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
10 November 2022
Intel/SRC Research Program Review

SAFARI ETH zürich Carnegie Mellon

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
  - Our systems need to make more adaptive and better decisions

In This Task… (Task #2946.001)

- We focus on designing memory systems to handle large amounts of data and emerging data-intensive workloads
- Goal: solve two different yet related & synergistic problems
- We explore (and exploit the synergy between)
  - Memory system design for AI/ML workloads/accelerators
  - AI/ML techniques for improving memory system designs

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 (2)

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

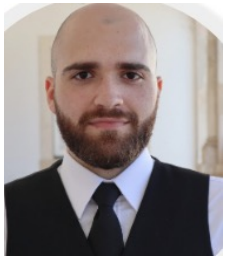




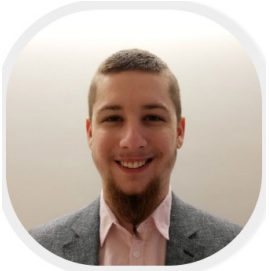
- More students/postdocs to be added as the task evolves

Task Information #2946.001 (1)

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


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

Industry Liaisons

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

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

Two Major Thrusts

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

2. AI/ML techniques for improving (memory) system designs

Challenge and Opportunity for Future

Self-Optimizing (Data-Driven) Computing Architectures

System Architecture Design Today

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

Can we design fundamentally intelligent architectures?

An Intelligent Architecture

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

We need to rethink design (of all controllers)

Self-Optimizing Memory Prefetchers

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

Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

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

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

Learning-Based Off-Chip Load Predictors

- **Best Paper Award at MICRO 2022**

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

Rahul Bera[1] Konstantinos Kanellopoulos[1] Shankar Balachandran[2] David Novo[3]
Ataberk Olgun[1] Mohammad Sadrosadati[1] Onur Mutlu[1]

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

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

Self-Optimizing Hybrid SSD Controllers

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

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

Gagandeep Singh[1] Rakesh Nadig[1] Jisung Park[1] Rahul Bera[1] Nastaran Hajinazar[1]
David Novo[3] Juan Gómez-Luna[1] Sander Stuijk[2] Henk Corporaal[2] Onur Mutlu[1]
[1]ETH Zürich [2]Eindhoven University of Technology [3]LIRMM, Univ. Montpellier, CNRS

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

Self-Optimizing Memory Controllers

- Engin Ipek, Onur Mutlu, José F. Martínez, and Rich Caruana, **"Self Optimizing Memory Controllers: A Reinforcement Learning Approach"**
  *Proceedings of the 35th International Symposium on Computer Architecture (**ISCA**)*, pages 39-50, Beijing, China, June 2008.

Self-Optimizing Memory Controllers: A Reinforcement Learning Approach

Engin İpek[1,2] **Onur Mutlu**[2] **José F. Martínez**[1] **Rich Caruana**[1]

[1]Cornell University, Ithaca, NY 14850 USA
[2] Microsoft Research, Redmond, WA 98052 USA

Pythia: Prefetching using Reinforcement Learning

Self-Optimizing Memory Prefetchers

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

Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

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

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

Pythia

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

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

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

SAFARI Research Group
safari.ethz.ch

ETH zürich

intel

TU Delft

1 Mainly use one program context info. for prediction

2 Lack inherent system awareness

3 Lack in-silicon customizability

Why do prefetchers not perform well?

Pythia

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

Can be **customized in silicon** to change program context information or prefetching objective on the fly

Brief Overview of Pythia

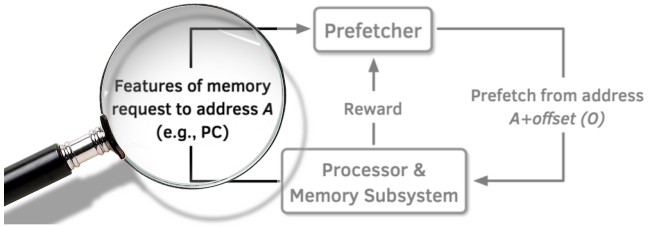
Pythia formulates prefetching as a **reinforcement learning** problem

What is State?

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

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

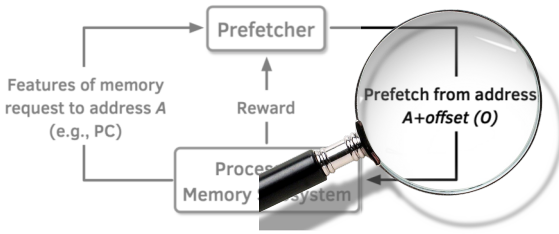
- Feature = control-flow + data-flow



- **Control-flow examples**
  - PC
  - Branch PC
  - Last-3 PCs, …
- **Data-flow examples**
  - Cacheline address
  - Physical page number
  - Delta between two cacheline addresses
  - Last 4 deltas, …

What is Action?

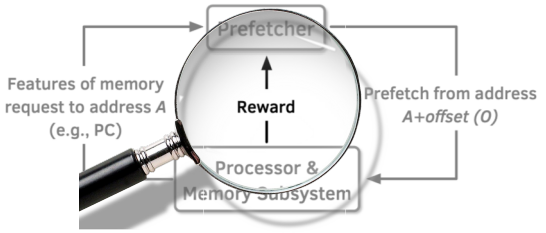
Given a demand access to address A
the action is to **select prefetch offset "O"**



- **Action-space**: 127 actions in the range [-63, +63]
  - For a machine with 4KB page and 64B cacheline
- Upper and lower limits ensure prefetches do not cross **physical page boundary**
- A **zero offset** means **no prefetch** is generated
- We further **prune** action-space by design-space exploration

What is Reward?

- Defines the **objective** of Pythia

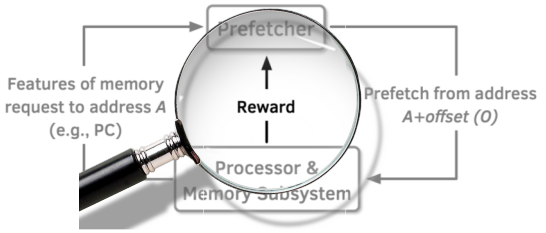


- Encapsulates two metrics:
  - **Prefetch usefulness** (e.g., accurate, late, out-of-page, …)
  - **System-level feedback** (e.g., mem. b/w usage, cache pollution, energy, …)

- We demonstrate Pythia with **memory bandwidth usage** as the system-level feedback in the paper

What is Reward?

- **Seven** distinct reward levels
  - *Accurate and timely* (R_{AT})
  - *Accurate but late* (R_{AL})
  - *Loss of coverage* (R_{CL})
  - *Inaccurate*
    - With low memory b/w usage (R_{IN}-L)
    - With high memory b/w usage (R_{IN}-H)
  - *No-prefetch*
    - With low memory b/w usage (R_{NP}-L)
    - With high memory b/w usage(R_{NP}-H)



- Values are set at design time via **automatic design-space exploration**
  - Can be **customized** further in silicon for higher performance

Basic Pythia Configuration

- Derived from **automatic design-space exploration**

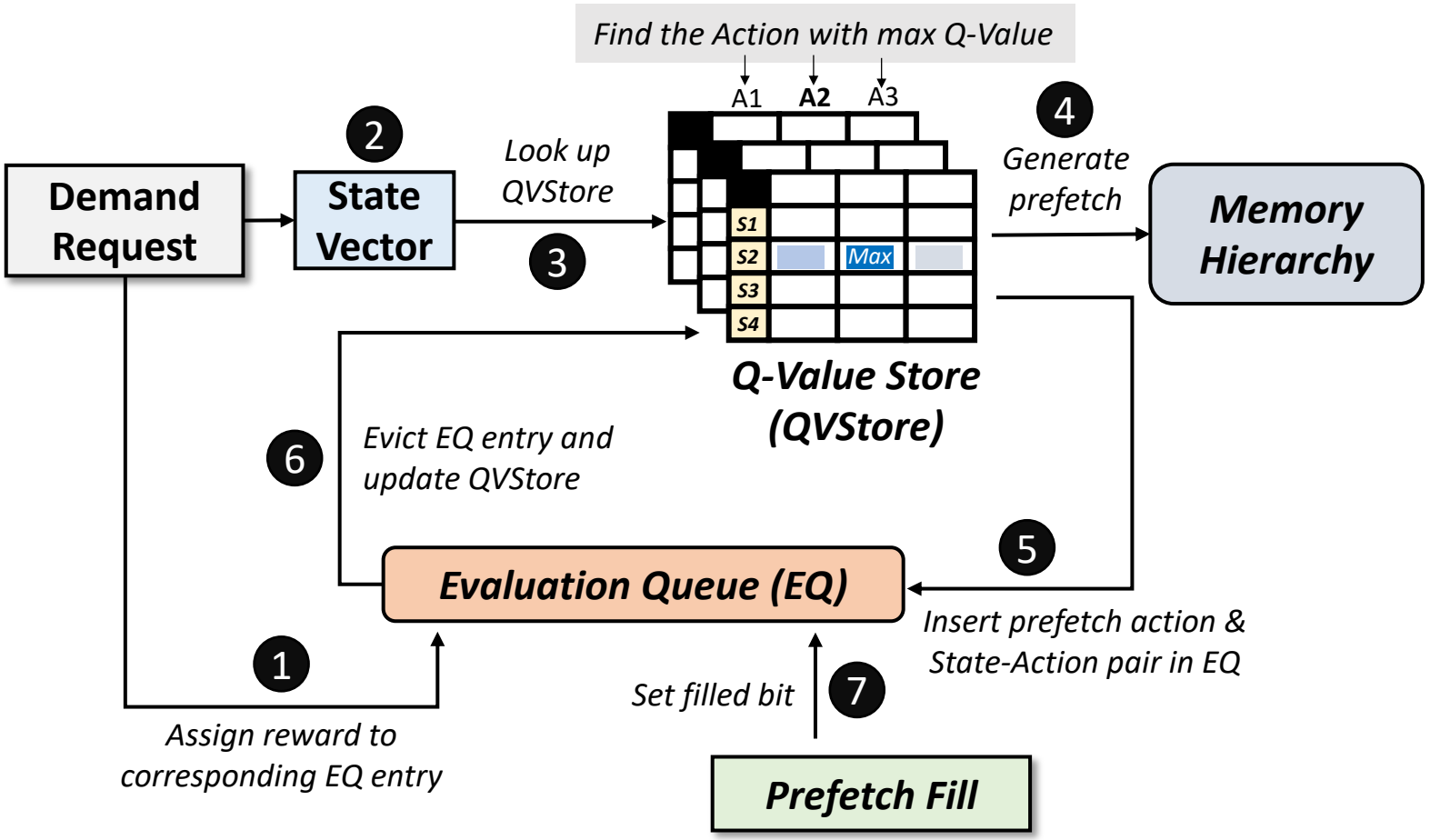
- **State:** 2 features
  - PC+Delta
  - Sequence of last-4 deltas

- **Actions:** 16 prefetch offsets
  - Ranging between -6 to +32. Including 0.

- **Rewards:**
  - R_{AT} = +20; R_{AL} = +12; R_{NP}-H=-2; R_{NP}-L=-4;
  - R_{IN}-H=-14; R_{IN}-L=-8; R_{CL}=-12

More Detailed Pythia Overview

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

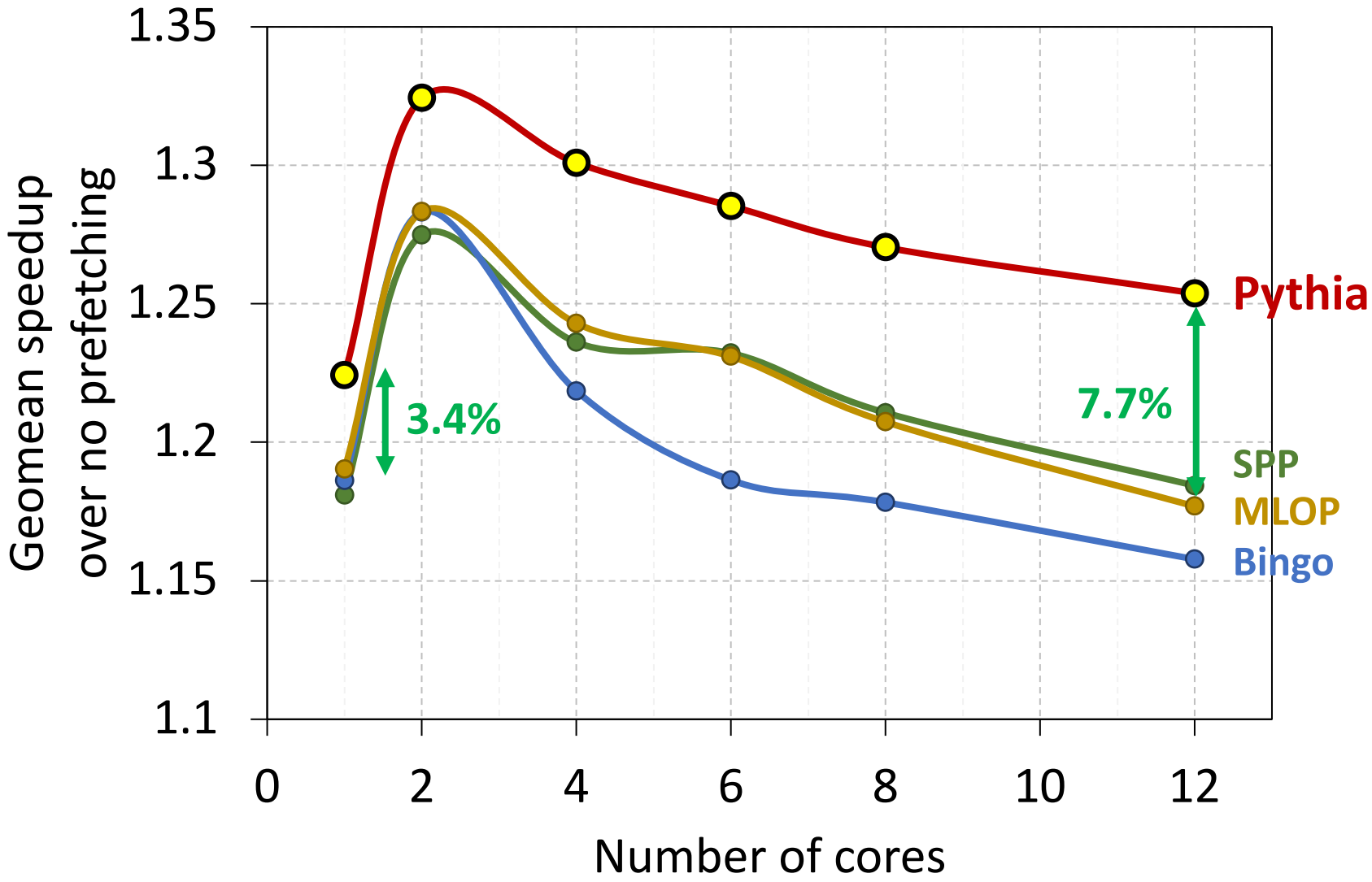


Simulation Methodology

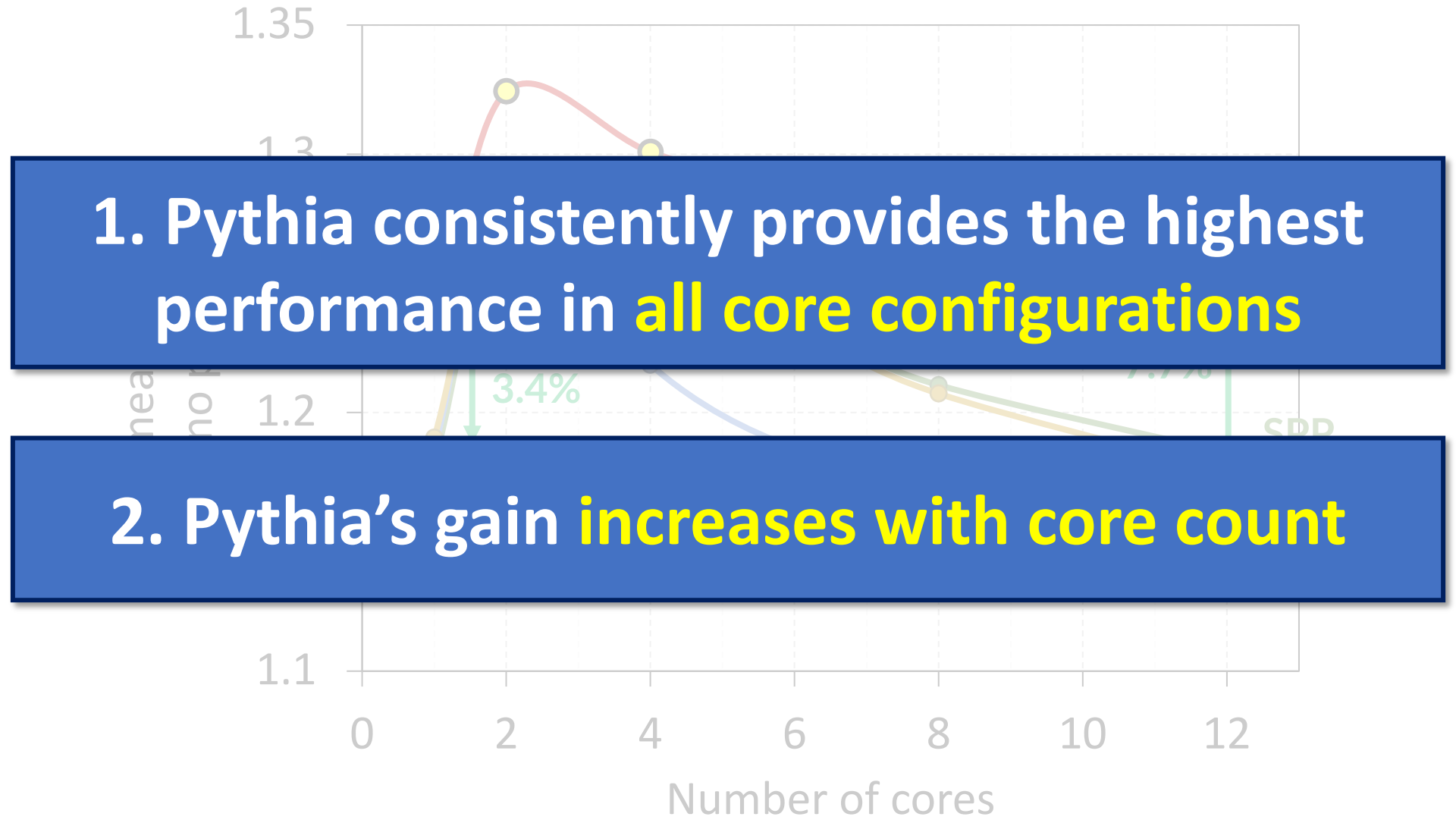
- **Champsim** [3] trace-driven simulator
- **150** single-core memory-intensive workload traces
  - SPEC CPU2006 and CPU2017
  - PARSEC 2.1
  - Ligra
  - Cloudsuite
- Homogeneous and heterogeneous multi-core mixes
- **Five** state-of-the-art prefetchers
  - SPP [Kim+, MICRO'16]
  - Bingo [Bakhshalipour+,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

[3] https://github.com/ChampSim/ChampSim

Performance with Varying Core Count



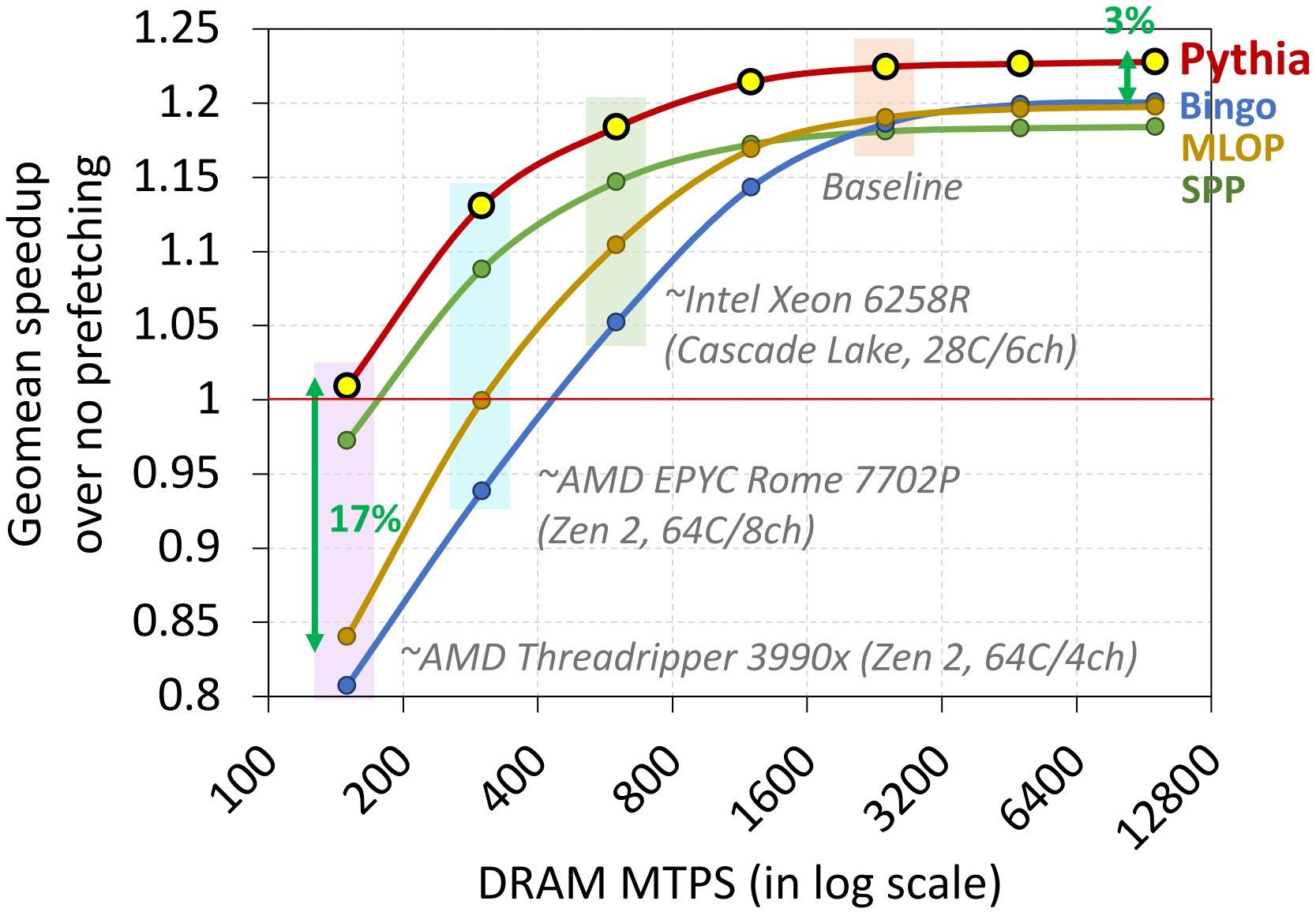
Performance with Varying Core Count



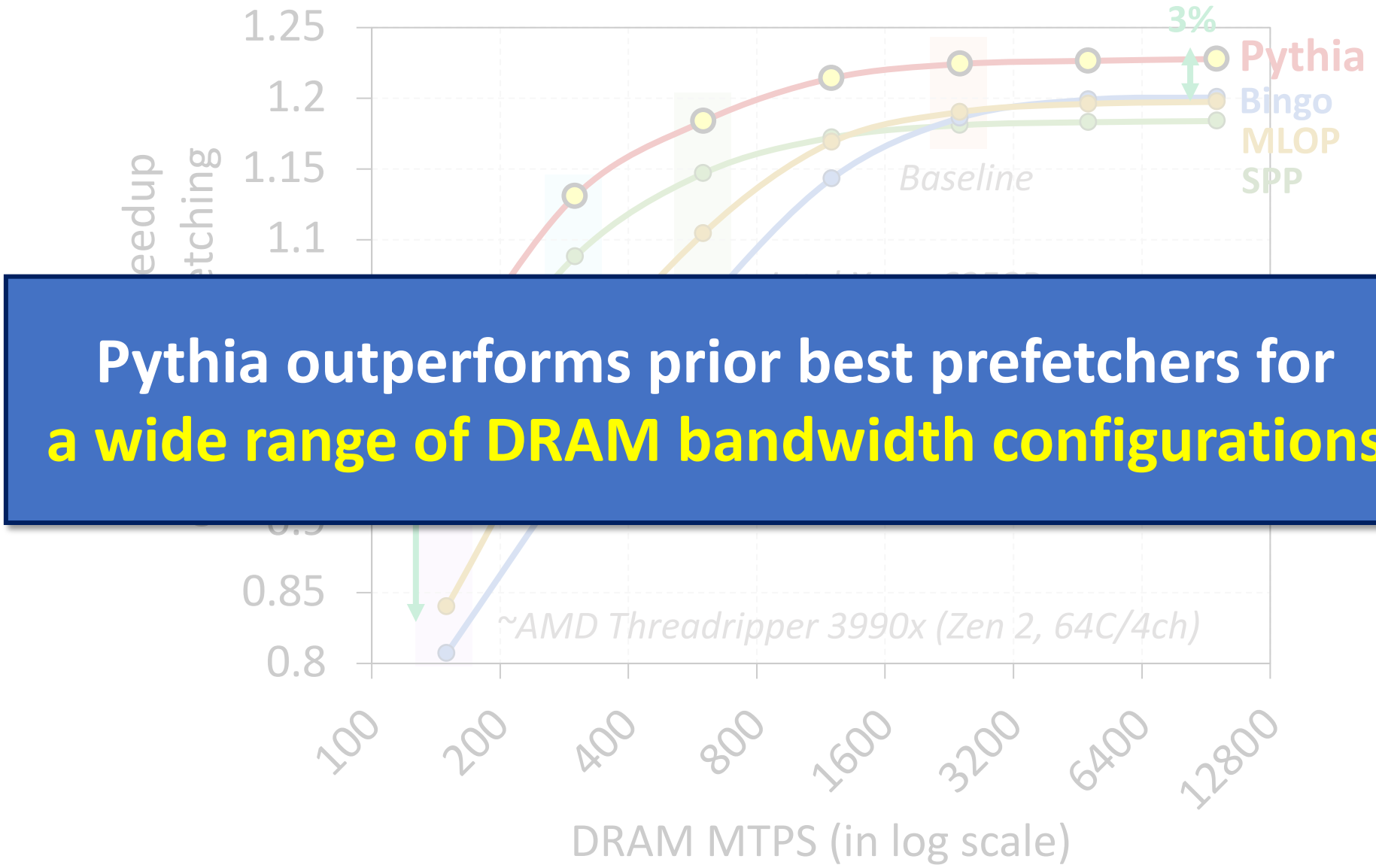
1. Pythia consistently provides the highest performance in **all core configurations**

