

Hermes & Sibyl:

ML-Driven Memory & Storage Management

Onur Mutlu

omutlu@gmail.com

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

27 September 2023

VMware

SAFARI

ETH zürich

Carnegie Mellon

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

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.



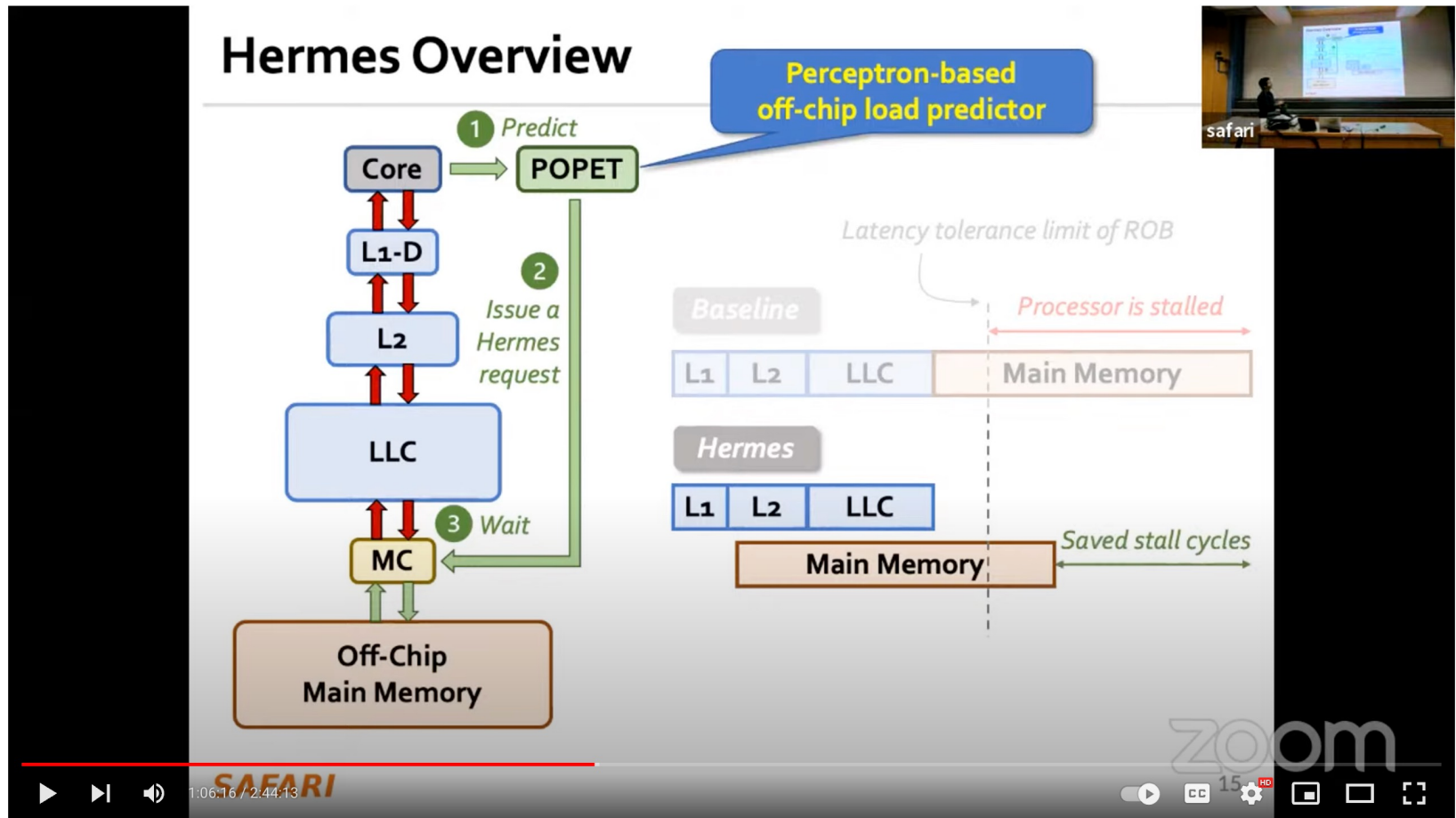
Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction

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>




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

The Key Problem

Long-latency **off-chip** load requests



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



Limit performance

Traditional Solutions



1

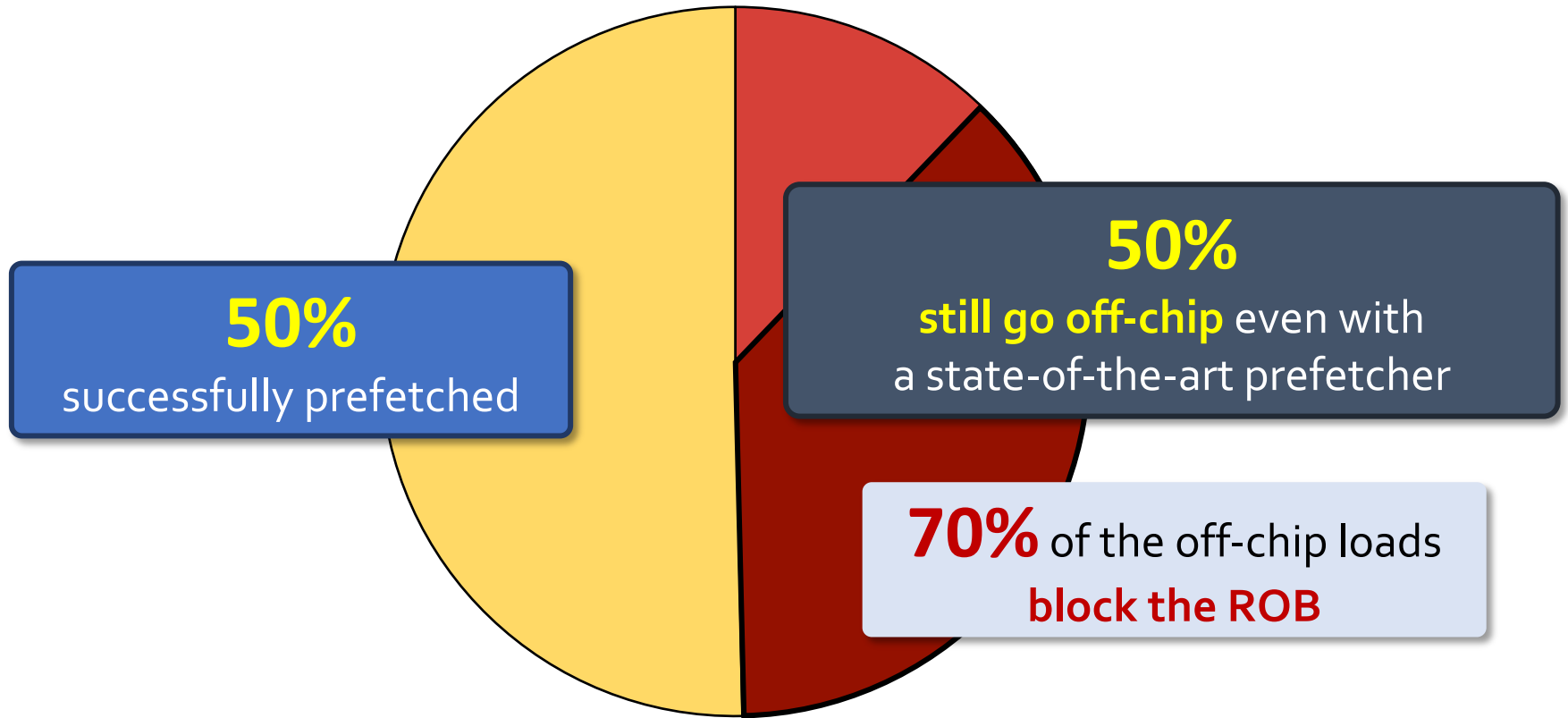
Employ sophisticated **prefetchers**

2

Increase size of on-chip caches

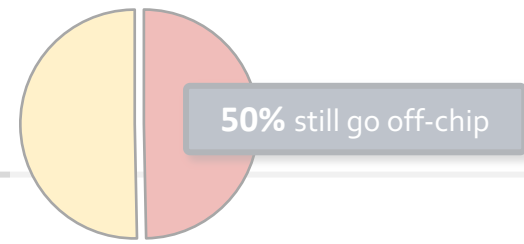
Key Observation 1

Many loads still go off-chip

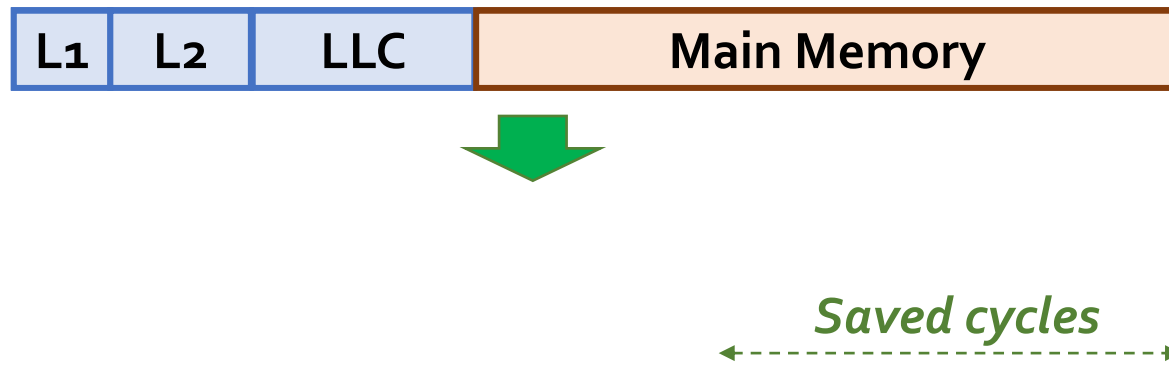


off-chip loads without any prefetcher

Key Observation 2

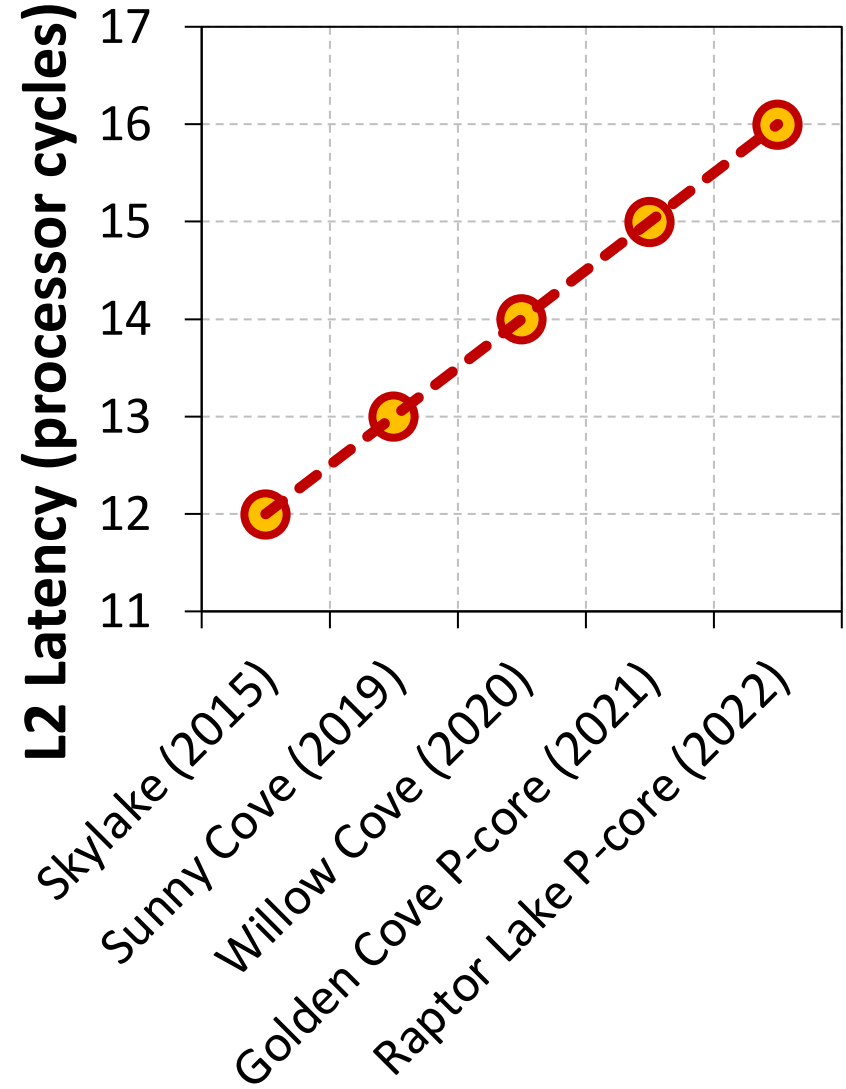
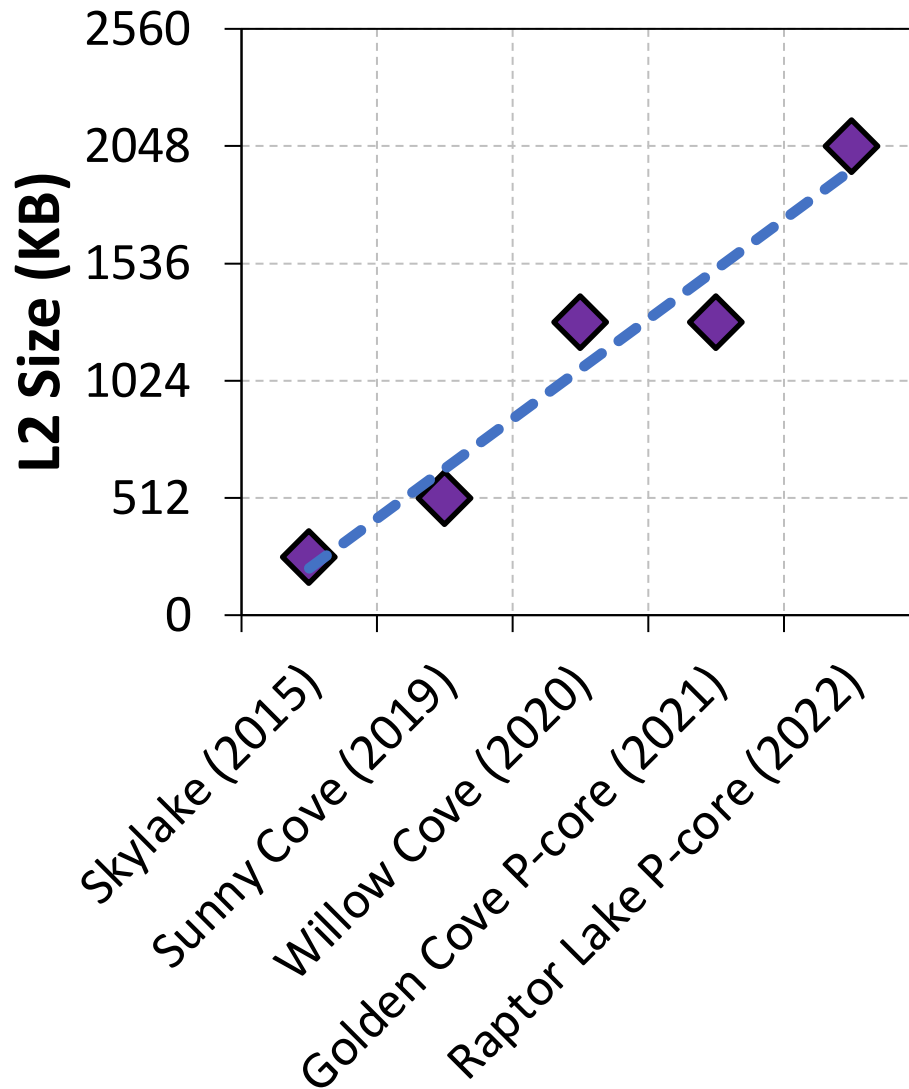


On-chip cache access latency
significantly contributes to off-chip load latency



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

Key Contribution



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

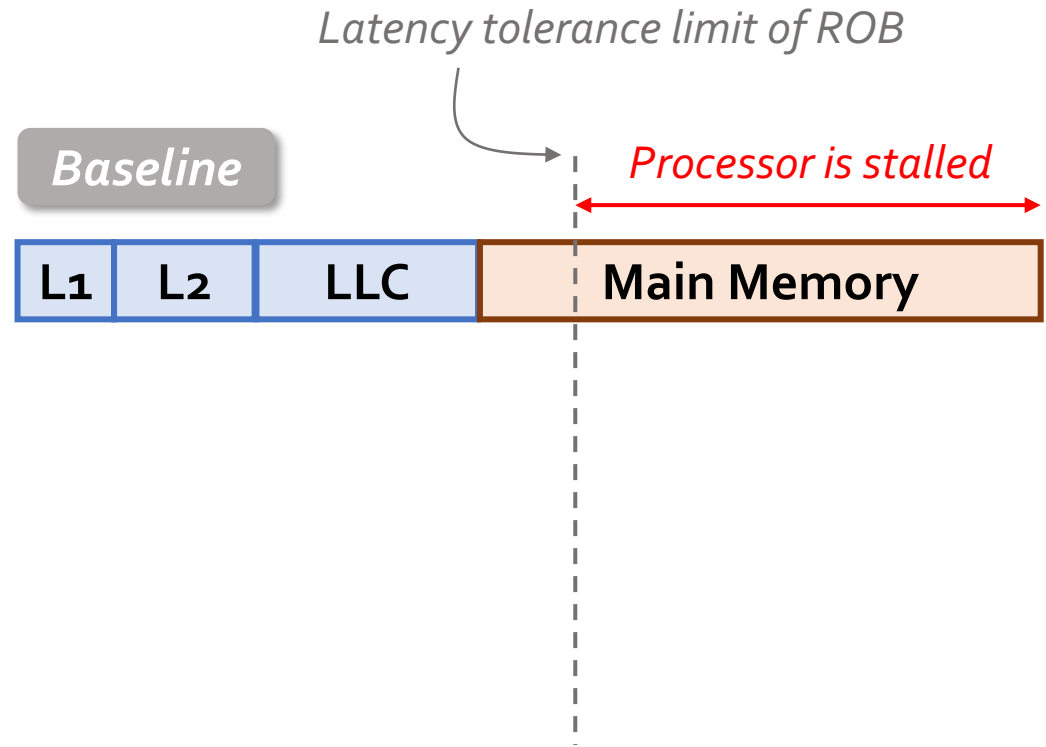
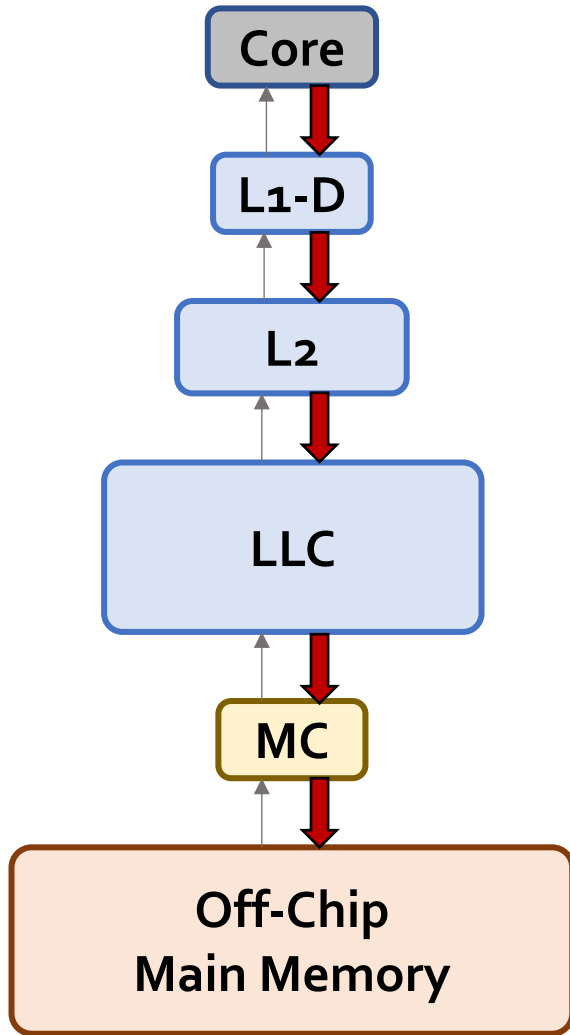


That predicts which loads are likely to **go off-chip**

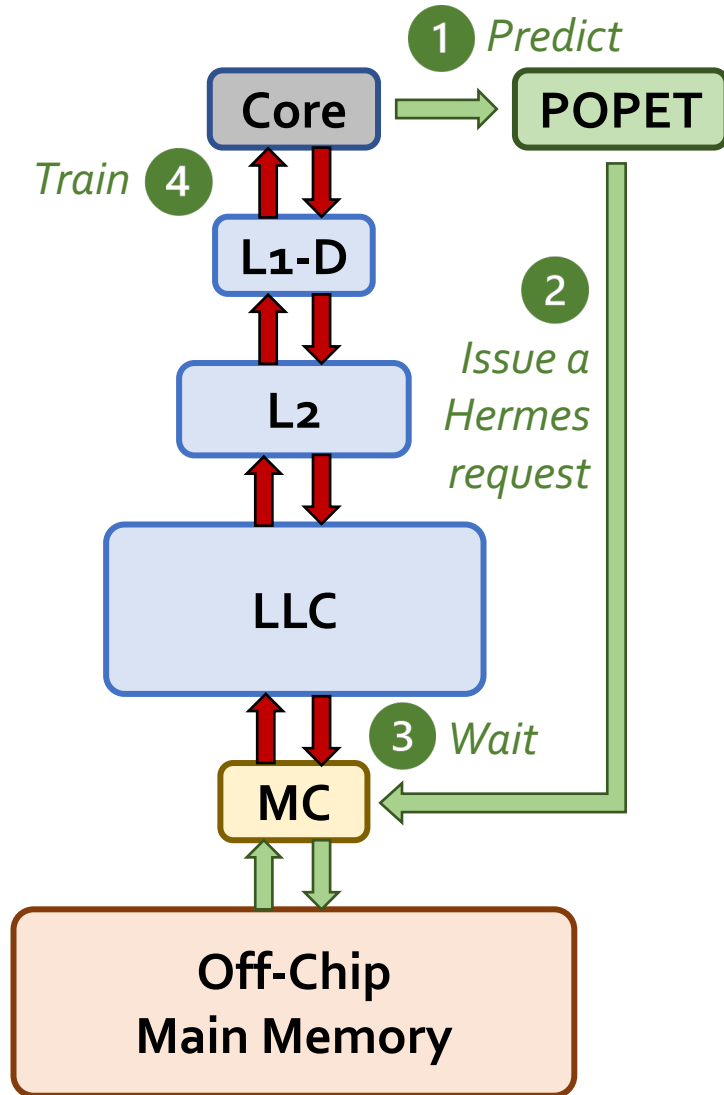


By **learning** from
multiple program context information

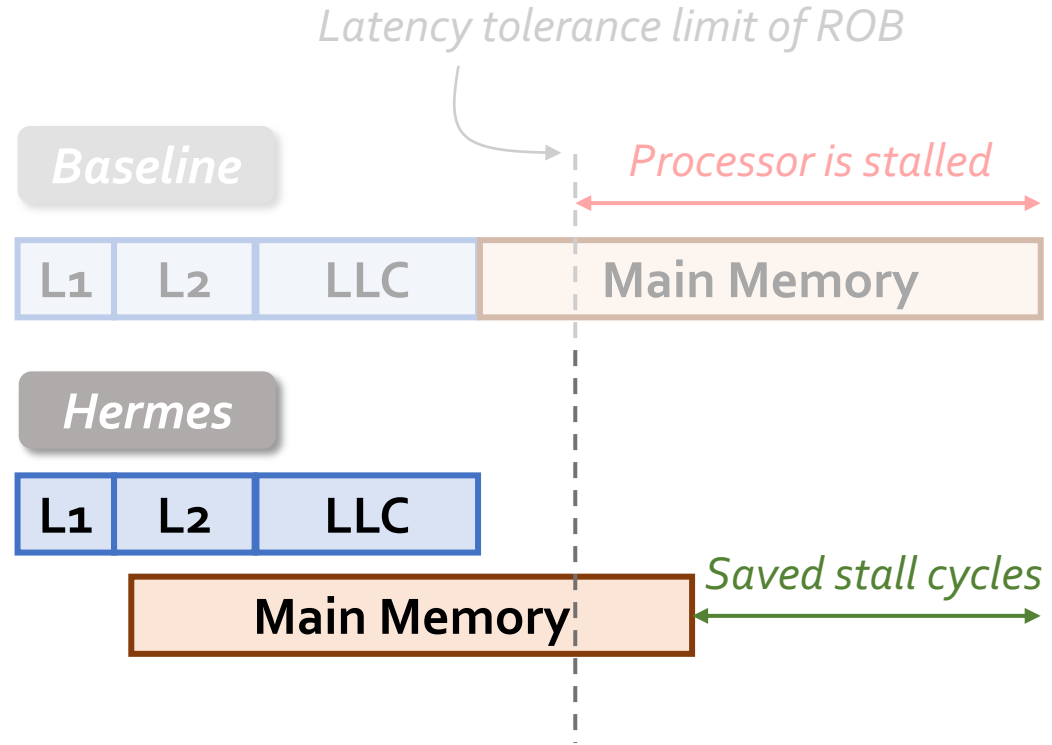
Hermes Overview



Hermes Overview



Perceptron-based
off-chip load predictor



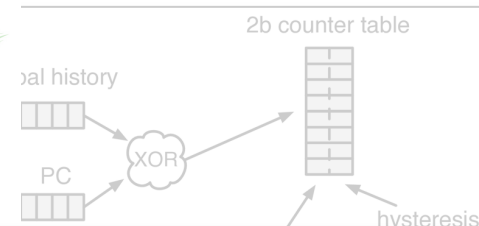
Designing the Off-Chip Load Predictor

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

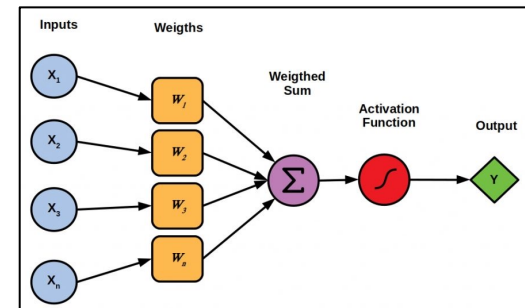
Correlate different program features with off-chip loads



Low storage overhead

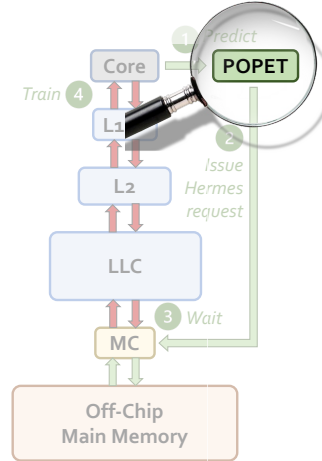
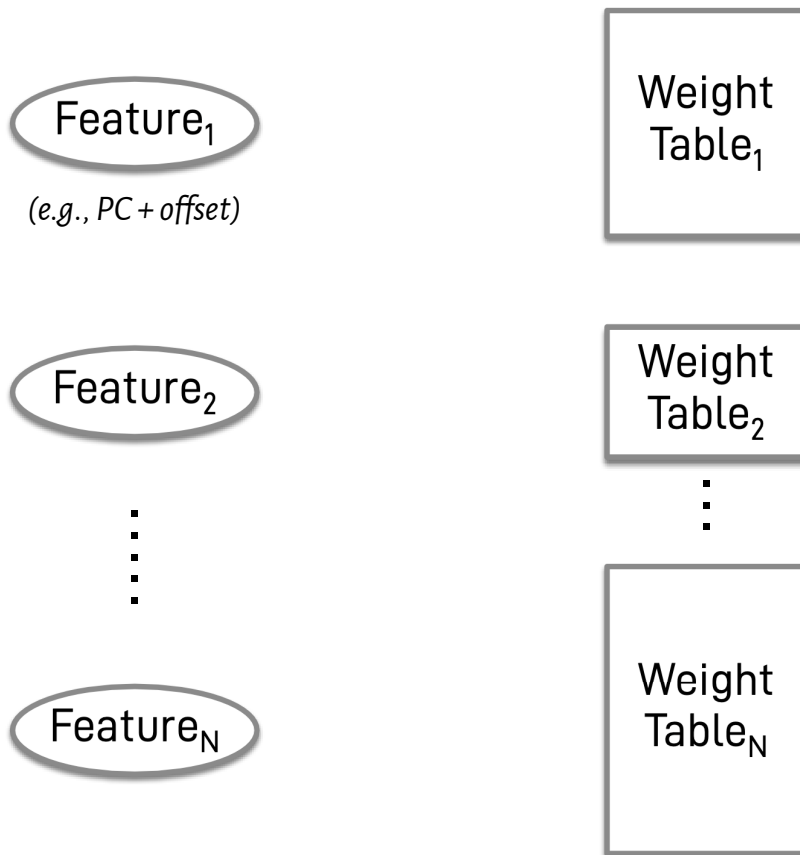


Low design complexity



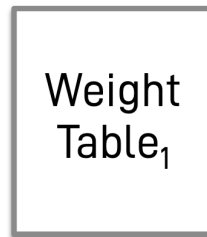
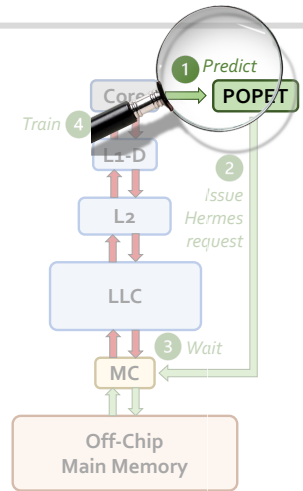
POPET: Perceptron-Based Off-Chip Predictor

