Harmonia

Enhancing Data Placement and Migration in Hybrid Storage Systems via Multi-Agent Reinforcement Learning

Rakesh Nadig, Vamanan Arulchelvan, Rahul Bera, Taha Shahroodi, Gagandeep Singh, Andreas Kosmas Kakolyris, İsmail Emir Yüksel, Mohammad Sadrosadati, Jisung Park and Onur Mutlu

ICS 2026

SAFARI ETH zürich

AMD

POSTECH

Talk Outline

- Background and Motivation
- HSS Data Management using Reinforcement Learning
- Harmonia: Overview
- Evaluation and Key Results
- Conclusion

Talk Outline

Background and Motivation

HSS Data Management using Reinforcement Learning

Harmonia

Evaluation and Key Results

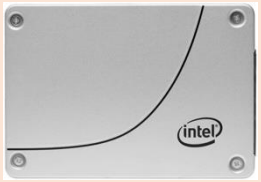
Conclusion

Hybrid Storage System (HSS)

Unified Virtual Address Space

High-End Devices

Mid-Range Devices



Low-End Devices

Higher performance, higher cost/GB

Lower cost/GB, lower performance

Hybrid Storage System (HSS)

An HSS *combines multiple storage devices* with different *performance and cost characteristics* under a *single unified address space*

Hybrid Storage System (HSS)

AI/ML Infrastructure

...

High-Performance Computing (HPC) Clusters

HSS

Edge & Embedded Systems

Enterprise Storage Systems

Factors Affecting HSS Performance

HSS performance depends on two key factors:

Where is the data placed?

Data Placement

Select the best-fit device for each new write

When and where to migrate data?

Data Migration

Reorganize data across devices over time

Factors Affecting HSS Performance

HSS performance depends on two key factors:

Where is the data placed?

Data Placement

Select the best-fit device for each new write

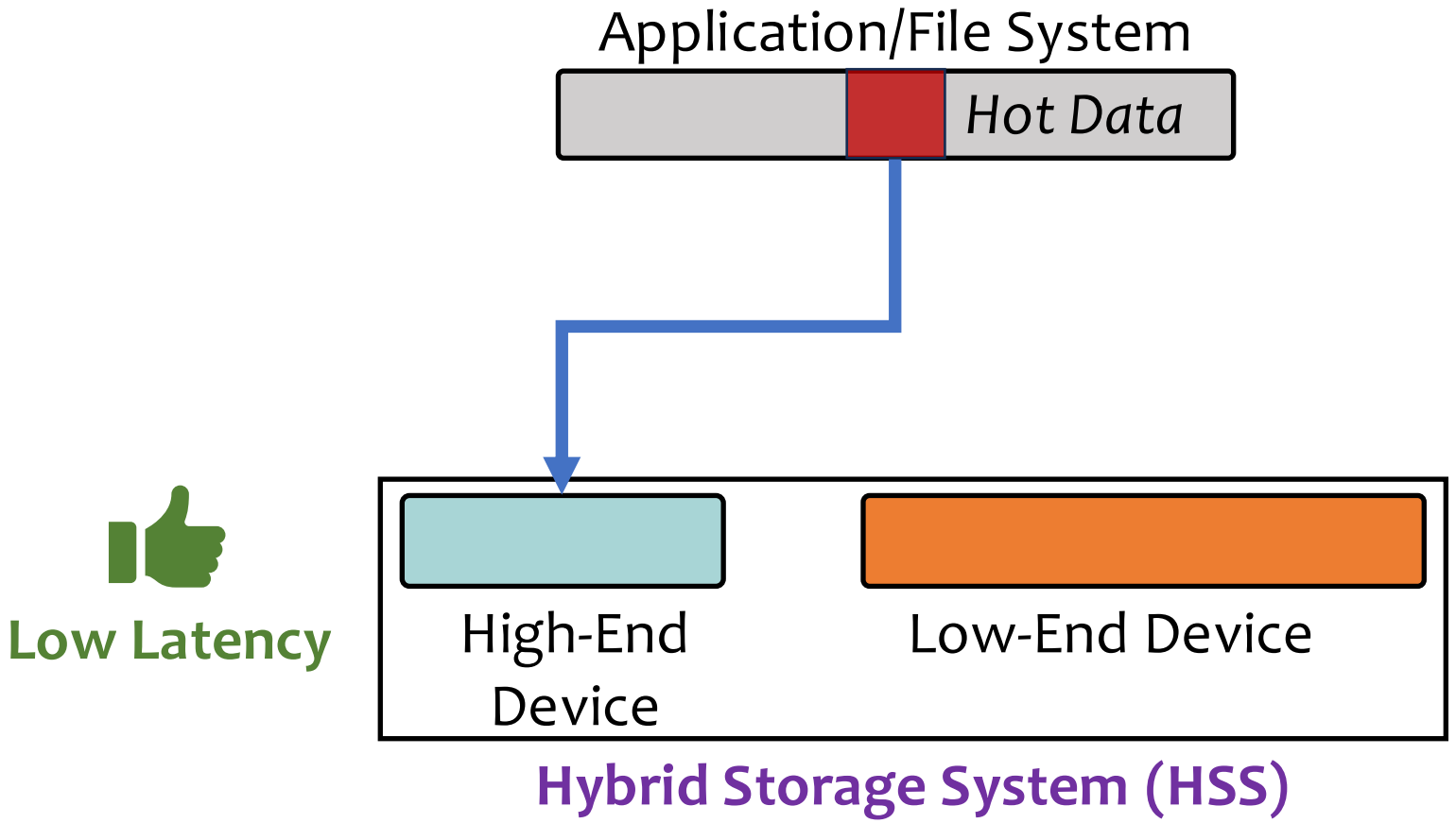
When and where to migrate data?

Data Migration

Reorganize data across devices over time

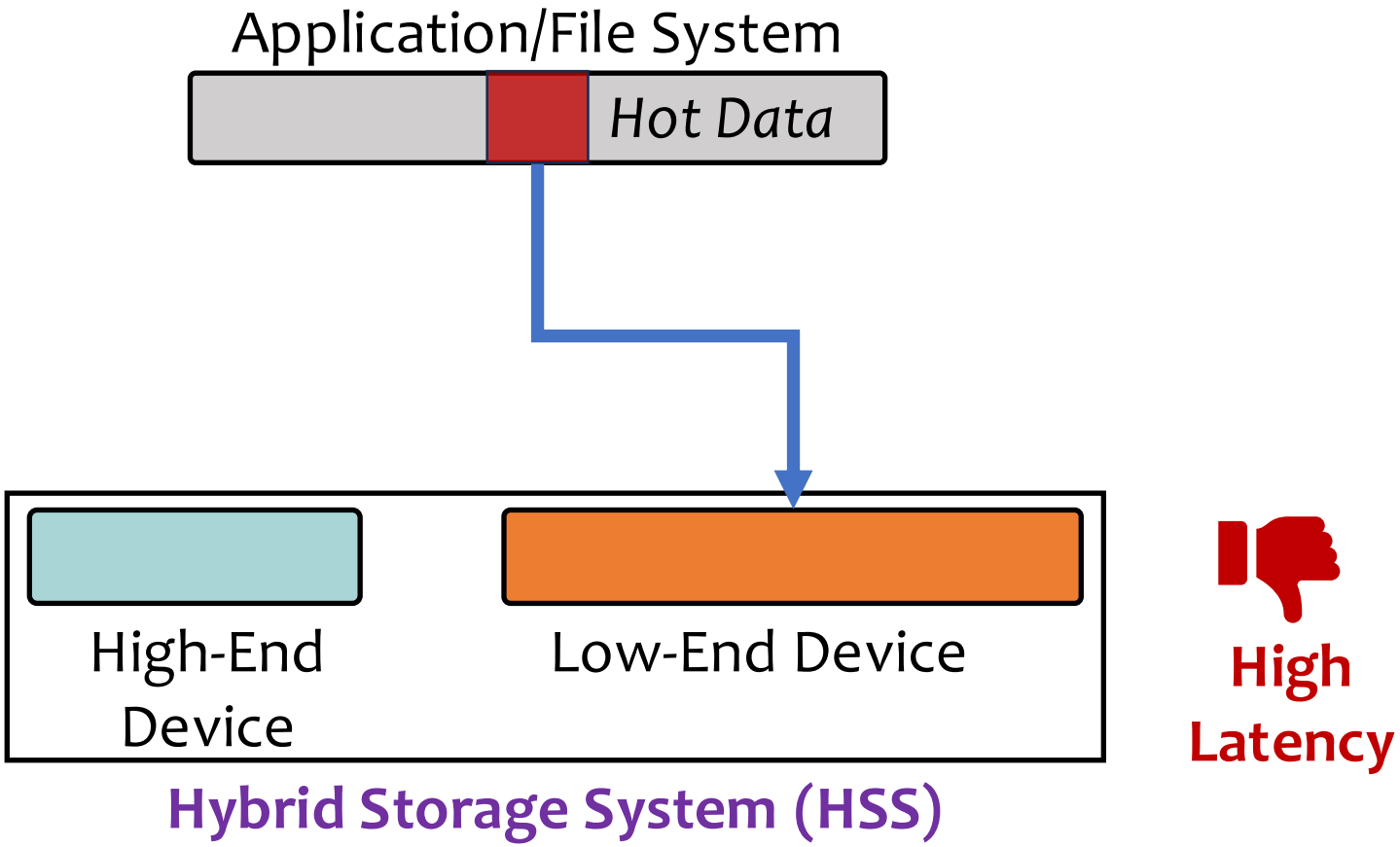
Data Placement

Choose the best-suited HSS storage device for each incoming write or allocation request



Data Placement

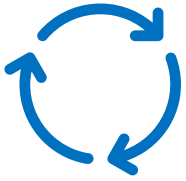
Choosing the best-suitable HSS storage device for each incoming write or allocation request



Data Placement: Key Requirements

Need to operate with *low latency* on the *critical path* of *I/O handling*

Make accurate, yet *low-overhead* decisions



Adapt to changes in workload and system conditions

Factors Affecting HSS Performance

HSS performance depends on two key factors:

Where is the data placed?

↓

Data Placement

Select the best-fit device for each new write

When and where to migrate data?

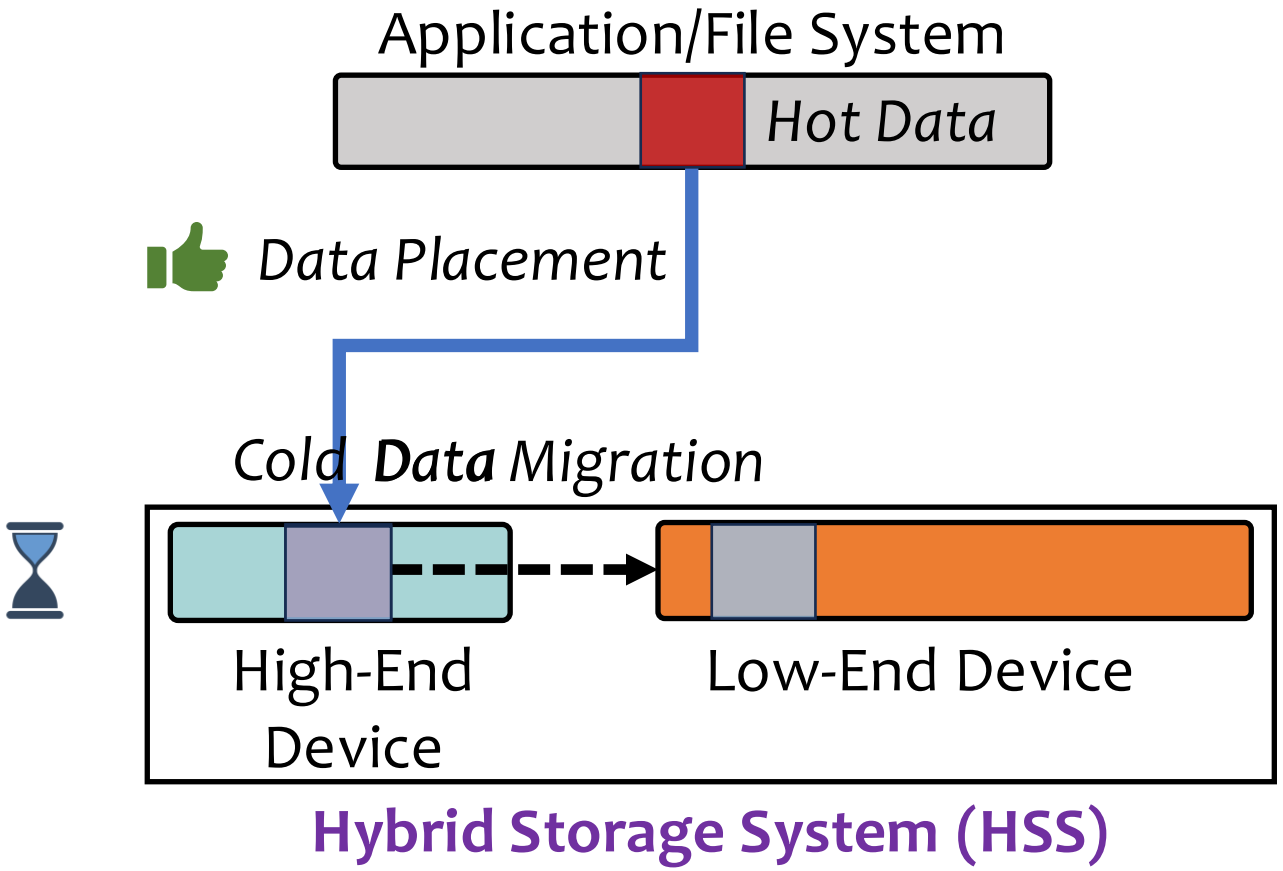
↓

Data Migration

Reorganize data across devices over time

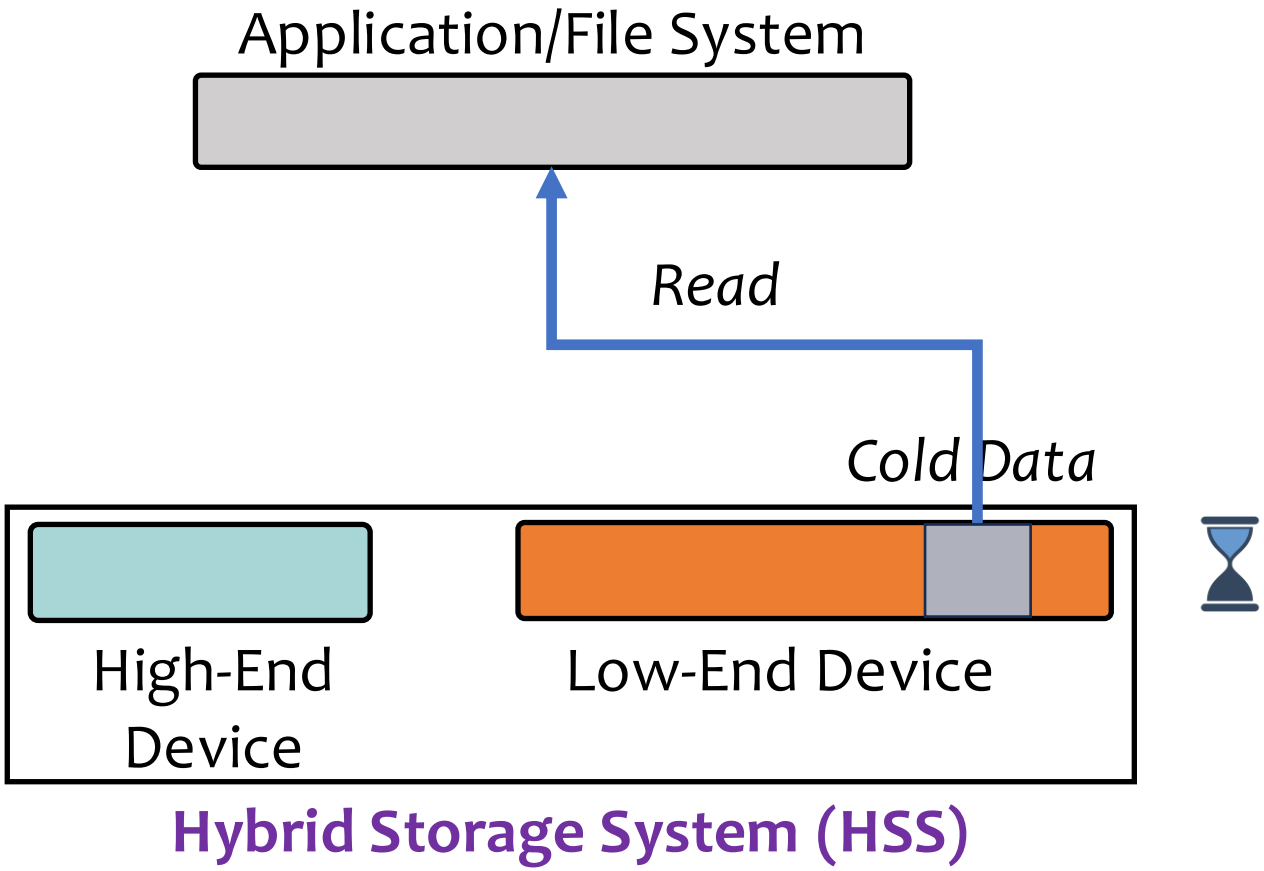
Data Migration

Reorganize data across HSS devices (e.g., prefetch hot data to the high-end devices and evict cold data to the low-end devices) to sustain high HSS performance



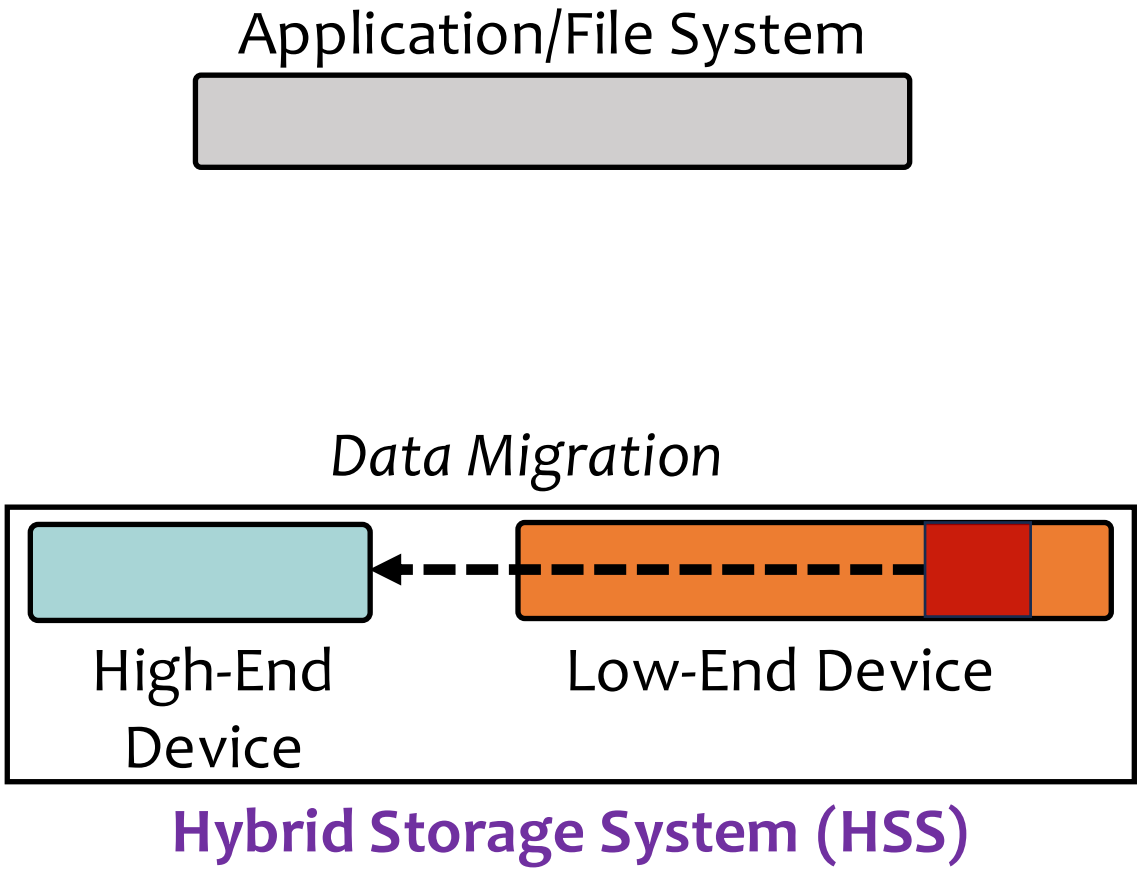
Data Migration

Reorganize data across HSS devices (e.g., prefetch hot data to the faster devices and evict cold data to the slower devices) to sustain high HSS performance



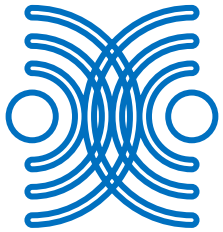
Data Migration

Reorganize data across HSS devices (e.g., prefetch hot data to the faster devices and evict cold data to the slower devices) to sustain high HSS performance



Data Migration: Key Requirements

Need to operate in the background, off the critical path of I/O handling



Minimize interference with foreground I/O operations

Perform *timely migrations* to *adapt* to *workload and system changes*

Key Limitations of Prior Techniques

Prior HSS techniques *do not optimize* both data placement and migration

Naive combination of prior placement and migration techniques provides *sub-optimal performance* due to *lack of coordination*