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

Performance with Varying DRAM Bandwidth

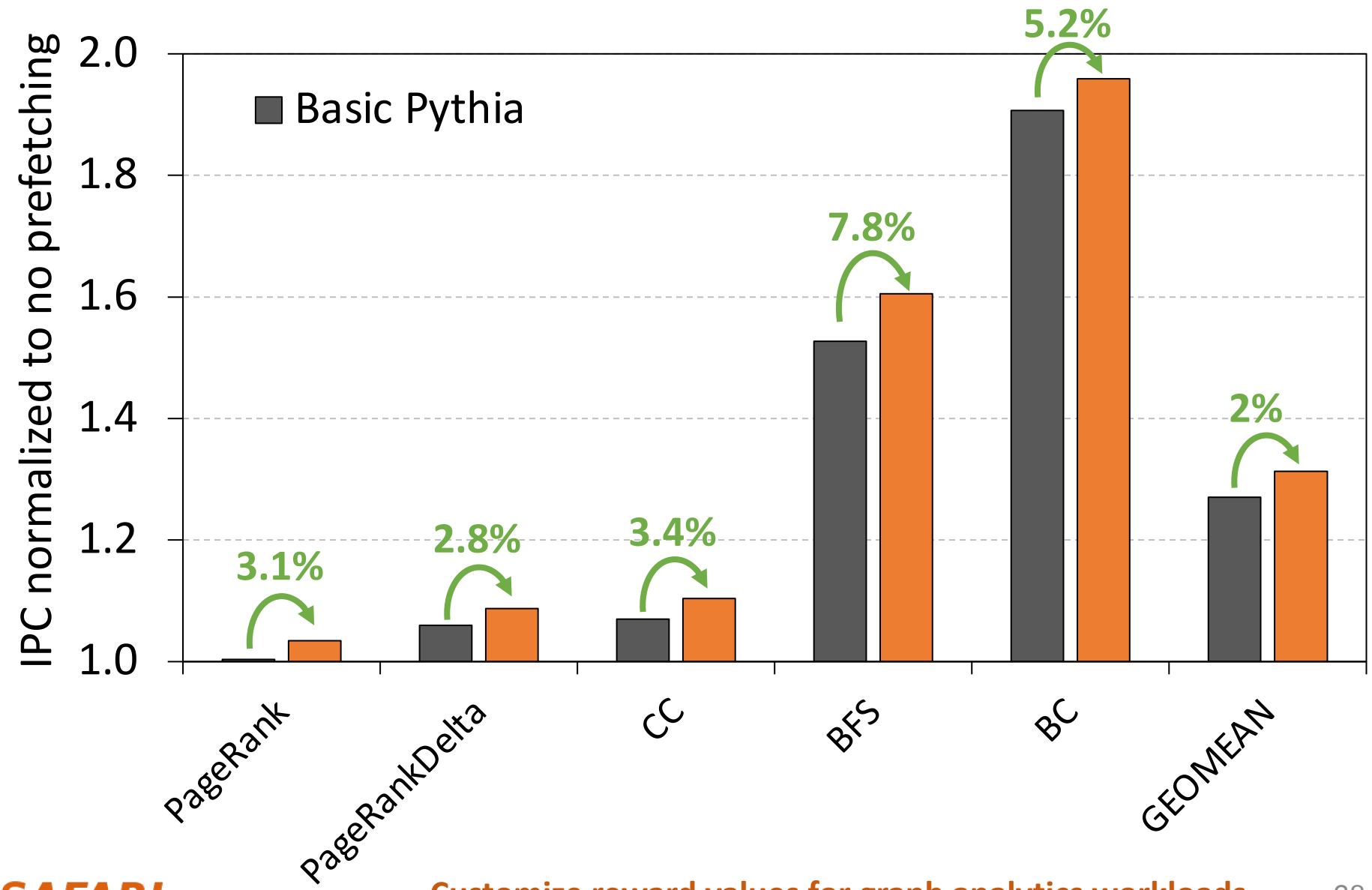


Performance with Varying DRAM Bandwidth



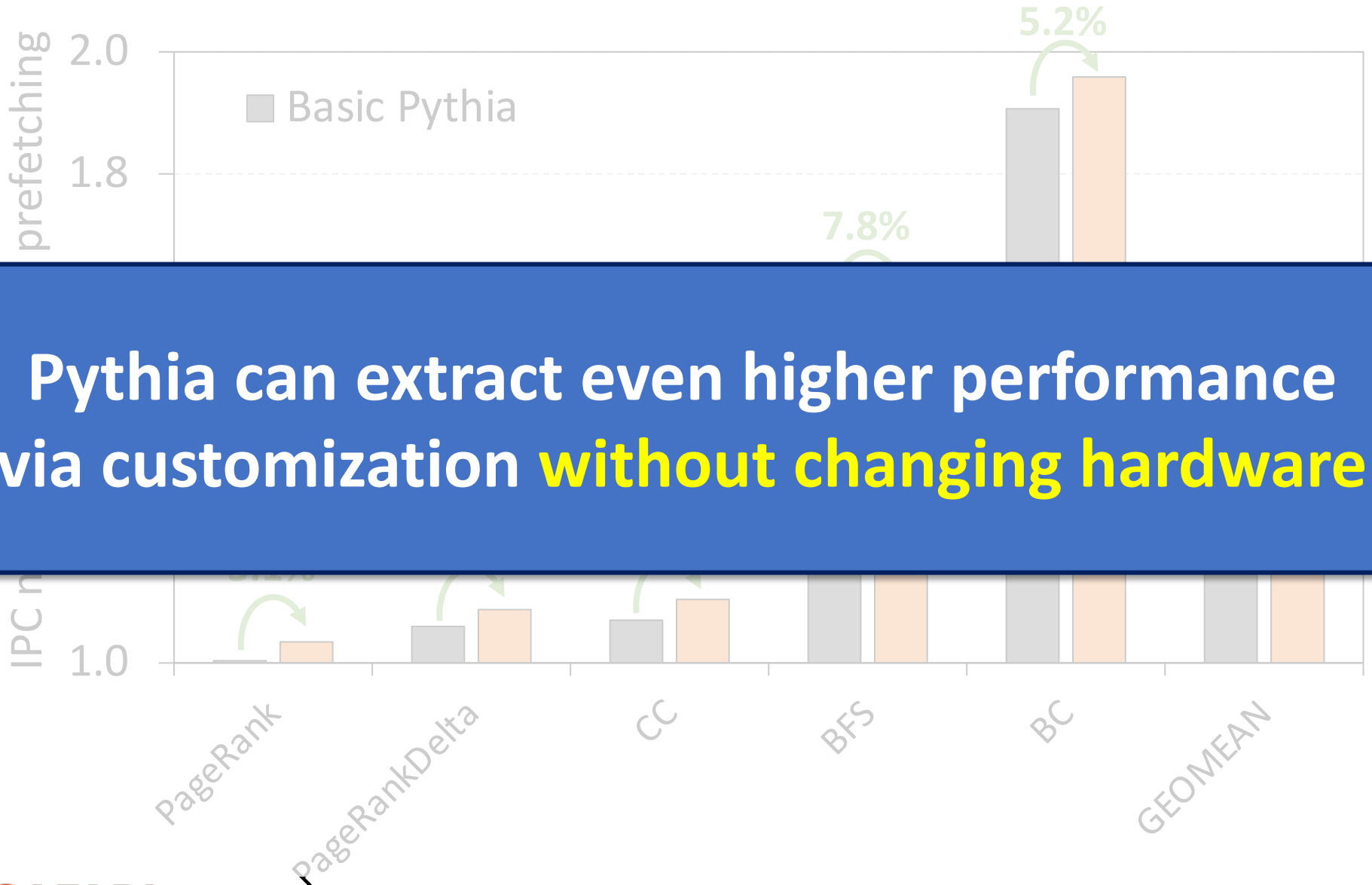
Pythia outperforms prior best prefetchers for a wide range of DRAM bandwidth configurations

Performance Improvement via Customization



Customize reward values for graph analytics workloads

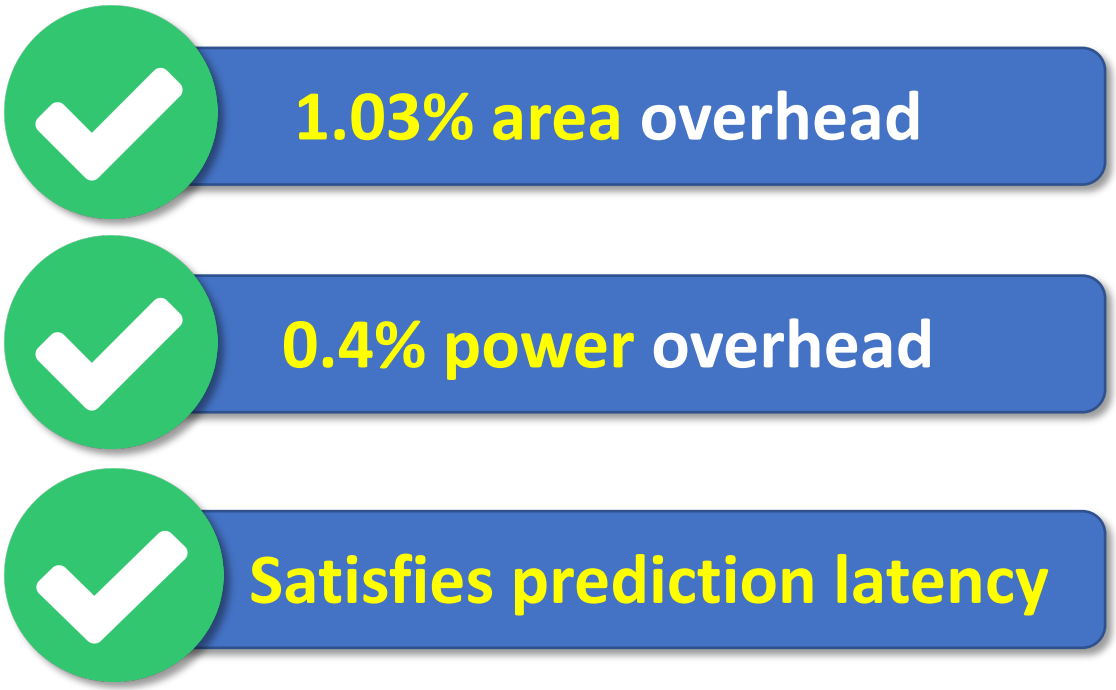
Performance Improvement via Customization



Pythia can extract even higher performance via customization without changing hardware

Pythia's Overhead

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



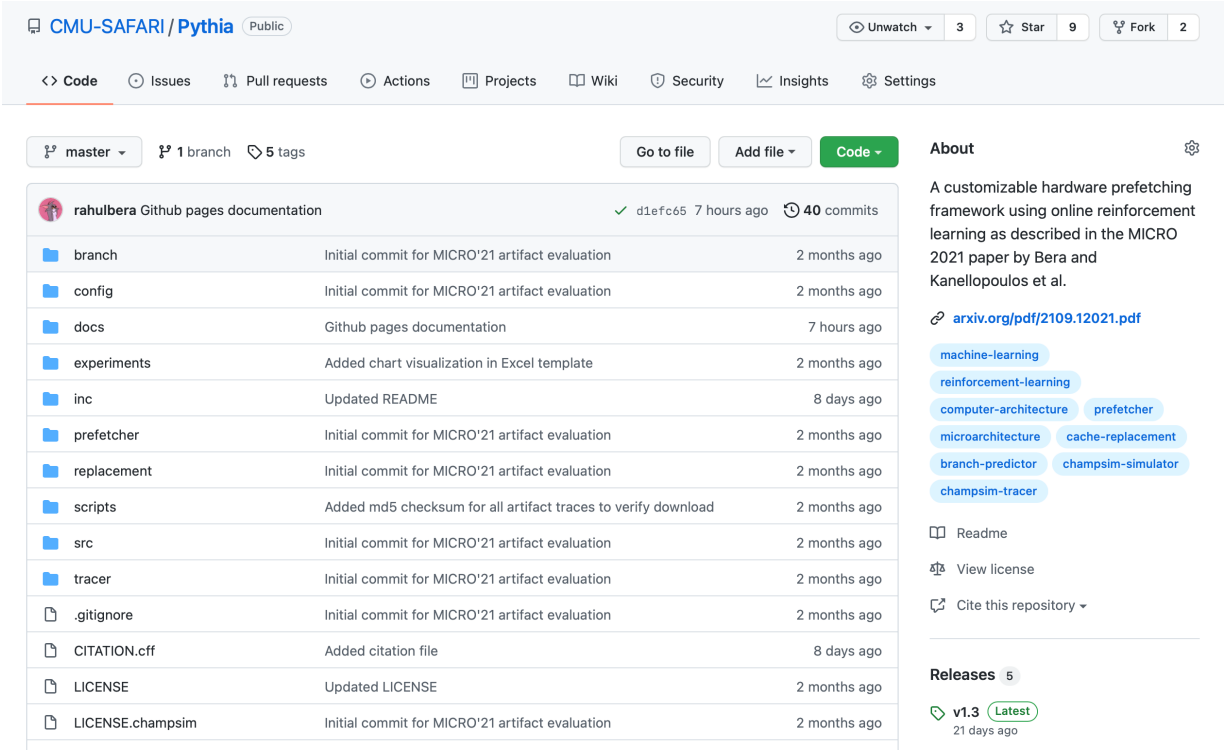
of a desktop-class 4-core Skylake processor (Xeon D2132IT, 60W)

[4] https://www.chisel-lang.org

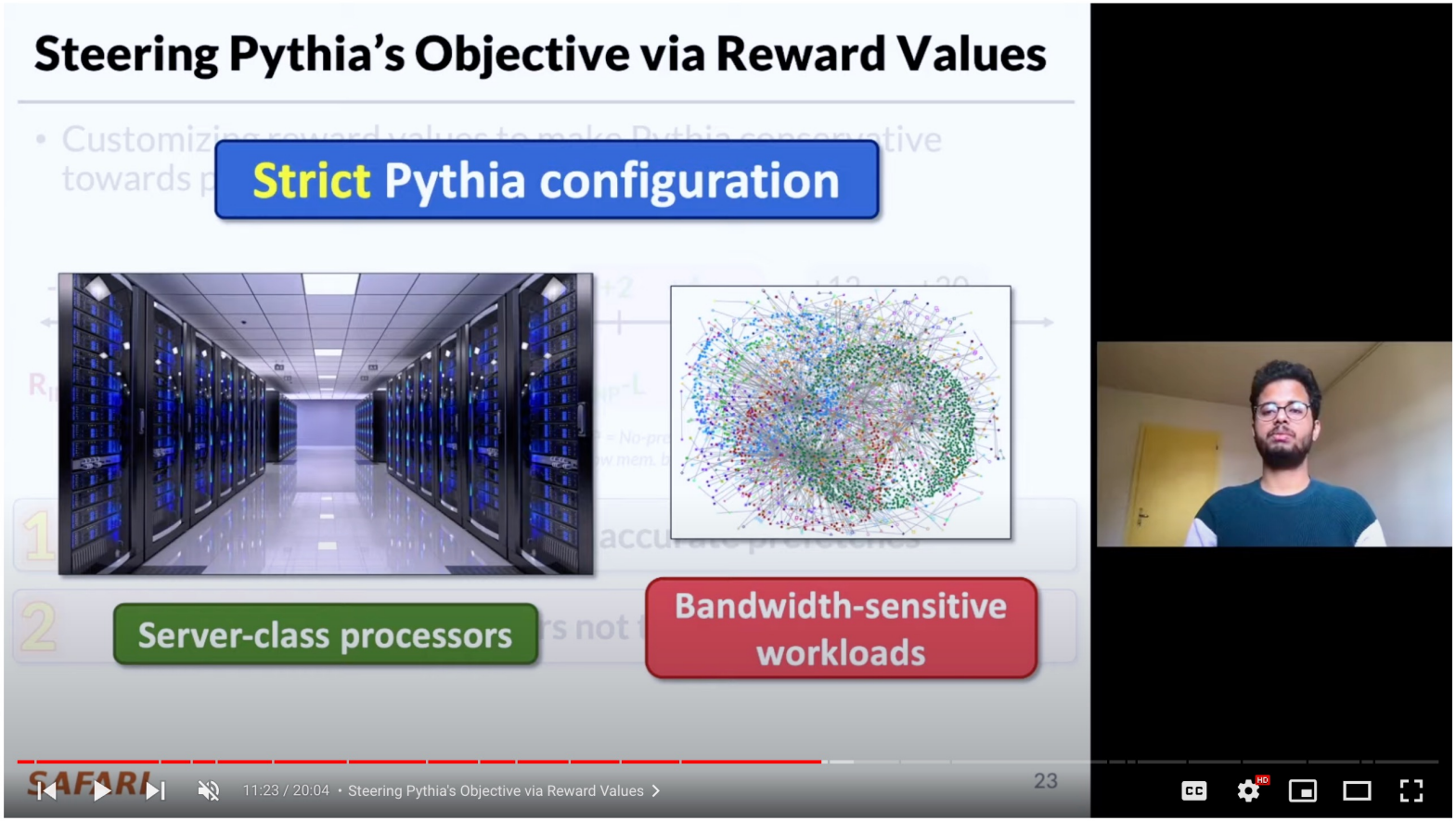
Pythia is Open Source

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

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



Pythia Talk Video



MICRO 2021 Conference Presentations

Pythia: A Customizable Prefetching Framework Using Reinforcement Learning - MICRO'21 Long Talk

Onur Mutlu Lectures
28.9K subscribers

Analytics | Edit video | 22 | Share | Download | Clip | Save

661 views 11 months ago
Talk: "Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning"
Full Conference Talk at MICRO 2021 by Rahul Bera

https://www.youtube.com/watch?v=6UMFRW3VFPo&list=PL5Q2soXY2Zi--0LrXSQ9sST3N0k0bXp51&index=8

A Lot More in the Pythia Paper

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

Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

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

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

Pythia

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

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

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

SAFARI Research Group
safari.ethz.ch

ETH zürich

intel

TU Delft

Hermes: Perceptron-Based Off-Chip Load Prediction

Learning-Based Off-Chip Load Predictors

- **Best Paper Award at MICRO 2022**

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

Rahul Bera[1] Konstantinos Kanellopoulos[1] Shankar Balachandran[2] David Novo[3]
Ataberk Olgun[1] Mohammad Sadrosadati[1] Onur Mutlu[1]

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

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

HERMES

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

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

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

Problem

Long-latency off-chip load requests

Often **stall** processor by
blocking instruction retirement from
Reorder Buffer (ROB)

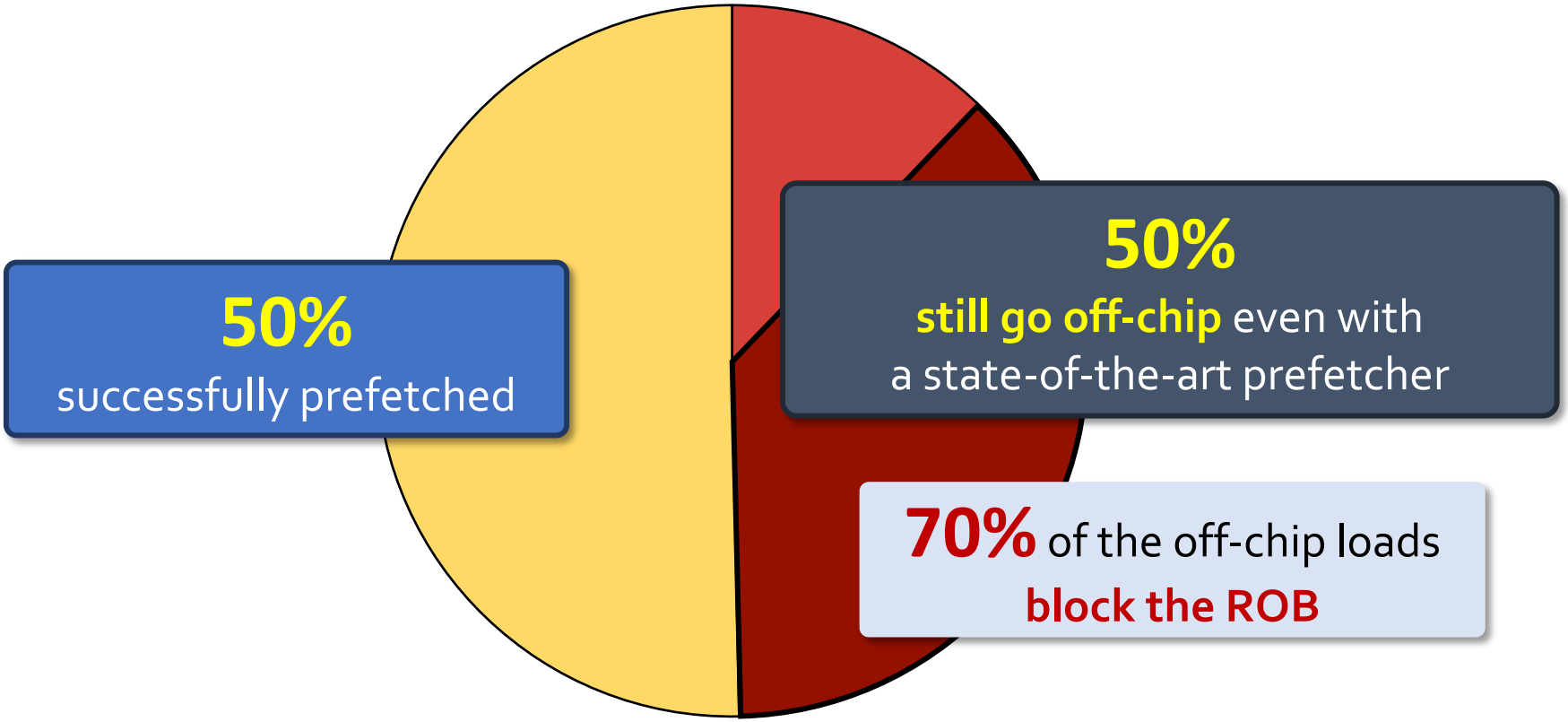
Limit performance

Traditional Solutions

1. Employ sophisticated **prefetchers**
2. **Increase size** of on-chip caches

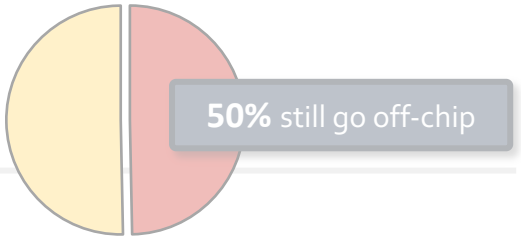
Key Observation 1

Many loads still go off-chip

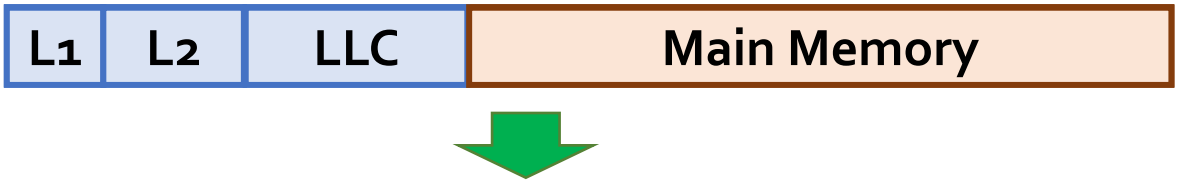


 # off-chip loads without any prefetcher

Key Observation 2



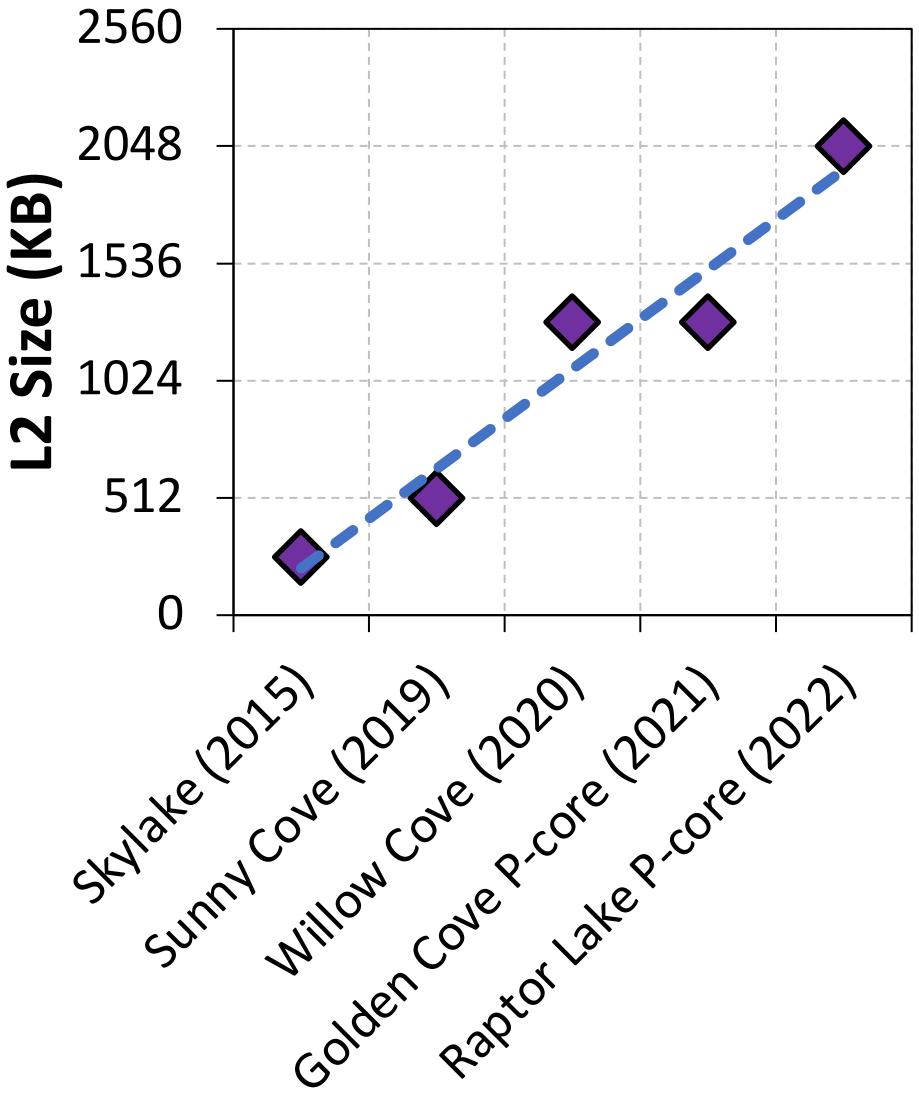
On-chip cache access latency
significantly contributes to off-chip load latency