- **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**

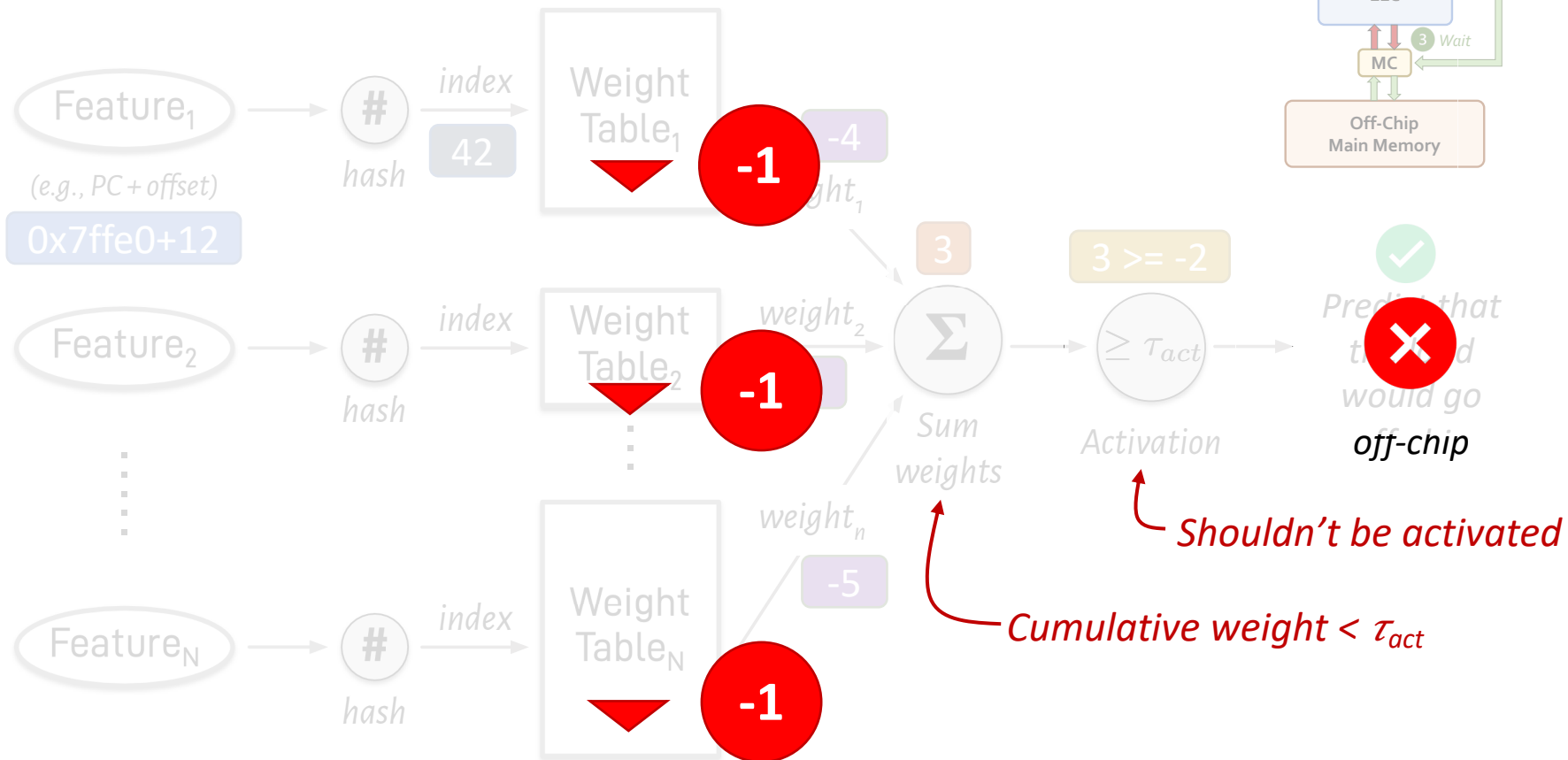
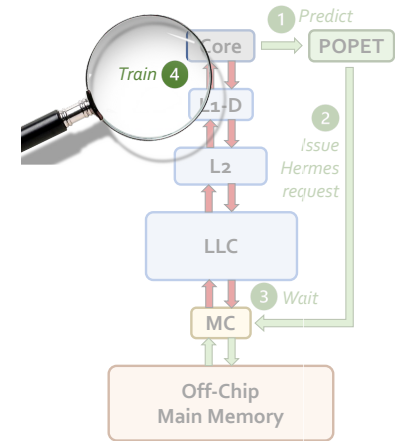


⋮



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

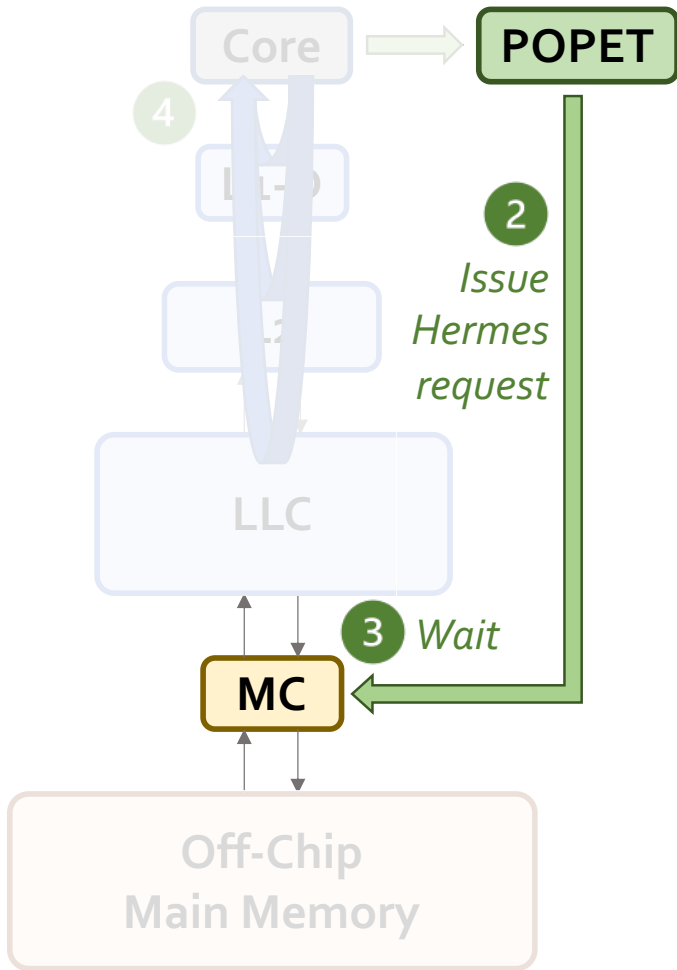
LLC Prefetchers

- 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**

Latency Configuration



- **Cache round-trip latency**

- L1-D: **5** cycles
- L2: **15** cycles
- LLC: **55** cycles

- **Hermes request issue latency**

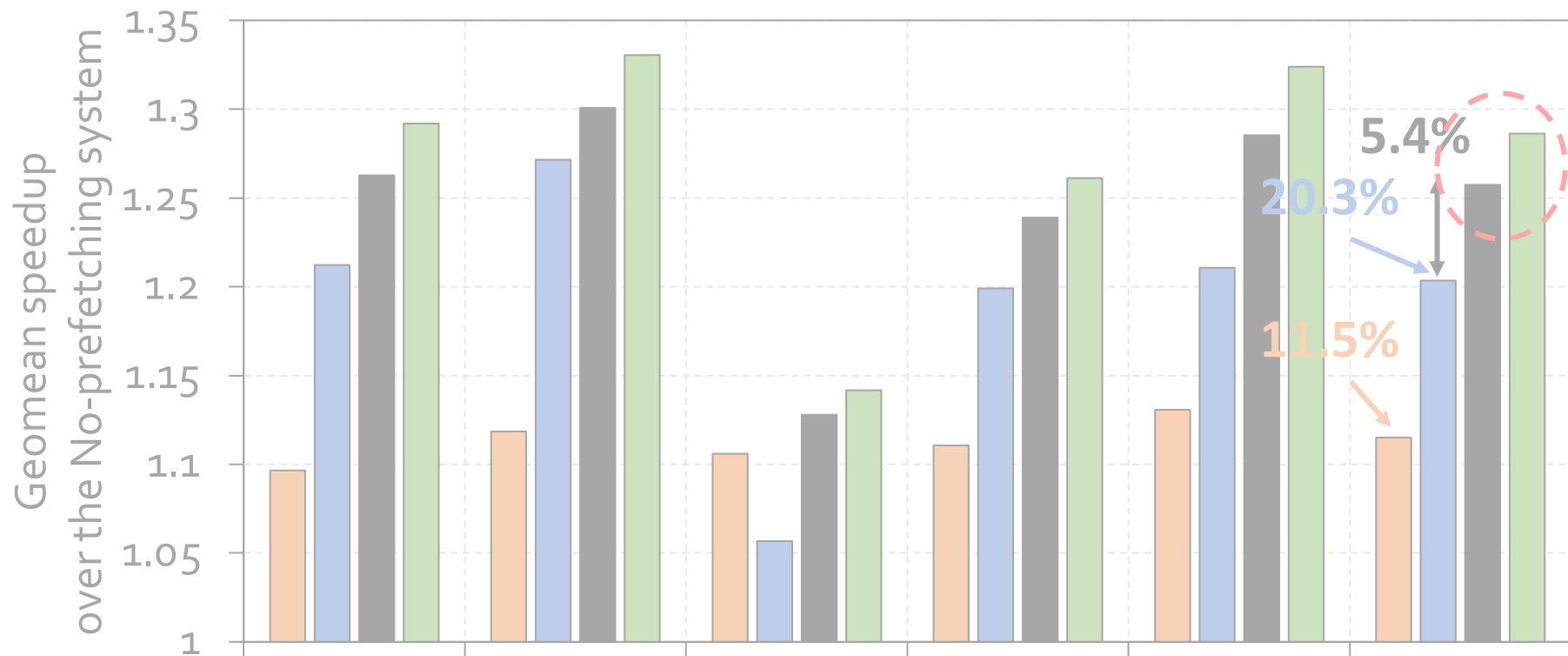
(incurred after address translation)

Depends on

- **Interconnect** between POPET and MC



Single-Core Performance Improvement

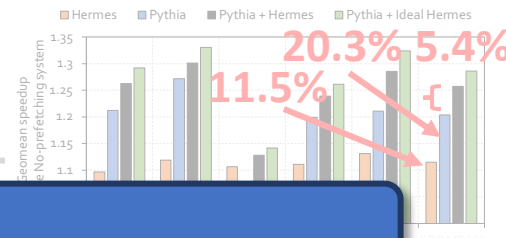


Hermes alone provides nearly

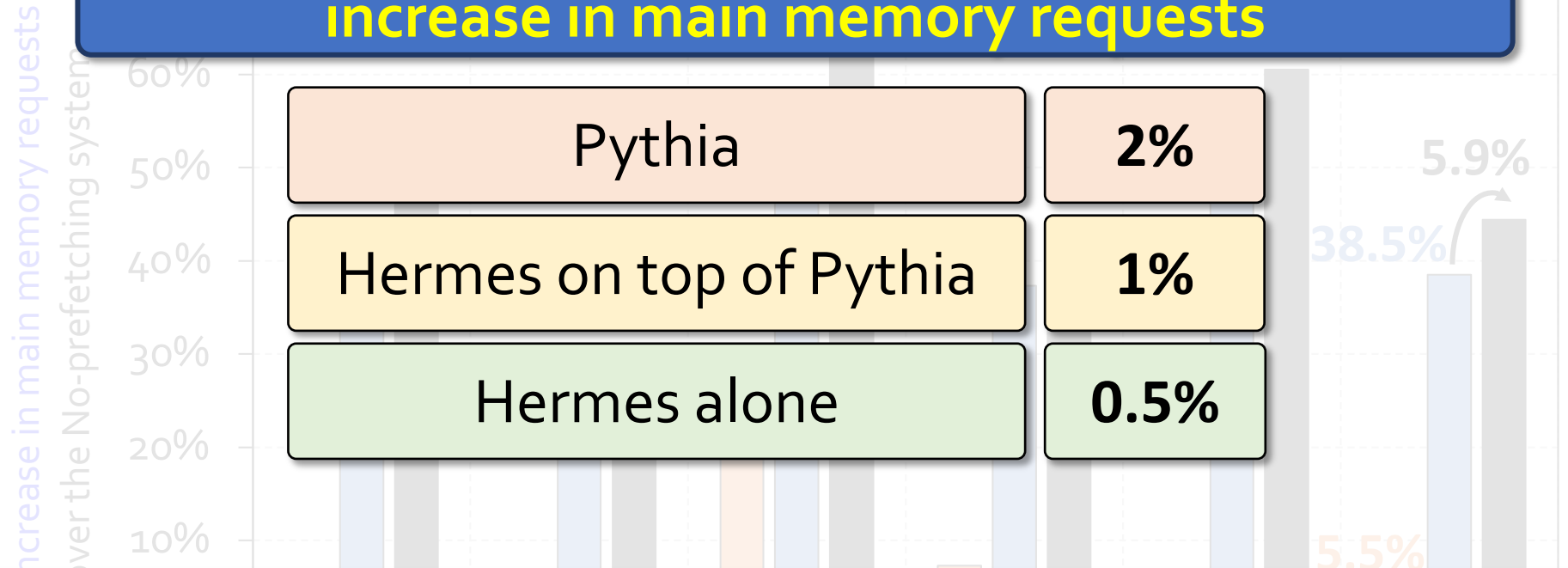
Hermes provides nearly **90%** performance benefit of **Ideal Hermes** that has an **ideal off-chip load predictor**

with only **2/3** storage overhead

Increase in Main Memory Requests

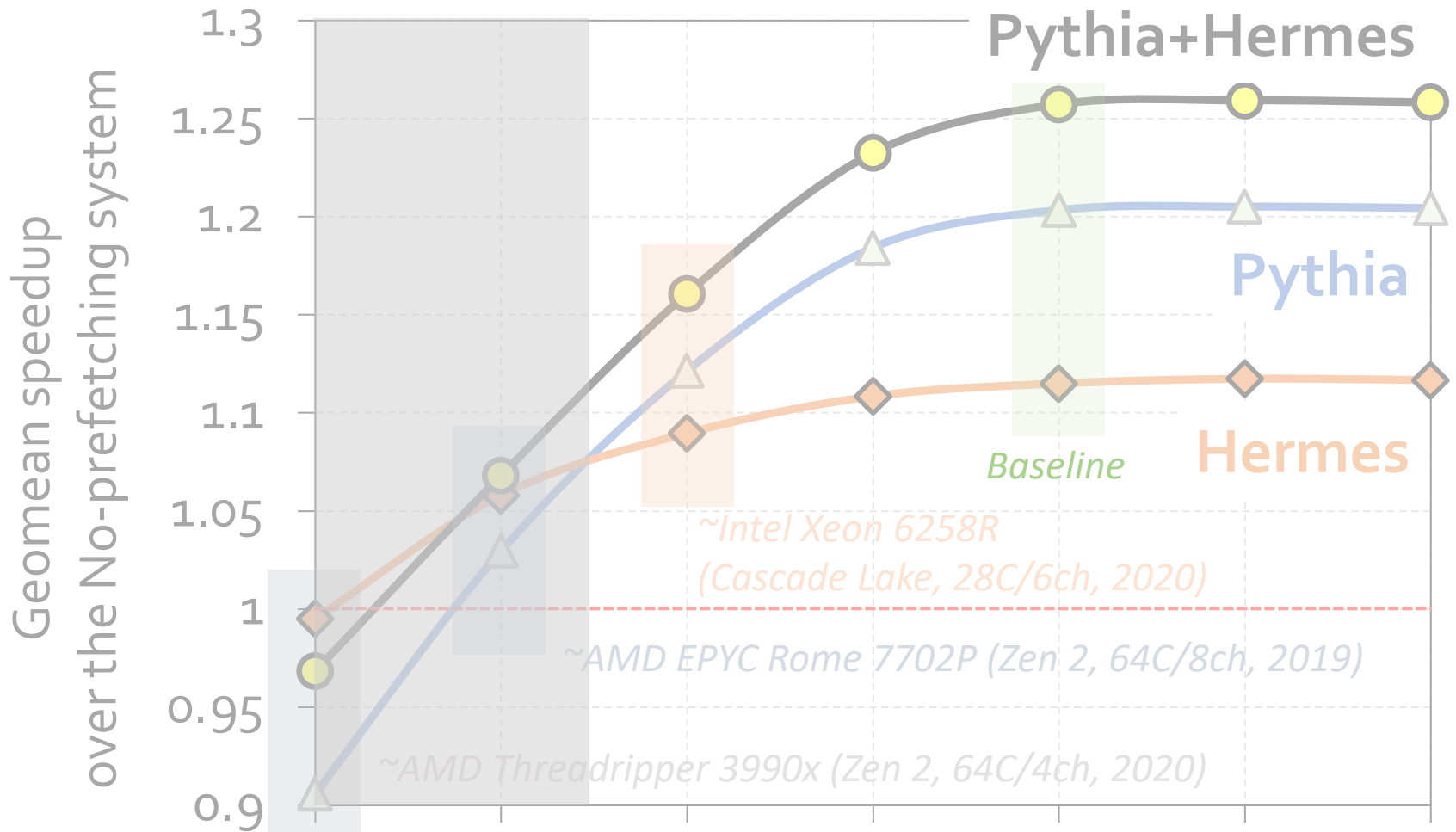


For **every 1% performance** benefit,
increase in main memory requests



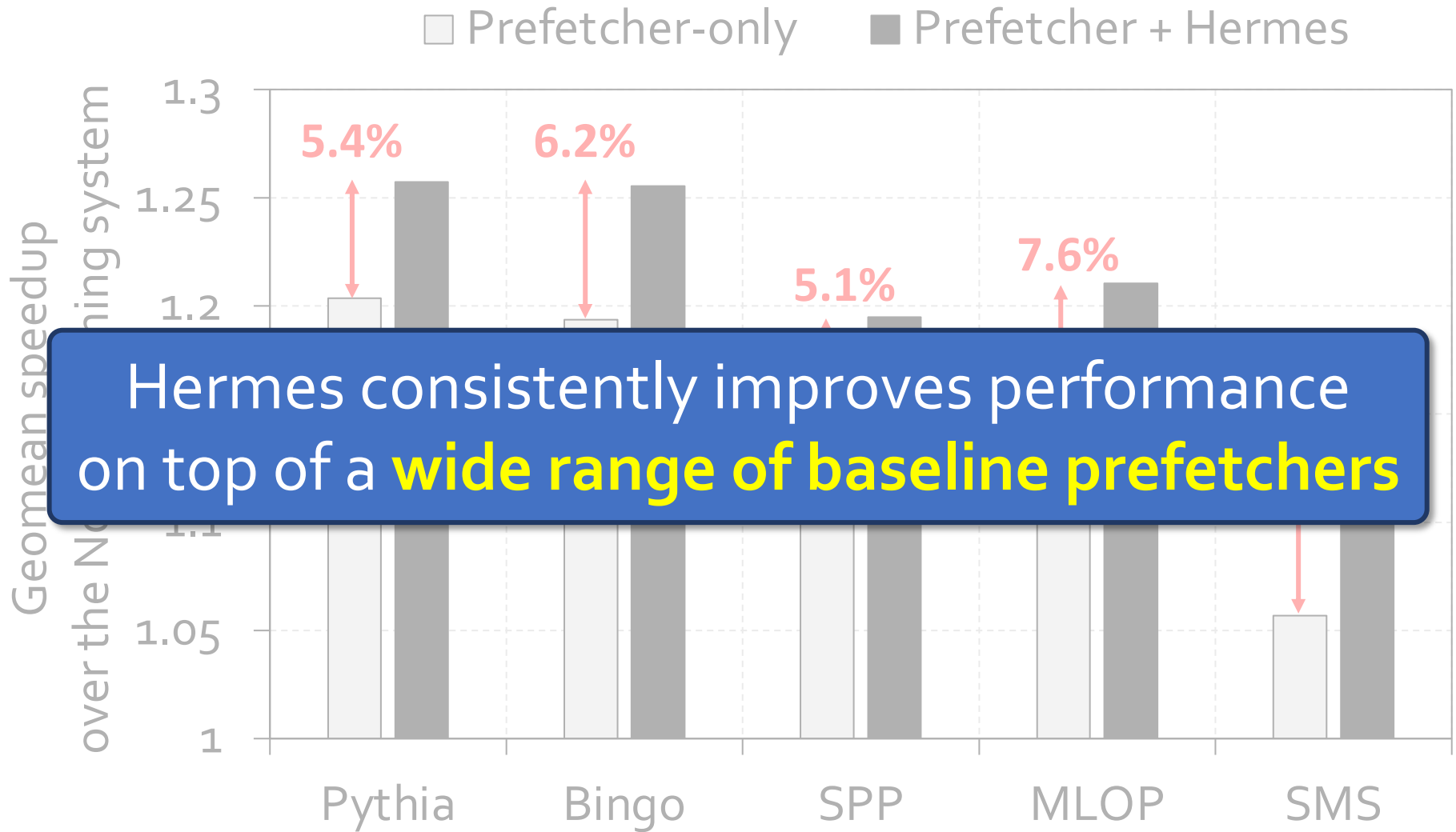
Hermes is more **bandwidth-efficient**
than even an efficient prefetcher like Pythia

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*

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

More in the 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>

To Summarize...

Summary

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

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 Sourced



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...



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>



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

Hermes Discussion

- **FAQs**

- [What are the selected set of program features?](#)
- [Can you provide some intuition on why these features work?](#)
- [What happens in case of a misprediction?](#)
- [What's the performance headroom for off-chip prediction?](#)
- [Do you see a variance of different features in final prediction accuracy?](#)

- **Simulation Methodology**

- [System parameters](#)
- [Evaluated workloads](#)

- **More Results**

- [Percentage of off-chip requests](#)
- [Reduction in stall cycles by reducing the critical path](#)
- [Fraction of off-chip load requests](#)
- [Accuracy and coverage of POPET](#)
- [Effect of different features](#)
- [Are all features required?](#)
- [1C performance](#)
- [1C performance line graph](#)
- [1C performance against prior predictors](#)
- [Effect on stall cycles](#)
- [8C performance](#)
- Sensitivity:
 - [Hermes request issue latency](#)
 - [Cache hierarchy access latency](#)
 - [Activation threshold](#)
 - [ROB size](#)
 - [LLC size](#)
- [Power overhead](#)
- [Accuracy without prefetcher](#)
- [Main memory request overhead with different prefetchers](#)

Hermes Paper [MICRO 2022]

- 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>

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**

Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

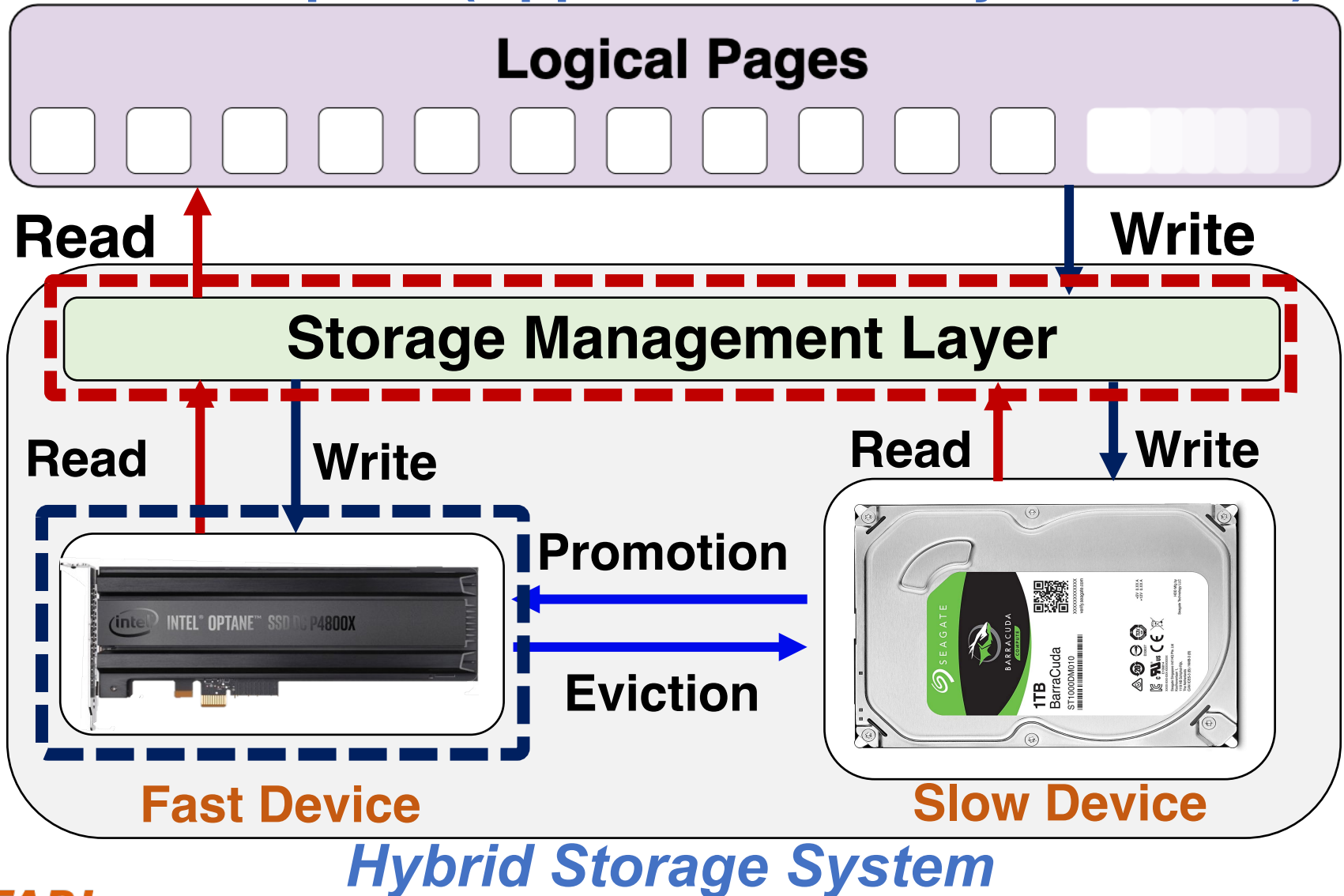
Sibyl: Overview

Evaluation of Sibyl and Key Results

Conclusion

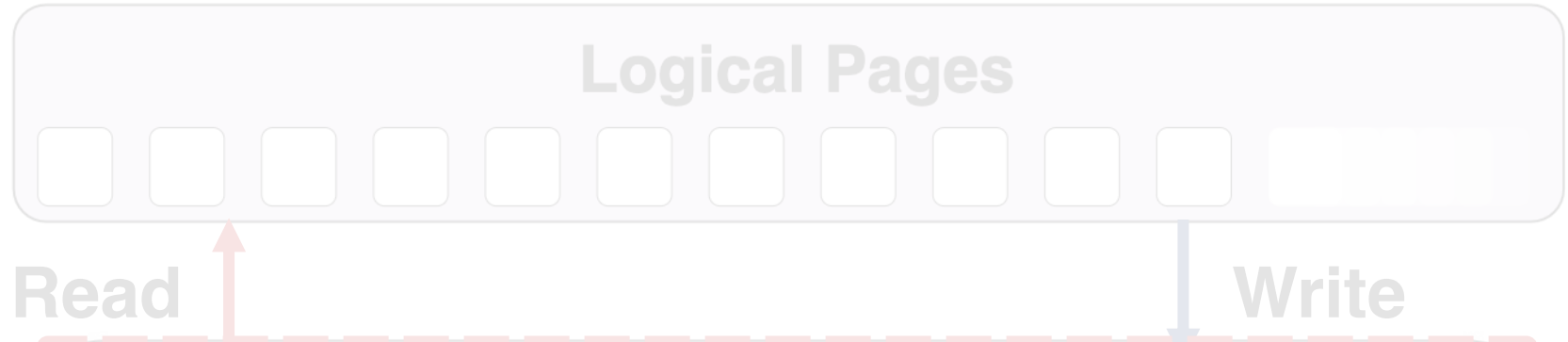
Hybrid Storage System Basics

Address Space (Application/File System View)



Hybrid Storage System Basics

Logical Address Space (Application/File System View)



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



Key Shortcomings in Prior Techniques

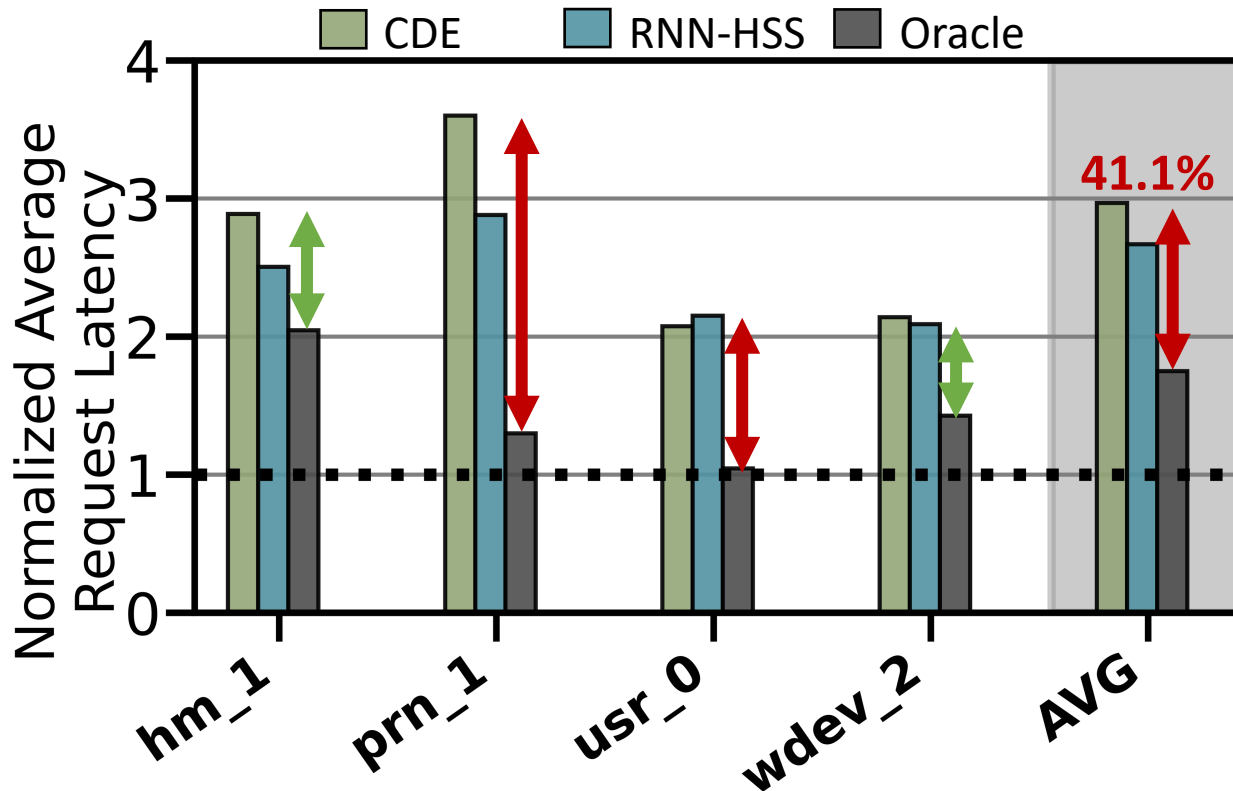
We observe **two key shortcomings** that significantly limit the performance benefits of prior techniques

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

Lack of Adaptivity (1/2)

