

ML-Assisted Memory & Storage Management

Onur Mutlu

omutlu@gmail.com

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

21 May 2025

Stanford AI-Boosted Chip Design Lecture

SAFARI

ETH zürich

Data-Driven (Self-Optimizing) Architectures

System Architecture Design Today

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

**Can we design
fundamentally intelligent architectures?**

An Intelligent Architecture

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

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

Self-Optimizing Memory Controllers

- Engin Ipek, Onur Mutlu, José F. Martínez, and Rich Caruana,
"Self Optimizing Memory Controllers: A Reinforcement Learning Approach"
Proceedings of the 35th International Symposium on Computer Architecture (ISCA), pages 39-50, Beijing, China, June 2008.
Selected to the ISCA-50 25-Year Retrospective Issue covering 1996-2020 in 2023 (Retrospective (pdf) Full Issue).

Self-Optimizing Memory Controllers: A Reinforcement Learning Approach

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

¹Cornell University, Ithaca, NY 14850 USA

²Microsoft Research, Redmond, WA 98052 USA

Self-Optimizing Memory Prefetchers

Rahul Bera, Konstantinos Kanellopoulos, Anant Nori, Taha Shahroodi, Sreenivas Subramoney, and Onur Mutlu,
"Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning"
Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.

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

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

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

[[Talk Video](#) (20 minutes)]

[[Lightning Talk Video](#) (1.5 minutes)]

[[Pythia Source Code](#) (Officially Artifact Evaluated with All Badges)]

[[arXiv version](#)]

Officially artifact evaluated as available, reusable and reproducible.



Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera¹

Konstantinos Kanellopoulos¹

Anant V. Nori²

Taha Shahroodi^{3,1}

Sreenivas Subramoney²

Onur Mutlu¹

¹ETH Zürich

²Processor Architecture Research Labs, Intel Labs

³TU Delft

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

Learning-Based Off-Chip Load Predictors

- Rahul Bera, Konstantinos Kanellopoulos, Shankar Balachandran, David Novo, Ataberk Olgun, Mohammad Sadrosadati, and Onur Mutlu,
"Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction"

Proceedings of the 55th International Symposium on Microarchitecture (MICRO), Chicago, IL, USA, October 2022.

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

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

[[Talk Video](#) (12 minutes)]

[[Lecture Video](#) (25 minutes)]

[[arXiv version](#)]

[[Source Code \(Officially Artifact Evaluated with All Badges\)](#)]

Officially artifact evaluated as available, reusable and reproducible.

Best paper award at MICRO 2022.



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

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

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

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

Self-Optimizing 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

¹ETH Zürich

²Eindhoven University of Technology

³LIRMM, Univ. Montpellier, CNRS

A Blueprint for Fundamentally Better Architectures

- Onur Mutlu,
"Intelligent Architectures for Intelligent Computing Systems"
*Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (**DATE**), Virtual, February 2021.*
[Slides (pptx) (pdf)]
[IEDM Tutorial Slides (pptx) (pdf)]
[Short DATE Talk Video (11 minutes)]
[Longer IEDM Tutorial Video (1 hr 51 minutes)]

Intelligent Architectures for Intelligent Computing Systems

Onur Mutlu
ETH Zurich
omutlu@gmail.com

Data-centric

Data-driven

Data-aware



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 Bera¹

Konstantinos Kanellopoulos¹

Anant V. Nori²

Taha Shahroodi^{3,1}

Sreenivas Subramoney²

Onur Mutlu¹

¹ETH Zürich

²Processor Architecture Research Labs, Intel Labs

³TU Delft

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



Pythia

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

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

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



1

Mainly use **one**
program context info.
for prediction

2

Lack **inherent system**
awareness

3

Lack **in-silicon**
customizability



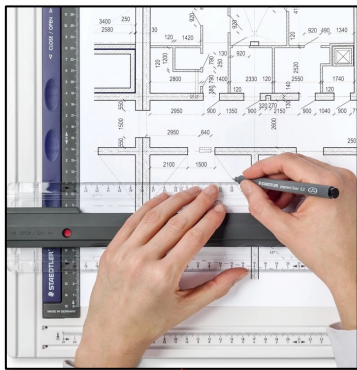
Why do prefetchers
not perform well?



Lack of In-silicon Customizability

- Feature **statically** selected at design time
 - **Rigid hardware** designed specifically to exploit that feature
- **No way to change** program feature and/or change prefetcher's objective **in silicon**
 - **Cannot adapt** to a wide range of workload demands

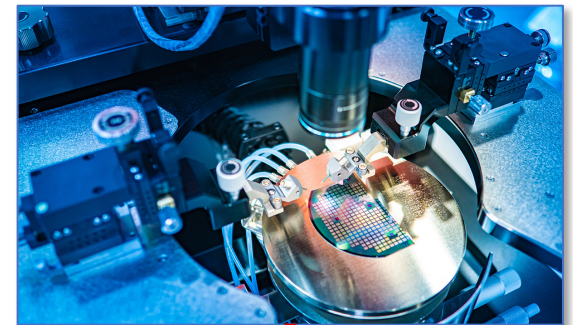
Design from scratch



Verify



Fabricate



Our Goal

A **prefetching framework** that can:

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

Our Proposal



Pythia

Formulates prefetching as a
reinforcement learning problem

Basics of Reinforcement Learning (RL)

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

Agent

Environment

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

Formulating Prefetching as RL

What is State?

- **k -dimensional** vector of features

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

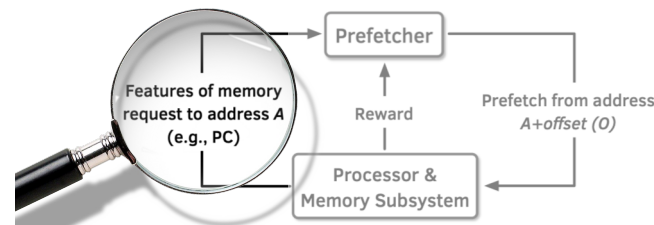
- Feature = control-flow + data-flow

- **Control-flow examples**

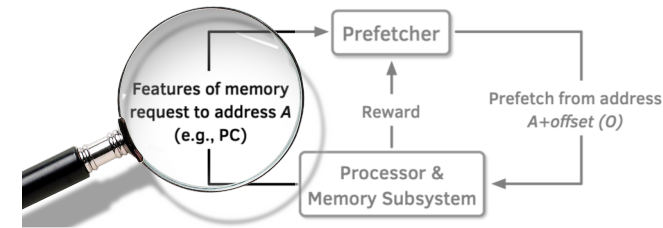
- PC
- Branch PC
- Last-3 PCs, ...

- **Data-flow examples**

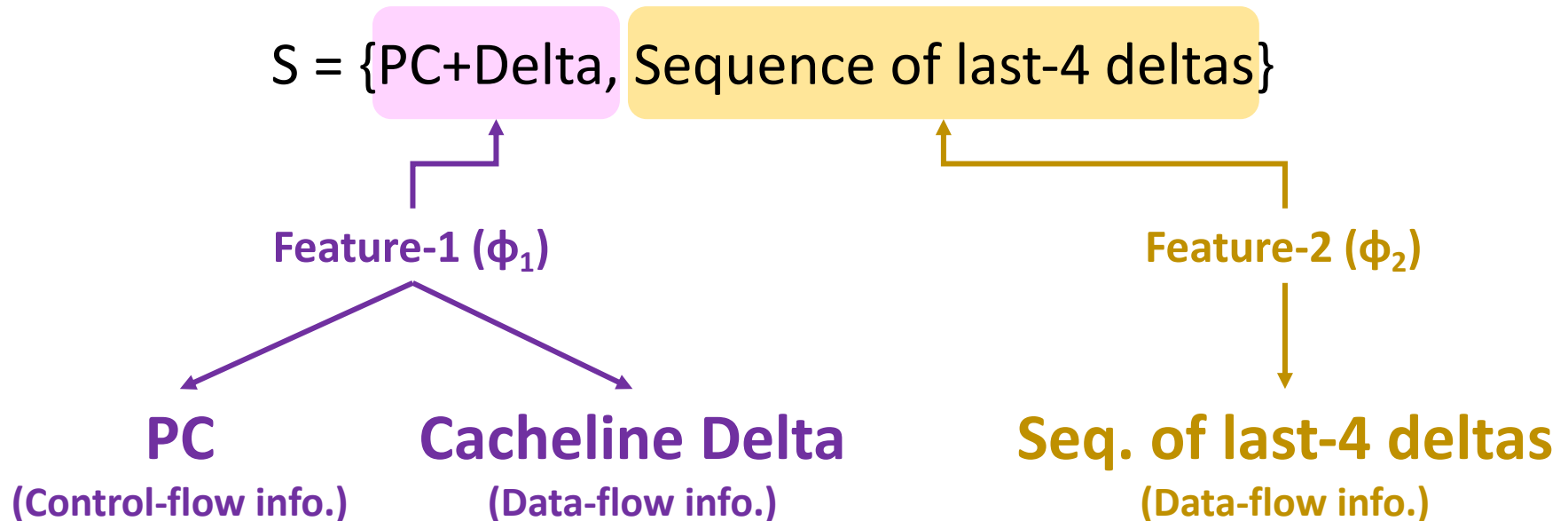
- Cacheline address
- Physical page number
- Delta between two cacheline addresses
- Last 4 deltas, ...



What is State?

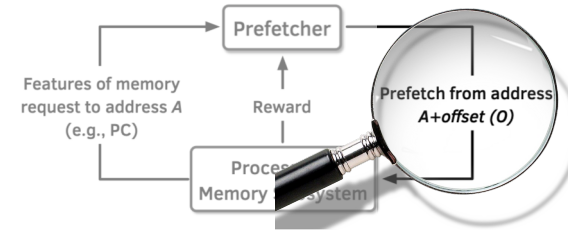


Example of state information



What is Action?

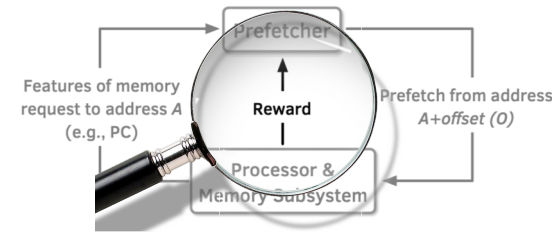
Given a demand access to address A
the action is to **select prefetch offset “0”**



- **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



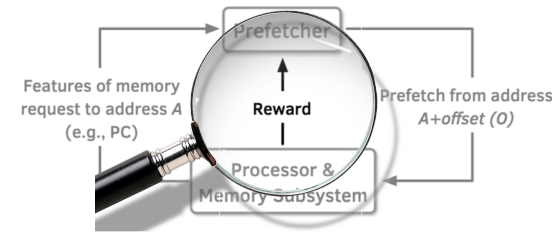
What is Reward?

- **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})

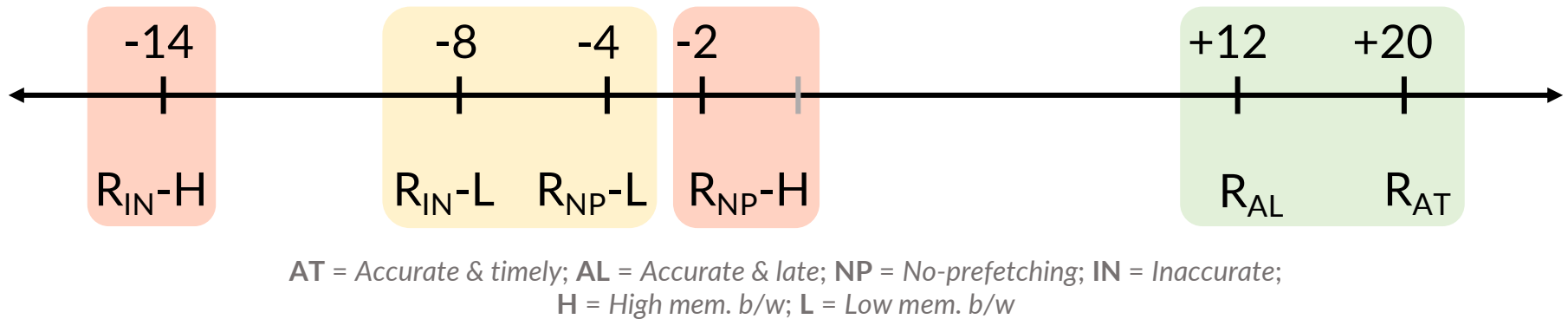
- Values are set at design time via **automatic design-space exploration**

- Can be **customized** further in silicon for higher performance



Steering Pythia's Objective via Reward Values

- Example reward configuration for
 - Generating **accurate prefetches**
 - Making **bandwidth-aware** prefetch decisions



1

Highly prefers to generate accurate prefetches

2

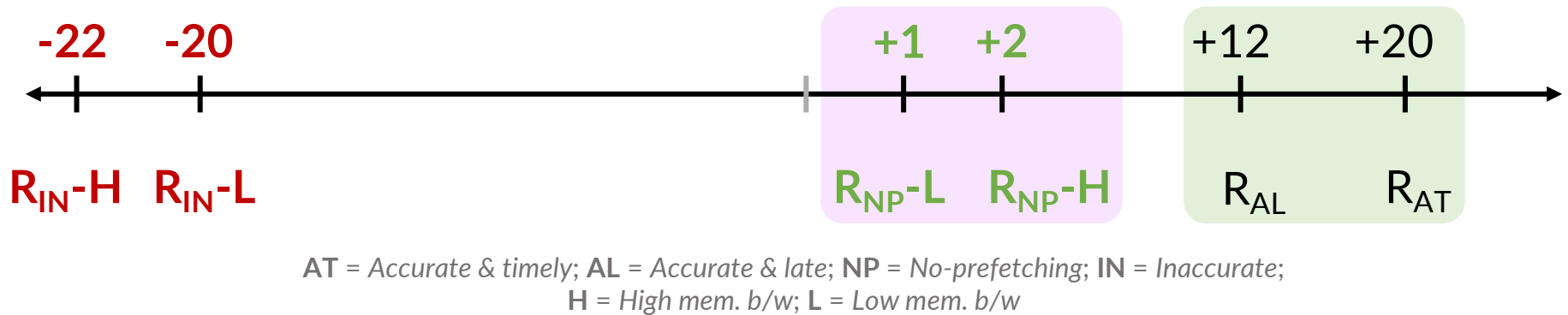
Prefers not to prefetch if memory bandwidth usage is low

3

Strongly prefers not to prefetch if memory bandwidth usage is high

Steering Pythia's Objective via Reward Values

- Customizing reward values to make Pythia **conservative** towards prefetching



1

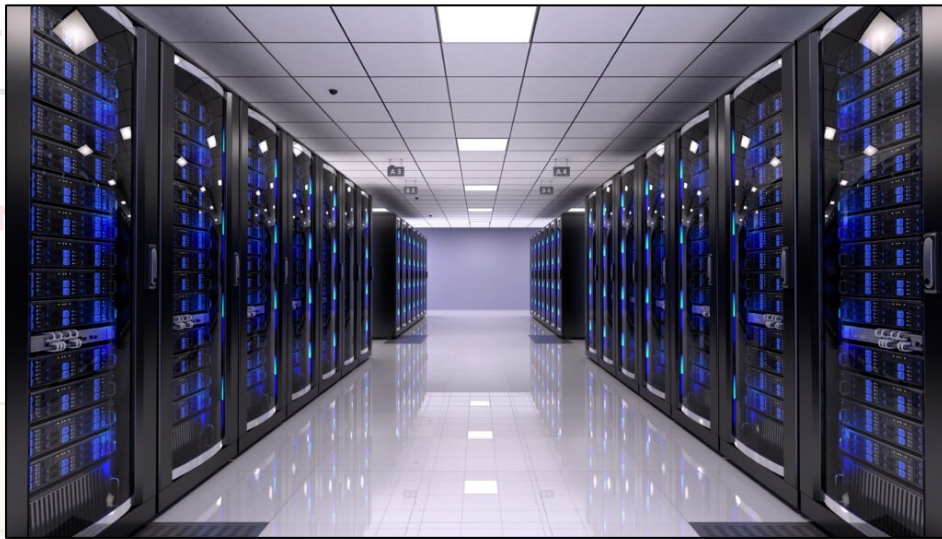
Highly prefers to generate accurate prefetches

2

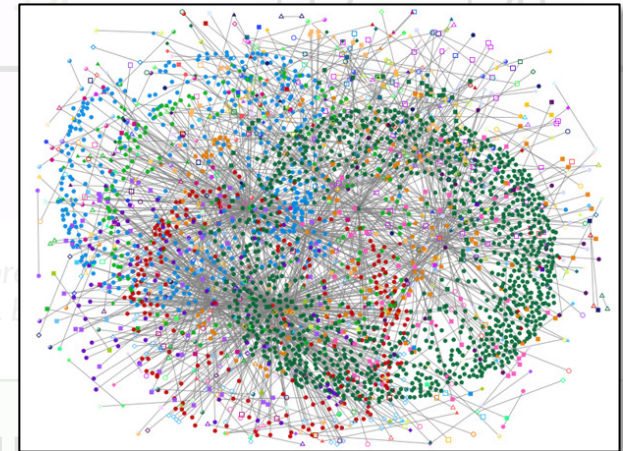
Otherwise prefers not to prefetch

Steering Pythia's Objective via Reward Values

Strict Pythia configuration



Server-class processors



Bandwidth-sensitive workloads

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

List of Evaluated Features

Table 3: List of program control-flow and data-flow components used to derive the list of features for exploration

Control-flow Component	Data-flow Component
(1) PC of load request	(1) Load cacheline address
(2) PC-path (XOR-ed last-3 PCs)	(2) Page number
(3) PC XOR-ed branch-PC	(3) Page offset
(4) None	(4) Load address delta
	(5) Sequence of last-4 offsets
	(6) Sequence of last-4 deltas
	(7) Offset XOR-ed with delta
	(8) None

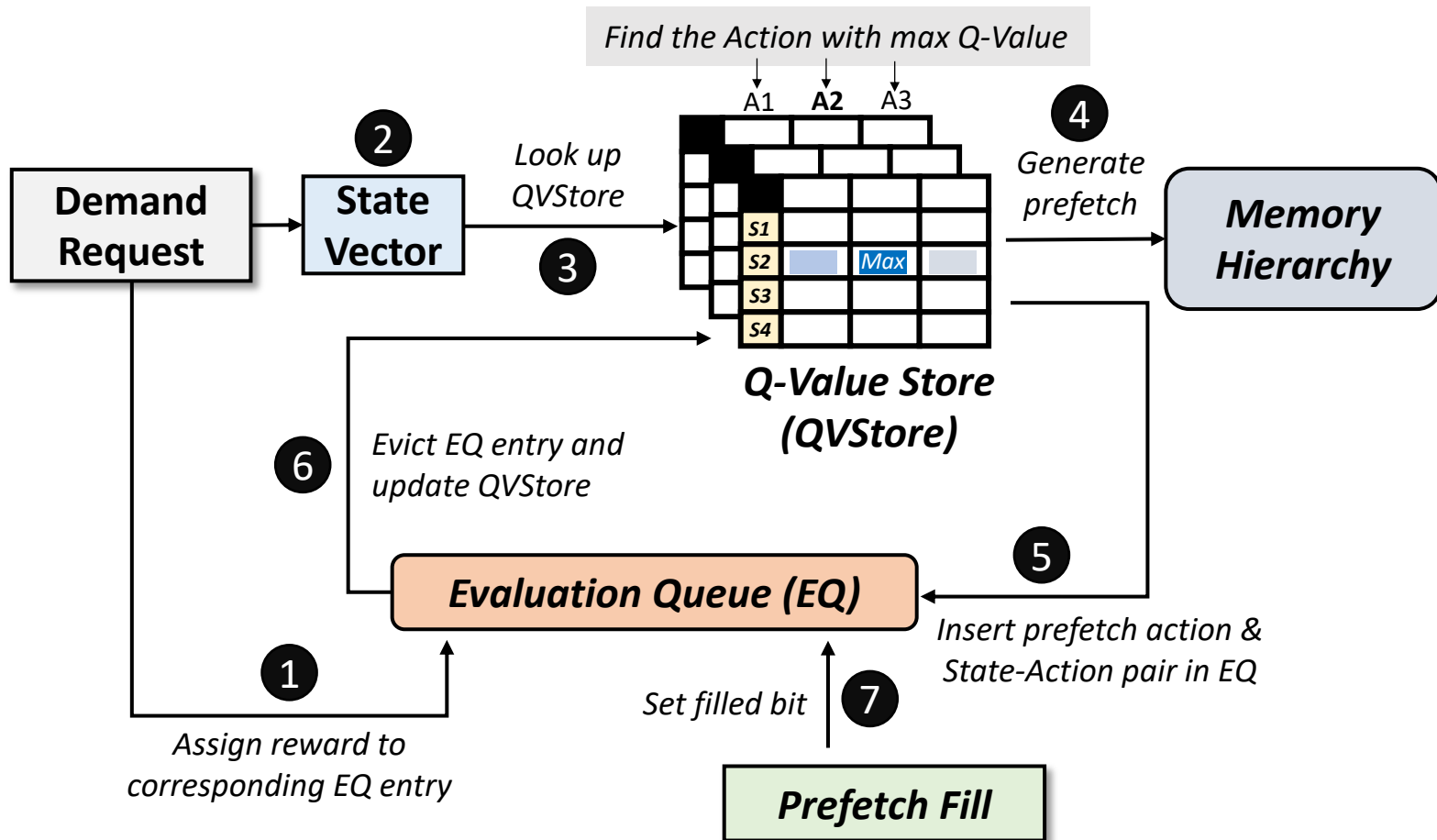
Basic Pythia Configuration

Table 2: Basic Pythia configuration derived from our automated design-space exploration

Features	PC+Delta, Sequence of last-4 deltas
Prefetch Action List	$\{-6, -3, -1, 0, 1, 3, 4, 5, 10, 11, 12, 16, 22, 23, 30, 32\}$
Reward Level Values	$\mathcal{R}_{AT}=20, \mathcal{R}_{AL}=12, \mathcal{R}_{CL}=-12, \mathcal{R}_{IN}^H=-14,$ $\mathcal{R}_{IN}^L=-8, \mathcal{R}_{NP}^H=-2, \mathcal{R}_{NP}^L=-4$
Hyperparameters	$\alpha = 0.0065, \gamma = 0.556, \epsilon = 0.002$

More Detailed Pythia Overview

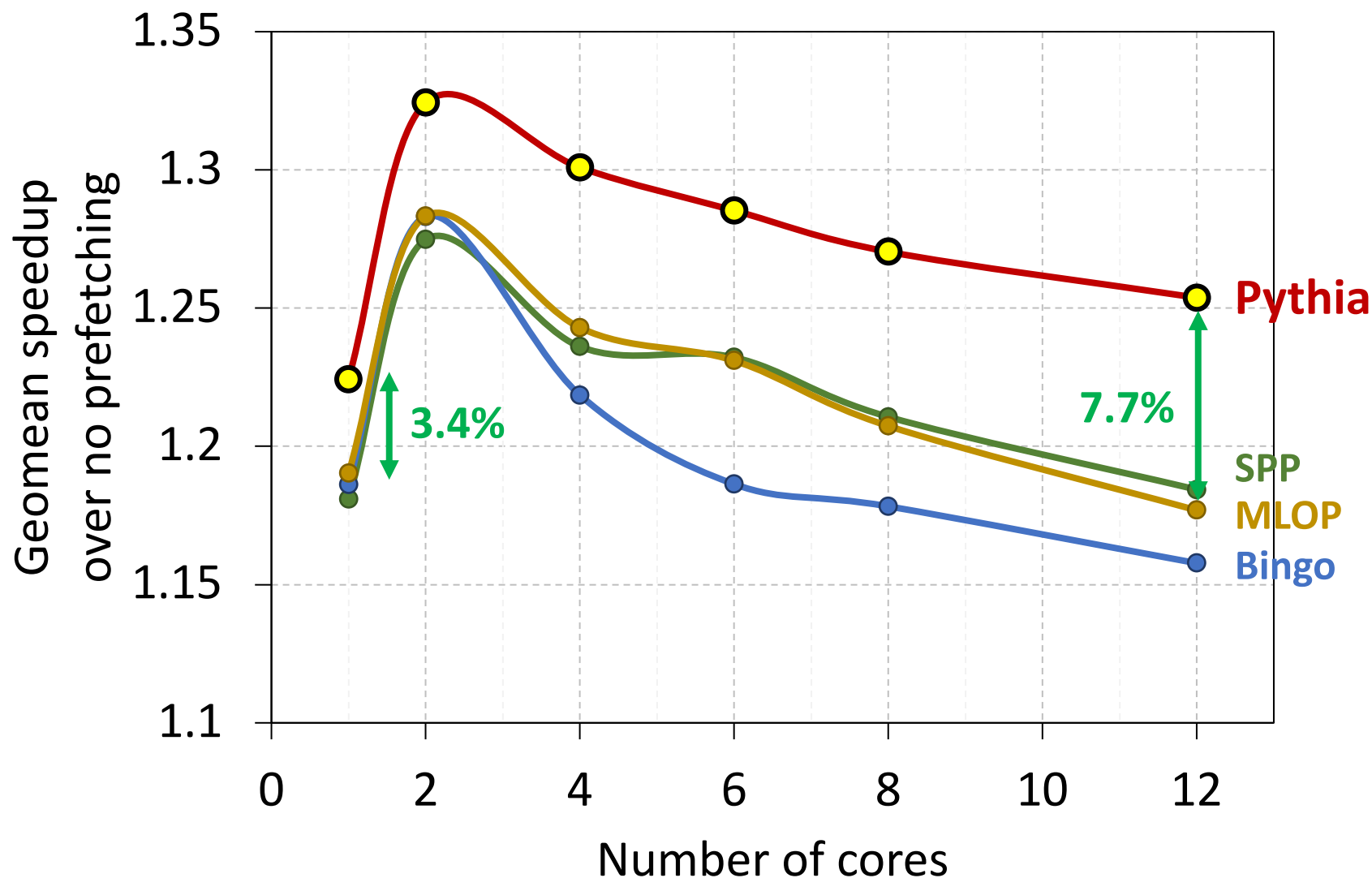
- **Q-Value Store**: Records Q-values for *all* state-action pairs
- **Evaluation Queue**: A FIFO queue of recently-taken actions



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]