Lack extensibility to more devices and configurations

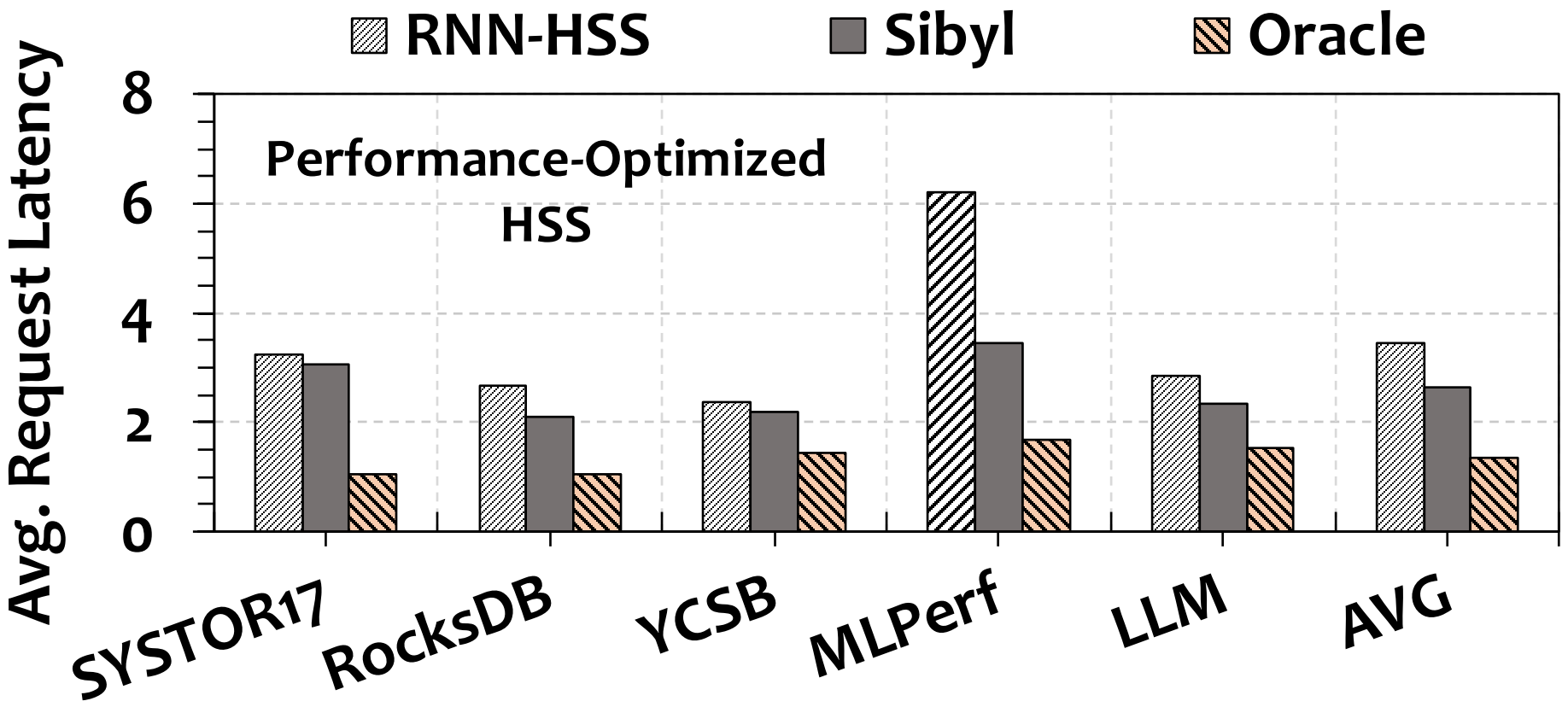
Key Limitations of Prior Techniques

Prior HSS techniques *do not optimize* both data placement and migration

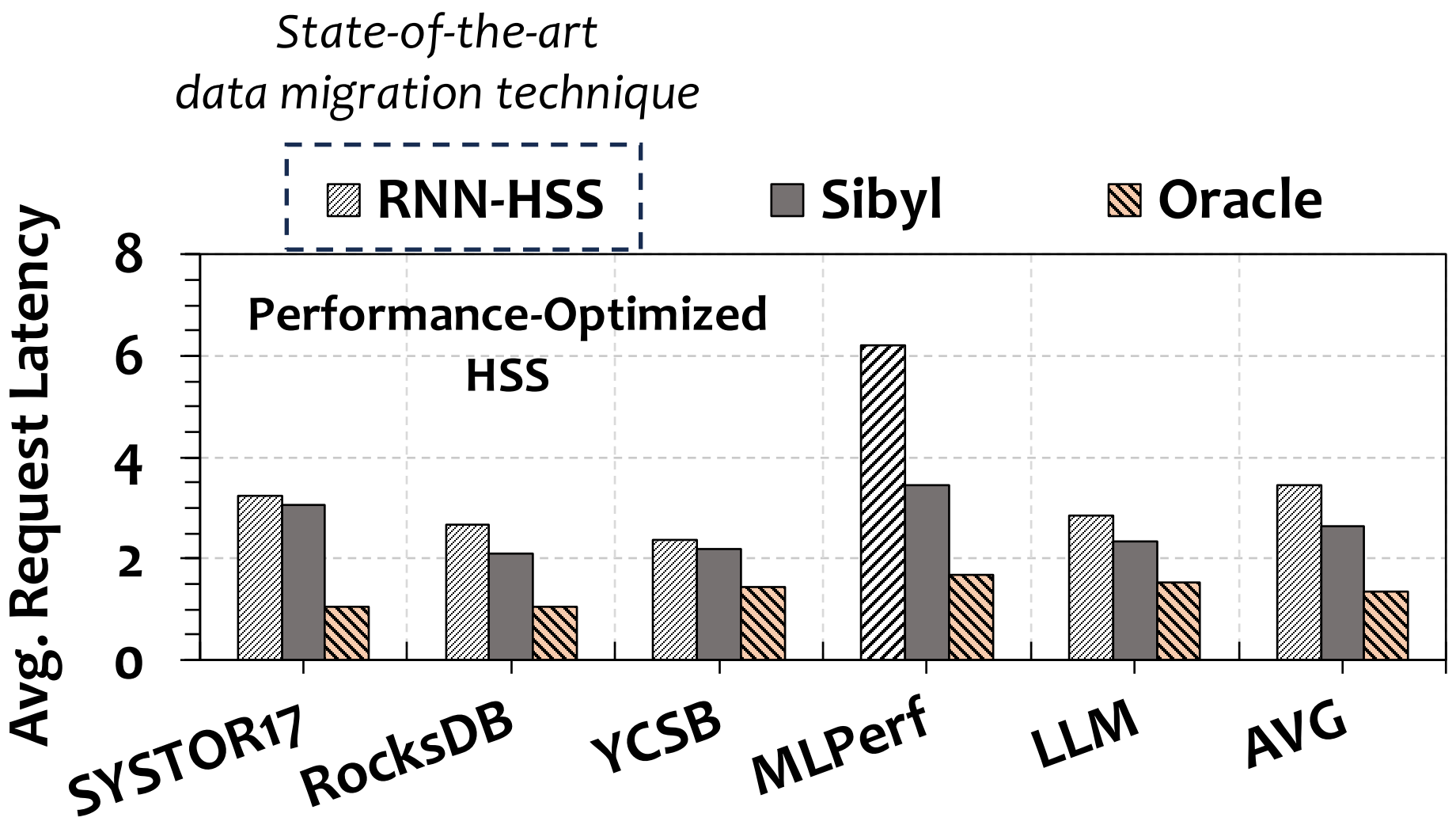
Naive combination of prior placement and migration techniques provides *sub-optimal performance* due to *lack of coordination*

Lack extensibility to more devices and configurations

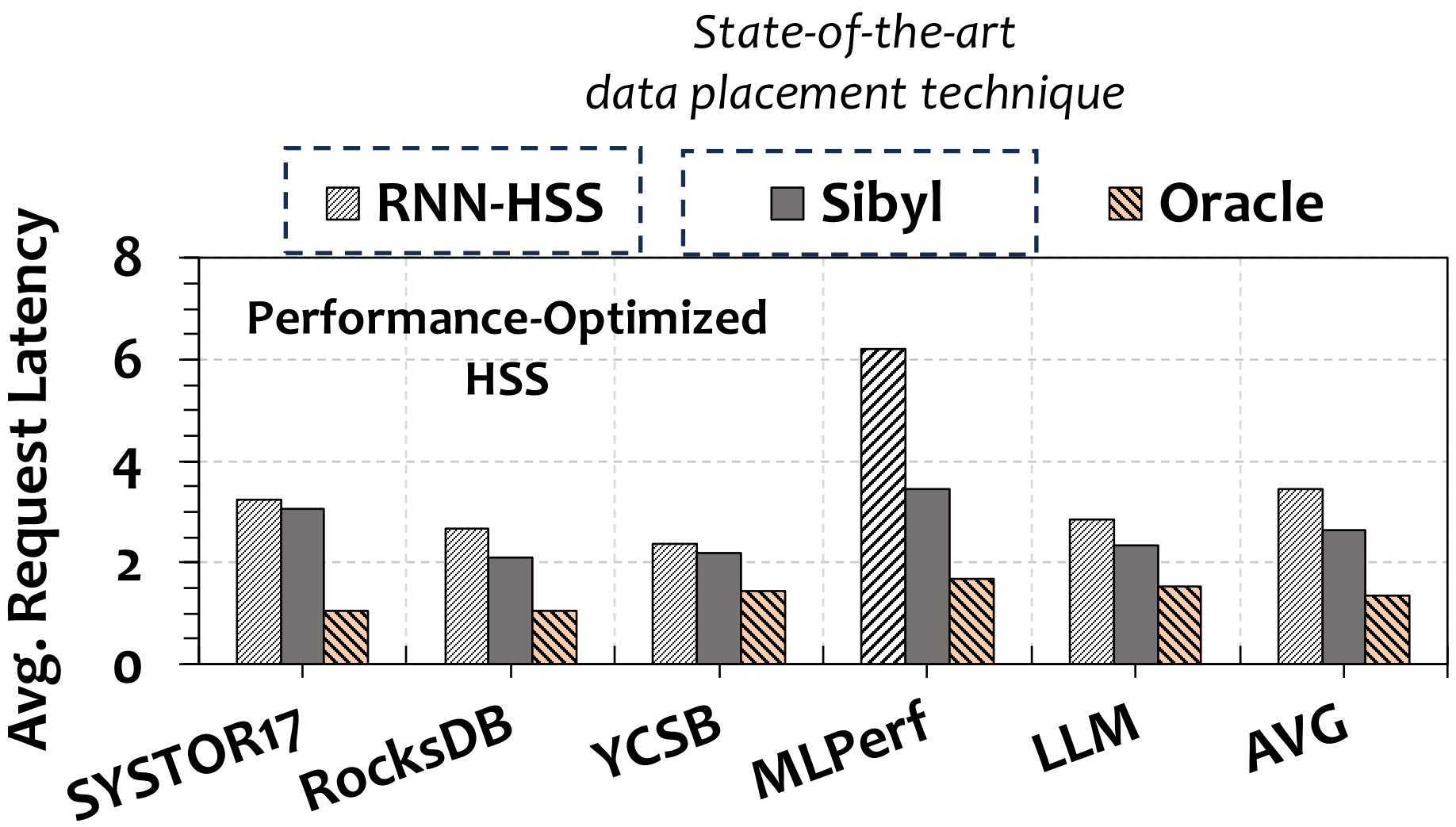
Lack of Holistic Data Management in HSS



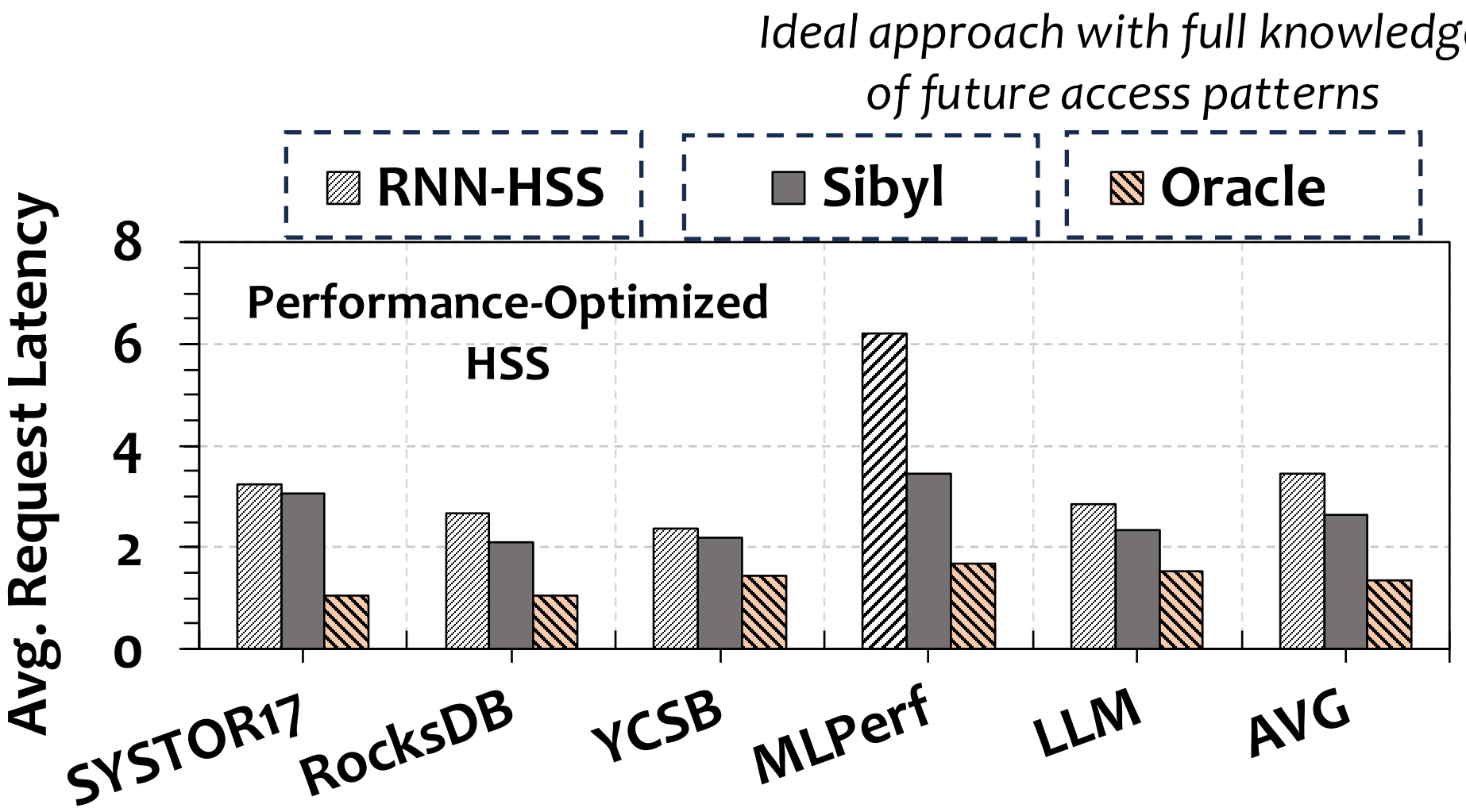
Lack of Holistic Data Management in HSS



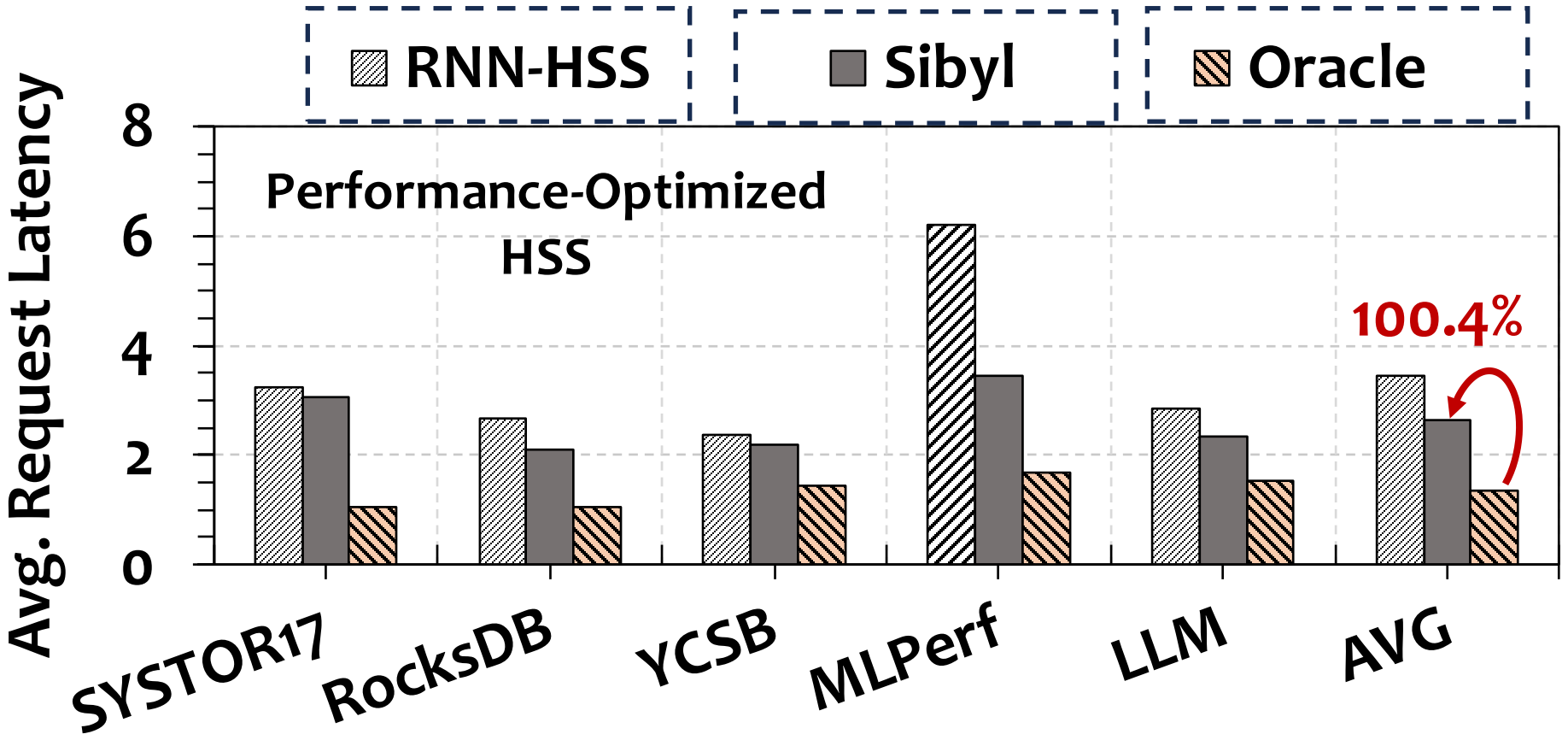
Lack of Holistic Data Management in HSS



Lack of Holistic Data Management in HSS



Lack of Holistic Data Management in HSS



Lack of Holistic Data Management in HSS

The *large latency gap* between Sibyl and Oracle is due to Sibyl's *heuristic migration policy* which does not *adapt to workload and system changes*

The *latency gap becomes wider* in a *cost-optimized HSS* where Sibyl's heuristic migration policy performs *costly data evictions*

Key Limitations of Prior Techniques

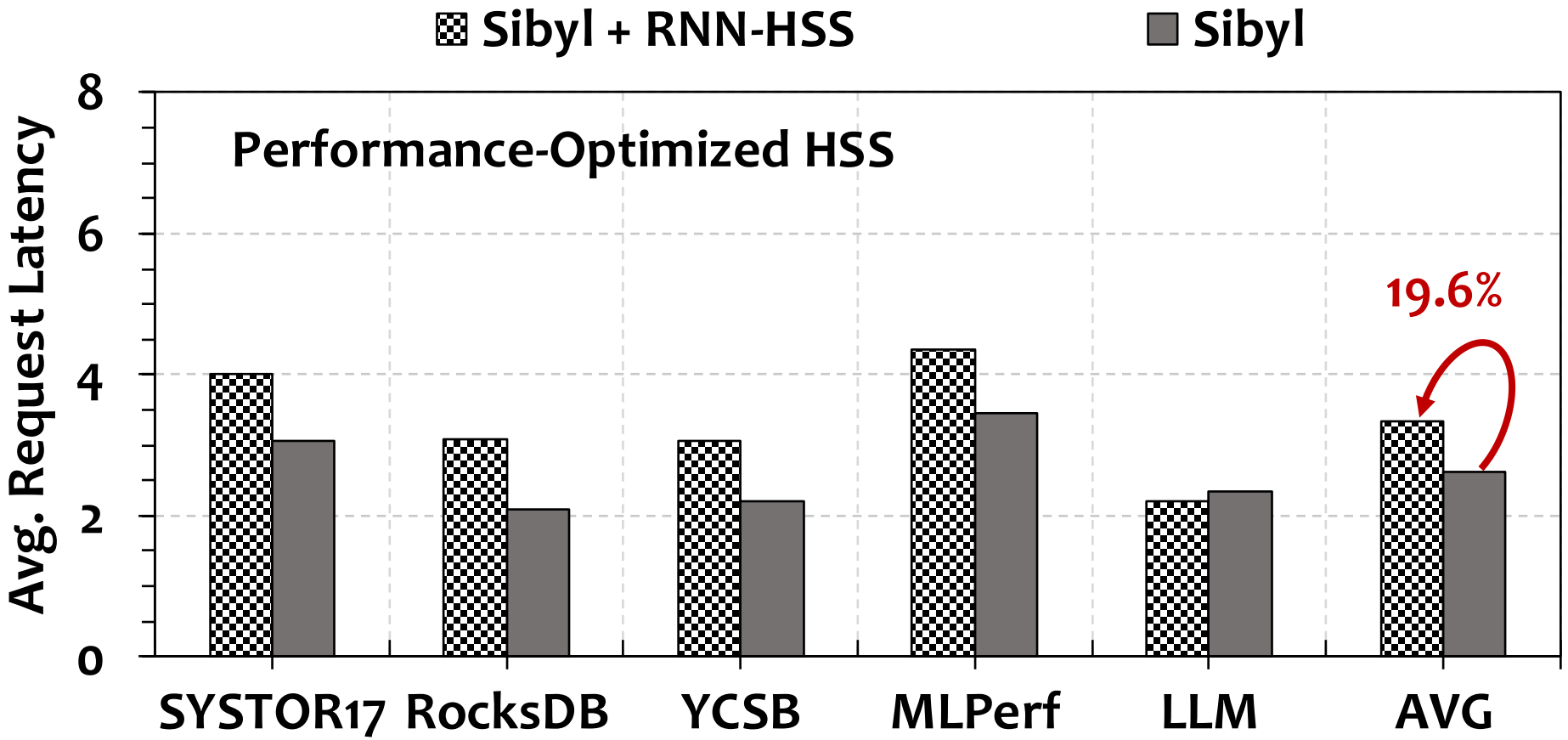
Prior HSS techniques *do not optimize* both data placement and migration

Naive combination of prior placement and migration techniques provides *sub-optimal performance* due to *lack of coordination*

Lack extensibility to more devices and configurations

Lack of Coordination in Extended Techniques

- What happens if we combine the state-of-the art placement and migration techniques?