Workload Changes

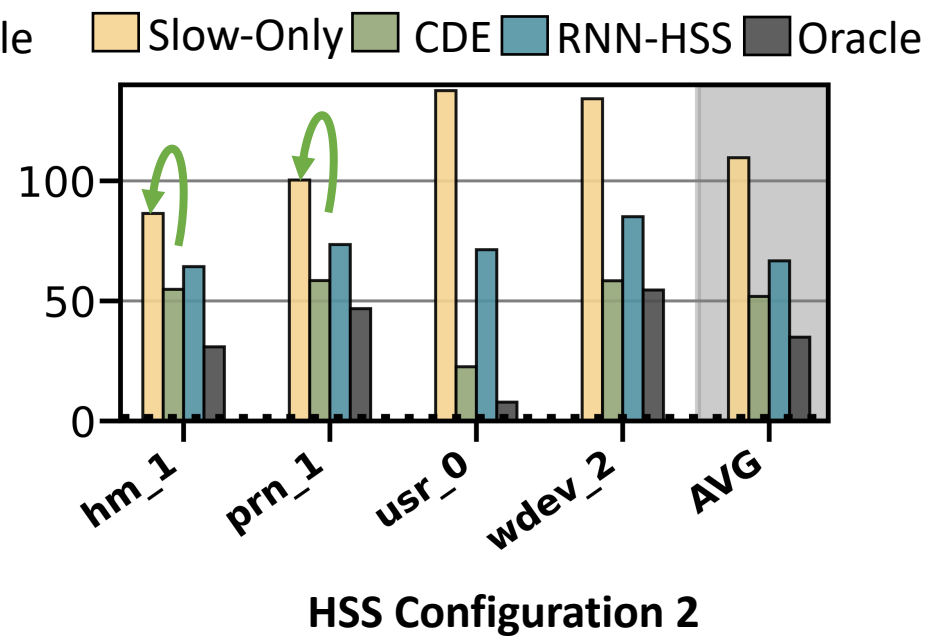
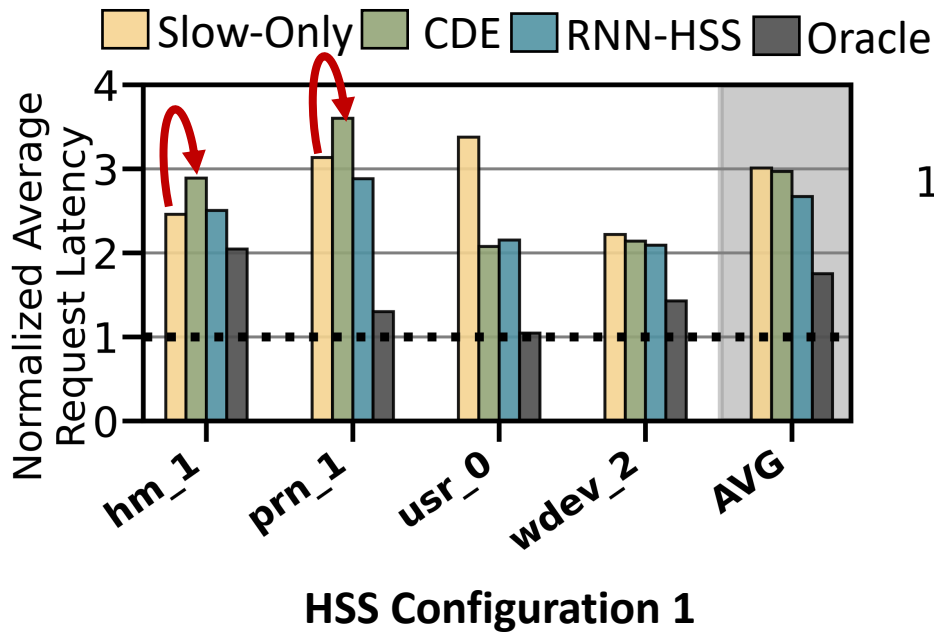
Prior data placement techniques consider only a **few workload characteristics** that are **statically tuned**



Lack of Adaptivity (2/2)

Changes in Device Types and Configurations

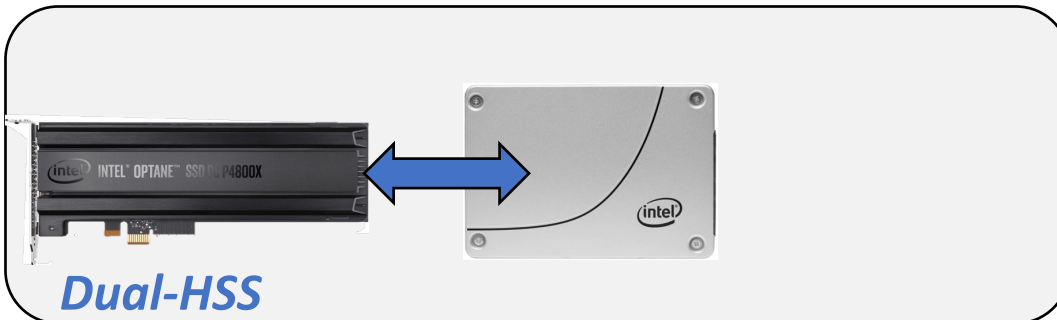
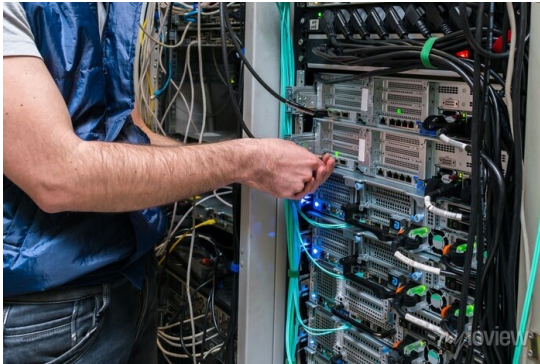
Do not consider **underlying storage device characteristics** (e.g., changes in the level asymmetry in read/write latencies, garbage collection)



Lack of Extensibility (1/2)

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

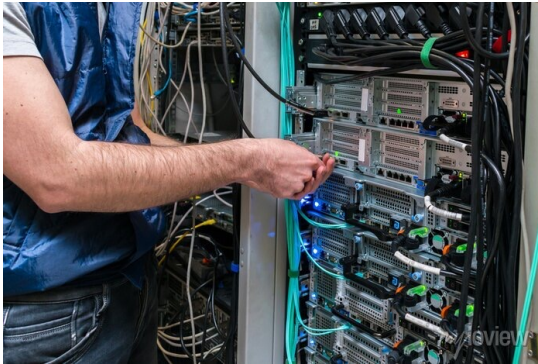
Change in storage configuration



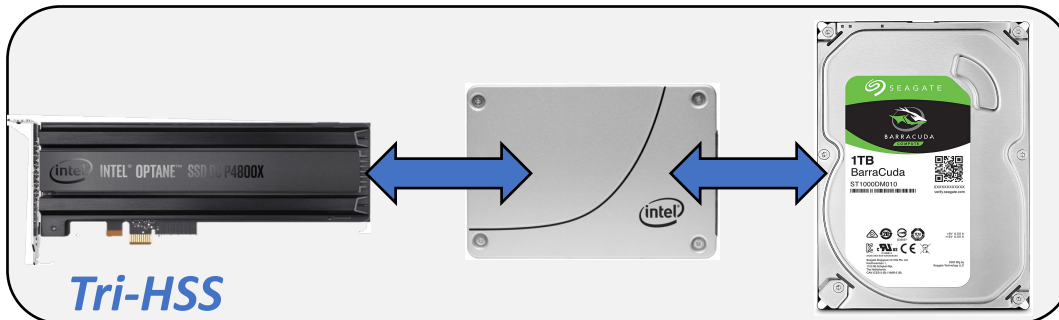
Lack of Extensibility (2/2)

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>

Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

Sibyl: Overview

Evaluation of Sibyl and Key Results

Conclusion

Basics of Reinforcement Learning (RL)



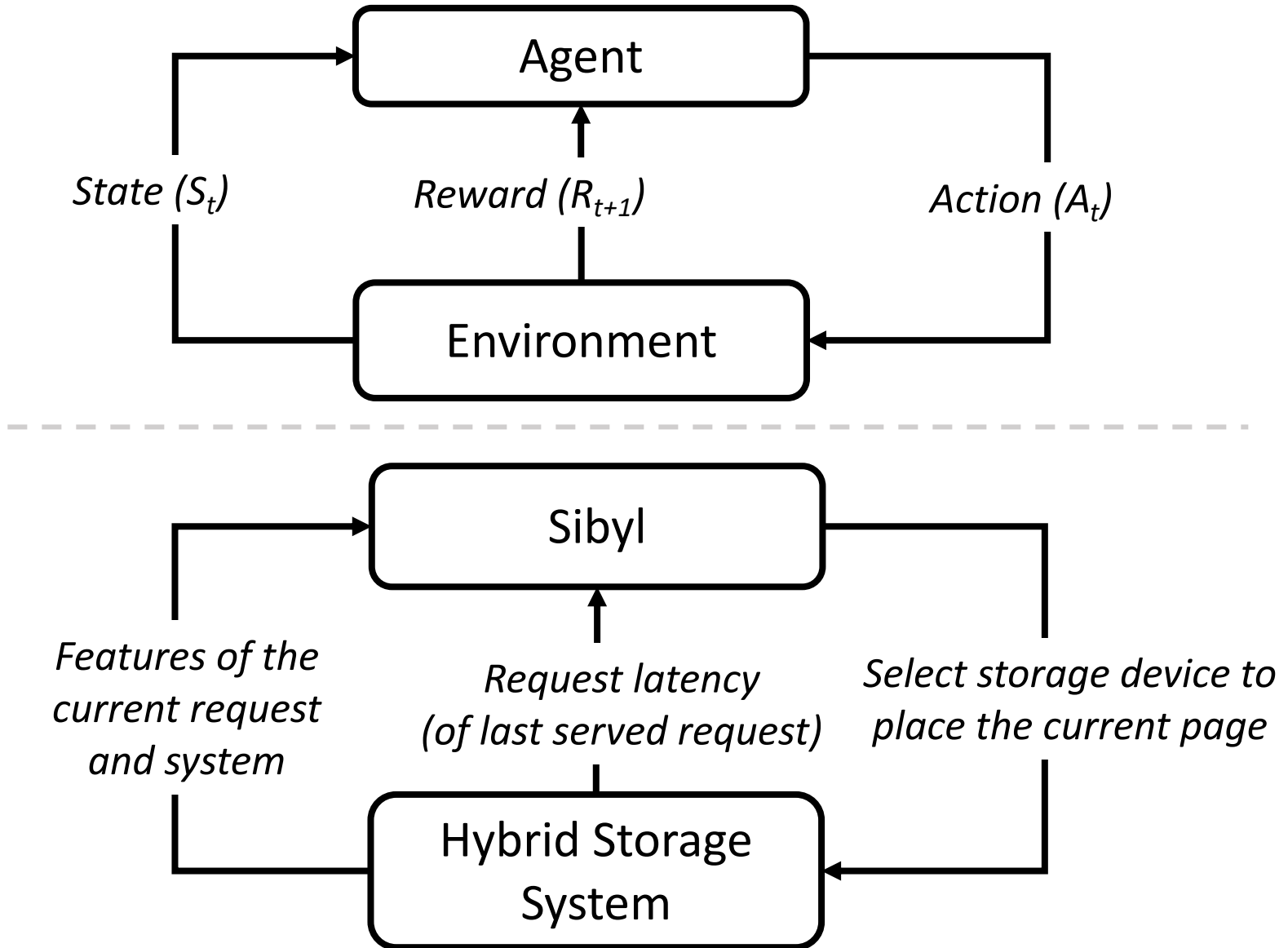
The diagram illustrates the basic components of Reinforcement Learning. It consists of two rounded rectangular boxes. The top box is labeled 'Agent' and the bottom box is labeled 'Environment'. The boxes are vertically aligned and centered on the slide.

Agent

Environment

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

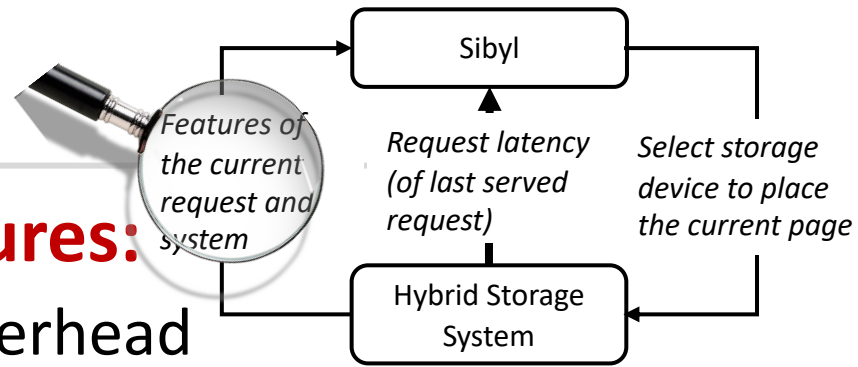
Formulating Data Placement as RL



What is State?

- **Limited number of state features:**

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



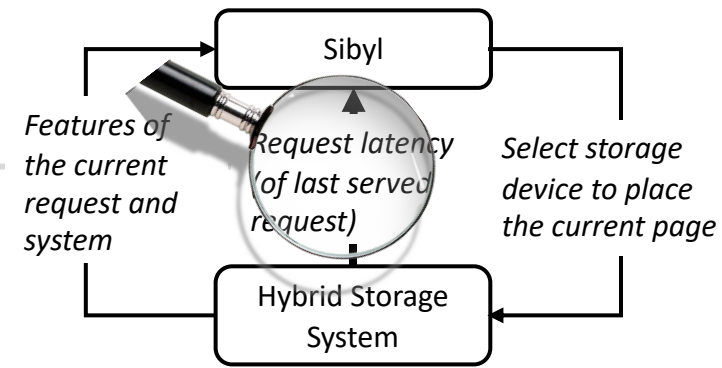
- **6-dimensional** vector of state features

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

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

What is Reward?

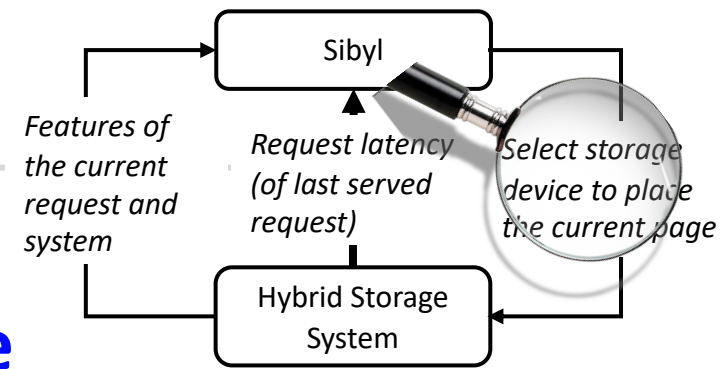
- Defines the **objective** of Sibyl



- 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

What is Action?

- At every new page request, the action is to **select a storage device**
- Action can be **easily extended** to any number of storage devices
- Sibyl learns to **proactively evict or promote** a page



Talk Outline

Key Shortcomings of Prior Data Placement Techniques

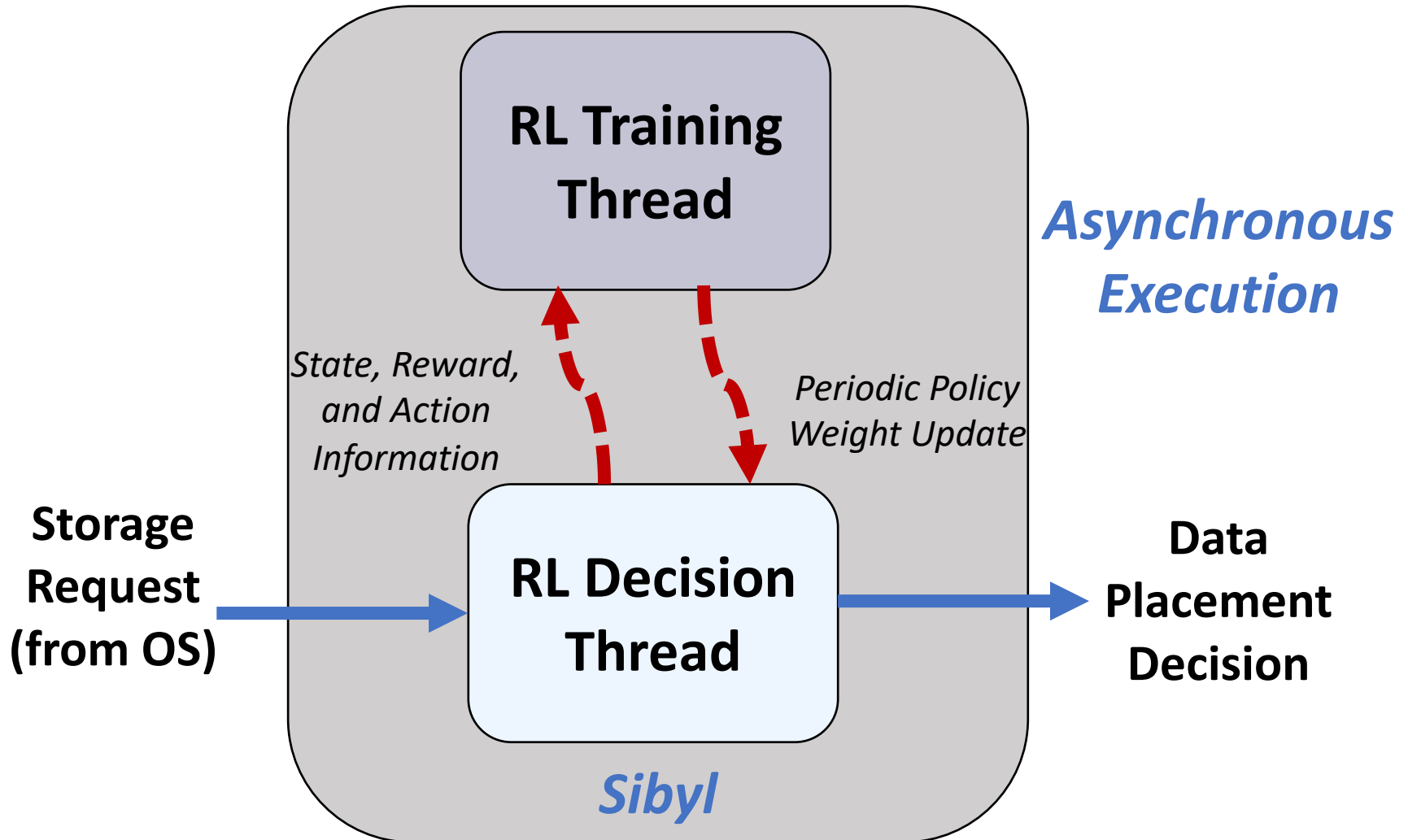
Formulating Data Placement as Reinforcement Learning

