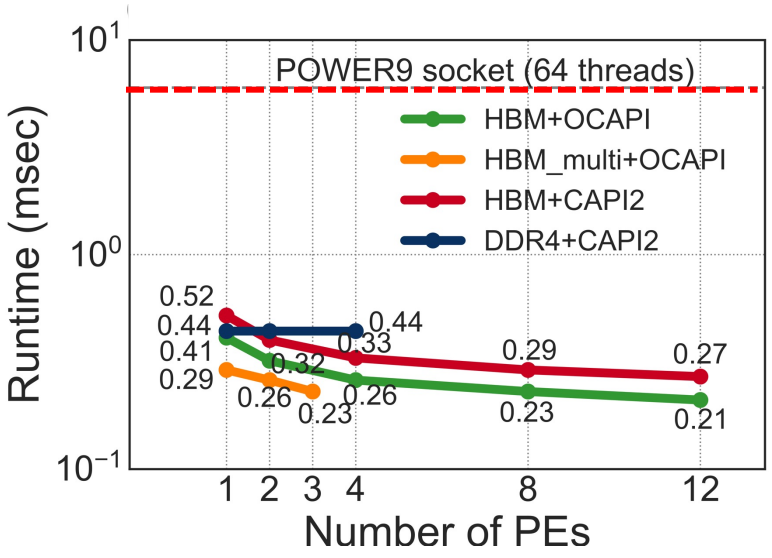


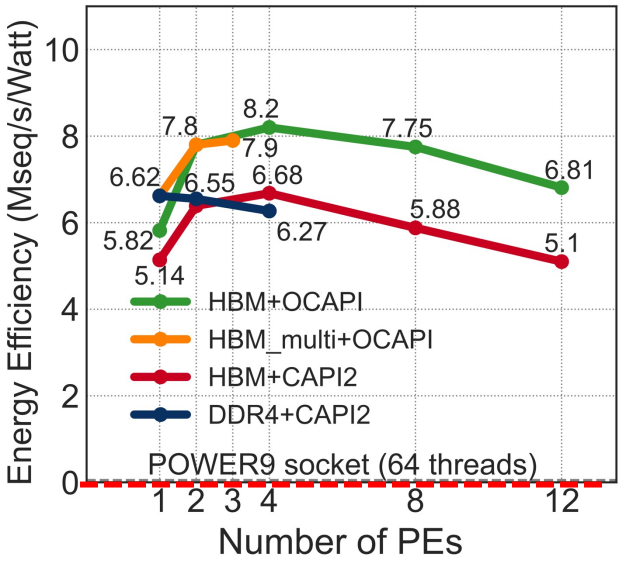
Source: AlphaData

HBM-based FPGA board

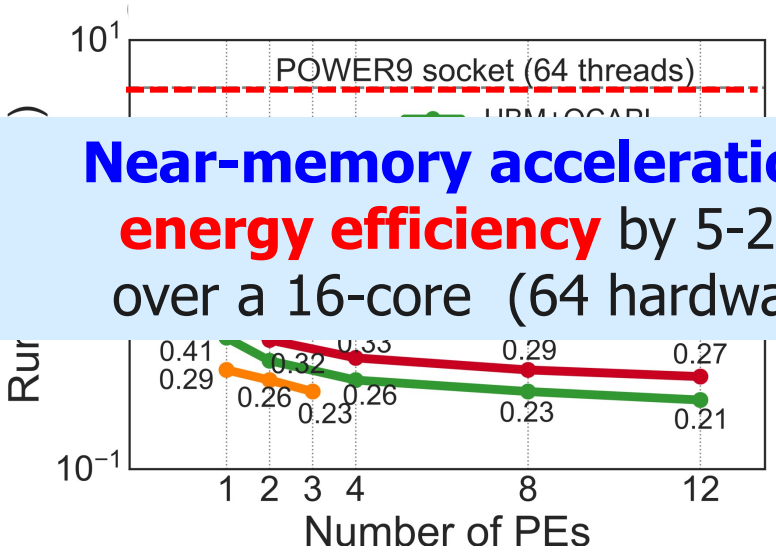
Near-HBM FPGA-based accelerator

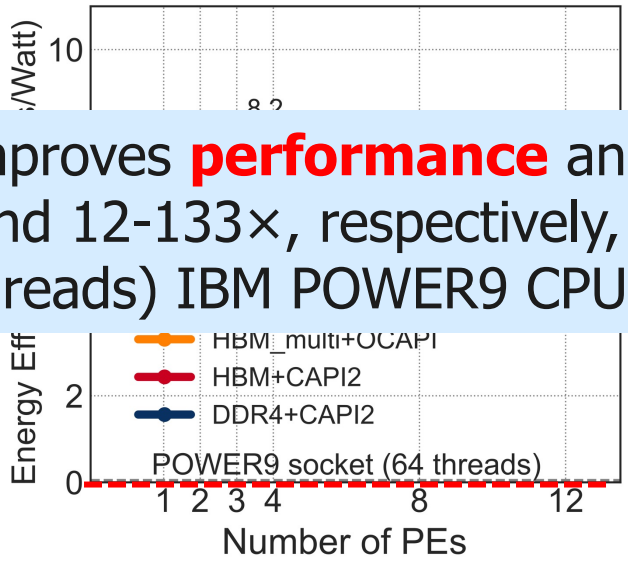
Key Results of Near-Memory Acceleration





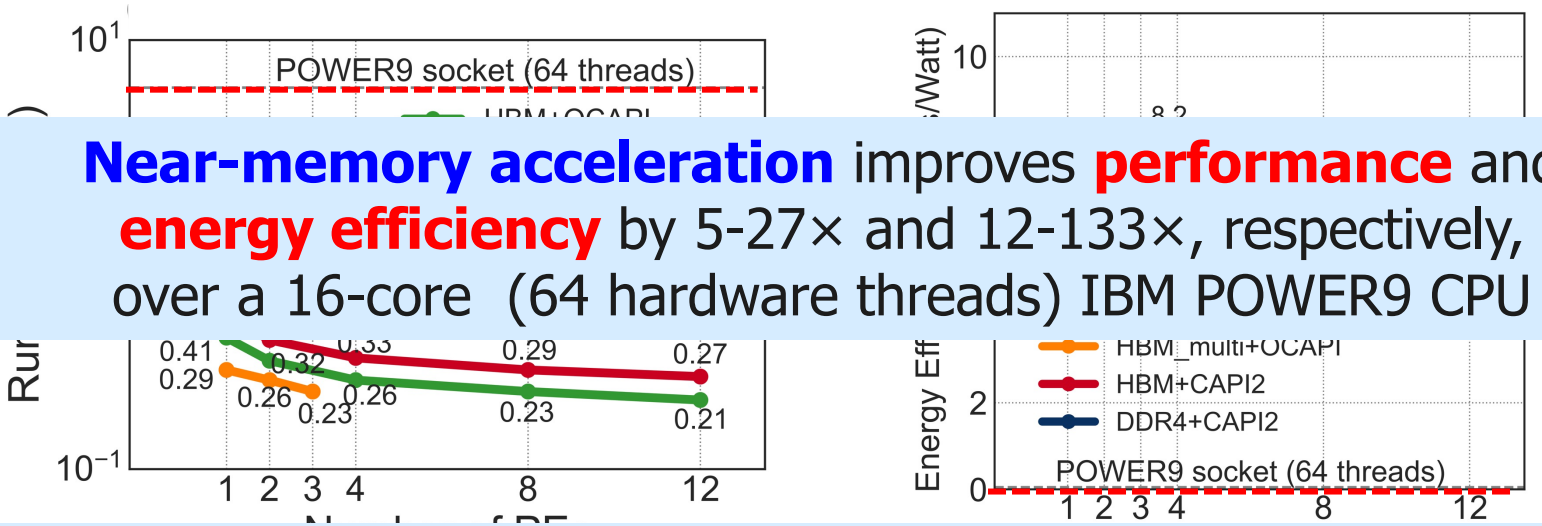
Key Results of Near-Memory Acceleration





Near-memory acceleration improves **performance** and **energy efficiency** by 5-27× and 12-133×, respectively, over a 16-core (64 hardware threads) IBM POWER9 CPU

Key Results of Near-Memory Acceleration



Near-memory acceleration improves **performance** and **energy efficiency** by 5-27× and 12-133×, respectively, over a 16-core (64 hardware threads) IBM POWER9 CPU

Single channel & multiple channel HBM designs
Open-source: https://github.com/CMU-SAFARI

Semiconductor Research Corporation

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

Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY
ETH Zürich
SAFARI
IBM Research | Zurich