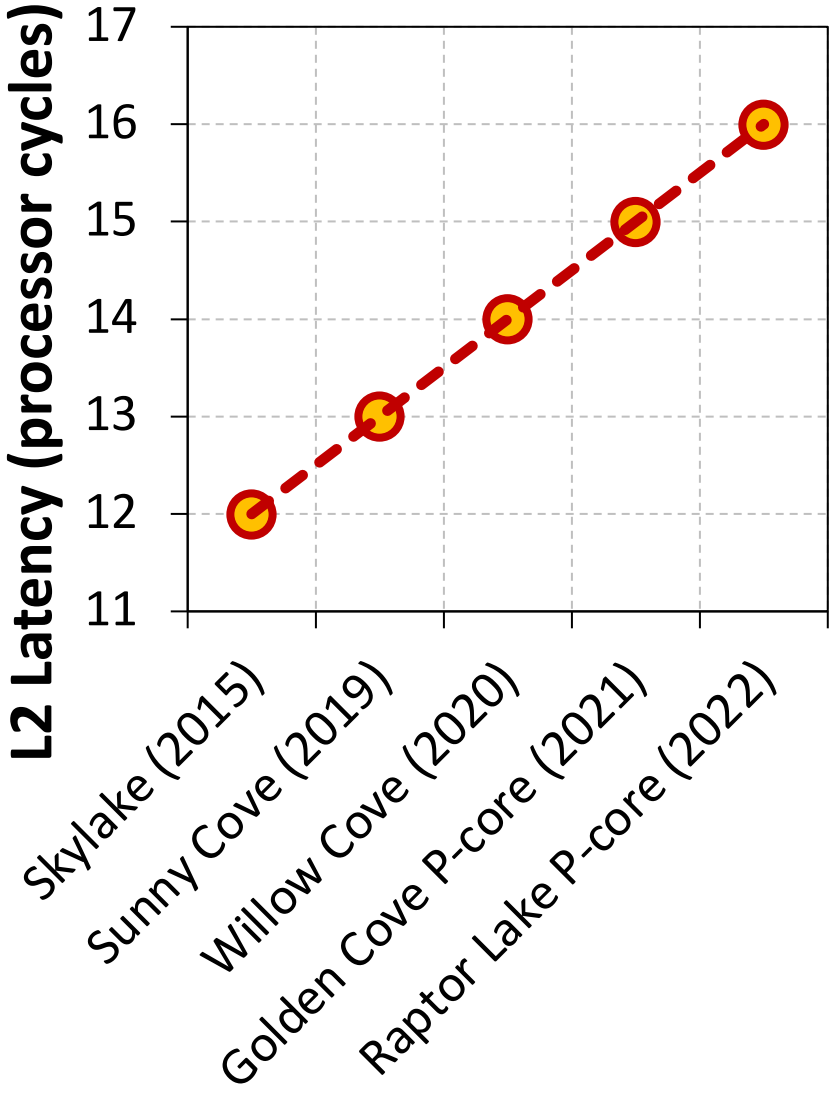


Saved cycles

40% of the **stalls** can be eliminated by **removing on-chip cache access latency from critical path**

Caches are Getting Bigger and Slower…





Our Goal

Improve processor performance
by **removing on-chip cache access latency**
from the **critical path of off-chip loads**

HERMES

Predicts which load requests
are likely to **go off-chip**

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

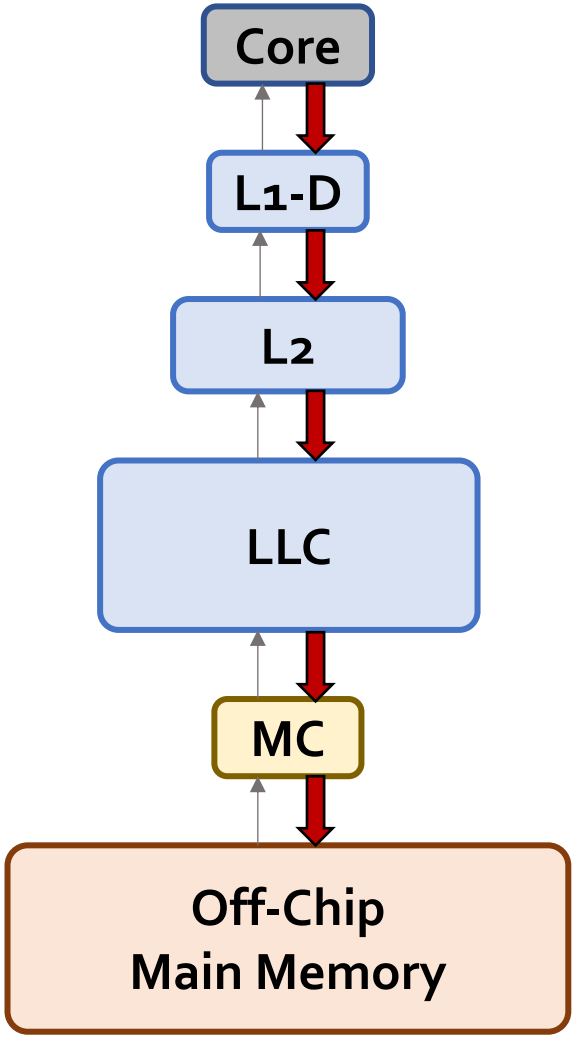
Hermes: Key Contribution

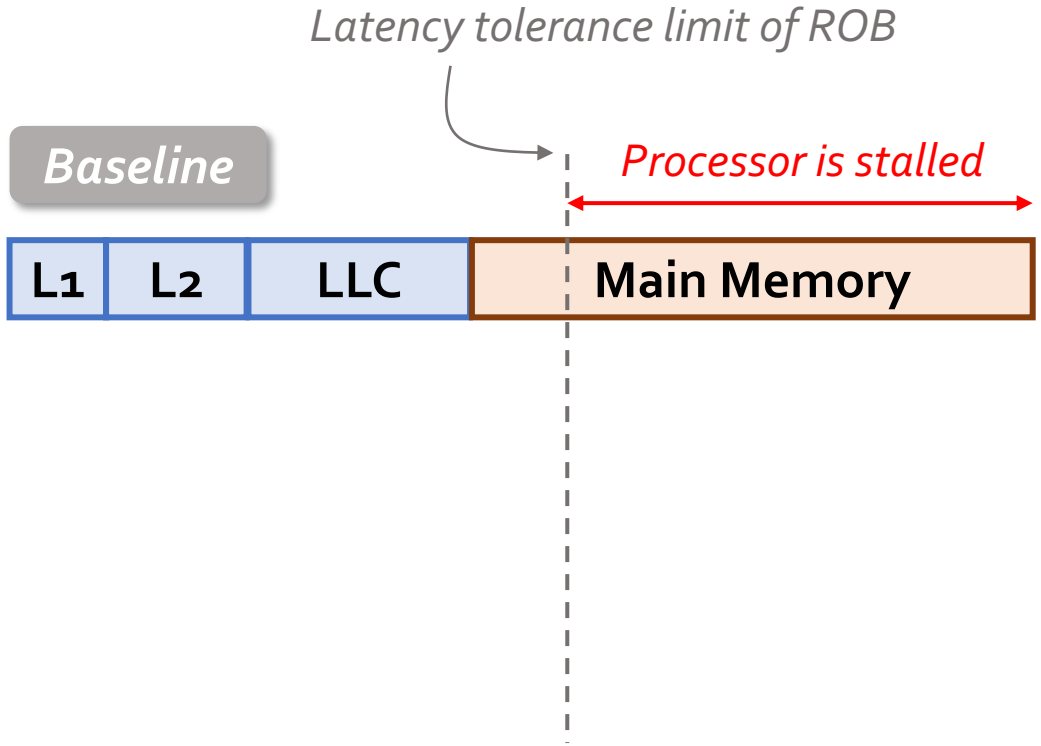
Hermes employs **the first perceptron-based** off-chip load predictor

That predicts which loads are likely to **go off-chip**

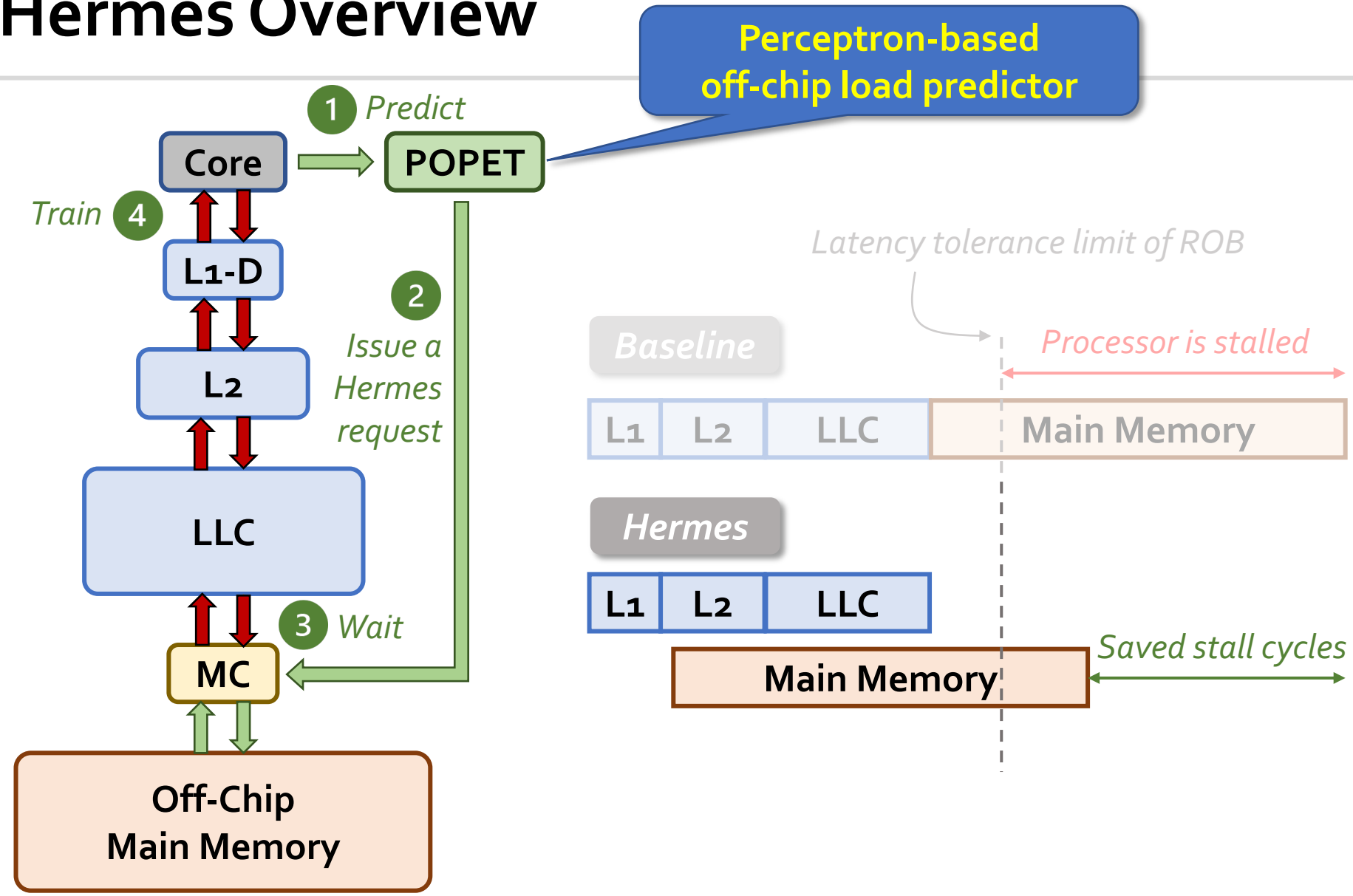
By **learning** from **multiple program context information**

Hermes Overview





Hermes Overview



Designing the Off-Chip Load Predictor

History-based prediction

HMP [Yoaz+, ISCA'99] for the L1-D cache

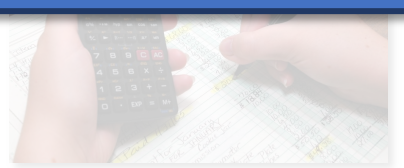
Using branch-predictor-like hybrid predictor:
Global, Gshare, and GSkew

POPET provides
both **higher accuracy** and **higher performance**
than predictors inspired from these previous works

- Metadata size increases with cache hierarchy size

May need to track **all** cache operations

- Gets complex depending on the cache hierarchy configuration (e.g., inclusivity, bypassing,...)

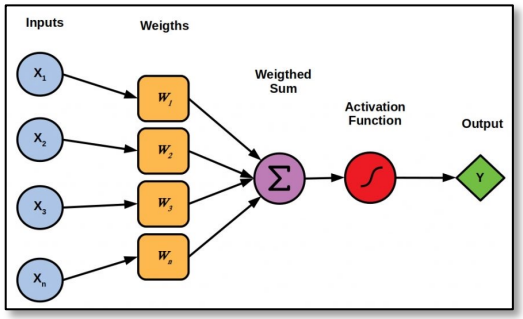


Learning from program behavior

Correlate different program features with off-chip loads

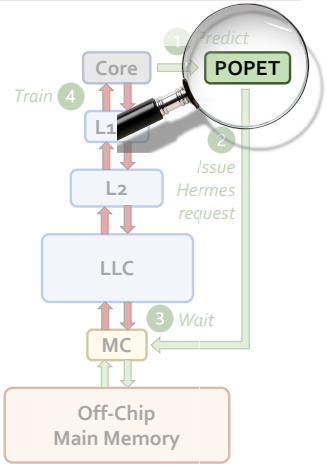
Low storage overhead

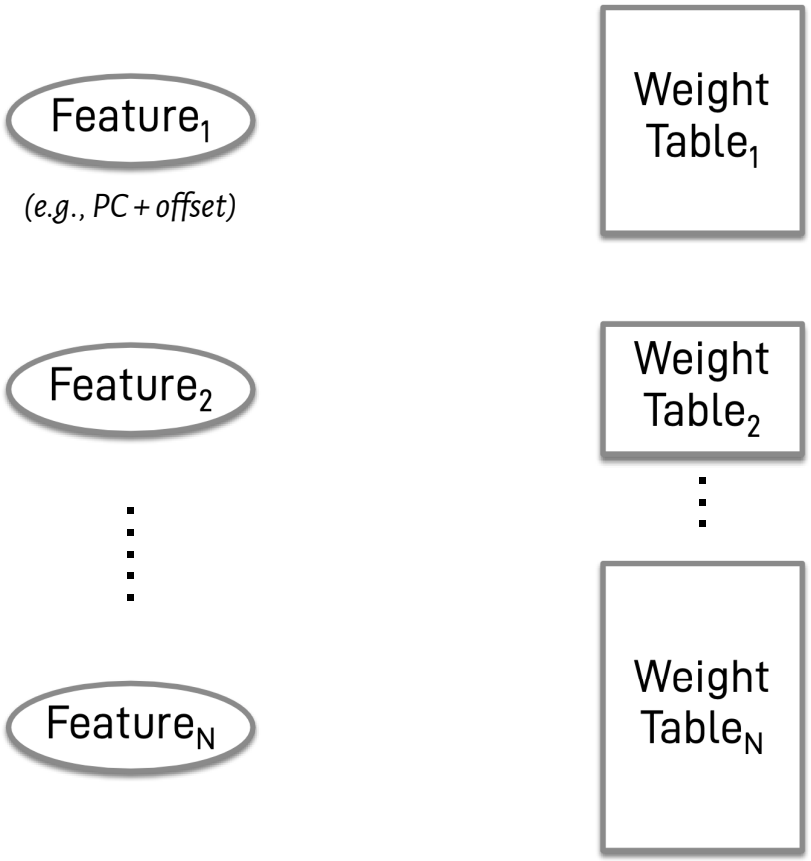
Low design complexity



POPET: Perceptron-Based Off-Chip Predictor

- Multi-feature hashed perceptron model[1]
  - Each feature has its own *weight table*
    - Stores correlation between feature value and off-chip prediction

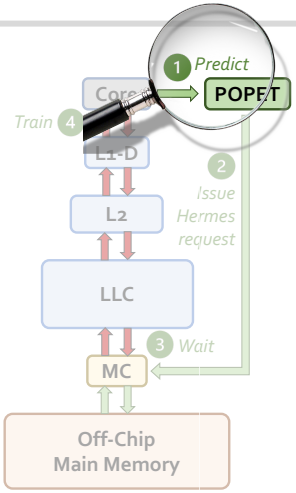


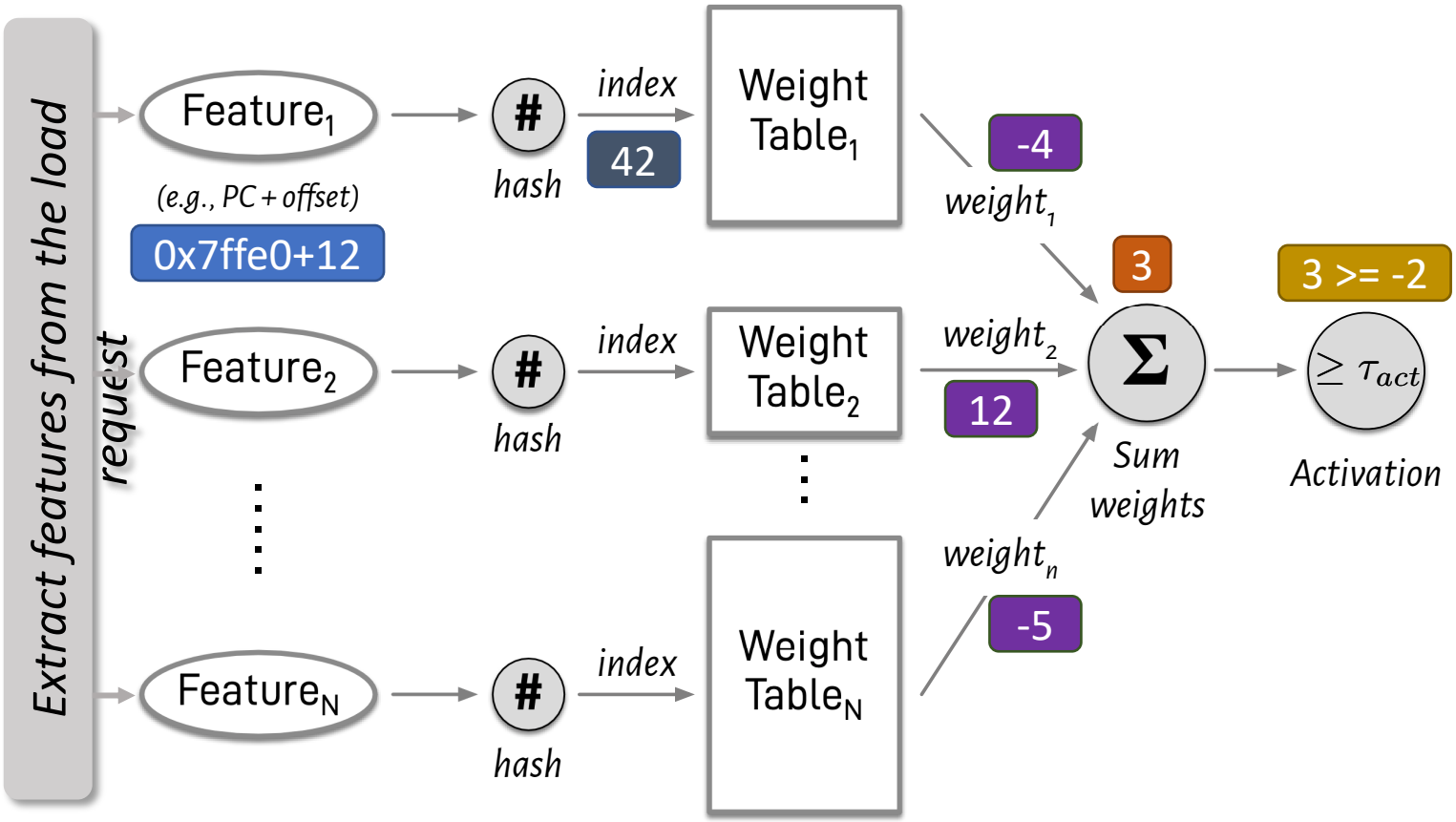


[1] D. Tarjan and K. Skadron, "Merging Path and Gshare Indexing in Perceptron Branch Prediction," TACO, 2005

Predicting using POPET

- Uses simple **table lookups**, **addition**, and **comparison**

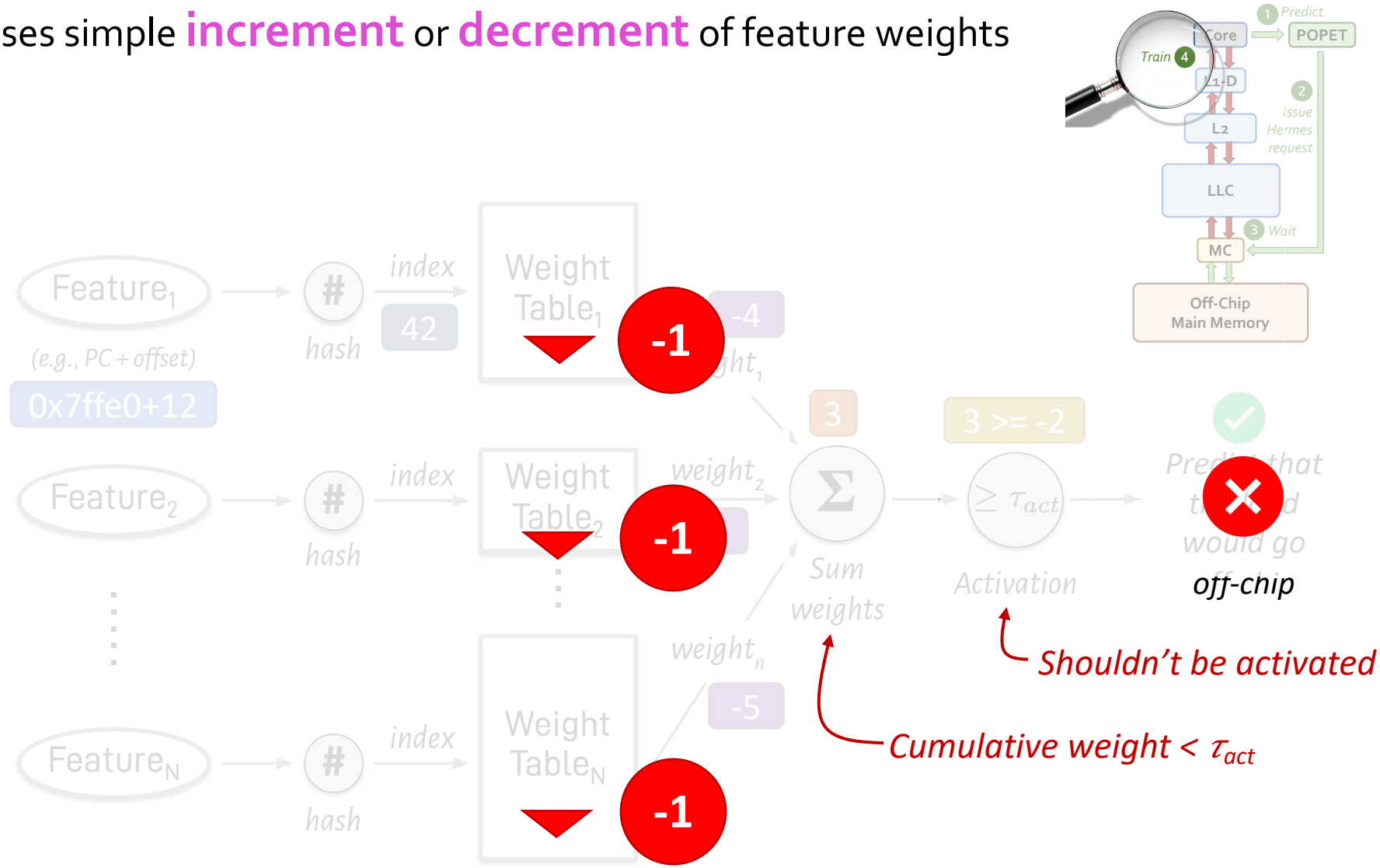




Predict that the load would go off-chip

Training POPET

- Uses simple **increment** or **decrement** of feature weights



Evaluation

Simulation Methodology

- **ChampSim** trace driven simulator
- **110 single-core** memory-intensive traces
  - SPEC CPU 2006 and 2017
  - PARSEC 2.1
  - Ligra
  - Real-world applications
- **220 eight-core** memory-intensive trace mixes

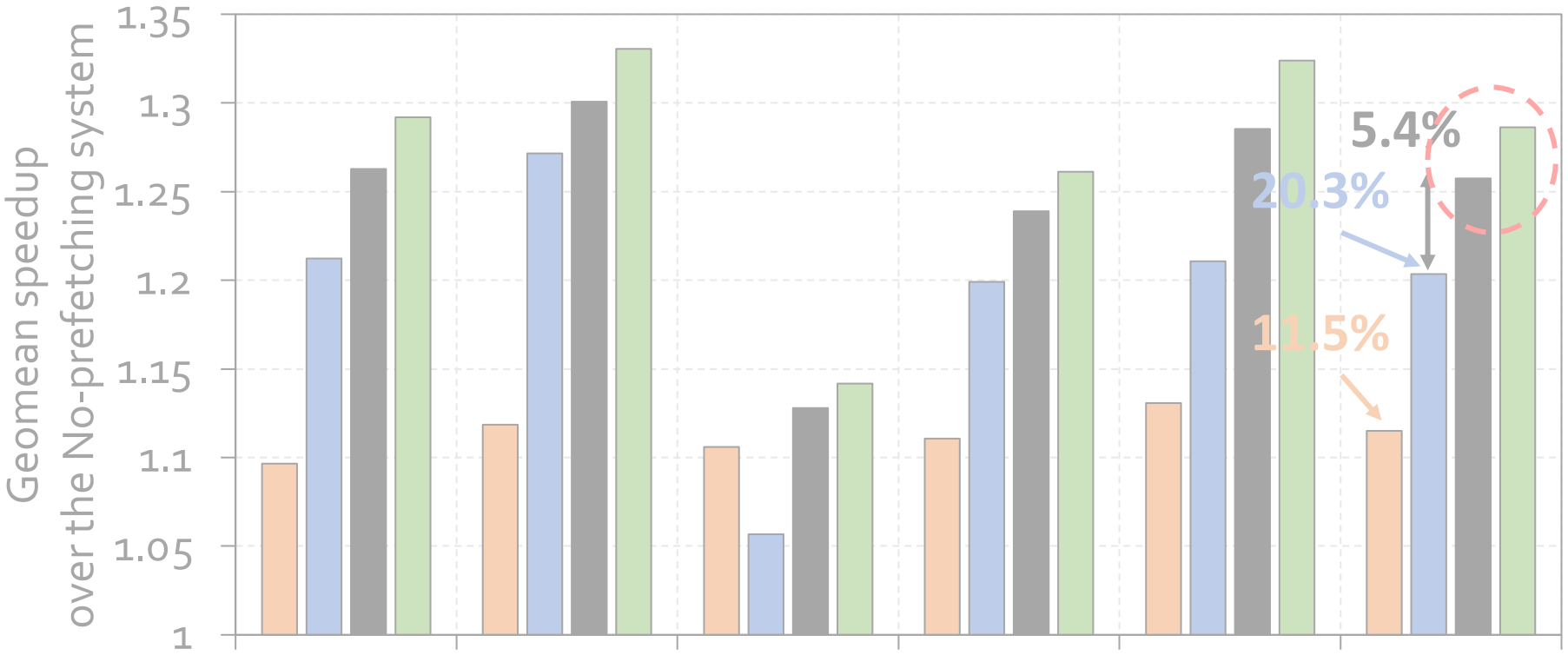
LLC Prefetchers

- Pythia [Bera+, MICRO'21]
- Bingo [Bakshalipour+, HPCA'19]
- MLOP [Shakerinava+, 3rd Prefetching Championship'19]
- SPP + Perceptron filter [Bhatia+, ISCA'20]
- SMS [Somogyi+, ISCA'06]