Lack of Coordination in Extended Techniques

- What happens if we combine the state-of-the art placement and migration techniques?

Naive combinations provide *sub-optimal performance* because the constituent policies are oblivious to each other's decisions

The extended techniques make *conflicting placement and migration decisions* due to *lack of coordination*

Key Limitations of Prior Techniques

Prior HSS techniques *do not optimize* both data placement and migration

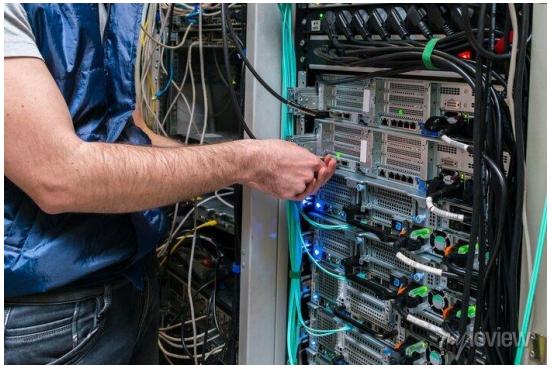
Naive combination of prior placement and migration techniques provide *sub-optimal performance* due to *lack of coordination*

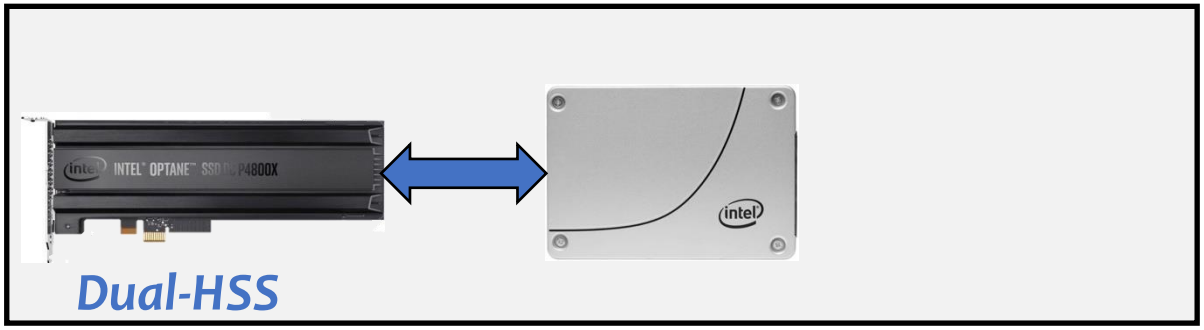
Lack extensibility to more devices and configurations

Lack of Extensibility (1/2)

Prior HSS data-management techniques are rigid, and require significant designer effort to extend the policies to more devices

Changes in HSS configuration

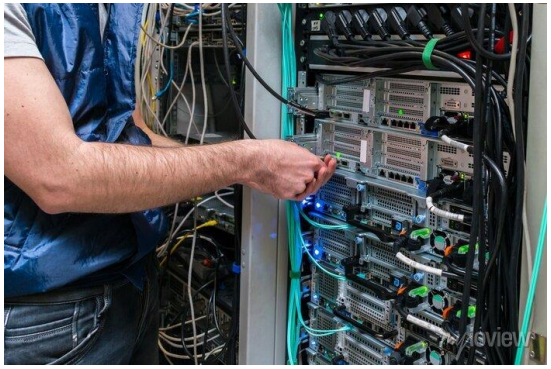




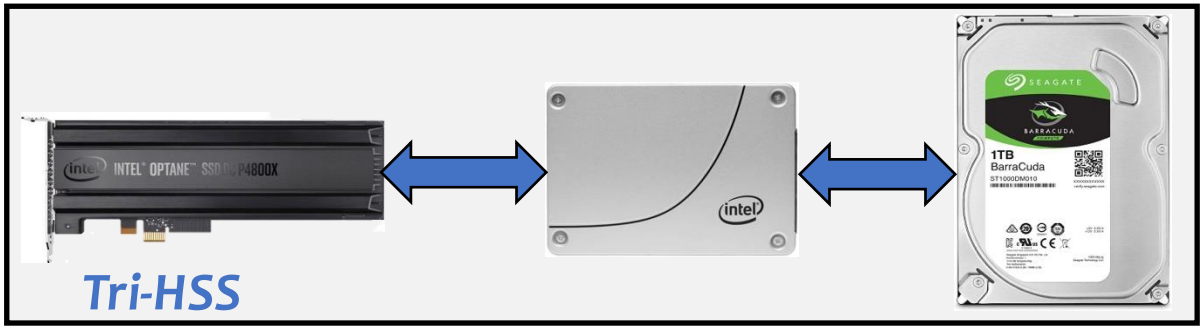
Lack of Extensibility (2/2)

Prior HSS-data management techniques are rigid, and require significant designer effort to extend the policies to more devices

Changes in HSS configuration



Design a new policy



Our Goal

A **holistic data-management mechanism** that

1. ***Performs combined optimization*** of both data placement and data migration policies

2. ***Achieves coordination*** between data placement and data migration policies

3. ***Provides easy extensibility*** to incorporate a wide range of HSS configurations

Harmonia

An *online multi-agent reinforcement learning* technique for HSS data management

Harmonia consists of *two autonomous, coordinating RL agents* that co-optimize *data placement and data migration*

Harmonia is the Greek goddess of harmony and balance
https://en.wikipedia.org/wiki/Harmonia

Talk Outline

Background and Motivation

HSS Data Management using Reinforcement Learning

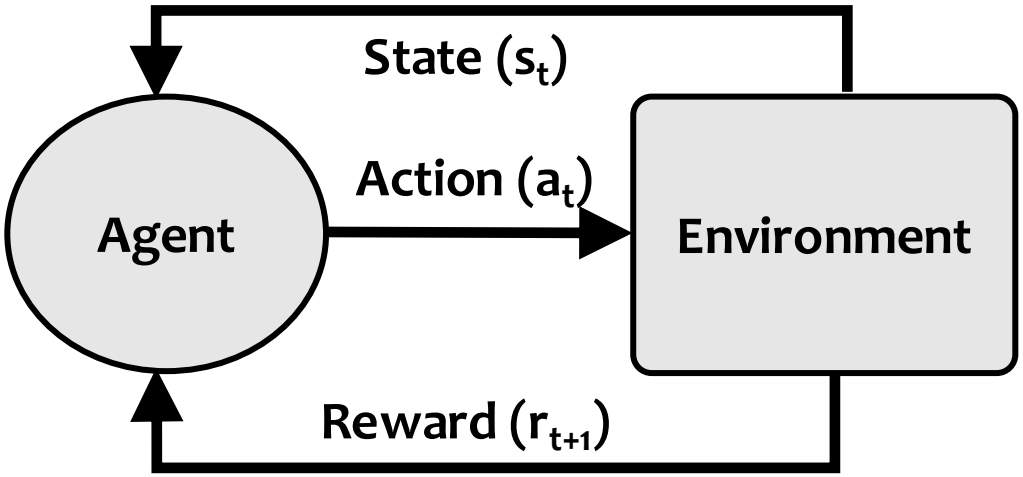
Harmonia

Evaluation and Key Results

Conclusion

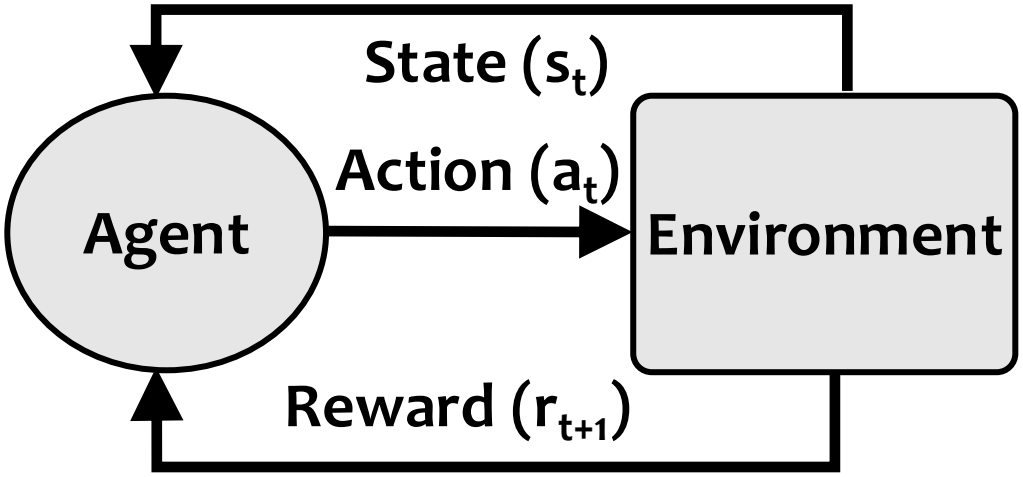
Basics of Reinforcement Learning (RL)

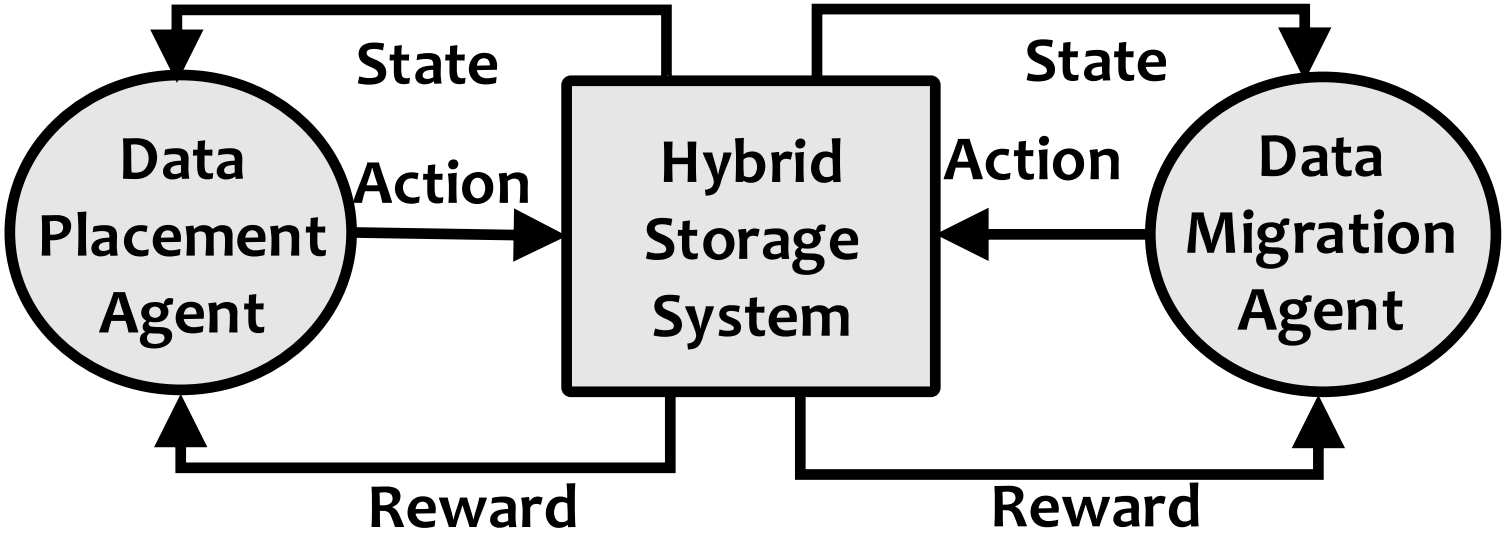
- An algorithmic approach where an RL agent learns to select the best **action** for the current state to maximize long-term **reward**



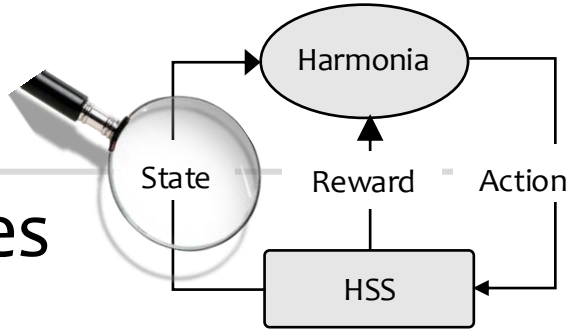
- Agent stores **Q-values** for *every* state-action pair
  - **Expected return** for taking an action in a state

Formulating HSS Data Management using RL





Harmonia's State

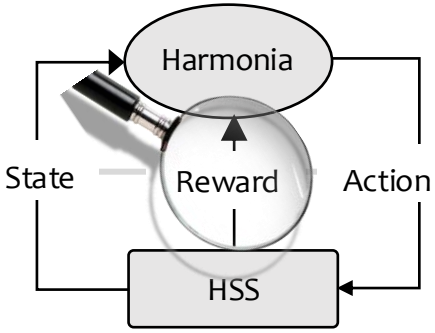


- **7-dimensional** vector of state features

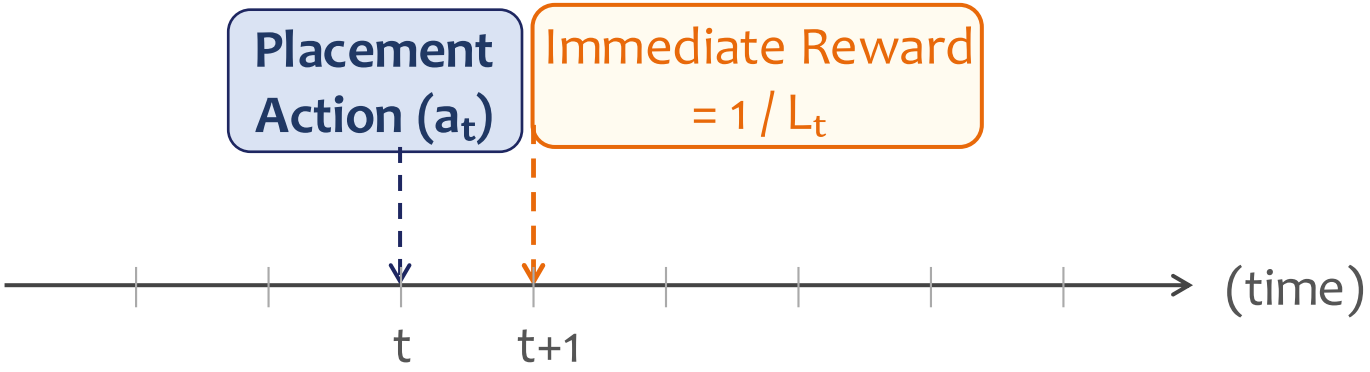
✓ Request Size	✓ Free Space in the Fast Device
✓ Request Type	✓ Current Location of a Page
✓ Access Interval of a Page	✓ Migration Interval of a Page
✓ Access Frequency of a Page	

- **State features common for both agents**
  - Captures workload changes and system conditions
- **Limited number of state features**
  - Reduce the implementation overhead
  - RL agent is more sensitive to reward
- We **quantize the state features** into bins to reduce storage overhead

Harmonia's Reward (1/2)

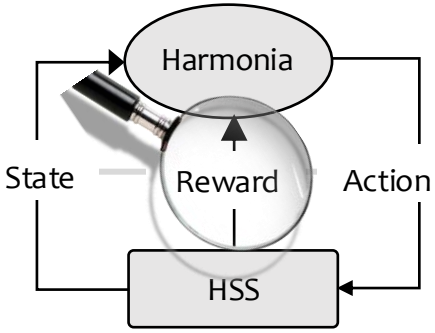


- Defines Harmonia's objective
- Data Placement Agent
  - Performs data placement on the critical path of I/O handling
  - **Objective**: Reduce average I/O request latency of current request

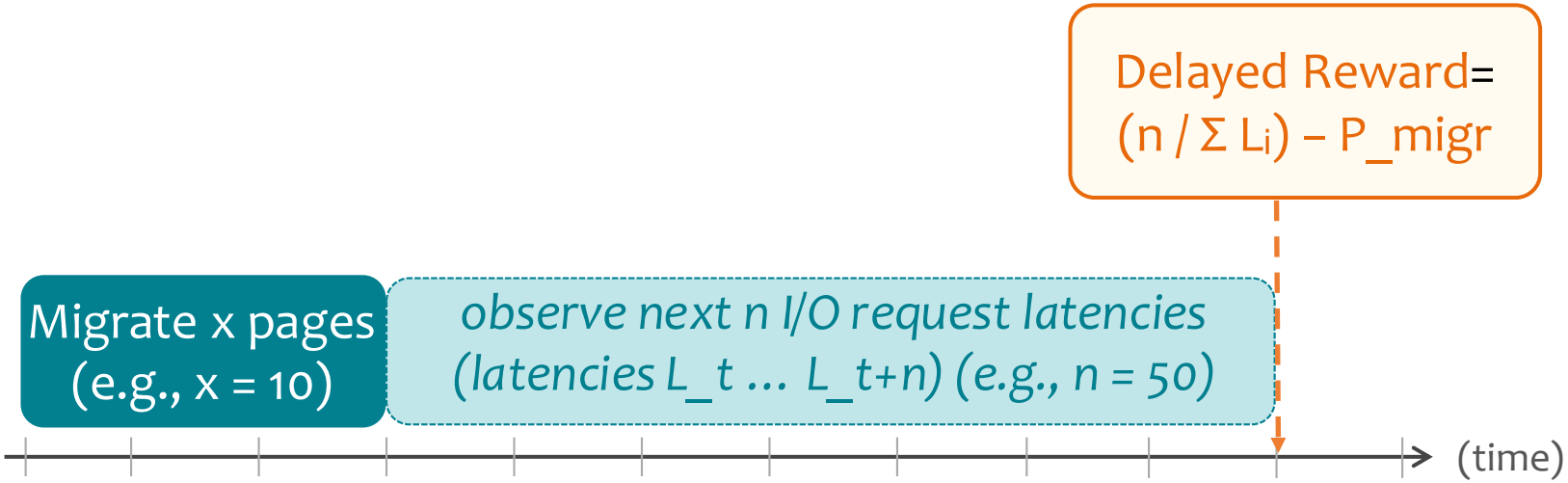


- Harmonia learns to balance device utilization
  - Overutilization of high-end device leads to higher latency due to contention, queuing delays and garbage collection
  - Selectively placing less-critical data on low-end devices preserves high-end device capacity

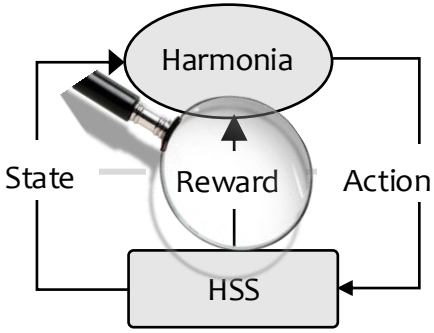
Harmonia's Reward (2/2)



- Data Migration Agent
  - Runs in the background
  - **Objective:** Maximizes long-term HSS performance by reorganizing data across HSS devices



Harmonia's Reward (2/2)



- Data Migration Agent
  - Runs in the background
  - **Objective:** Maximizes long-term HSS performance by reorganizing data across HSS devices

Application may not immediately access prefetched data

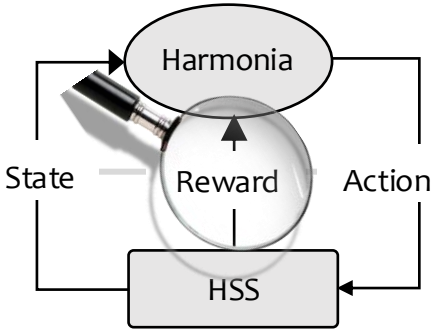
Delayed Reward= $(n / \Sigma L_i) - P_migr$

Migrate x pages (e.g., x = 10)

observe next n I/O request latencies (latencies L_t … L_t+n) (e.g., n = 50)

(time)

Harmonia's Reward (2/2)



- Data Migration Agent
  - Runs in the background
  - **Objective:** Maximizes long-term HSS performance by reorganizing data across HSS devices

Performance gains from freed high-end device capacity is seen on future placements

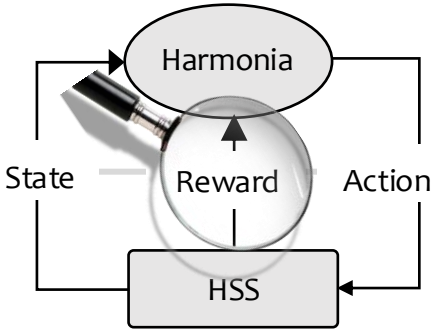
Delayed Reward= $(n / \Sigma L_i) - P_migr$