Performance with Varying Core Count



Performance with Varying Core Count

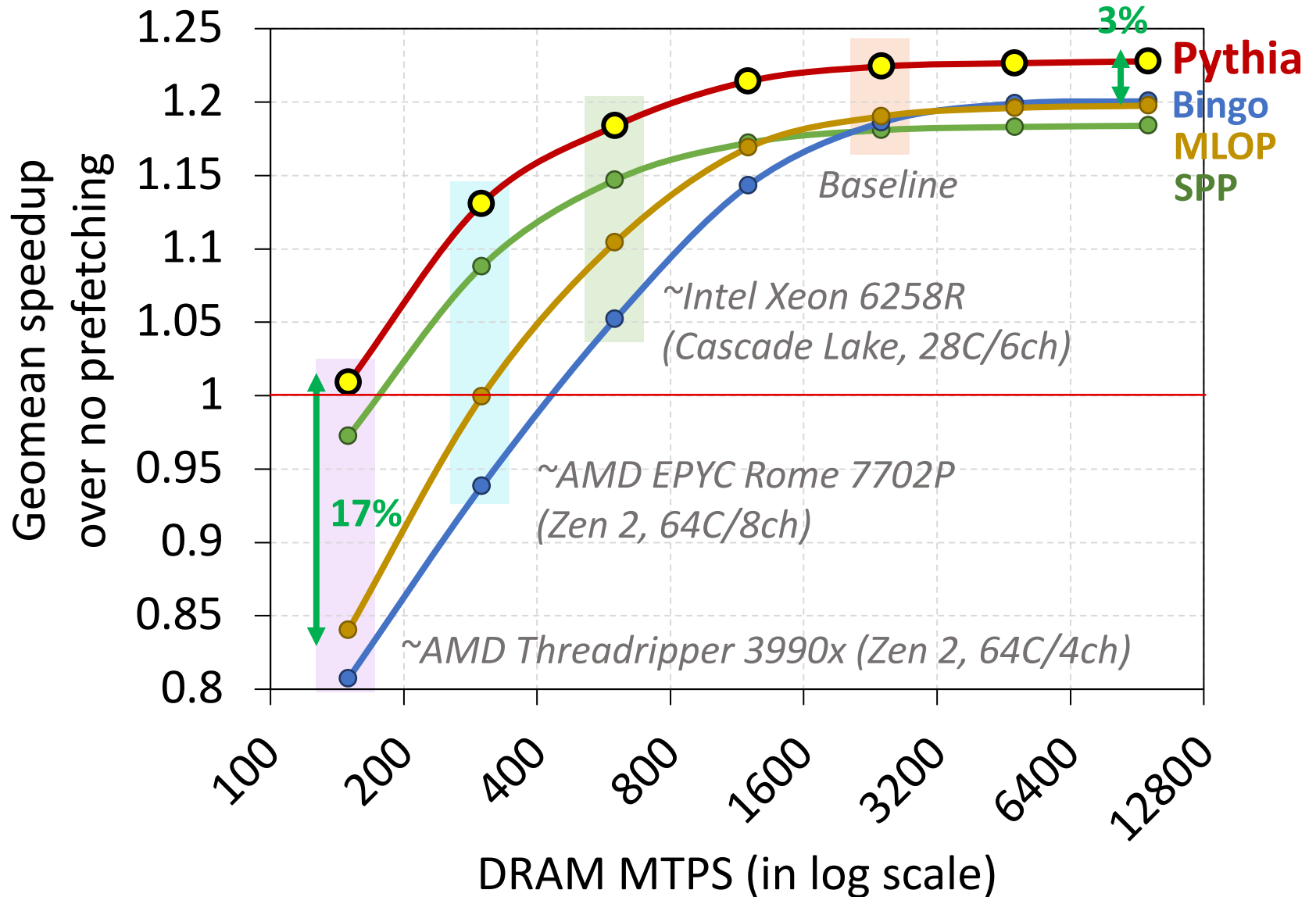


The graph shows performance on the y-axis (ranging from 1.1 to 1.35) against the number of cores on the x-axis (ranging from 0 to 12). Pythia (red line) starts at ~1.28 at 2 cores, peaks at ~1.32 at 4 cores, and then declines. Other models (blue, green, orange lines) show lower performance, with a green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

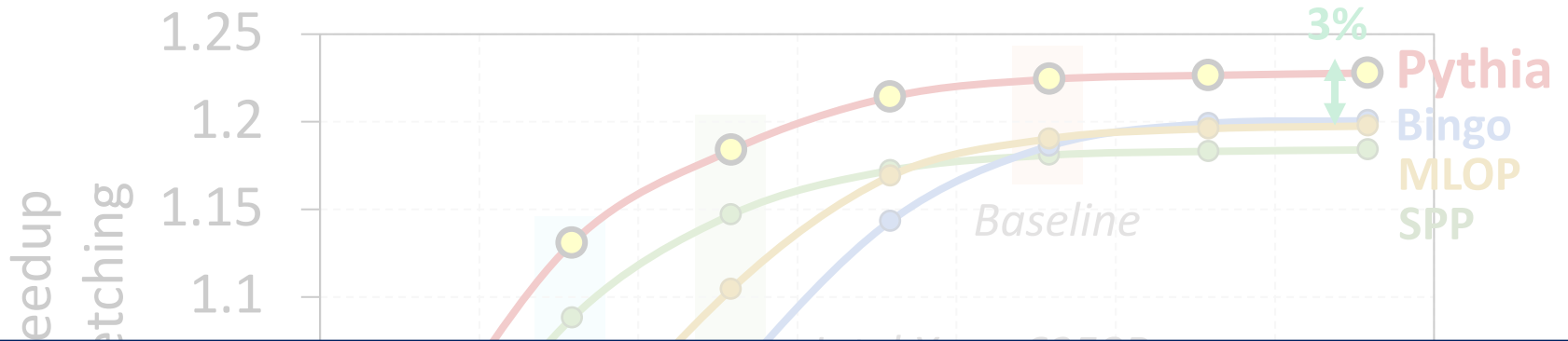
1. Pythia consistently provides the highest performance in **all core configurations**

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

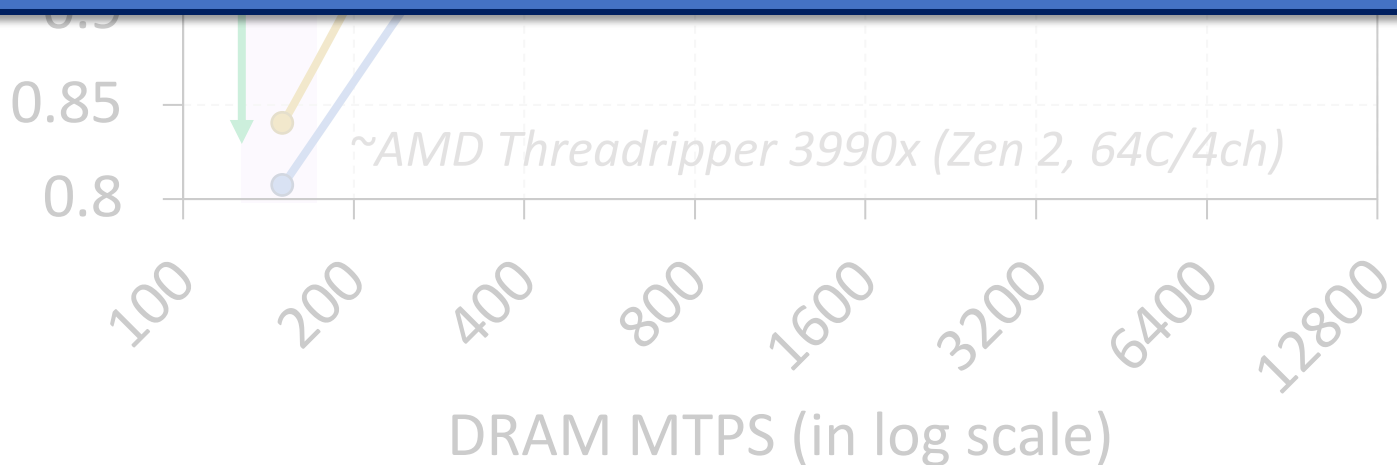
Performance with Varying DRAM Bandwidth



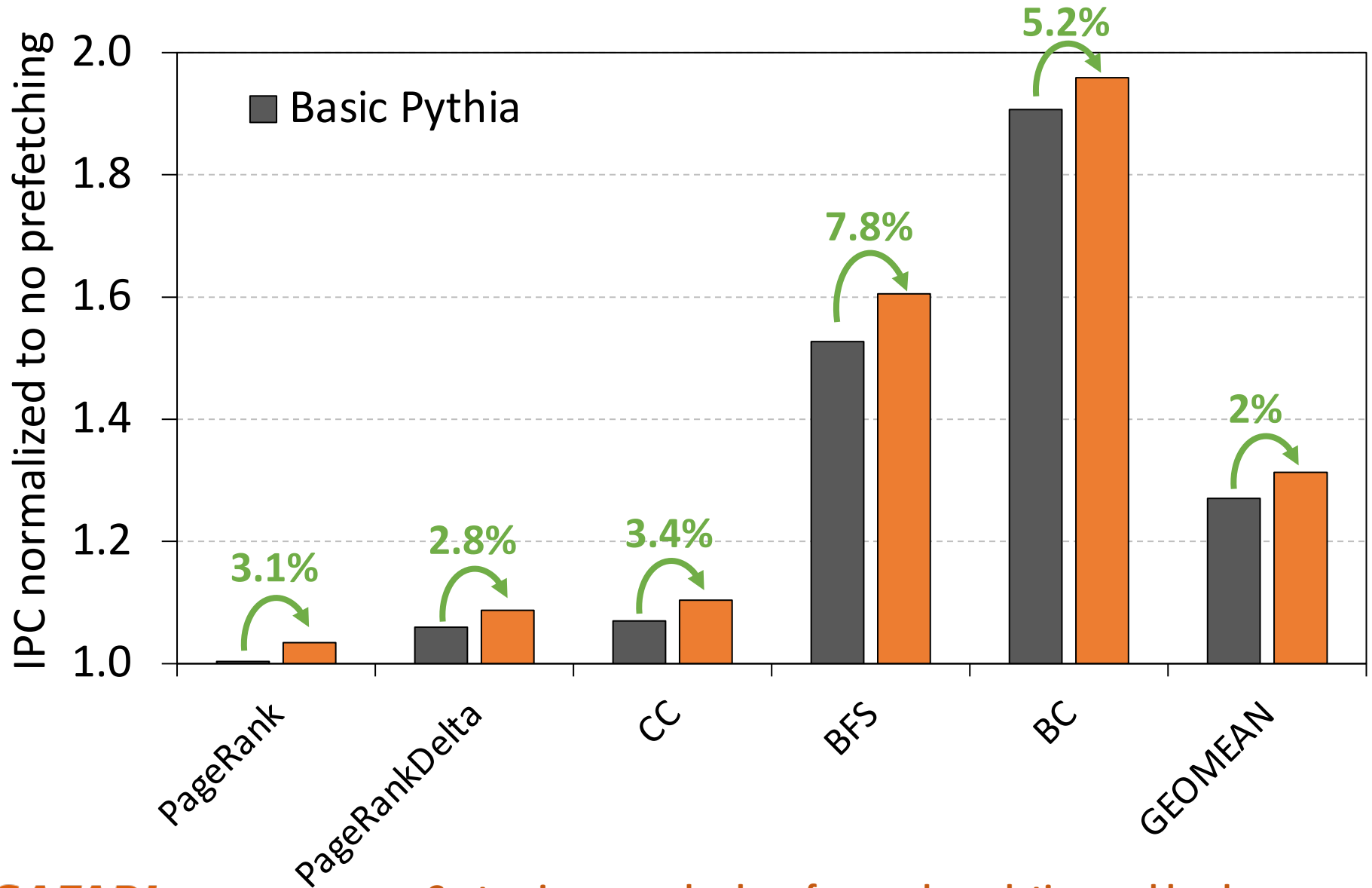
Performance with Varying DRAM Bandwidth



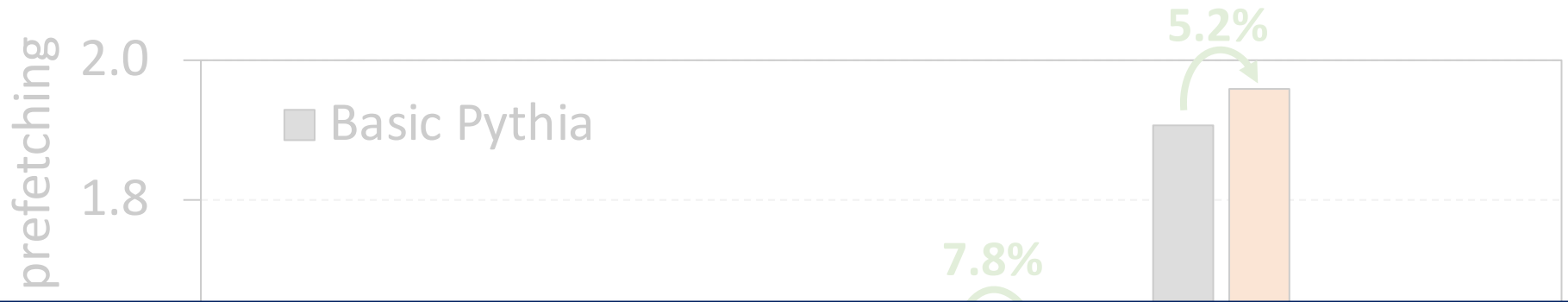
Pythia outperforms prior best prefetchers for a wide range of DRAM bandwidth configurations



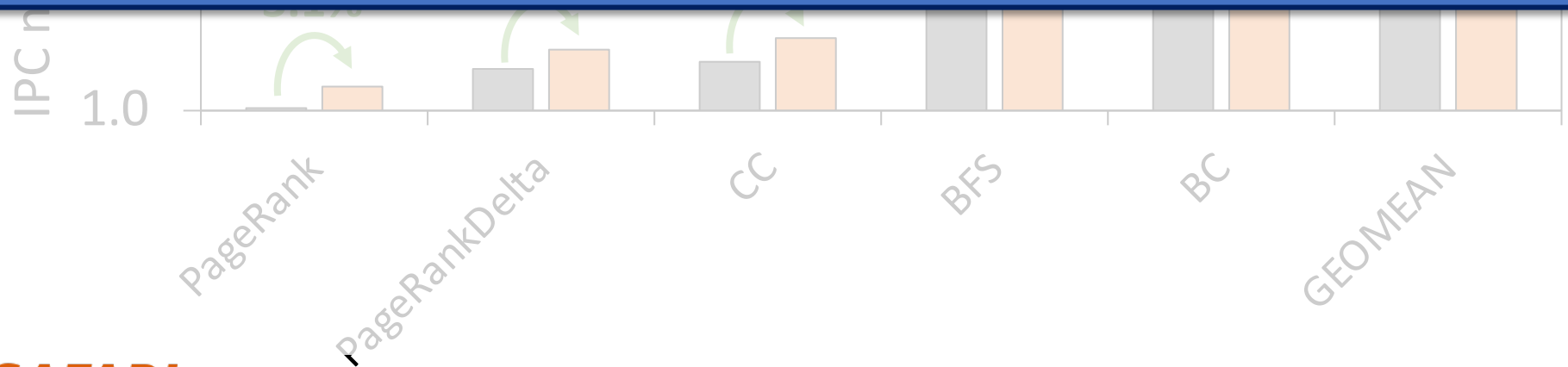
Performance Improvement via Customization



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



1.03% area overhead



0.4% power overhead



Satisfies prediction latency

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

Pythia is Open Source



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

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

The screenshot shows the GitHub repository for CMU-SAFARI/Pythia. The repository is public and has 3 unwatchers, 9 stars, and 2 forks. It has 1 branch and 5 tags. The repository is managed by rahulbera. The file list includes: branch, config, docs, experiments, inc, prefetcher, replacement, scripts, src, tracer, .gitignore, CITATION.cff, LICENSE, and LICENSE.champsim. The right sidebar shows the repository description, a link to the arXiv paper, and a list of related topics: machine-learning, reinforcement-learning, computer-architecture, prefetcher, microarchitecture, cache-replacement, branch-predictor, champsim-simulator, and champsim-tracer. The releases section shows v1.3 as the latest release, 21 days ago.

CMU-SAFARI/Pythia Public

Unwatch 3 Star 9 Fork 2

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 5 tags Go to file Add file Code

rahulbera Github pages documentation ✓ d1efc65 7 hours ago 40 commits

branch	Initial commit for MICRO'21 artifact evaluation	2 months ago
config	Initial commit for MICRO'21 artifact evaluation	2 months ago
docs	Github pages documentation	7 hours ago
experiments	Added chart visualization in Excel template	2 months ago
inc	Updated README	8 days ago
prefetcher	Initial commit for MICRO'21 artifact evaluation	2 months ago
replacement	Initial commit for MICRO'21 artifact evaluation	2 months ago
scripts	Added md5 checksum for all artifact traces to verify download	2 months ago
src	Initial commit for MICRO'21 artifact evaluation	2 months ago
tracer	Initial commit for MICRO'21 artifact evaluation	2 months ago
.gitignore	Initial commit for MICRO'21 artifact evaluation	2 months ago
CITATION.cff	Added citation file	8 days ago
LICENSE	Updated LICENSE	2 months ago
LICENSE.champsim	Initial commit for MICRO'21 artifact evaluation	2 months ago

About

A customizable hardware prefetching framework using online reinforcement learning as described in the MICRO 2021 paper by Bera and Kanellopoulos et al.

arxiv.org/pdf/2109.12021.pdf

machine-learning reinforcement-learning computer-architecture prefetcher microarchitecture cache-replacement branch-predictor champsim-simulator champsim-tracer

Readme View license Cite this repository

Releases 5



v1.3 Latest 21 days ago

Pythia Talk Video

Steering Pythia's Objective via Reward Values

- Customizing reward values to make Pythia conservative towards p

Strict Pythia configuration



1
2

Server-class processors

Bandwidth-sensitive workloads

SAFARI

11:23 / 20:04 • Steering Pythia's Objective via Reward Values >

23

CC Settings Full Screen

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

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 Bera¹

Konstantinos Kanellopoulos¹

Anant V. Nori²

Taha Shahroodi^{3,1}

Sreenivas Subramoney²

Onur Mutlu¹

¹ETH Zürich

²Processor Architecture Research Labs, Intel Labs

³TU Delft

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



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>



Hermes: Perceptron-Based Off-Chip Load Prediction

Learning-Based Off-Chip Load Predictors

- Rahul Bera, Konstantinos Kanellopoulos, Shankar Balachandran, David Novo, Ataberk Olgun, Mohammad Sadrosadati, and Onur Mutlu,
"Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction"

Proceedings of the 55th International Symposium on Microarchitecture (MICRO), Chicago, IL, USA, October 2022.

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

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

[[Talk Video](#) (12 minutes)]

[[Lecture Video](#) (25 minutes)]

[[arXiv version](#)]

[[Source Code \(Officially Artifact Evaluated with All Badges\)](#)]

Officially artifact evaluated as available, reusable and reproducible.

Best paper award at MICRO 2022.



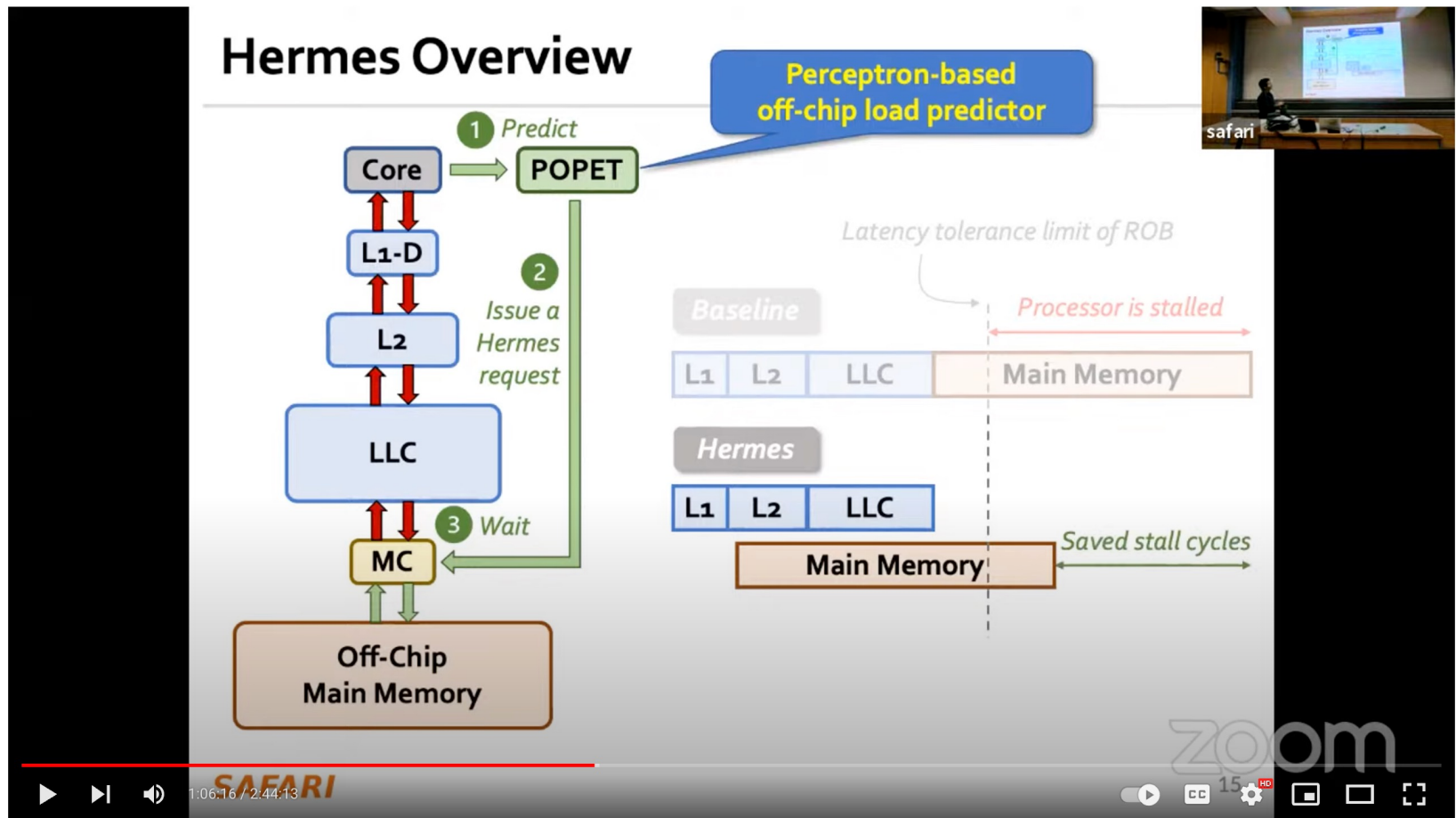
Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction

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

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

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

Hermes Talk Video



Computer Architecture - Lecture 18: Cutting-Edge Research in Computer Architecture (Fall 2022)



Onur Mutlu Lectures
32.9K subscribers

Analytics

Edit video

23



Share

Download

Clip

Save



2.4K views Streamed 5 months ago Livestream - Computer Architecture - ETH Zürich (Fall 2022)
Computer Architecture, ETH Zürich, Fall 2022 (<https://safari.ethz.ch/architecture/f...>)