Off-Chip Predictors

- **History-based**: **HMP** [Yoaz+, ISCA'99]
- **Tracking-based**: Address Tag-Tracking based Predictor (**TTP**)
- **Ideal Off-chip Predictor**

Single-Core Performance Improvement

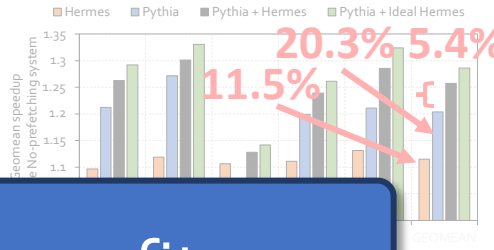


Hermes alone provides nearly

Hermes provides nearly **90%** performance benefit of **Ideal Hermes** that has an **ideal off-chip load predictor**

with only **1/5th** storage overhead

Increase in Main Memory Requests

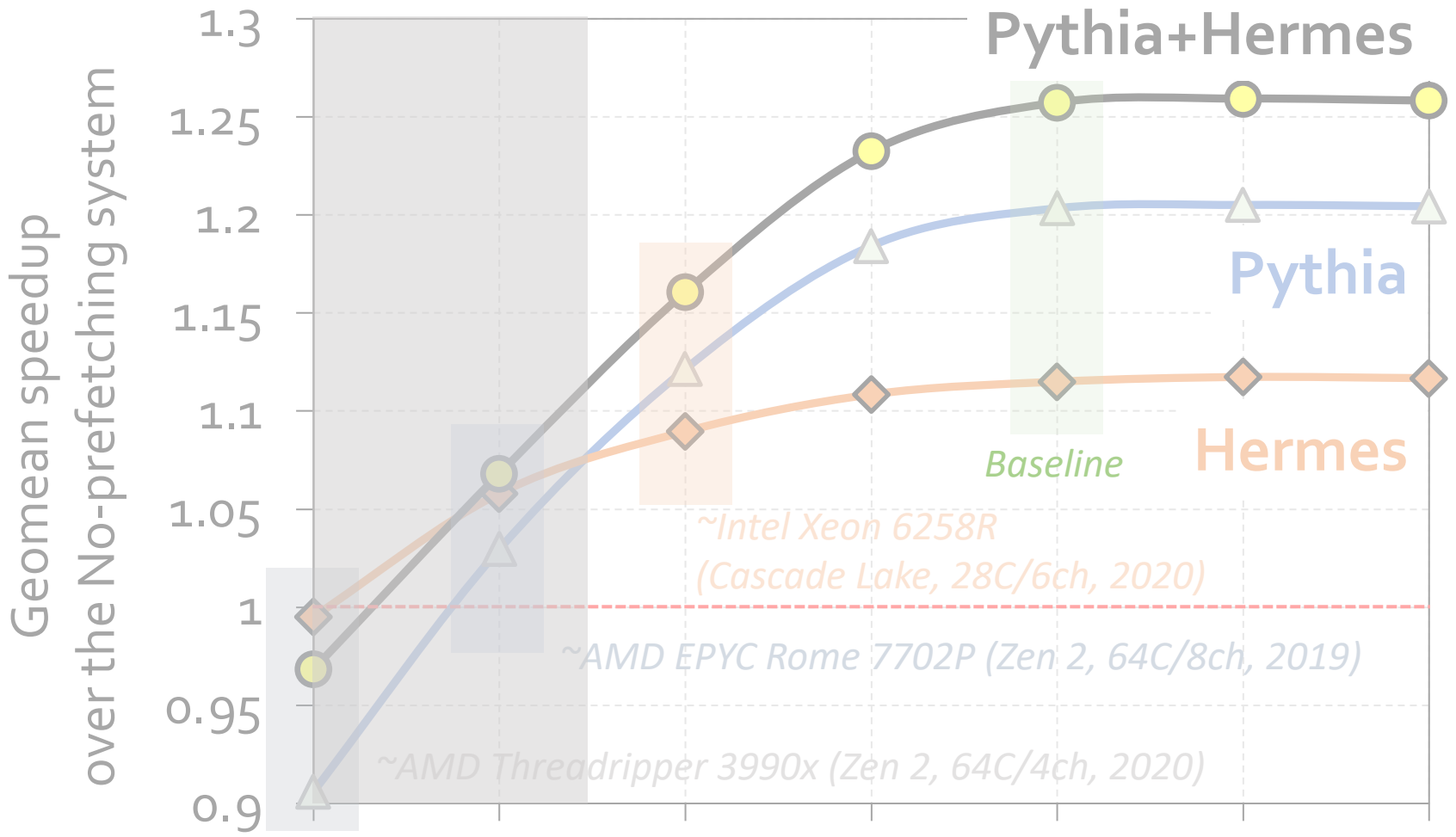


For **every 1% performance** benefit,
increase in main memory requests

Pythia	**2%**
Hermes on top of Pythia	**1%**
Hermes alone	**0.5%**

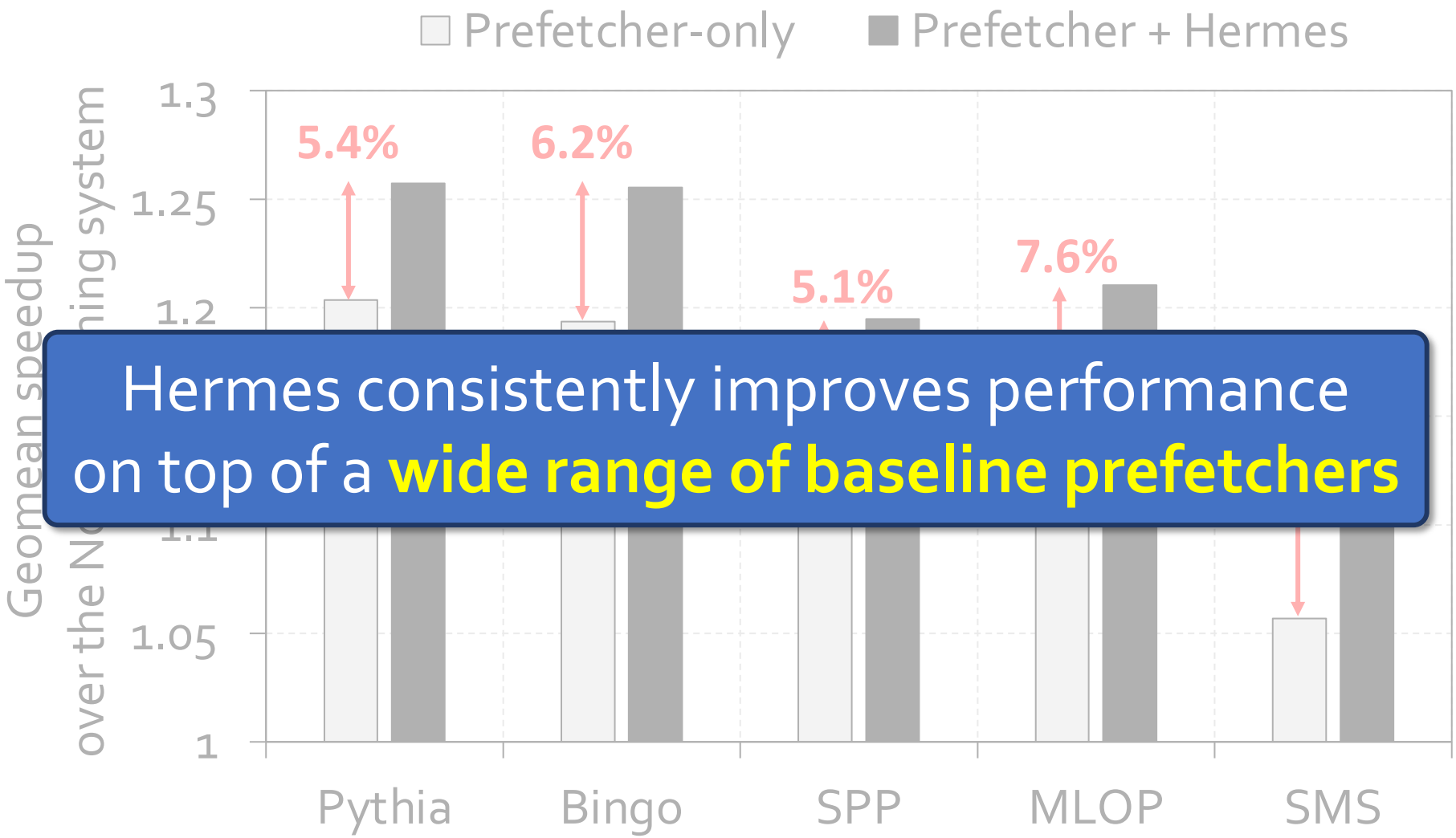
Hermes is more **bandwidth-efficient**
than even an efficient prefetcher like Pythia

Performance with Varying Memory Bandwidth



Hermes+Pythia outperforms Pythia
across all bandwidth configurations

Performance with Varying Baseline Prefetcher



Hermes consistently improves performance on top of a **wide range of baseline prefetchers**

Overhead of Hermes

4 KB storage overhead

1.5% power overhead*

**On top of an Intel Alder Lake-like performance-core [2] configuration*

[2] https://www.anandtech.com/show/16881/a-deep-dive-into-intels-alder-lake-microarchitectures/3

A Lot More in the Hermes Paper

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

Rahul Bera[1] Konstantinos Kanellopoulos[1] Shankar Balachandran[2] David Novo[3]
Ataberk Olgun[1] Mohammad Sadrosadati[1] Onur Mutlu[1]

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

Long-latency load requests continue to limit the performance of modern high-performance processors. To increase the latency tolerance of a processor, architects have primarily relied on two key techniques: sophisticated data prefetchers and large on-chip caches. In this work, we show that: (1) even a sophisticated state-of-the-art prefetcher can only predict half of the off-chip load requests on average across a wide range of workloads, and (2) due to the increasing size and complexity of on-chip caches, a large fraction of the latency of an off-chip load request is spent accessing the on-chip cache hierarchy to solely determine that it needs to go off-chip.

The goal of this work is to accelerate off-chip load requests by removing the on-chip cache access latency from their critical path. To this end, we propose a new technique called Hermes, whose key idea is to: (1) accurately predict which load requests

off-chip main memory (i.e., an *off-chip load*) often stalls the processor core by blocking the instruction retirement from the reorder buffer (ROB), thus limiting the core's performance [88, 91, 92]. To increase the latency tolerance of a core, computer architects primarily rely on two key techniques. First, they employ increasingly sophisticated hardware prefetchers that can learn complex memory address patterns and fetch data required by future load requests before the core demands them [28, 32, 33, 35, 75]. Second, they significantly scale up the size of the on-chip cache hierarchy with each new generation of processors [10, 11, 16].

Key problem. Despite recent advances in processor core design, we observe two key trends in new processor designs that leave a significant opportunity for performance improvement on the table. First, even a sophisticated state-of-the-art

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

A New Approach to Latency Reduction

Hermes advocates for **off-chip load prediction**,
a **different** form of speculation than
load address prediction employed by prefetchers

Off-chip load prediction can be applied **by itself**
or **combined with load address prediction**
to provide performance improvement

Hermes: Summary

Hermes employs **the first perceptron-based** off-chip load predictor

High accuracy
(77%)

High coverage
(74%)

Low storage overhead
(4KB/core)

High performance improvement
over best prior baseline
(5.4%)

High performance per bandwidth

Hermes is Open Source

All workload traces

13 prefetchers

- Stride [Fu+, MICRO'92]
- Streamer [Chen and Baer, IEEE TC'95]
- SMS [Somogyi+, ISCA'06]
- AMPM [Ishii+, ICS'09]
- Sandbox [Pugsley+, HPCA'14]
- BOP [Michaud, HPCA'16]
- SPP [Kim+, MICRO'16]
- Bingo [Bakshalipour+, HPCA'19]
- SPP+PPF [Bhatia+, ISCA'19]
- DSPatch [Bera+, MICRO'19]
- MLOP [Shakerinava+, DPC-3'19]
- IPCP [Pakalapati+, ISCA'20]
- Pythia [Bera+, MICRO'21]

9 off-chip predictors

Predictor type	Description
Base	Always NO
Basic	Simple confidence counter-based threshold
Random	Random Hit-miss predictor with a given positive probability
HMP-Local	Hit-miss predictor [Yoaz+, ISCA'99] with local prediction
HMP-GShare	Hit-miss predictor with GShare prediction
HMP-GSkew	Hit-miss predictor with GSkew prediction
HMP-Ensemble	Hit-miss predictor with all three types combined
TTP	Tag-tracking based predictor
Perc	Perceptron-based OCP used in this paper

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

Easy To Define Your Own Off-Chip Predictor

- Just extend the OffchipPredBase class

```
8   class OffchipPredBase
9   {
10  public:
11      uint32_t cpu;
12      string type;
13      uint64_t seed;
14      uint8_t dram_bw; // current DRAM bandwidth bucket
15
16      OffchipPredBase(uint32_t _cpu, string _type, uint64_t _seed) : cpu(_cpu), type(_type), seed(_seed)
17      {
18          srand(seed);
19          dram_bw = 0;
20      }
21      ~OffchipPredBase() {}
22      void update_dram_bw(uint8_t _dram_bw) { dram_bw = _dram_bw; }
23
24      virtual void print_config();
25      virtual void dump_stats();
26      virtual void reset_stats();
27      virtual void train(ooo_model_instr *arch_instr, uint32_t data_index, LSQ_ENTRY *lq_entry);
28      virtual bool predict(ooo_model_instr *arch_instr, uint32_t data_index, LSQ_ENTRY *lq_entry);
29  };
30
31  #endif /* OFFCHIP_PRED_BASE_H */
32
```

Easy To Define Your Own Off-Chip Predictor

- Define your own `train()` and `predict()` functions

```
19  void OffchipPredBase::train(ooo_model_instr *arch_instr, uint32_t data_index, LSQ_ENTRY *lq_entry)
20  {
21      // nothing to train
22  }
23
24  bool OffchipPredBase::predict(ooo_model_instr *arch_instr, uint32_t data_index, LSQ_ENTRY *lq_entry)
25  {
26      // predict randomly
27      // return (rand() % 2) ? true : false;
28      return false;
29  }
```

- Get statistics like accuracy (stat name *precision*) and coverage (stat name *recall*) out of the box

```
Core_0_offchip_pred_true_pos 2358716
Core_0_offchip_pred_false_pos 276883
Core_0_offchip_pred_false_neg 132145
Core_0_offchip_pred_precision 89.49
Core_0_offchip_pred_recall 94.69
```

Off-Chip Prediction Can Further Enable…

Prioritizing loads that are likely go off-chip in **cache queues** and **on-chip network routing**

Better instruction scheduling of data-dependent instructions

Other ideas to improve **performance** and **fairness** in multi-core system design…

Learning-Based Off-Chip Load Predictors

- **Best Paper Award at MICRO 2022**

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

Rahul Bera[1] Konstantinos Kanellopoulos[1] Shankar Balachandran[2] David Novo[3]
Ataberk Olgun[1] Mohammad Sadrosadati[1] Onur Mutlu[1]

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

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

HERMES

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

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

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

SAFARI Research Group
safari.ethz.ch

ETH zürich

intel

LIRMM

Two Major Thrusts

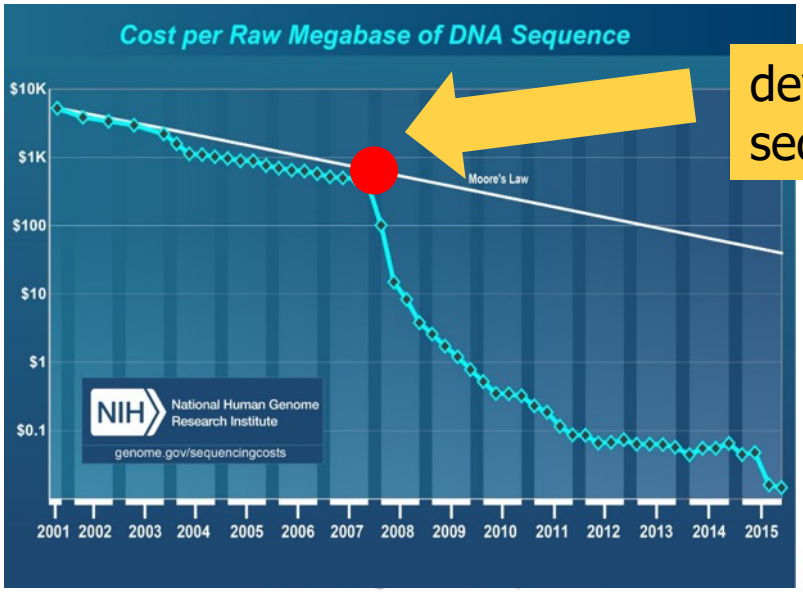
1. (Memory) system design for AI/ML workloads/accelerators

2. AI/ML techniques for improving (memory) system designs

Challenge and Opportunity for Future

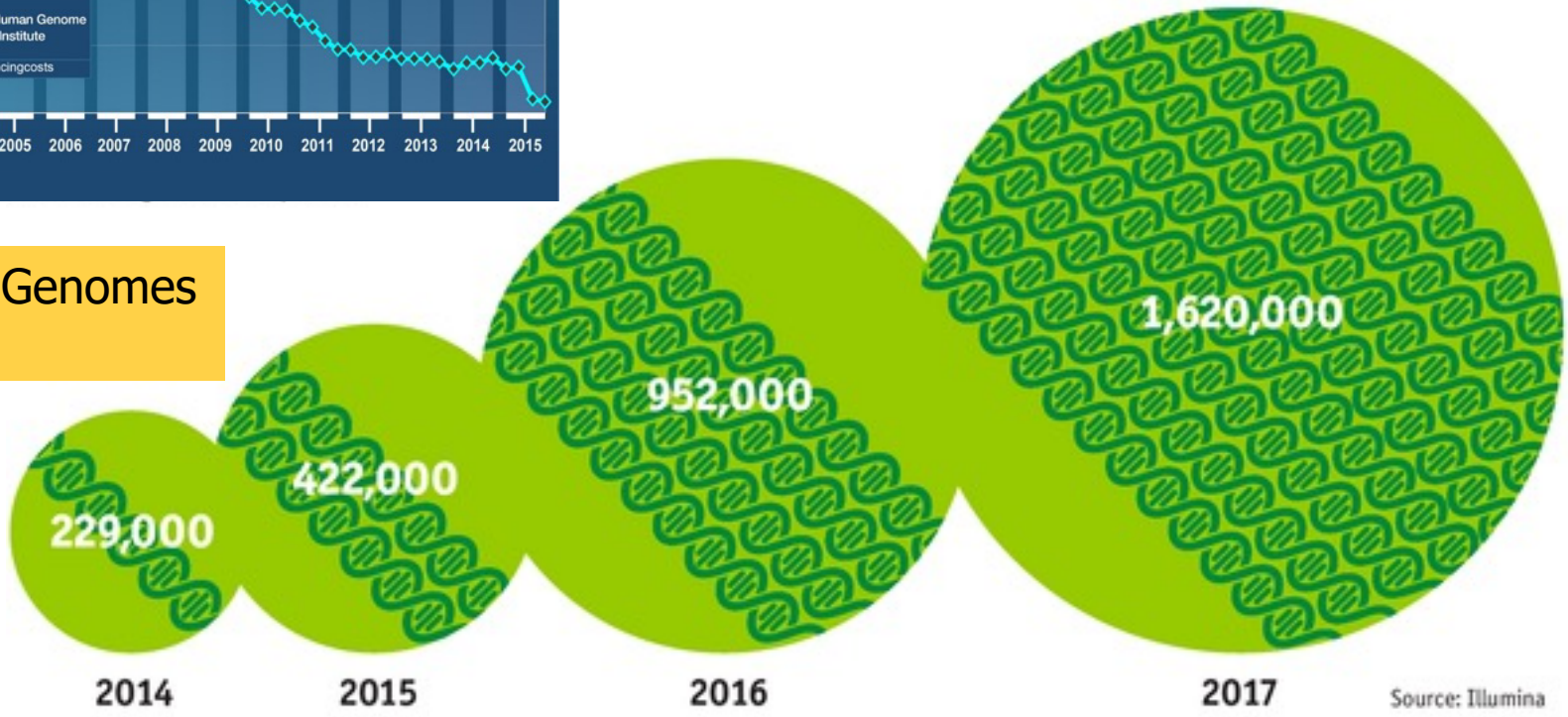
Accelerating Emerging Workloads

Data is Key for Future Workloads



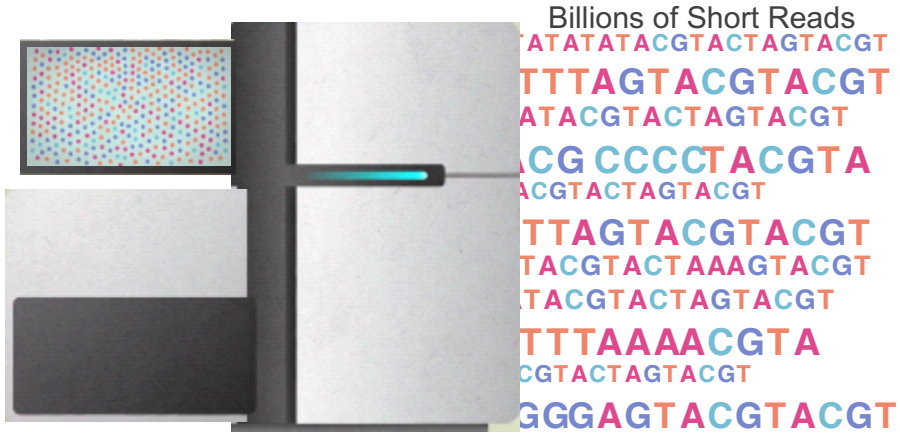
development of high-throughput sequencing (HTS) technologies

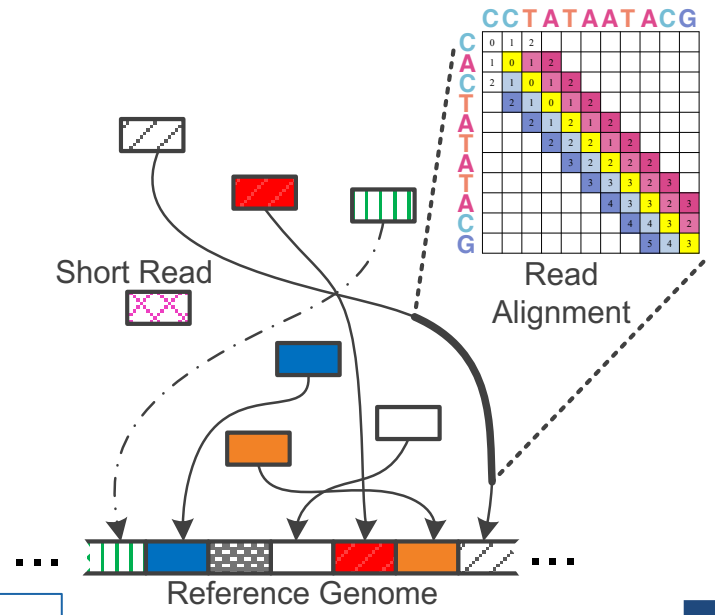
Number of Genomes Sequenced



The Economist

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





1 Sequencing

2 Read Mapping

Genome Analysis

Data → performance & energy bottleneck

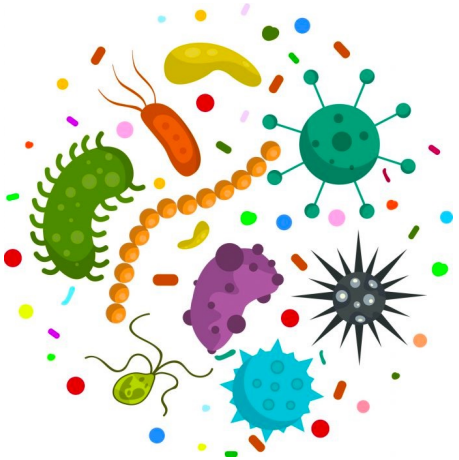
read4: CGCTTCCAT
read5: CCATGACGC
read6: TTCCATGAC

3 Variant Calling

4 Scientific Discovery

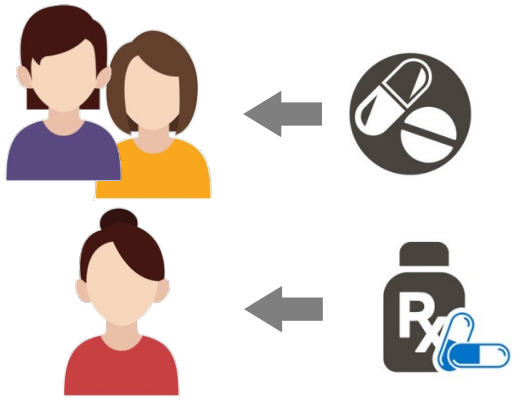
We Need Faster & Scalable Genome Analysis

Understanding **genetic variations, species, evolution, …**



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

Rapid surveillance of **disease outbreaks**



Developing **personalized medicine**

And, many, many other applications …

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

Senol Cali+, "**Nanopore Sequencing Technology and Tools for Genome Assembly: Computational Analysis of the Current State, Bottlenecks and Future Directions**," Briefings in Bioinformatics, 2018.
[Open arxiv.org version]