Sibyl: Overview

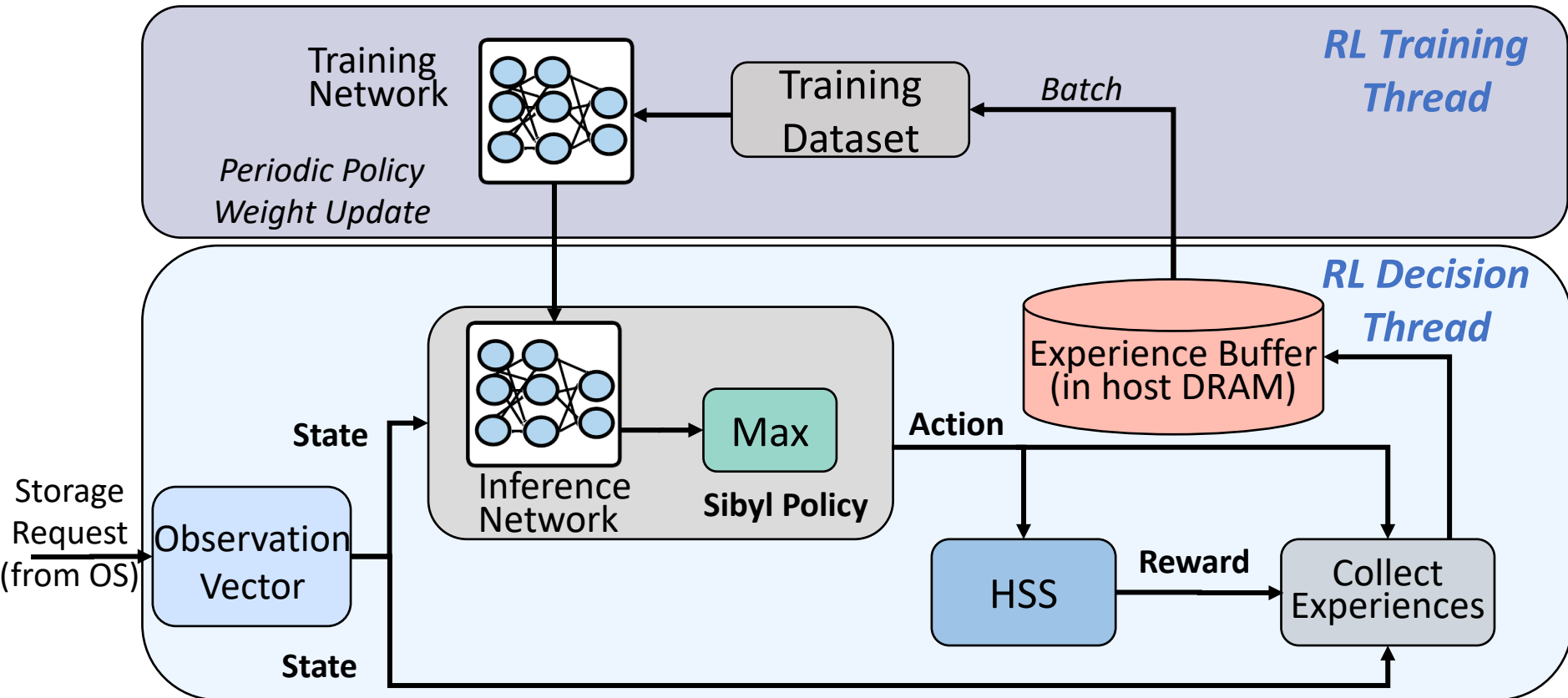
Evaluation of Sibyl and Key Results

Conclusion

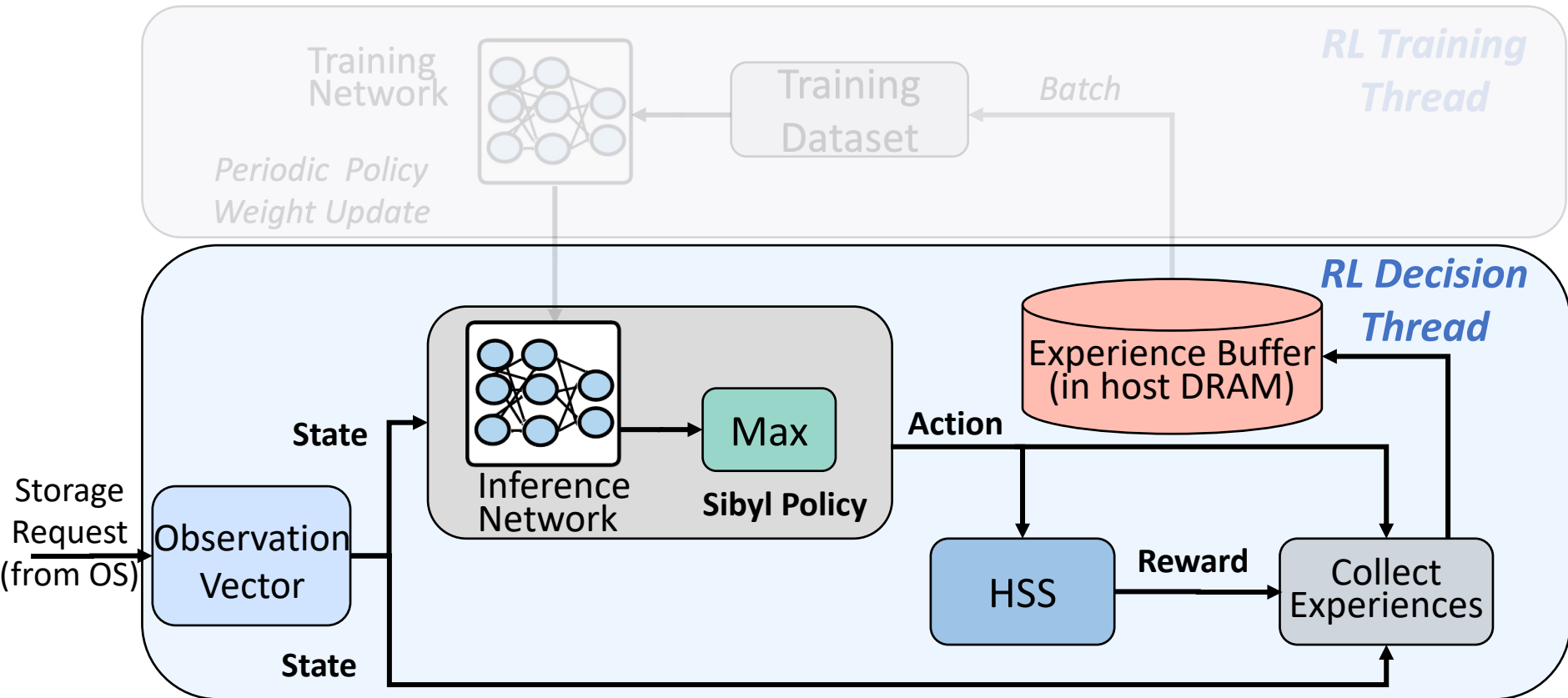
Sibyl Execution



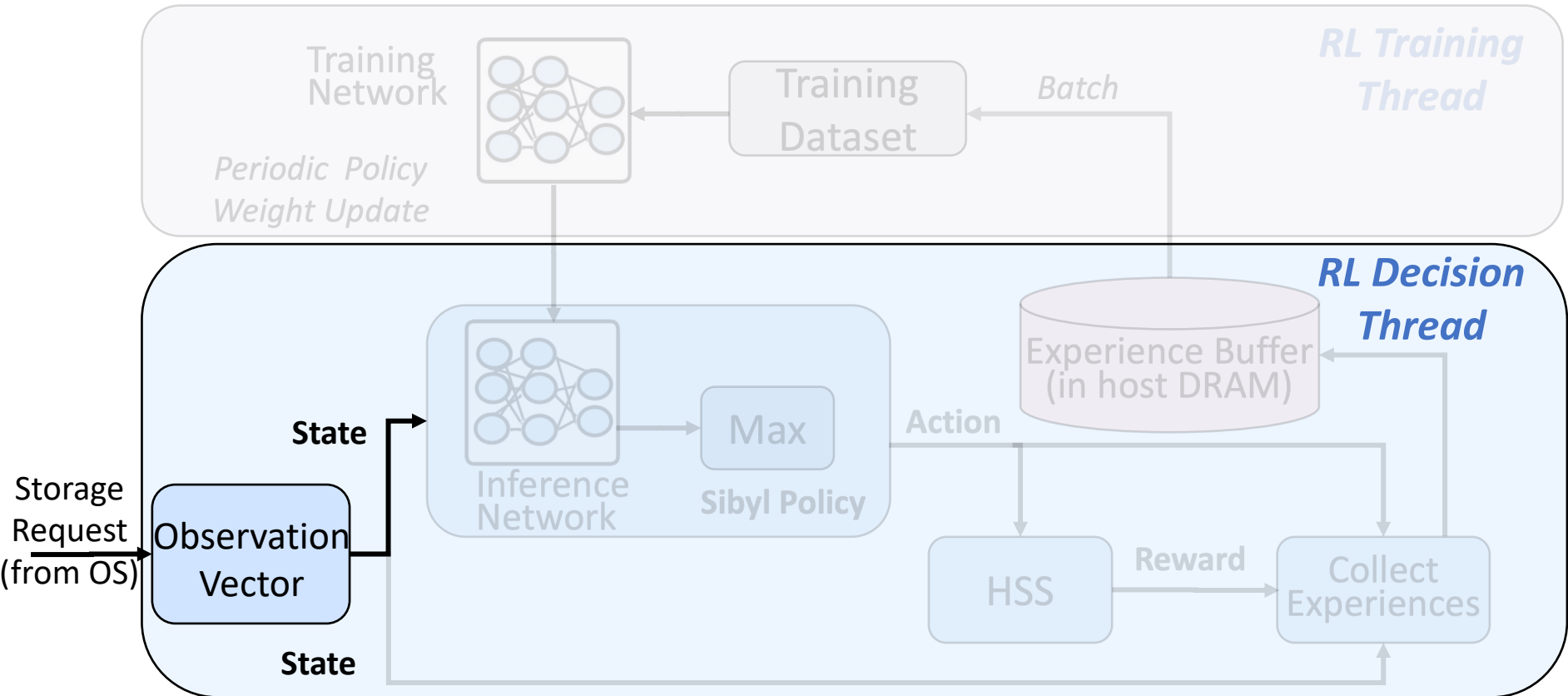
Sibyl Design: Overview



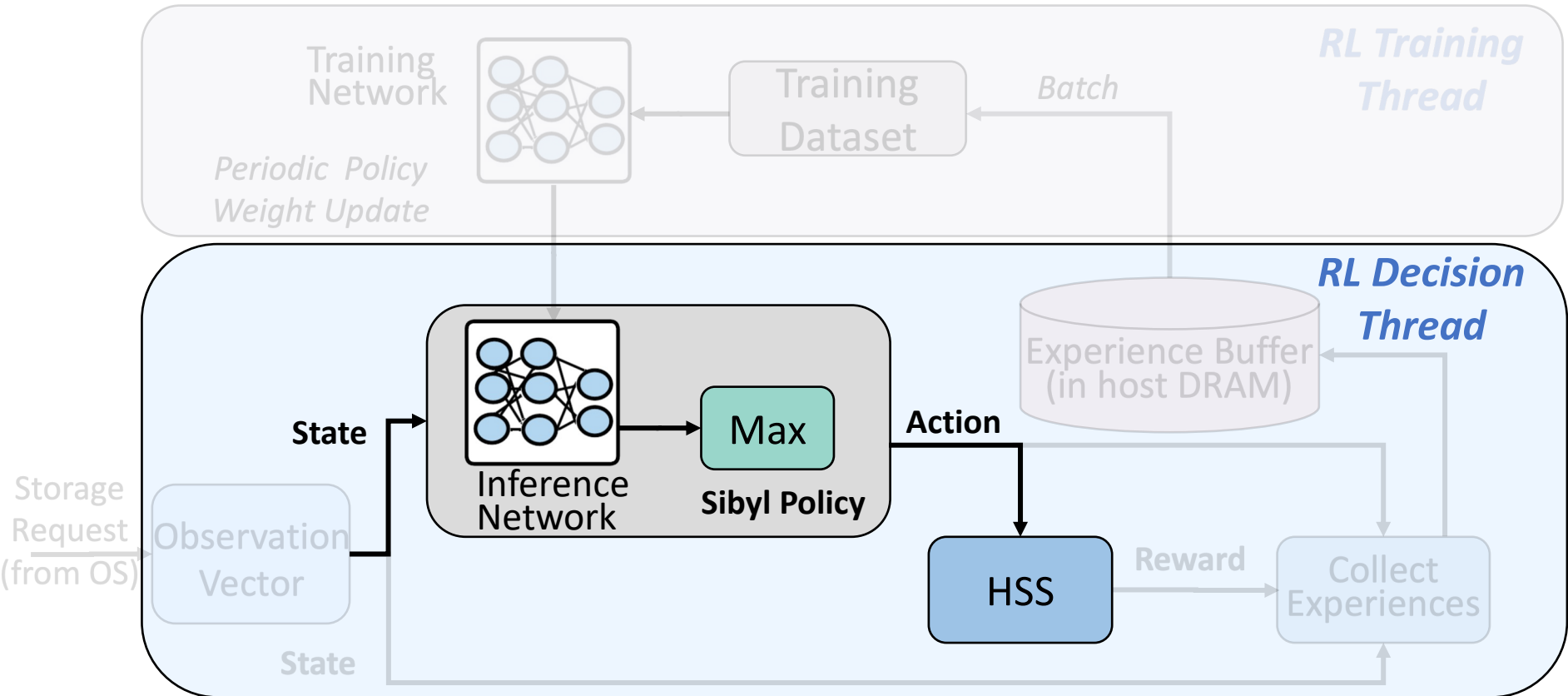
RL Decision Thread



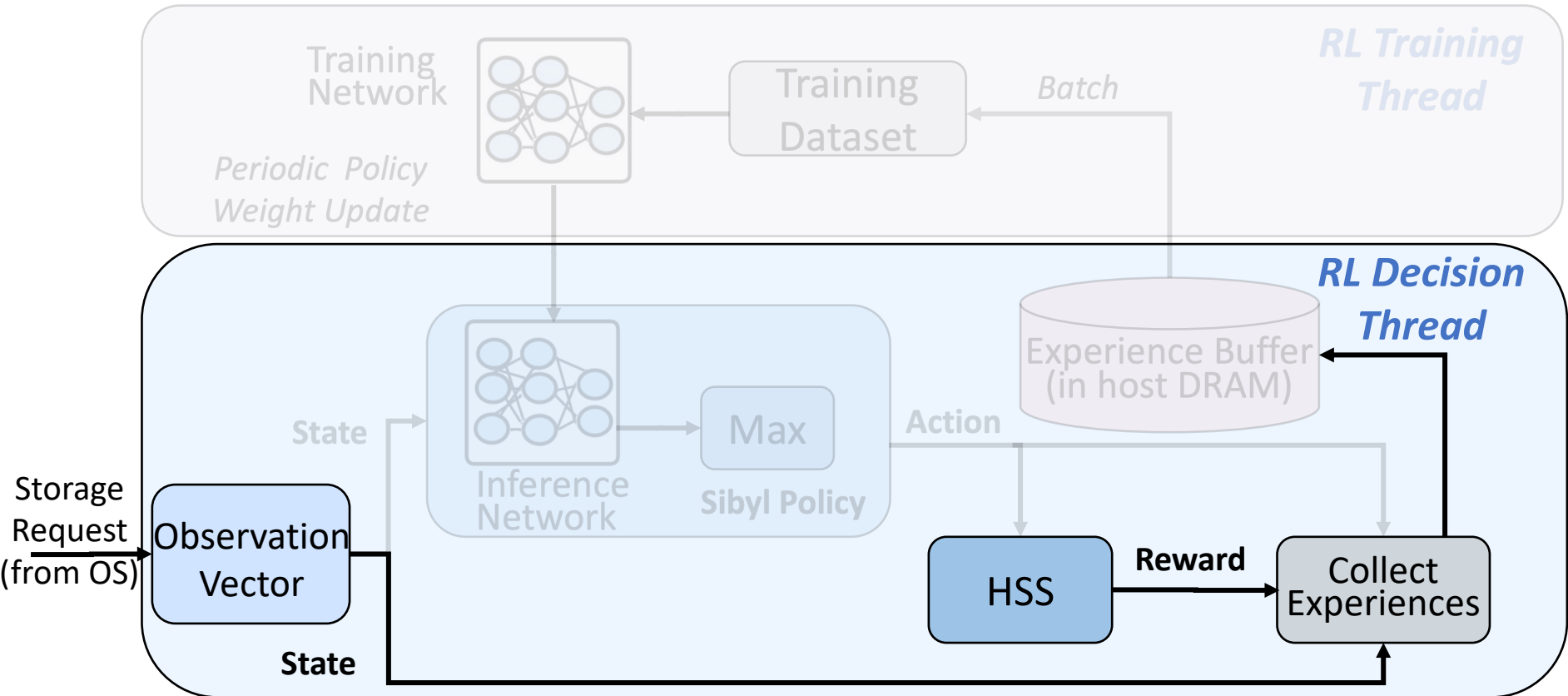
RL Decision Thread



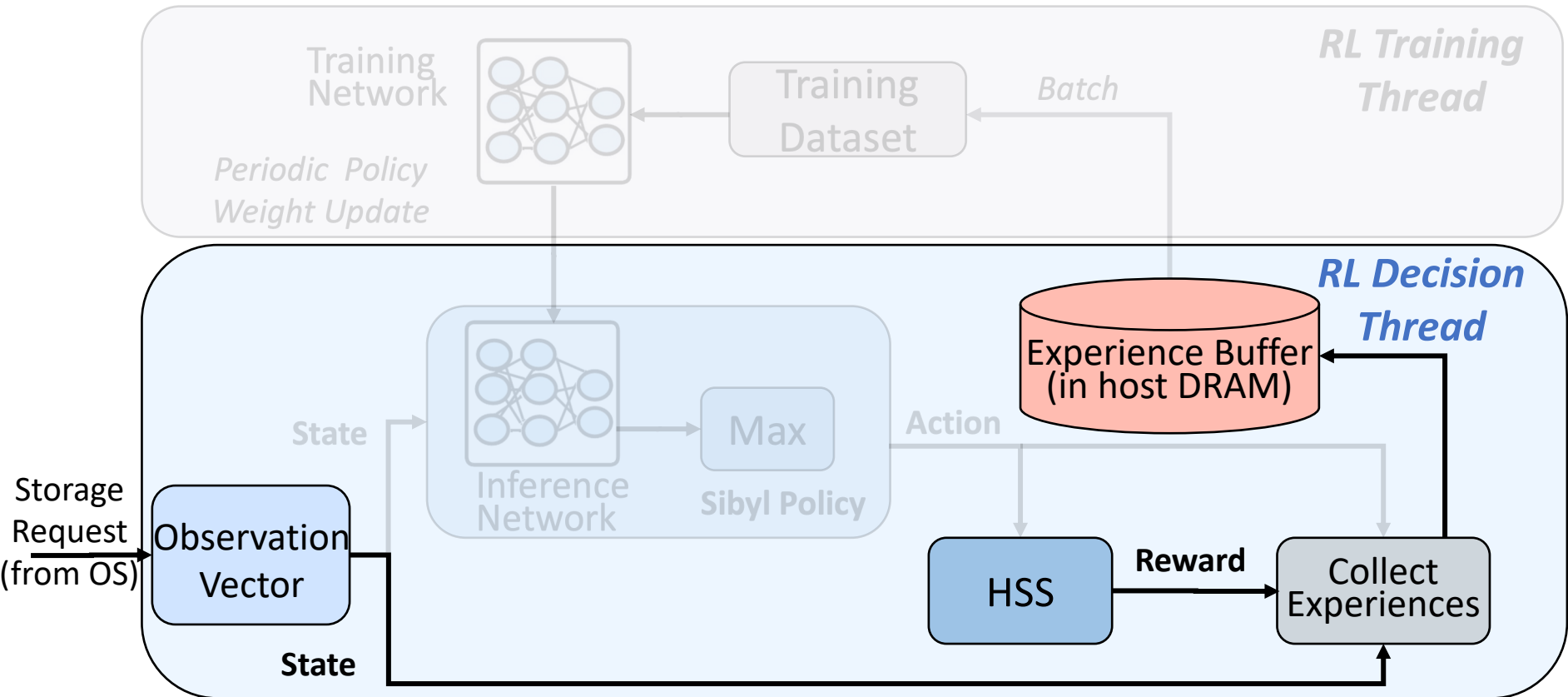
RL Decision Thread



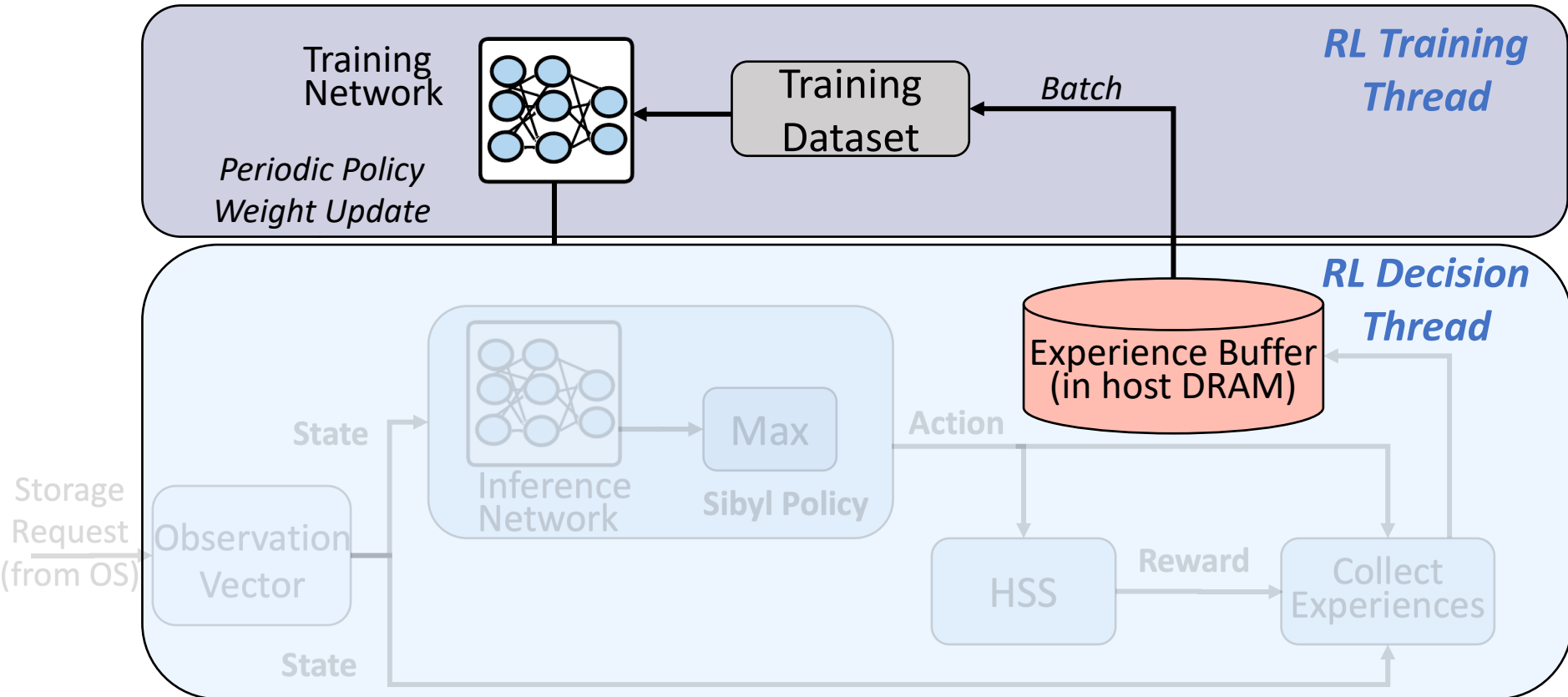
RL Decision Thread



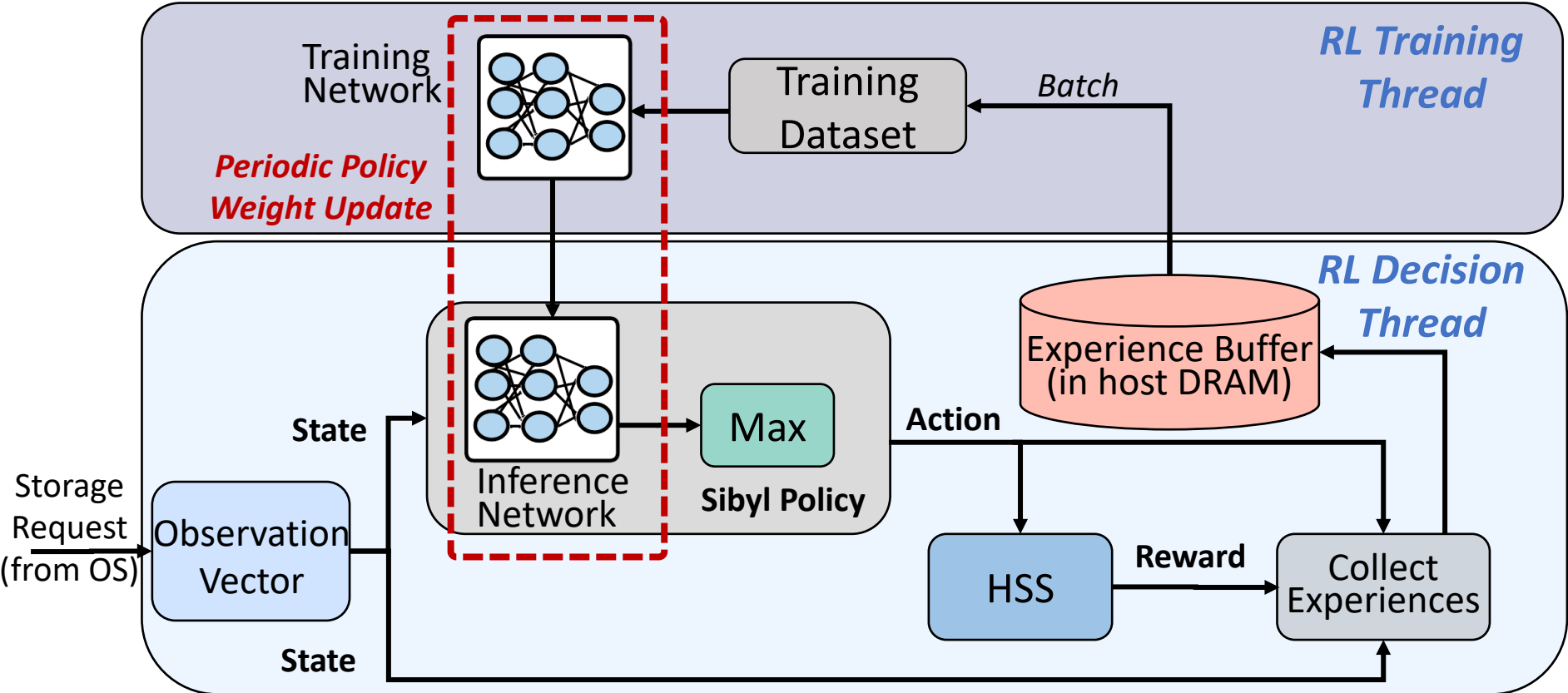
RL Decision Thread



RL Training Thread



Periodic Weight Transfer



Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

Sibyl: Overview

Evaluation of Sibyl and Key Results

Conclusion

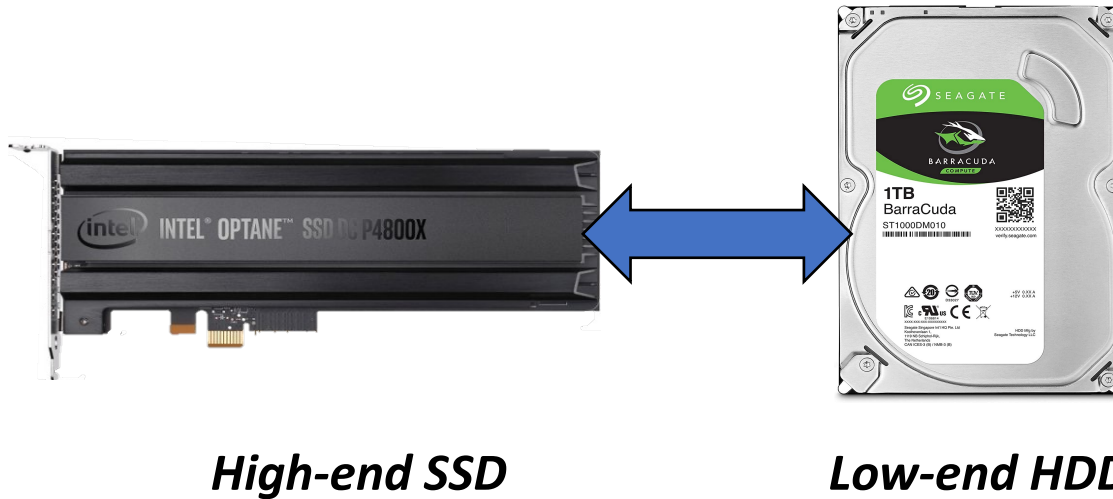
Evaluation Methodology (1/3)

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

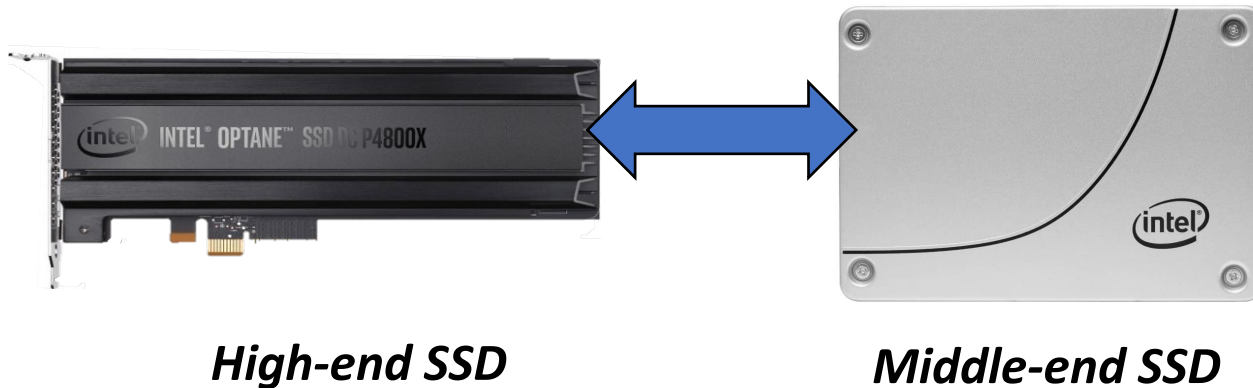


Evaluation Methodology (2/3)

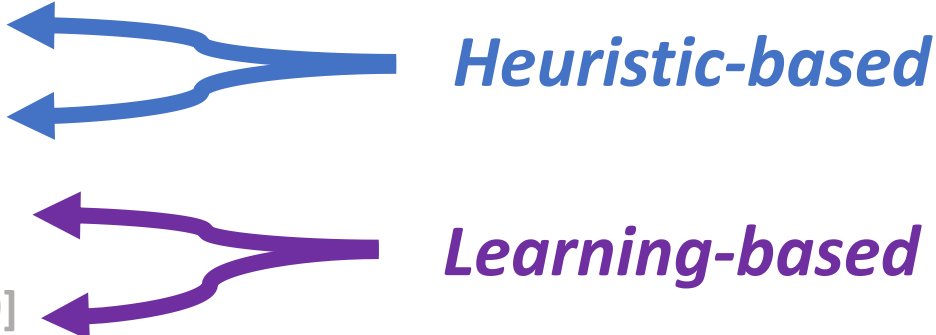
Cost-Oriented HSS Configuration



Performance-Oriented HSS Configuration



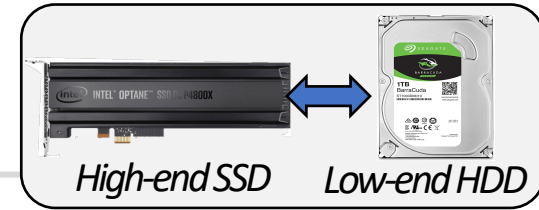
Evaluation Methodology (3/3)

- **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]

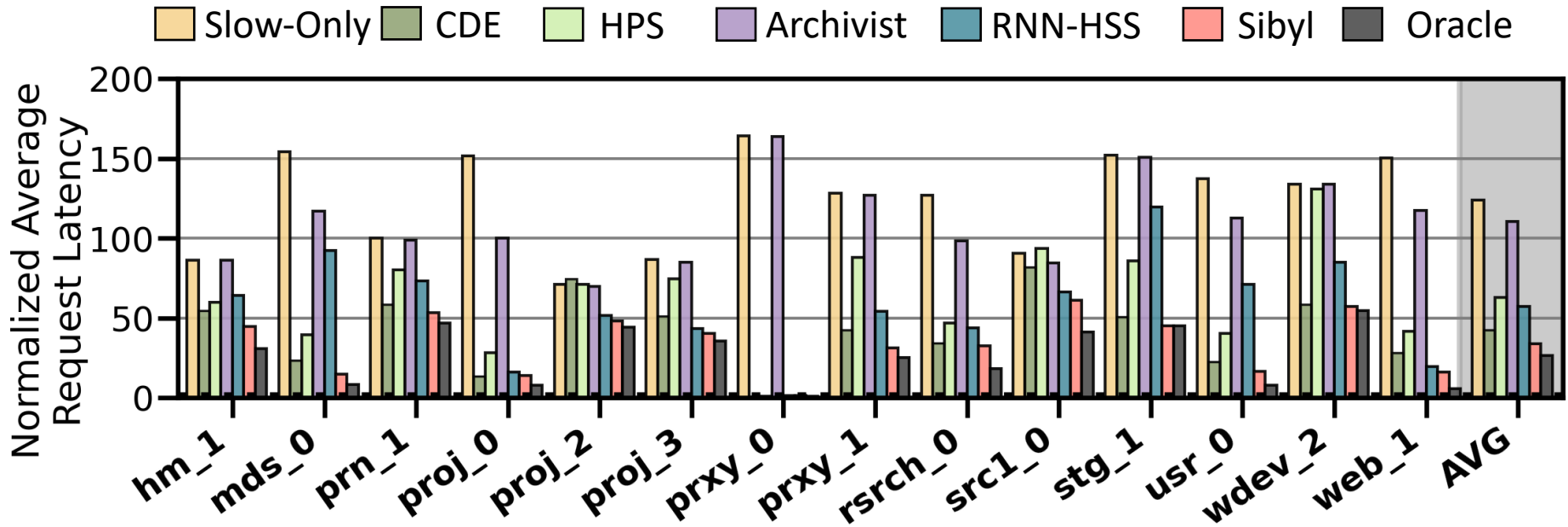
Heuristic-based

Learning-based

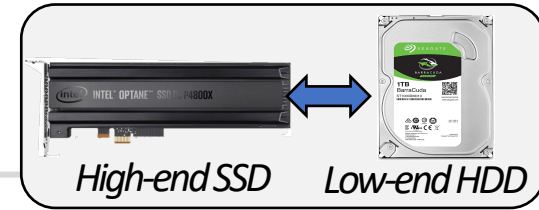
Performance Analysis



Cost-Oriented HSS Configuration

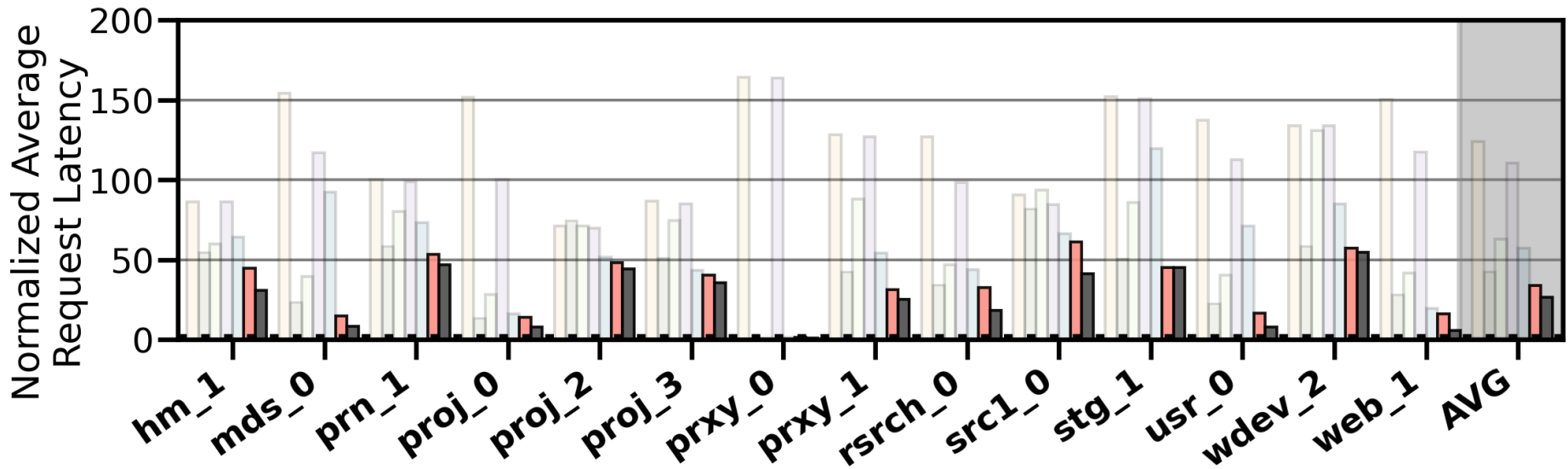


Performance Analysis



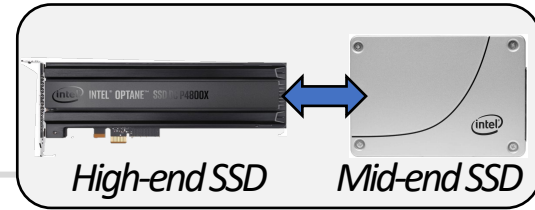
Cost-Oriented HSS Configuration

Slow-Only CDE HPS Archivist RNN-HSS Sibyl Oracle

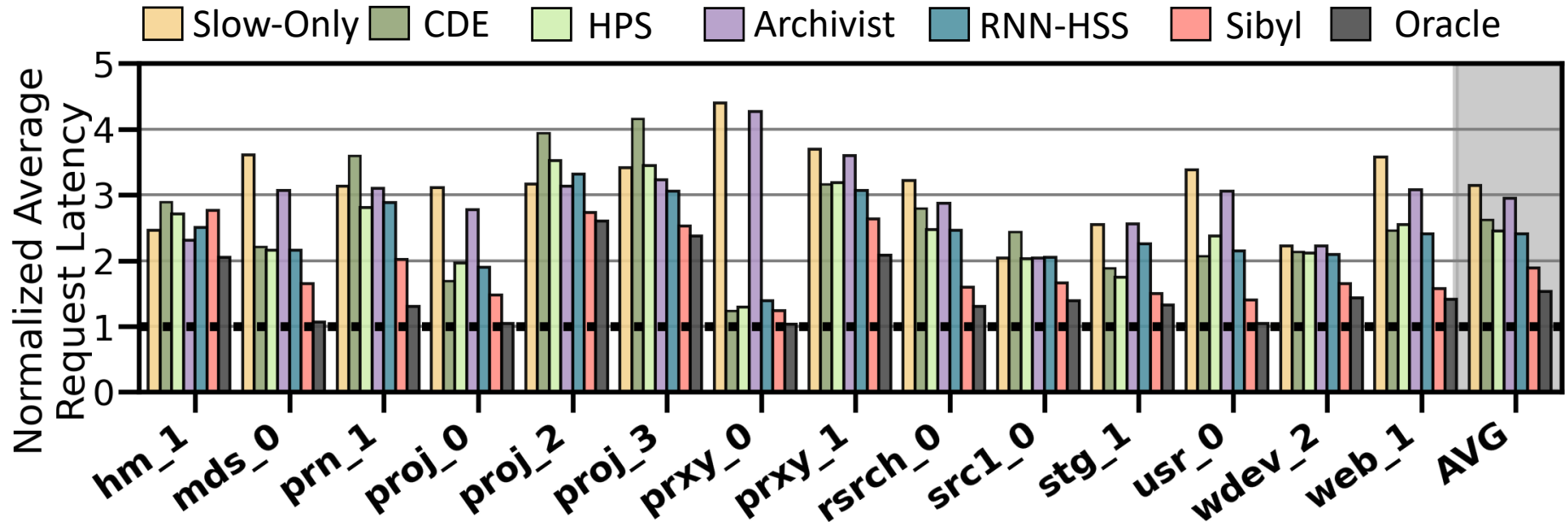


Sibyl consistently **outperforms all the baselines**
for all the workloads

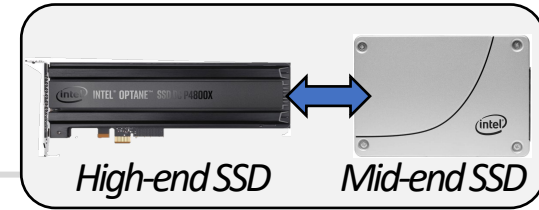
Performance Analysis



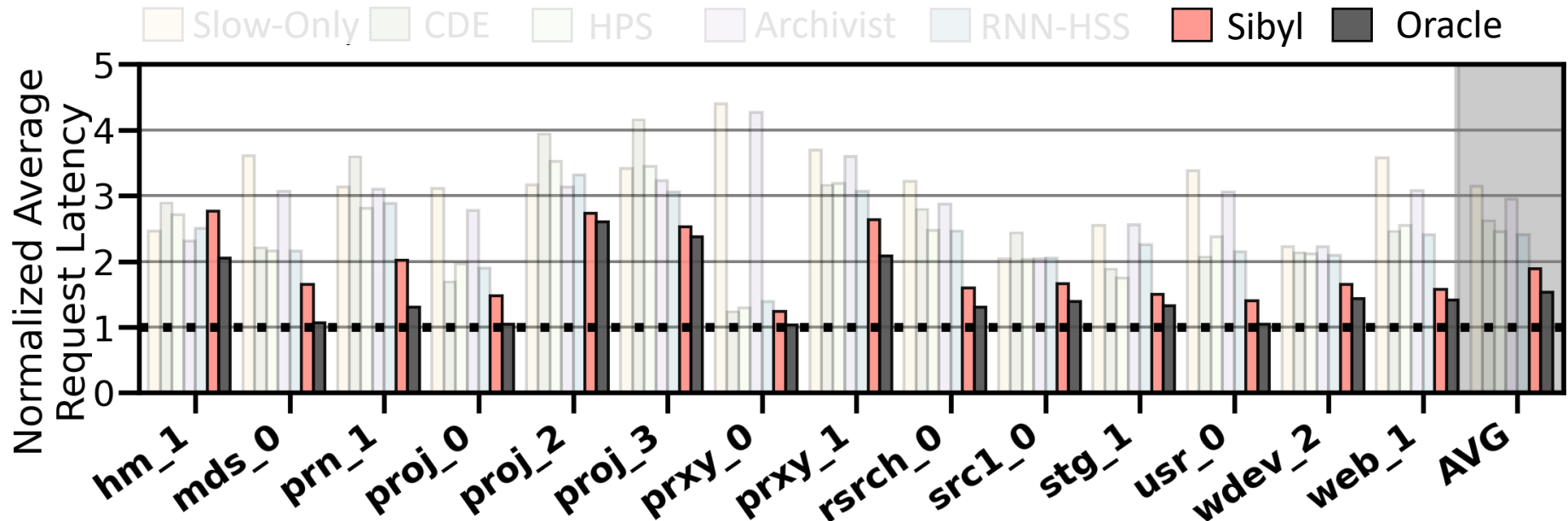
Performance-Oriented HSS Configuration



Performance Analysis

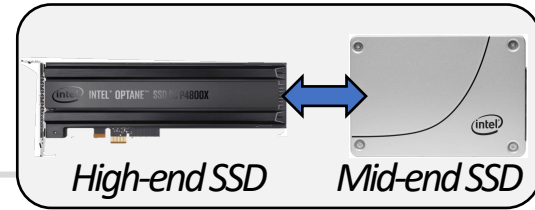


Performance-Oriented HSS Configuration

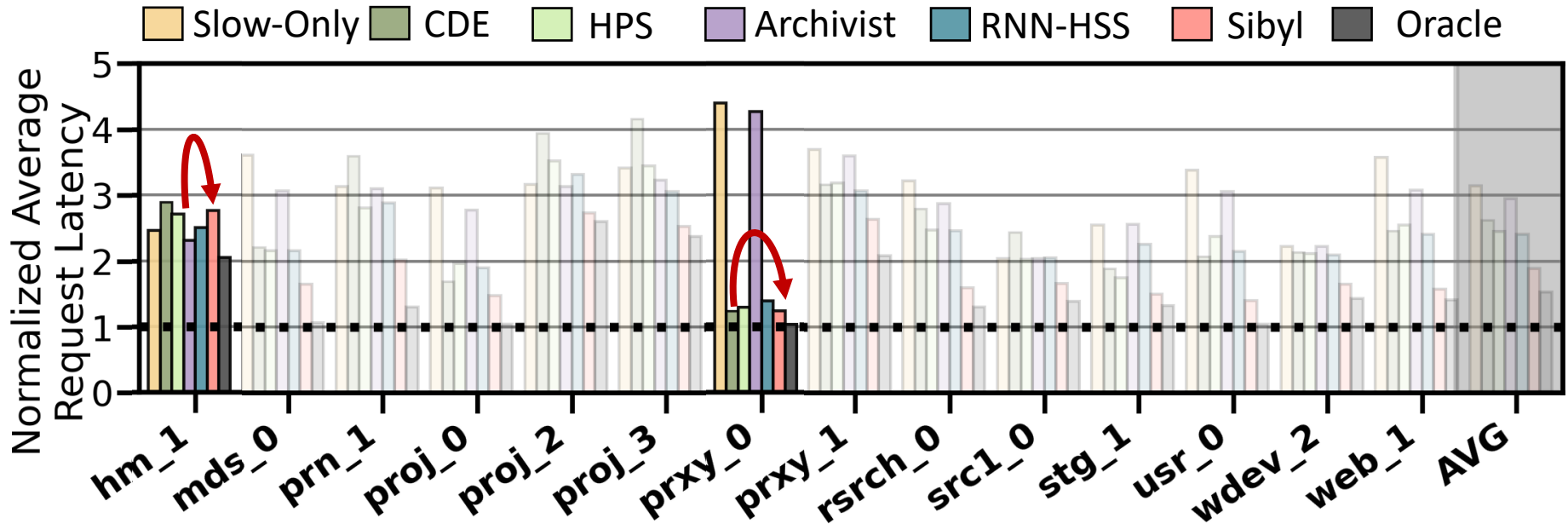


Sibyl provides **21.6% performance improvement** by **dynamically adapting its data placement policy**