SAFARI

<https://www.youtube.com/watch?v=PWWBtrL60dQ&t=3609s>



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

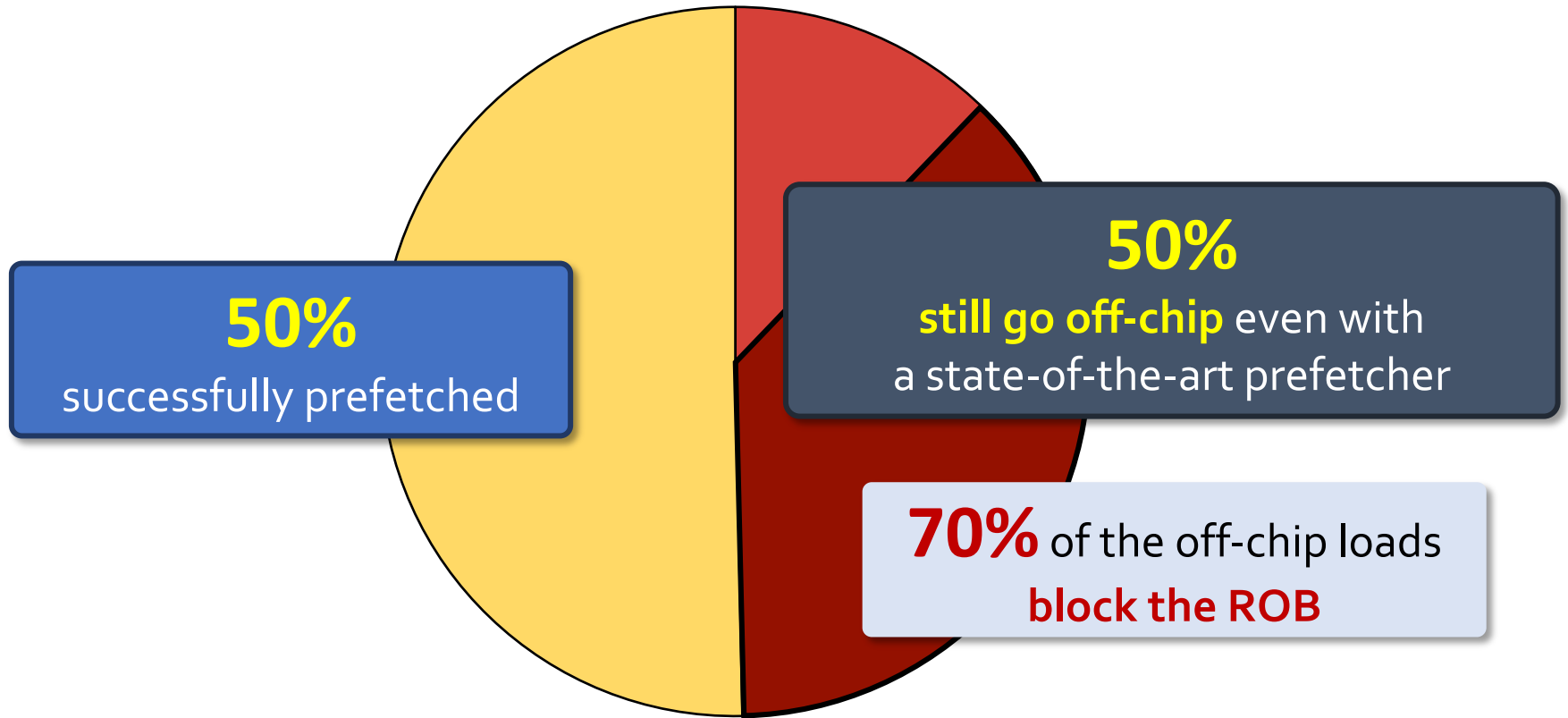
Employ sophisticated **prefetchers**

2

Increase size of on-chip caches

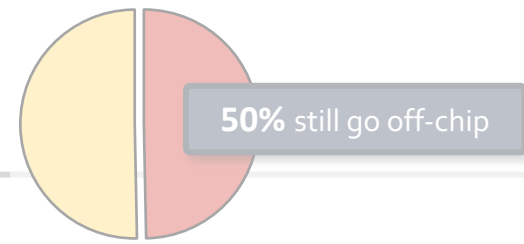
Key Observation 1

Many loads still go off-chip

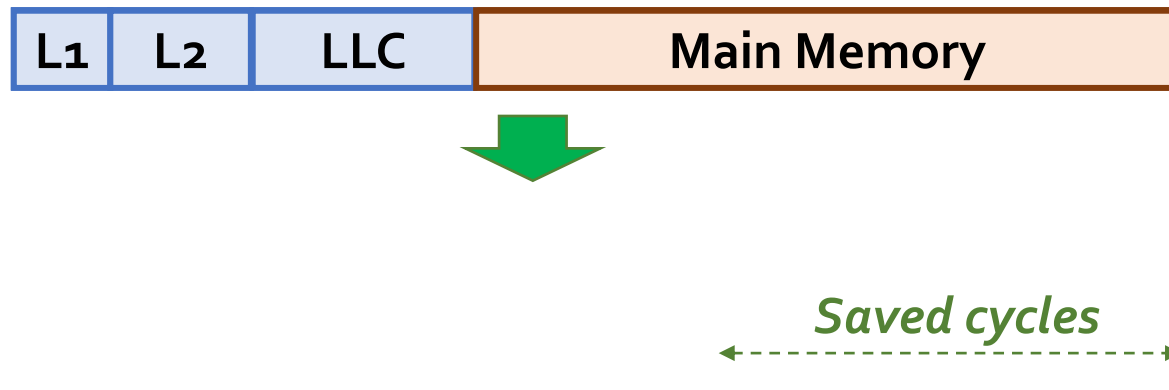


off-chip loads without any prefetcher

Key Observation 2

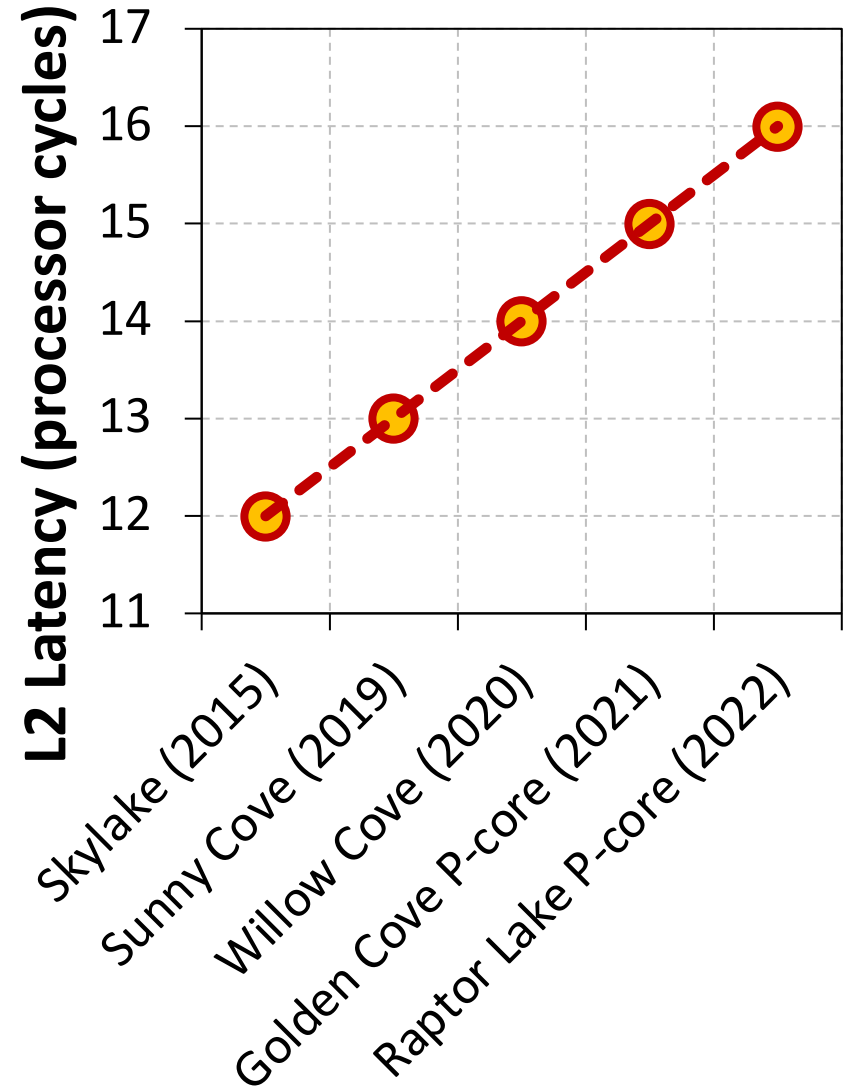
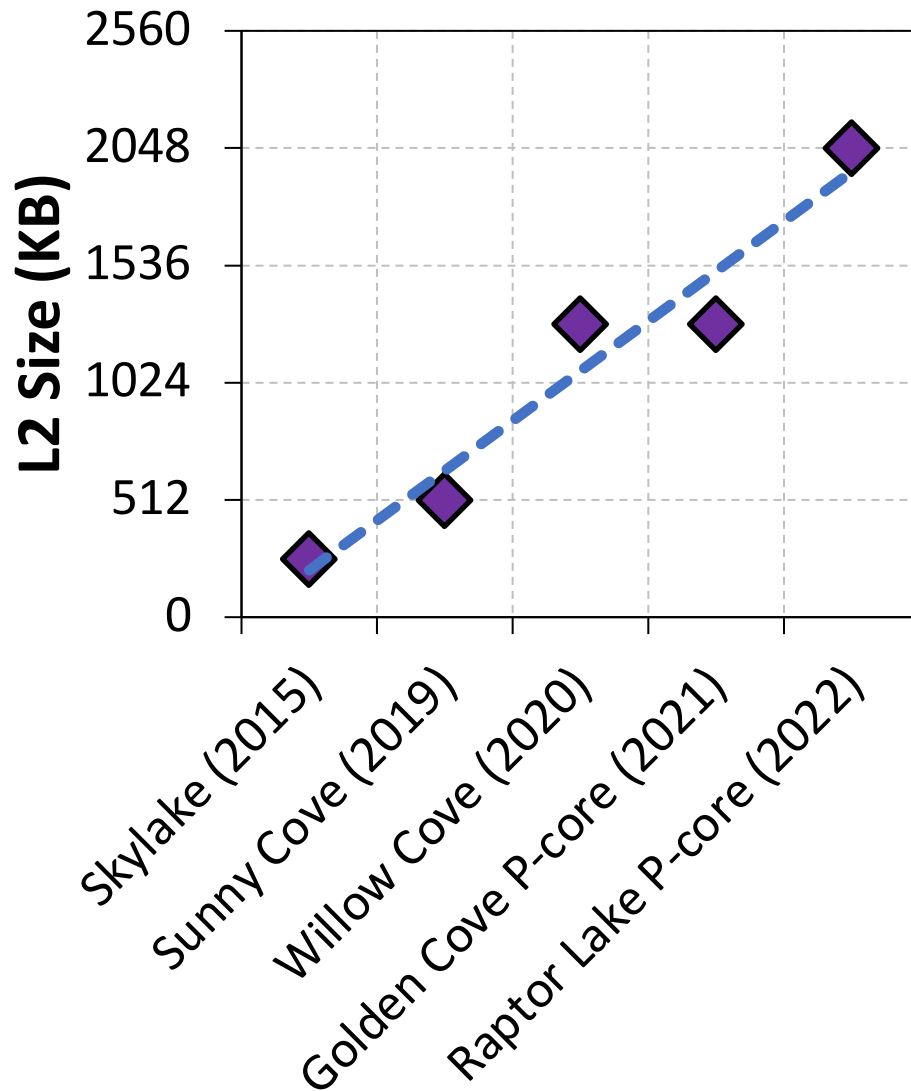


On-chip cache access latency
significantly contributes to off-chip load latency



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

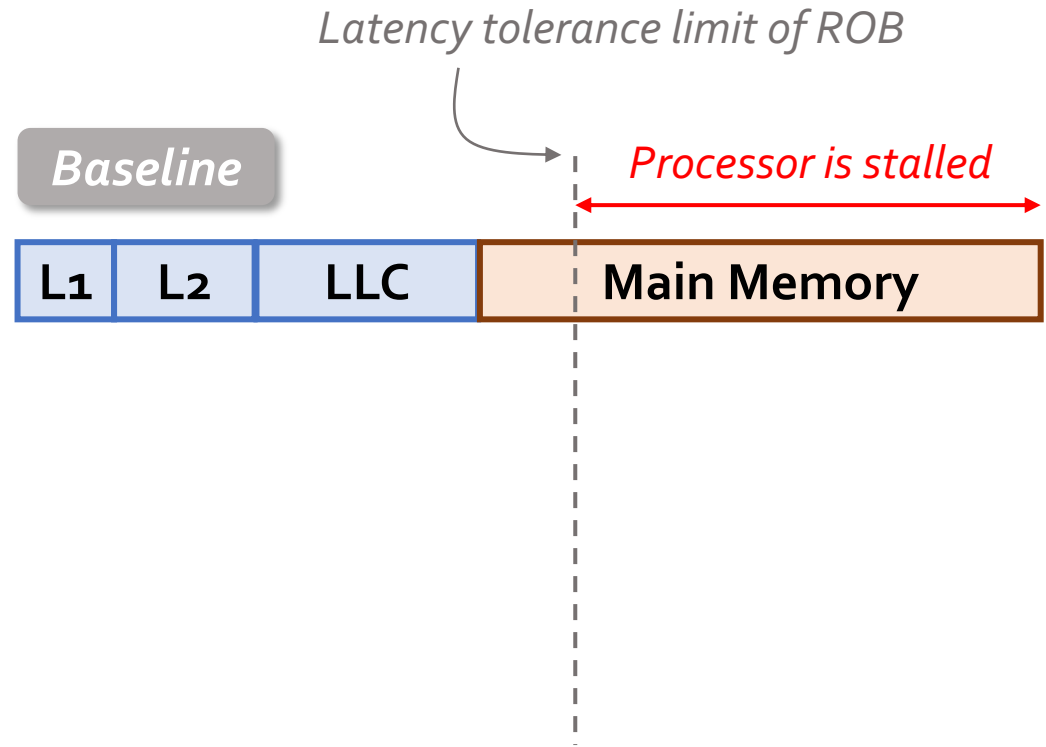
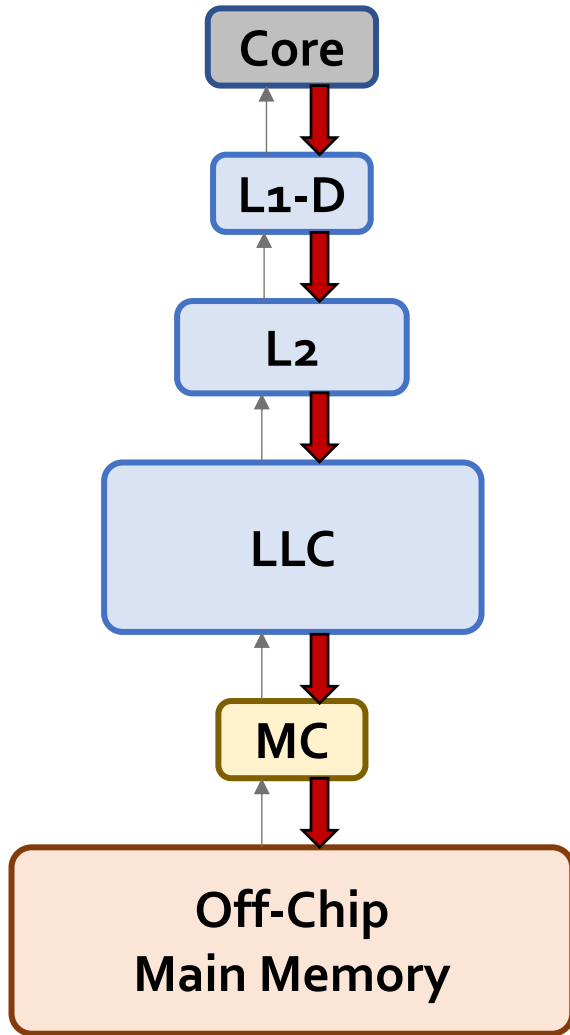


That predicts which loads are likely to **go off-chip**

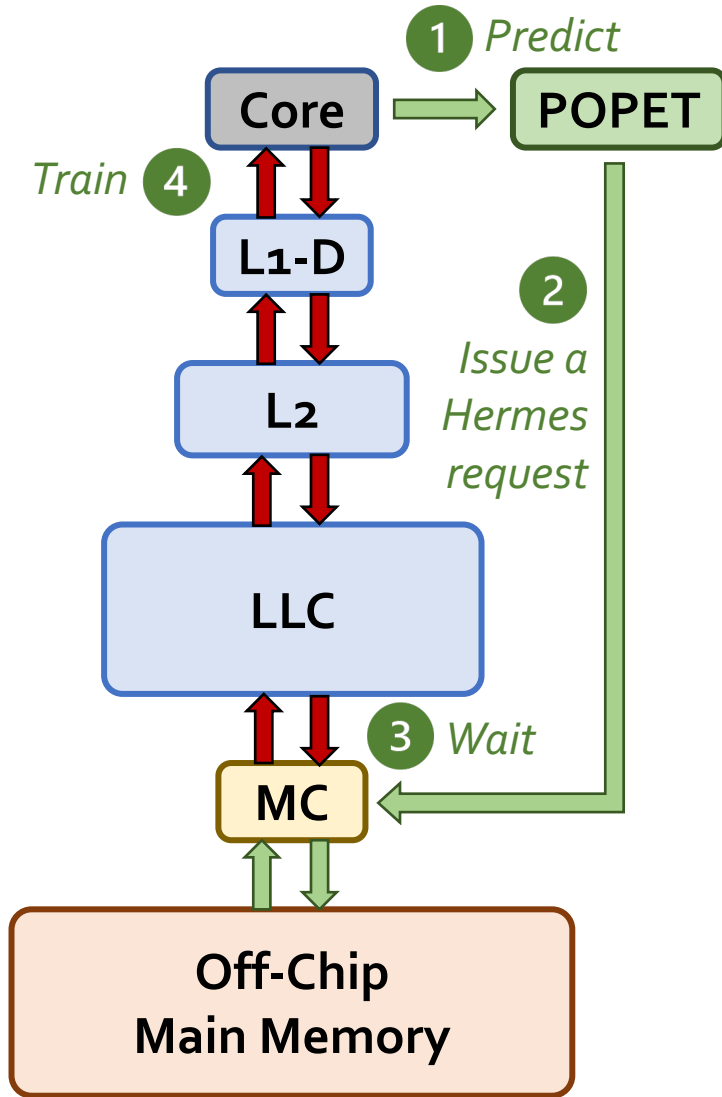


By **learning** from
multiple program context information

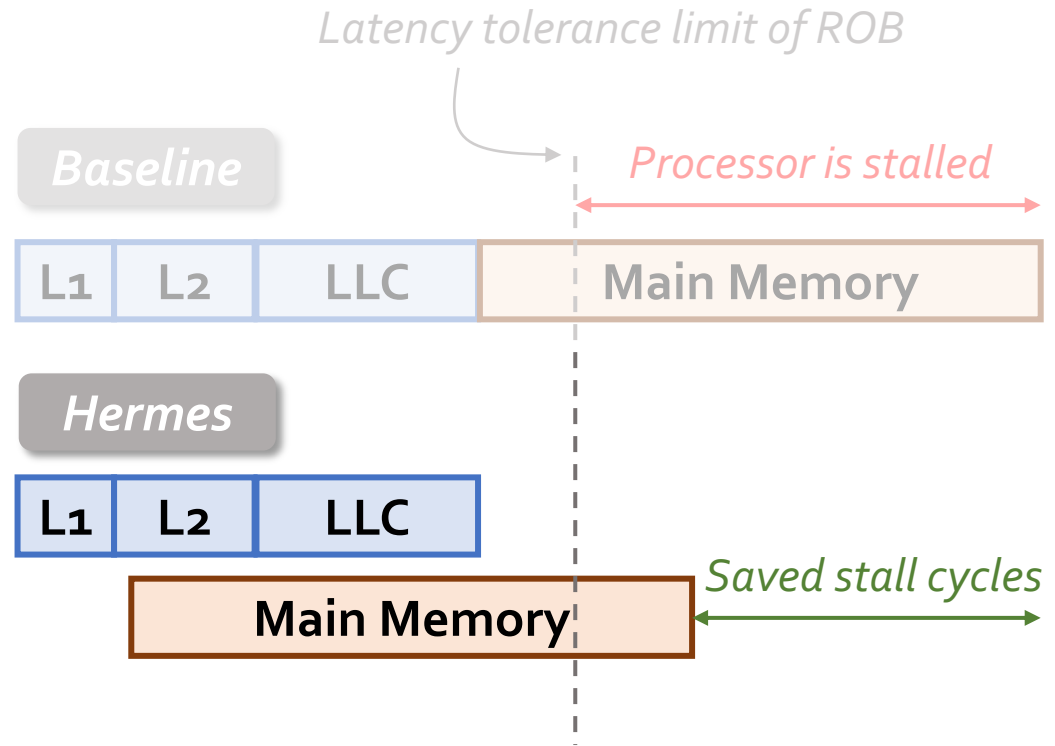
Hermes Overview



Hermes Overview



Perceptron-based
off-chip load predictor



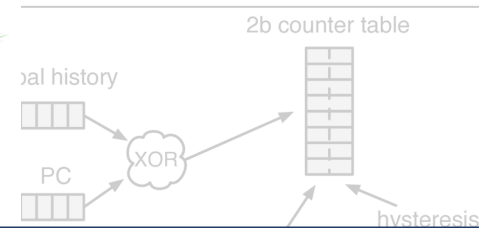
Designing the Off-Chip Load Predictor

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

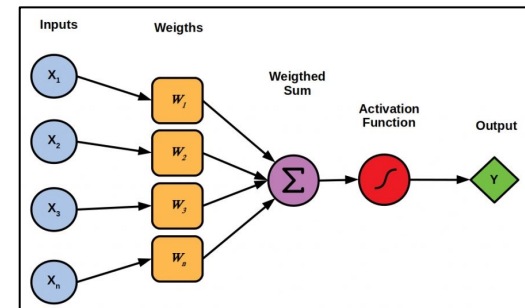
Correlate different program features with off-chip loads



Low storage overhead

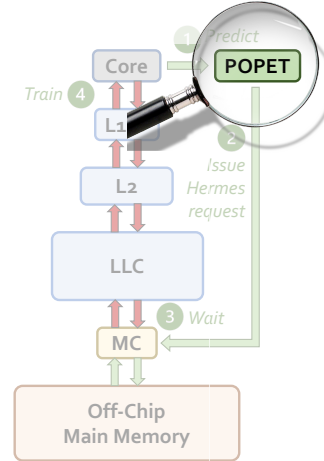
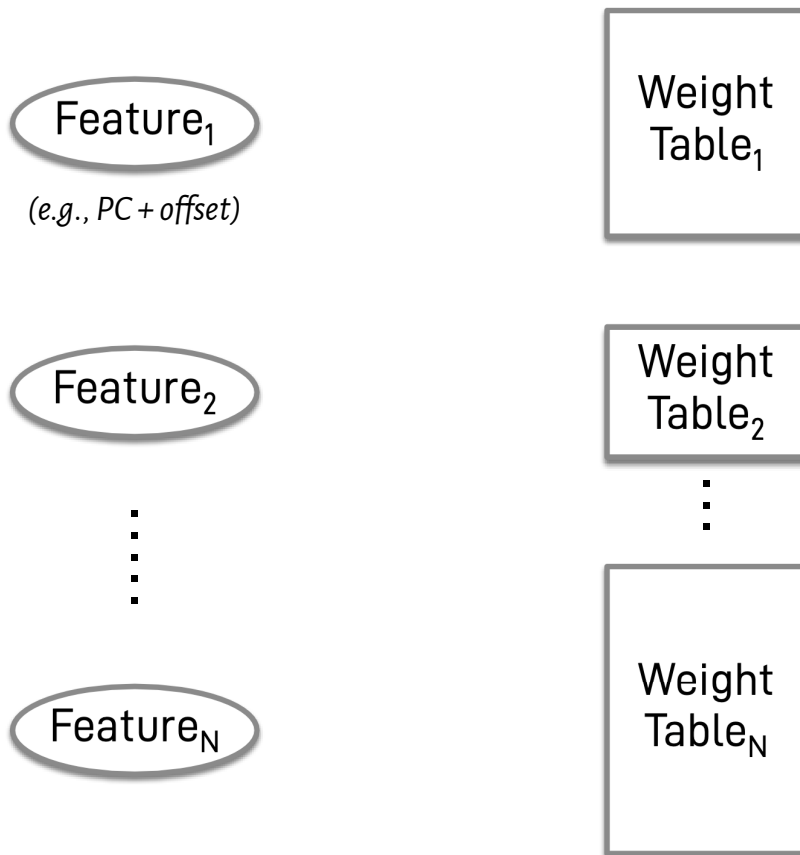


Low design complexity



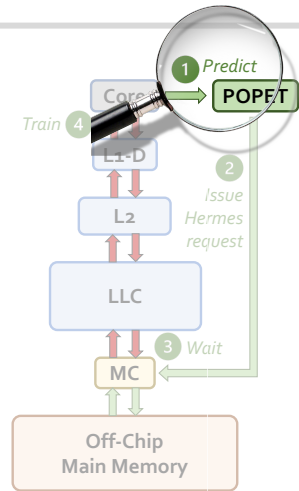
POPET: Perceptron-Based Off-Chip Predictor

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

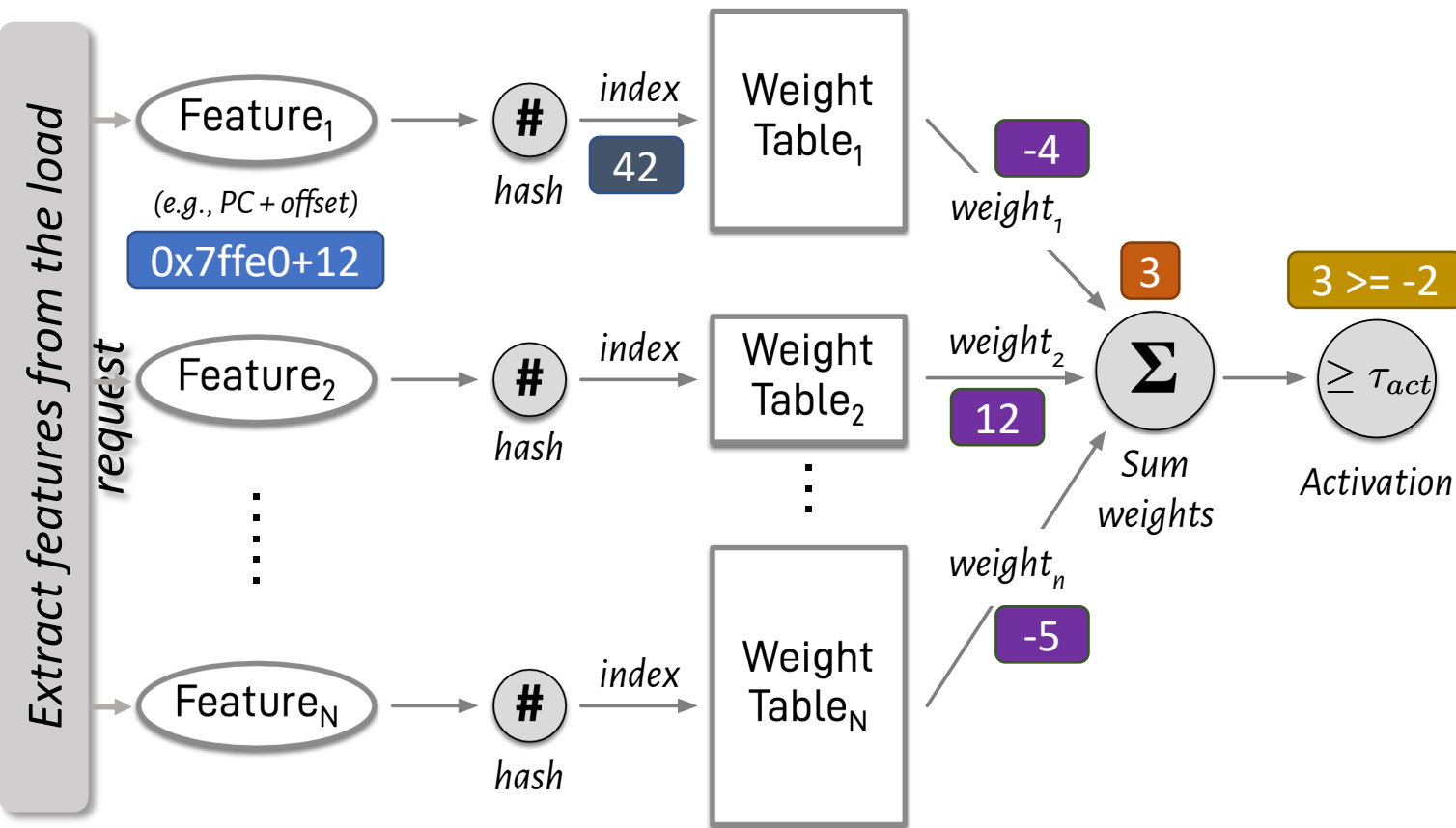


Predicting using POPET

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

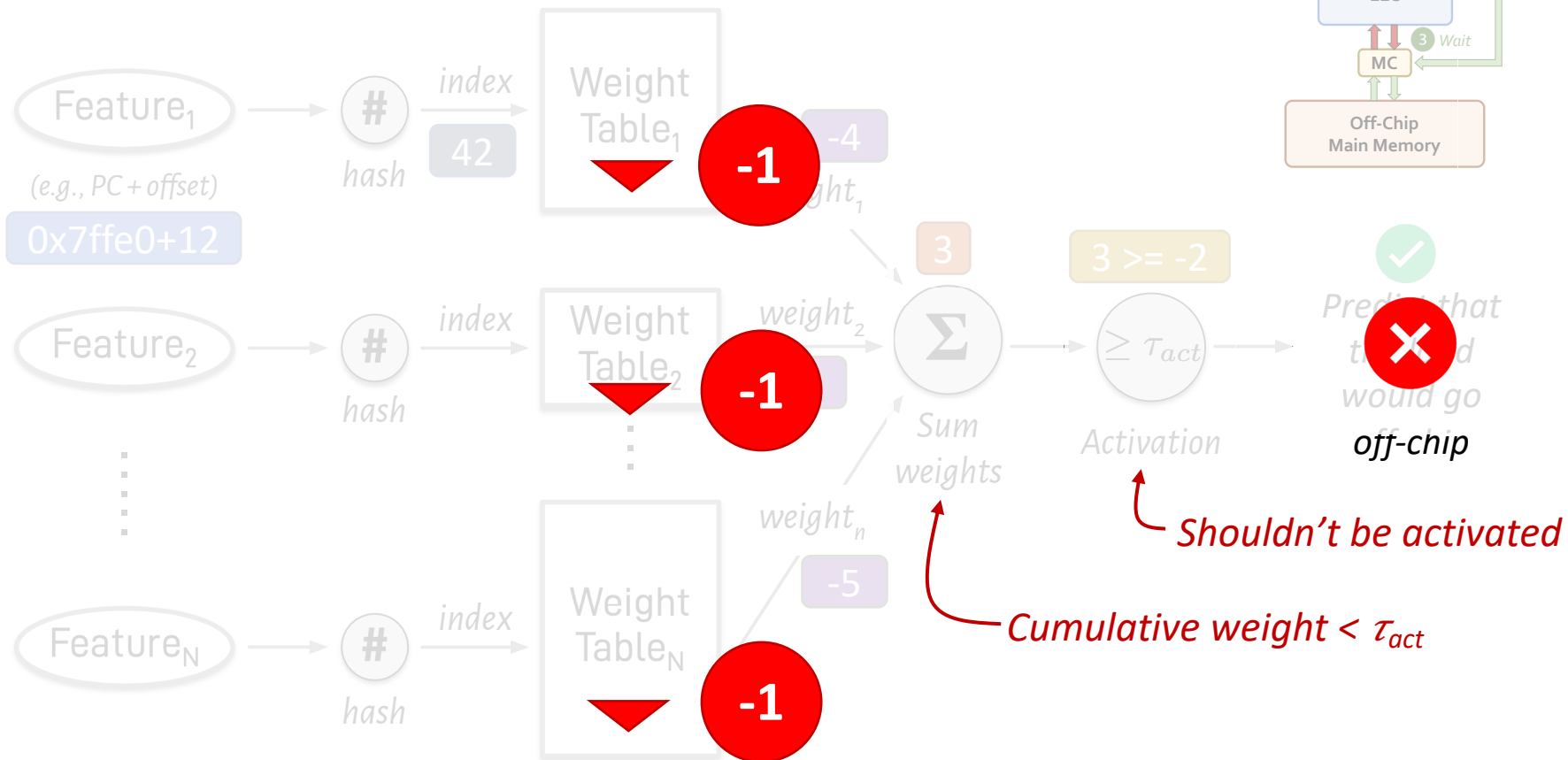
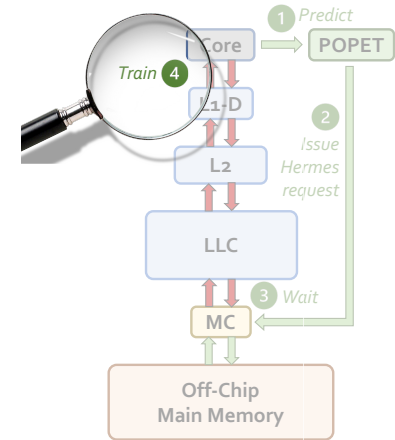


✓
Predict that
the load
would go
off-chip



Training POPET

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



Features Used in Hermes

Table 1: The initial set of program features used for automated feature selection. \oplus represents a bitwise XOR operation.