New Genome Sequencing Technologies

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

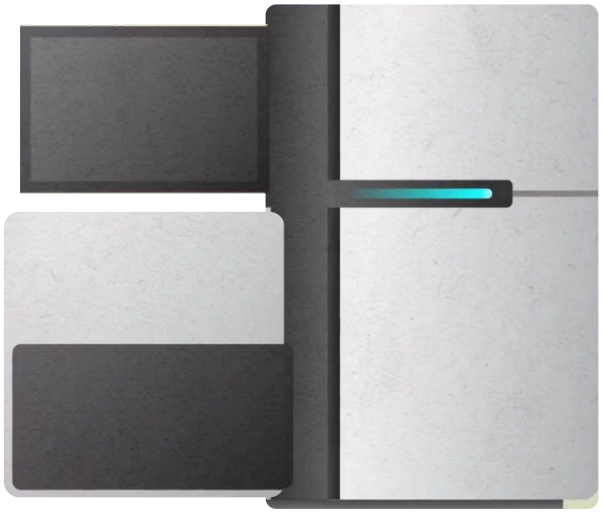
Damla Senol Cali ✉, Jeremie S Kim, Saugata Ghose, Can Alkan, Onur Mutlu

Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017
Published: 02 April 2018 **Article history** ▾

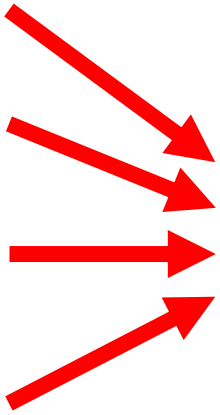
Oxford Nanopore MinION

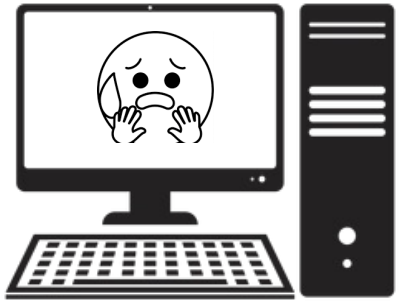
Data → performance & energy bottleneck

Problems with (Genome) Analysis Today



Special-Purpose Machine
for **Data Generation**





General-Purpose Machine
for **Data Analysis**

FAST

SLOW

Slow and inefficient processing capability
Large amounts of data movement

This picture is similar for many "data generators & analyzers" today

Accelerating Genome Analysis [IEEE MICRO 2020]

- Mohammed Alser, Zulal Bingol, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  IEEE Micro (***IEEE MICRO***), Vol. 40, No. 5, pages 65-75, September/October 2020.
  [Slides (pptx)(pdf)]
  [Talk Video (1 hour 2 minutes)]

Accelerating Genome Analysis: A Primer on an Ongoing Journey

Mohammed Alser
ETH Zürich

Zülal Bingöl
Bilkent University

Damla Senol Cali
Carnegie Mellon University

Jeremie Kim
ETH Zurich and Carnegie Mellon University

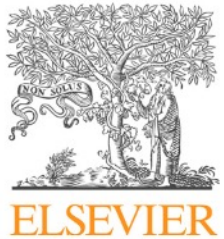
Saugata Ghose
University of Illinois at Urbana–Champaign and Carnegie Mellon University

Can Alkan
Bilkent University

Onur Mutlu
ETH Zurich, Carnegie Mellon University, and Bilkent University

Beginner Reading on Genome Analysis

Mohammed Alser, Joel Lindegger, Can Firtina, Nour Almadhoun, Haiyu Mao, Gagandeep Singh, Juan Gomez-Luna, Onur Mutlu
"**From Molecules to Genomic Variations to Scientific Discovery: Intelligent Algorithms and Architectures for Intelligent Genome Analysis**"
Computational and Structural Biotechnology Journal, 2022
[Source code]



journal homepage: www.elsevier.com/locate/csbj

Review

From molecules to genomic variations: Accelerating genome analysis via intelligent algorithms and architectures

Mohammed Alser *, Joel Lindegger, Can Firtina, Nour Almadhoun, Haiyu Mao, Gagandeep Singh, Juan Gomez-Luna, Onur Mutlu *

ETH Zurich, Gloriastrasse 35, 8092 Zürich, Switzerland

https://arxiv.org/pdf/2205.07957.pdf

Accelerating Approximate String Matching

- Damla Senol Cali, Gurpreet S. Kalsi, Zulal Bingol, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu, **"GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"**
  *Proceedings of the 53rd International Symposium on Microarchitecture (**MICRO**)*, Virtual, October 2020.
  [Lighting Talk Video (1.5 minutes)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (18 minutes)]
  [Slides (pptx) (pdf)]

GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali[†⋈] Gurpreet S. Kalsi[⋈] Zülal Bingöl[▽] Can Firtina[◇] Lavanya Subramanian[‡] Jeremie S. Kim[◇†]
Rachata Ausavarungnirun[⊙] Mohammed Alser[◇] Juan Gomez-Luna[◇] Amirali Boroumand[†] Anant Nori[⋈]
Allison Scibisz[†] Sreenivas Subramoney[⋈] Can Alkan[▽] Saugata Ghose[⋆†] Onur Mutlu[◇†▽]

[†]*Carnegie Mellon University* [⋈]*Processor Architecture Research Lab, Intel Labs* [▽]*Bilkent University* [◇]*ETH Zürich*
[‡]*Facebook* [⊙]*King Mongkut's University of Technology North Bangkok* [⋆]*University of Illinois at Urbana–Champaign*

Accelerating Sequence-to-Graph Mapping

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

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

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

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

Accelerating Basecalling + Read Mapping

- Appears at MICRO 2022

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

Haiyu Mao[1] Mohammed Alser[1] Mohammad Sadrosadati[1] Can Firtina[1] Akanksha Baranwal[1]
Damla Senol Cali[2] Aditya Manglik[1] Nour Almadhoun Alserr[1] Onur Mutlu[1]
[1]*ETH Zürich* [2]*Bionano Genomics*

 https://arxiv.org/pdf/2209.08600.pdf

Designing & Accelerating Basecallers

A Framework for Designing Efficient Deep Learning-Based Genomic Basecallers

Gagandeep Singh[a] Mohammed Alser*[a] Alireza Khodamoradi*[b]
Kristof Denolf[b] Can Firtina[a] Meryem Banu Cavlak[a]
Henk Corporaal[c] Onur Mutlu[a]
[a]ETH Zürich [b]AMD [c]Eindhoven University of Technology

Nanopore sequencing is a widely-used high-throughput genome sequencing technology that can sequence long fragments of a genome. Nanopore sequencing generates noisy electrical signals that need to be converted into a standard string of DNA nucleotide bases (i.e., A, C, G, T) using a computational step called *basecalling*. The accuracy and speed of basecalling have critical implications for every subsequent step in genome analysis. Currently, basecallers are developed mainly based on deep learning techniques to provide high sequencing accuracy without considering the compute demands of such tools. We observe that state-of-the-art basecallers (i.e., Guppy, Bonito, Fast-Bonito) are slow, inefficient, and memory-hungry

 https://arxiv.org/pdf/2211.03079.pdf

In-Storage Genome Filtering [ASPLOS 2022]

- Nika Mansouri Ghiasi, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alserr, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu,
  "GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis"
  *Proceedings of the 27th International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Virtual, February-March 2022.
  [Talk Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Lightning Talk Video (90 seconds)]
  [Talk Video (17 minutes)]

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

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

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

Accelerating Climate Modeling

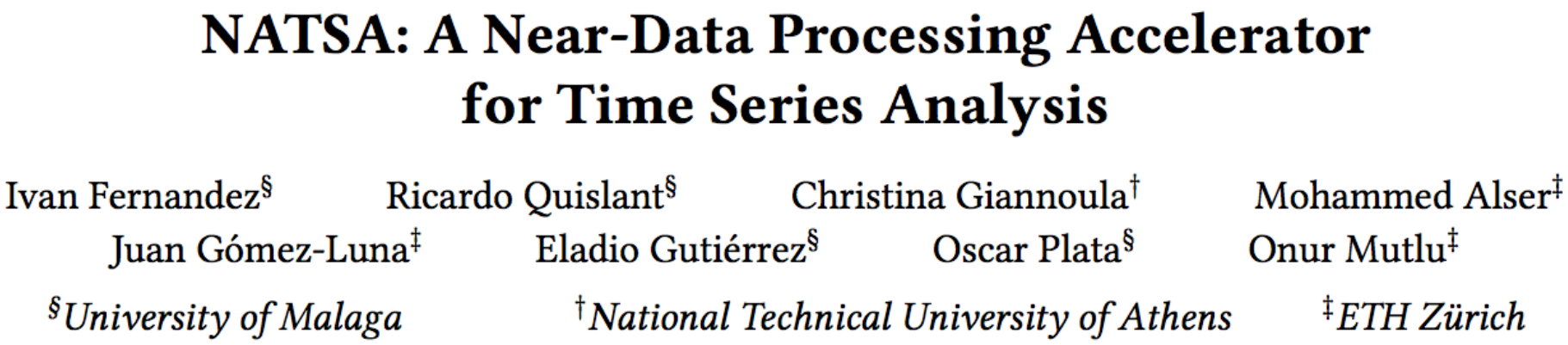
- Gagandeep Singh, Dionysios Diamantopoulos, Christoph Hagleitner, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal,
  "NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling"
  *Proceedings of the 30th International Conference on Field-Programmable Logic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

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

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

Accelerating Time Series Analysis

- Ivan Fernandez, Ricardo Quislant, Christina Giannoula, Mohammed Alser, Juan Gómez-Luna, Eladio Gutiérrez, Oscar Plata, and Onur Mutlu,
  "NATSA: A Near-Data Processing Accelerator for Time Series Analysis"
  *Proceedings of the 38th IEEE International Conference on Computer Design (**ICCD**)*, Virtual, October 2020.
  [Slides (pptx) (pdf)]
  [Talk Video (10 minutes)]
  [Source Code]



NATSA: A Near-Data Processing Accelerator for Time Series Analysis

Ivan Fernandez[§] Ricardo Quislant[§] Christina Giannoula[†] Mohammed Alser[‡]
Juan Gómez-Luna[‡] Eladio Gutiérrez[§] Oscar Plata[§] Onur Mutlu[‡]

[§]*University of Malaga* [†]*National Technical University of Athens* [‡]*ETH Zürich*

Accelerating Graph Pattern Mining

- Maciej Besta, Raghavendra Kanakagiri, Grzegorz Kwasniewski, Rachata Ausavarungnirun, Jakub Beránek, Konstantinos Kanellopoulos, Kacper Janda, Zur Vonarburg-Shmaria, Lukas Gianinazzi, Ioana Stefan, Juan Gómez-Luna, Marcin Copik, Lukas Kapp-Schwoerer, Salvatore Di Girolamo, Nils Blach, Marek Konieczny, Onur Mutlu, and Torsten Hoefler,
  "SISA: Set-Centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems"
  *Proceedings of the 54th International Symposium on Microarchitecture (**MICRO**)*, Virtual, October 2021.
  [Slides (pdf)]
  [Talk Video (22 minutes)]
  [Lightning Talk Video (1.5 minutes)]
  [Full arXiv version]

SISA: Set-Centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems

Maciej Besta[1], Raghavendra Kanakagiri[2], Grzegorz Kwasniewski[1], Rachata Ausavarungnirun[3], Jakub Beránek[4], Konstantinos Kanellopoulos[1], Kacper Janda[5], Zur Vonarburg-Shmaria[1], Lukas Gianinazzi[1], Ioana Stefan[1], Juan Gómez-Luna[1], Marcin Copik[1], Lukas Kapp-Schwoerer[1], Salvatore Di Girolamo[1], Nils Blach[1], Marek Konieczny[5], Onur Mutlu[1], Torsten Hoefler[1]

[1]ETH Zurich, Switzerland [2]IIT Tirupati, India [3]King Mongkut's University of Technology North Bangkok, Thailand [4]Technical University of Ostrava, Czech Republic [5]AGH-UST, Poland

Accelerating HTAP Database Systems

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

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

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

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

Accelerating Neural Network Inference

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

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

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

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

Accelerating Neural Network Inference

- **Appears in IEEE Micro**

Accelerating Neural Network Inference with Processing-in-DRAM: From the Edge to the Cloud

Geraldo F. Oliveira
ETH Zürich

Juan Gómez-Luna
ETH Zürich

Saugata Ghose
University of Illinois Urbana-Champaign

Amirali Boroumand
Google

Onur Mutlu
ETH Zürich

https://arxiv.org/pdf/2209.08938.pdf

FPGA-based Near-Memory Analytics

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

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

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

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

GenASM: Approximate String Matching Accelerator for Genomics

GenASM Framework [MICRO 2020]

- Damla Senol Cali, Gurpreet S. Kalsi, Zulal Bingol, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu, **"GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"**
  *Proceedings of the 53rd International Symposium on Microarchitecture (**MICRO**)*, Virtual, October 2020.
  [Lighting Talk Video (1.5 minutes)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (18 minutes)]
  [Slides (pptx) (pdf)]

GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali[†⋈] Gurpreet S. Kalsi[⋈] Zülal Bingöl[▽] Can Firtina[◇] Lavanya Subramanian[‡] Jeremie S. Kim[◇†]
Rachata Ausavarungnirun[⊙] Mohammed Alser[◇] Juan Gomez-Luna[◇] Amirali Boroumand[†] Anant Nori[⋈]
Allison Scibisz[†] Sreenivas Subramoney[⋈] Can Alkan[▽] Saugata Ghose[⋆†] Onur Mutlu[◇†▽]

[†]*Carnegie Mellon University* [⋈]*Processor Architecture Research Lab, Intel Labs* [▽]*Bilkent University* [◇]*ETH Zürich*
[‡]*Facebook* [⊙]*King Mongkut's University of Technology North Bangkok* [⋆]*University of Illinois at Urbana–Champaign*

GenASM: A High Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali

Carnegie Mellon University

(dsenol@andrew.cmu.edu)

Gurpreet S. Kalsi[2], Zulal Bingol[3], Can Firtina[4], Lavanya Subramanian[5], Jeremie Kim[1,4], Rachata Ausavarungnirun[6,1], Mohammed Alser[4], Juan Gomez-Luna[4], Amirali Boroumand[1], Anant Nori[2], Allison Scibisz[1], Sreenivas Subramoney[2], Can Alkan[3], Saugata Ghose[7,1], and Onur Mutlu[4,1,3]

[1] Carnegie Mellon [2] intel [3] Bilkent University [4] ETH zürich

[5] facebook [6] King Mongkut's University of Technology North Bangkok [7] University of Illinois Urbana-Champaign [1,4] SAFARI

Genome Sequencing

- **Genome sequencing:** Enables us to determine the order of the DNA sequence in an organism's genome
  - Plays a pivotal role in:
    - Personalized medicine
    - Outbreak tracing
    - Understanding of evolution
    - …

Genome

DNA

- Modern genome sequencing machines extract smaller randomized fragments of the original DNA sequence, known as **reads**
  - *Short reads:* a few hundred base pairs, error rate of ~0.1%
  - *Long reads:* thousands to millions of base pairs, error rate of 10–15%

Genome Sequence Analysis

- **Read mapping:** *First key step* in genome sequence analysis (GSA)
  - Aligns reads to one or more possible locations within the reference genome, and
  - Finds the matches and differences between the read and the reference genome segment at that location

- Multiple steps of read mapping require ***approximate string matching***
  - Approximate string matching (ASM) enables read mapping to account for sequencing errors and genetic variations in the reads

- Bottlenecked by the computational power and memory bandwidth limitations of existing systems

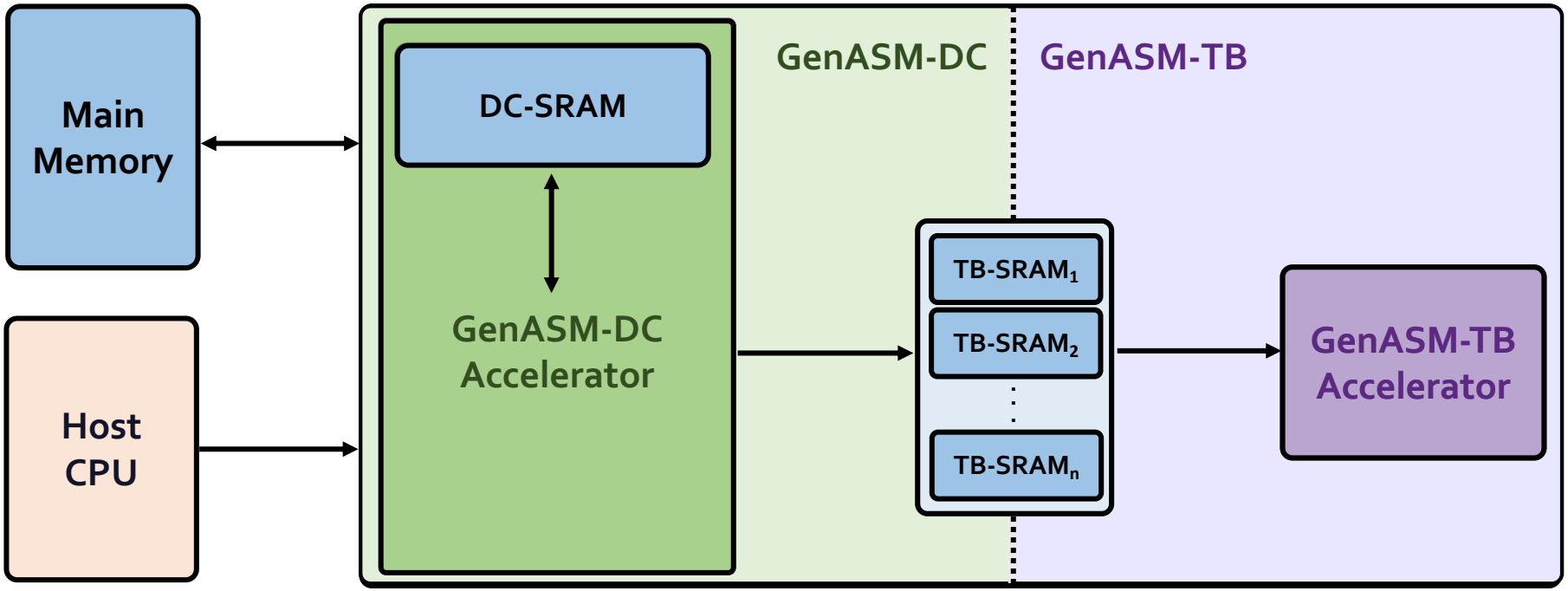
GenASM: ASM Framework for GSA

Our Goal:

Accelerate approximate string matching
by designing a fast and flexible framework,
which can accelerate *multiple steps* of genome sequence analysis

- **GenASM:** *First* ASM acceleration framework for GSA
  - Based upon the *Bitap* algorithm
    - Uses fast and simple bitwise operations to perform ASM
  - Modified and extended ASM algorithm
    - Highly-parallel Bitap with long read support
    - Bitvector-based novel algorithm to perform *traceback*
  - Co-design of our modified scalable and memory-efficient algorithms with low-power and area-efficient hardware accelerators

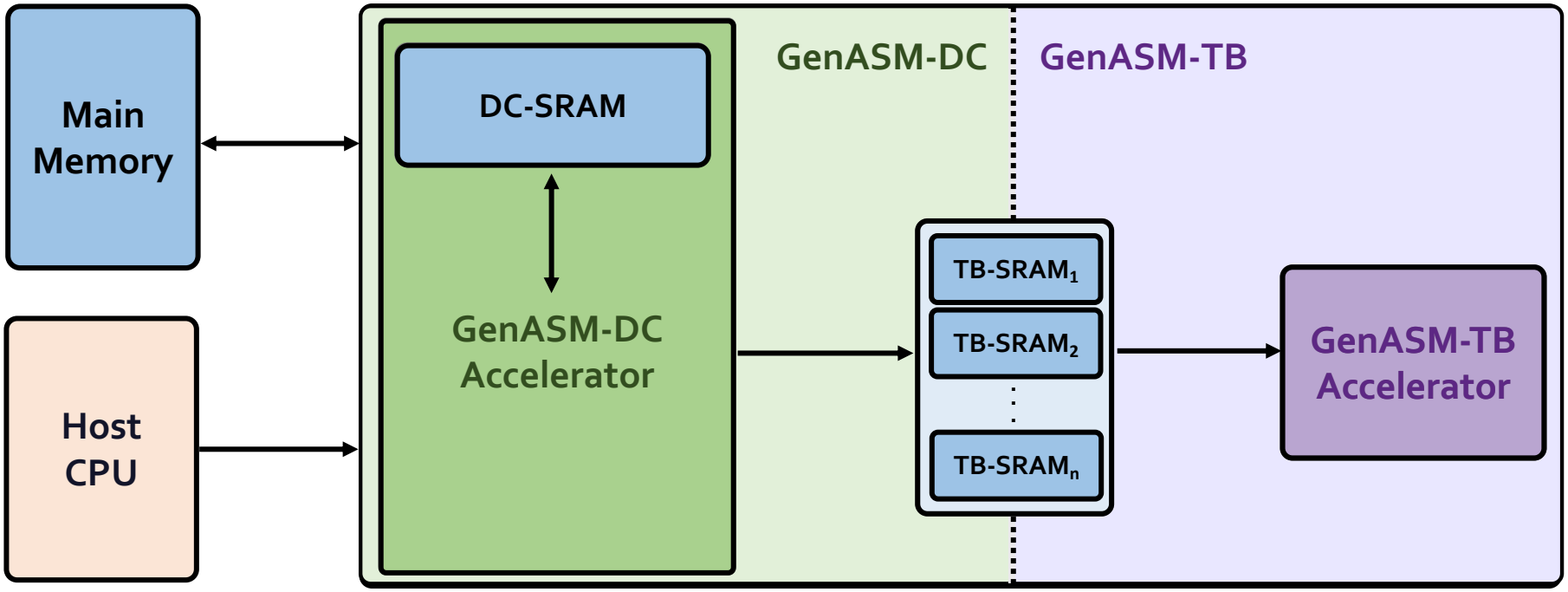
GenASM: Hardware Design



GenASM-DC:
generates bitvectors and performs edit **D**istance **C**alculation

GenASM-TB:
performs **T**race**B**ack and assembles the optimal alignment

GenASM: Hardware Design



Our *specialized compute units* and *on-chip SRAMs* help us to:

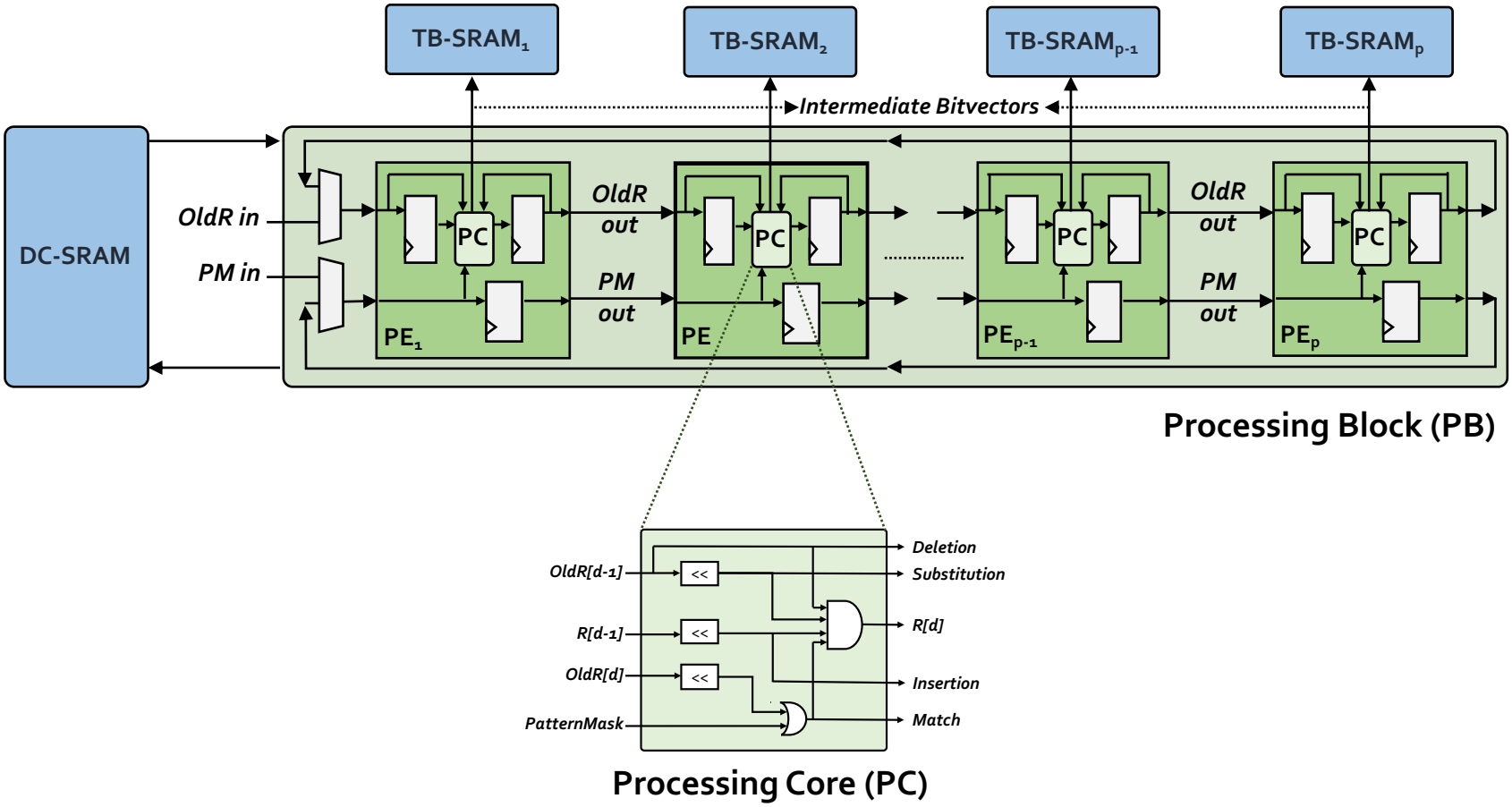
→ Match the rate of computation with memory capacity and bandwidth