Migrate x pages (e.g., x = 10)

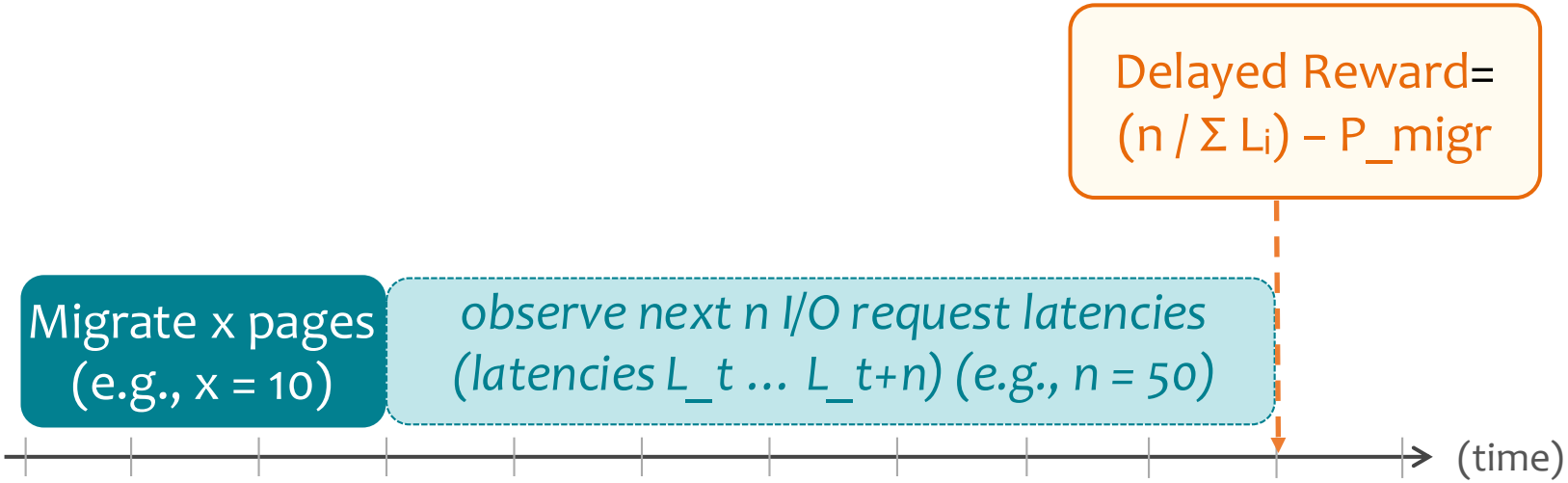
observe next n I/O req[illegible] latencies (latencies L_t … L_t+n) (e.g., [illegible] = 50)

(time)

Harmonia's Reward (2/2)

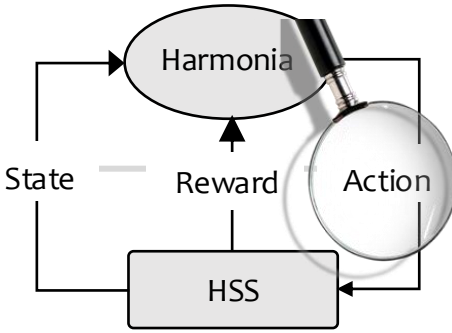


- Data Migration Agent
  - Runs in the background
  - **Objective:** Maximizes long-term HSS performance by reorganizing data across HSS devices



- Latency encapsulates the device's internal state
  - (e.g., read/write latencies, garbage collection latency, queuing delays, …)

What is Action?



- Data Placement
  - For every new I/O request, the action is to **select a storage device to place** the I/O request data
- Data Migration
  - For every page already placed in HSS, **select the storage device to migrate** the data
- Action can be **easily extended** to any number of storage devices

Talk Outline

Background and Motivation

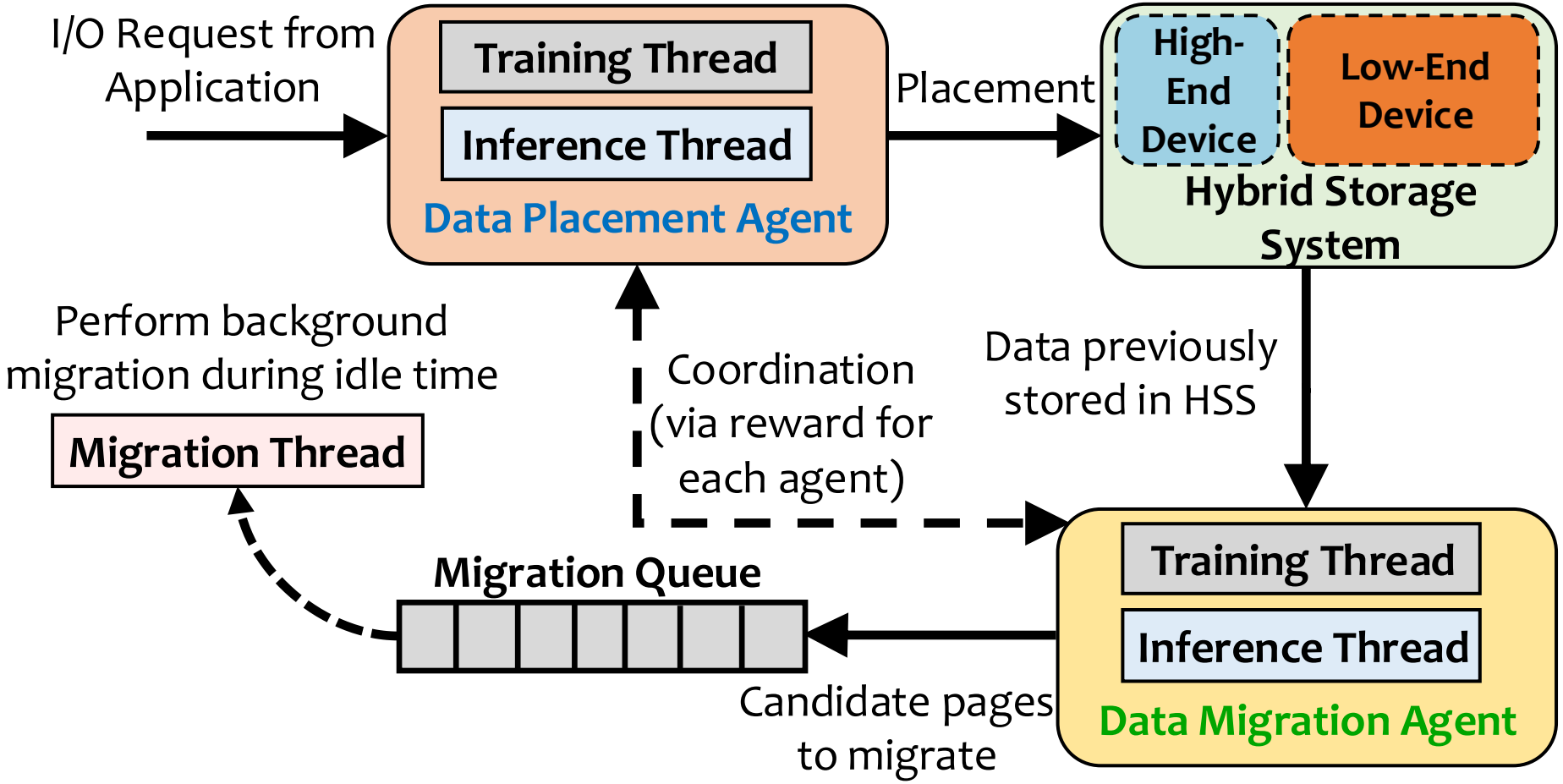
HSS Data Management using Reinforcement Learning

Harmonia

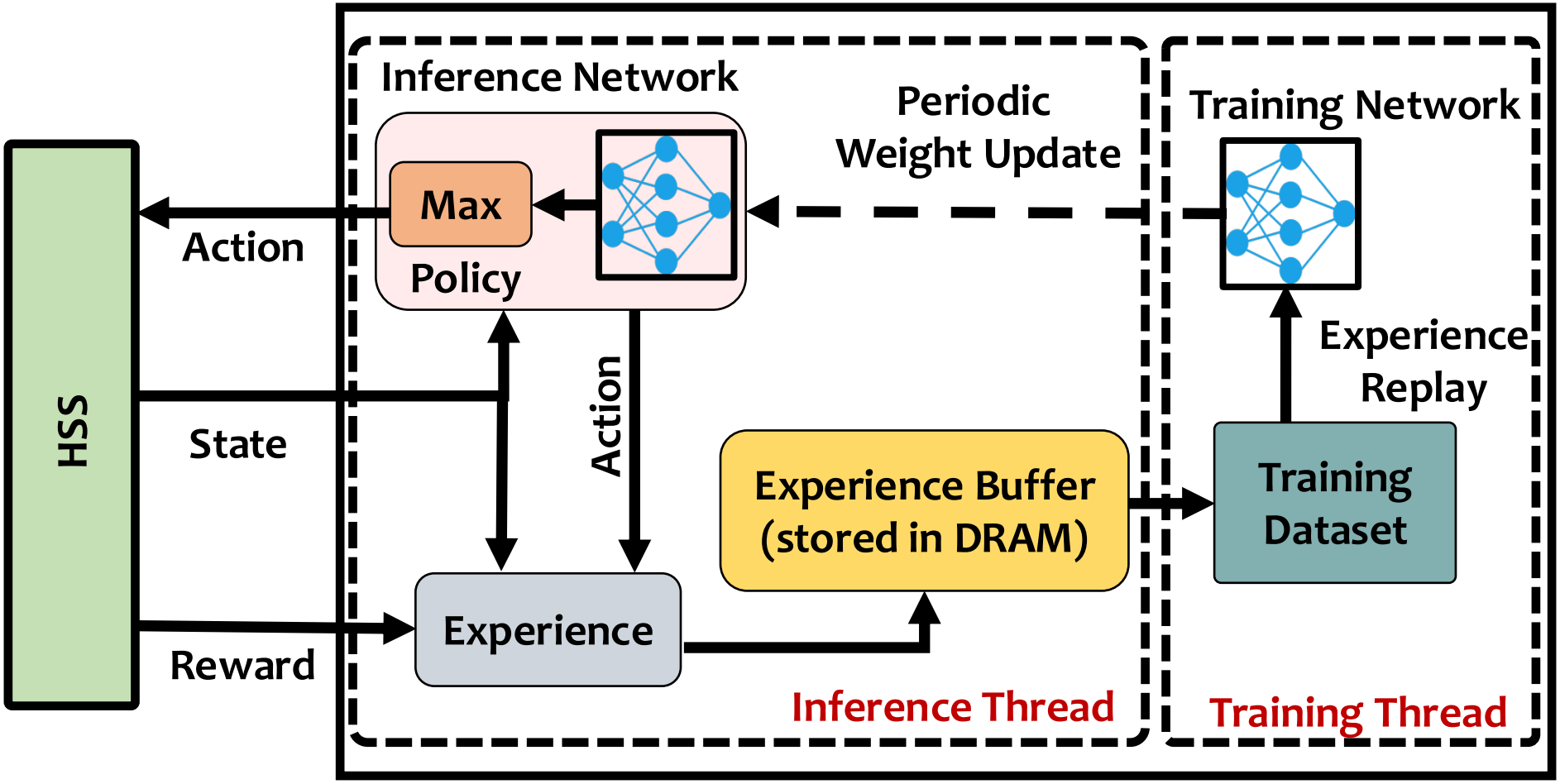
Evaluation and Key Results

Conclusion

Harmonia: Overview



Harmonia: RL Agent Design



More in the Paper

- Coordination between Harmonia's agents
  - Harmonia (with coordination) outperforms its uncoordinated variant by 26.7% (59.8%) on a performance- (cost-) optimized HSS
- Harmonia's adaptability to changes in workload access patterns
- Detailed RL agent design
  - RL algorithm
  - Hyper-parameter tuning
  - Training and inference network design
- Detailed workload characterization
- Detailed analysis of why RL is a good fit for HSS data management
- Detailed background on the prior data placement and migration techniques

More in the Paper

Harmonia: Enhancing Data Placement and Migration in Hybrid Storage Systems via Multi-Agent Reinforcement Learning

Rakesh Nadig[§] Vamanan Arulchelvan[§] Rahul Bera[§] Taha Shahroodi [§] Gagandeep Singh[†]
Andreas Kakolyris [§] İsmail Emir Yüksel [§] Mohammad Sadrosadati [§] Jisung Park[∇]
Onur Mutlu[§]

[§]*ETH Zürich* [†]*AMD Research* [∇]*POSTECH*

https://arxiv.org/pdf/2503.20507

Talk Outline

Background and Motivation

HSS Data Management using Reinforcement Learning

Harmonia

Evaluation and Key Results

Conclusion

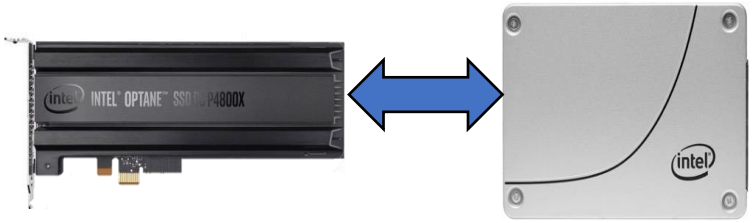
Evaluation Methodology (1/2)

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

Cost-Optimized Dual HSS

High-end SSD Low-end HDD

Performance-Optimized Dual HSS



High-end SSD Mid-Range SSD

Evaluation Methodology (2/2)

- **Eight Baselines**

Data Placement

- **CDE** [Matsui+, Proc. IEEE 2017]
- **Sibyl** [Singh+, ISCA 2022]

Data Migration

- **RNN-HSS** [Doudali+, HPDC 2019]
- **K-SVM** [Shetti+, NAS 2022]

Extended Techniques

- **Sibyl + K-SVM**
- **Sibyl + RNN-HSS**
- **CDE + RL-Migr**
- **SAPM (Single Agent for Placement and Migration)**

- **25 Data-Intensive Workloads**

Yahoo Cloud-Serving Benchmark

Cloud Database workloads
(e.g., insert, update, read, scan)

SYSTOR'17

Traffic on enterprise virtual desktop infrastructure (VDI)

RocksDB

Key-Value Store

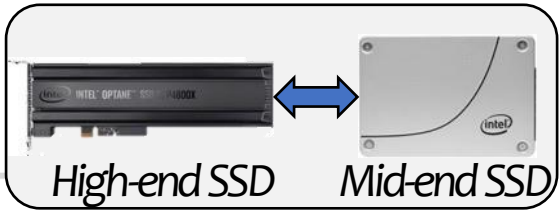
MLPerf Storage

Image Classification and Cosmology Parameter Prediction

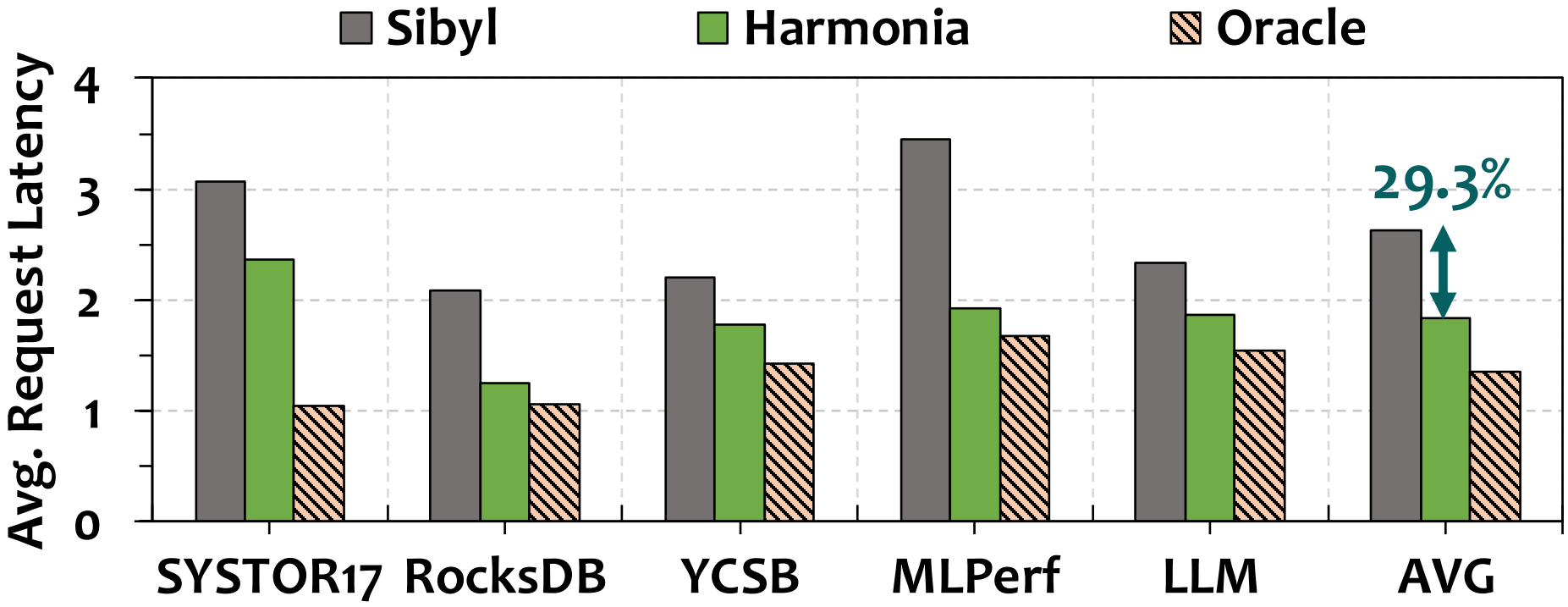
Large Language Models

Checkpoint writes, loading of weights and model parameters, and KV Cache accesses

Performance Analysis (1/2)

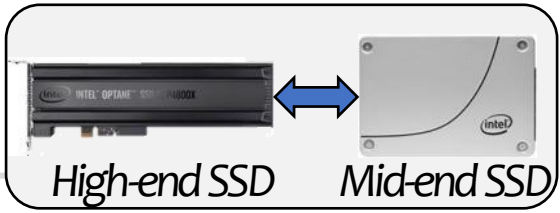


Performance-Optimized HSS

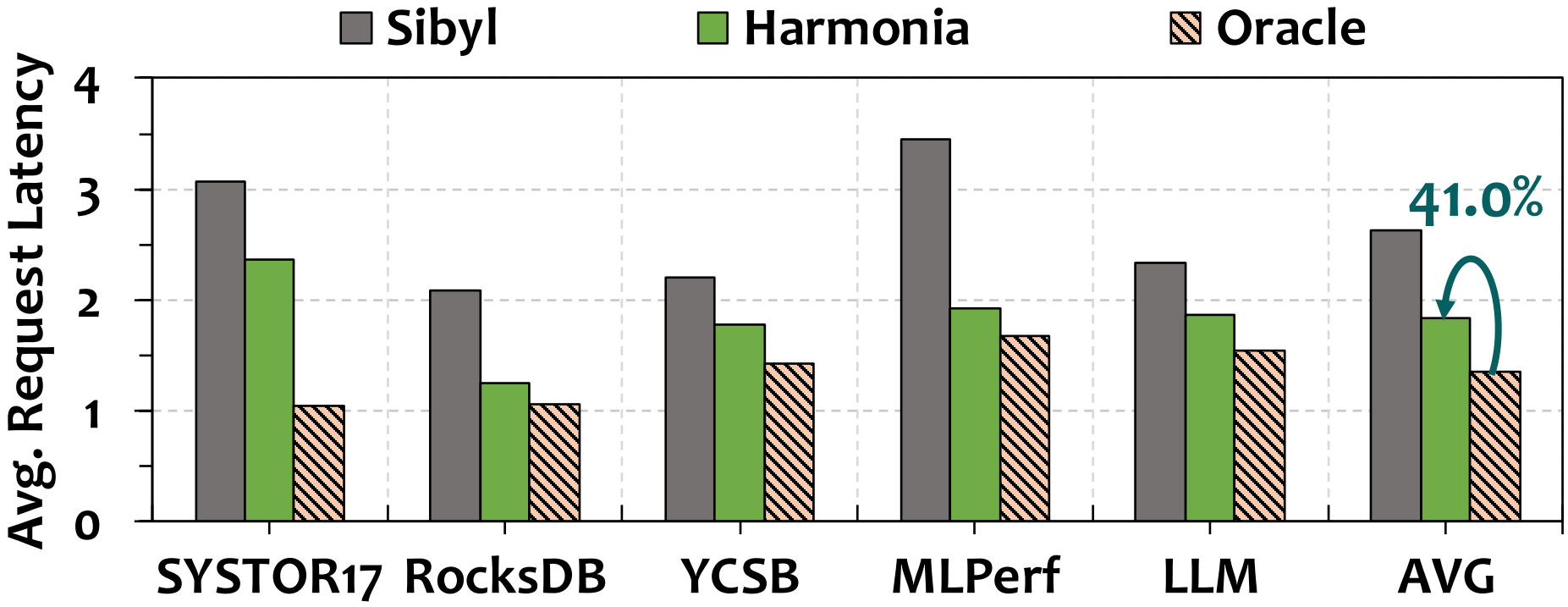


Harmonia significantly *reduces average I/O latency* over Sibyl by *co-optimizing and coordinating* the *placement and migration* policies

Performance Analysis (1/2)