Features without control-flow information	Features with control-flow information
1. Load virtual address	8. Load PC
2. Virtual page number	9. $PC \oplus$ load virtual address
3. Cacheline offset in page	10. $PC \oplus$ virtual page number
4. First access	11. $PC \oplus$ cacheline offset
5. Cacheline offset + first access	12. $PC +$ first access
6. Byte offset in cacheline	13. $PC \oplus$ byte offset
7. Word offset in cacheline	14. $PC \oplus$ word offset
	15. Last-4 load PCs
	16. Last-4 PCs

Table 2: POPET configuration parameters

<i>Selected features</i>	<ul style="list-style-type: none"> • $PC \oplus$ cacheline offset • $PC \oplus$ byte offset • $PC +$ first access • Cacheline offset + first access • Last-4 load PCs
<i>Threshold values</i>	$\tau_{act} = -18, T_N = -35, T_P = 40$

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

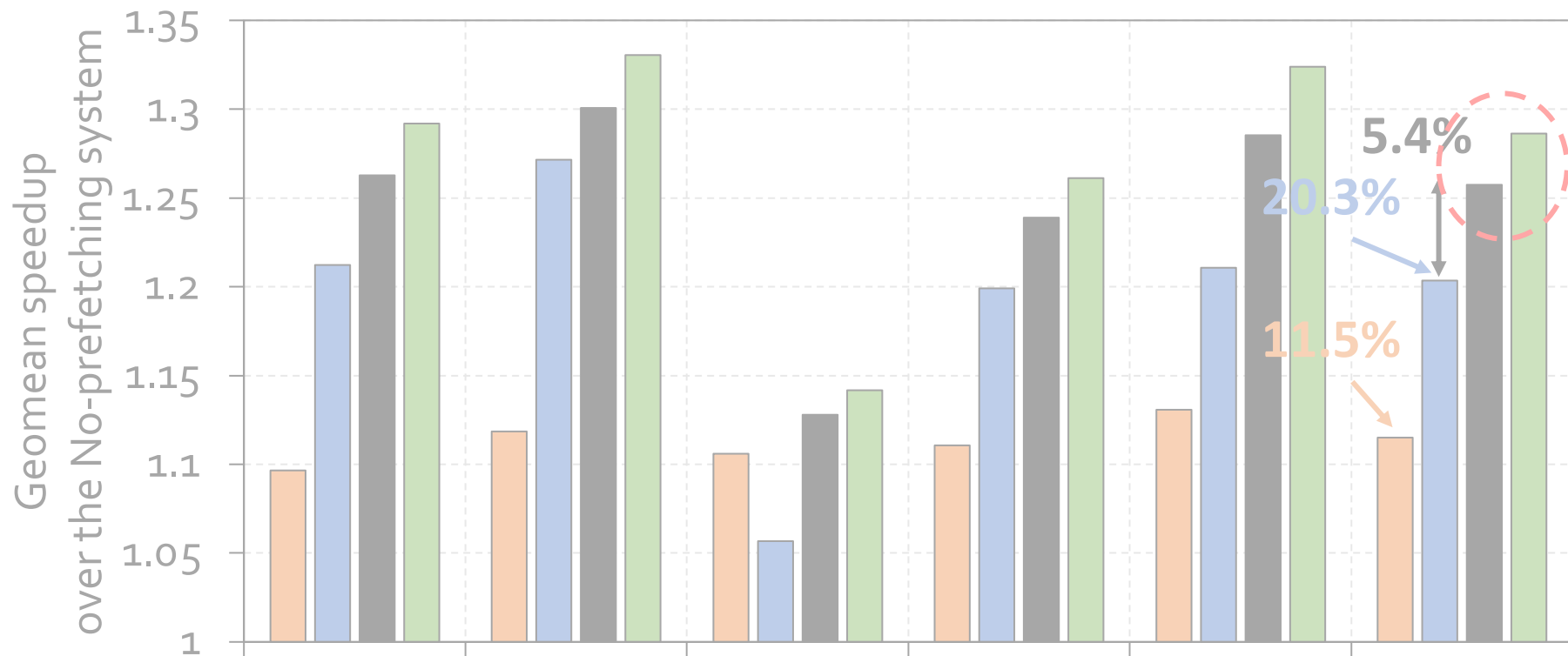
LLC Prefetchers

- 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**

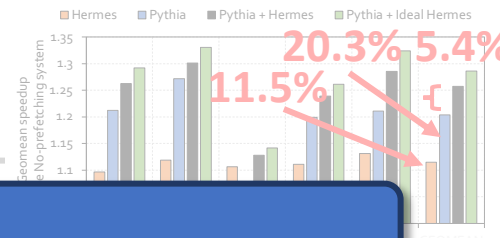
Single-Core Performance Improvement



Hermes alone provides nearly

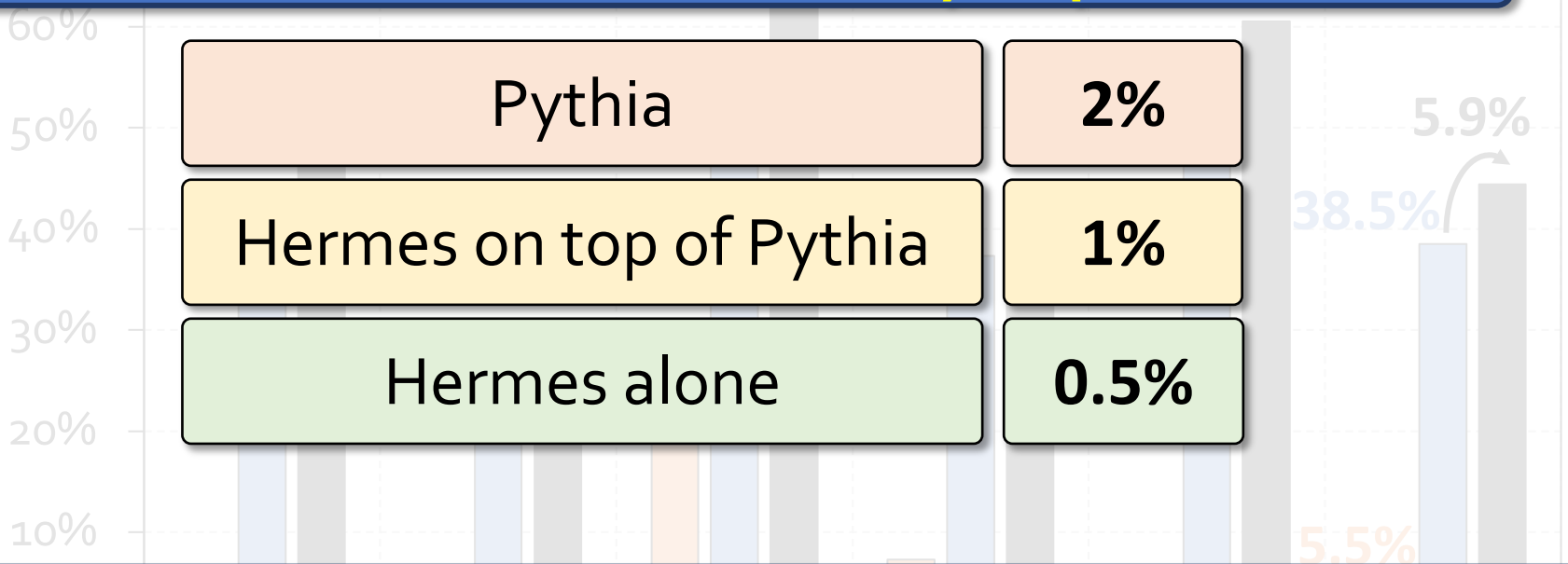
Hermes provides nearly **90%** performance benefit of **Ideal Hermes** that has an **ideal off-chip load predictor** with only **2.5%** storage overhead

Increase in Main Memory Requests



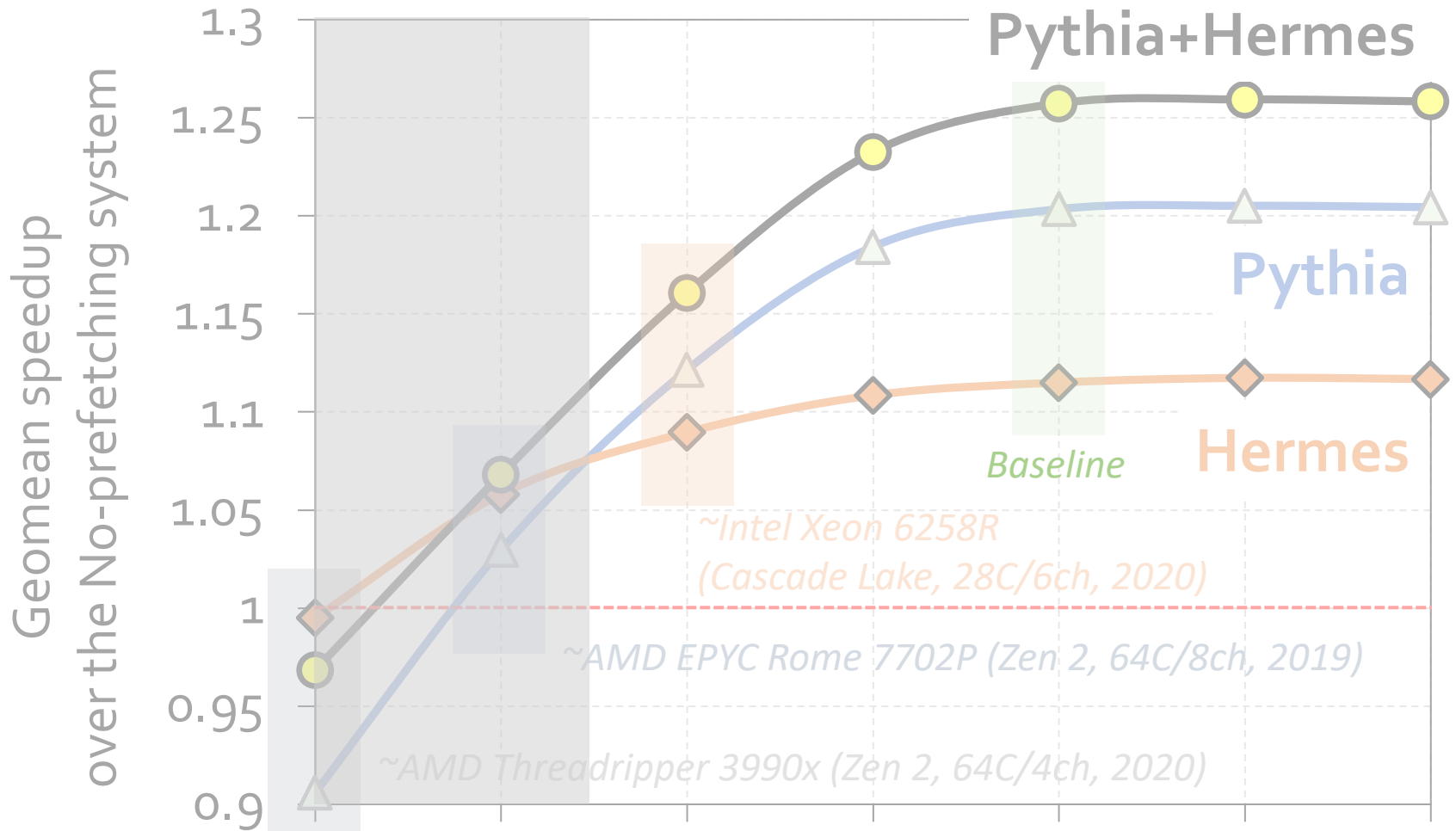
For **every 1% performance** benefit,
increase in main memory requests

Increase in main memory requests over the No-prefetching system



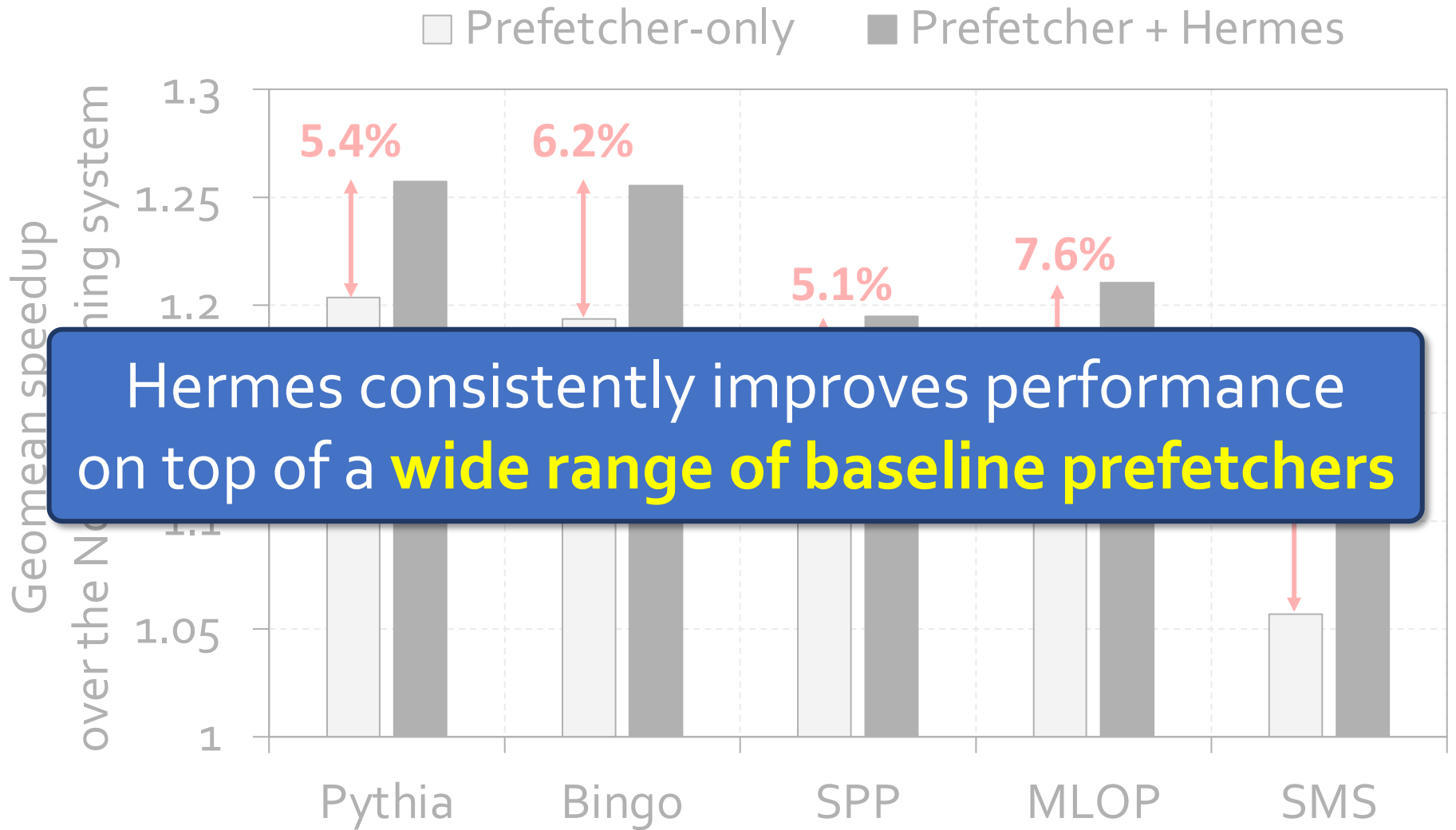
Hermes is more **bandwidth-efficient**
than even an efficient prefetcher like Pythia

Performance with Varying Memory Bandwidth



Hermes+Pythia outperforms Pythia
across all bandwidth configurations

Performance with Varying Baseline Prefetcher



Overhead of Hermes



4 KB storage overhead



1.5% power overhead*

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

A Lot More in the Hermes Paper



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

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

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

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 re-order 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

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

- Rahul Bera, Konstantinos Kanellopoulos, Shankar Balachandran, David Novo, Ataberk Olgun, Mohammad Sadrosadati, and Onur Mutlu,
"Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction"
Proceedings of the 55th International Symposium on Microarchitecture (MICRO), Chicago, IL, USA, October 2022.
[[Slides \(pptx\)](#)] ([pdf](#))
[[Longer Lecture Slides \(pptx\)](#)] ([pdf](#))
[[Talk Video](#) (12 minutes)]
[[Lecture Video](#) (25 minutes)]
[[arXiv version](#)]
[[Source Code \(Officially Artifact Evaluated with All Badges\)](#)]
Officially artifact evaluated as available, reusable and reproducible.
Best paper award at MICRO 2022.



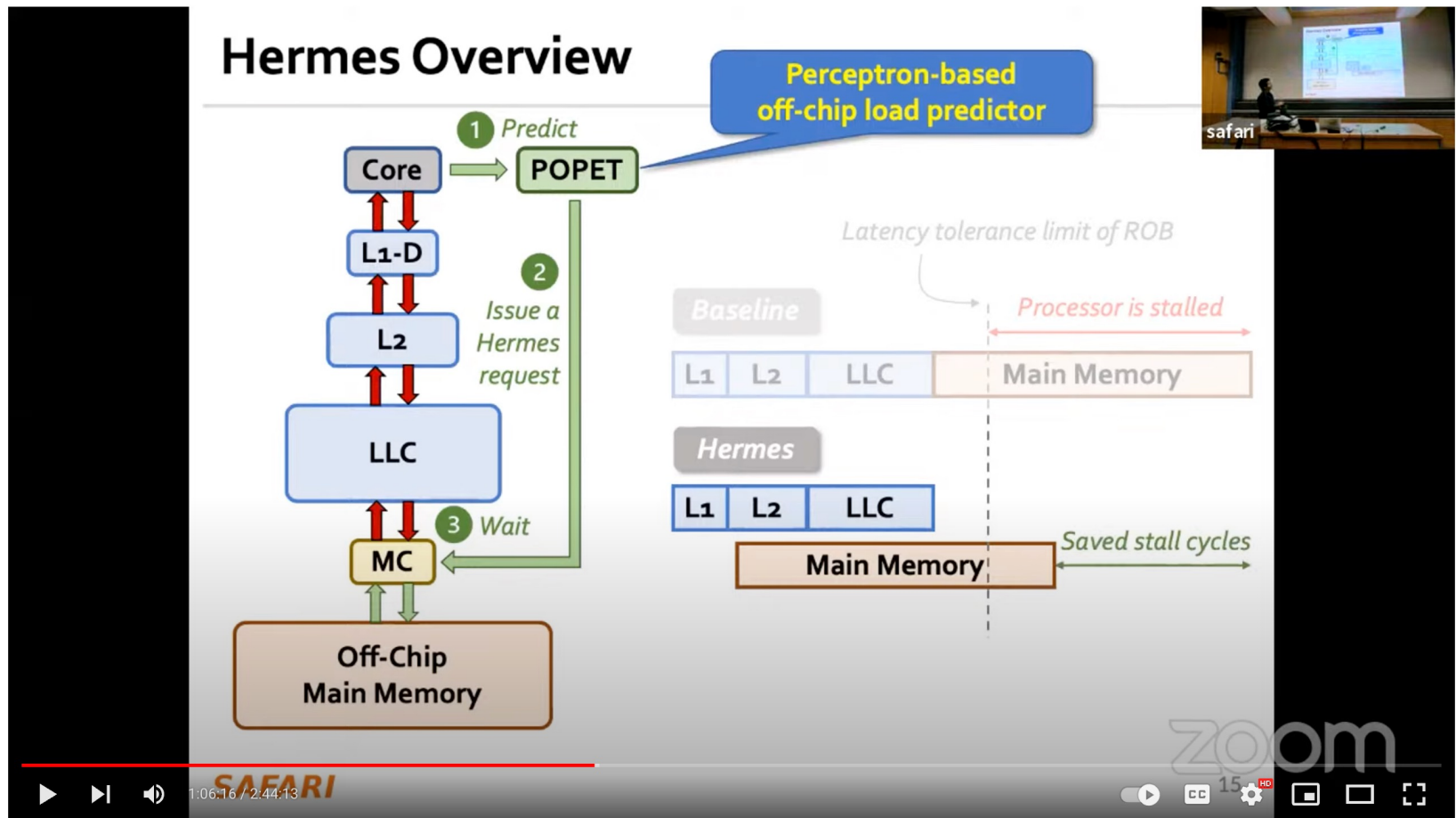
Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction

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

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

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

Hermes Talk Video



Computer Architecture - Lecture 18: Cutting-Edge Research in Computer Architecture (Fall 2022)



Onur Mutlu Lectures
32.9K subscribers

Analytics

Edit video

23



Share

Download

Clip

Save



2.4K views Streamed 5 months ago Livestream - Computer Architecture - ETH Zürich (Fall 2022)
Computer Architecture, ETH Zürich, Fall 2022 (<https://safari.ethz.ch/architecture/f...>)

SAFARI

<https://www.youtube.com/watch?v=PWWBtrL60dQ&t=3609s>



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>



Reinforcement Learning Based **DRAM Controllers**

DRAM Controller: Functions

- Ensure correct operation of DRAM (refresh and timing)
- Service DRAM requests while obeying timing constraints of DRAM chips
 - Constraints: resource conflicts (bank, bus, channel), minimum write-to-read delays
 - Translate requests to DRAM command sequences
- Buffer and schedule requests for high performance + QoS
 - Reordering, row-buffer, bank, rank, bus management
- Manage power consumption and thermals in DRAM
 - Turn on/off DRAM chips, manage power modes

Why Are DRAM Controllers Difficult to Design?

- Need to obey **DRAM timing constraints** for correctness
 - There are many (50+) timing constraints in DRAM
 - tWTR: Minimum number of cycles to wait before issuing a read command after a write command is issued
 - tRC: Minimum number of cycles between the issuing of two consecutive activate commands to the same bank
 - ...
- Need to **keep track of many resources** to prevent conflicts
 - Channels, banks, ranks, data bus, address bus, row buffers
- Need to handle **DRAM refresh**
- Need to **manage power** consumption
- Need to **optimize performance & QoS** (in the presence of constraints)
 - Reordering is not simple
 - Fairness and QoS needs complicates the scheduling problem

Many DRAM Timing Constraints

Latency	Symbol	DRAM cycles	Latency	Symbol	DRAM cycles
Precharge	t_{RP}	11	Activate to read/write	t_{RCD}	11
Read column address strobe	CL	11	Write column address strobe	CWL	8
Additive	AL	0	Activate to activate	t_{RC}	39
Activate to precharge	t_{RAS}	28	Read to precharge	t_{RTP}	6
Burst length	t_{BL}	4	Column address strobe to column address strobe	t_{CCD}	4
Activate to activate (different bank)	t_{RRD}	6	Four activate windows	t_{FAW}	24
Write to read	t_{WTR}	6	Write recovery	t_{WR}	12

Table 4. DDR3 1600 DRAM timing specifications

- From Lee et al., “[DRAM-Aware Last-Level Cache Writeback: Reducing Write-Caused Interference in Memory Systems](#),” HPS Technical Report, April 2010.

More on DRAM Operation

- Kim et al., "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," ISCA 2012.
- Lee et al., "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013.