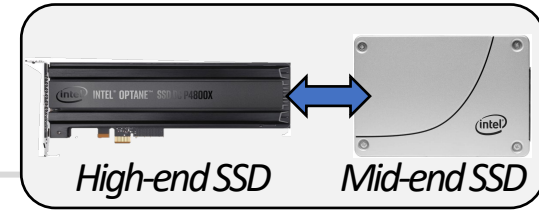
Performance Analysis



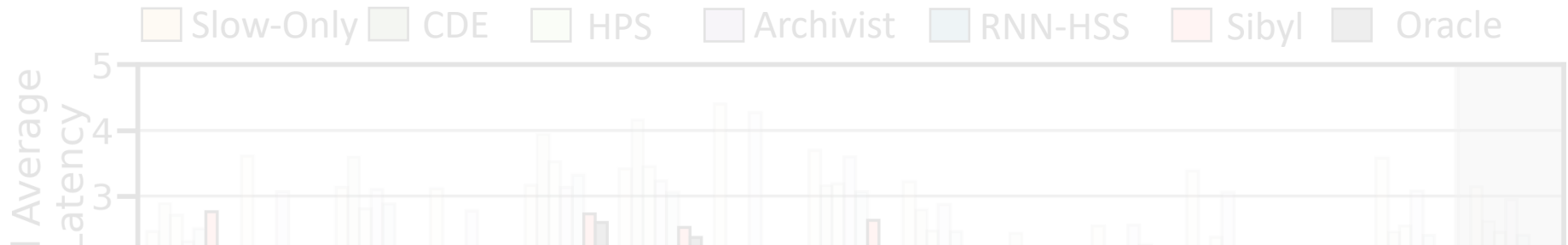
Performance-Oriented HSS Configuration



Performance Analysis

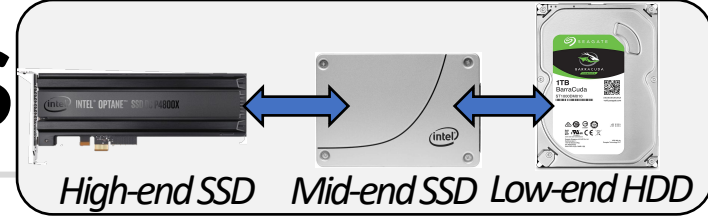


Performance-Oriented HSS Configuration



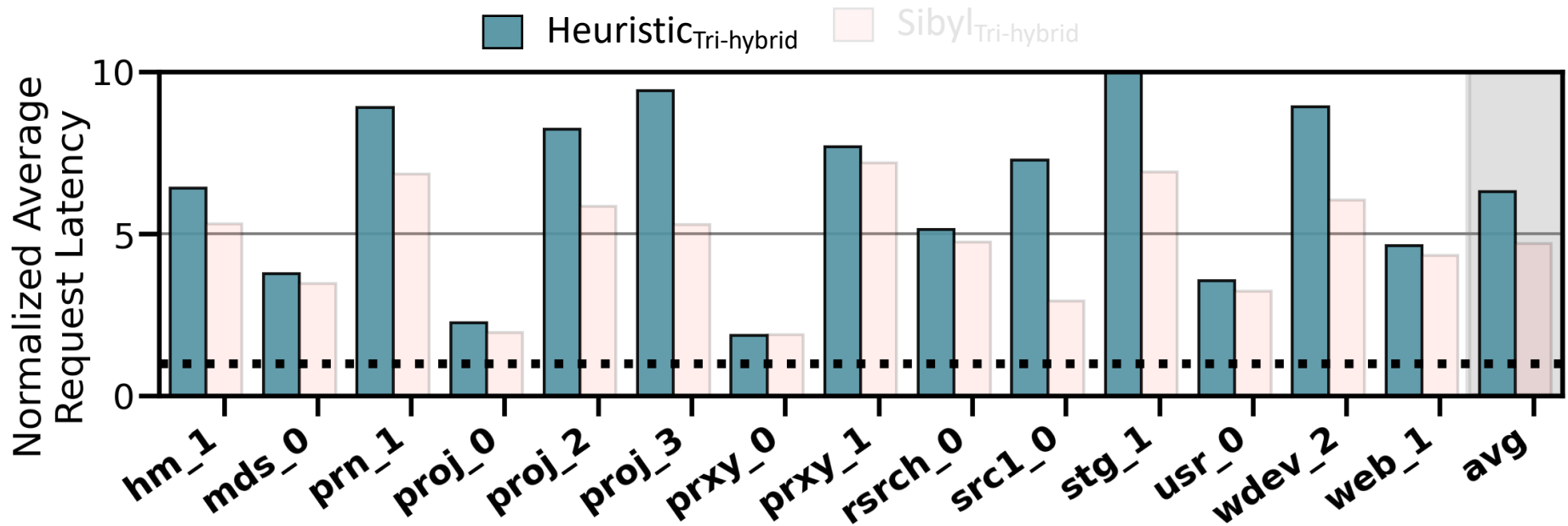
Sibyl achieves **80% of the performance of an oracle policy** that has complete knowledge of future access patterns

Performance on Tri-HSS

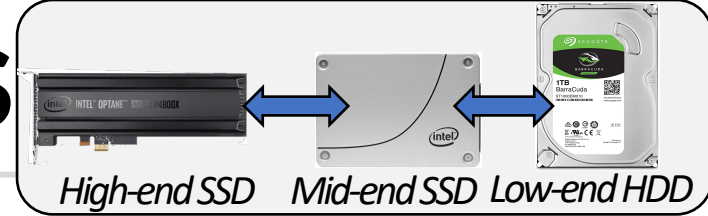


Extending Sibyl for **more devices**:

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

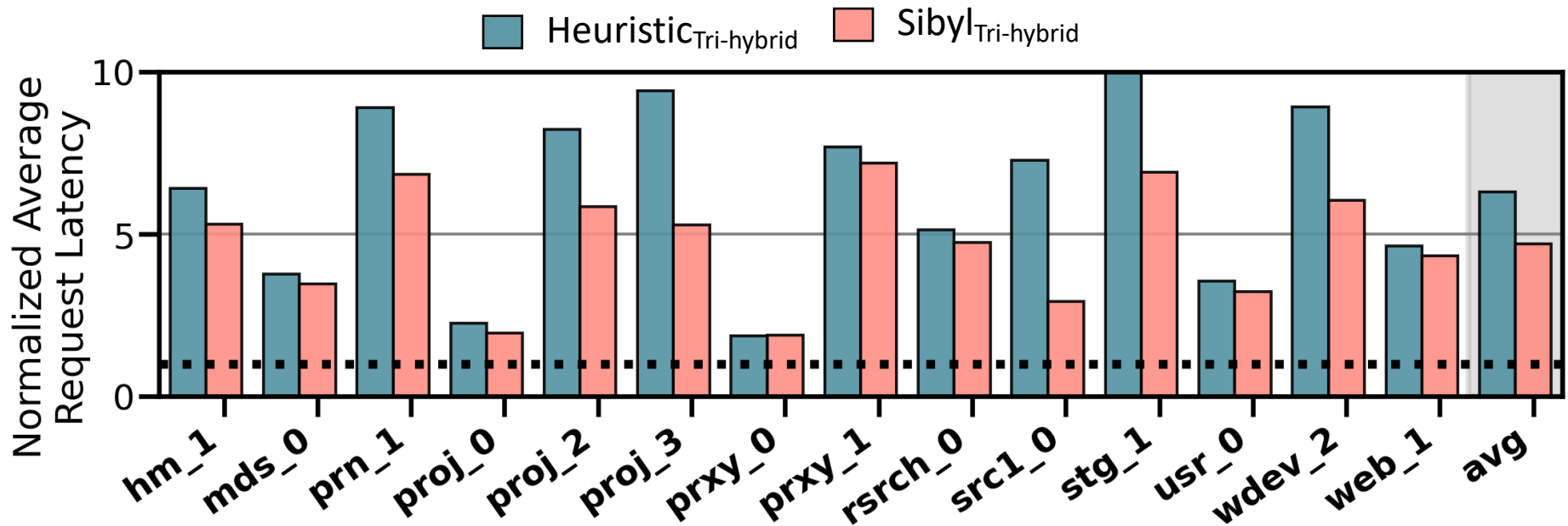


Performance on Tri-HSS

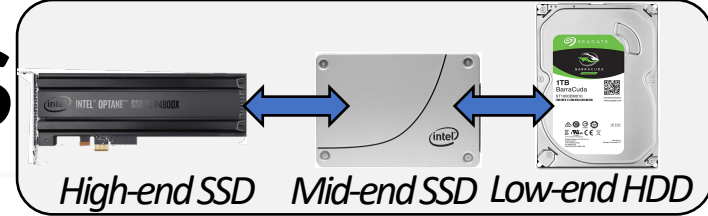


Extending Sibyl for **more devices**:

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 **~10ns**
- Training latency of **~2us**



Small area overhead



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>

Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

Sibyl: Overview

Evaluation of Sibyl and Key Results

Conclusion

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**

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

ISCA 2022 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

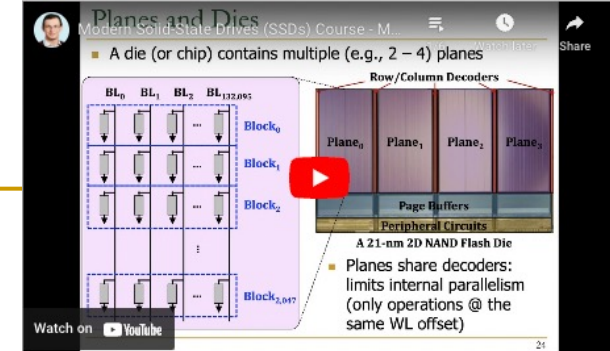
Gagandeep Singh¹ Rakesh Nadig¹ Jisung Park¹ Rahul Bera¹ Nastaran Hajinazar¹
David Novo³ Juan Gómez-Luna¹ Sander Stuijk² Henk Corporaal² Onur Mutlu¹

¹ETH Zürich

²Eindhoven University of Technology

³LIRMM, Univ. Montpellier, CNRS

SSD Course (Spring 2023)



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	

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&p=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>

Comp Arch (Fall 2021)

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>


Computer Architecture - Fall 2021

Recent Changes
Media Manager
Sitemap

Trace: readings · start · schedule

Home
Announcements
Materials

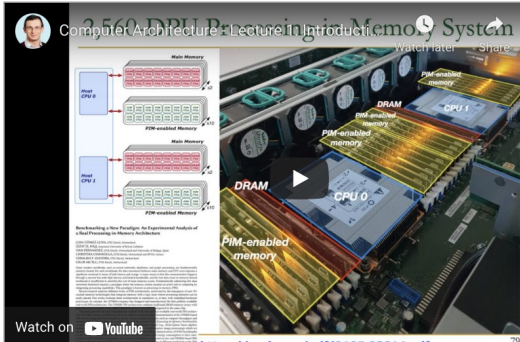
- Lectures/Schedule
- Lecture Buzzwords
- Readings
- HWs
- Labs
- Exams
- Related Courses
- Tutorials


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



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

Recorded Lecture Playlist



Watch on  <https://www.youtube.com/watch?v=Ucp0TTmvqOE?e=4236>

Fall 2021 Lectures & Schedule

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

Hermes & Sibyl:

ML-Driven Memory & Storage Management

Onur Mutlu

omutlu@gmail.com

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

27 September 2023

VMware

SAFARI

ETH zürich

Carnegie Mellon

Hermes Discussion

- **FAQs**

- [What are the selected set of program features?](#)
- [Can you provide some intuition on why these features work?](#)
- [What happens in case of a misprediction?](#)
- [What's the performance headroom for off-chip prediction?](#)
- [Do you see a variance of different features in final prediction accuracy?](#)

- **Simulation Methodology**

- [System parameters](#)
- [Evaluated workloads](#)

- **More Results**

- [Percentage of off-chip requests](#)
- [Reduction in stall cycles by reducing the critical path](#)
- [Fraction of off-chip load requests](#)
- [Accuracy and coverage of POPET](#)
- [Effect of different features](#)
- [Are all features required?](#)
- [1C performance](#)
- [1C performance line graph](#)
- [1C performance against prior predictors](#)
- [Effect on stall cycles](#)
- [8C performance](#)
- Sensitivity:
 - [Hermes request issue latency](#)
 - [Cache hierarchy access latency](#)
 - [Activation threshold](#)
 - [ROB size](#)
 - [LLC size](#)
- [Power overhead](#)
- [Accuracy without prefetcher](#)
- [Main memory request overhead with different prefetchers](#)

HERMES BACKUP

Initial Set of Program Features

Features without control-flow information	Features with control-flow information
<ol style="list-style-type: none">1. Load virtual address2. Virtual page number3. Cacheline offset in page4. First access5. Cacheline offset + first access6. Byte offset in cacheline7. Word offset in cacheline	<ol style="list-style-type: none">8. Load PC9. $PC \oplus$ load virtual address10. $PC \oplus$ virtual page number11. $PC \oplus$ cacheline offset12. $PC +$ first access13. $PC \oplus$ byte offset14. $PC \oplus$ word offset15. Last-4 load PCs16. Last-4 PCs

Selected Set of Program Features

Five features

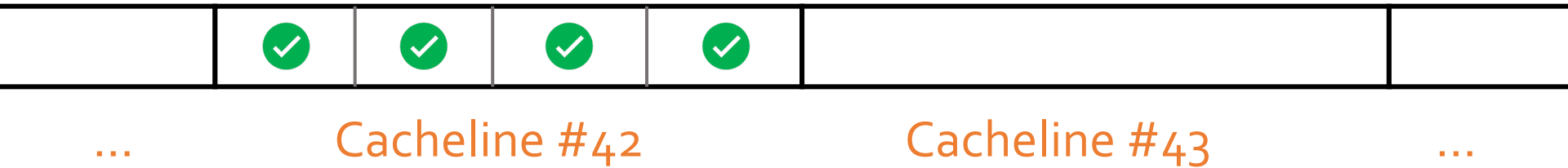
- $PC \oplus$ cacheline offset
- $PC \oplus$ byte offset
- $PC \llcorner$ first access
- Cacheline offset \llcorner first access
- Last-4 load PCs

A **binary hint** that represents whether or not a cacheblock has been recently touched

When A Feature Works/Does Not Work?

Trace: 462.libquantum-1343B

PC: 0x401442



Without prefetcher

- PC + first access
- Cacheline offset + first access

With a simple stride prefetcher

- Cacheline offset + first access

What Happens in case of a Misprediction?

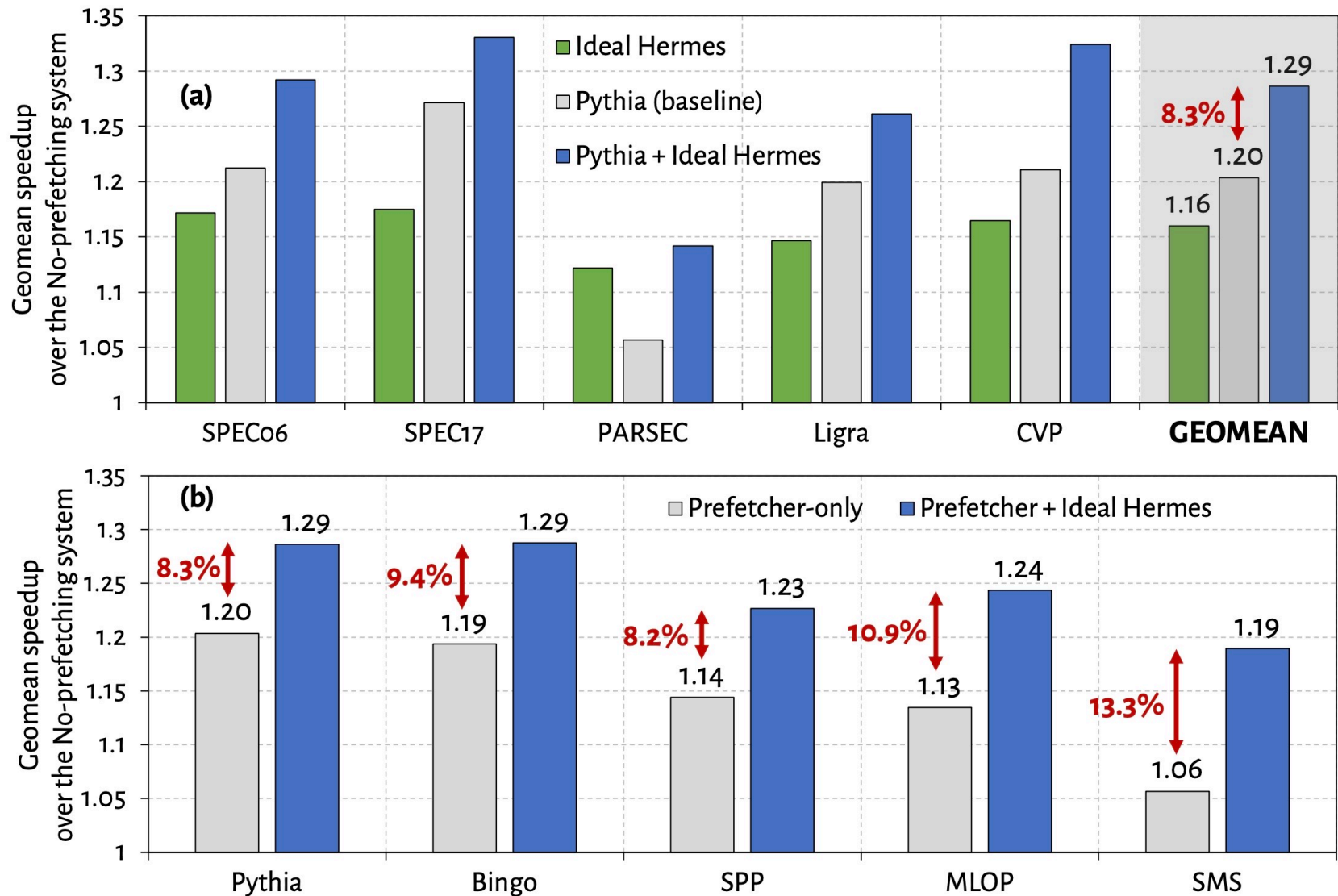
- Two cases of mispredictions:
- Predicted on-chip but actually goes off-chip
 - Loss of performance improvement opportunity

No need for misprediction detection and recovery

- Predicted off-chip but actually is on-chip
 - Memory controller forwards the data to LLC if and only if a load to the same address have already missed LLC and arrived at the memory controller

No need for misprediction detection and recovery

Performance Headroom of Off-Chip Prediction



System Parameters

Table 4: Simulated system parameters

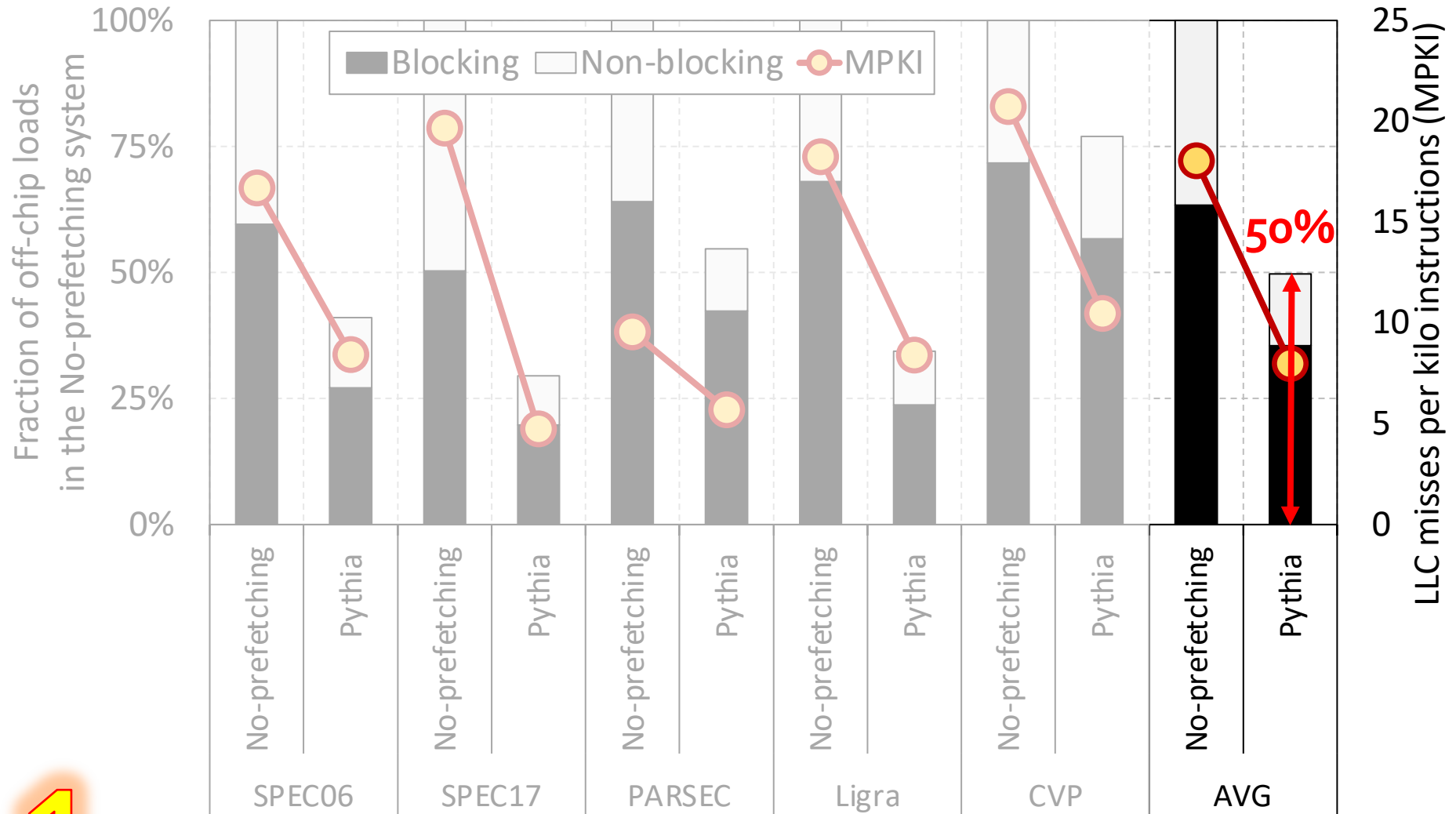
Core	1 and 8 cores, 6-wide fetch/execute/commit, 512-entry ROB, 128/72-entry LQ/SQ, Perceptron branch predictor [61] with 17-cycle misprediction penalty
L1/L2 Caches	Private, 48KB/1.25MB, 64B line, 12/20-way, 16/48 MSHRs, LRU, 5/15-cycle round-trip latency [25]
LLC	3MB/core, 64B line, 12 way, 64 MSHRs/slice, SHiP [122], 55-cycle round-trip latency [24, 25], Pythia prefetcher [32]
Main Memory	1C: 1 channel, 1 rank per channel; 8C: 4 channels, 2 ranks per channel; 8 banks per rank, DDR4-3200 MTPS, 64b data-bus per channel, 2KB row buffer per bank, tRCD=12.5ns, tRP=12.5ns, tCAS=12.5ns
Hermes	Hermes-O/P: 6/18-cycle Hermes request issue latency

Evaluated Workloads

Table 5: Workloads used for evaluation

Suite	#Workloads	#Traces	Example Workloads
SPEC06	14	22	gcc, mcf, cactusADM, lbm, ...
SPEC17	11	23	gcc, mcf, pop2, fotonik3d, ...
PARSEC	4	12	canneal, facesim, raytrace, ...
Ligra	11	20	BFS, PageRank, Radii, ...
CVP	33	33	integer, floating-point, server, ...