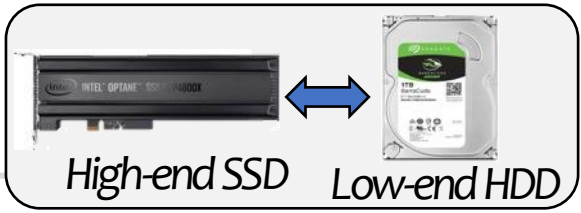


Performance-Optimized HSS

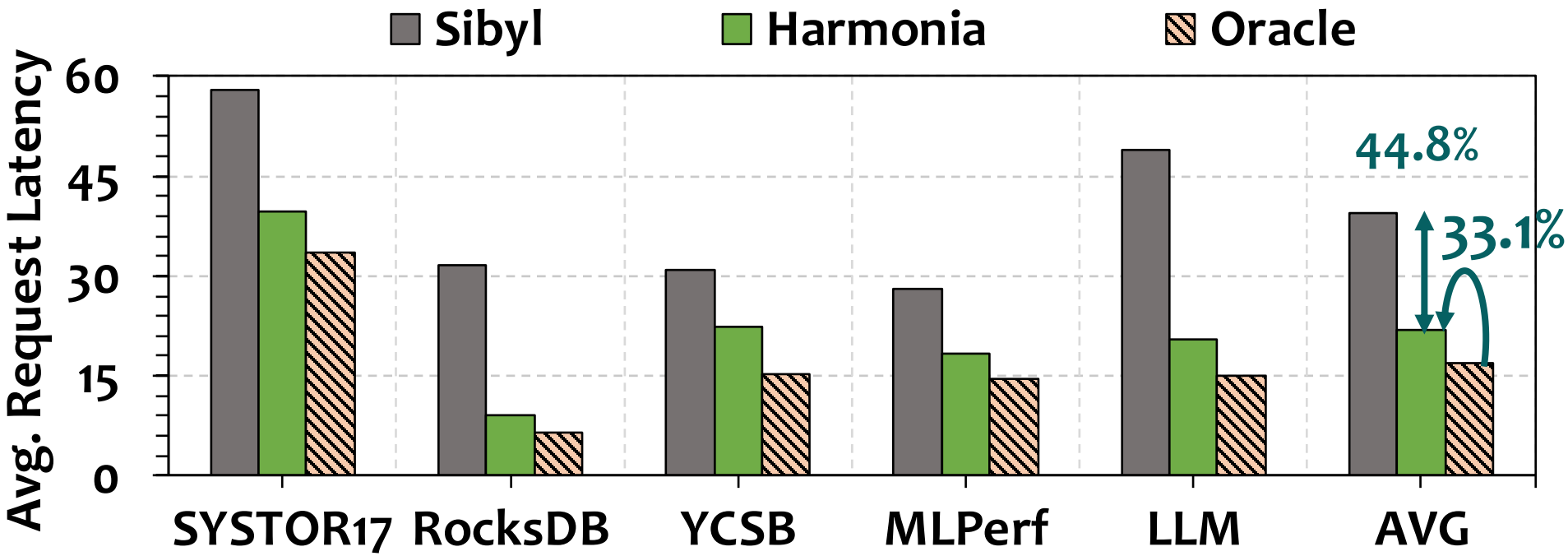


Harmonia's average I/O latency is *closer to* Oracle than prior approaches *without the knowledge of workload's future access patterns*

Performance Analysis (2/2)



Cost-Optimized HSS

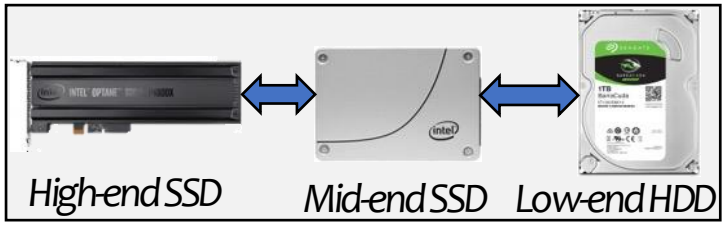


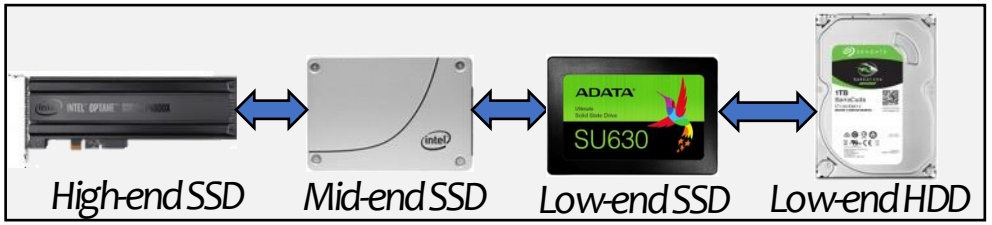
Harmonia's *proactive background migration* and *coordinated decisions* reduces the *average I/O latency* compared to prior approaches

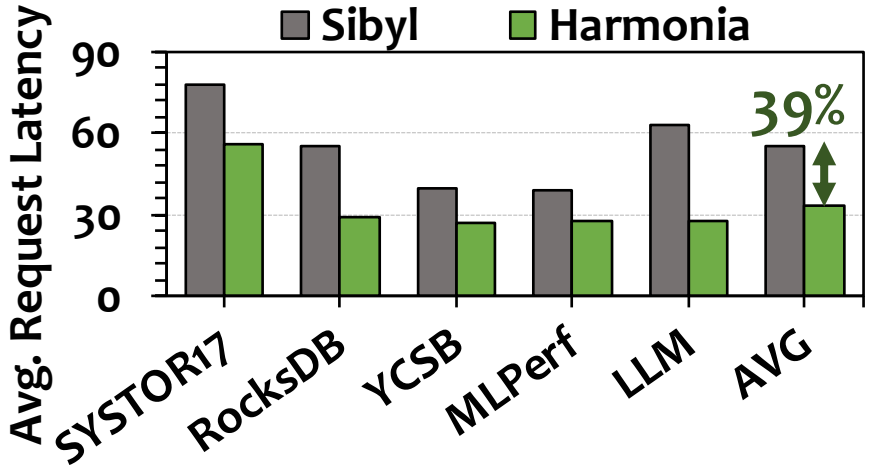
Extensibility Analysis

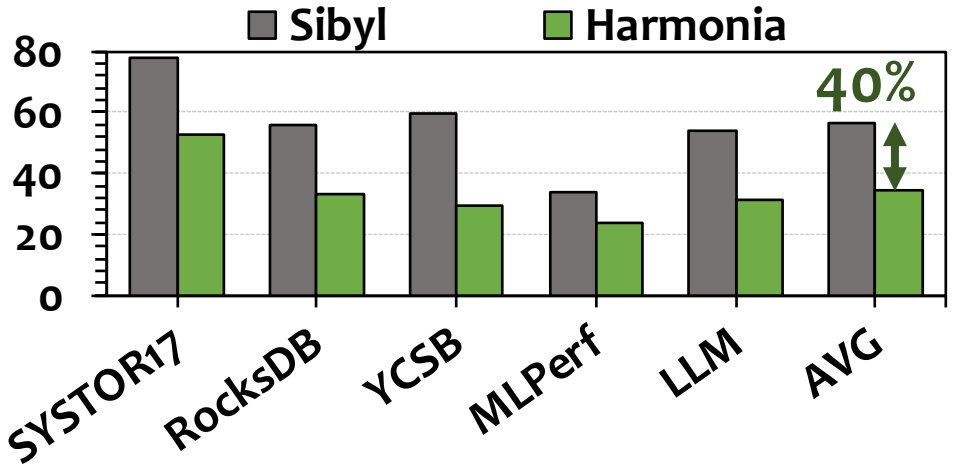
Extending Harmonia for **more devices:**

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







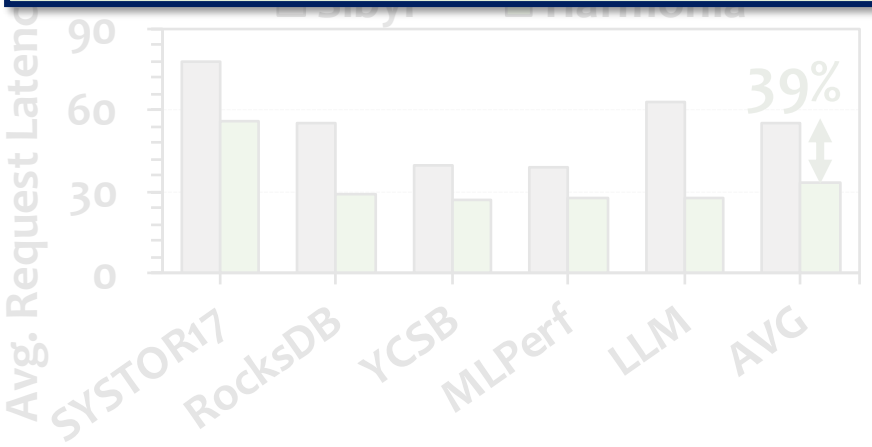


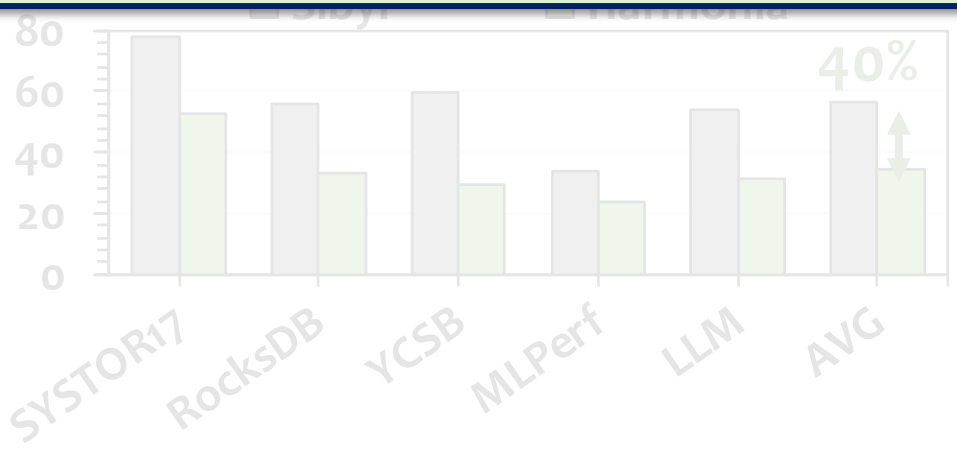
Extensibility Analysis

Extending Harmonia for **more devices:**

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

Harmonia provides *greater latency reduction* than prior approaches as the *number of devices increases* in the HSS





Overhead Analysis

- **206 KiB** of total storage cost
  - Experience buffers, inference and training networks
- **32-bit** metadata overhead per page for state features
- Inference Latency **- 240ns**
- Training Latency **- 53µs (background)**

Small storage overhead

Small inference overhead

More in the Paper (1/3)

- **Throughput (IOPS) evaluation**
  - Harmonia provides high end-to-end throughput compared to baseline policies because it's state and reward structures indirectly captures throughput
- Evaluation on unseen workloads
  - Harmonia provides high performance benefits when evaluated on unseen workloads that are not used for hyperparameter tuning
- **Performance sensitivity** to
  - Migration queue size
  - Number of incoming requests considered in migration agent's reward
- **Convergence** of Harmonia's policies
  - Both data-placement and data-migration policies converge in less than 8K I/O requests

More in the Paper (2/3)

- Harmonia's impact on device lifetimes
- Evaluation using multi-programmed workloads
- Evaluation on persistent-memory based HSS
  - Harmonia provides higher performance benefits when evaluated on a PMEM-based HSS
- Detailed overhead analysis
- More observations and insights from our evaluation

More in the Paper (3/3)

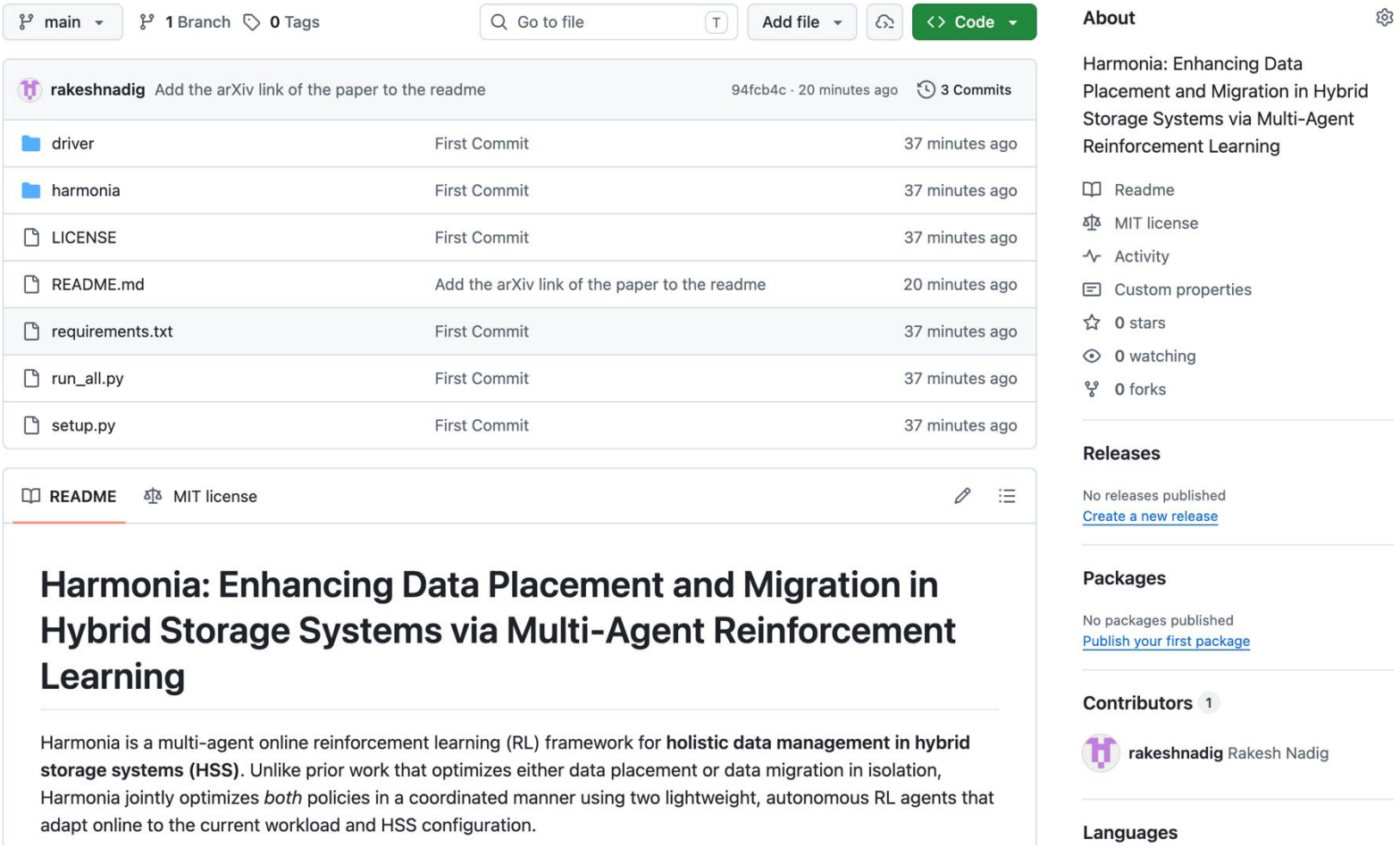
Harmonia: Enhancing Data Placement and Migration in Hybrid Storage Systems via Multi-Agent Reinforcement Learning

Rakesh Nadig[§] Vamanan Arulchelvan[§] Rahul Bera[§] Taha Shahroodi [§] Gagandeep Singh[†]
Andreas Kakolyris [§] İsmail Emir Yüksel [§] Mohammad Sadrosadati [§] Jisung Park[∇]
Onur Mutlu[§]

[§]*ETH Zürich* [†]*AMD Research* [∇]*POSTECH*

https://arxiv.org/pdf/2503.20507

Harmonia is Open-Source



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

Talk Outline

Background and Motivation

HSS Data Management using Reinforcement Learning

Harmonia

Evaluation and Key Results

Conclusion

Conclusion

First work to use *multi-agent reinforcement learning*
to optimize data management
in hybrid storage systems

Reduces average I/O latency
by *29.3%/44.8%* over the *best-performing prior work*
on performance-optimized/cost-optimized dual HSS

Provides greater *latency reduction* than the
prior approaches *when extended to more storage devices*

Low inference latency and storage *overheads*

Harmonia

Enhancing Data Placement and Migration in Hybrid Storage Systems via Multi-Agent Reinforcement Learning

Rakesh Nadig, Vamanan Arulchelvan, Rahul Bera, Taha Shahroodi, Gagandeep Singh, Andreas Kosmas Kakolyris, İsmail Emir Yüksel, Mohammad Sadrosadati, Jisung Park and Onur Mutlu

ICS 2026

SAFARI ETH zürich

AMD

POSTECH

FAQ (1/2)

- Executive Summary
- Lack of Holistic Data Management in HSS
- Lack of Coordination in Extended Techniques
- Limitations of a Single-Agent RL Technique
- Workload Access Patterns
- State Features in Harmonia
- Training and Inference Network
- Hyperparameter Tuning
- Evaluation Methodology
- Workload Characteristics

FAQ (2/2)

- Multi-Programmed Workloads
- Results
  - Throughput Analysis
  - Device Lifetimes
  - Multi-Programmed Workloads
  - Tail Latency
  - Harmonia's Adaptivity
  - Sensitivity Analysis
  - Write Traffic Distribution
  - Unseen Workloads
  - Impact of Coordination

Executive Summary (1/2)

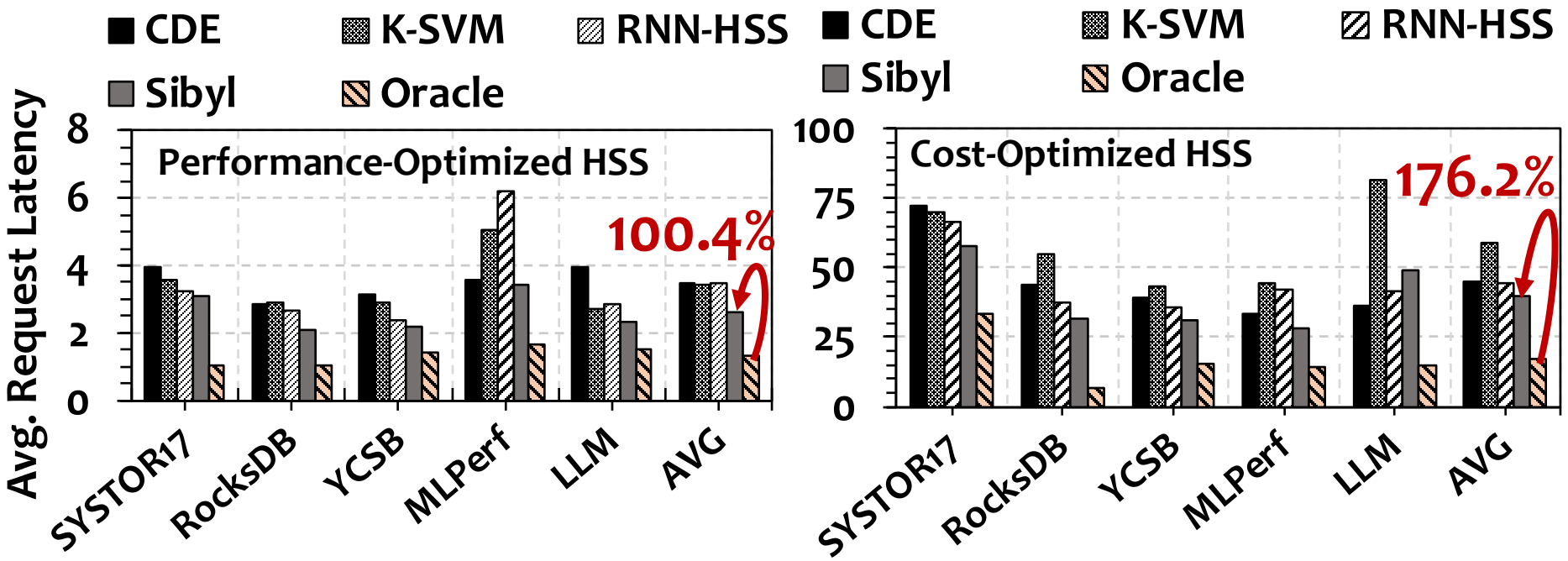
- **Background**:
  - A hybrid storage system (HSS) uses multiple heterogeneous storage devices to provide scalable storage capacity at high performance
  - The performance of an HSS highly depends on the effectiveness of its (1) data-placement, and (2) data-migration policies
- **Problem**: Two key shortcomings of prior HSS data-management techniques:
  - Prior techniques focus on improving only data-placement or only data-migration policies, but do not optimize both
  - Naively combining prior data-placement and data-migration techniques provide sub-optimal performance due to lack of coordination between the two policies
- **Goal**: Design a holistic data management technique that
  - optimizes both data-placement and data-migration policies, and
  - achieves coordination between the two policies

Executive Summary (2/2)

- **Contribution:** Harmonia, the first multi-agent online reinforcement learning-based HSS data-management technique that:
  - Performs **combined optimization** of both data placement and data migration policies
  - Achieves **coordination** between data placement and data migration to avoid conflicting decisions
  - Can **easily extend** to any number of storage devices
- **Key Results:** Evaluate on a **real system with multiple HSS configurations** using 25 data-intensive workloads
  - Harmonia **improves latency by 29.3%/44.8%** compared to the state-of-the-art data placement technique in performance-optimized/cost-optimized dual-HSS configuration
  - In a tri-/quad-HSS configuration, Harmonia outperforms the state-of-the-art policy by **39%/40%**
  - Harmonia's performance benefits come with low latency (**240ns** for inference) and storage overheads (**206 KiB** in DRAM for both RL agents together)