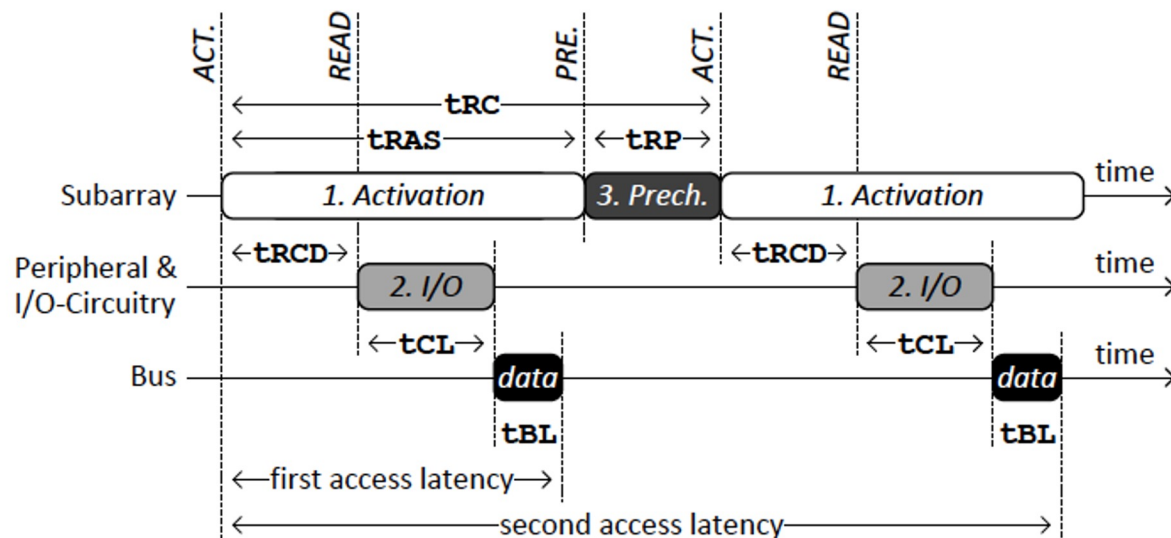


Figure 5. Three Phases of DRAM Access

Table 2. Timing Constraints (DDR3-1066) [43]

Phase	Commands	Name	Value
1	ACT → READ	t_{RCD}	15ns
	ACT → WRITE		
	ACT → PRE	t_{RAS}	37.5ns
2	READ → data	t_{CL}	15ns
	WRITE → data	t_{CWL}	11.25ns
	data burst	t_{BL}	7.5ns
3	PRE → ACT	t_{RP}	15ns
1 & 3	ACT → ACT	t_{RC} ($t_{RAS}+t_{RP}$)	52.5ns

DRAM Scheduling Policies (I)

- **FCFS** (first come first served)
 - Oldest request first
- **FR-FCFS** (first ready, first come first served)
 1. Row-hit first
 2. Oldest first

Goal: Maximize row buffer hit rate → maximize DRAM throughput

DRAM Scheduling Policies (II)

- A scheduling policy is a request prioritization order

- Prioritization can be based on
 - Request age
 - Row buffer hit/miss status
 - Request type (prefetch, read, write)
 - Requestor type (load miss or store miss)
 - Request **criticality**
 - Oldest miss in the core?
 - How many instructions in core are dependent on it?
 - Will it stall the processor?
 - Interference caused to other cores
 - ...

Memory Performance Attacks [USENIX SEC'07]

- Thomas Moscibroda and Onur Mutlu,
"Memory Performance Attacks: Denial of Memory Service in Multi-Core Systems"
*Proceedings of the 16th USENIX Security Symposium (**USENIX SECURITY**), pages 257-274, Boston, MA, August 2007. Slides (ppt)*

Memory Performance Attacks: Denial of Memory Service in Multi-Core Systems

*Thomas Moscibroda Onur Mutlu
Microsoft Research
{moscitho,onur}@microsoft.com*

STFM [MICRO'07]

- Onur Mutlu and Thomas Moscibroda,
"Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors"
Proceedings of the 40th International Symposium on Microarchitecture (MICRO), pages 146-158, Chicago, IL, December 2007. [[Summary](#)] [[Slides \(ppt\)](#)]

Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors

Onur Mutlu Thomas Moscibroda

Microsoft Research
{onur,moscitho}@microsoft.com

- Onur Mutlu and Thomas Moscibroda,
"Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems"
Proceedings of the 35th International Symposium on Computer Architecture (ISCA), pages 63-74, Beijing, China, June 2008.
[Summary] [Slides (ppt)]

Parallelism-Aware Batch Scheduling:

Enhancing both Performance and Fairness of Shared DRAM Systems

Onur Mutlu Thomas Moscibroda
Microsoft Research
{onur,moscitho}@microsoft.com

On PAR-BS

- Variants implemented in Samsung SoC memory controllers

Effective platform level approach and DRAM accesses are crucial to system performance. This paper touches this topics and suggest a superior approach to current known techniques.

Review from ISCA 2008

ATLAS Memory Scheduler [HPCA'10]

- Yoongu Kim, Dongsu Han, Onur Mutlu, and Mor Harchol-Balter, **"ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers"**
Proceedings of the 16th International Symposium on High-Performance Computer Architecture (HPCA), Bangalore, India, January 2010. [Slides \(pptx\)](#)

ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers

Yoongu Kim Dongsu Han Onur Mutlu Mor Harchol-Balter
Carnegie Mellon University

Thread Cluster Memory Scheduling [MICRO'10]

- Yoongu Kim, Michael Papamichael, Onur Mutlu, and Mor Harchol-Balter,

"Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior"

*Proceedings of the 43rd International Symposium on Microarchitecture (**MICRO**), pages 65-76, Atlanta, GA, December 2010. [Slides \(pptx\)](#) ([pdf](#))*

Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior

Yoongu Kim

yoonguk@ece.cmu.edu

Michael Papamichael

papamix@cs.cmu.edu

Onur Mutlu

onur@cmu.edu

Mor Harchol-Balter

harchol@cs.cmu.edu

Carnegie Mellon University

BLISS [ICCD'14, TPDS'16]

- Lavanya Subramanian, Donghyuk Lee, Vivek Seshadri, Harsha Rastogi, and Onur Mutlu,
"The Blacklisting Memory Scheduler: Achieving High Performance and Fairness at Low Cost"
Proceedings of the 32nd IEEE International Conference on Computer Design (ICCD), Seoul, South Korea, October 2014.
[[Slides \(pptx\)](#) ([pdf](#))]

The Blacklisting Memory Scheduler: Achieving High Performance and Fairness at Low Cost

Lavanya Subramanian, Donghyuk Lee, Vivek Seshadri, Harsha Rastogi, Onur Mutlu
Carnegie Mellon University
{lsubrama,donghyu1,visesh,harshar,onur}@cmu.edu

Staged Memory Scheduling: CPU-GPU [ISCA'12]

- Rachata Ausavarungnirun, Kevin Chang, Lavanya Subramanian, Gabriel Loh, and Onur Mutlu,
"Staged Memory Scheduling: Achieving High Performance and Scalability in Heterogeneous Systems"
Proceedings of the 39th International Symposium on Computer Architecture (ISCA), Portland, OR, June 2012. [Slides \(pptx\)](#)

Staged Memory Scheduling: Achieving High Performance and Scalability in Heterogeneous Systems

Rachata Ausavarungnirun[†] Kevin Kai-Wei Chang[†] Lavanya Subramanian[†] Gabriel H. Loh[‡] Onur Mutlu[†]

[†]Carnegie Mellon University
{rachata,kevincha,lsubrama,onur}@cmu.edu

[‡]Advanced Micro Devices, Inc.
gabe.loh@amd.com

DASH: Heterogeneous Systems [TACO'16]

- Hiroyuki Usui, Lavanya Subramanian, Kevin Kai-Wei Chang, and Onur Mutlu,
"DASH: Deadline-Aware High-Performance Memory Scheduler for Heterogeneous Systems with Hardware Accelerators"
ACM Transactions on Architecture and Code Optimization (TACO), Vol. 12, January 2016.
Presented at the 11th HiPEAC Conference, Prague, Czech Republic, January 2016.
[Slides (pptx)] [pdf]
[Source Code]

DASH: Deadline-Aware High-Performance Memory Scheduler for Heterogeneous Systems with Hardware Accelerators

HIROYUKI USUI, LAVANYA SUBRAMANIAN, KEVIN KAI-WEI CHANG,
and ONUR MUTLU, Carnegie Mellon University

MISE: Predictable Performance [HPCA'13]

- Lavanya Subramanian, Vivek Seshadri, Yoongu Kim, Ben Jaiyen, and Onur Mutlu,

"MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems"

*Proceedings of the 19th International Symposium on High-Performance Computer Architecture (**HPCA**), Shenzhen, China, February 2013. Slides (pptx)*

MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems

Lavanya Subramanian

Vivek Seshadri

Yoongu Kim

Ben Jaiyen

Onur Mutlu

Carnegie Mellon University

ASM: Predictable Performance [MICRO'15]

- Lavanya Subramanian, Vivek Seshadri, Arnab Ghosh, Samira Khan, and Onur Mutlu,
"The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory"
Proceedings of the 48th International Symposium on Microarchitecture (MICRO), Waikiki, Hawaii, USA, December 2015.
[[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Session Slides \(pptx\)](#)] [[pdf](#)] [[Poster \(pptx\)](#)] [[pdf](#)]
[[Source Code](#)]

The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory

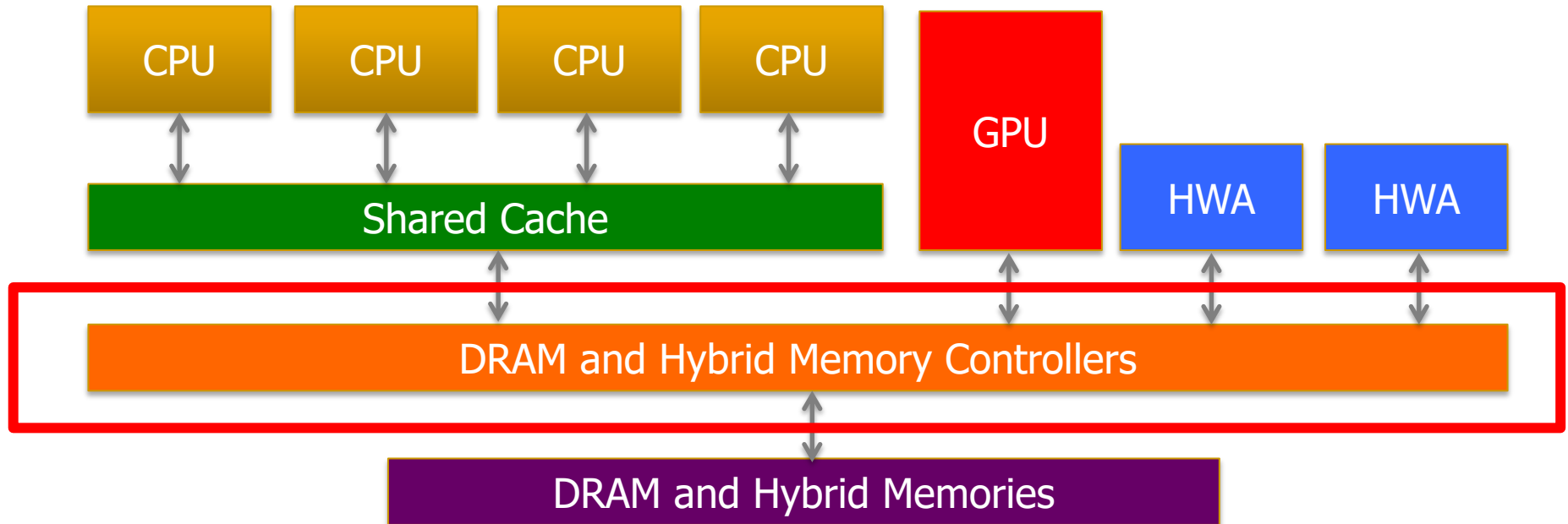
Lavanya Subramanian*§ Vivek Seshadri* Arnab Ghosh*†
Samira Khan*‡ Onur Mutlu*

*Carnegie Mellon University §Intel Labs †IIT Kanpur ‡University of Virginia

Memory Controllers
are critical to research

They will become
even more important

Memory Control is Getting More Complex



- Heterogeneous agents: CPUs, GPUs, and HWAs
- Main memory interference between CPUs, GPUs, HWAs

Many goals, many constraints, many metrics ...

Reality and Dream

- Reality: It is difficult to design a policy that maximizes performance, QoS, energy-efficiency, ...
 - Too many things to think about
 - Continuously changing workload and system behavior



- Dream: Wouldn't it be nice if the DRAM controller automatically found a good scheduling policy on its own?

Self-Optimizing DRAM Controllers

- Problem: DRAM controllers are difficult to design
 - It is difficult for human designers to design a policy that can adapt itself very well to different workloads and different system conditions
- Idea: A memory controller that adapts its scheduling policy to workload behavior and system conditions using machine learning.
- Observation: Reinforcement learning maps nicely to memory control.
- Design: Memory controller is a reinforcement learning agent
 - It dynamically and continuously learns and employs the best scheduling policy to maximize long-term performance.

Self-Optimizing DRAM Controllers

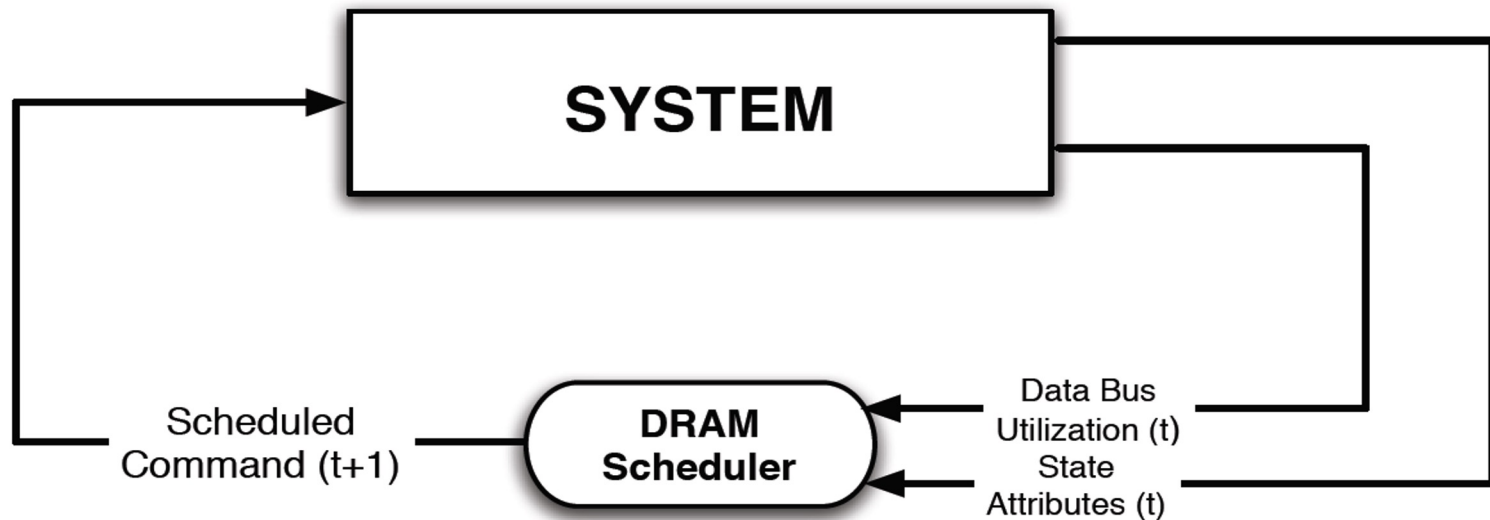


Goal: Learn to choose actions to maximize $r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$ ($0 \leq \gamma < 1$)

Figure 2: (a) Intelligent agent based on reinforcement learning principles;

Self-Optimizing DRAM Controllers

- Dynamically adapt the memory scheduling policy via interaction with the system at runtime
 - Associate system states and actions (commands) with long term reward values: **each action at a given state leads to a learned reward**
 - **Schedule command with highest estimated long-term reward value** in each state
 - **Continuously update reward values for** $\langle \text{state}, \text{action} \rangle$ pairs based on feedback from system



Self-Optimizing DRAM 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.

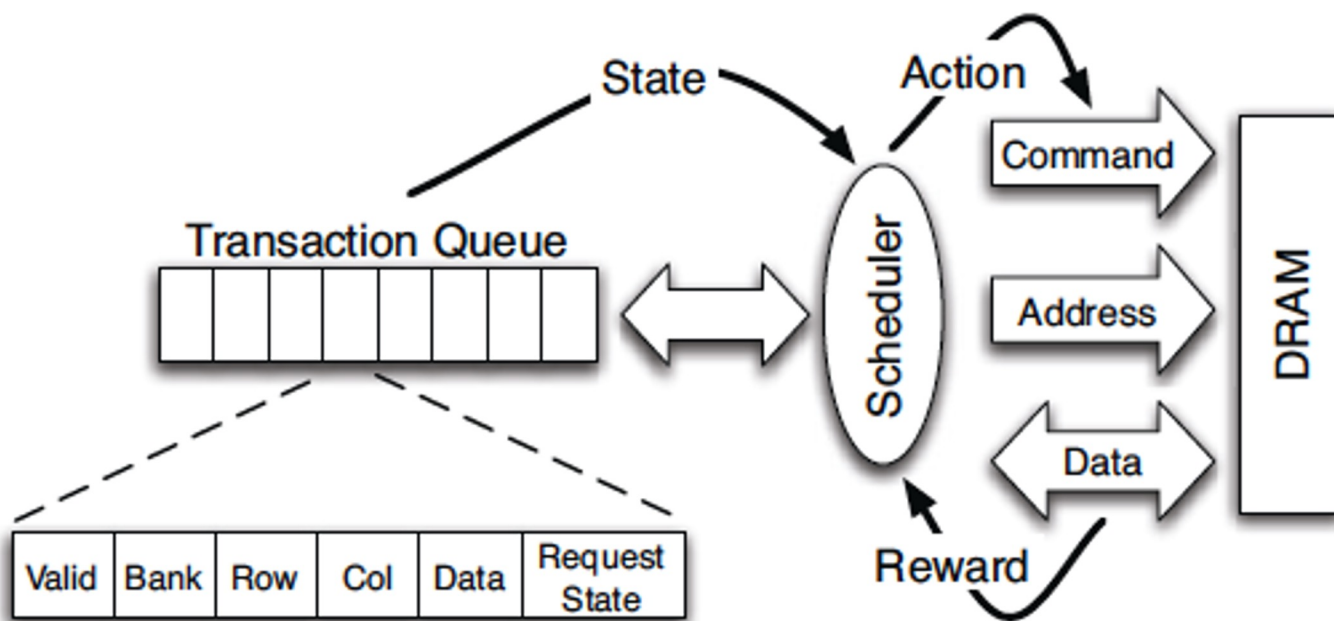


Figure 4: High-level overview of an RL-based scheduler.

States, Actions, Rewards

- Reward function

- +1 for scheduling Read and Write commands
- 0 at all other times

Goal is to maximize long-term data bus utilization

- State attributes

- Number of reads, writes, and load misses in transaction queue
- Number of pending writes and ROB heads waiting for referenced row
- Request's relative ROB order

- Actions

- Activate
- Write
- Read - load miss
- Read - store miss
- Precharge - pending
- Precharge - preemptive
- NOP

Performance Results

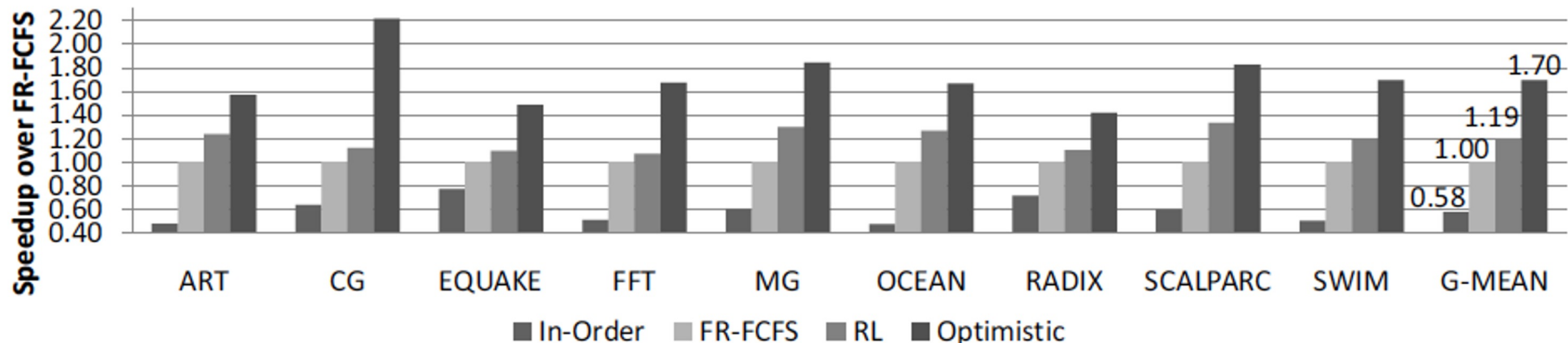


Figure 7: Performance comparison of in-order, FR-FCFS, RL-based, and optimistic memory controllers

Large, robust performance improvements over many human-designed policies

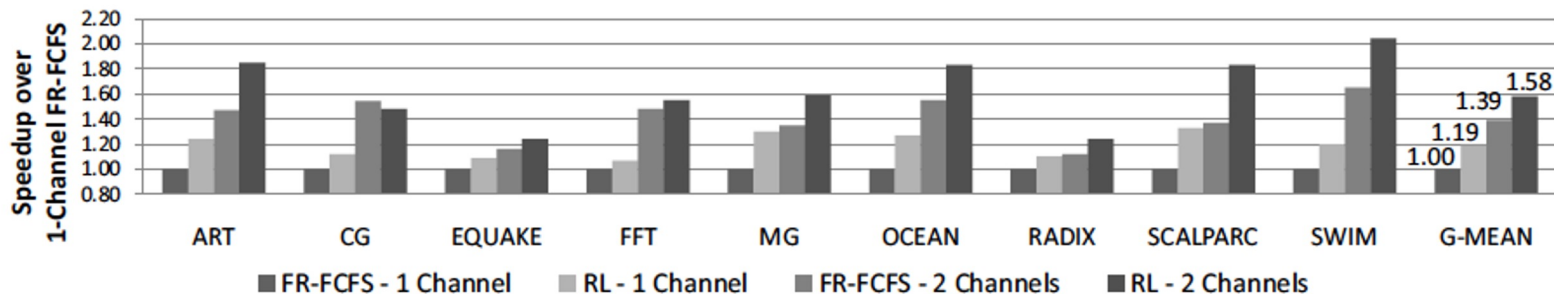


Figure 15: Performance comparison of FR-FCFS and RL-based memory controllers on systems with 6.4GB/s and 12.8GB/s peak DRAM bandwidth

Self Optimizing DRAM Controllers

- + Continuous learning in the presence of changing environment

- + Reduced designer burden in finding a good scheduling policy.

Designer specifies:

- 1) What system variables might be useful

- 2) What target to optimize, but not how to optimize it

- How to specify different objectives? (e.g., fairness, QoS, ...)

- Hardware complexity?

- Design mindset and flow

More on Self-Optimizing DRAM Controllers (I)

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

Self-Optimizing Memory Controllers: A Reinforcement Learning Approach

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

¹Cornell University, Ithaca, NY 14850 USA

²Microsoft Research, Redmond, WA 98052 USA

More on Self-Optimizing DRAM Controllers (II)

- Janani Mukundan and José F. Martinez
“[MORSE: Multi-Objective Reconfigurable Self-Optimizing Memory Scheduler](#)”
Proceedings of the 18th International Symposium on High Performance Computer Architecture (HPCA), New Orleans, Louisiana, February 2012.