→ Achieve high performance and power efficiency

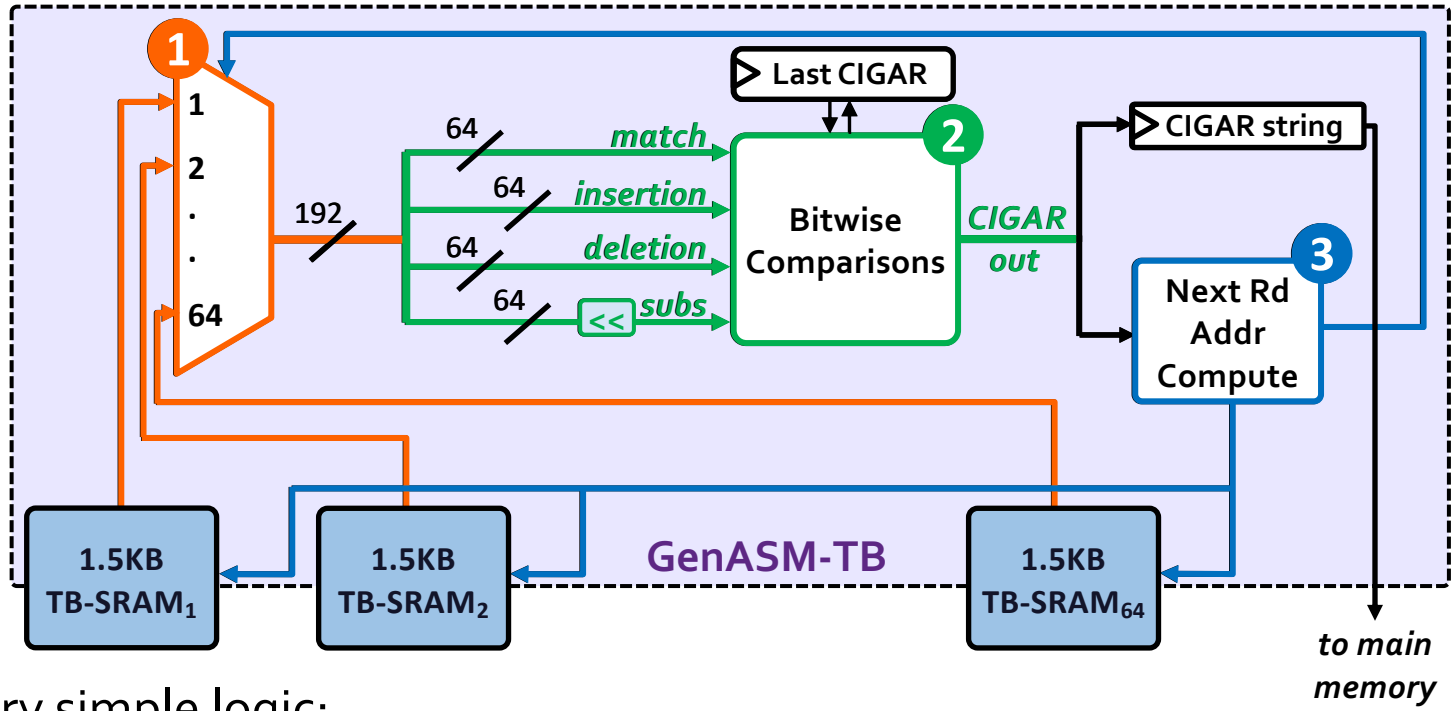
→ Scale linearly in performance with the number of parallel compute units that we add to the system

GenASM-DC: Hardware Design

- **Linear cyclic systolic array** based accelerator
  - Designed to maximize parallelism and minimize memory bandwidth and memory footprint



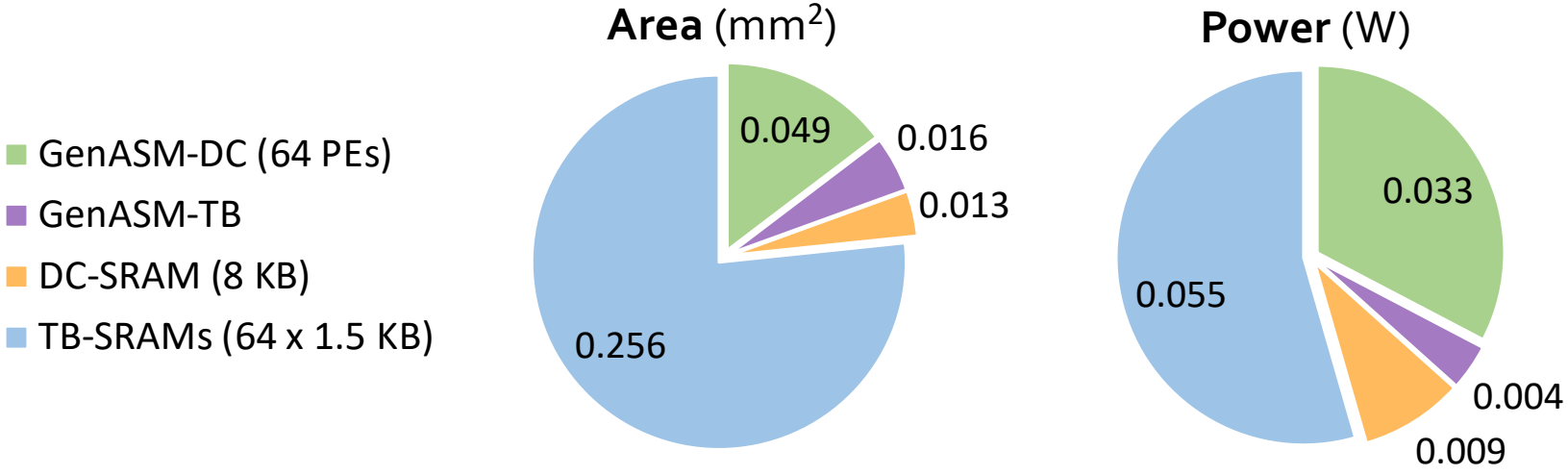
GenASM-TB: Hardware Design



- Very simple logic:
  - 1 Reads the bitvectors from one of the TB-SRAMs using the computed address
  - 2 Performs the required bitwise comparisons to find the traceback output for the current position
  - 3 Computes the next TB-SRAM address to read the new set of bitvectors

Key Results – Area and Power

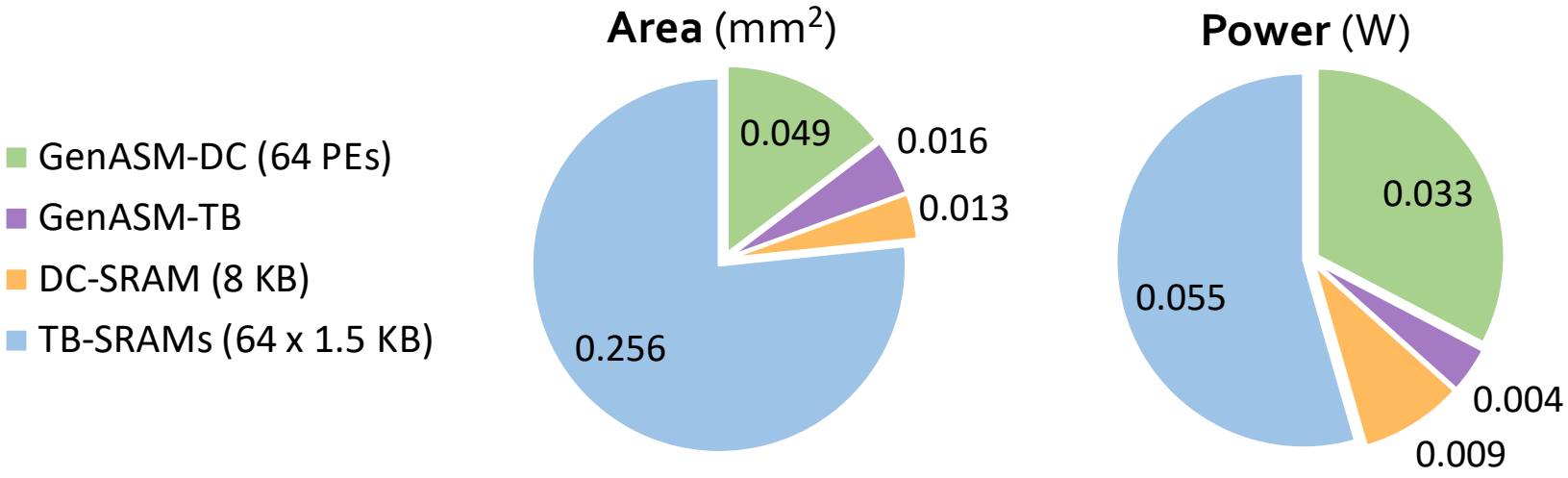
- Based on our **synthesis** of **GenASM-DC** and **GenASM-TB** accelerator datapaths using the Synopsys Design Compiler with a **28nm** LP process:
  - Both GenASM-DC and GenASM-TB operate **@ 1GHz**



Total (1 vault):	0.334 mm^2	0.101 W
Total (32 vaults):	10.69 mm^2	3.23 W
% of a Xeon CPU core:	**1%**	**1%**

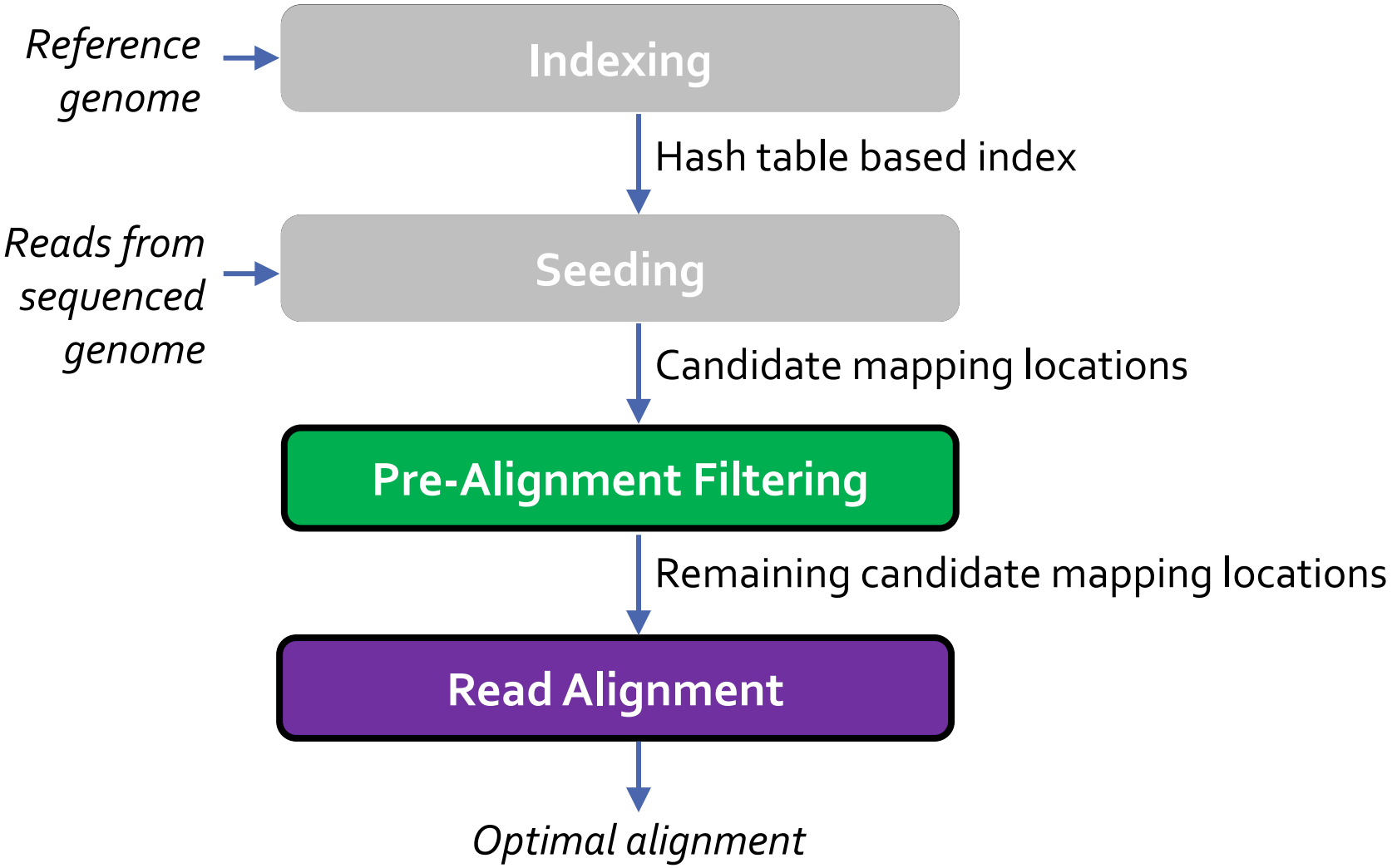
Key Results – Area and Power

- Based on our **synthesis** of **GenASM-DC** and **GenASM-TB** accelerator datapaths using the Synopsys Design Compiler with a **28nm** LP process:
  - Both GenASM-DC and GenASM-TB operate **@ 1GHz**



GenASM has low area and power overheads

Use Cases of GenASM



Use Cases of GenASM (cont'd.)

(1) Read Alignment Step of Read Mapping

- Find the optimal alignment of how reads map to candidate reference regions

(2) Pre-Alignment Filtering for Short Reads

- Quickly identify and filter out the unlikely candidate reference regions for each read

(3) Edit Distance Calculation

- Measure the similarity or distance between two sequences

- Other possible use cases of GenASM in our paper:
  - Read-to-read overlap finding, hash-table based indexing, whole genome alignment, generic text search

Key Results

(1) Read Alignment

- **116×** speedup, **37×** less power than **Minimap2** (state-of-the-art **SW**)
- **111×** speedup, **33×** less power than **BWA-MEM** (state-of-the-art **SW**)
- **3.9×** better throughput, **2.7×** less power than **Darwin** (state-of-the-art **HW**)
- **1.9×** better throughput, **82%** less logic power than **GenAx** (state-of-the-art **HW**)

(2) Pre-Alignment Filtering

- **3.7×** speedup, **1.7×** less power than **Shouji** (state-of-the-art **HW**)

(3) Edit Distance Calculation

- **22–12501×** speedup, **548–582×** less power than **Edlib** (state-of-the-art **SW**)
- **9.3–400×** speedup, **67×** less power than **ASAP** (state-of-the-art **HW**)

More on GenASM Framework [MICRO 2020]

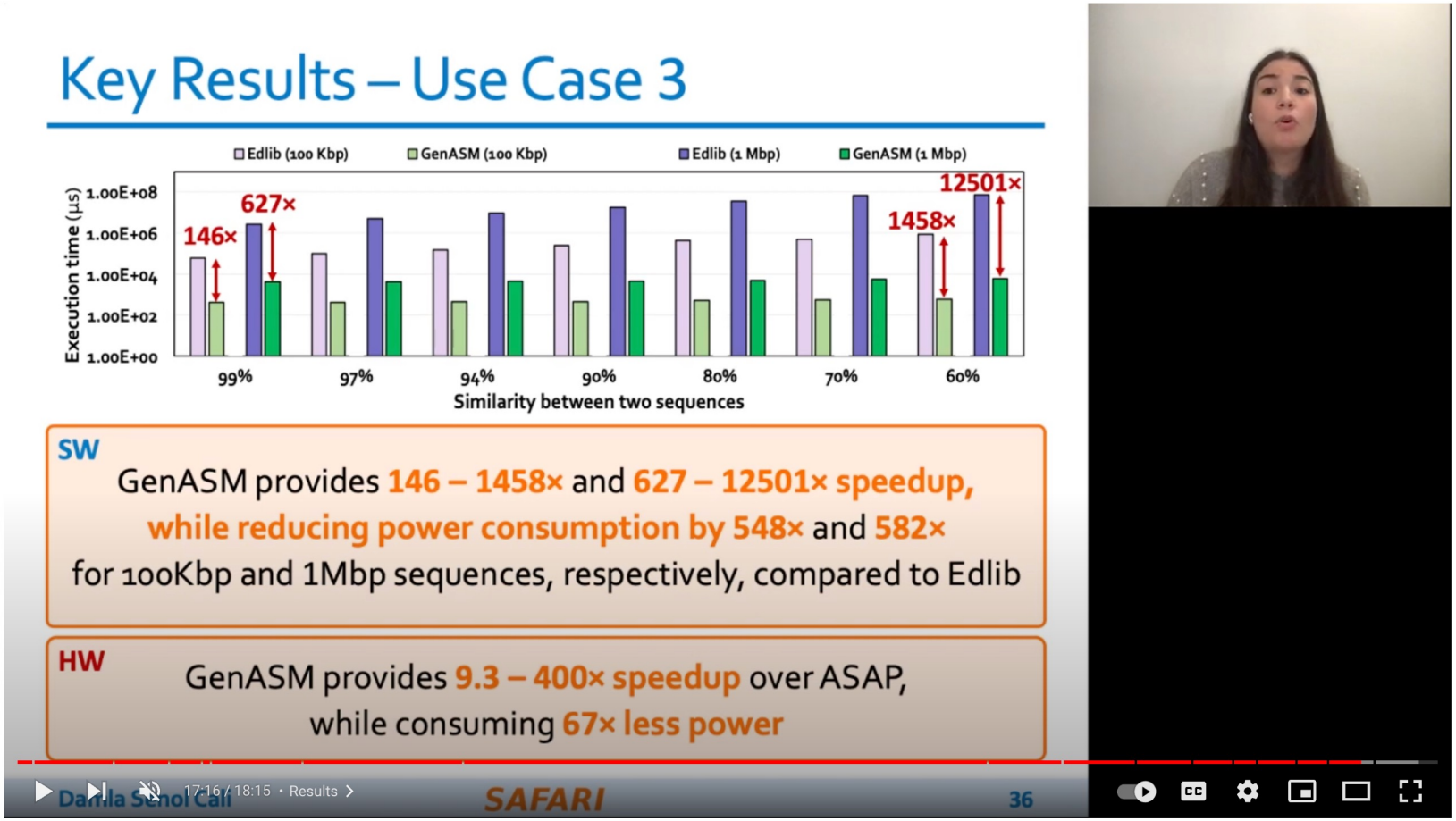
- Damla Senol Cali, Gurpreet S. Kalsi, Zulal Bingol, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu, **"GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"**
  *Proceedings of the 53rd International Symposium on Microarchitecture (**MICRO**)*, Virtual, October 2020.
  [Lighting Talk Video (1.5 minutes)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (18 minutes)]
  [Slides (pptx) (pdf)]

GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali[†⋈] Gurpreet S. Kalsi[⋈] Zülal Bingöl[▽] Can Firtina[◇] Lavanya Subramanian[‡] Jeremie S. Kim[◇†]
Rachata Ausavarungnirun[⊙] Mohammed Alser[◇] Juan Gomez-Luna[◇] Amirali Boroumand[†] Anant Nori[⋈]
Allison Scibisz[†] Sreenivas Subramoney[⋈] Can Alkan[▽] Saugata Ghose[⋆†] Onur Mutlu[◇†▽]

[†]*Carnegie Mellon University* [⋈]*Processor Architecture Research Lab, Intel Labs* [▽]*Bilkent University* [◇]*ETH Zürich*
[‡]*Facebook* [⊙]*King Mongkut's University of Technology North Bangkok* [⋆]*University of Illinois at Urbana–Champaign*

GenASM Talk Video



GenASM: Accelerating Approximate String Matching in Genomics (MICRO'20 Full Talk, Damla Senol Cali)

Onur Mutlu Lectures
28.9K subscribers
Analytics | Edit video | 12 | Share | Download | Clip | Save

384 views 1 year ago
Presented at MICRO 2020, October 2020 by Damla Senol Cali.

https://www.youtube.com/watch?v=NFU2GANPJNU

GenASM: A High Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali

Carnegie Mellon University

(dsenol@andrew.cmu.edu)

Gurpreet S. Kalsi[2], Zulal Bingol[3], Can Firtina[4], Lavanya Subramanian[5], Jeremie Kim[1,4], Rachata Ausavarungnirun[6,1], Mohammed Alser[4], Juan Gomez-Luna[4], Amirali Boroumand[1], Anant Nori[2], Allison Scibisz[1], Sreenivas Subramoney[2], Can Alkan[3], Saugata Ghose[7,1], and Onur Mutlu[4,1,3]

[1] Carnegie Mellon [2] intel [3] Bilkent University [4] ETH zürich

[5] facebook [6] King Mongkut's University of Technology North Bangkok [7] University of Illinois Urbana-Champaign [1,4] SAFARI

SeGraM: Sequence-to-Graph Mapping Accelerator for Genomics

Accelerating Sequence-to-Graph Mapping

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

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

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

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

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

Damla Senol Cali, Ph.D.

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

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

Carnegie Mellon
ETH zürich
Bilkent University
intel
University of Illinois Urbana-Champaign
SAFARI

Problem: Sequence-to-Graph Mapping

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

Linear Reference: ACGTACGT

Read: ACGG

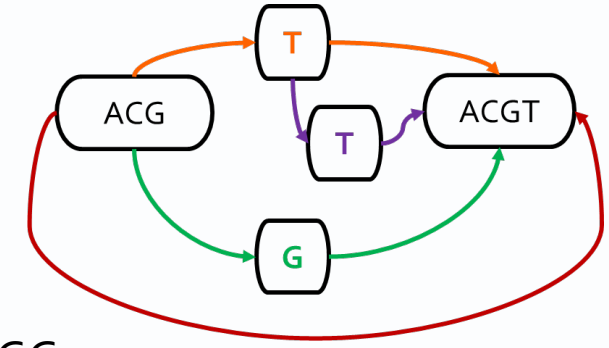
Alternative Sequence: ACGGACGT

Alternative Sequence: ACGTTACGT

Alternative Sequence: ACG–ACGT

Sequence-to-Sequence (S2S) Mapping

Graph-based Reference:



Read: ACGG

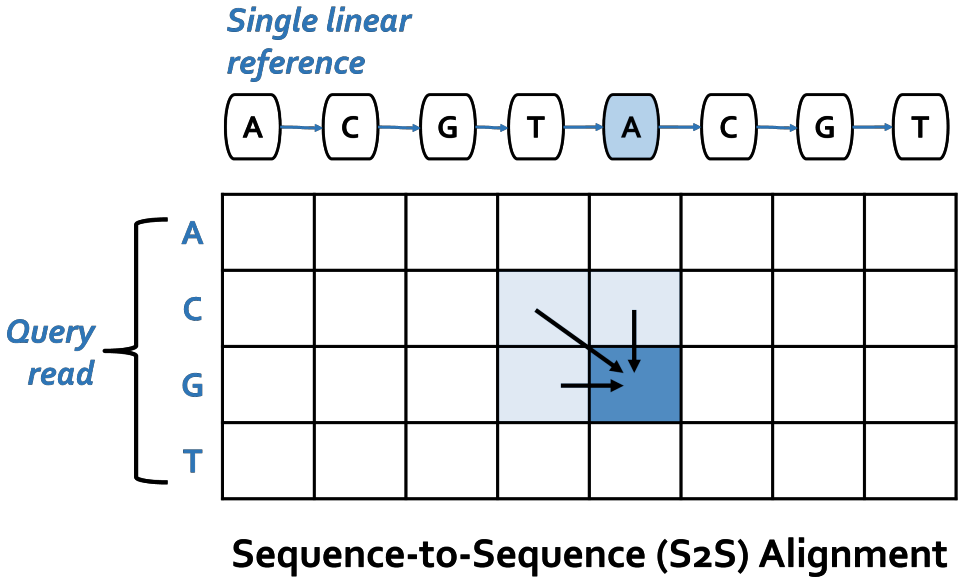
Sequence-to-Graph (S2G) Mapping

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

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

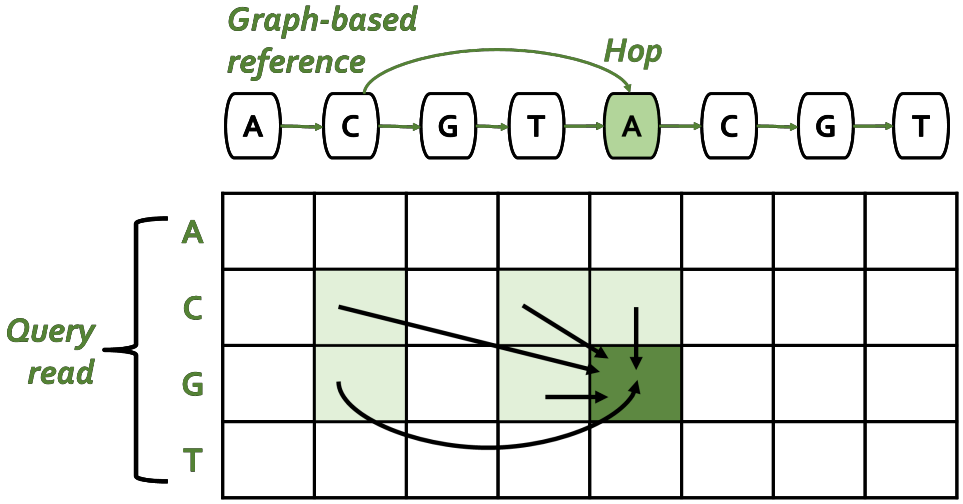
with **no prior hardware design.**

S2S vs. S2G Alignment



Sequence-to-Sequence (S2S) Alignment

S2S vs. S2G Alignment



Sequence-to-Graph (S2G) Alignment

In contrast to S2S alignment,

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

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

from the non-neighboring character to the current character

SeGraM: First Graph Mapping Accelerator

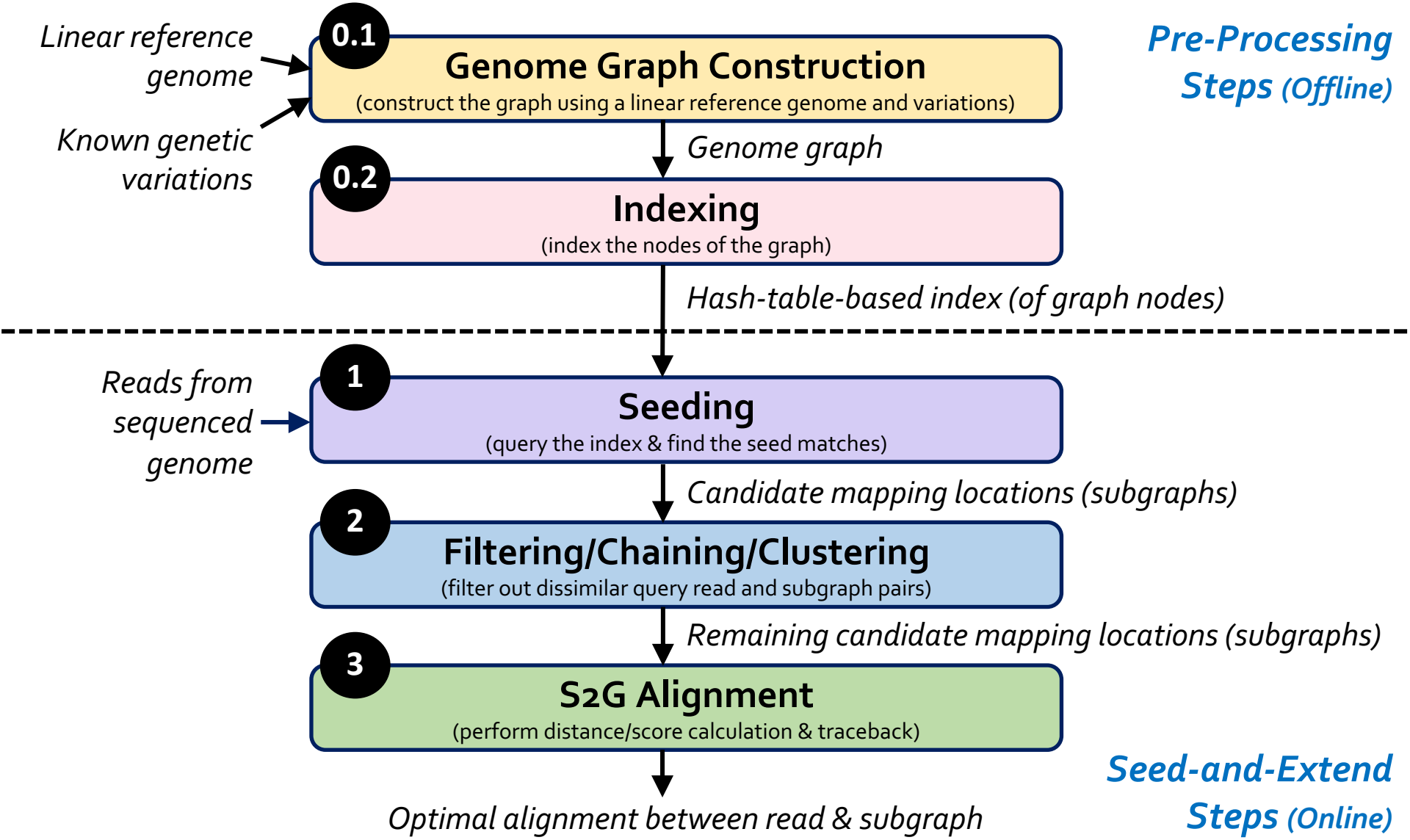
Our Goal:

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

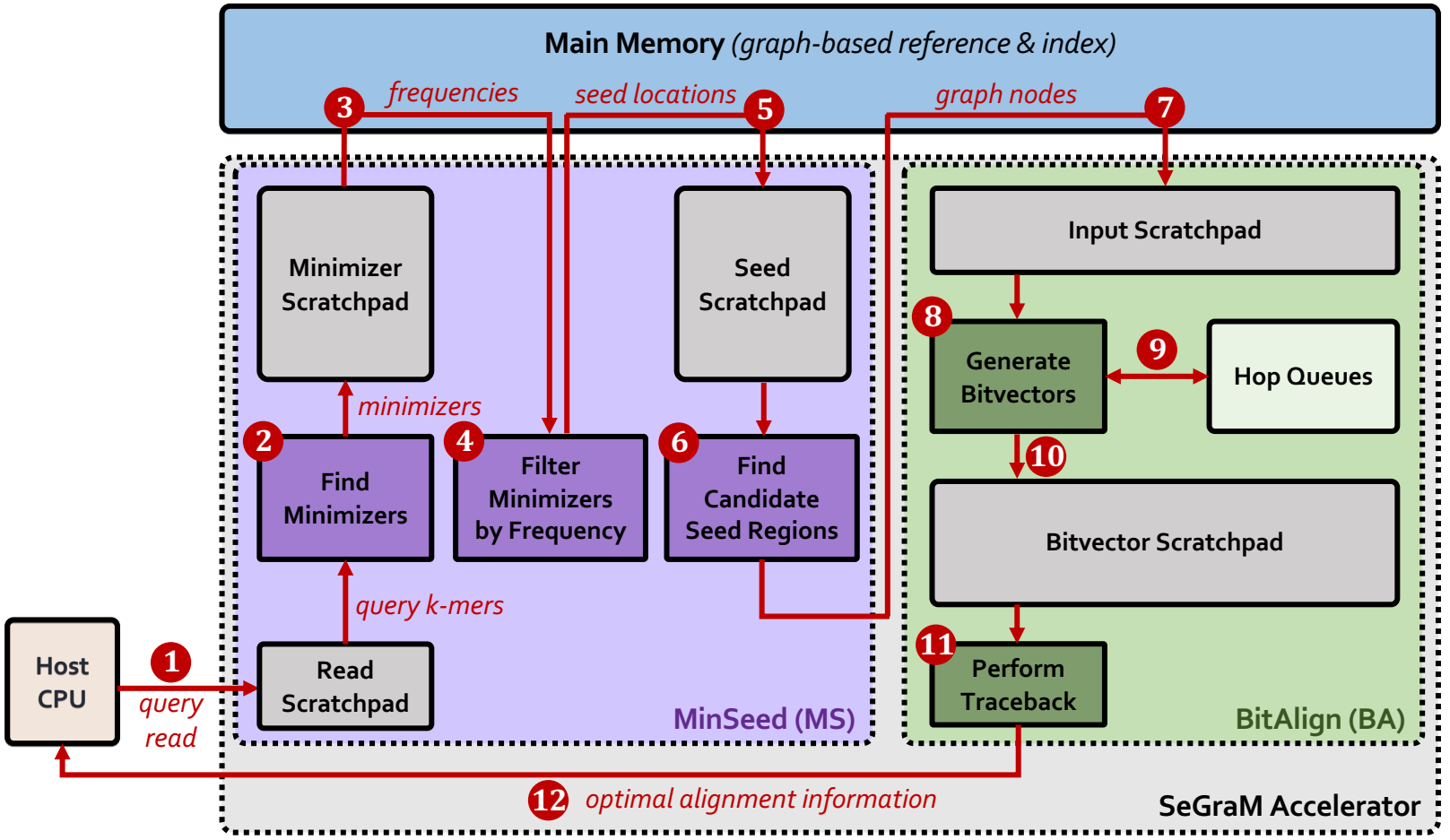
SeGraM: *First universal algorithm/hardware co-designed genomic mapping accelerator* that can effectively and efficiently support:

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

Sequence-to-Graph Mapping Pipeline



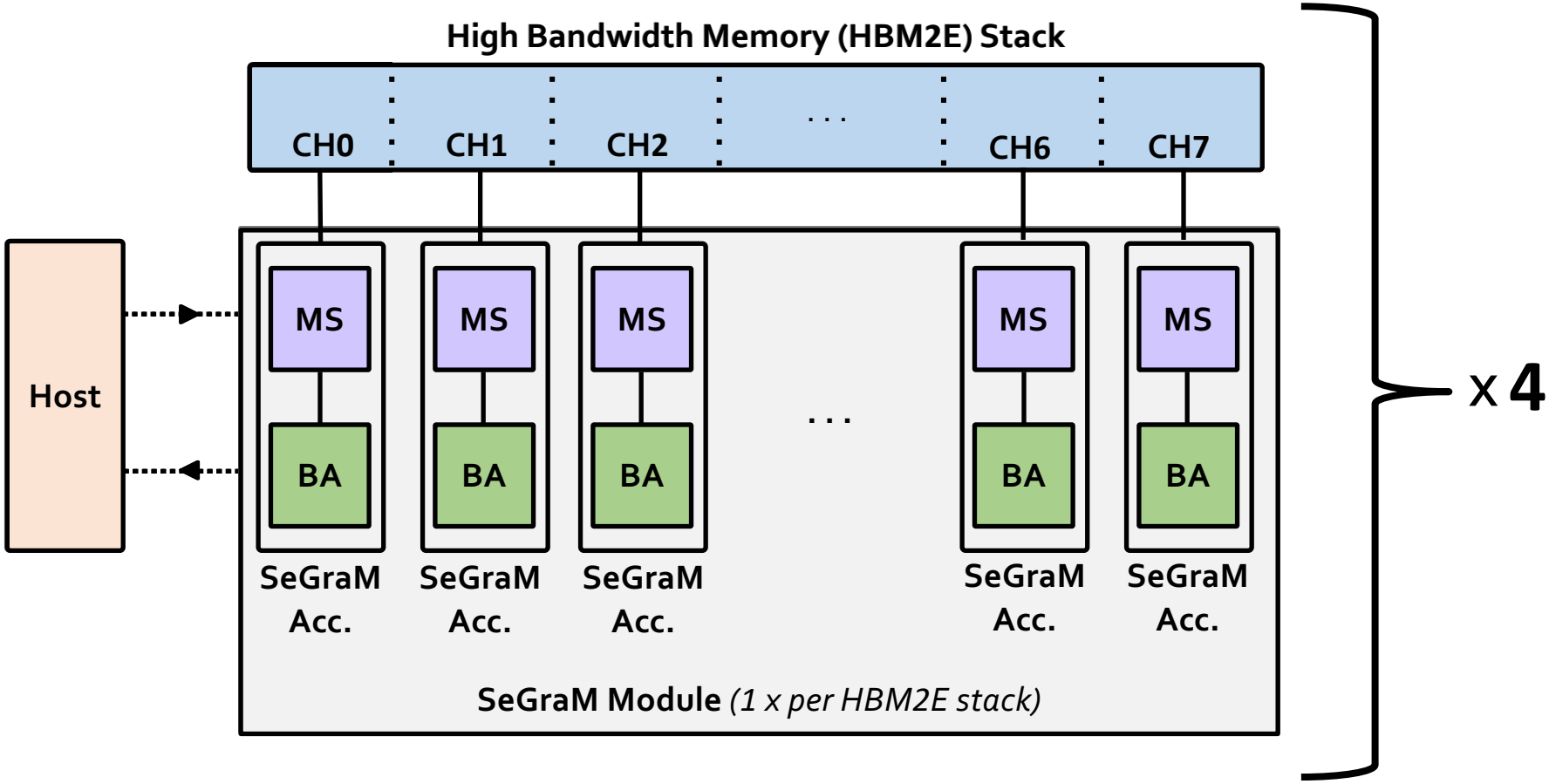
SeGraM Hardware Design



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

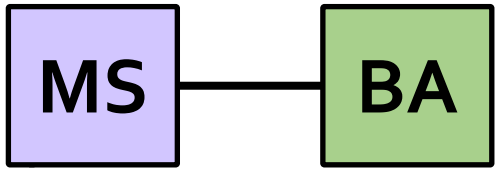
BitAlign (BA): *first* hardware accelerator for (**Bit**vector-based) sequence-to-graph **Align**ment

Overall System Design of SeGraM

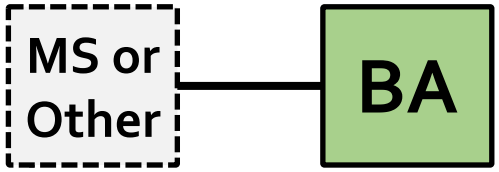


Use Cases of SeGraM

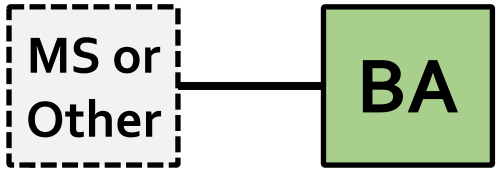
(1) Sequence-to-Graph Mapping



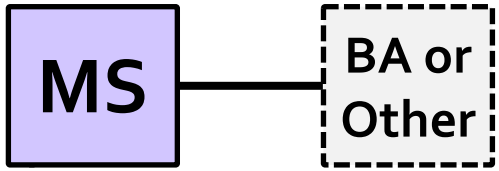
(2) Sequence-to-Graph Alignment



(3) Sequence-to-Sequence Alignment



(4) Seeding



Use Cases & Key Results

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

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

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

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

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

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

SeGraM is Open Source

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

damlasenolcali Update README.md — 45d20dc 6 days ago — 26 commits

src	BitAlign source files	6 days ago
LICENSE	Initial commit	7 months ago
README.md	Update README.md	6 days ago

README.md

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

SeGraM is a universal genomic mapping accelerator that supports both sequence-to-graph mapping and sequence-to sequence mapping, for both short and long reads. SeGraM consists of two main components: (1) MinSeed, the first minimizer-based seeding accelerator, which finds the candidate mapping locations (i.e., subgraphs) in a given genome graph; and (2) BitAlign, the first bitvector-based sequence-to-graph alignment accelerator, which performs alignment between a given read and the subgraph identified by MinSeed. MinSeed is built upon a memory-efficient minimizer-based seeding algorithm, and BitAlign is built upon our novel bitvector-based, highly-parallel sequence-to-graph alignment algorithm.

In MinSeed, the minimizer-based seeding approach decreases the memory footprint of the index and provides speedup during seed queries. MinSeed logic requires only basic operations (e.g., comparisons, simple arithmetic operations, scratchpad read-write operations) that are implemented with simple logic. Due to frequent memory accesses required for fetching the seeds, we couple MinSeed with High-Bandwidth Memory (HBM) to enable low-latency and highly-parallel memory access, which alleviates the memory latency bottleneck.

In BitAlign, we design a new bitvector-based alignment approach, which is amenable to efficient hardware acceleration. BitAlign employs a systolic-array-based design to circulate the internal data (i.e., bitvectors) generated by different processing elements, which provides scalability and reduces both memory bandwidth and memory footprint. In order to handle hops (i.e., non-neighbor nodes in the graph-based reference), BitAlign provides a simple design that contains queue structures between each processing element, which store the most recently generated bitvectors.

BitAlign

After MinSeed or any seeding tool determines the subgraphs to perform alignment for each query read, for each (read, subgraph) pair, BitAlign calculates the edit distance and the corresponding alignment between the two. In order to provide an efficient, hardware-friendly, and low-cost solution, we modify the sequence alignment algorithm of GenASM, which is bitvector-based, to support sequence-to-graph alignment, and we exploit the bit-parallelism that the GenASM algorithm provides.

GenASM

GenASM makes the bitvector-based Bitap algorithm suitable for efficient hardware implementation. GenASM shares a common characteristic with the well-known DP-based algorithms: both algorithms operate on tables. The key difference between GenASM-based alignment and DP-based alignment is that cell values are bitvectors in

Running SeGraM

Call the following two functions in `src/graph.c` and `src/align.c` files in your C code, respectively, or update the existing main() function in `src/main.c` file:

```
struct SeqNode* generateGraphFromGFA(char *filename, int *numNodes, int *numEdges);
bitalign_aligner(struct SeqNode *nodes, char *pattern, int startNode, int startOffset, int endNode,
```

For example:

```
#include "graph.c"
#include "align.c"

...

int numNodes, numEdges;

struct SeqNode *nodes = generateGraphFromGFA("test.gfa", &numNodes, &numEdges);
bitalign_aligner(nodes, "ACGTCATGCAGTCGTAACGTAGTCGTCACAGTCAGTCGTAGCTAGTA", 0, 0, 3, 41, 1, 5, 200, 6

deleteGraph(nodes, numNodes);

...
```

Limitations

- The current implementation follows the divide-and-conquer approach, and does not have the ability to disable it.
- The current implementation expects the start node and the start offset within the start node, along with the possible end node and the end offset within the end node to perform the alignment. In order to run BitAlign as a standalone tool without the need of an initial seeding approach, this requirement should be removed.
- The current implementation expects the graph representation of the reference text to be generated using the `struct SeqNode* generateGraphFromGFA(char *filename, int *numNodes, int *numEdges)` function in `graph.c` file under `src/` .

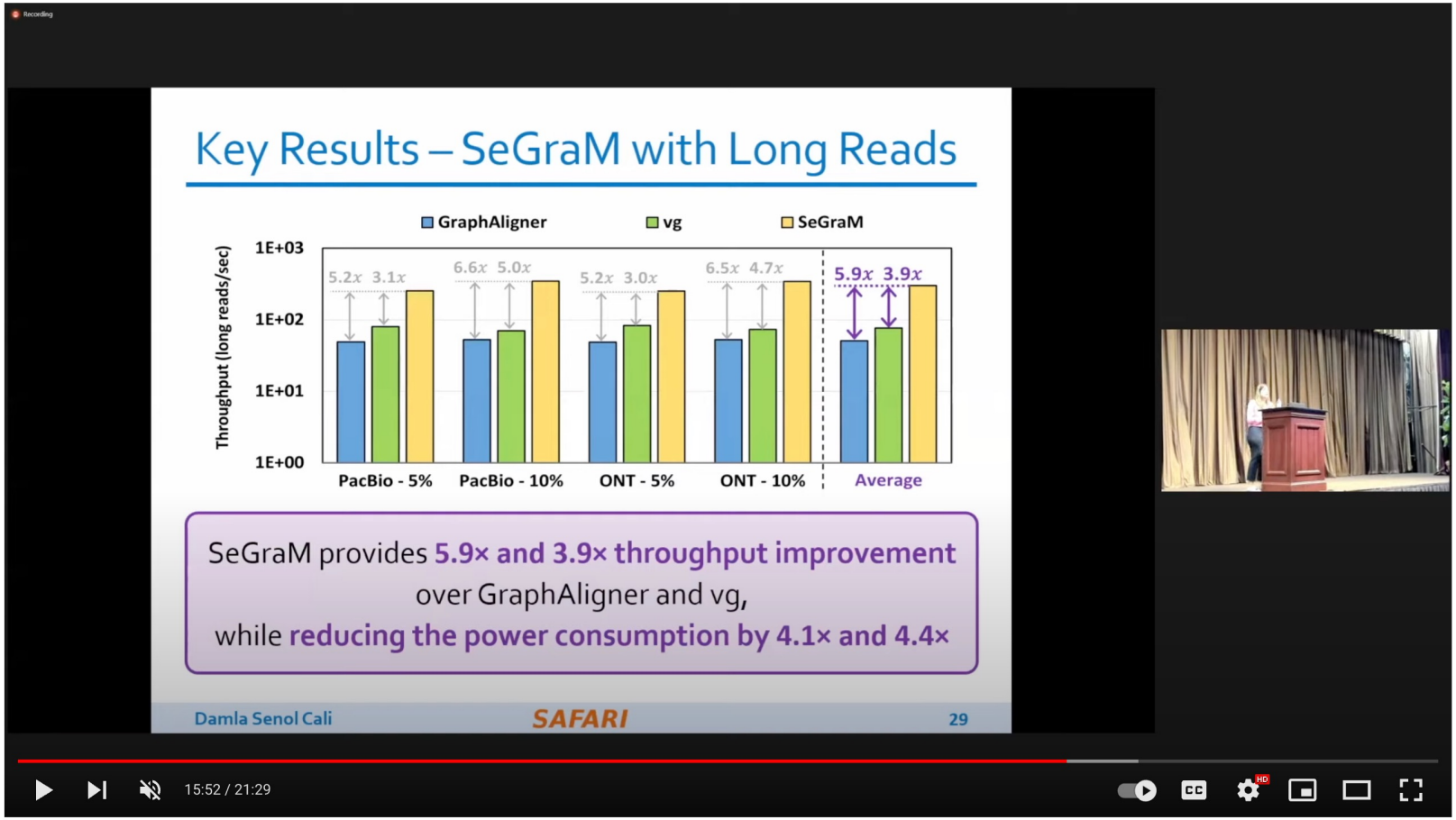
Datasets

We evaluate SeGraM using the latest major release of the human genome assembly, GRCh38, as the starting reference genome. To incorporate known genetic variations and thus form a genome graph, we use 7 VCF files for HG001-007 from the GIABproject (v3.3.2). Across the 24 graphs generated (one for each chromosome; 1–22, X, Y), in total, we have 20.4M nodes, 27.9 M edges, 3.1B sequence characters, and 7.1M variations.

For the read datasets, we generate four sets of long reads (i.e., PacBio and ONT datasets) using PBSIM2 and three sets of short reads (i.e., Illumina datasets) using Mason. For the PacBio and ONT datasets, we have reads of length 10kbp, each simulated with 5% and 10% error rates. The Illumina datasets have reads of length 100bp, 150bp, and 250bp, each simulated with a 1% error rate. Each dataset has 10,000 reads.

All our prepared datasets can be downloaded from this link. The unzipped directory has the following structure:

SeGraM Talk Video



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

Onur Mutlu Lectures
28.9K subscribers
Analytics | Edit video | 20 | Share | Download | Clip | Save

452 views 2 months ago
Talk at the 49th The International Symposium on Computer Architecture (ISCA), New York, NY, United States.
Title: SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping

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

More on SeGraM

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

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

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

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

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

Damla Senol Cali, Ph.D.

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

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

Carnegie Mellon
ETH zürich
Bilkent University
intel
University of Illinois Urbana-Champaign
SAFARI

Summary: Two Major Thrusts

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

2. AI/ML techniques for improving (memory) system designs

Challenge and Opportunity for Future

Self-Optimizing (Data-Driven) Computing Architectures

Challenge and Opportunity for Future

Accelerating Emerging Workloads

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
10 November 2022
Intel/SRC Research Program Review

SAFARI ETH zürich Carnegie Mellon

Backup Slides: SRC Task

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

Backup Slides: Pythia

Performance Improvement via Customization

- Reward value customization

- **Strict Pythia configuration**
  - **Increasing** the rewards for **no prefetching**
  - **Decreasing** the rewards for **inaccurate prefetching**

- Strict Pythia is **more conservative** in generating prefetch requests than the basic Pythia

- Evaluate on all **Ligra graph processing workloads**

More in the Paper

- Performance comparison with **unseen traces**
  - Pythia provides **equally high** performance benefits
- Comparison against **multi-level prefetchers**
  - Pythia **outperforms** prior best multi-level prefetchers
- Understanding Pythia's learning with **a case study**
  - We reason towards **the correctness** of Pythia's decision
- **Performance sensitivity** towards different features and hyperparameter values
- Detailed single-core and four-core performance

More in the Paper

- Performance comparison with unseen traces
  - Pythia provides equally high performance benefits
- Comparison against multi-level prefetchers

Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

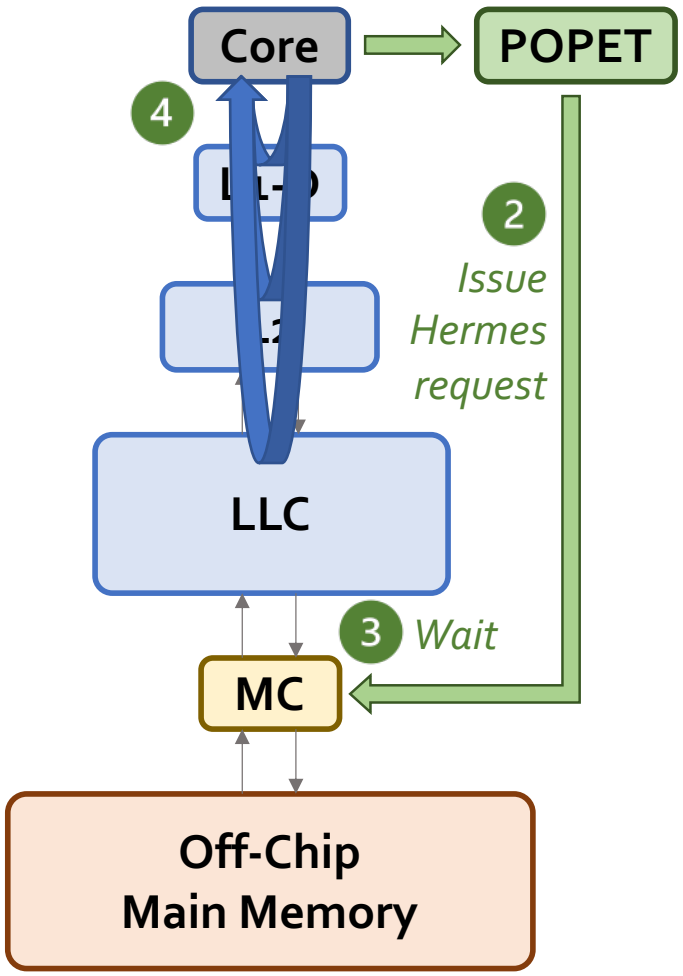
Rahul Bera[1] Konstantinos Kanellopoulos[1] Anant V. Nori[2] Taha Shahroodi[3,1]
Sreenivas Subramoney[2] Onur Mutlu[1]
[1]ETH Zürich [2]Processor Architecture Research Labs, Intel Labs [3]TU Delft

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

- Performance sensitivity towards different features and hyperparameter values
- Detailed single-core and four-core performance

Backup Slides: Hermes

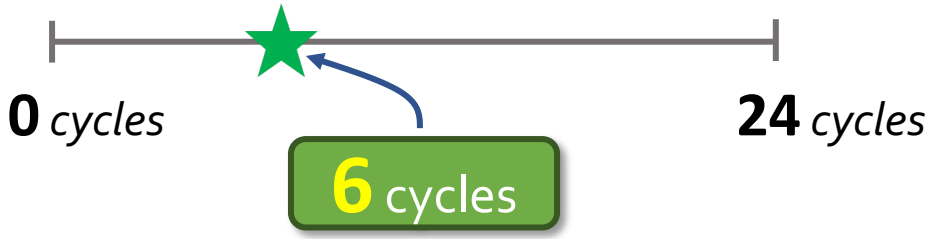
Latency Configuration



- **Cache *round-trip* latency**
  - L1-D: **5** cycles
  - L2: **15** cycles
  - LLC: **55** cycles

- **Hermes request issue latency**
  (incurred after address translation)

  Depends on
  - **Interconnect** between POPET and MC



More in the Paper

- Performance sensitivity to:
  - Cache hierarchy access latency
  - Hermes request issue latency
  - Activation threshold
  - ROB size (in extended version on arXiv)
  - LLC size (in extended version on arXiv)
- **Accuracy, coverage, and performance analysis** against **HMP** and **TTP**
- Understanding **usefulness of each program feature**
- Effect on **stall cycle reduction**
- **Performance** analysis on an **eight-core** system

Backup Slides: SeGraM

Genome Graphs

Genome graphs:

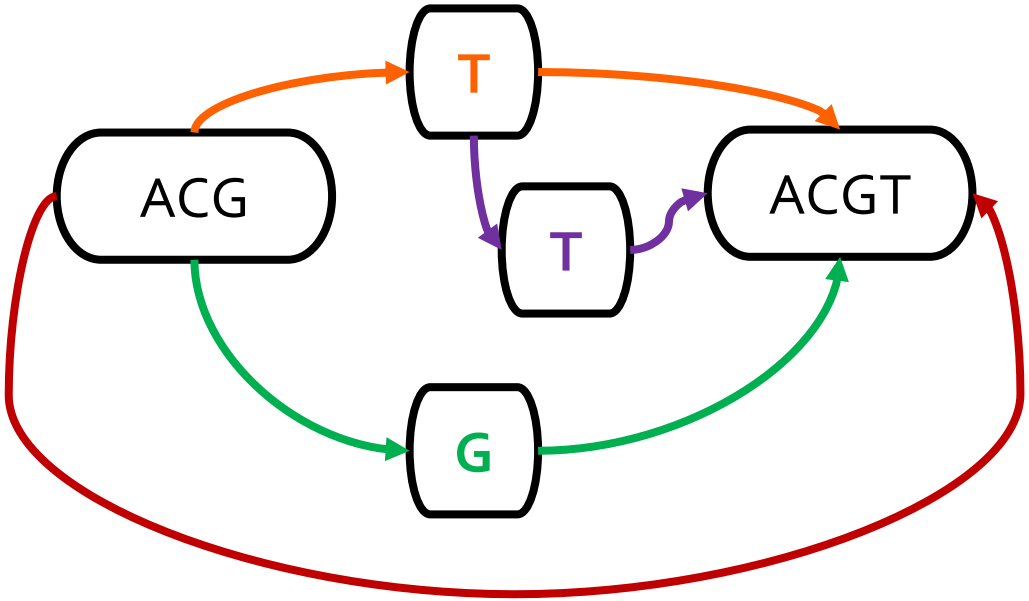
- Combine the linear reference genome with the known genetic variations in the entire population as a graph-based data structure
- Enable us to move away from aligning with a single linear reference genome (reference bias) and more accurately express the genetic diversity in a population