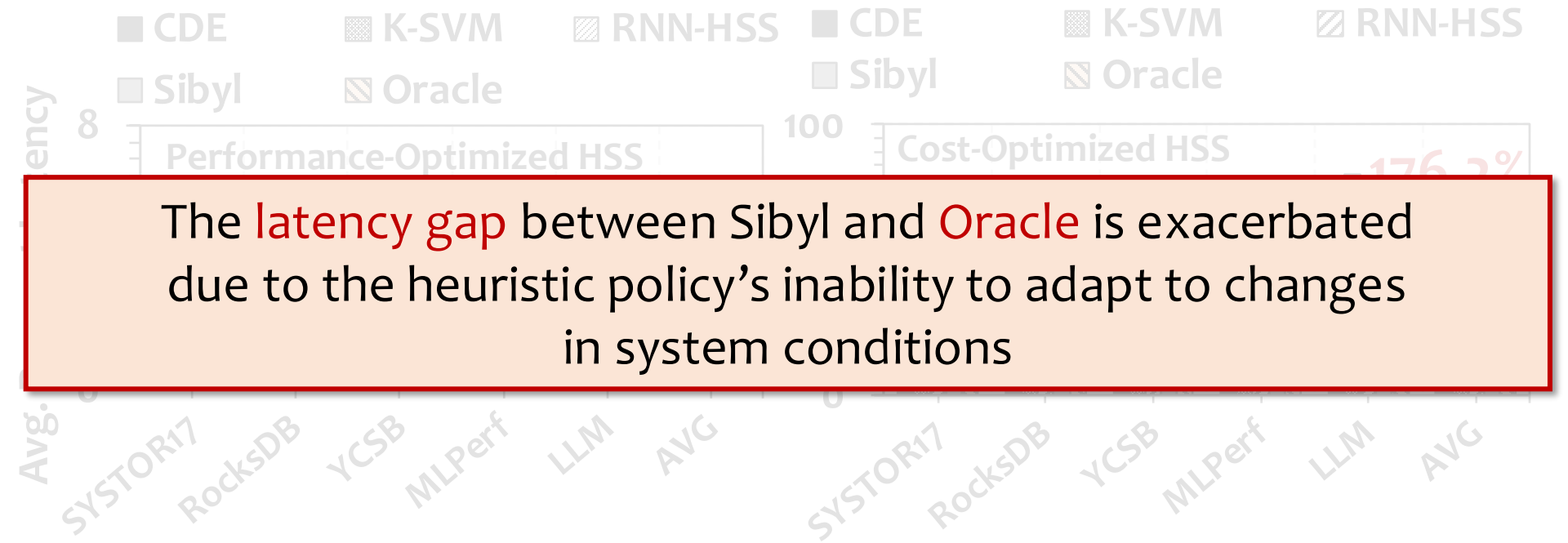
Lack of Holistic Data Management in HSS

- Prior techniques
  - do not optimize both data placement and migration policies together
  - use heuristic approaches, which do not adapt to changes in workload access patterns and HSS conditions



Lack of Holistic Data Management in HSS

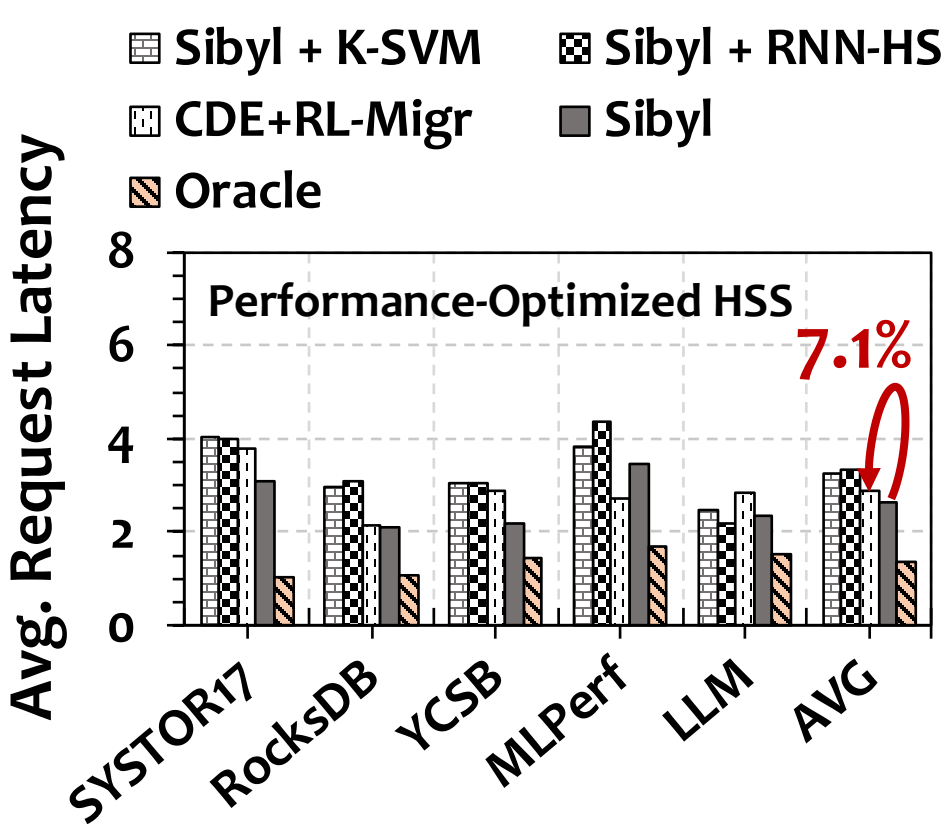
- Prior techniques
  - do not optimize both data placement and migration policies together
  - use heuristic approaches, which do not adapt to changes in workload access patterns and HSS conditions

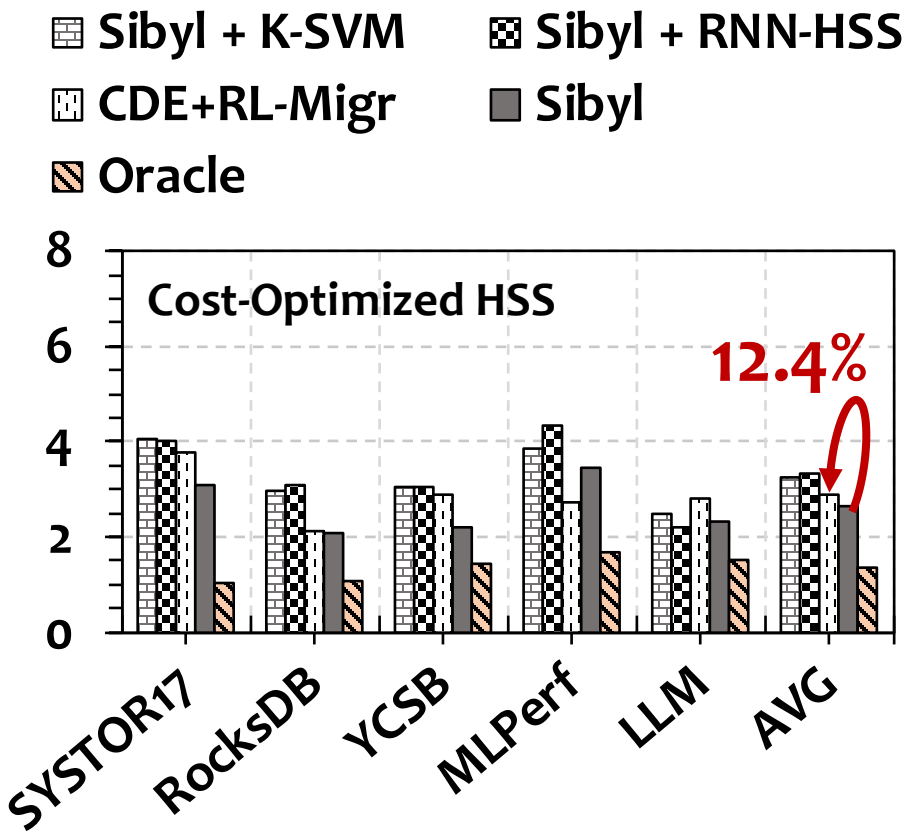


The latency gap between Sibyl and Oracle is exacerbated due to the heuristic policy's inability to adapt to changes in system conditions

Lack of Coordination in Extended Techniques

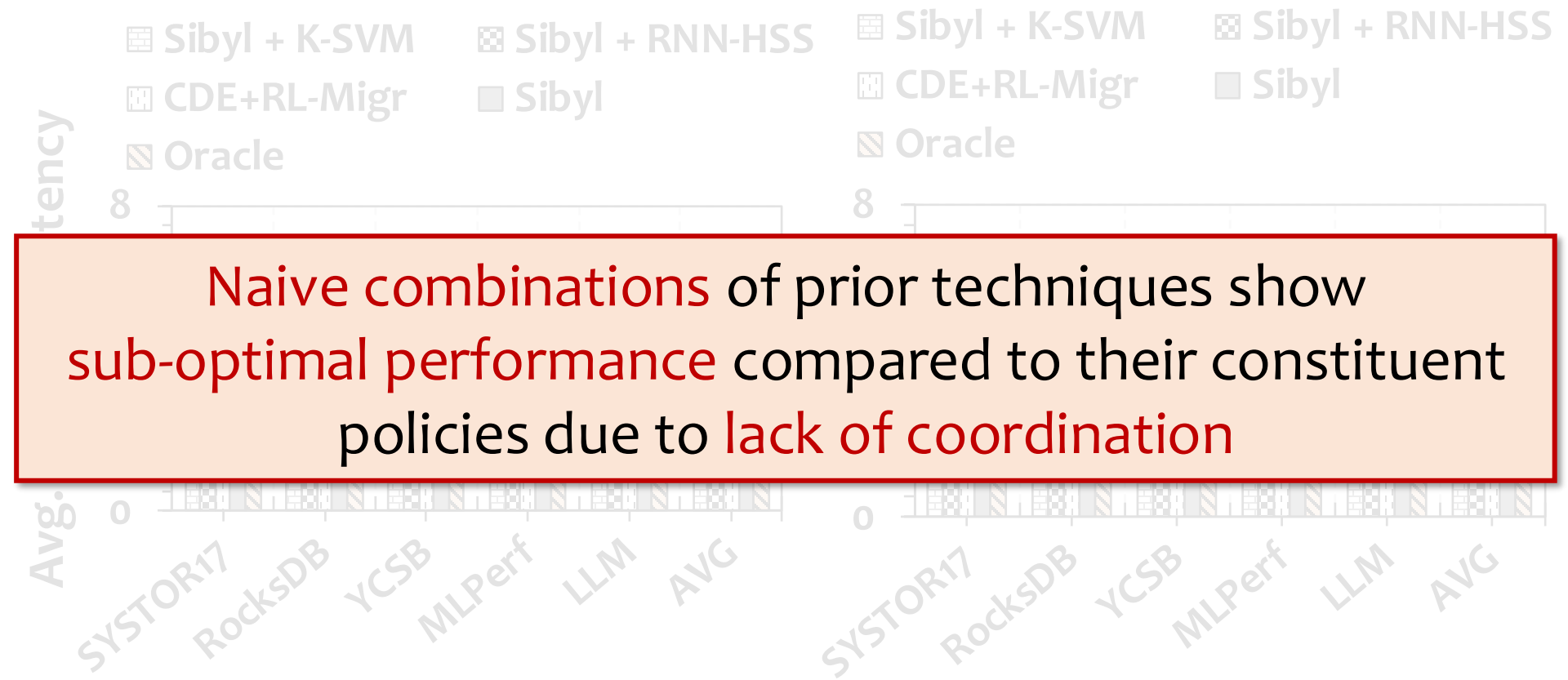
- *What happens if we combine the best placement and migration techniques?*





Lack of Coordination in Extended Techniques

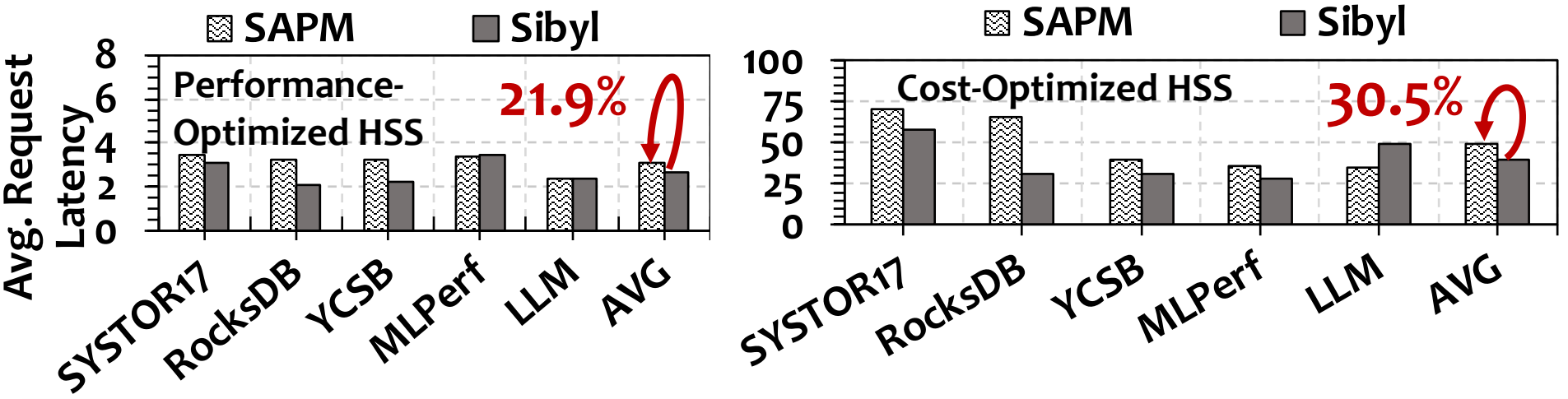
- *What happens if we combine the best placement and migration techniques?*



Naive combinations of prior techniques show sub-optimal performance compared to their constituent policies due to lack of coordination

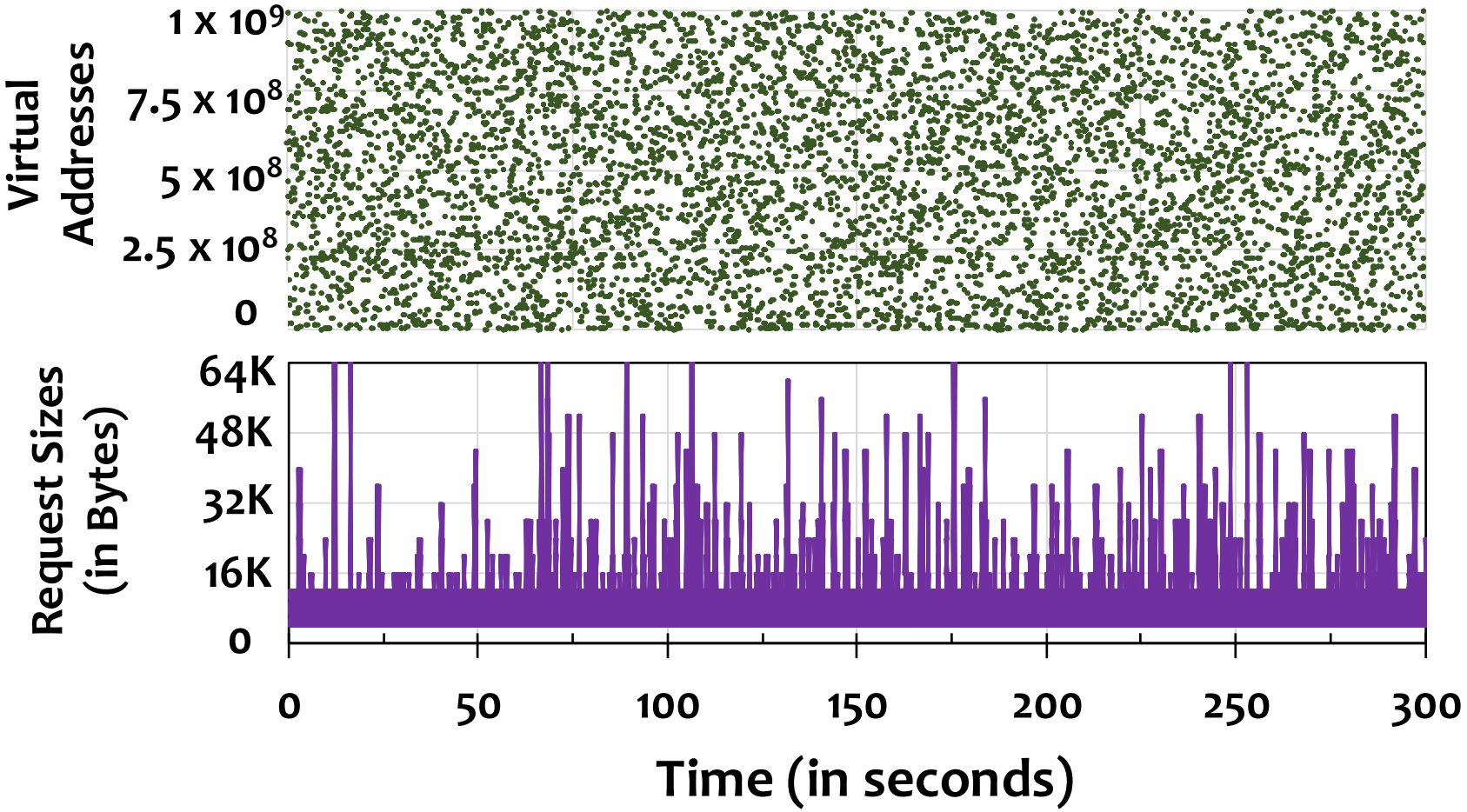
Limitations of a Single-Agent RL Technique

- An RL-based technique can adapt to changes in workload and HSS configurations
  - Sibyl is the best-performing prior HSS data-management technique
- Data placement and data migration are two different tasks in HSS with different objectives
- A single RL agent based technique cannot optimize different tasks concurrently because it relies on task similarity to learn multiple tasks



SAPM has higher latency than Sibyl because it cannot learn optimal policies for two different tasks concurrently

Workload Access Patterns



State Features in Harmonia

Feature	Description	# of bins	Encoding (bits)
req_type	Request type (read/write)	2	1
req_size	Request size (in pages)	8	3
acc_intr	Access interval of the requested page	64	8
acc_freq	Access frequency of the requested page	64	8
fast_cap	Free space in the fast storage device	8	3
curr_dev	Storage device where the requested page currently resides	2	1
migr_intr	Migration interval of a page	64	8

Hyper-Parameter Tuning

- Different hyper-parameter configurations were chosen using the design of experiments (DoE) technique

Hyper Parameter	Design Space	Placement Agent	Migration Agent
Discount Factor (γ)	0-1	0.9	0.1
Learning Rate (α)	$1e^{-5} - 1e^{0}$	$1e^{-3}$	$1e^{-2}$
Exploration Rate (ϵ)	0-1	0.001	0.001
Batch Size	64-256	128	256
Experience Buffer Size	10-10000	1000	1000

Evaluation Methodology

Host System	AMD Ryzen 7 2700G[110], 8-cores@3.5 GHz, 8×64/32 KiB L1-I/D, 4 MiB L2, 8 MiB L3, 16 GiB RDIMM DDR4 2666 MHz
Storage Devices	**Characteristics**
H: Intel Optane SSD P4800X [47]	375 GB, PCIe 3.0 NVMe, SLC, R/W: 2.4/2 GB/s, random R/W: 550000/500000 IOPS
M: Intel SSD D3-S4510 [48]	1.92 TB, SATA TLC (3D), R/W: 560/510 MB/s, random R/W: 895000/21000 IOPS
L: Seagate HDD ST1000DM010 [108]	1 TB, SATA 6Gb/s 7200 RPM Max. Sustained Transfer Rate: 210 MB/s
$\mathbf{L_{SSD}}$: ADATA SU630 SSD [111]	960 GB, SATA, TLC, R/W: 520/450 MB/s
PMEM (Emulated): Intel Optane Persistent Memory 200 Series [112]	128 GB, Memory Mode R/W: 7.45/2.25 GB/s (256B)
HSS Configurations	**Devices**
Performance-Optimized	high-end (H) & middle-end (M)
Cost-Optimized	high-end (H) & low-end (L)
HSS with PMEM	Emulated PMEM (PMEM) & high-end SSD (H)
Tri-HSS	high-end (H) & middle-end (M) & low-end (L)
Quad-HSS	high-end (H) & middle-end (M) & low-end SSD (L_{SSD}) & low-end HDD (L)