MORSE: Multi-objective Reconfigurable Self-optimizing Memory Scheduler

Janani Mukundan José F. Martínez

Computer Systems Laboratory
Cornell University
Ithaca, NY, 14850 USA

<http://m3.csl.cornell.edu/>

Memory Controllers
are critical to research

They will become
even more important

Sibyl: Reinforcement Learning based Data Placement in Hybrid SSDs

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

¹ETH Zürich

²Eindhoven University of Technology

³LIRMM, Univ. Montpellier, CNRS

Sibyl

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

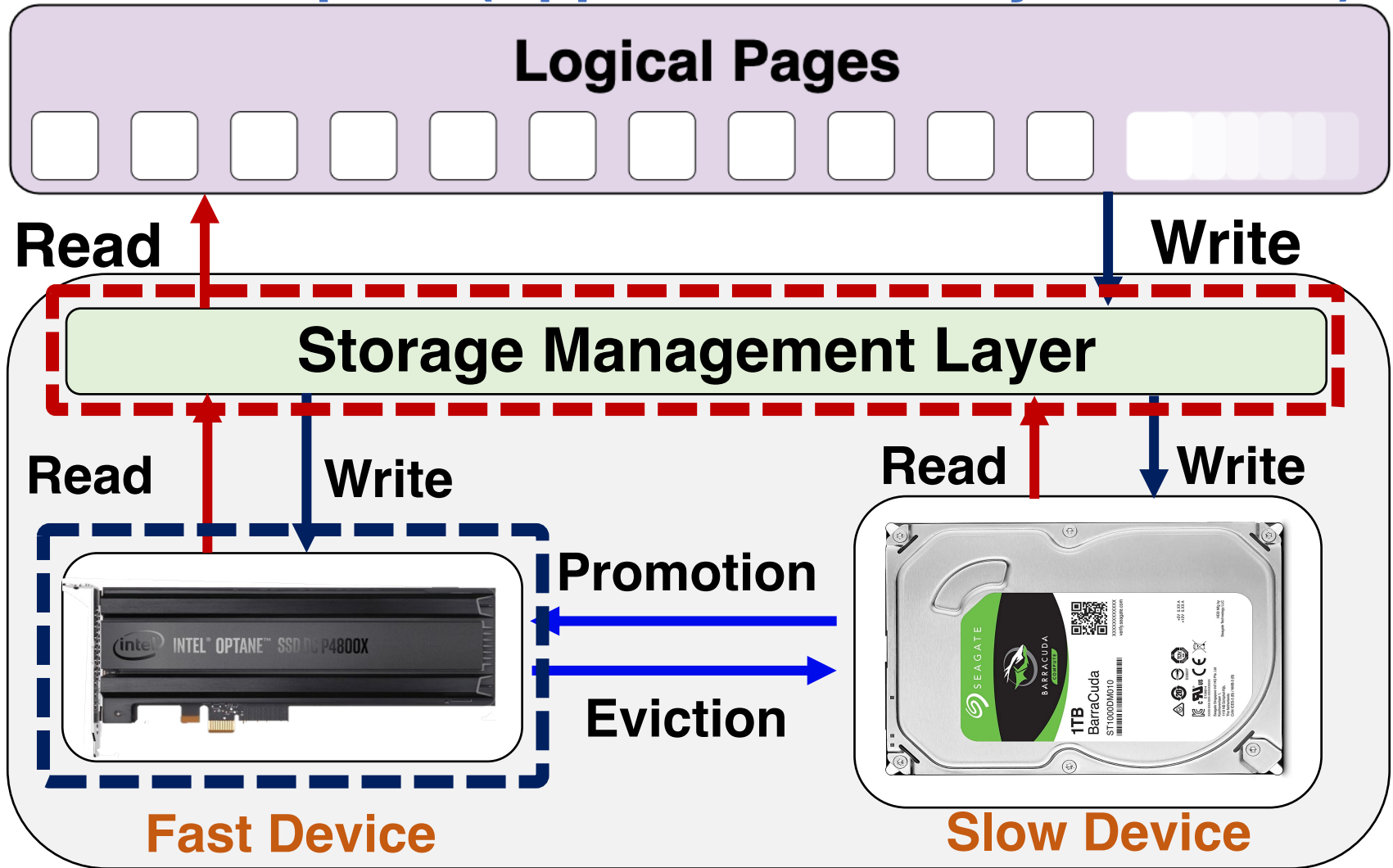
Gagandeep Singh, Rakesh Nadig, Jisung Park,
Rahul Bera, Nastaran Hajinazar, David Novo,
Juan Gómez Luna, Sander Stuijk, Henk Corporaal,
Onur Mutlu

Executive Summary

- **Background:** A hybrid storage system (HSS) uses multiple different storage devices to provide high and scalable storage capacity at high performance
- **Problem:** Two key shortcomings of prior data placement policies:
 - Lack of **adaptivity to:**
 - **Workload changes**
 - **Changes in device types and configurations**
 - Lack of **extensibility** to more devices
- **Goal:** Design a data placement technique that provides:
 - **Adaptivity**, by **continuously learning and adapting** to the **application and underlying device characteristics**
 - **Easy extensibility** to incorporate a wide range of hybrid storage configurations
- **Contribution:** Sibyl, the first reinforcement learning-based data placement technique in hybrid storage systems that:
 - Provides **adaptivity** to changing workload demands and underlying device characteristics
 - Can **easily extend** to any number of storage devices
 - Provides **ease of design and implementation** that requires only a small computation overhead
- **Key Results:** Evaluate on **real systems** using a wide range of workloads
 - Sibyl **improves performance by 21.6%** compared to the best previous data placement technique in dual-HSS configuration
 - In a tri-HSS configuration, Sibyl outperforms the state-of-the-art-policy policy by **48.2%**
 - Sibyl achieves **80% of the performance** of an oracle policy with storage overhead of only **124.4 KiB**

Hybrid Storage System Basics

Address Space (Application/File System View)



Hybrid Storage System

Hybrid Storage System Basics

Logical Address Space (Application/File System View)

Logical Pages

Performance of a hybrid storage system **highly depends** on the **storage management layer's** ability to manage diverse devices and workloads



Hybrid Storage System

Key Shortcomings in Prior Techniques

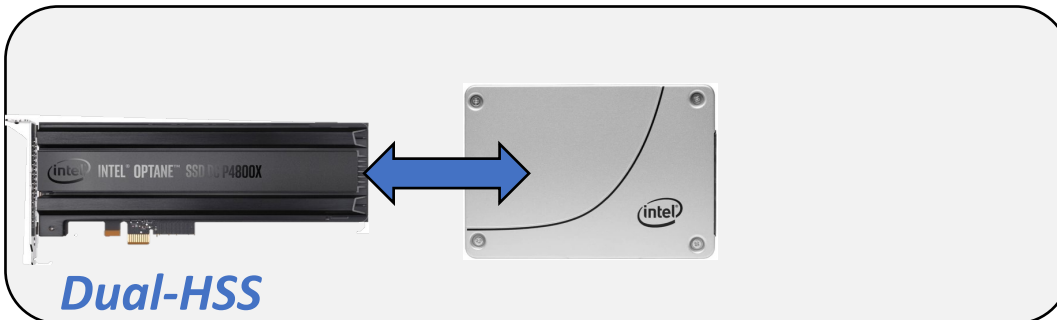
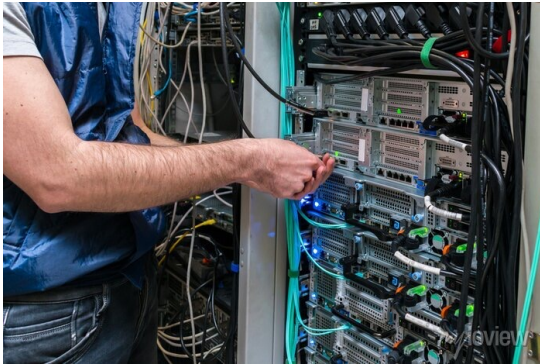
We observe **two key shortcomings** that significantly limit the performance benefits of prior techniques

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

Lack of Extensibility (1/2)

Rigid techniques that require significant effort to accommodate more than two devices

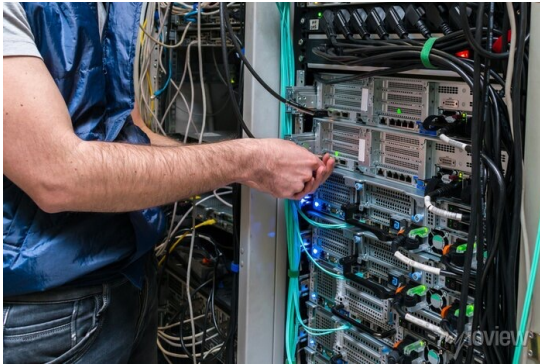
Change in storage configuration



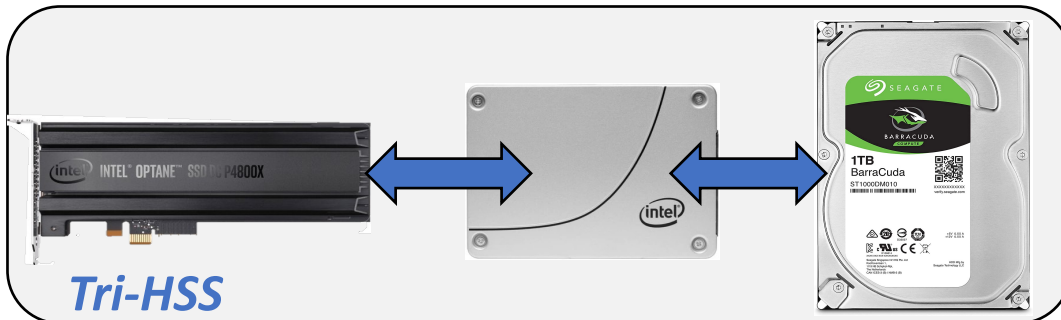
Lack of Extensibility (2/2)

Rigid techniques that require significant effort to accommodate more than two devices

Change in storage configuration



Design a new policy



Our Goal

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

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

Our Proposal



Sibyl

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

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

Basics of Reinforcement Learning (RL)



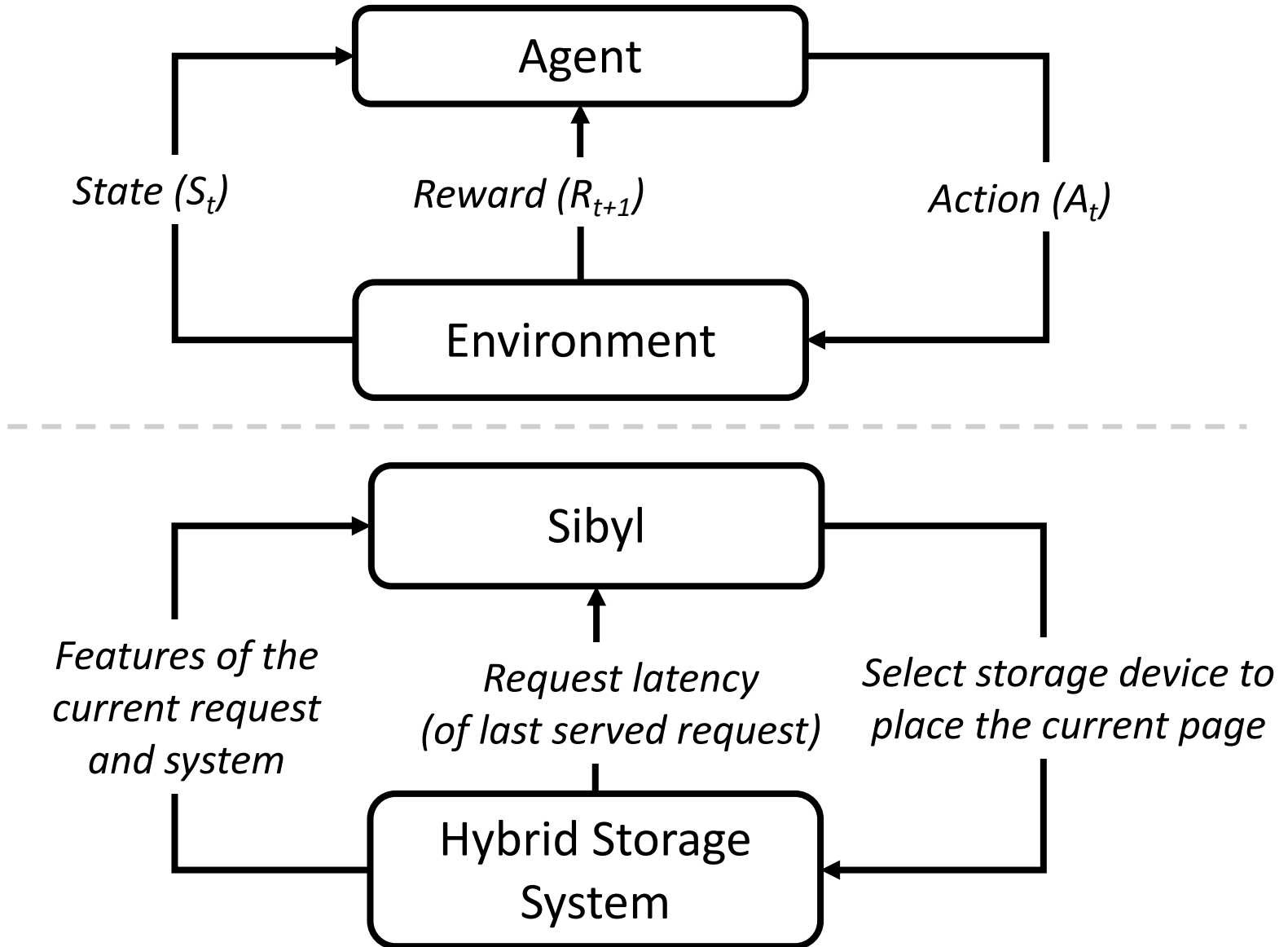
The diagram illustrates the basic components of Reinforcement Learning. It consists of two main parts: an 'Agent' and an 'Environment'. The 'Agent' is represented by a rounded rectangular box at the top, and the 'Environment' is represented by a larger rounded rectangular box at the bottom. The 'Environment' box is positioned directly below the 'Agent' box, indicating a direct interaction between them.

Agent

Environment

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

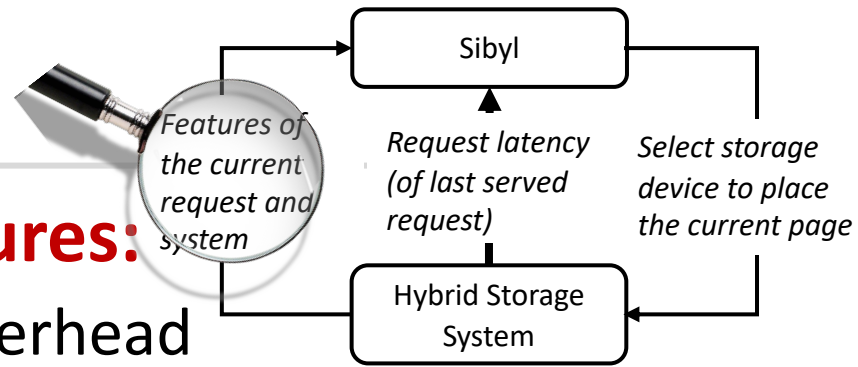
Formulating Data Placement as RL



What is State?

- **Limited number of state features:**

- Reduce the implementation overhead
- RL agent is more sensitive to reward



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

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

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

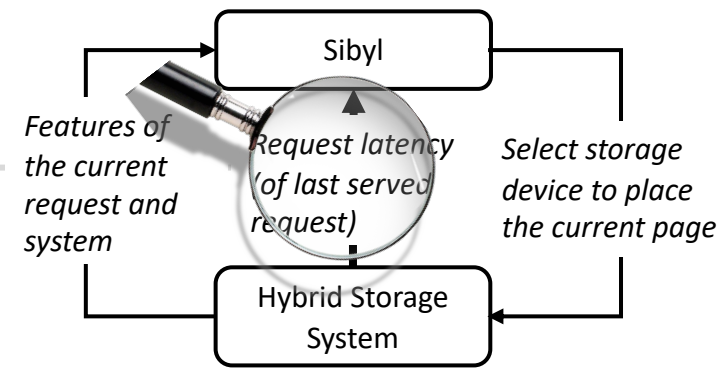
Selected State Attributes

Table 1: State features used by Sibyl

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

What is Reward?

- Defines the **objective** of Sibyl



- We formulate the reward as a function of the **request latency**
- Encapsulates three key aspects:
 - **Internal state of the device** (e.g., read/write latencies, the latency of garbage collection, queuing delays, ...)
 - **Throughput**
 - **Evictions**
- More details in the paper

Reward Function

Reward. After every data placement decision at time-step⁴ t , Sibyl gets a reward from the environment at time-step $t + 1$ that acts as a feedback to Sibyl's previous action. To achieve Sibyl's performance goal, we craft the reward function R as follows:

$$R = \begin{cases} \frac{1}{L_t} & \text{if no eviction of a page from the} \\ & \text{fast storage to the slow storage} \\ \max(0, \frac{1}{L_t} - R_p) & \text{in case of eviction} \end{cases} \quad (1)$$

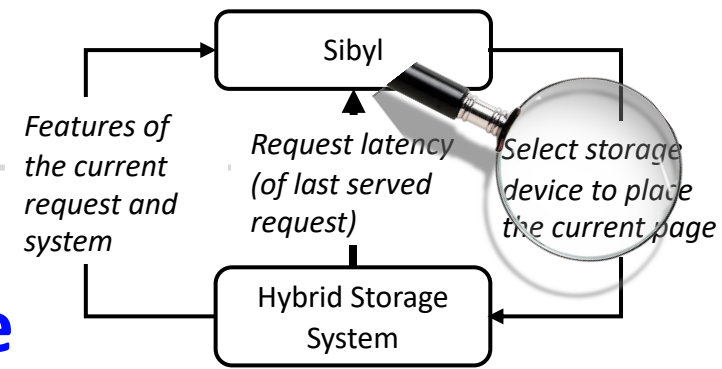
where L_t and R_p represent the last served request latency and eviction penalty, respectively. If the fast storage is running out of free space, there might be evictions in the background from the fast

⁴In HSS, a time-step is defined as a new storage request.

storage to the slow storage. Therefore, we add an eviction penalty (R_p) to guide Sibyl to place only performance-critical pages in the fast storage. We empirically select R_p to be equal to $0.001 \times L_e$ (L_e is the time spent in evicting pages from the fast storage to the slow storage), which prevents the agent from aggressively placing all requests into the fast storage device.

What is Action?

- At every new page request, the action is to **select a storage device**
- Action can be **easily extended** to any number of storage devices
- Sibyl **evicts** a page when the fast device utilization is 100%
- Sibyl **promotes** a page when there is an update from the application



Talk Outline

Key Shortcomings of Prior Data Placement Techniques

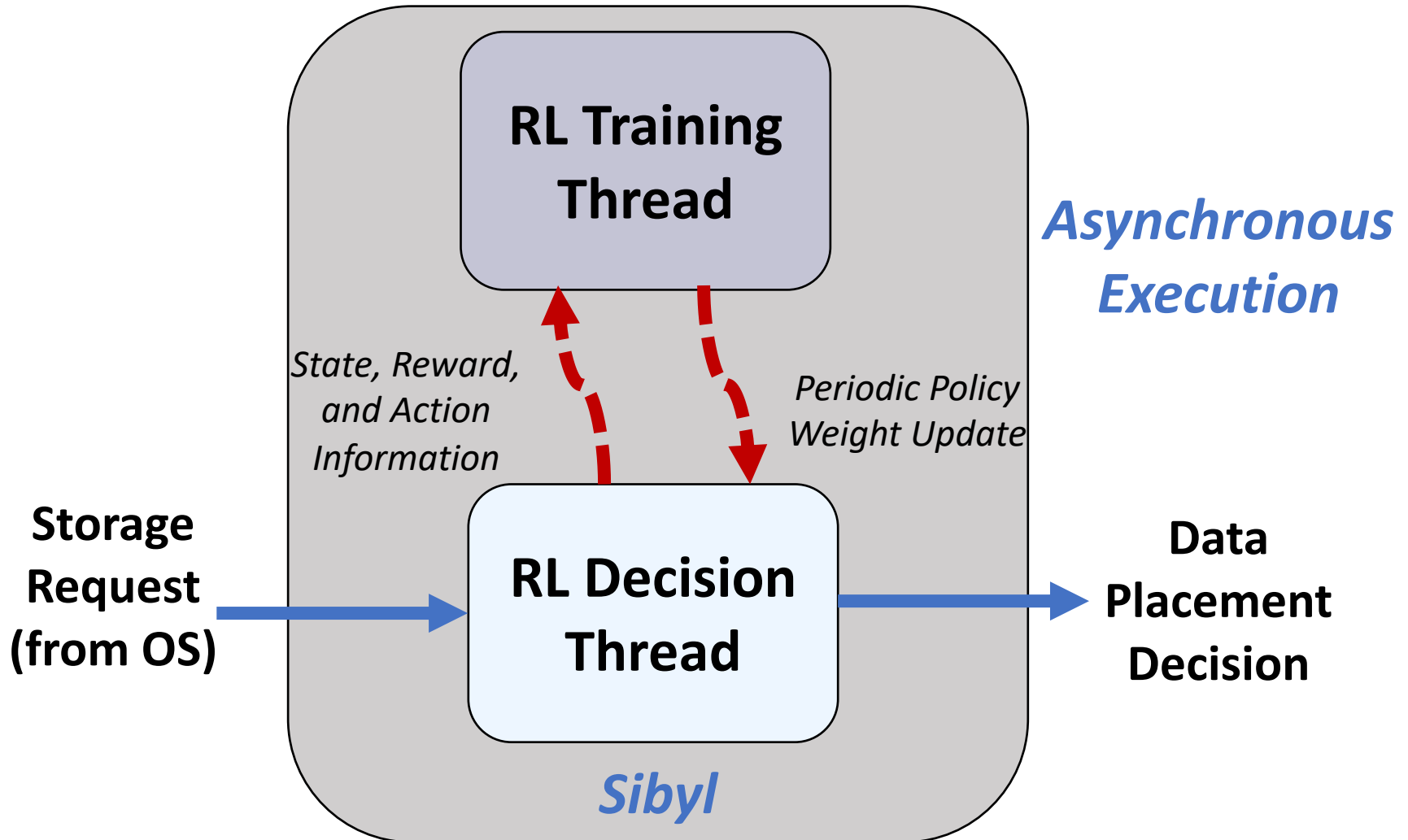
Formulating Data Placement as Reinforcement Learning