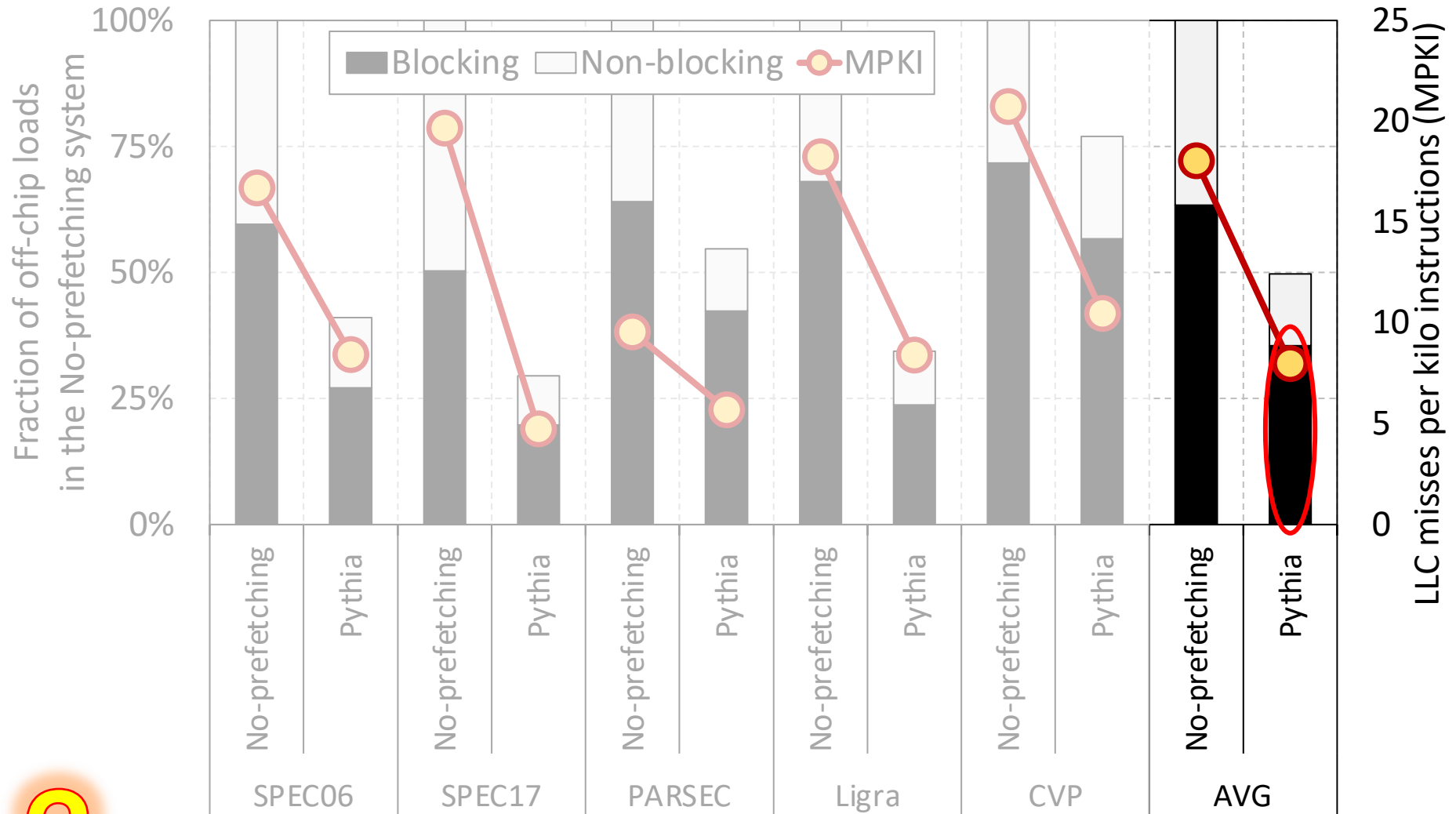
Observation: Not All Off-Chip Loads are Prefetched



1

Nearly **50%** of the loads are still **not prefetched**

Observation: Not All Off-Chip Loads are Prefetched



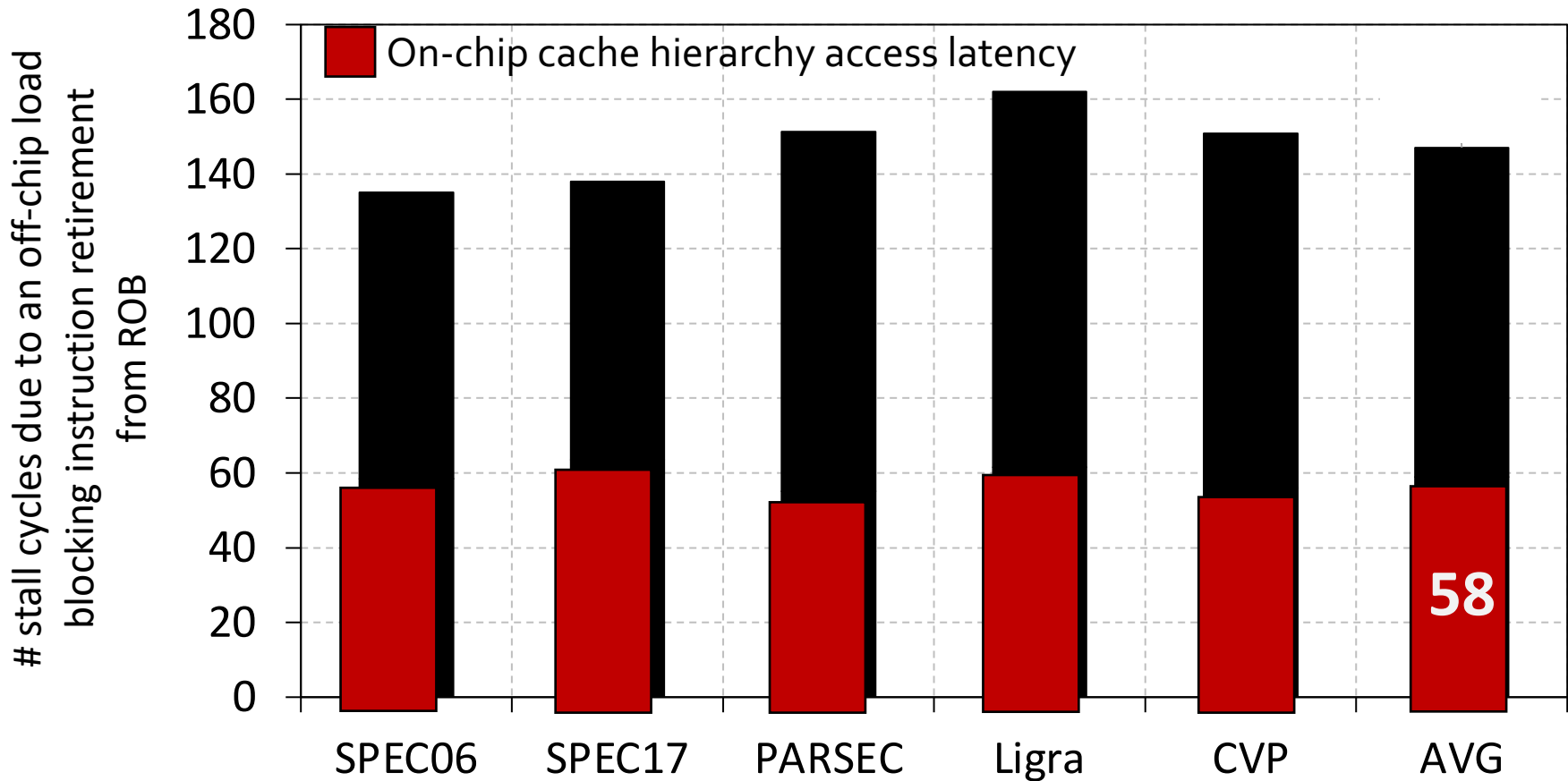
2

70% of these off-chip loads blocks ROB



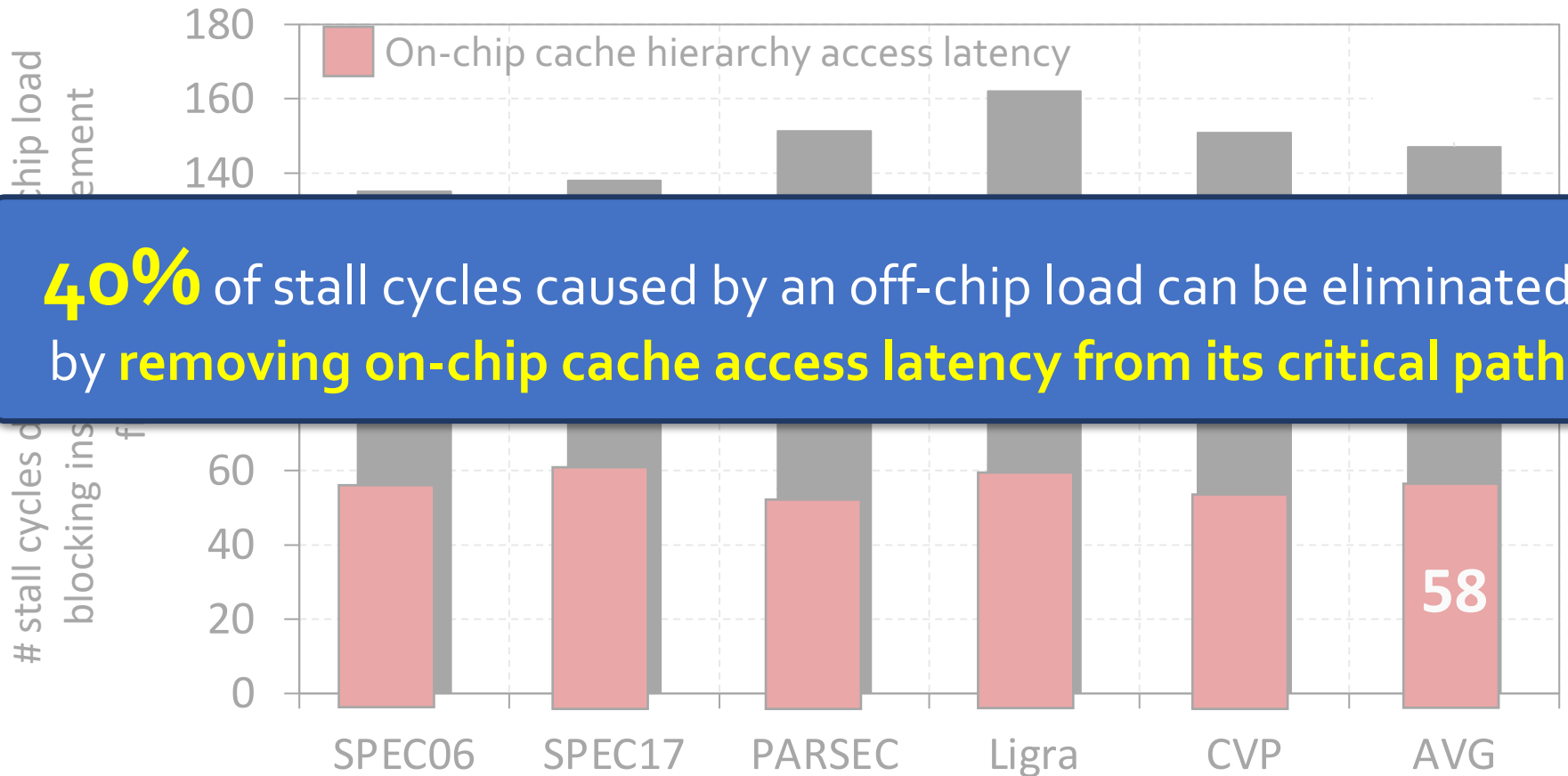
Observation: With Large Cache Comes Longer Latency

- On-chip cache access latency significantly contributes to the latency of an off-chip load

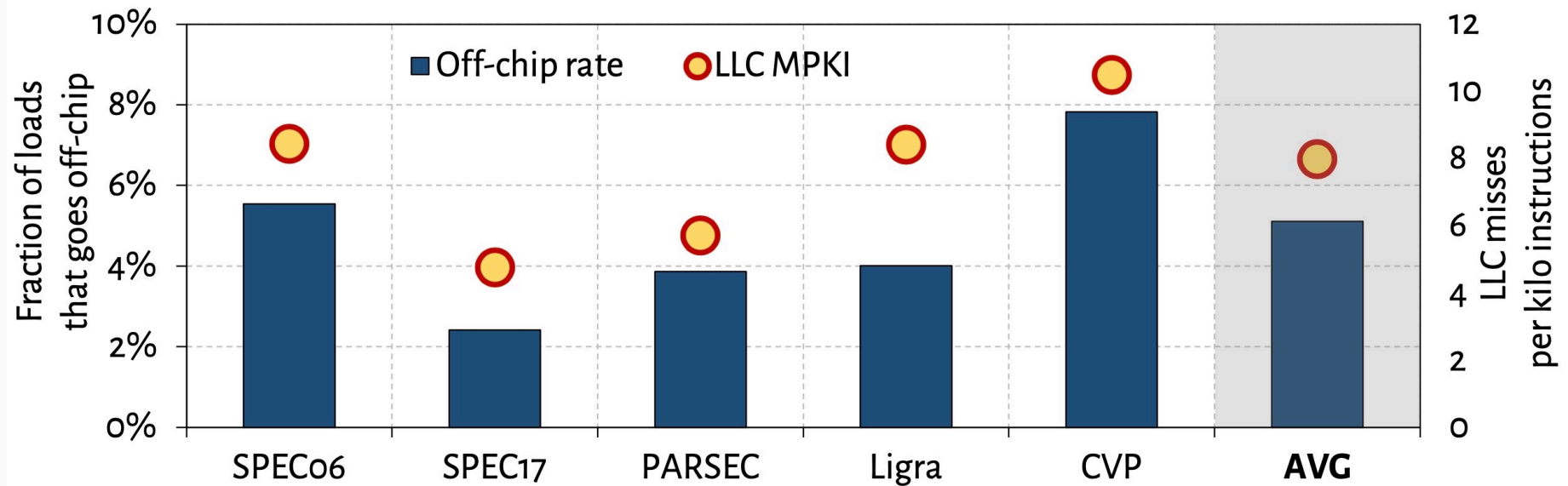


Observation: With Large Cache Comes Longer Latency

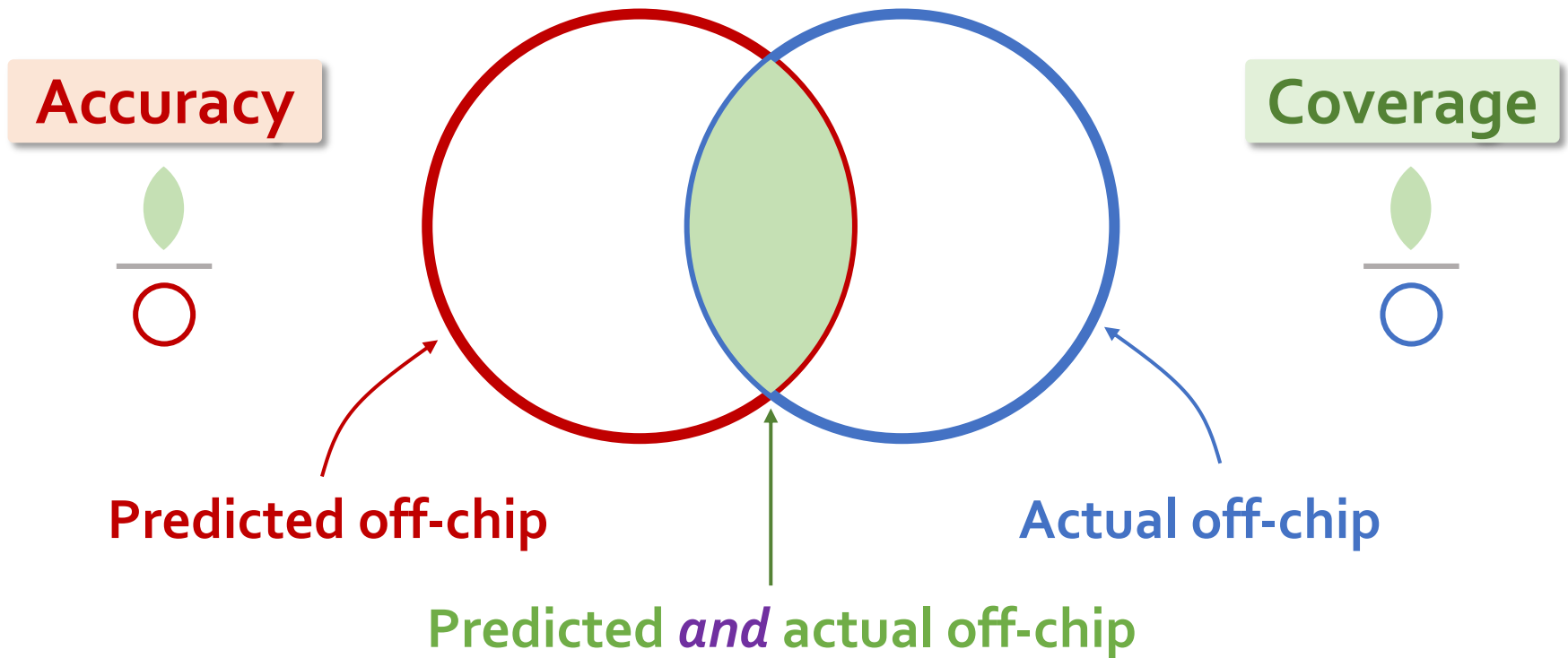
- On-chip cache access latency significantly contributes to the latency of an off-chip load



What Fraction of Load Requests Goes Off-Chip?

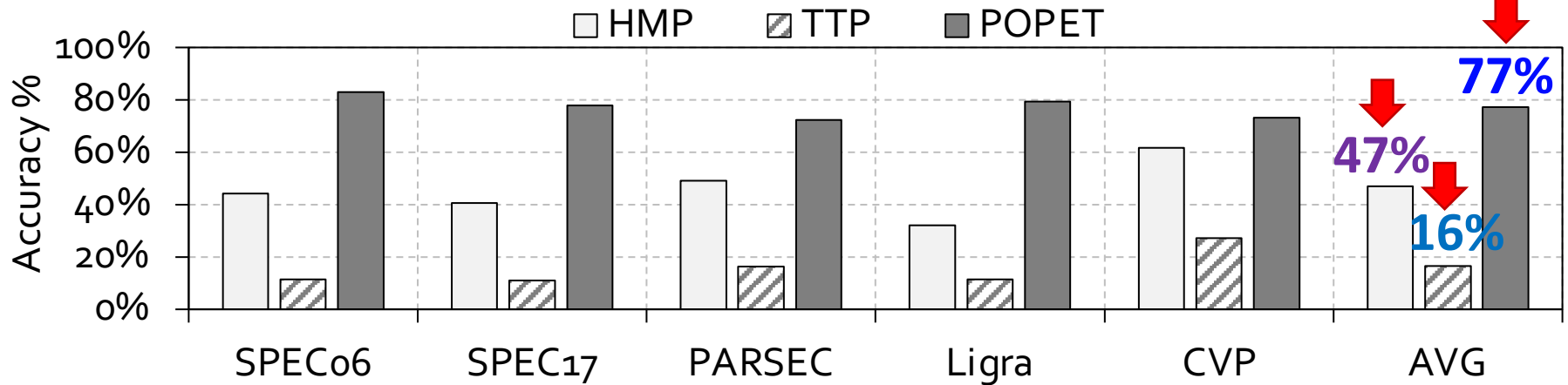


Off-Chip Prediction Quality: *Defining Metrics*

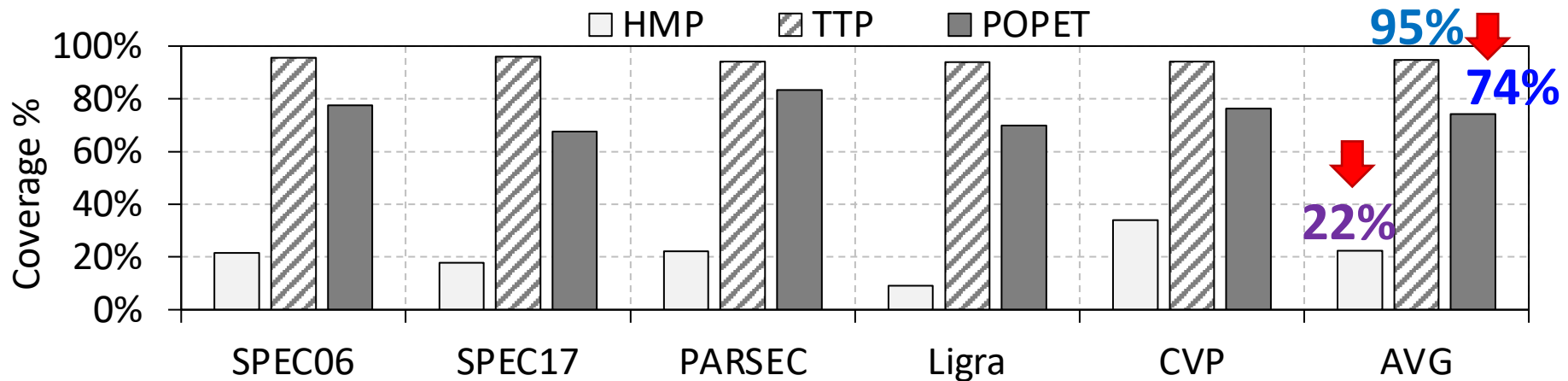


Off-Chip Prediction Quality: *Analysis*

Accuracy

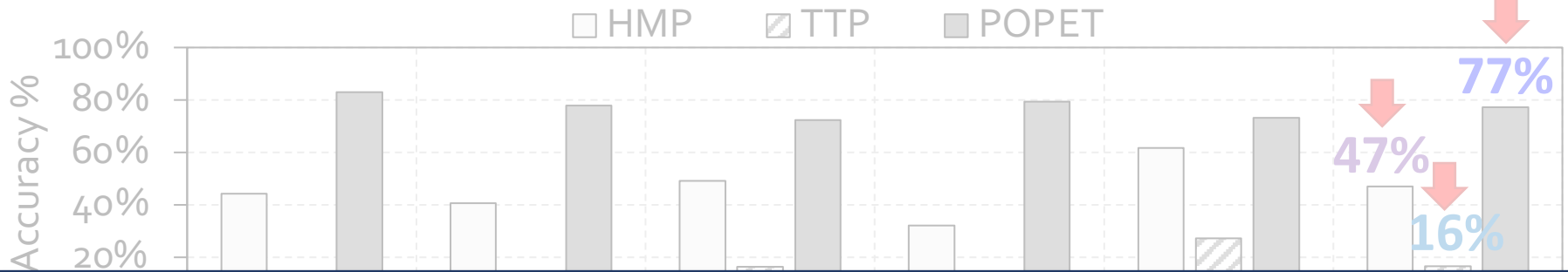


Coverage

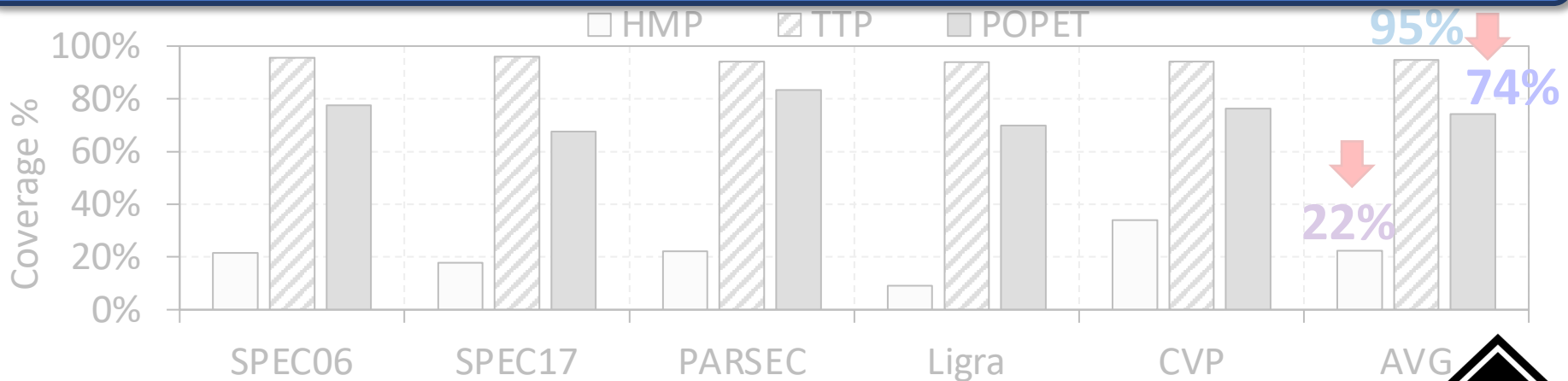


Off-Chip Prediction Quality: Analysis

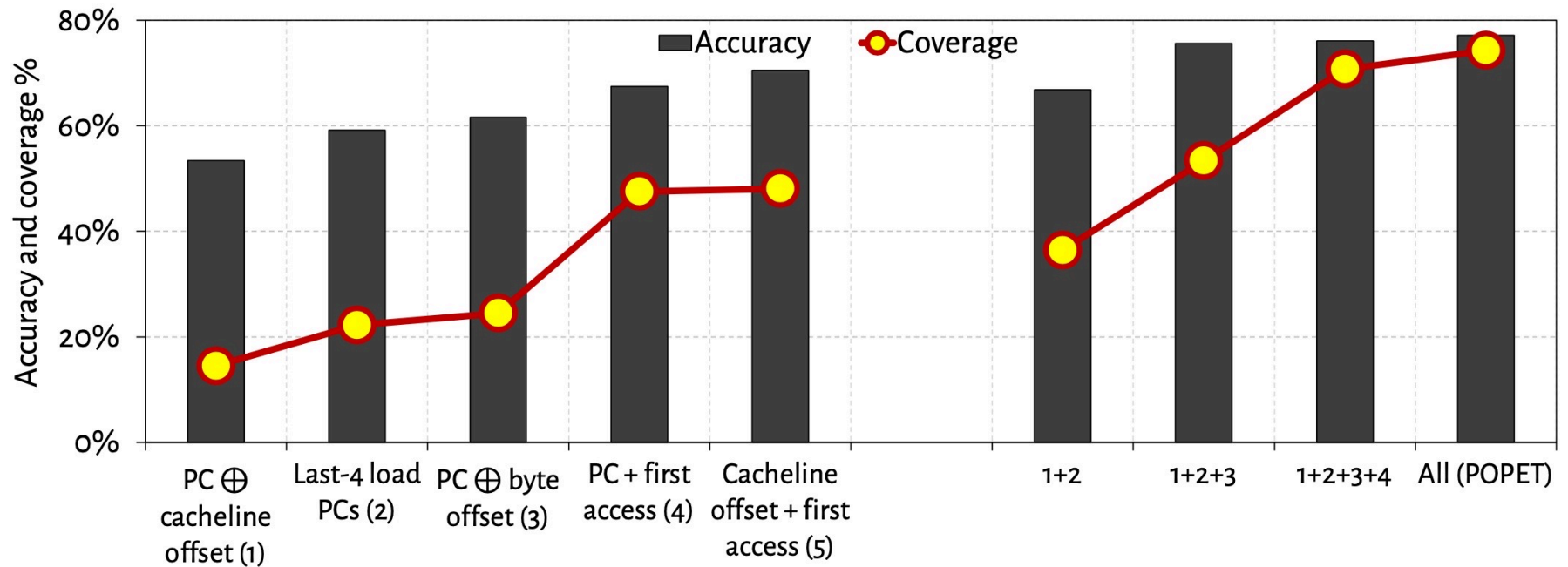
Accuracy



POPET provides off-chip predictions with **high-accuracy** and **high-coverage**

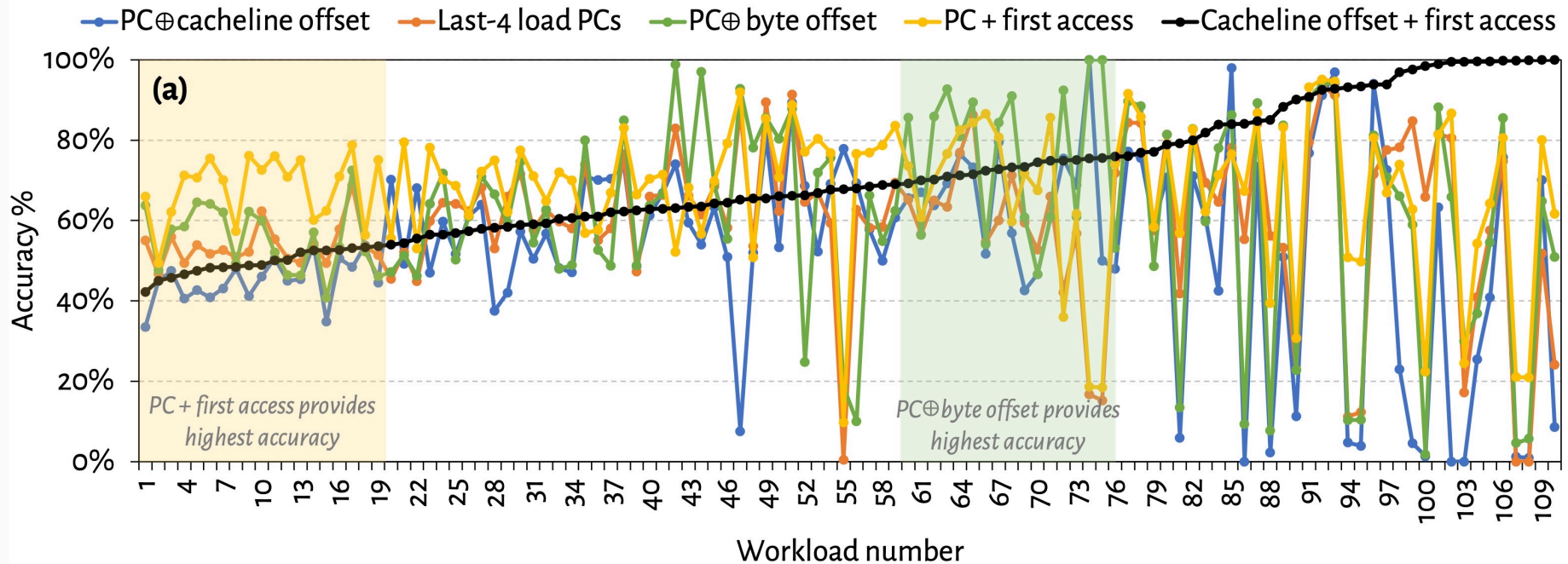


Effect of Different Features



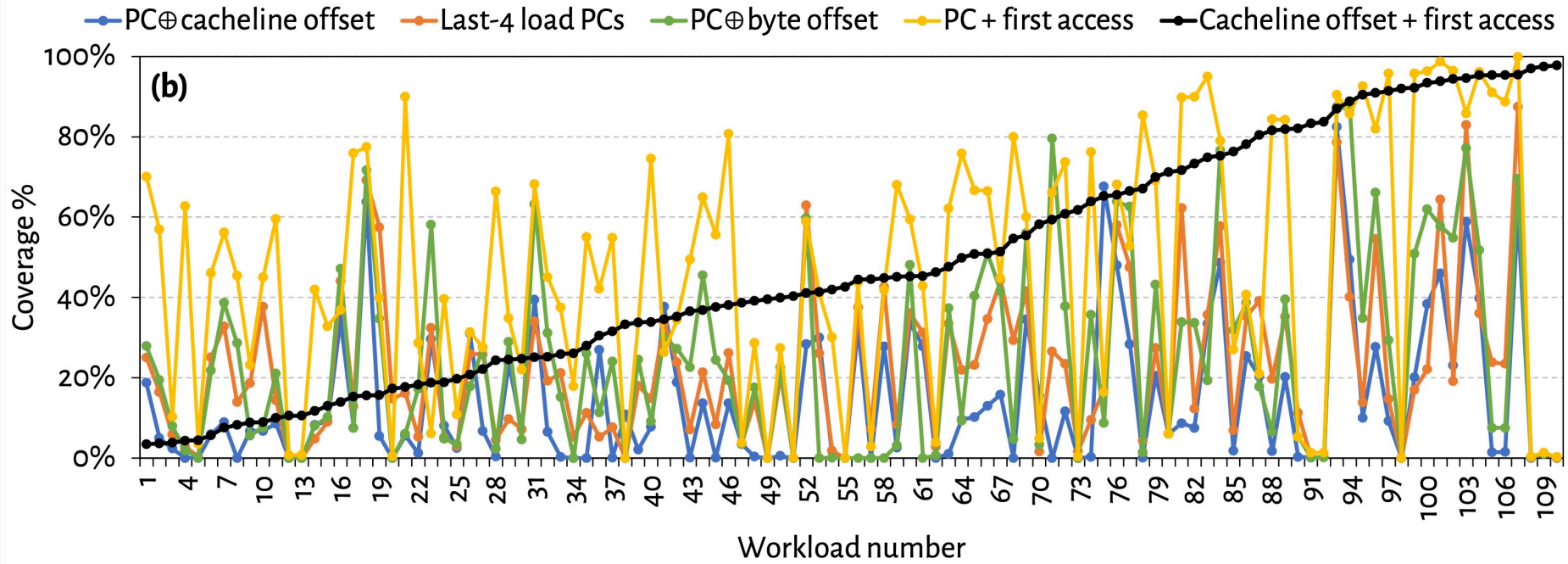
Combination of features provides both **higher accuracy and higher coverage** than any individual feature

Are All Features Required? (1)