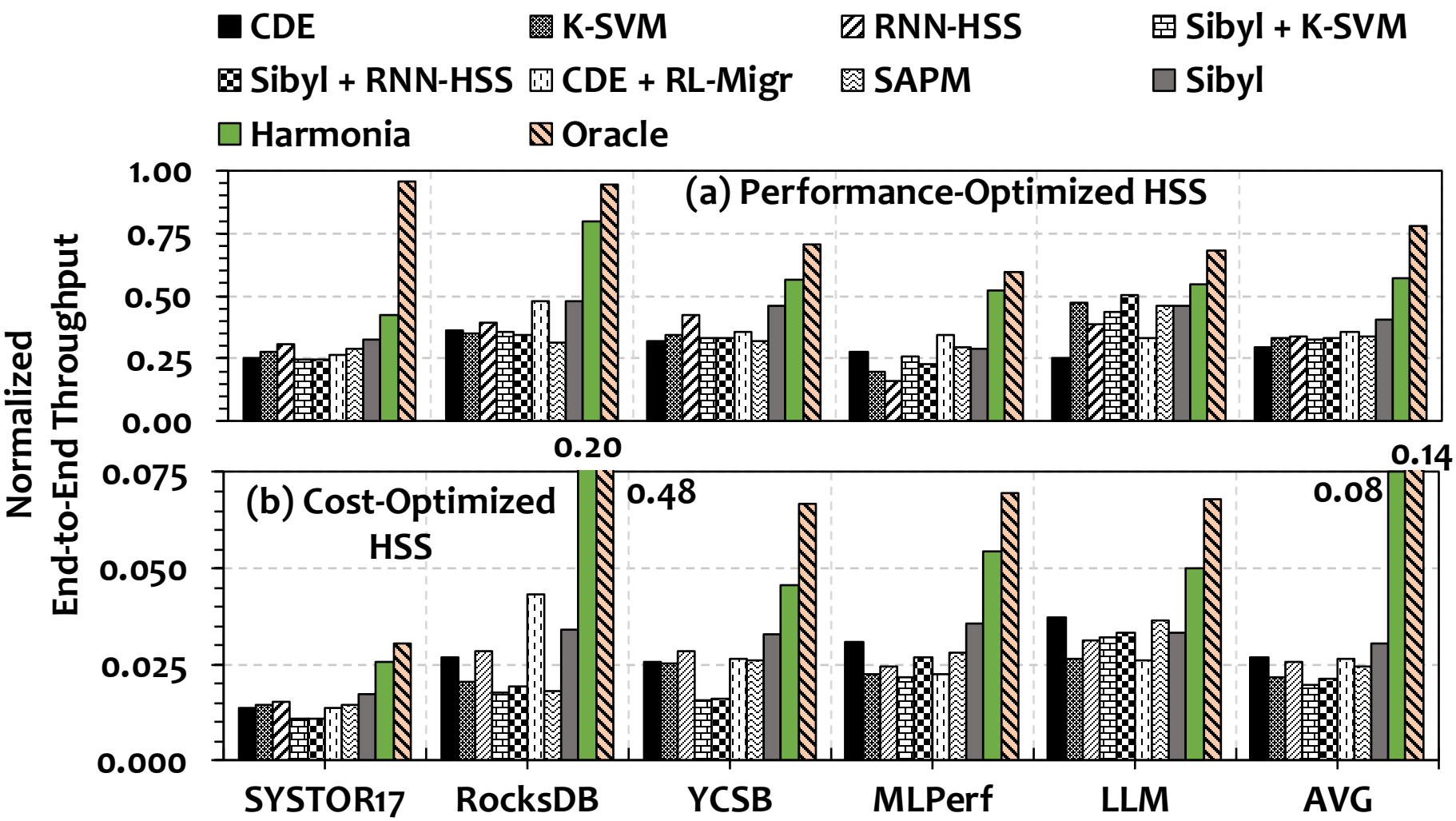
Workload Characteristics

Benchmark Suite	Traces	Read %	Avg. Request Size (KB)	Avg. Inter-Request Time (μs)
SYSTOR17 [262]	LUN0	0.2	31.7	1163.9
	LUN1	0.3	34.2	1864.1
	LUN2	7.6	31.1	1418.9
	LUN3	3.5	42.7	734.5
	LUN4	0.5	26.3	823.1
RocksDB [263]	ssd-00	79.9	108.9	66.4
	ssd-01	73.5	75.1	40.7
	ssd-02	79.9	7.5	3.3
	ssd-03	79.9	9.5	3.5
	ssd-04	79.9	7.8	3.6
YCSB [246]	YCSB-B	51.3	45.9	9.3
	YCSB-C	47.6	54.6	6.5
	YCSB-D	55.9	36.1	8.5
	YCSB-E	52.1	46.6	9.6
	YCSB-F	49.5	53.1	6.6
MLPerf Storage [264]	ResNet50	80.0	172.6	500.1
	CosmoFlow	83.4	180.1	1023.8
LLM [24, 265–267]	LLM-Checkpoint	10.5	256	94.8
	LLM-Weights	25.1	128.0	45.0
	LLM-KVCache	45.0	16.0	24.0
Baleen [247–249] (Unseen Workloads)	Region 1	87.9	3578.9	7.1
	Region 2	78.1	2781.6	6.5
	Region 3	74.4	2084.7	7.6
	Region 4	79.9	2850.8	4.0
	Region 5	66.8	2920.6	3.0

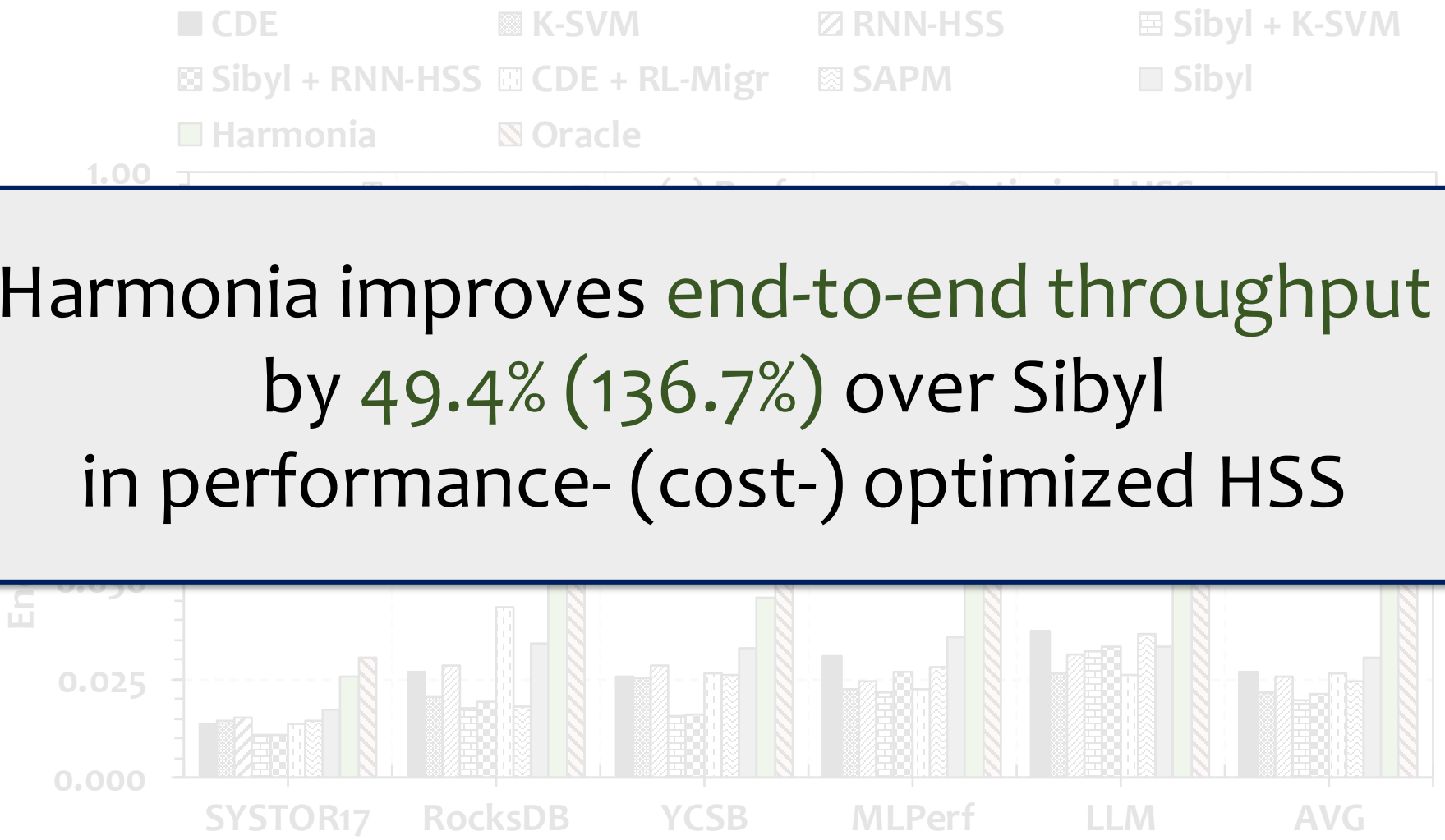
Multi-Programmed Workloads

Mix	Constituent Workloads [246, 262, 263]	Description
mix1	ssd-02 and LUN4	ssd-02 is read-intensive and LUN4 is write-intensive
mix2	LUN1 and ssd-04	LUN1 is write-intensive and ssd-04 is read-intensive
mix3	YCSB-C and YCSB-F	Both have near-equal read-write ratio
mix4	ssd-00, ssd-04, LUN0 and LUN1	Two read-intensive and two write-intensive workloads
mix5	ssd-00, LUN0, YCSB-C and YCSB-F	Read-intensive, write-intensive and two workloads with a near-equal read-write ratio
mix6	YCSB-B, YCSB-D, LUN0, LUN1, LUN4, ssd-00, ssd-02, ssd-03	Two with near-equal read-write ratio, three write-intensive and three read-intensive

Throughput Analysis (1/2)

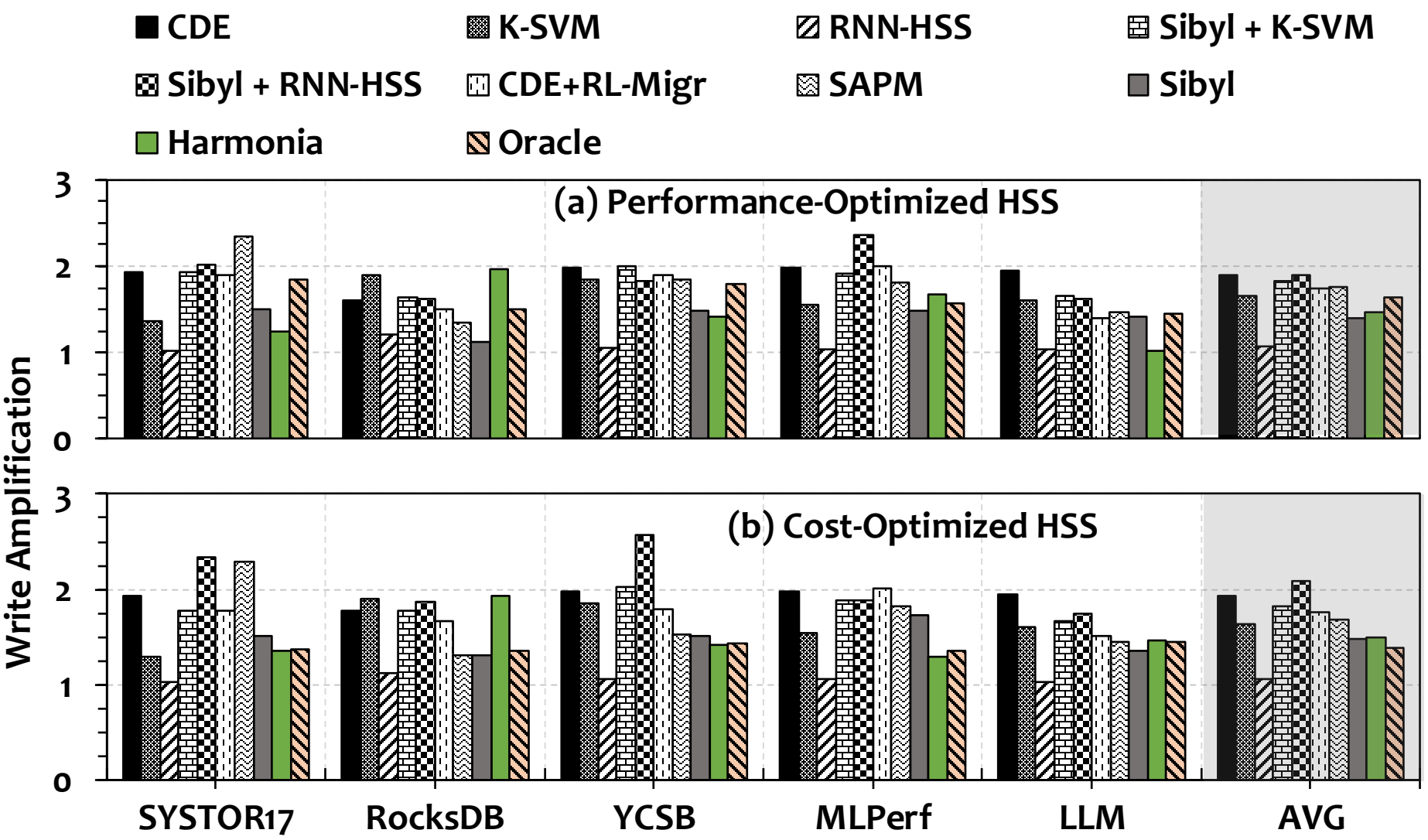


Throughput Analysis (2/2)

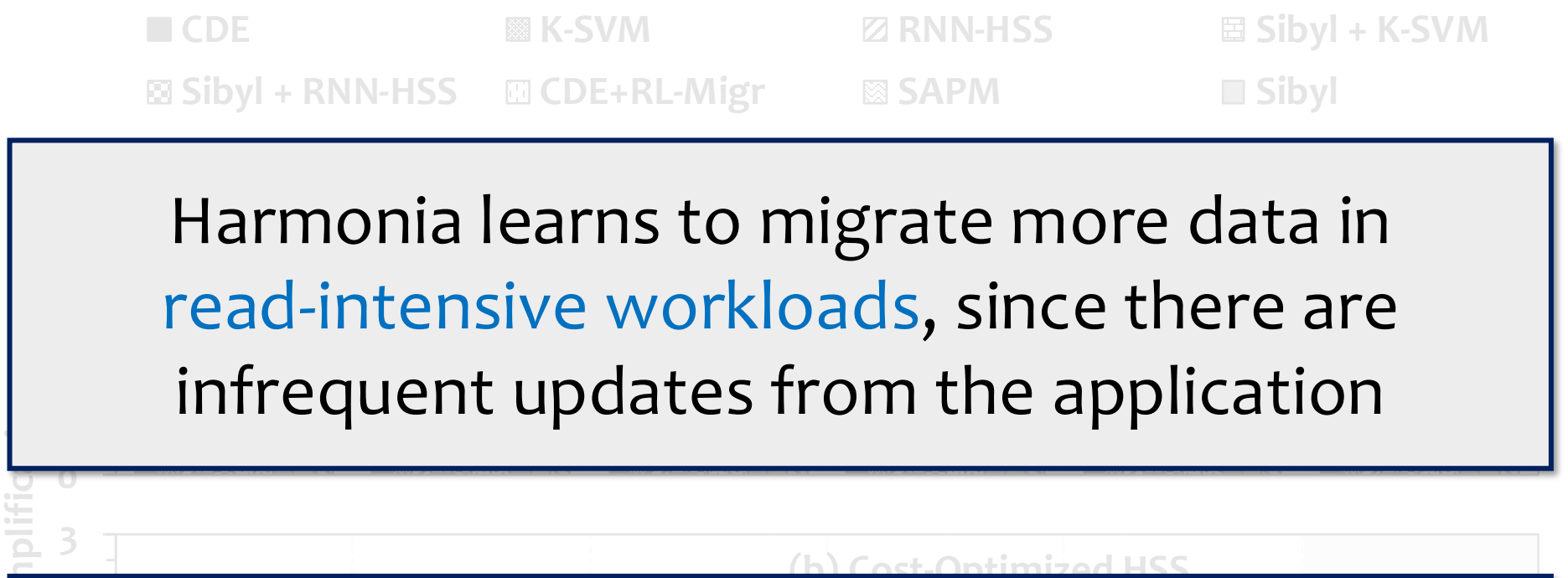


Harmonia improves end-to-end throughput by 49.4% (136.7%) over Sibyl in performance- (cost-) optimized HSS

Impact on Device Lifetimes (1/2)



Impact on Device Lifetimes (2/2)

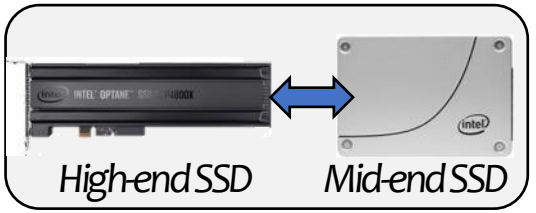


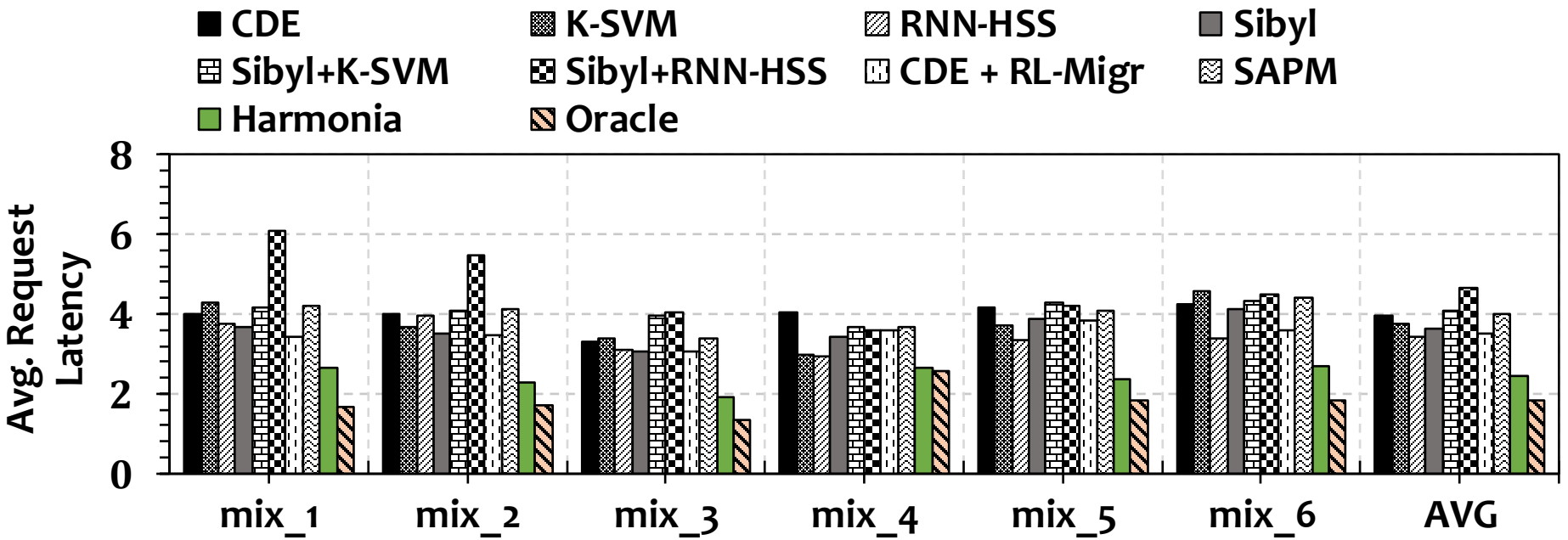
Harmonia learns to migrate more data in read-intensive workloads, since there are infrequent updates from the application

Harmonia migrates less data in write-intensive workloads as data is moved during updates from application , which causes lower write amplification

Multi-Programmed Workloads (1/2)

Performance-Optimized HSS

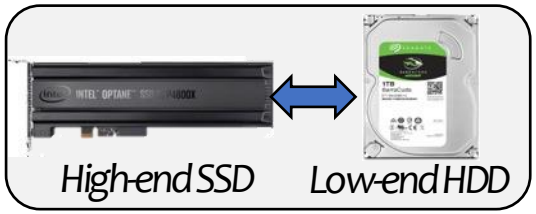


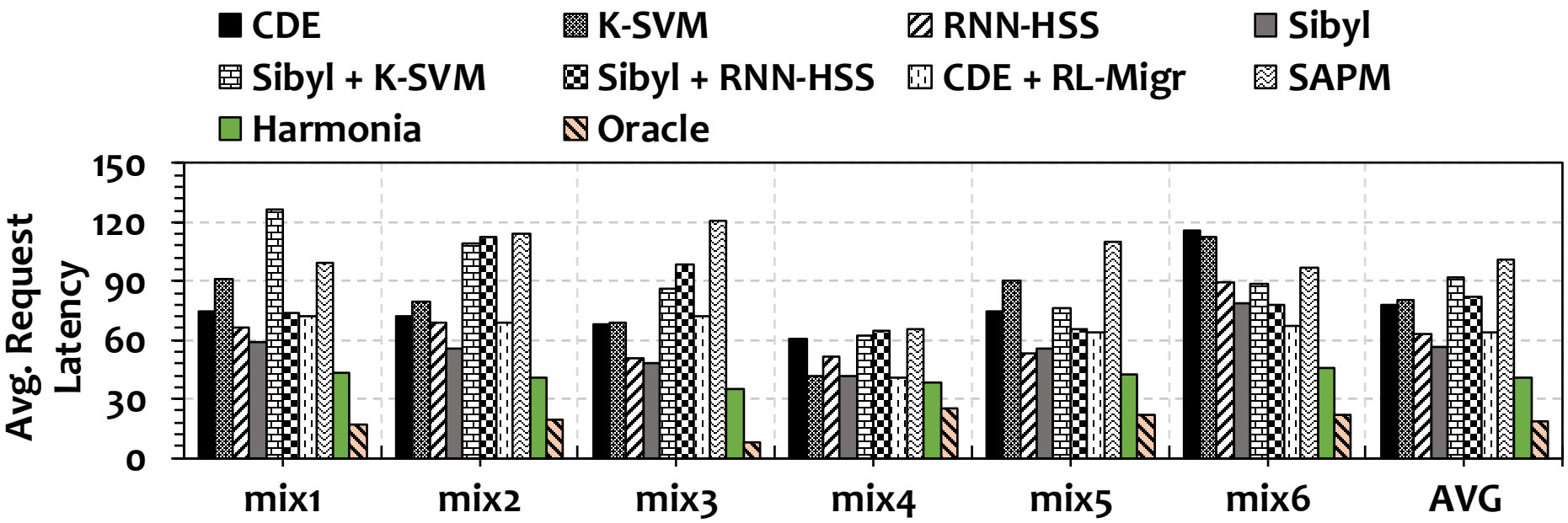


Harmonia improves latency by 33% on average over best-performing prior work, Sibyl