Sibyl: Overview

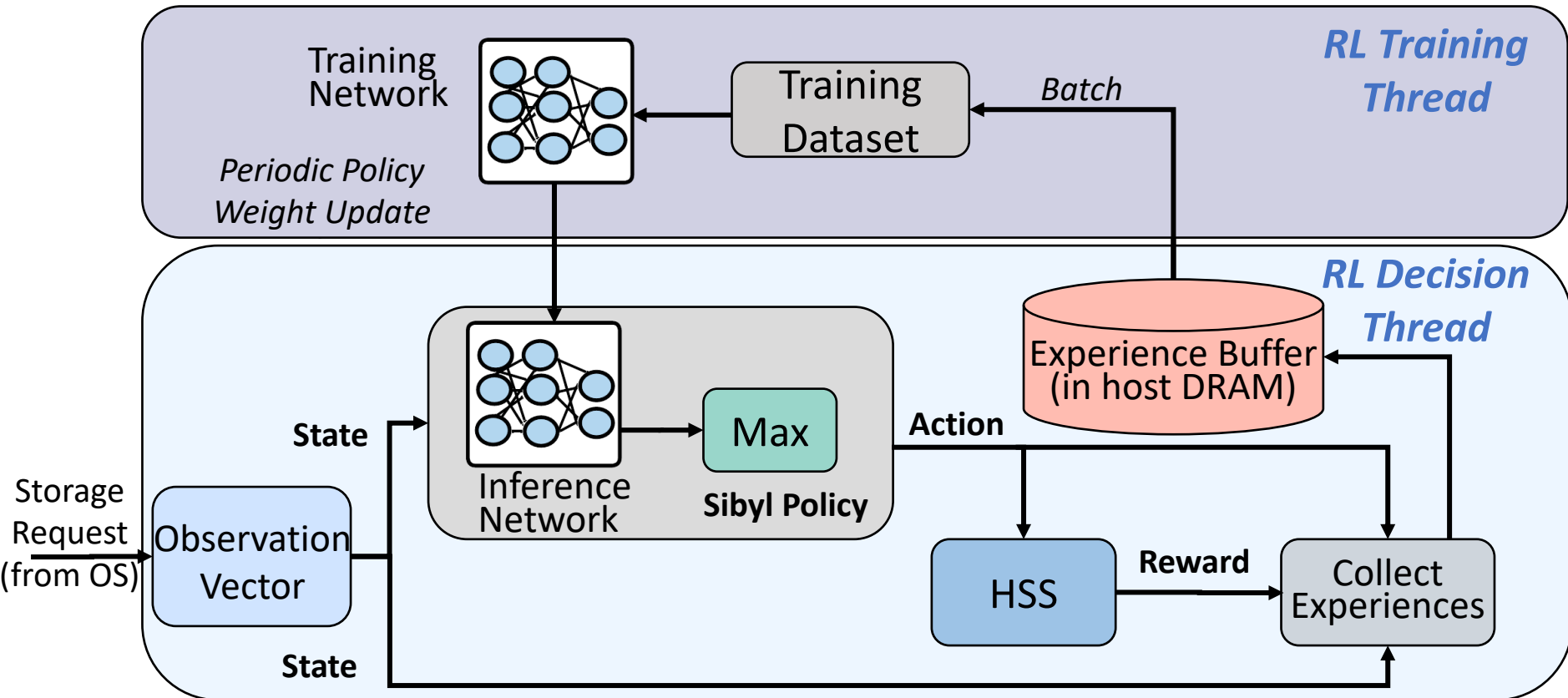
Evaluation of Sibyl and Key Results

Conclusion

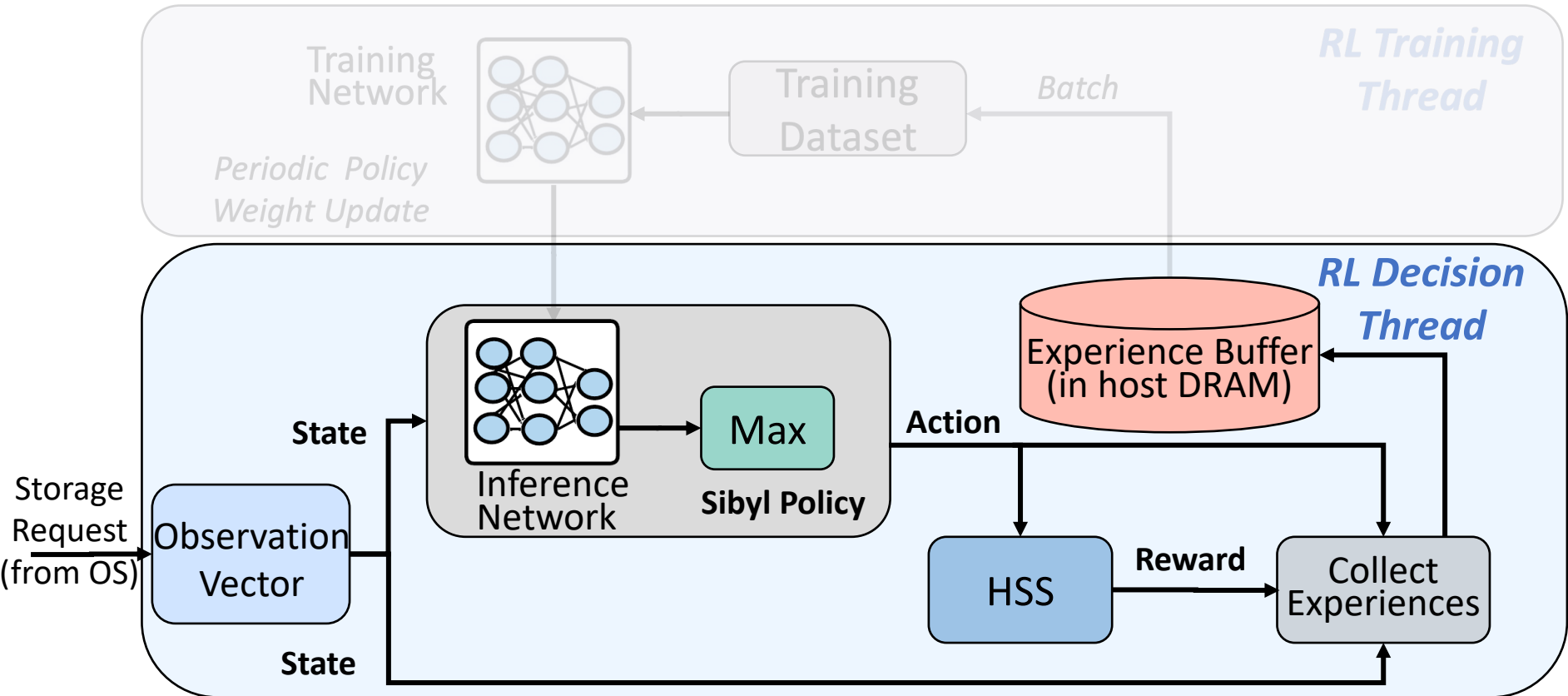
Sibyl Execution



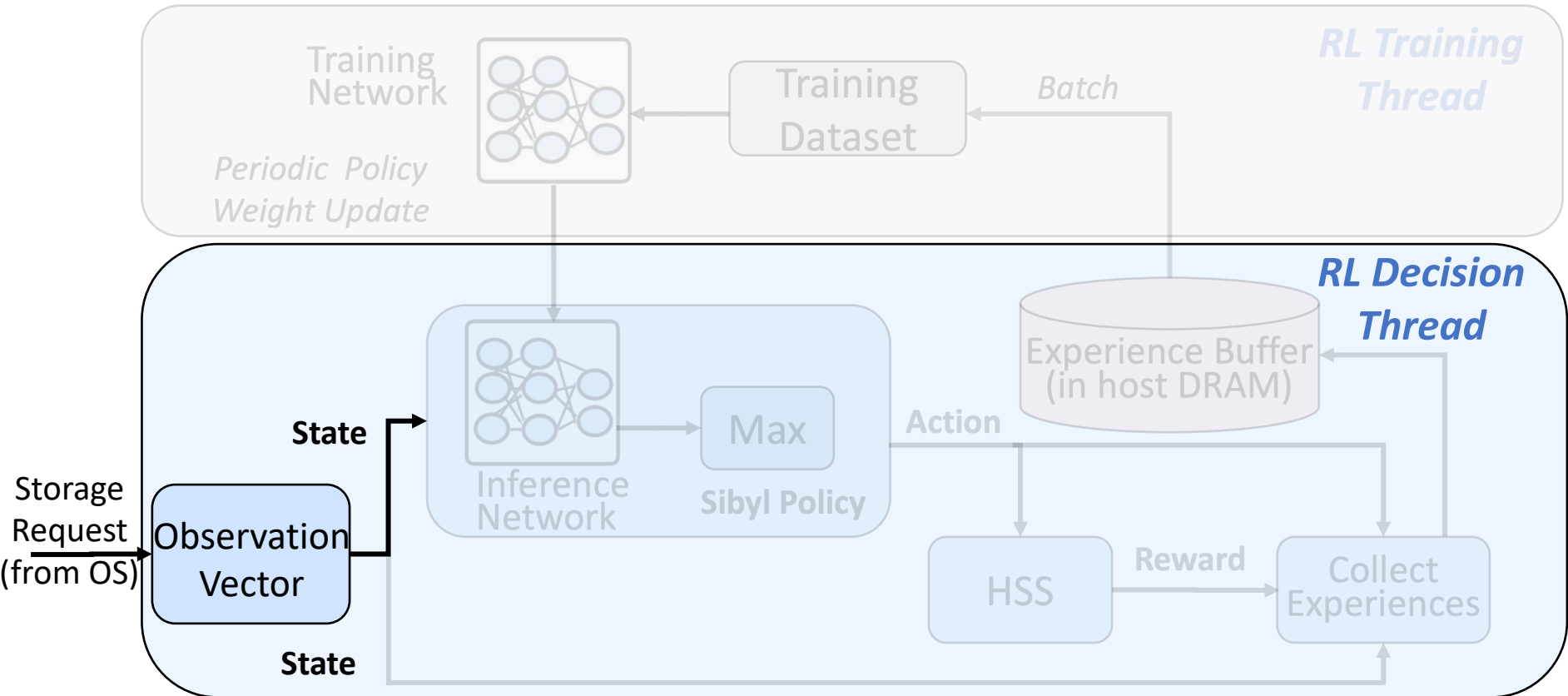
Sibyl Design: Overview



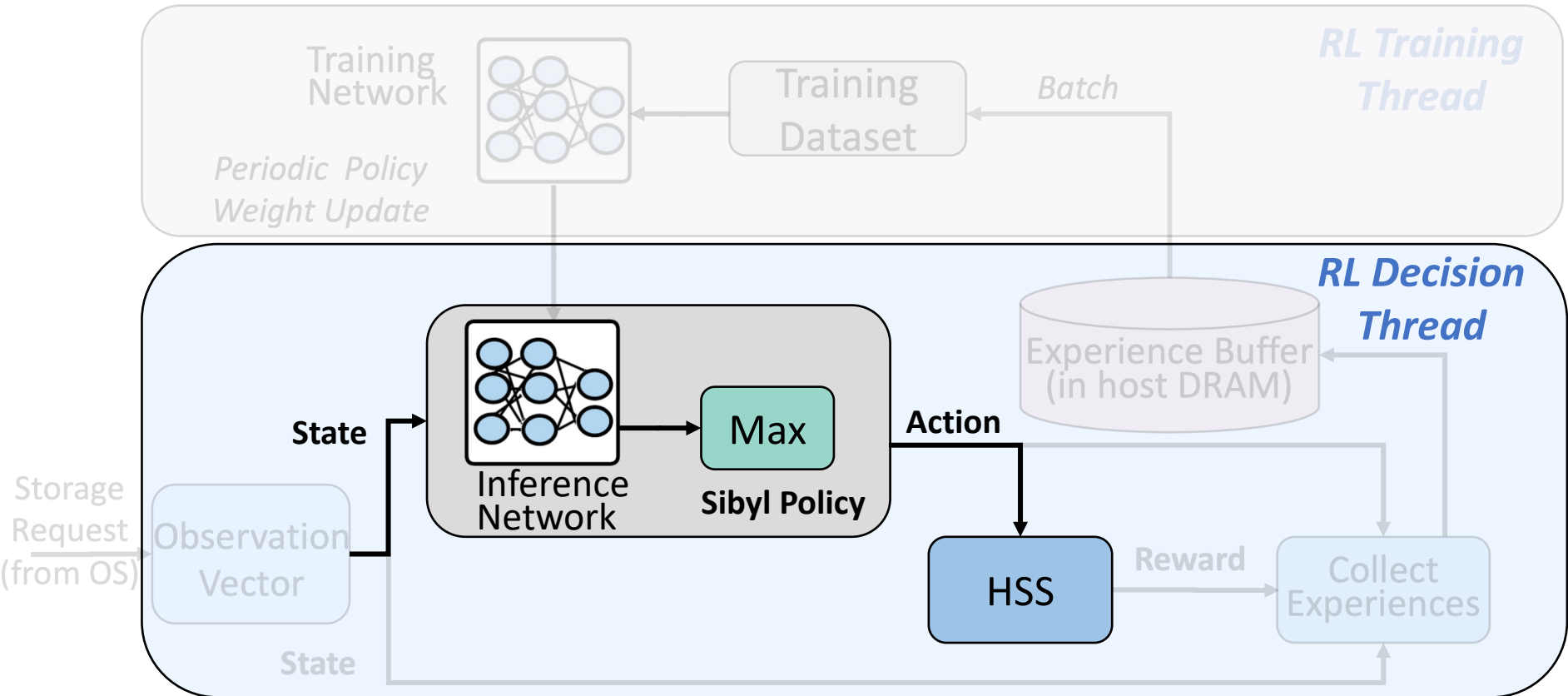
RL Decision Thread



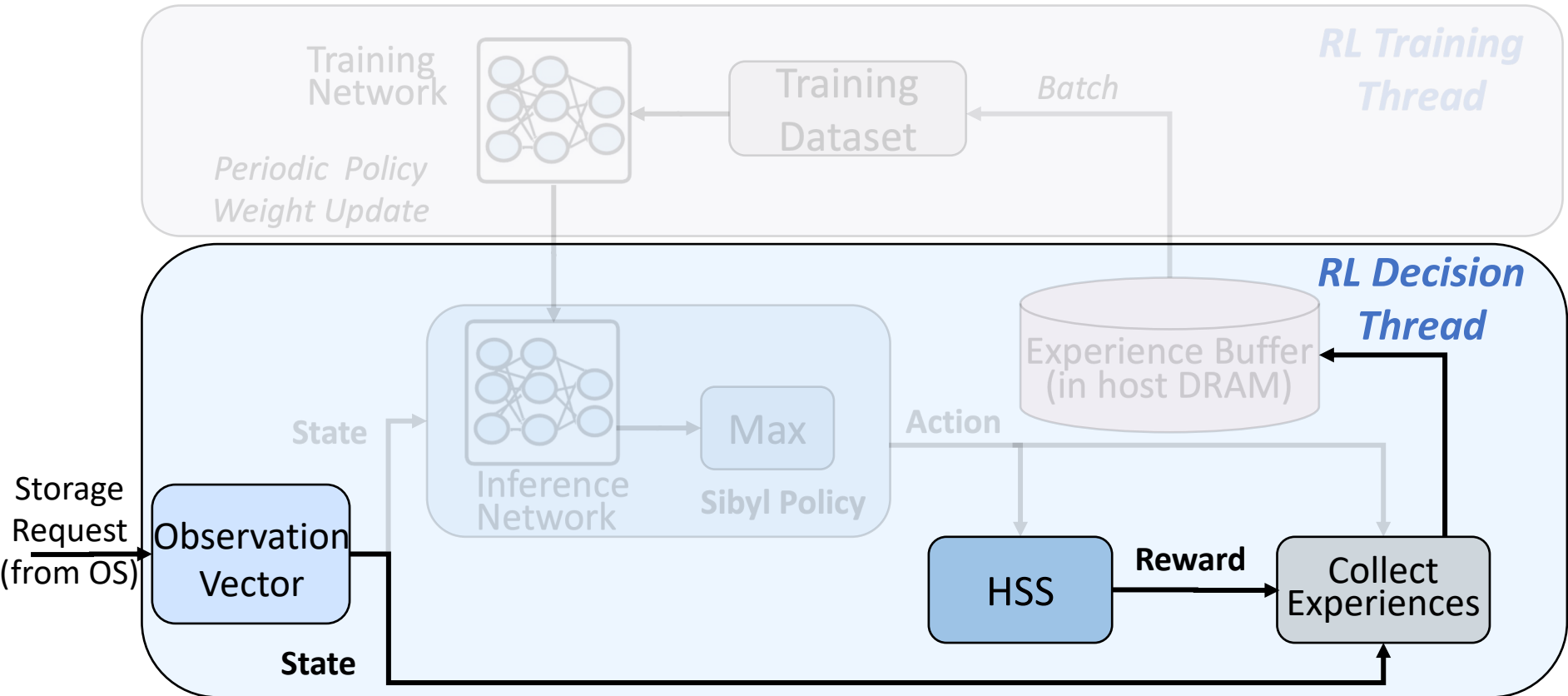
RL Decision Thread



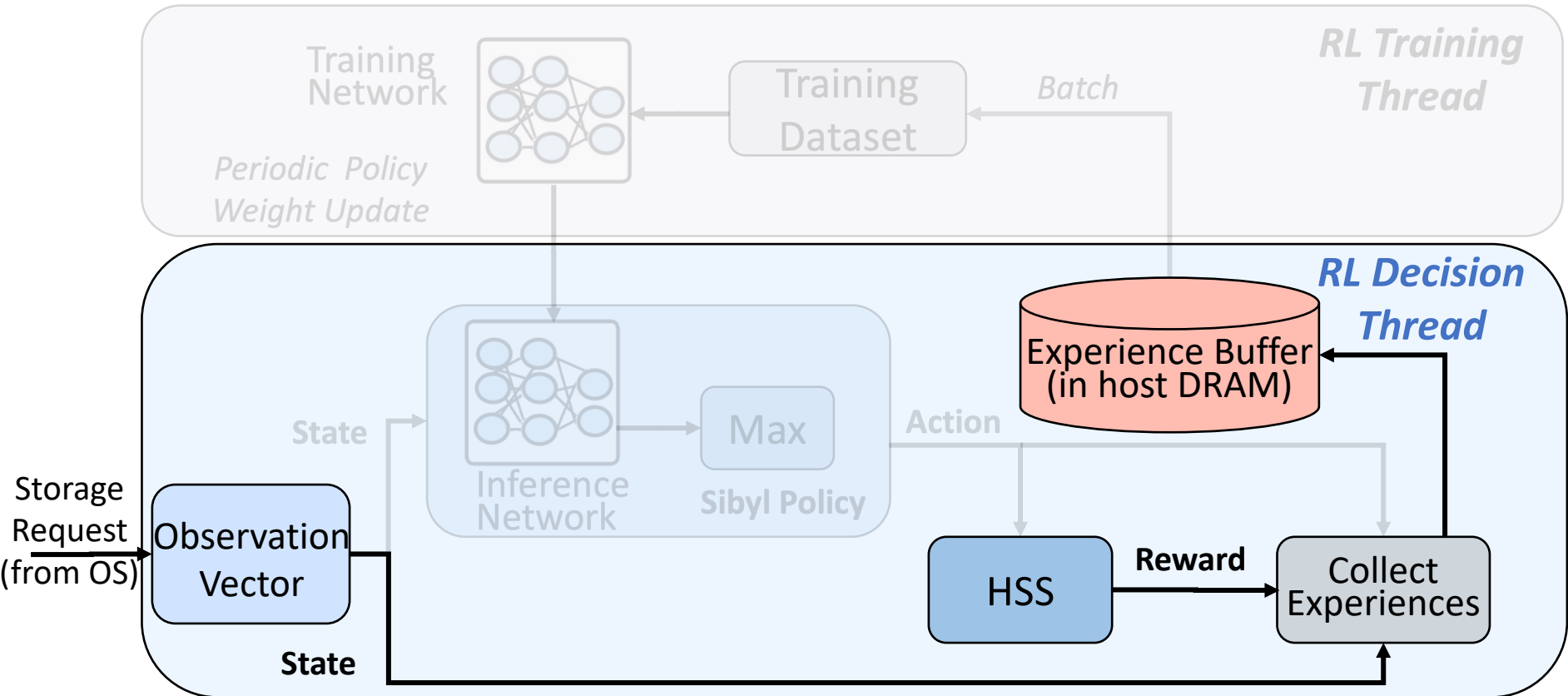
RL Decision Thread



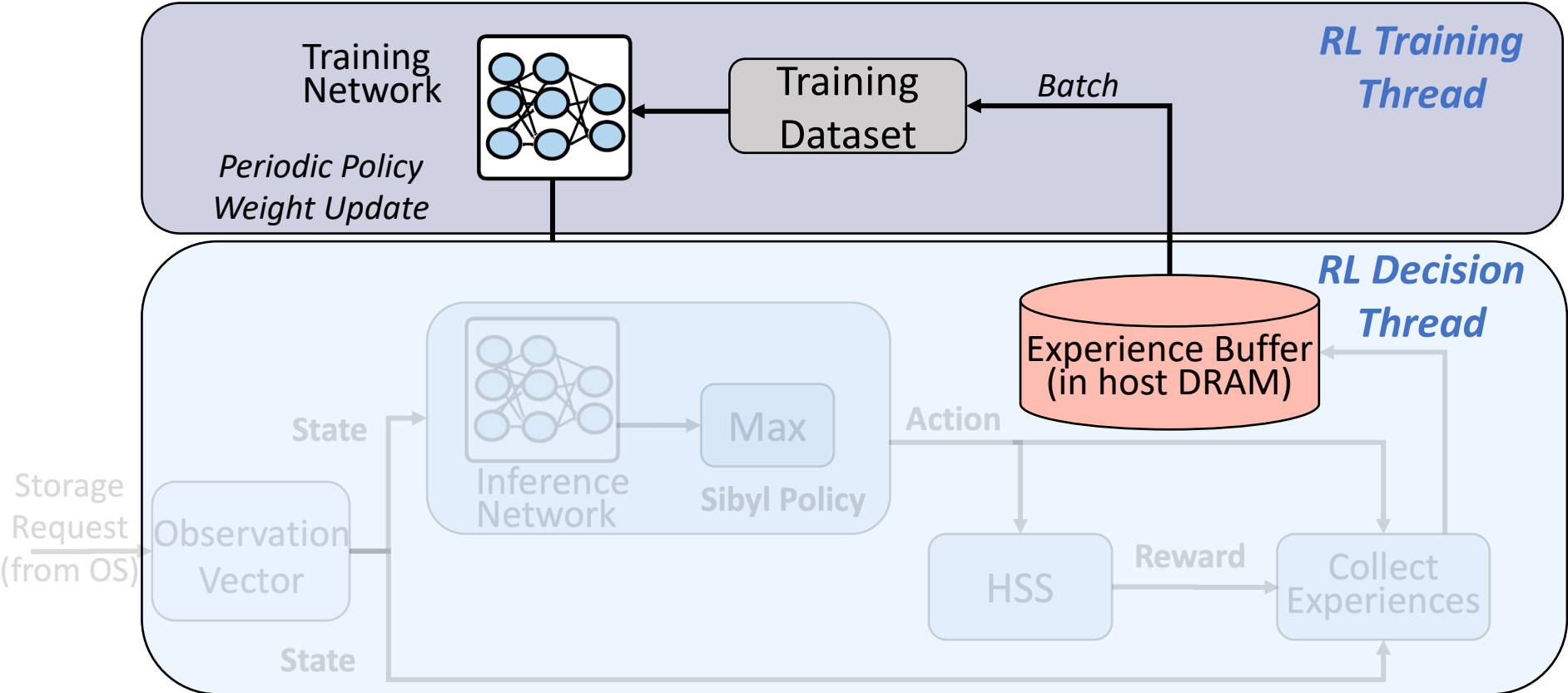
RL Decision Thread



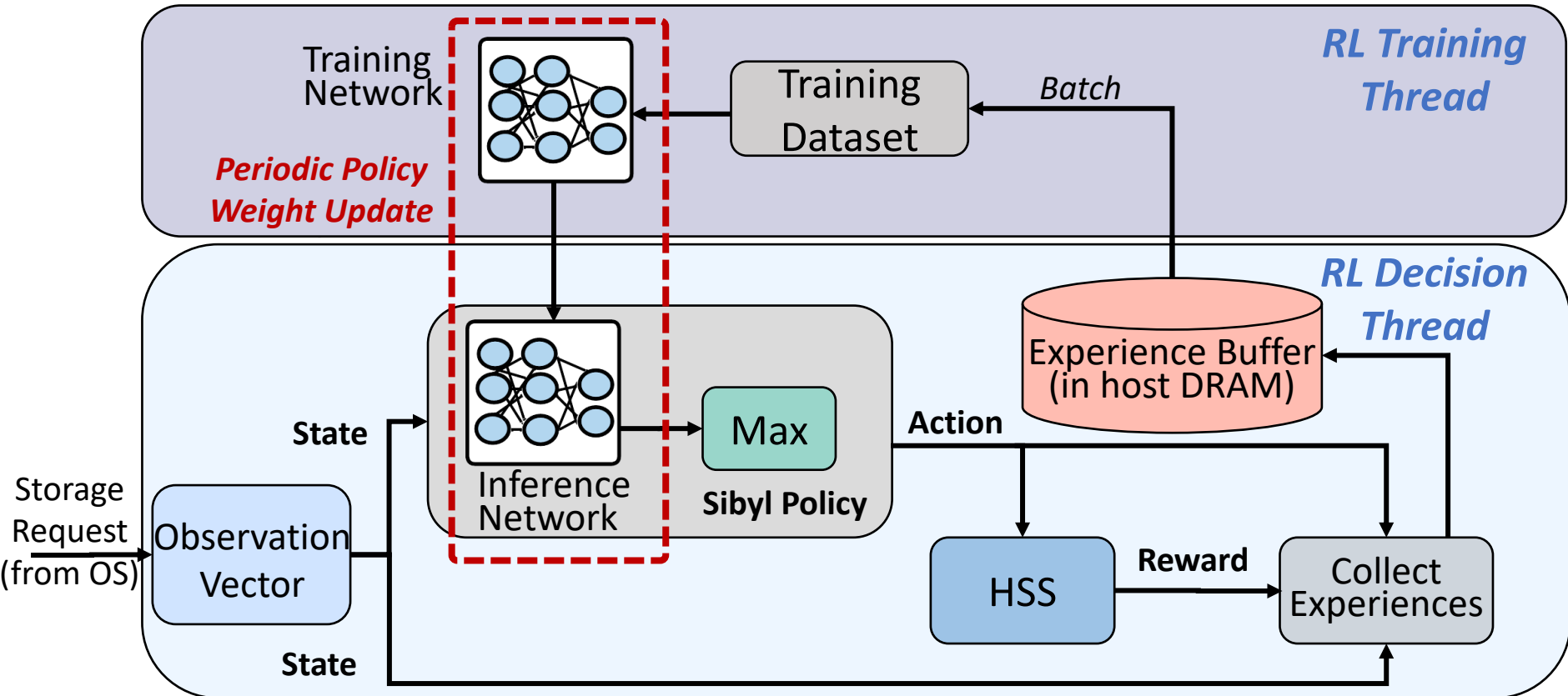
RL Decision Thread



RL Training Thread



Periodic Weight Transfer



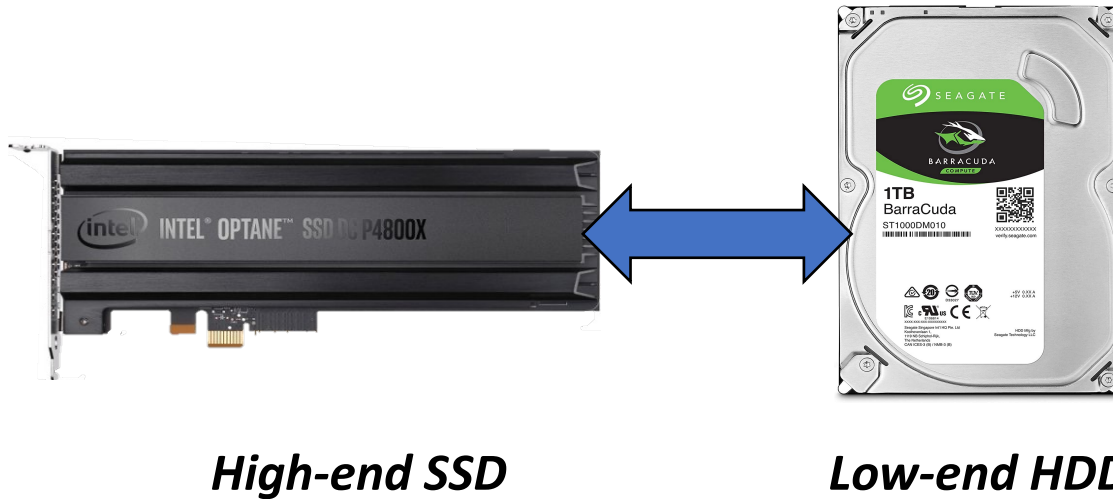
Evaluation Methodology (1/3)

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

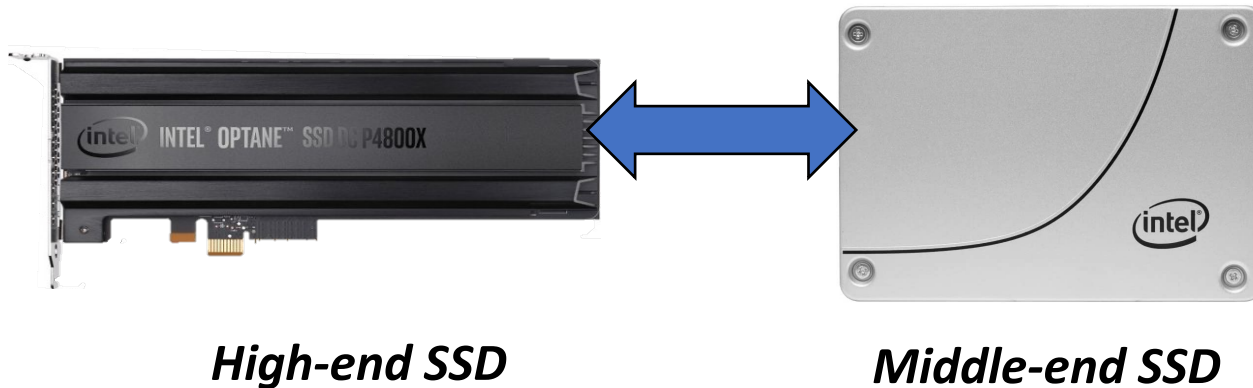


Evaluation Methodology (2/3)

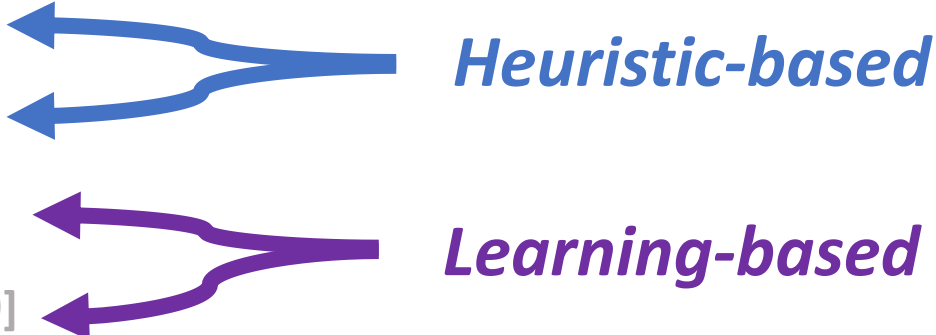
Cost-Oriented HSS Configuration



Performance-Oriented HSS Configuration



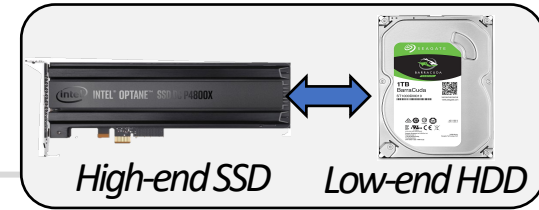
Evaluation Methodology (3/3)