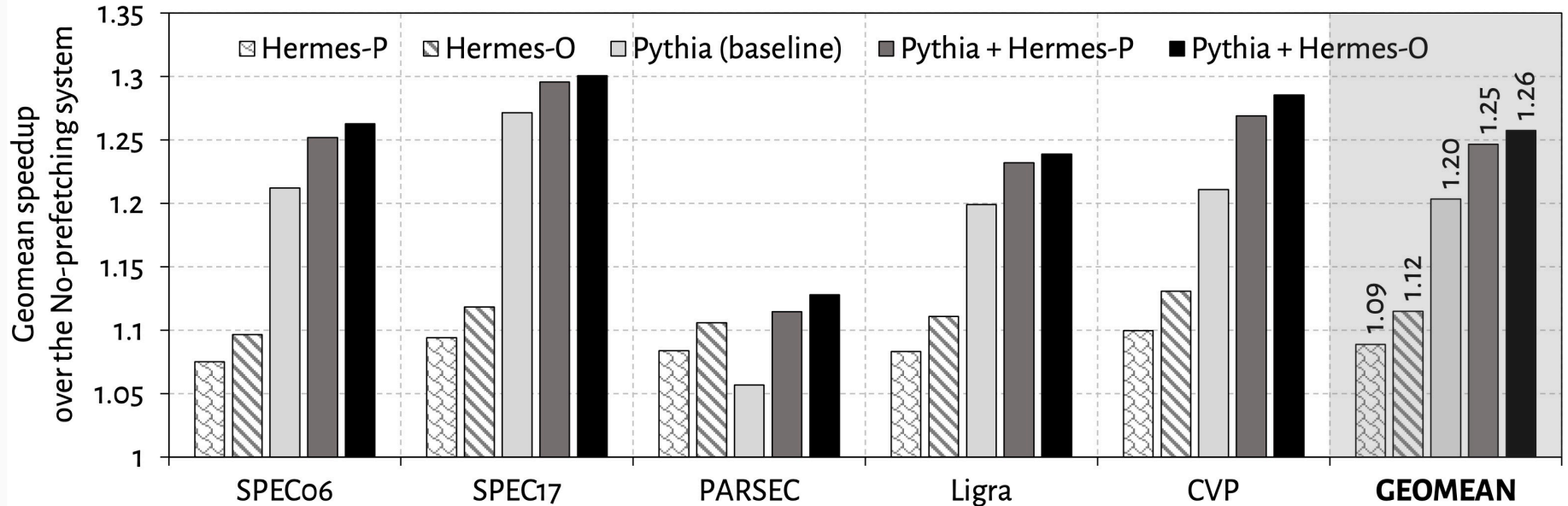
No single feature individually provides **highest prediction accuracy** across **all** workloads

Are All Features Required? (2)



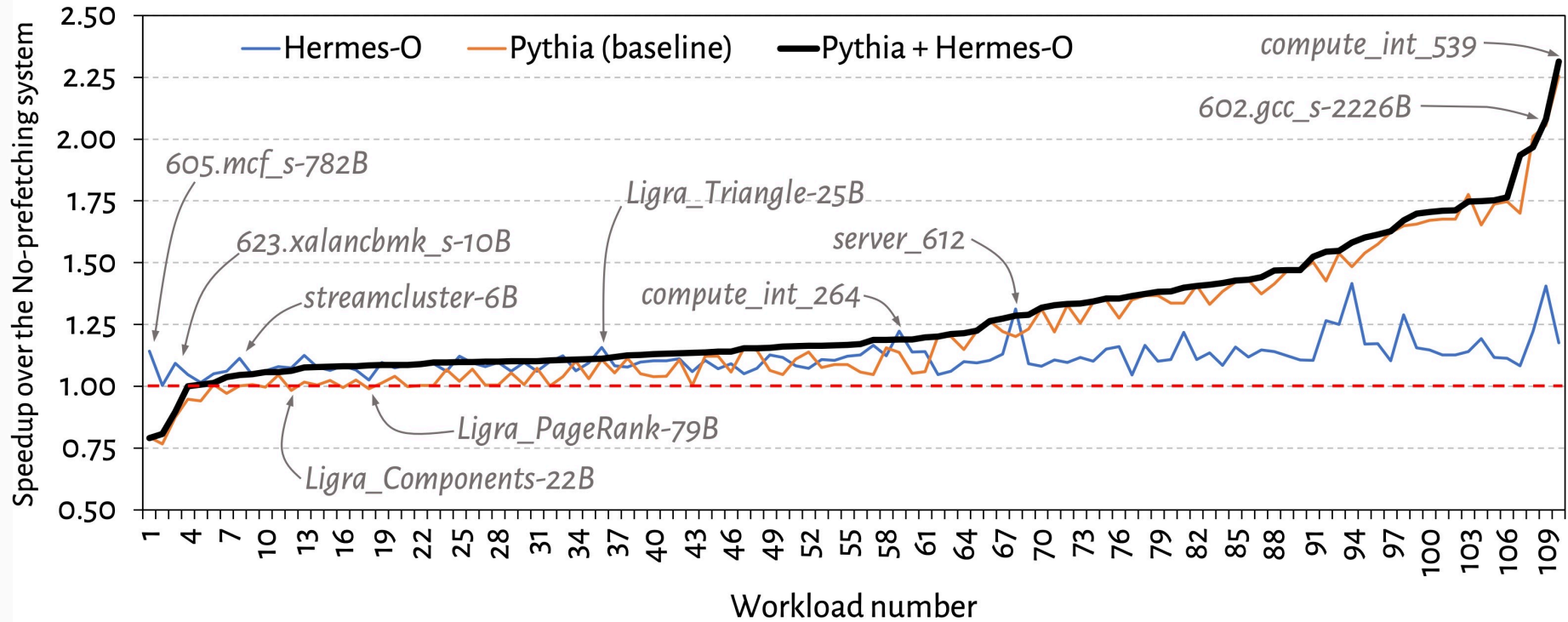
No single feature individually provides **highest prediction coverage** also across **all** workloads

Single-Core Performance

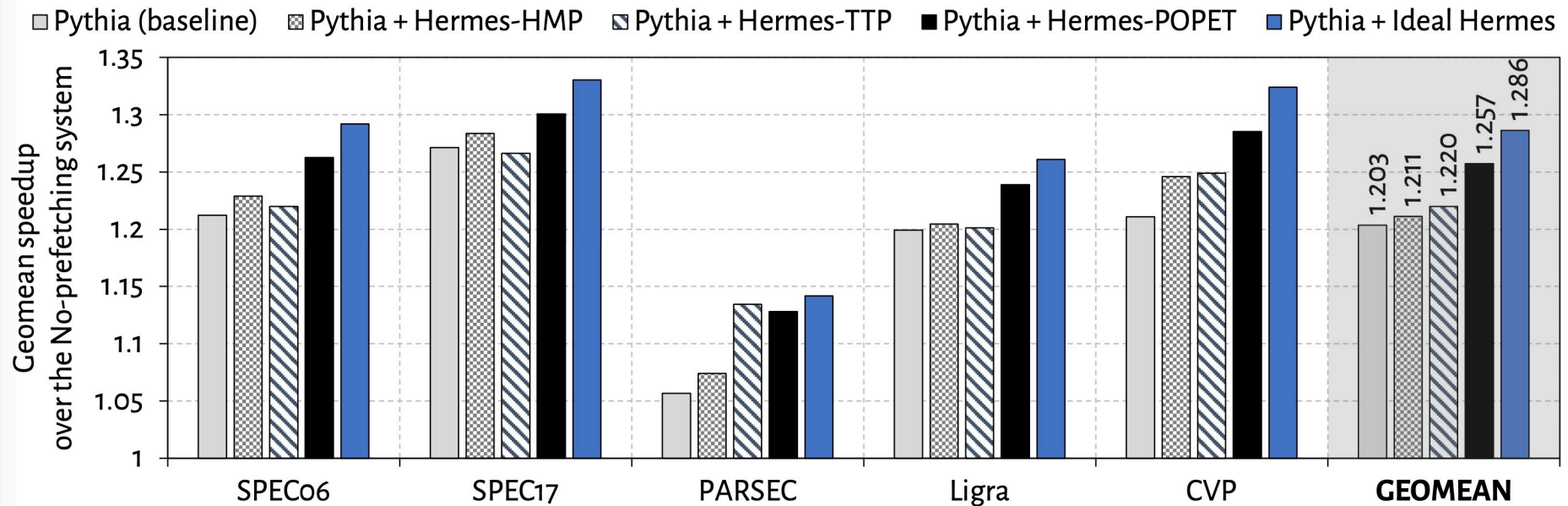


Hermes in combination with Pythia
outperforms **Pythia alone** in every workload category

Single-Core Performance Line Graph



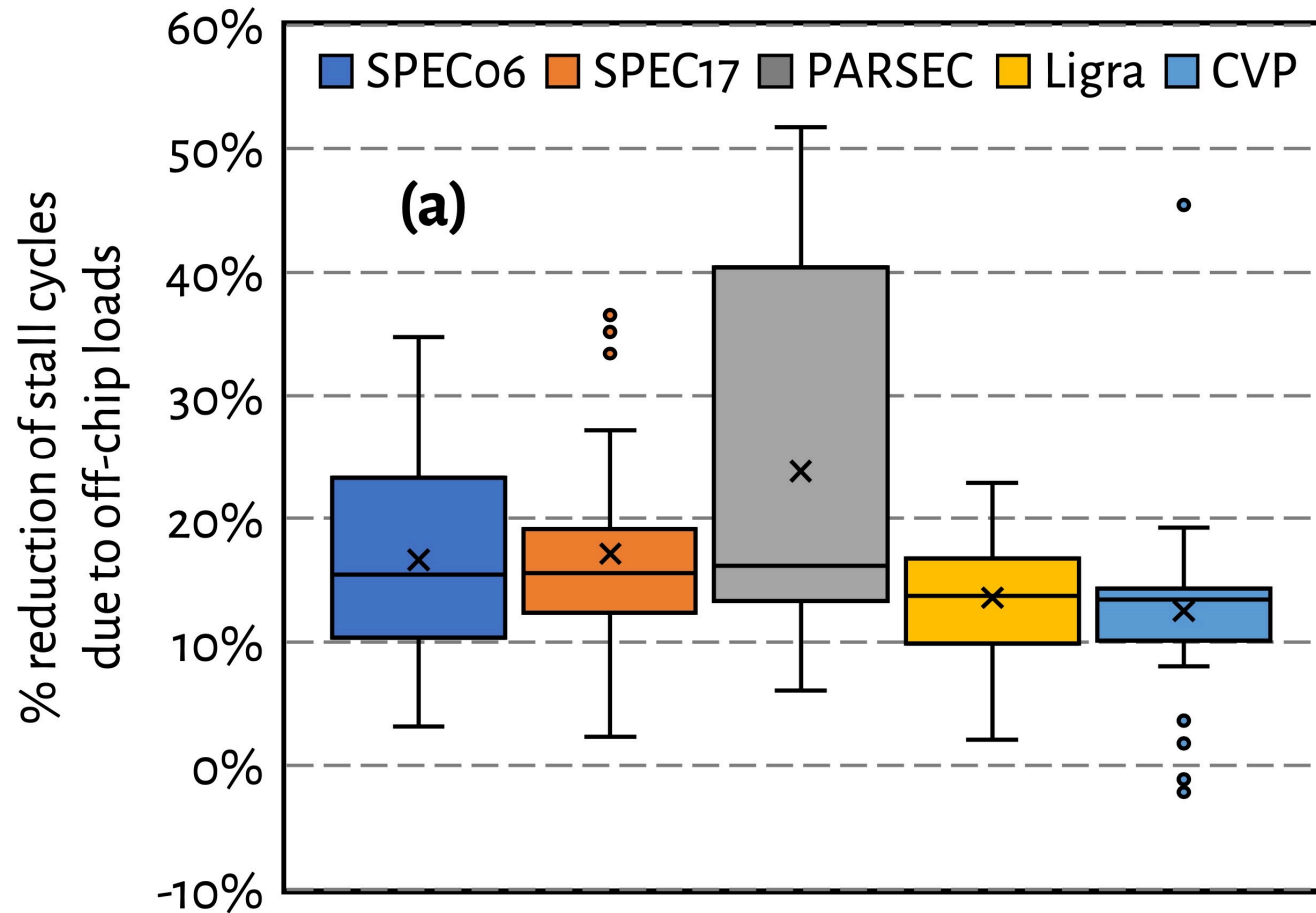
Single-Core Performance Against Prior Predictors



POPET provides higher performance benefit
than prior predictors

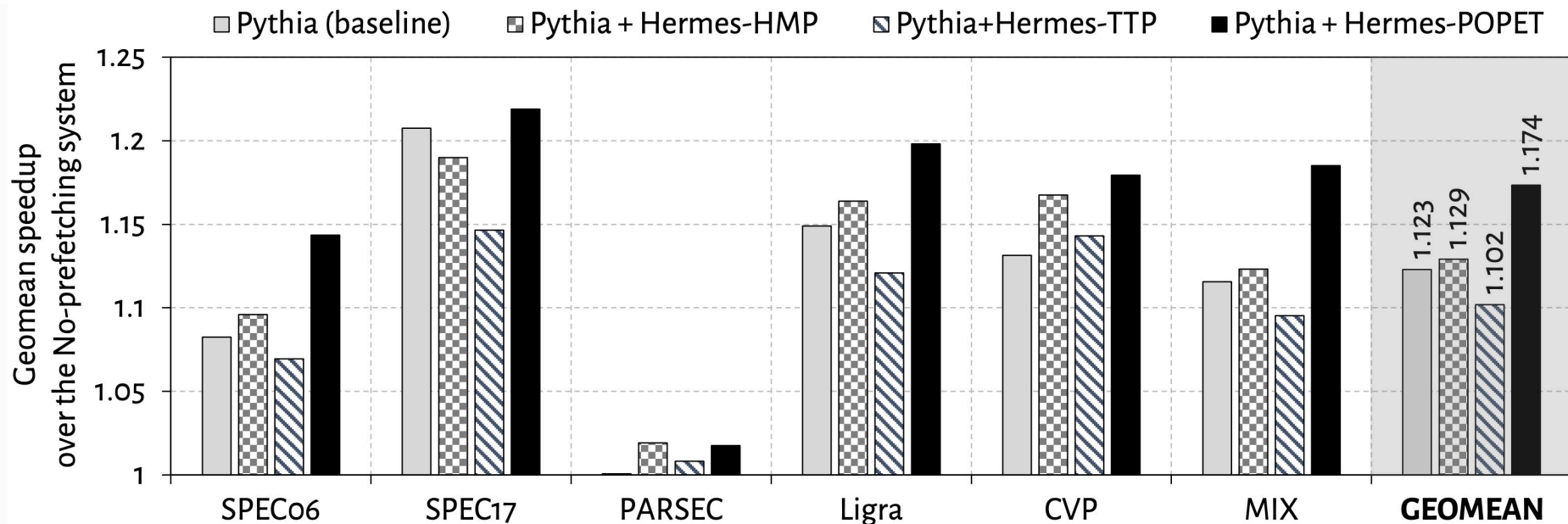
Hermes with POPET achieves nearly **90%** performance improvement of the Ideal Hermes

Effect on Stall Cycles



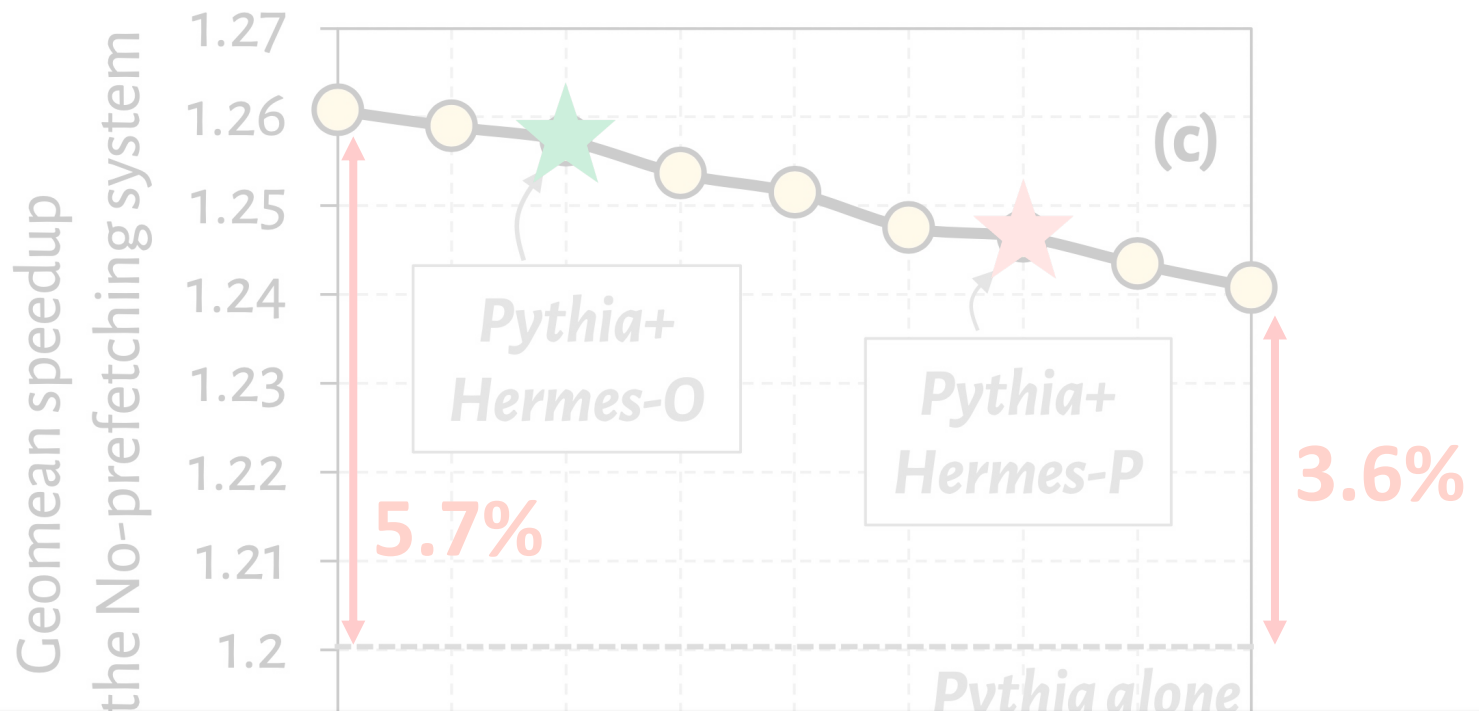
Hermes reduces off-chip load induced stall cycles on average by **16.2%** (up-to **51.8%**)

Eight-Core Performance



Hermes in combination with Pythia
outperforms Pythia alone by **5.1%** on average

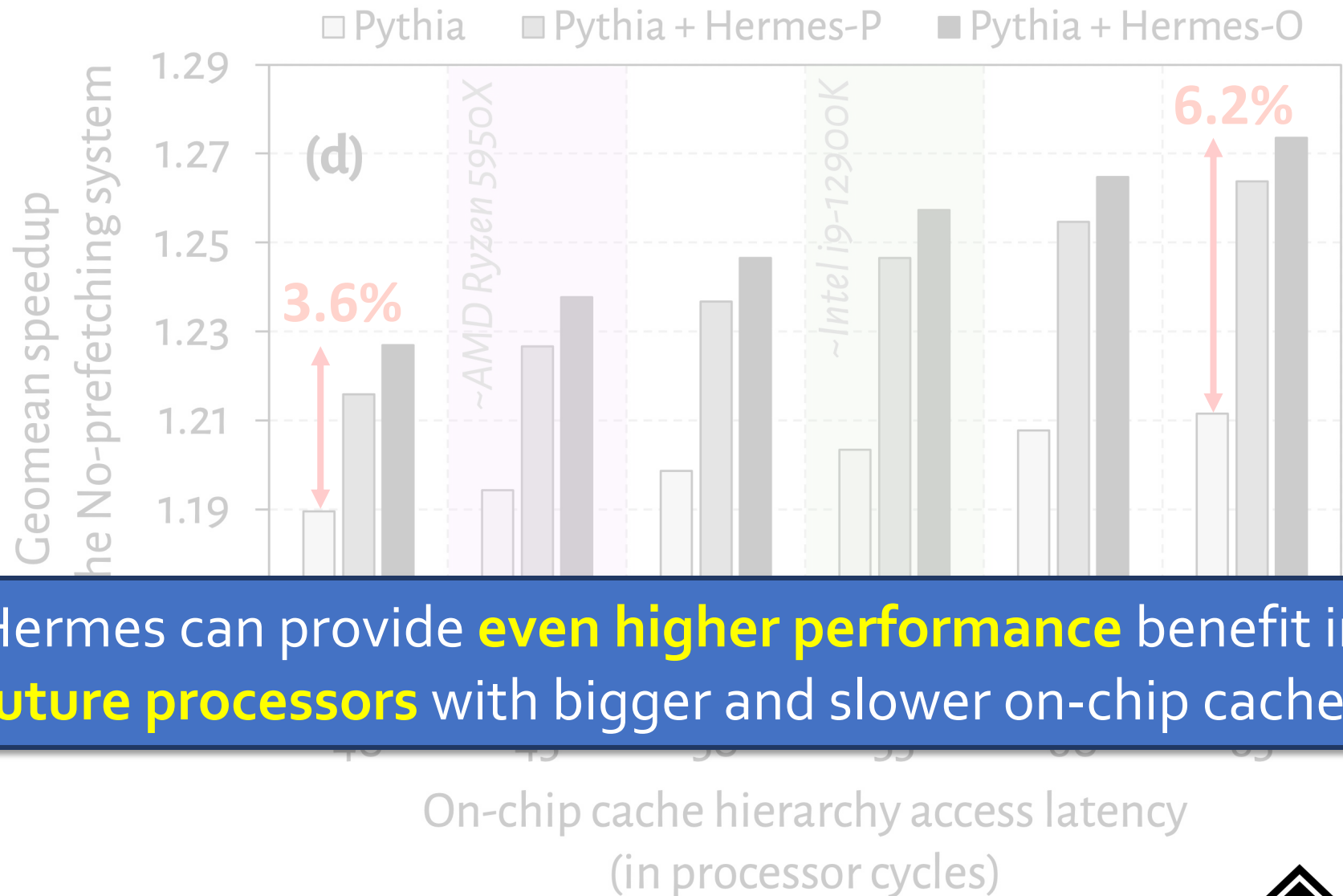
Effect of Hermes Request Issue Latency



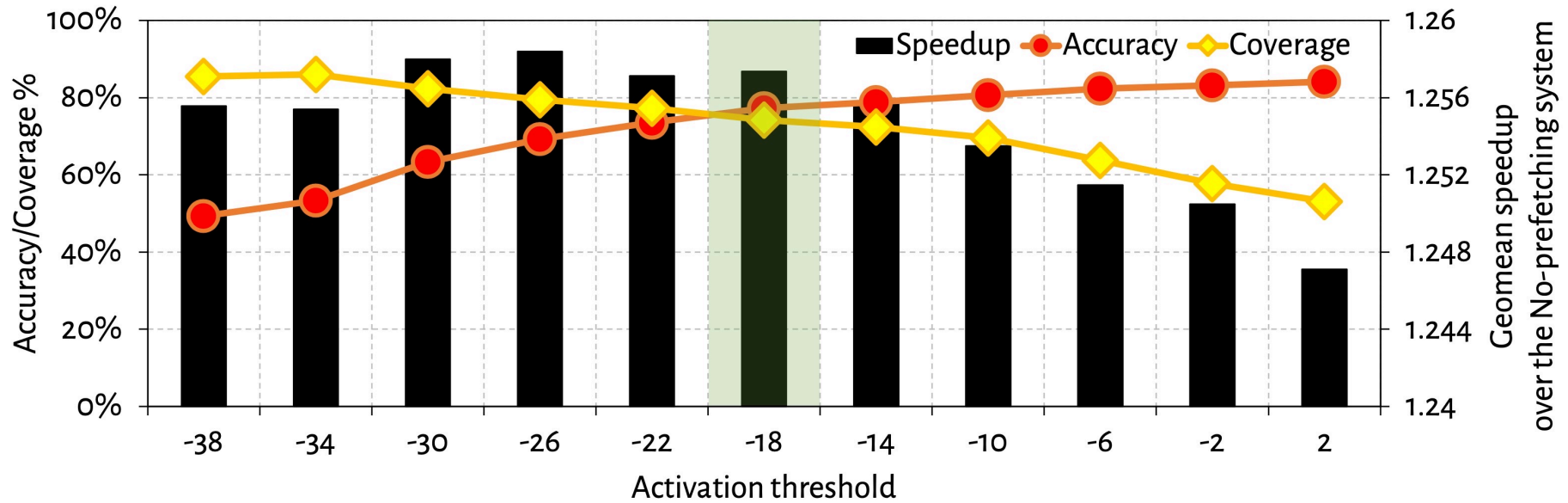
Hermes in combination with Pythia outperforms Pythia alone even with a **24**-cycle Hermes request issue latency

Hermes request issue latency
(in processor cycles)