Sequence #1: ACGTACGT

Sequence #2: ACGGACGT

Sequence #3: ACGTTACGT

Sequence #4: ACGACGT



Analysis of State-of-the-Art Tools

SW

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

Observation 1: Alignment step is the bottleneck

Observation 2: Alignment suffers from high cache miss rates

Observation 3: Seeding suffers from the DRAM latency bottleneck

Observation 4: Baseline tools scale sublinearly

HW

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

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

SeGraM: Universal Genomic Mapping Accelerator

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

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

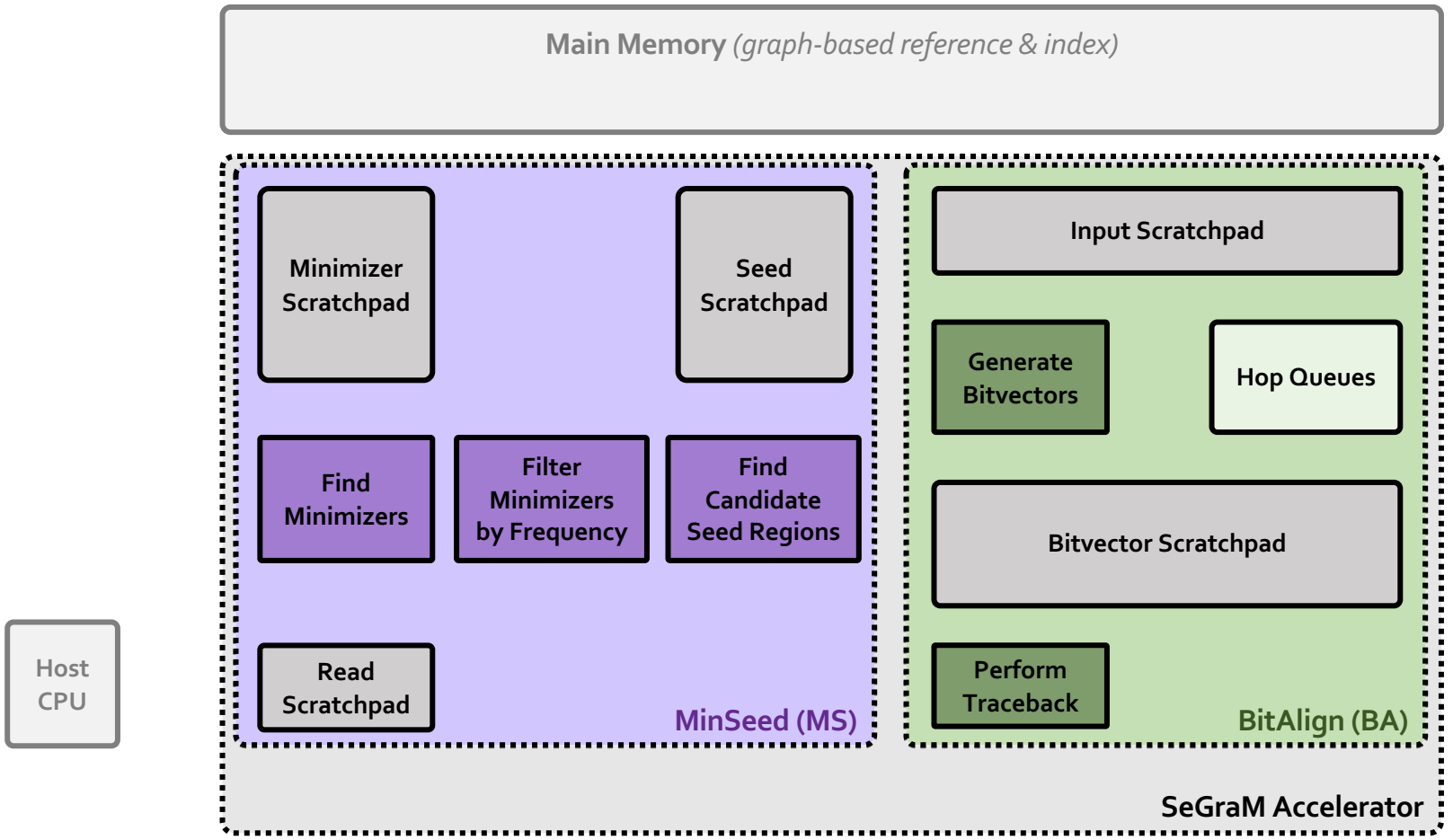
- We base SeGraM upon a minimizer-based seeding algorithm
- We propose a novel bitvector-based alignment algorithm to perform approximate string matching between a read and a graph-based reference genome

SW

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

HW

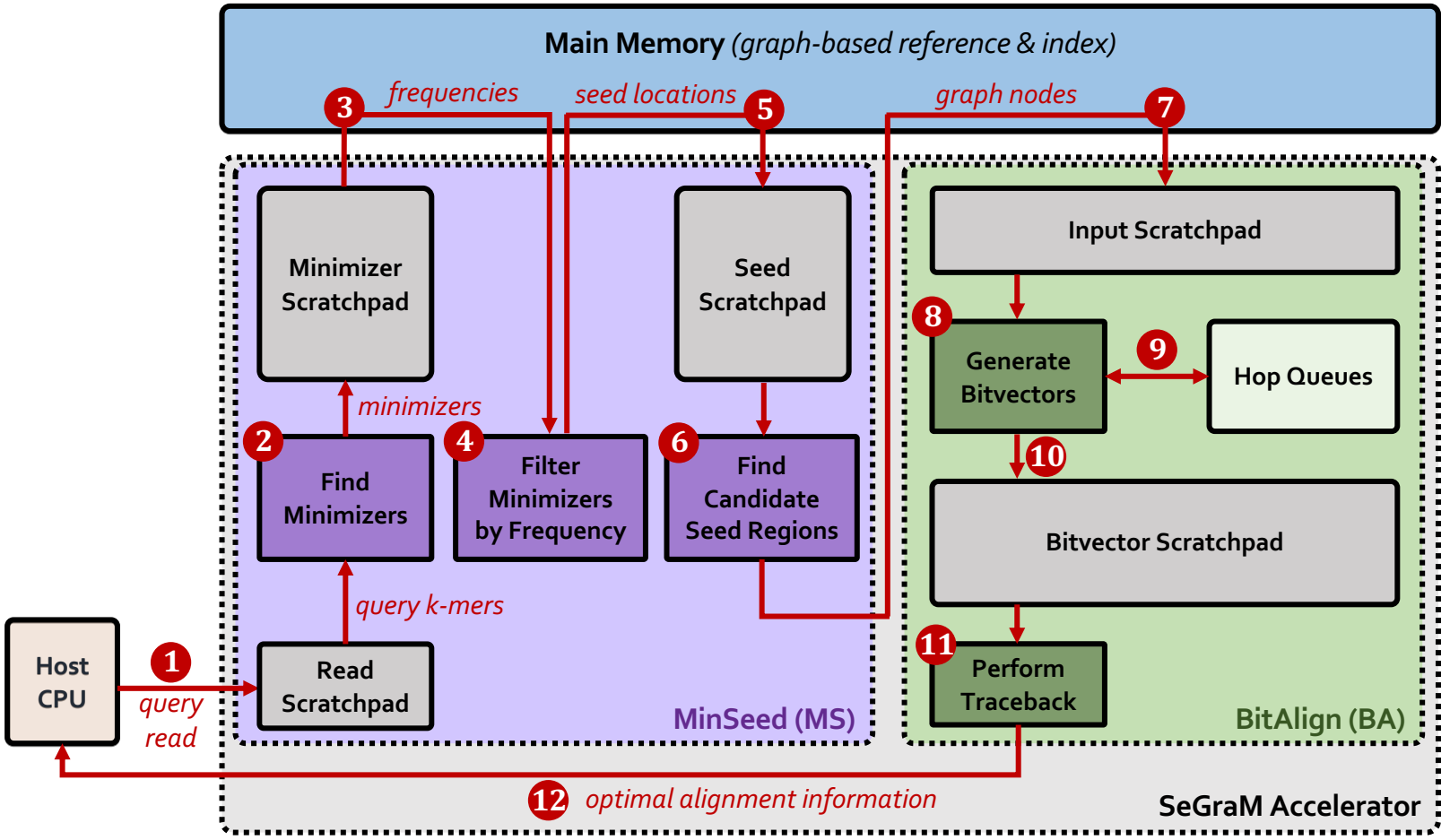
SeGraM Hardware Design



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

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

SeGraM Hardware Design

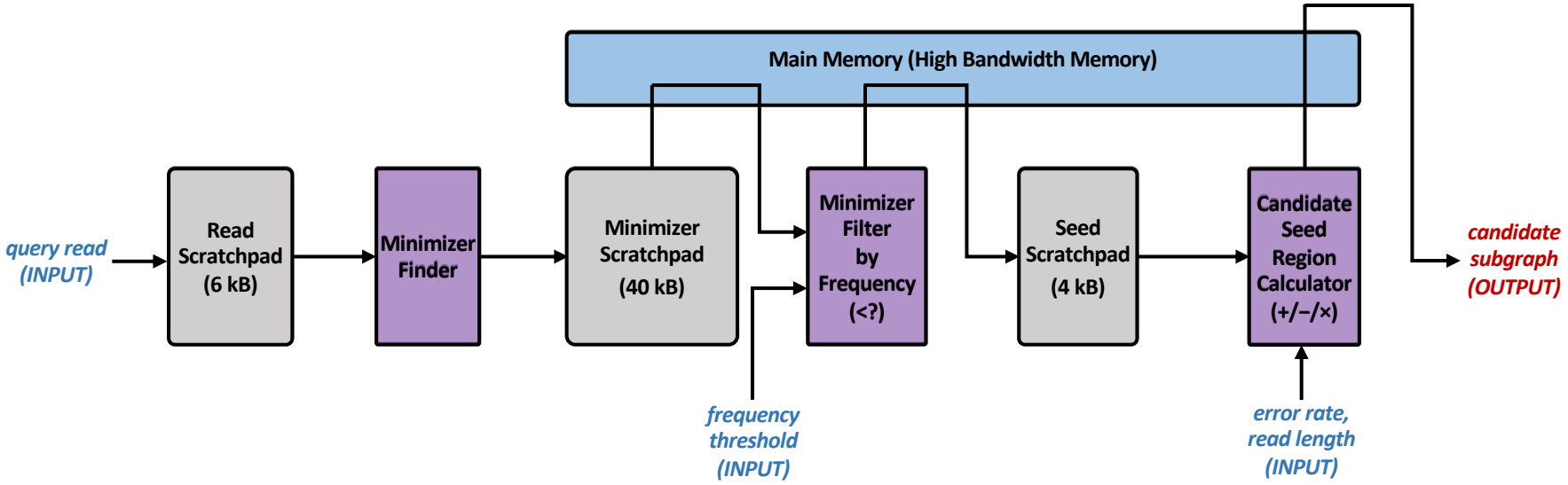


MinSeed: *first* hardware accelerator for **Min**imizer-based **Seed**ing

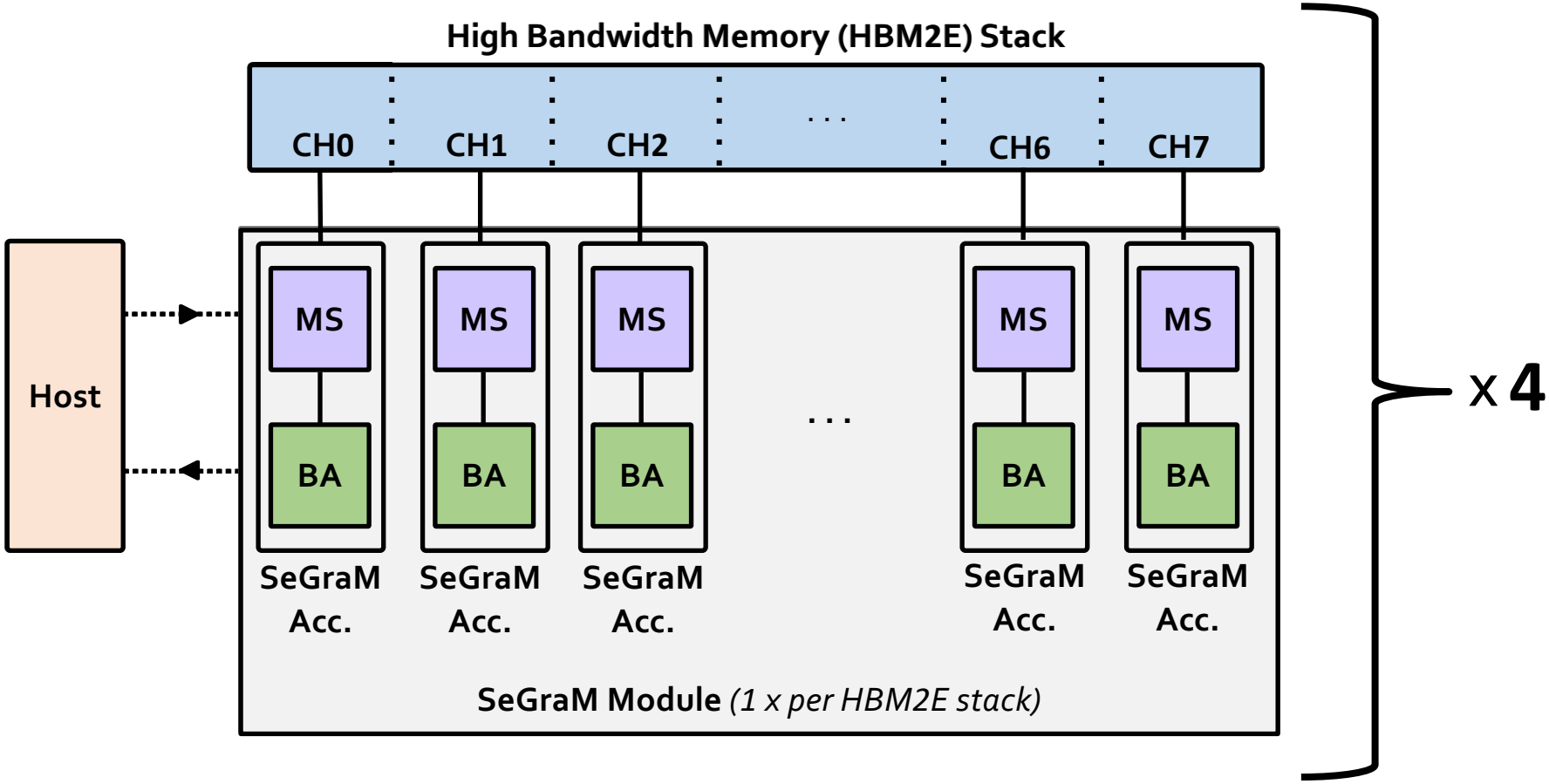
BitAlign: *first* hardware accelerator for (**Bit**vector-based) sequence-to-graph **Align**ment

MinSeed HW

- MinSeed = 3 computation modules + 3 scratchpads + memory interface
  - Computation modules: Implemented with simple logic
  - Scratchpads: 50kB in total; employ double buffering technique to hide the latency of MinSeed
  - High-Bandwidth Memory (HBM): Enables low-latency and highly-parallel memory access

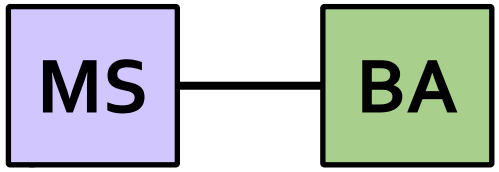


Overall System Design of SeGraM

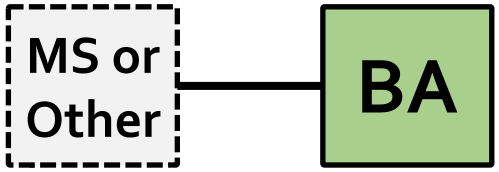


Use Cases of SeGraM

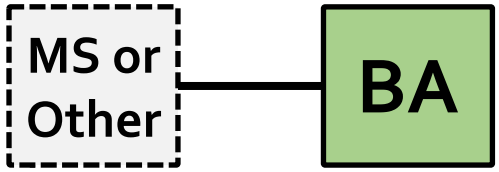
(1) Sequence-to-Graph Mapping



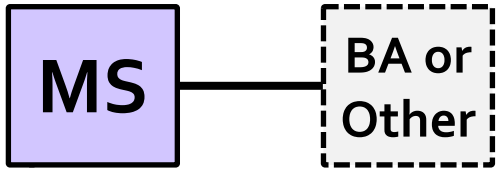
(2) Sequence-to-Graph Alignment



(3) Sequence-to-Sequence Alignment



(4) Seeding

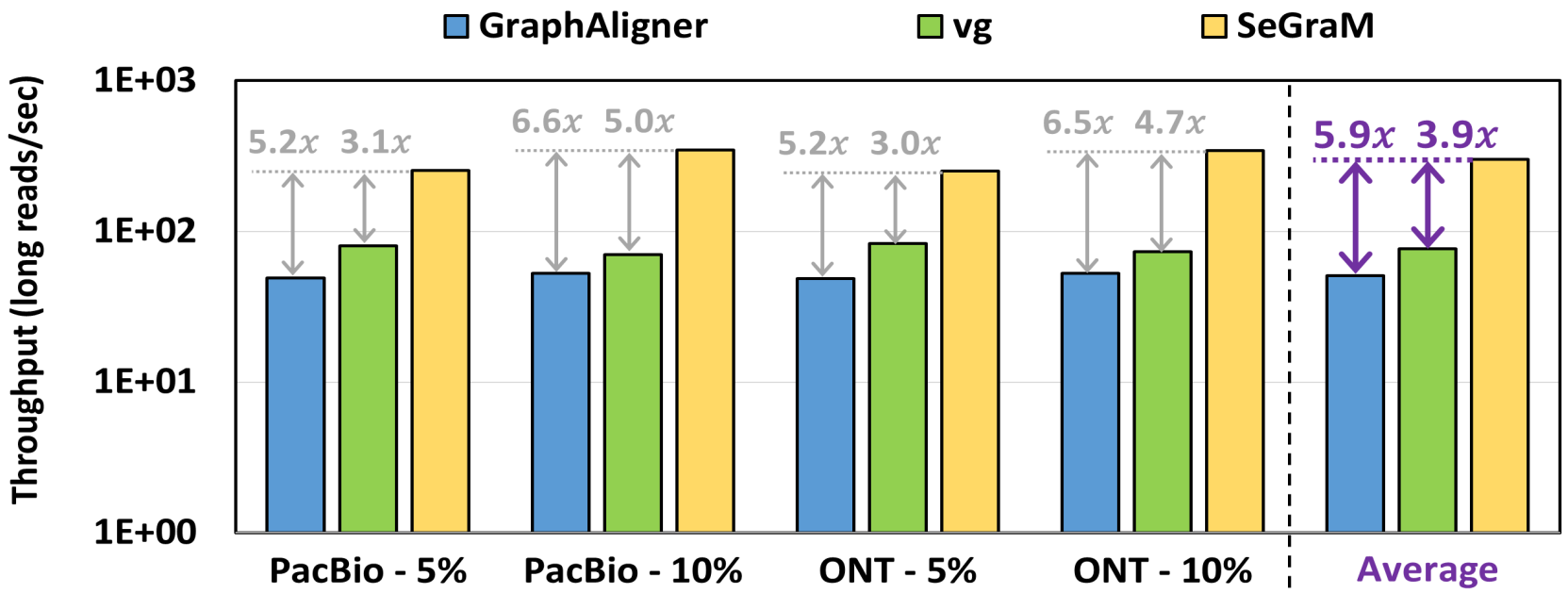


Key Results – Area & Power

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

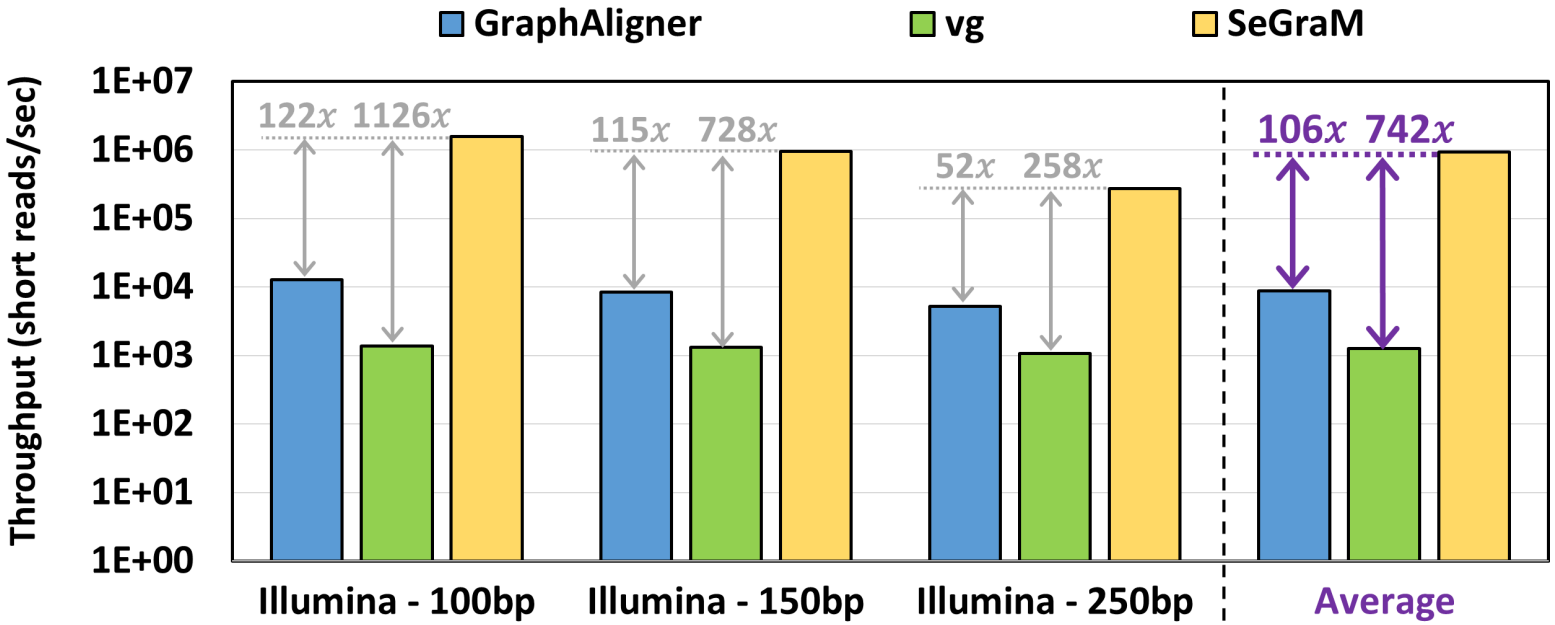
Component	Area (mm^2)	Power (mW)
MinSeed – Logic	0.017	10.8
Read Scratchpad (6 kB)	0.012	7.9
Minimizer Scratchpad (40 kB)	0.055	22.7
Seed Scratchpad (4 kB)	0.008	6.4
BitAlign – Edit Distance Calculation Logic with Hop Queue Registers (64 PEs)	0.393	378.0
BitAlign – Traceback Logic	0.020	2.7
Input Scratchpad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

Key Results – SeGraM with Long Reads



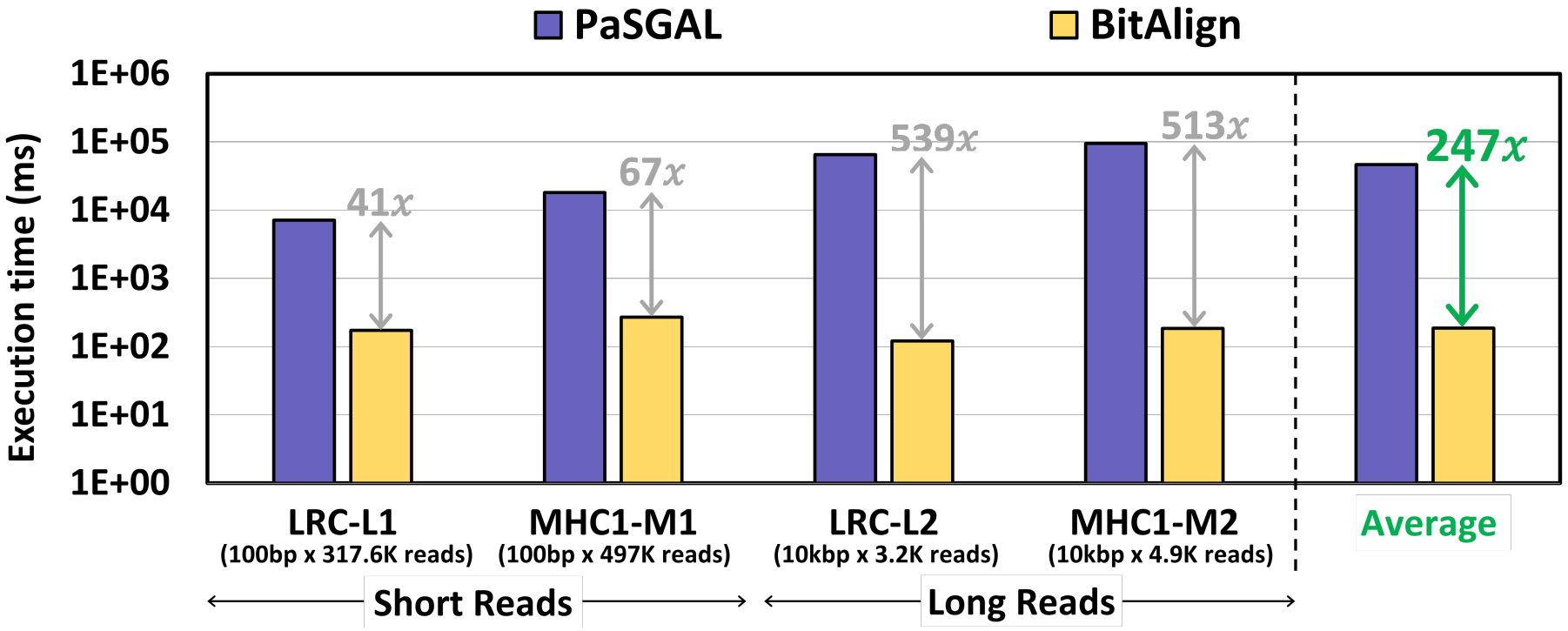
SeGraM provides **5.9× and 3.9× throughput improvement** over GraphAligner and vg, while **reducing the power consumption by 4.1× and 4.4×**

Key Results – SeGraM with Short Reads



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

Key Results – BitAlign (S2G Alignment)



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

Conclusion

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