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

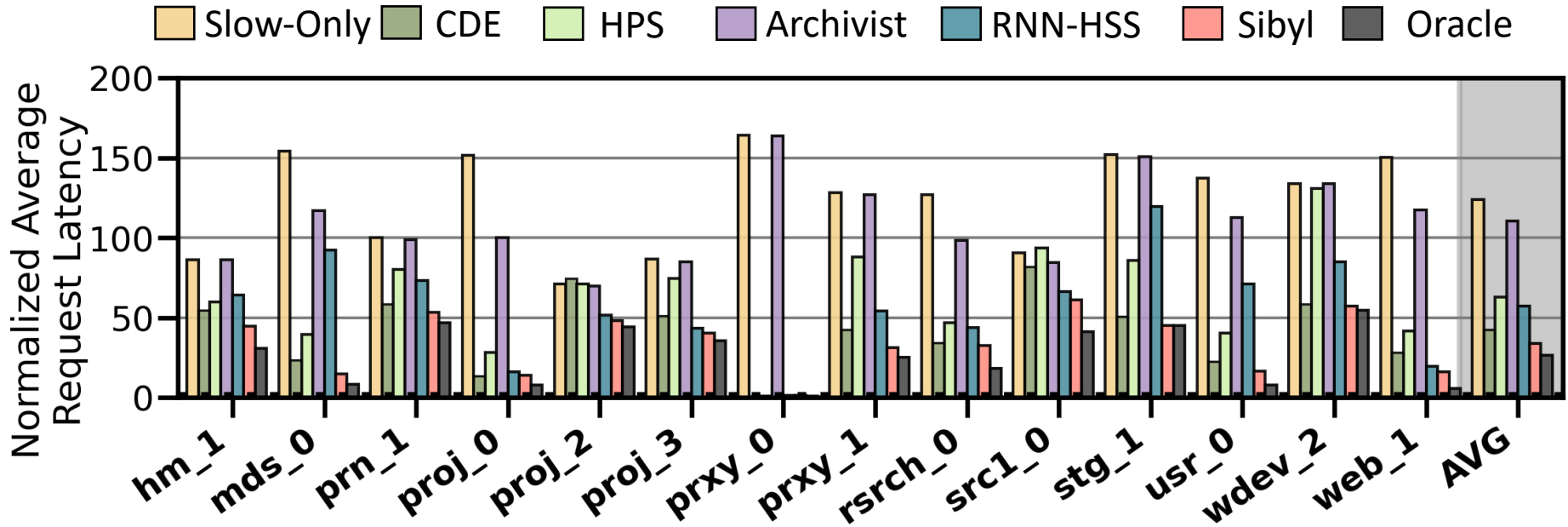
Heuristic-based

Learning-based

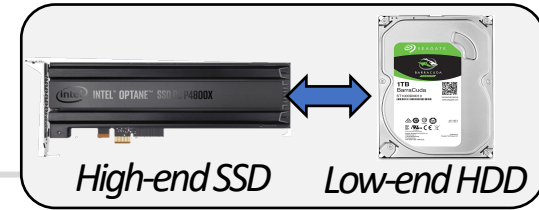
Performance Analysis



Cost-Oriented HSS Configuration

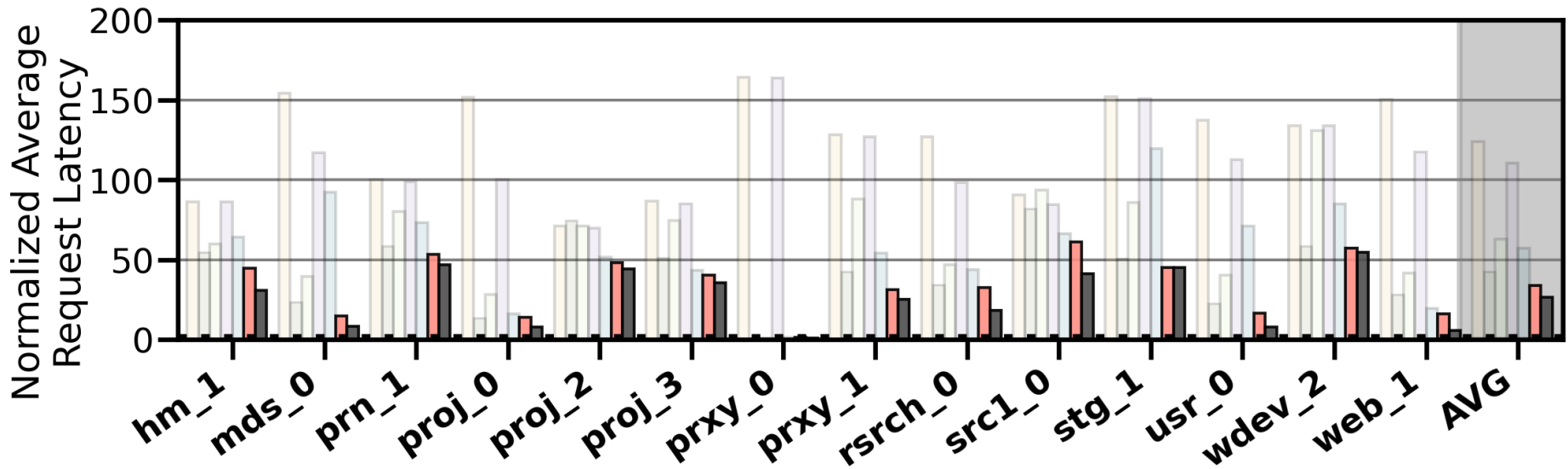


Performance Analysis



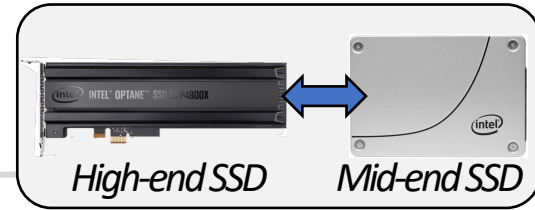
Cost-Oriented HSS Configuration

Slow-Only CDE HPS Archivist RNN-HSS Sibyl Oracle

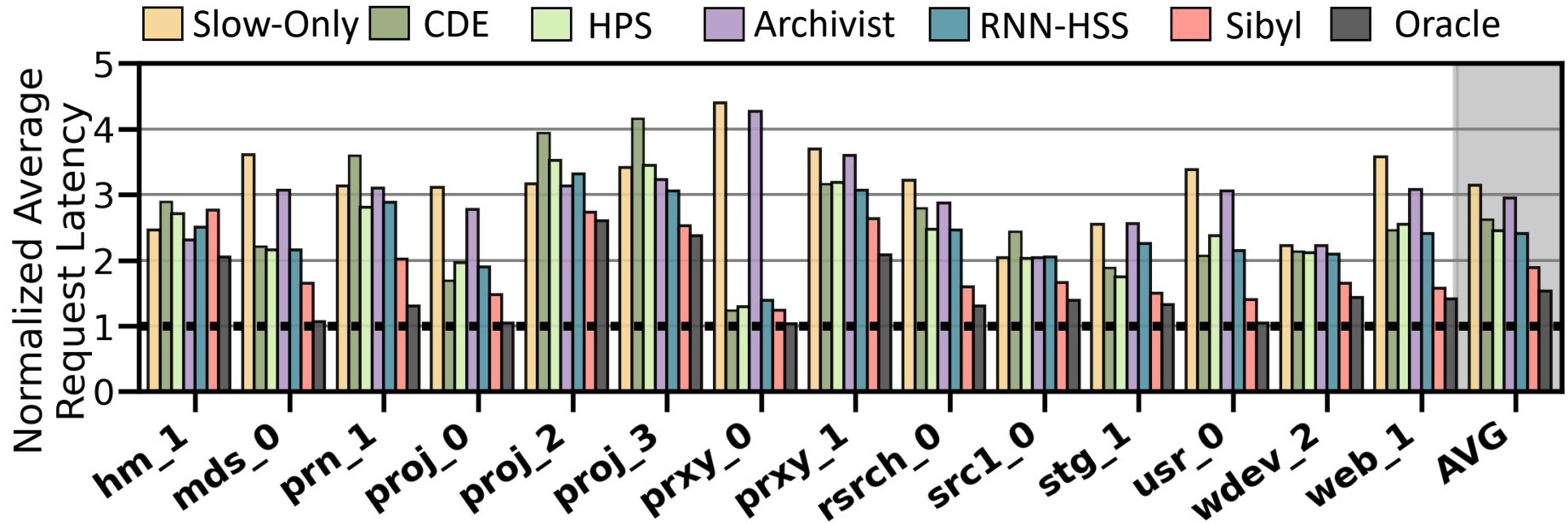


Sibyl consistently **outperforms all the baselines**
for all the workloads

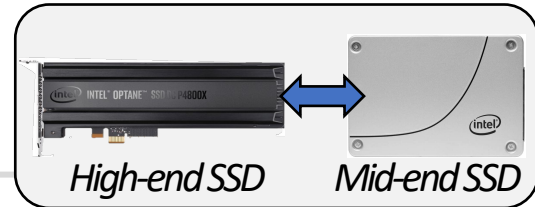
Performance Analysis



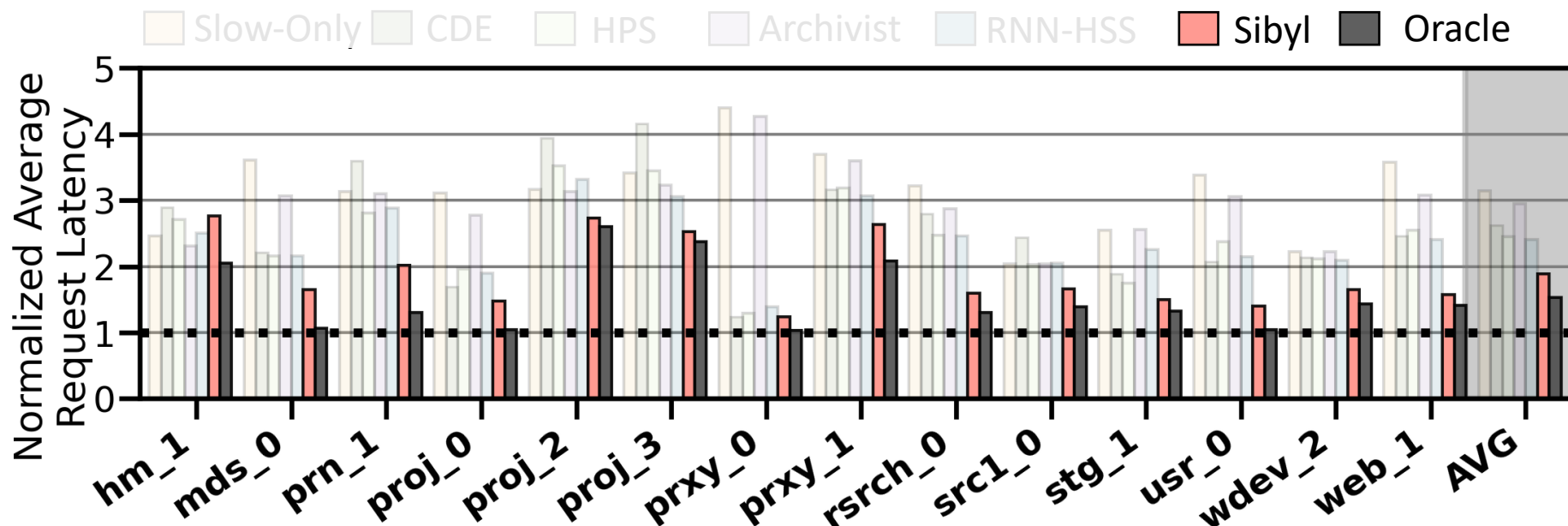
Performance-Oriented HSS Configuration



Performance Analysis

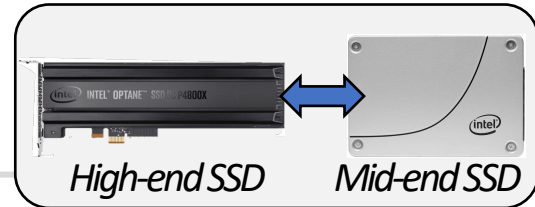


Performance-Oriented HSS Configuration

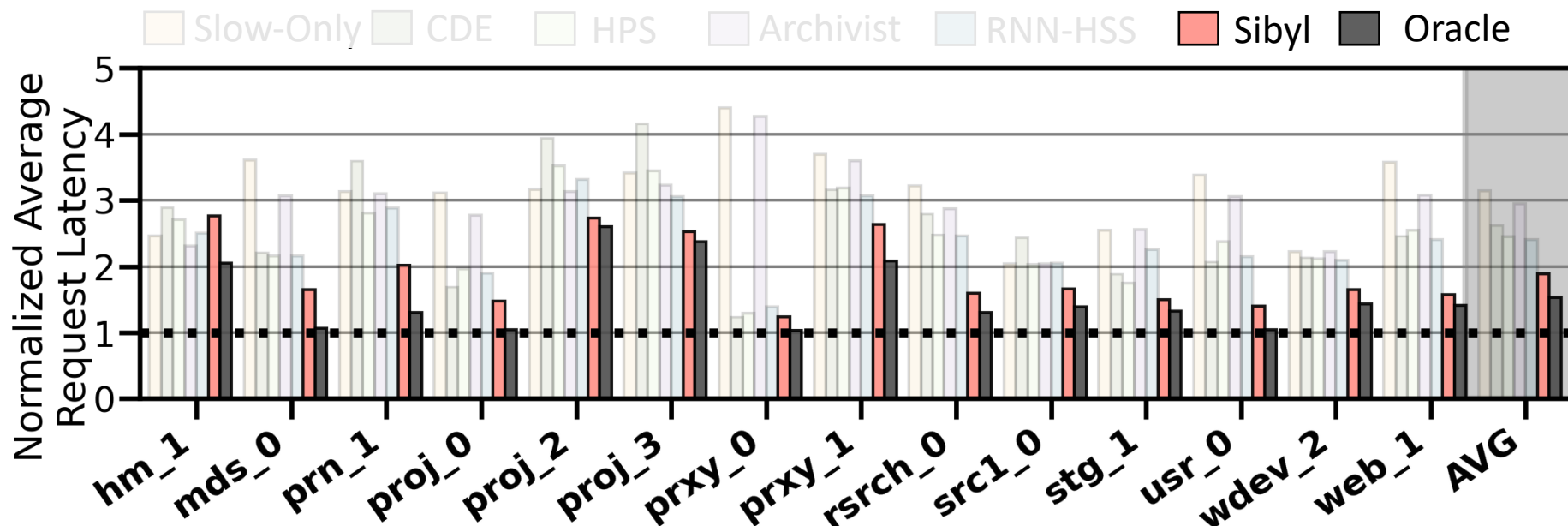


Sibyl provides **21.6% performance improvement** by **dynamically adapting its data placement policy**

Performance Analysis

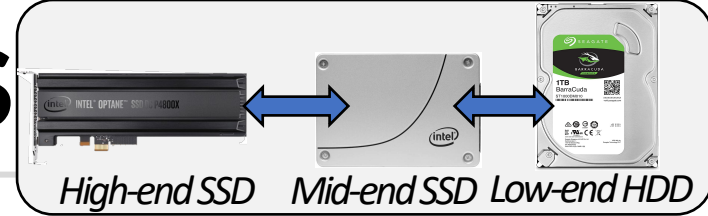


Performance-Oriented HSS Configuration



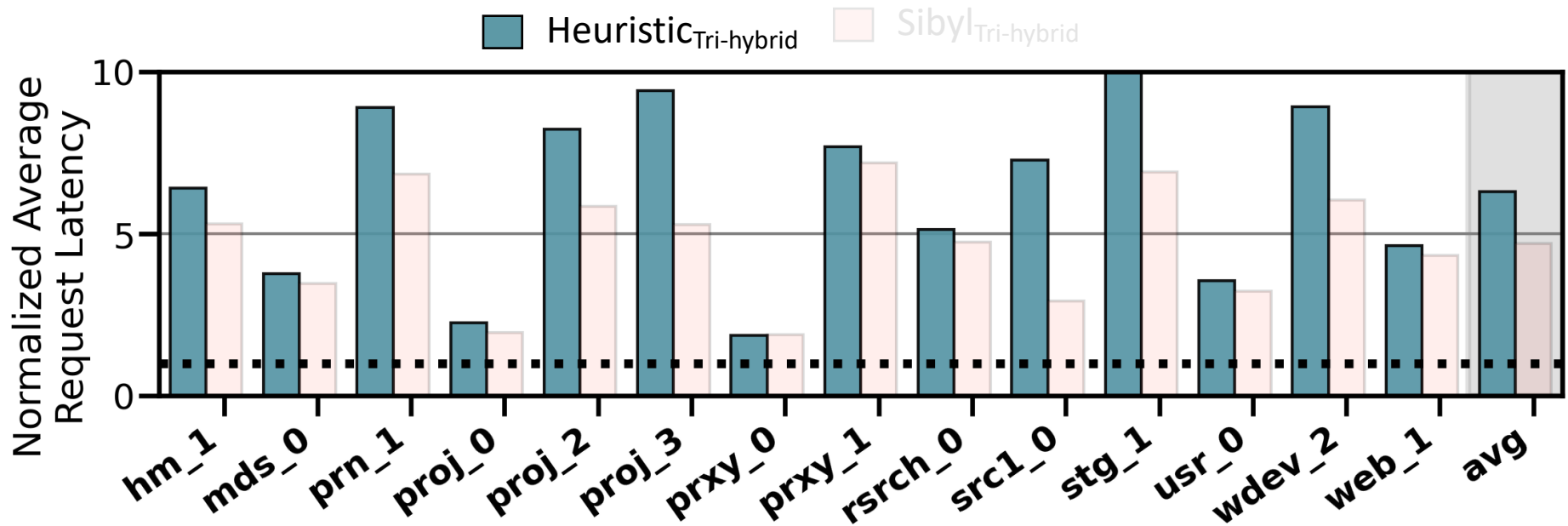
Sibyl achieves **80% of the performance of an oracle policy** that has complete knowledge of future access patterns

Performance on Tri-HSS

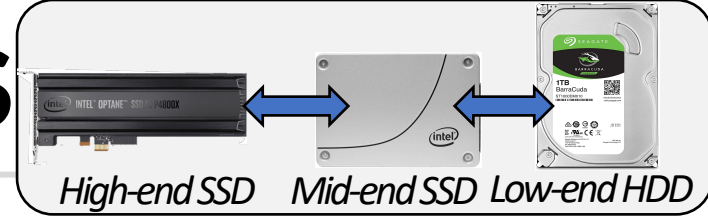


Extending Sibyl for **more devices**:

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

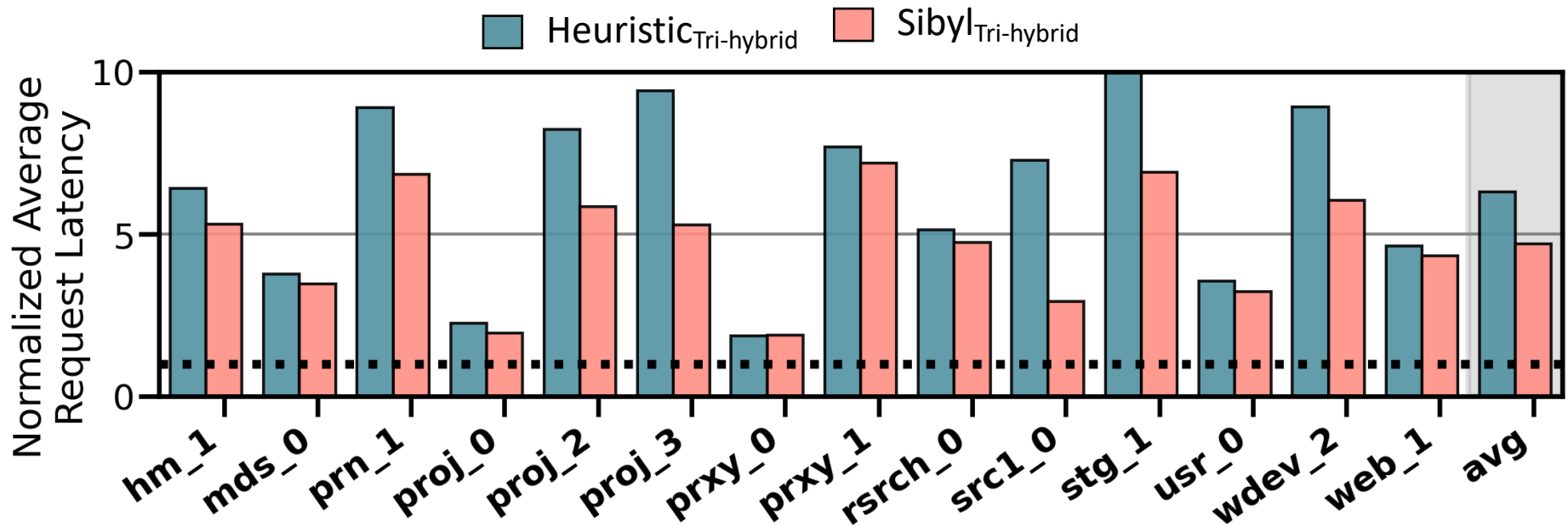


Performance on Tri-HSS

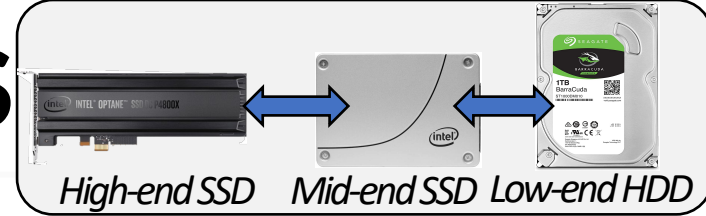


Extending Sibyl for **more devices**:

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



Performance on Tri-HSS



Extending Sibyl for **more devices**:

1. Add a new action

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

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

Sibyl's Overhead

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



Small inference overhead



Satisfies prediction latency

More in the Paper (1/3)

- **Throughput (IOPS) evaluation**

- Sibyl provides high IOPS compared to baseline policies because it **indirectly captures throughput (size/latency)**

- Evaluation on **unseen workloads**

- Sibyl can **effectively adapt** its policy to highly dynamic workloads

- Evaluation on **mixed workloads**

- Sibyl provides **equally-high performance** benefits as in single workloads

More in the Paper (2/3)

- Evaluation on **different features**
 - Sibyl **autonomously decides** which features are important to maximize the performance
- Evaluation with **different hyperparameter values**
- Sensitivity to **fast storage capacity**
 - Sibyl **provides scalability by dynamically adapting** its policy to available storage size
- **Explainability analysis** of Sibyl's decision making
 - **Explain Sibyl's actions** for different workload characteristics and device configurations

More in the Paper (3/3)

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

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

¹ETH Zürich

²Eindhoven University of Technology

³LIRMM, Univ. Montpellier, CNRS

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

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

Conclusion

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

Major Directions

- Consider **other optimization objectives**
 - **Energy consumption**, **endurance** of storage devices.....
 - Design **better reward** structures
- **Optimize data migration** in hybrid storage systems
 - Explore machine learning (ML) techniques to make data migration **adaptive** and **extensible**
 - How do we **coordinate multiple ML techniques**?
- How do we improve these policies in **other heterogeneous memory systems**?
 - DRAM + NVM, CPU Caches + DRAM
 - Design RL models keeping **latency constraints** in mind

Sibyl Paper, Slides, Videos

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

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

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

¹ETH Zürich

²Eindhoven University of Technology

³LIRMM, Univ. Montpellier, CNRS

SSD Course (Spring 2023)

Spring 2023 Edition:

- https://safari.ethz.ch/projects_and_seminars/spring2023/doku.php?id=modern_ssd

Fall 2022 Edition:

- https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=modern_ssd

Youtube Livestream (Spring 2023):

- https://www.youtube.com/watch?v=4VTwOMmsnJY&list=PL5Q2soXY2Zi_8qOM5Icpp8hB2SHtm4z57&pp=iAQB

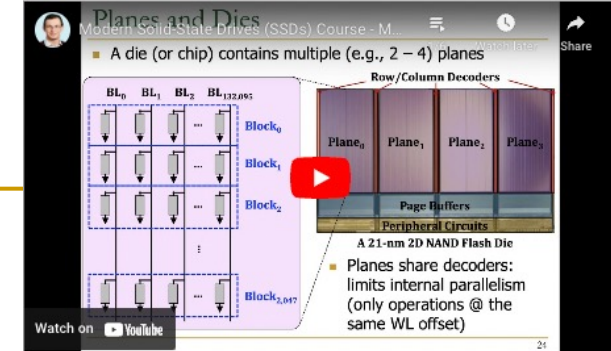
Youtube Livestream (Fall 2022):

- <https://www.youtube.com/watch?v=hqLrd-Uj0aU&list=PL5Q2soXY2Zi9BJhenUq4JI5bwhAMpAp13&pp=iAQB>

Project course

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

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



Fall 2022 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	06.10		M1: P&S Course Presentation PDF PPT	Required Recommended	
W2	12.10	YouTube Live	M2: Basics of NAND Flash-Based SSDs PDF PPT	Required Recommended	
W3	19.10	YouTube Live	M3: NAND Flash Read/Write Operations PDF PPT	Required Recommended	
W4	26.10	YouTube Live	M4: Processing inside NAND Flash PDF PPT	Required Recommended	
W5	02.11	YouTube Live	M5: Advanced NAND Flash Commands & Mapping PDF PPT	Required Recommended	
W6	09.11	YouTube Live	M6: Processing inside Storage PDF PPT	Required Recommended	
W7	23.11	YouTube Live	M7: Address Mapping & Garbage Collection PDF PPT	Required Recommended	
W8	30.11	YouTube Live	M8: Introduction to MQSim PDF PPT	Required Recommended	
W9	14.12	YouTube Live	M9: Fine-Grained Mapping and Multi-Plane Operation-Aware Block Management PDF PPT	Required Recommended	
W10	04.01.2023	YouTube Premiere	M10a: NAND Flash Basics PDF PPT	Required Recommended	
			M10b: Reducing Solid-State Drive Read Latency by Optimizing Read-Retry PDF PPT Paper	Required Recommended	
			M10c: Evanescence: Architectural Support for Efficient Data Sanitization in Modern Flash-Based Storage Systems PDF PPT Paper	Required Recommended	
			M10d: DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression PDF PPT Paper	Required Recommended	
W11	11.01	YouTube Live	M11: FLIN: Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives PDF PPT	Required	
W12	25.01	YouTube Premiere	M12: Flash Memory and Solid-State Drives PDF PPT	Recommended	

Comp Arch (Fall 2021)

Computer Architecture - Fall 2021

Recent Changes Media Manager Sitemap

Trace: readings start schedule

Home

Announcements

Materials

Lectures/Schedule

Lecture Buzzwords

Readings

HWs

Labs

Exams

Related Courses

Tutorials

Resources

Computer Architecture FS20: Course Webpage

Computer Architecture FS20: Lecture Videos

Digitaltechnik SS21: Course Webpage

Digitaltechnik SS21: Lecture Videos

Moodle

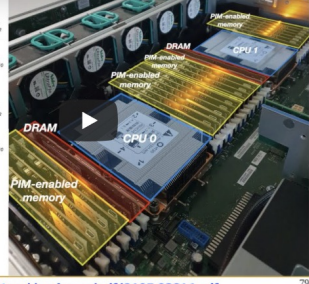
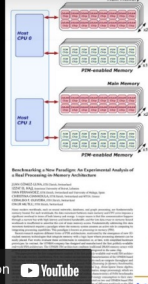
HotCRP

Verilog Practice Website (HDLBits)

Lecture Video Playlist on YouTube

Livestream Lecture Playlist

2.560-DPU Processing in Memory System



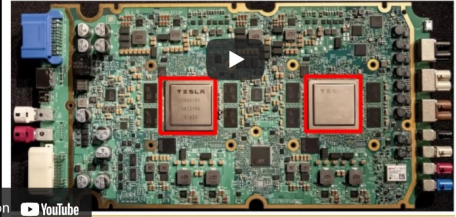
Watch on YouTube

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

Recorded Lecture Playlist

TESLA Full Self-Driving Computer (2021)

- ML accelerator: 260 mm², 6 billion transistors, 600 GFLOPS GPU, 12 ARM 2.2 GHz CPUs.
- Two redundant chips for better safety.



Watch on YouTube

<https://www.youtube.com/watch?v=Ucp0TTmvqQE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN>

Fall 2021 Lectures & Schedule

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

- **Fall 2021 Edition:**
 - ❑ <https://safari.ethz.ch/architecture/fall2021/doku.php?id=schedule>
- **Fall 2020 Edition:**
 - ❑ <https://safari.ethz.ch/architecture/fall2020/doku.php?id=schedule>
- **Youtube Livestream (2021):**
 - ❑ https://www.youtube.com/watch?v=4yfkM_5EFgo&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF
- **Youtube Livestream (2020):**
 - ❑ <https://www.youtube.com/watch?v=c3mPdZA-Fmc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN>
- Master's level course
 - ❑ Taken by Bachelor's/Masters/PhD students
 - ❑ Cutting-edge research topics + fundamentals in Computer Architecture
 - ❑ 5 Simulator-based Lab Assignments
 - ❑ Potential research exploration
 - ❑ Many research readings

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

ML-Assisted Memory & Storage Management

Onur Mutlu

omutlu@gmail.com

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

21 May 2025

Stanford AI-Boosted Chip Design Lecture

SAFARI

ETH zürich