Effect of Cache Hierarchy Access Latency



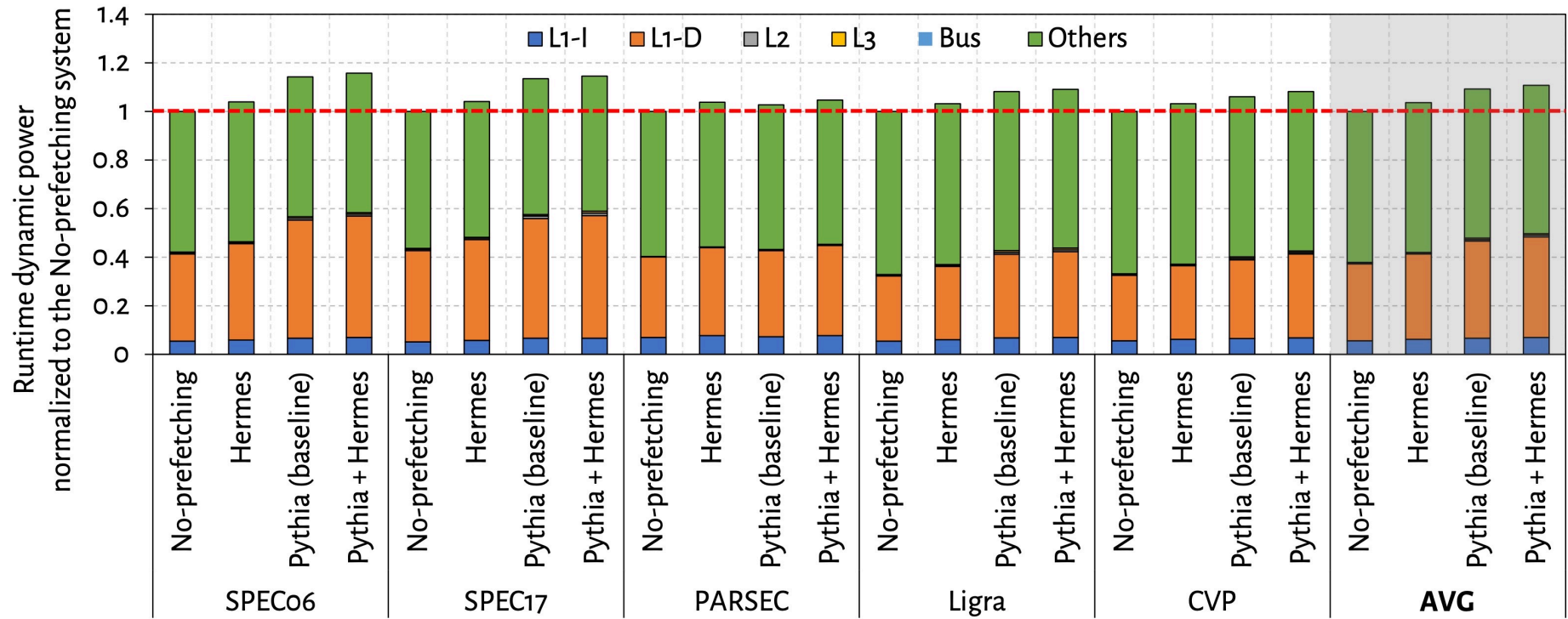
Effect of Activation Threshold



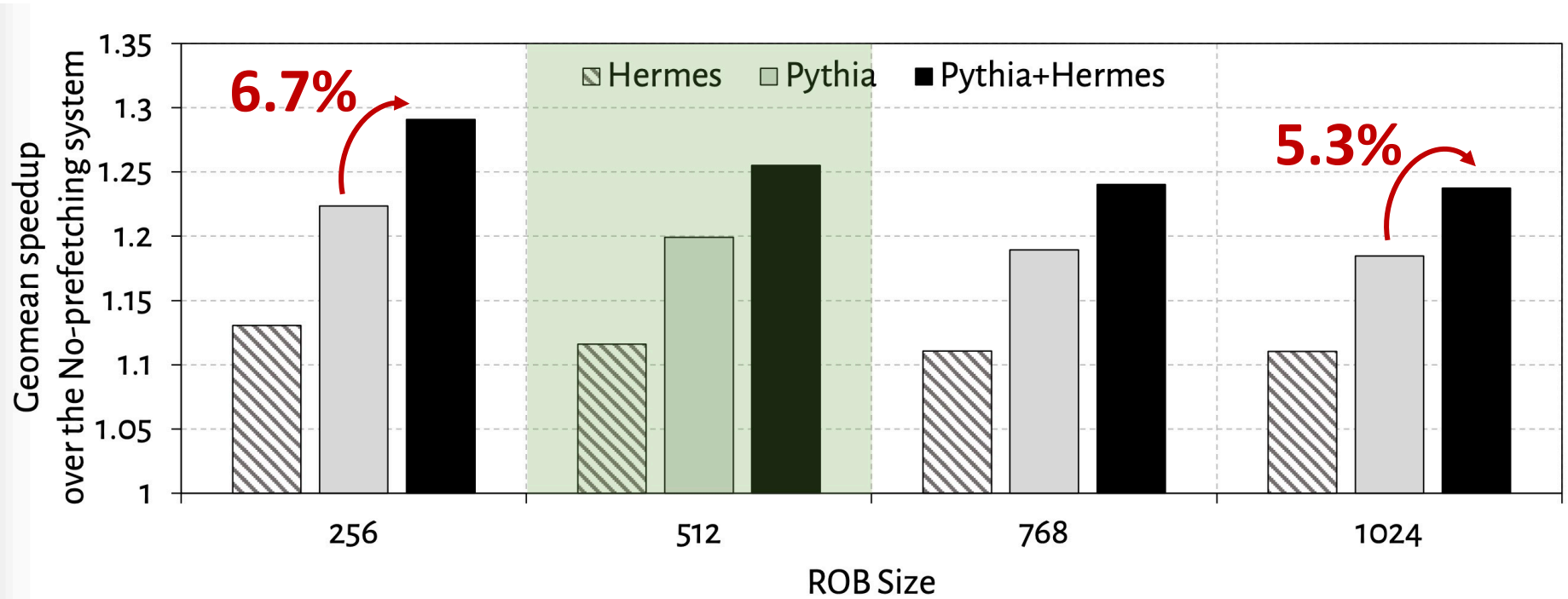
With increase in activation threshold

1. Accuracy increases
2. Coverage decreases

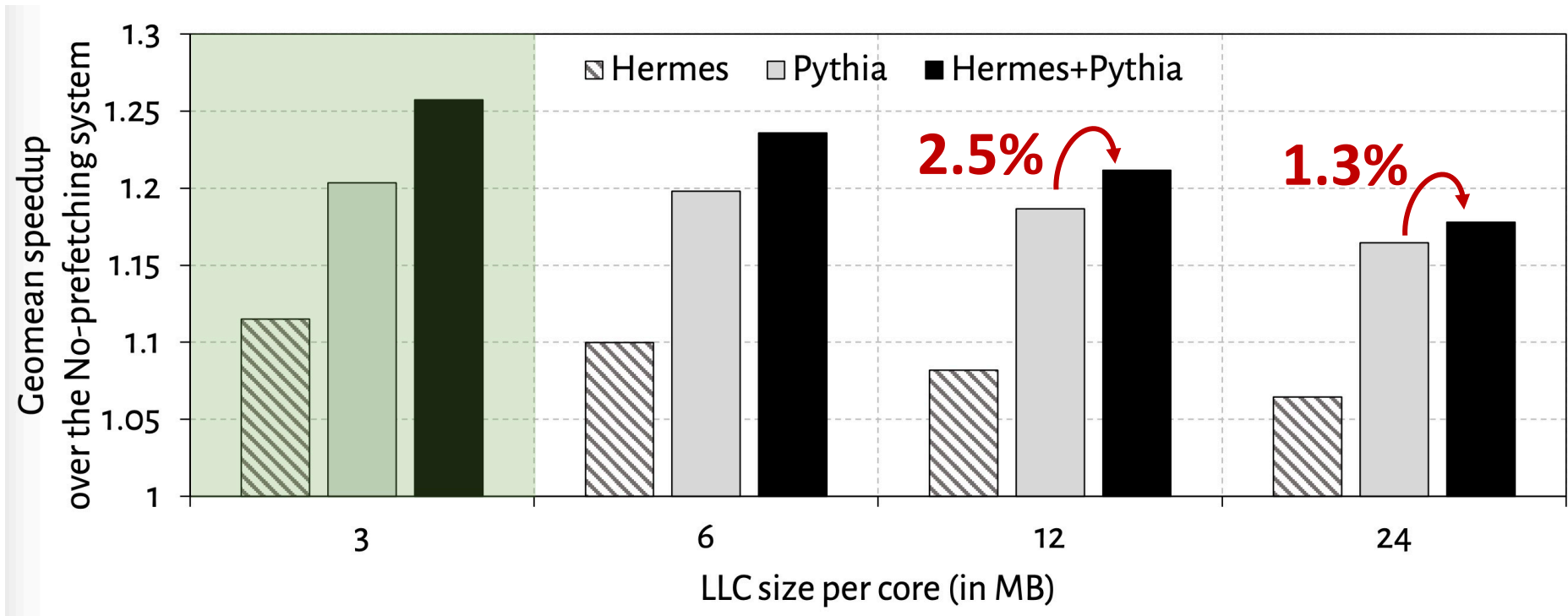
Power Overhead



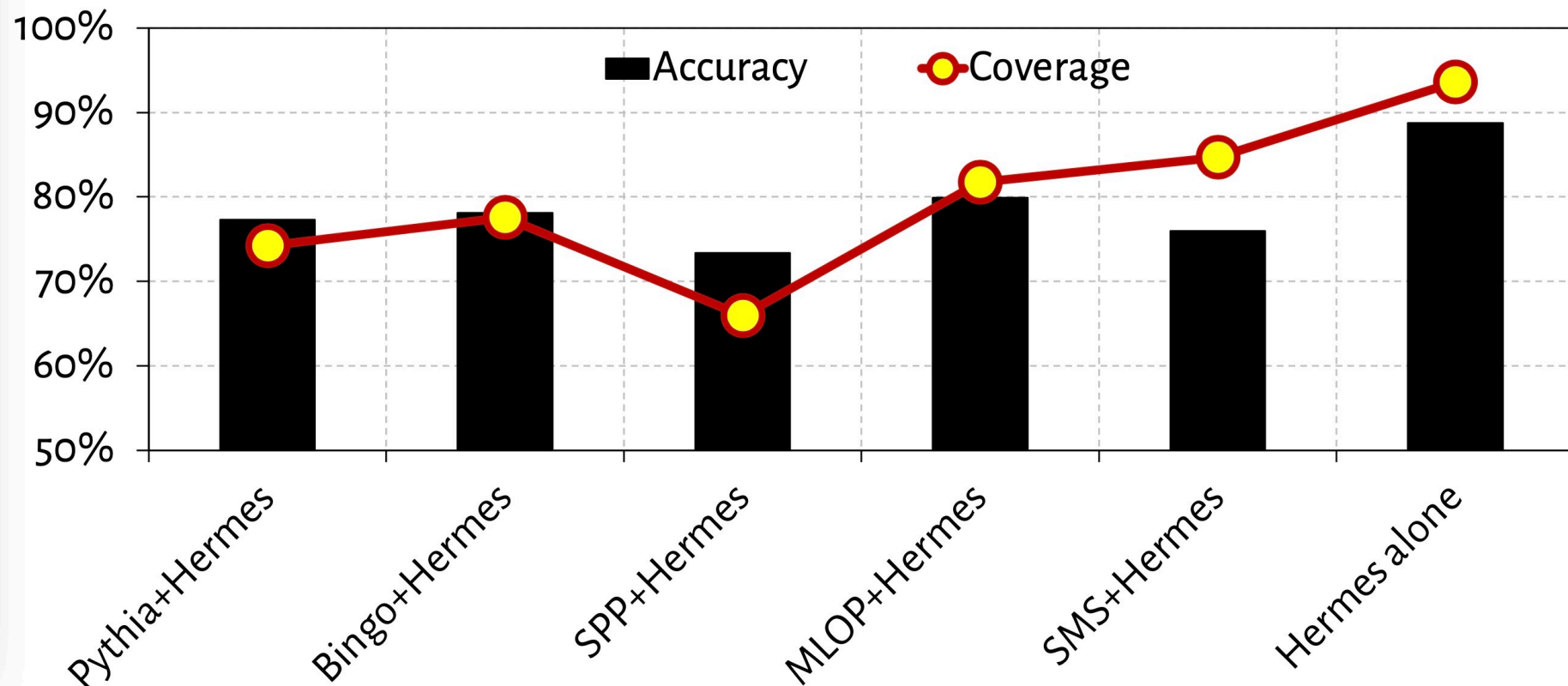
Effect of ROB Size



Effect of LLC Size

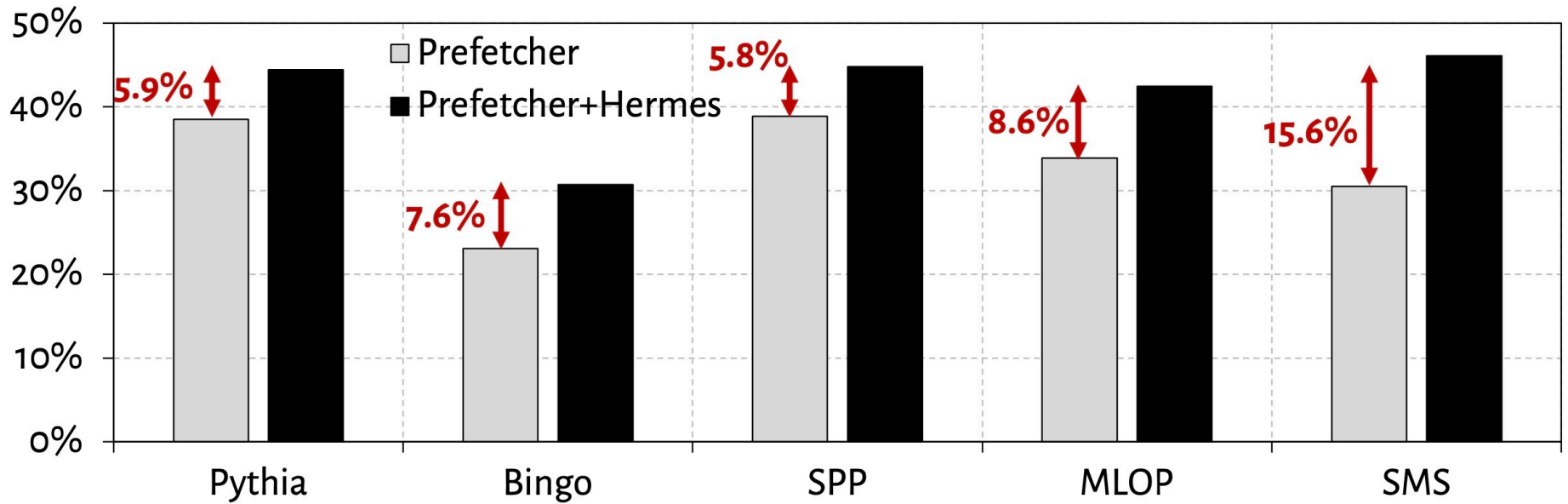


Accuracy and Coverage with Different Prefetchers



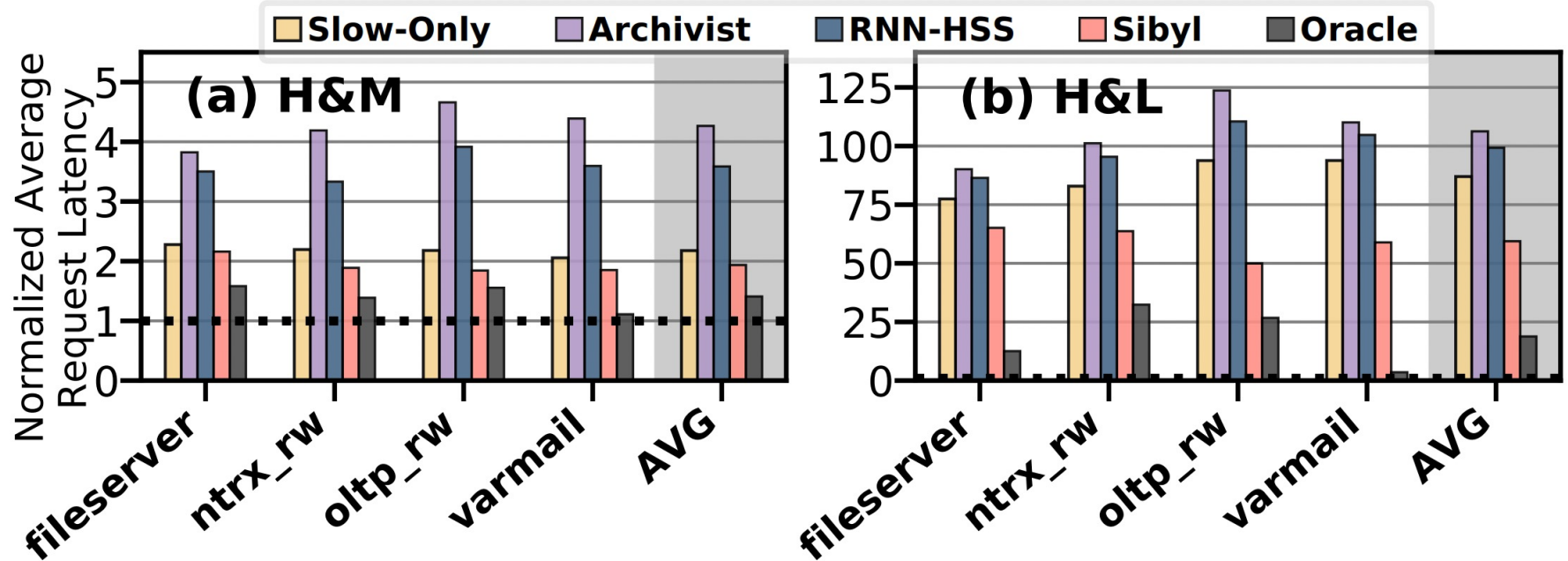
POPET's **accuracy and coverage increases significantly** in absence of a data prefetcher

Increase in Main Memory Requests



SIBYL BACKUP

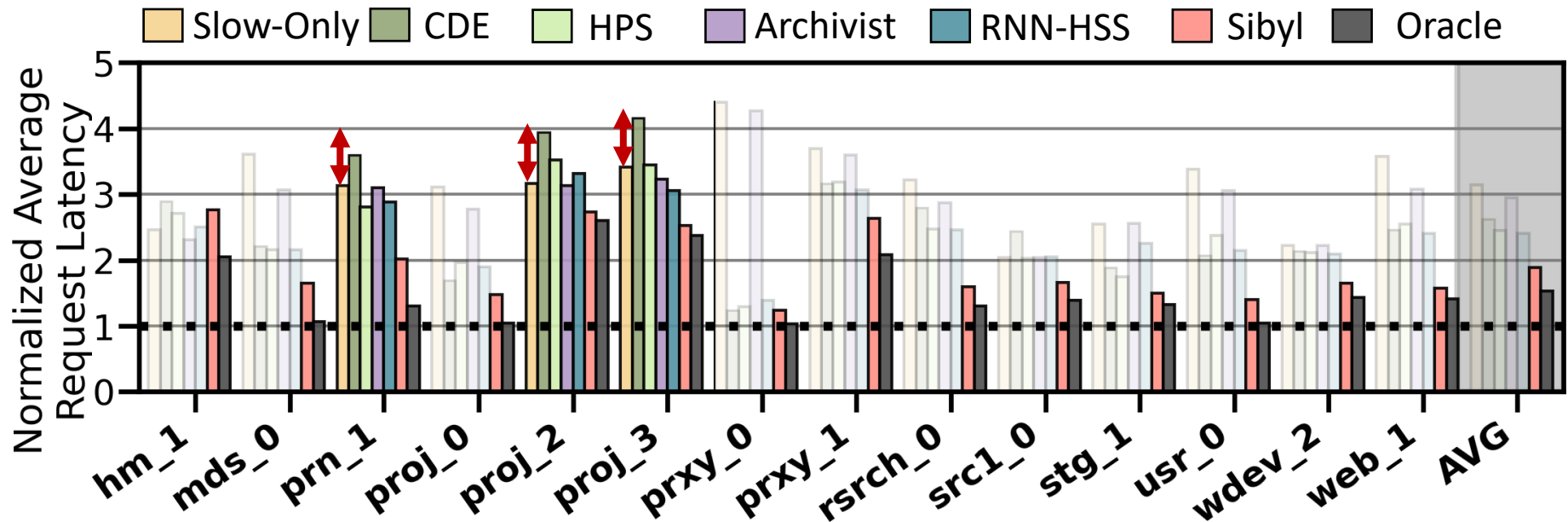
Performance on Unseen Workloads



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

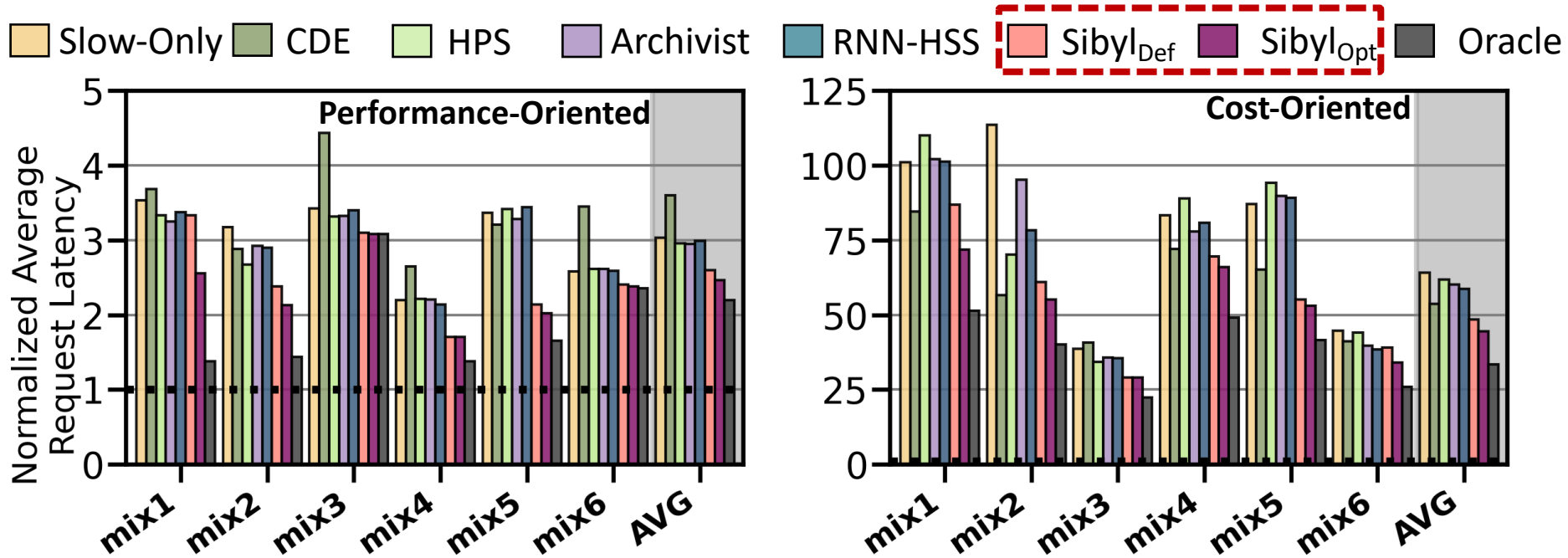
Performance Analysis

Performance-Oriented HSS Configuration

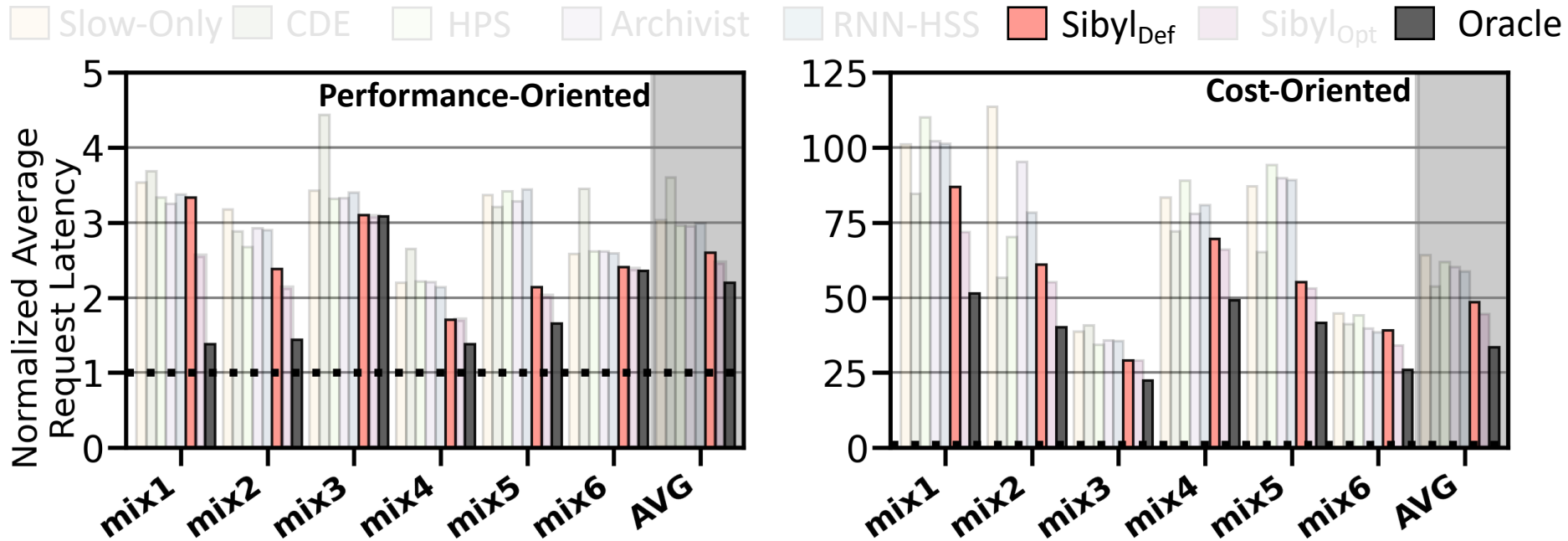


Baseline policies **are ineffective** for many workloads even when compared to Slow-Only

Performance on Mixed Workloads

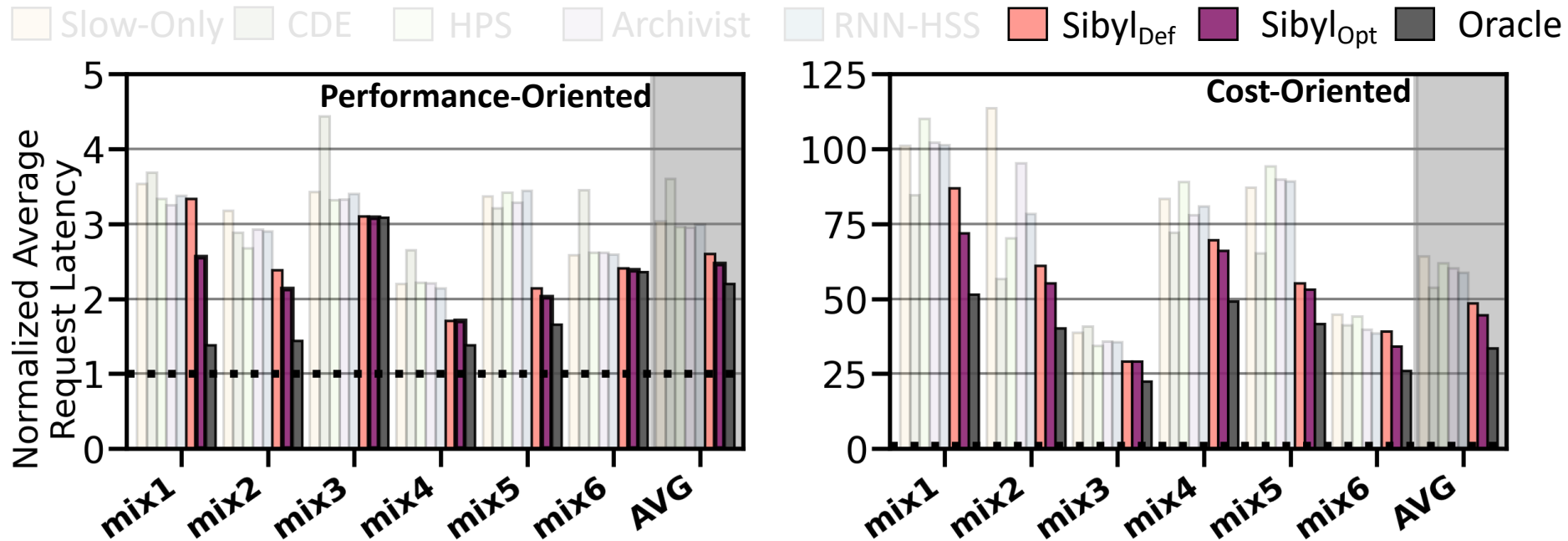


Performance on Mixed Workloads



Sibyl_{Def} **outperforms** baseline data placement techniques by up to **27.9%**

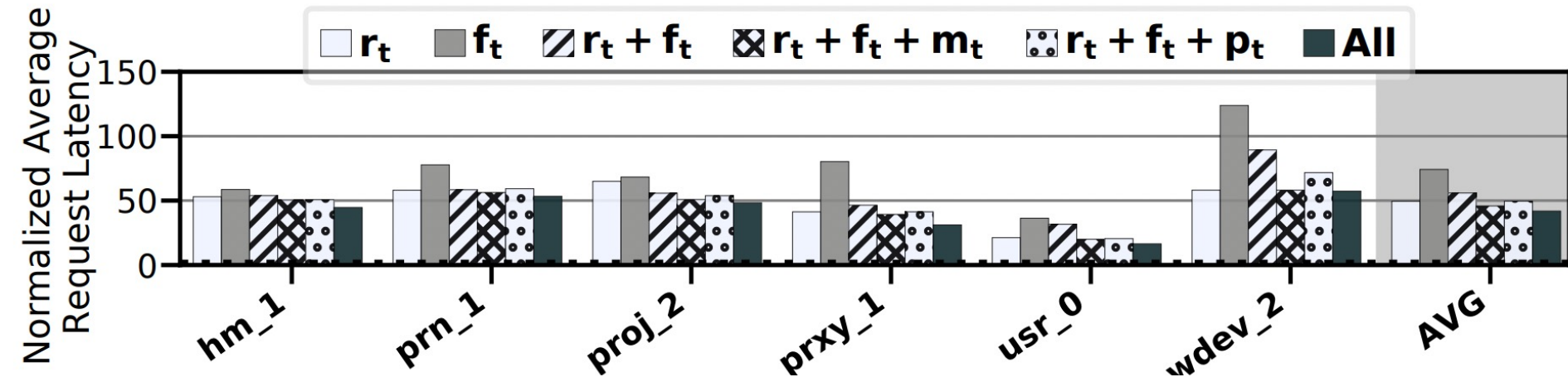
Performance on Mixed Workloads



Sibyl_{Def} **outperforms** baseline data placement techniques by up to **27.9%**

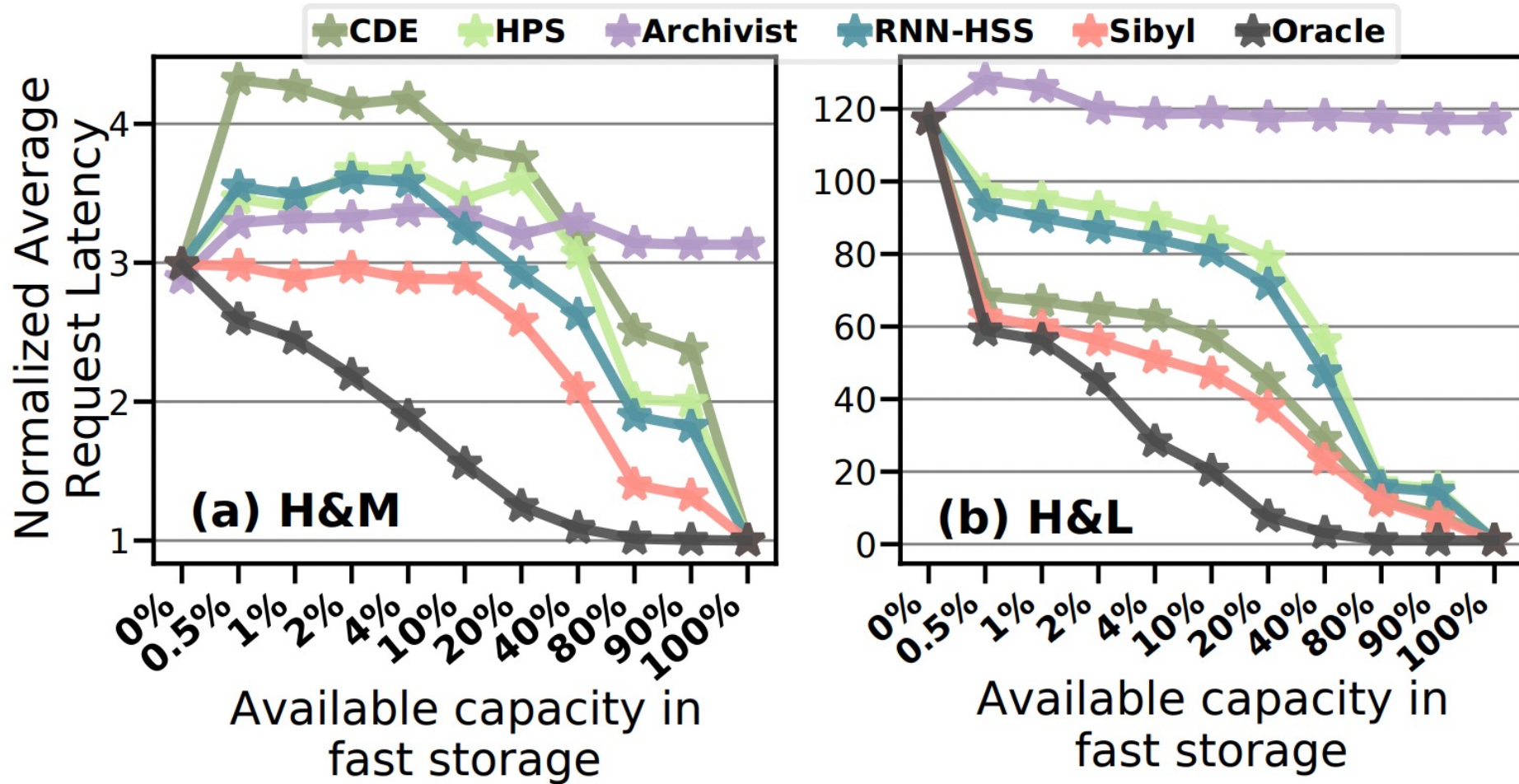
Sibyl_{Opt} **provides 7.2% higher average performance** than Sibyl_{Def}

Performance With Different Features

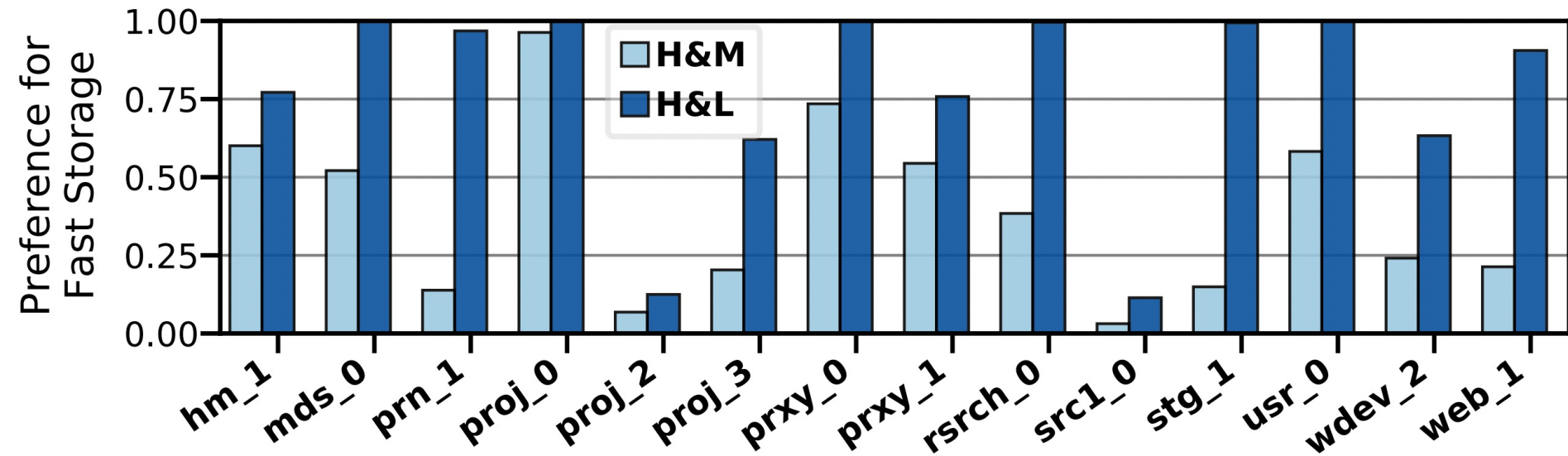


Sibyl autonomously decides which features are important to maximize the performance of the running workload

Sensitivity to Fast Storage Capacity



Explainability Analysis



Training and Inference Network

- Training and inference network **allow parallel execution**
- **Observation vector as the input**
- **Produces probability distribution of Q-values**