Multi-Programmed Workloads (2/2)

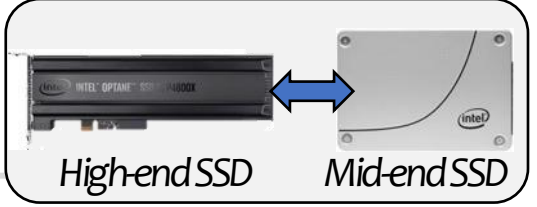
Cost-Optimized HSS

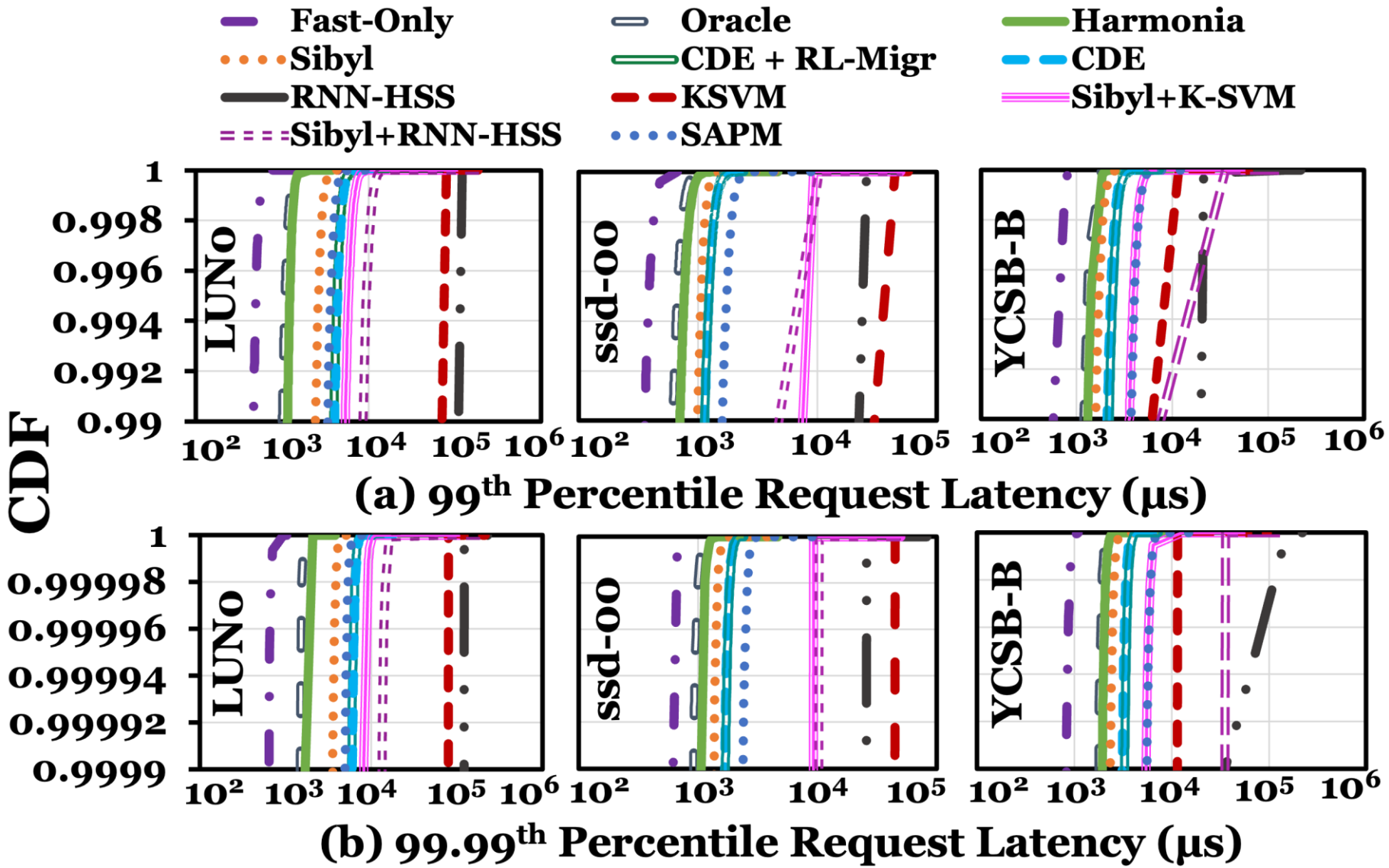




Harmonia improves latency by 25% over Sibyl due to faster learning of access pattern variations compared to prior approaches

Tail Latency



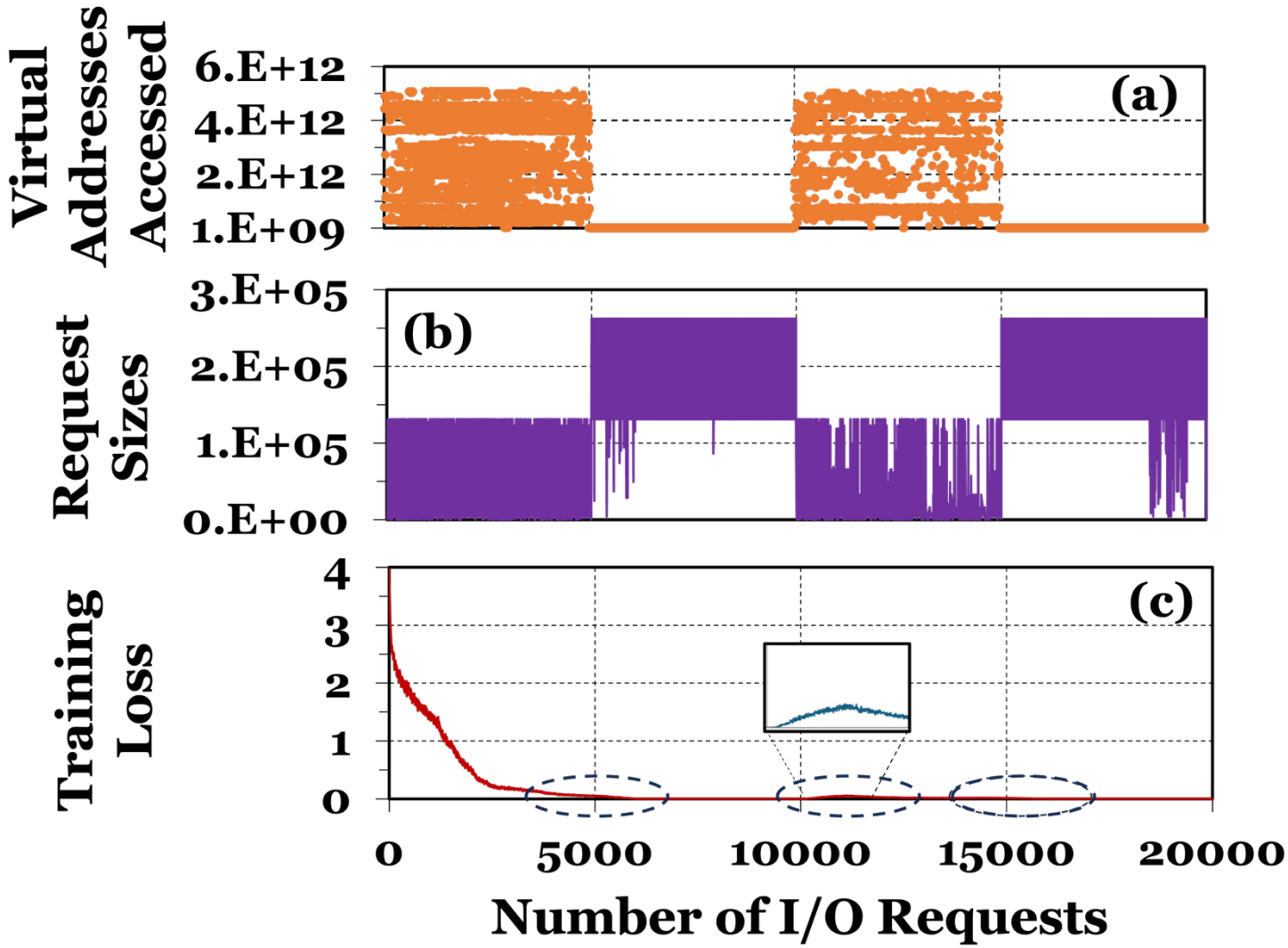


Tail Latency

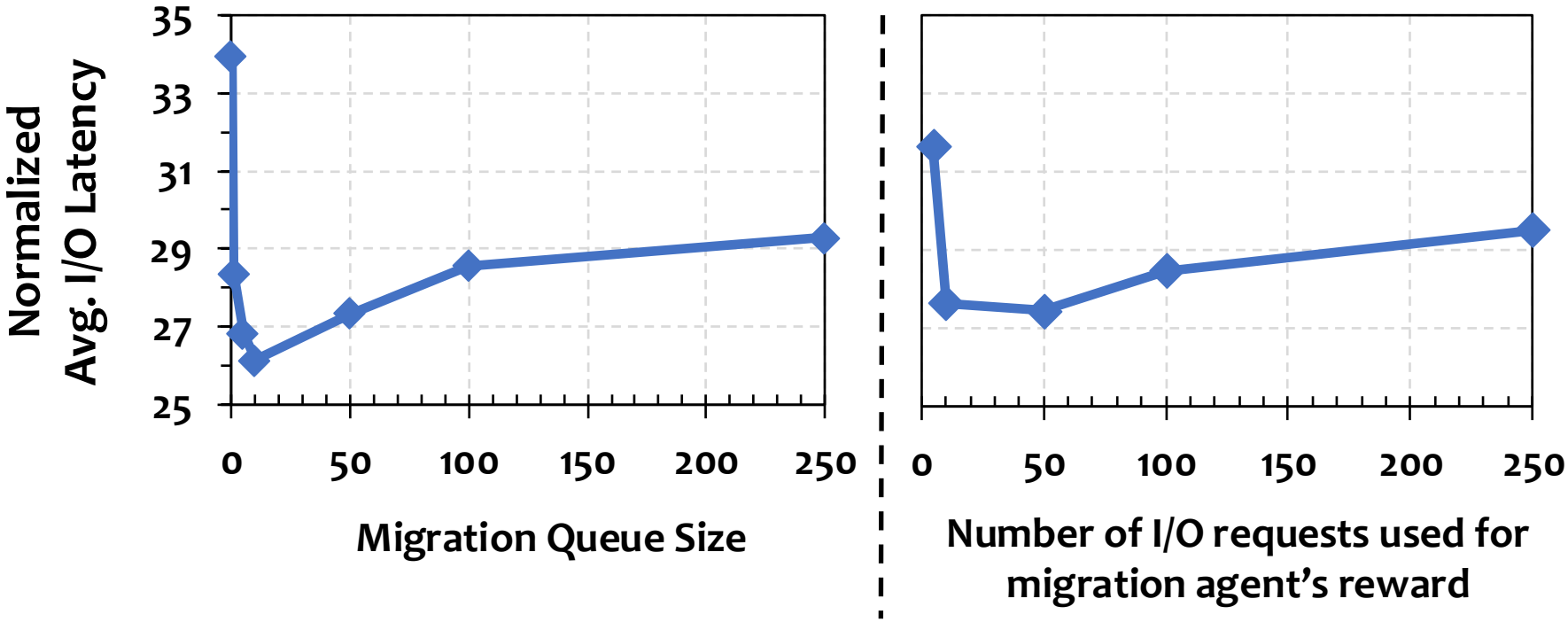
Prior approaches have high tail latency because they migrate data in the critical path of I/O request handling

Harmonia improves tail latency by migrating data during system idle times

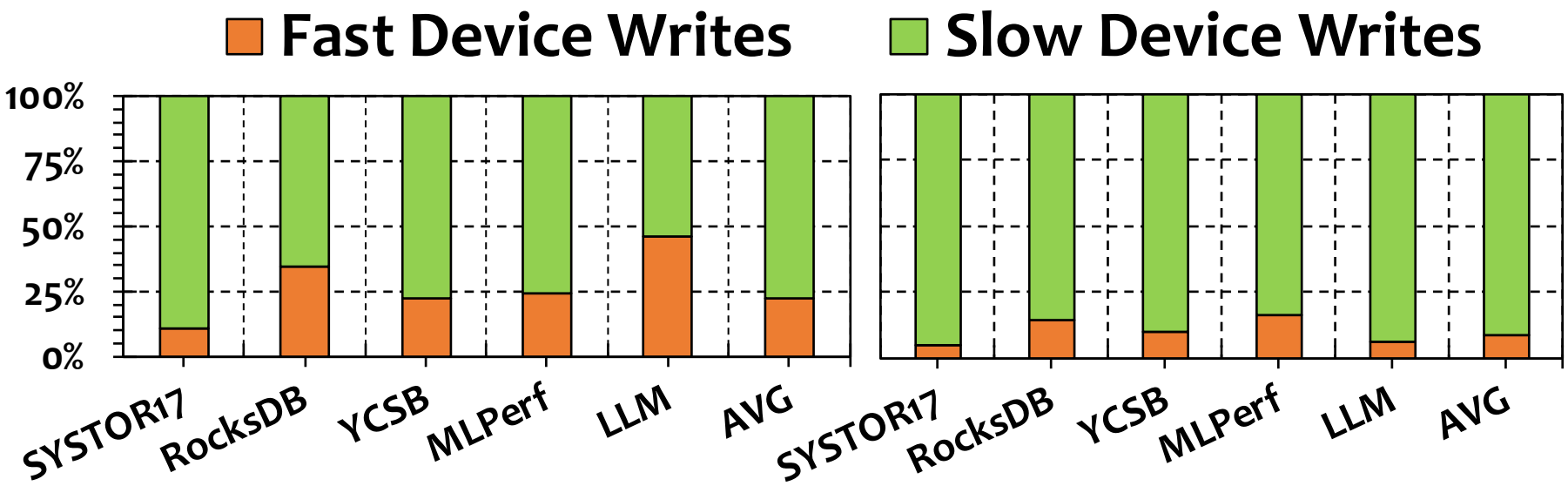
Harmonia's Adaptivity



Sensitivity Analysis



Write Traffic Distribution

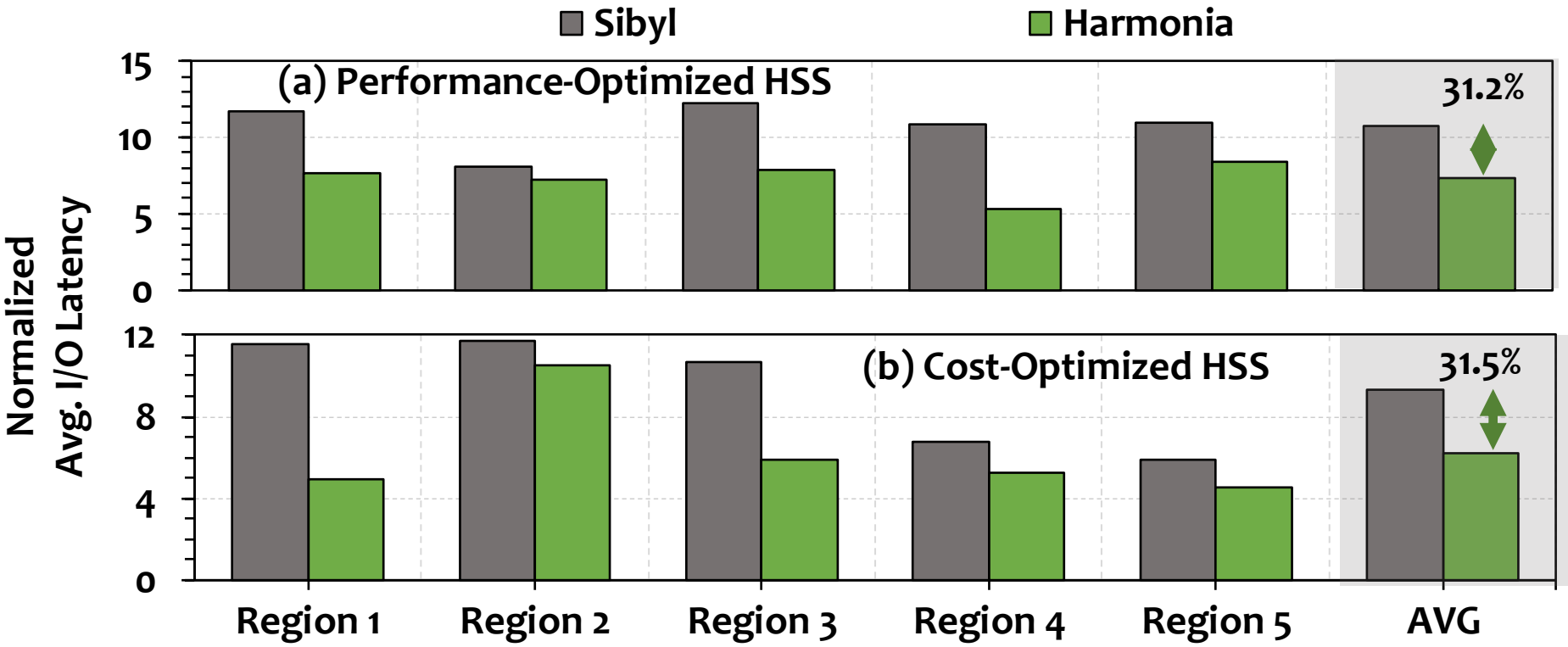


(a) Performance-Optimized HSS

(b) Cost-Optimized HSS

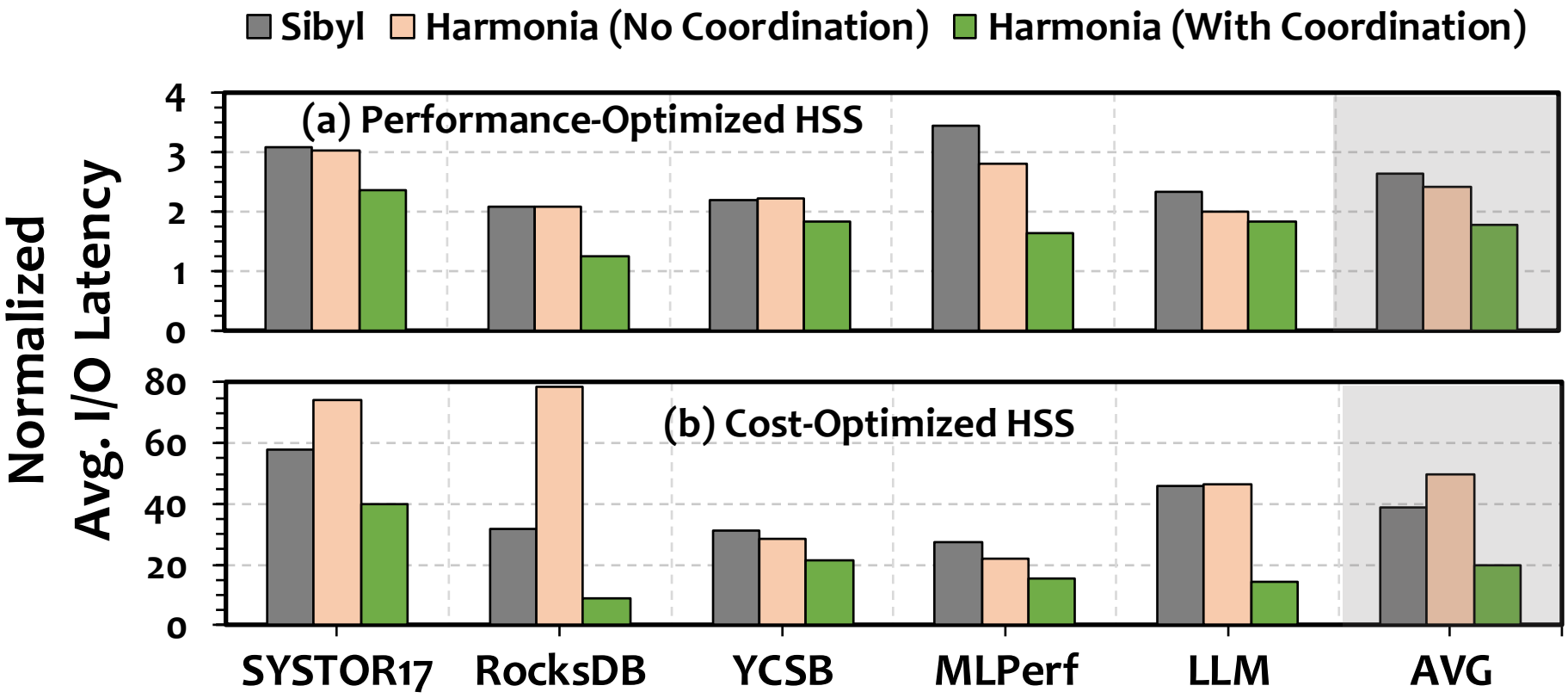
Unseen Workloads

- We use Meta's Baleen workloads to test Harmonia's latency on unseen workloads



Harmonia improves performance by 31.2% (31.5%) over Sibyl due to adaptive data placement and data migration

Impact of Coordination



Harmonia outperforms its uncoordinated variant by 26.7% (59.8%) by avoiding conflicting decisions