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
Seminar in Computer Architecture
02.05.2024
Executive Summary

- **Background**: A hybrid storage system (HSS) uses multiple different storage devices to provide high performance and high capacity at low cost.

- **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 changes in workload and underlying device characteristics
  - **Easy extensibility** to incorporate a wide range of HSS 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**

[SAFARI](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
- Critical Analysis and Discussion
Hybrid Storage System Basics

Address Space (Application/File System View)

Storage Management Layer

Logical Pages

Fast Device

Promotion

Eviction

Slow Device

Hybrid Storage System
Performance of a hybrid storage system highly depends on the storage management layer’s ability to manage diverse devices and workloads.
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)

---

**HSS Configuration 1**

<table>
<thead>
<tr>
<th></th>
<th>Slow-Only</th>
<th>CDE</th>
<th>RNN-HSS</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>hm_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prn_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usr_0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wdev_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVG</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**HSS Configuration 2**

<table>
<thead>
<tr>
<th></th>
<th>Slow-Only</th>
<th>CDE</th>
<th>RNN-HSS</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>hm_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prn_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usr_0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wdev_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVG</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lack of Extensibility (1/2)

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

Change in storage configuration

Dual-HSS
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
Agent learns to take an **action** in a given **state** to maximize a numerical **reward**.
Formulating Data Placement as RL

Agent

Environment

Sibyl

Hybrid Storage System

State ($S_t$)

Reward ($R_{t+1}$)

Action ($A_t$)

Features of the current request and system

Request latency (of last served request)

Select storage device to place the current page

Features of the current request and system
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 **evicts** a page when the fast device utilization is 100%

• Sibyl **promotes** a page when there is an update from the application
Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

Sibyl: Overview

Evaluation of Sibyl and Key Results

Conclusion

Critical Analysis and Discussion
Sibyl Execution

RL Training Thread

Periodic Policy Weight Update

State, Reward, and Action Information

RL Decision Thread

Data Placement Decision

Asynchronous Execution

Storage Request (from OS)

Sibyl
Sibyl Design: Overview

Training Network

Periodic Policy Weight Update

Observation Vector

State

Storage Request (from OS)

Inference Network

Max

Sibyl Policy

State

RL Training Thread

Training Dataset

Batch

Experience Buffer (in host DRAM)

RL Decision Thread

HSS

Reward

Collect Experiences

Periodic Policy Weight Update

Observation Vector

State

Storage Request (from OS)

Inference Network

Max

Sibyl Policy

State

RL Training Thread

Training Dataset

Batch

Experience Buffer (in host DRAM)

RL Decision Thread

HSS

Reward

Collect Experiences
RL Decision Thread

Training Network

Periodic Policy Weight Update

Inference Network

Max

HSS

Collect Experiences

Experience Buffer (in host DRAM)

Batch

Observation Vector

State

Storage Request (from OS)

State

Action

Reward

RL Decision Thread

Sibyl Policy

RL Training Thread

SAFARI
RL Decision Thread

Training Network

Observation Vector

Storage Request (from OS)

Periodic Policy Weight Update

Training Dataset

Batch

RL Training Thread

RL Decision Thread

Inference Network

Max

Sibyl Policy

Experience Buffer (in host DRAM)

Action

HSS

Reward

Collect Experiences

State

SAFARI
RL Decision Thread

Inference Network

Max

State

Action

Sibyl Policy

Inference Network

Experience Buffer (in host DRAM)

HSS

Reward

Collect Experiences

Batch

Training Dataset

Periodic Policy Weight Update

Training Network

RL Decision Thread

Storage Request (from OS)
RL Decision Thread

Training Network

Observation Vector

State

Inference Network

Max

Sibyl Policy

HSS

Reward

Collect Experiences

Experience Buffer (in host DRAM)

Action

Batch

Training Dataset

Periodic Policy Weight Update

Storage Request (from OS)
RL Decision Thread

- **Observation Vector**
  - State

- **Inference Network**
  - Max
  - Sibyl Policy

- **Experience Buffer (in host DRAM)**
  - HSS
  - Reward
  - Collect Experiences

- **Training Network**
  - Periodic Policy Weight Update
  - Training Dataset
  - Batch

- **RL Decision Thread**
  - RL Training Thread
RL Training Thread

Periodic Policy Weight Update

Training Network → Training Dataset → Batch

Experience Buffer (in host DRAM)

State → Observation Vector → Inference Network → Max Sibyl Policy

Action → HSS → Reward

Collect Experiences

SAFARI
Periodic Weight Transfer

- **Training Network**
  - Periodic Policy Weight Update
- **Inference Network**
- **Training Dataset**
- **Experience Buffer (in host DRAM)**
- **HSS**
- **Collect Experiences**

- **Observation Vector**
- **State**

- **State**

- **Batch**

- **Max**

- **Sibyl Policy**

- **RL Training Thread**

- **RL Decision Thread**

- **Reward**
Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

Sibyl: Overview

Evaluation of Sibyl and Key Results

Conclusion

Critical Analysis and Discussion
Evaluation Methodology (1/3)

• **Real system** with various HSS configurations
  - Dual-hybrid and tri-hybrid systems
Evaluation Methodology (2/3)

Cost-Oriented HSS Configuration

High-end SSD  Low-end HDD

Performance-Oriented HSS Configuration

High-end SSD  Middle-end SSD
• **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]
Performance Analysis

Cost-Oriented HSS Configuration

Normalized Average Request Latency

- Slow-Only
- CDE
- HPS
- Archivist
- RNN-HSS
- Sibyl
- Oracle

Data points for various workloads and configurations are shown, indicating the performance comparison across different SSD and HDD options.
Performance Analysis

Cost-Oriented HSS Configuration

Sibyl consistently outperforms all the baselines for all the workloads
Performance Analysis

Performance-Oriented HSS Configuration

- Slow-Only
- CDE
- HPS
- Archivist
- RNN-HSS
- Sibyl
- Oracle

Normalized Average Request Latency

- High-end SSD
- Mid-end SSD

Safari
Performance Analysis

Performance-Oriented HSS Configuration

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

Critical Analysis and Discussion
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*
Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

Sibyl: Overview

Evaluation of Sibyl and Key Results

Conclusion

Critical Analysis and Discussion
**Strengths**

- **First work** to use RL for data placement in HSS

- Sibyl is evaluated on **real HSS configurations**

- Sibyl is **open source** -> Enables future research on ML-based techniques for HSS

- **Explanability analysis** is critical for ML-driven architectures
  - Sibyl’s actions are explainable

- Sibyl is **easily extensible** to different HSS configurations
Weaknesses

• Sibyl does not consider other optimization objectives such as energy efficiency, lifetime of storage devices

• Sibyl’s promotion and eviction policies are less adaptive to changes in workload access patterns and HSS conditions

• Sibyl stores metadata for each page, which can add a significant storage overhead

• Sibyl’s hyperparameters are tuned for the chosen workloads -> may not work across other workloads
Discussion (1/3)

• How do we simultaneously optimize multiple objectives (e.g., performance, energy efficiency, lifetime of storage devices) in HSS?
  - Optimizing multiple objectives may cause sub-optimal results
    • Higher performance improvement comes from fast device utilization, which reduces the lifetime of the fast device
  - A holistic reward function that carefully considers critical factors affecting each objective
    • I/O request latency for performance
    • Energy consumption of application reads and writes, and data migrations performed by the HSS
    • Number of writes performed on each storage device

  - Designing a good reward function remains a key challenge
Discussion (2/3)

- Can we optimize other policies of HSS using RL? What are the challenges?
  
  - Data placement, eviction and promotion have different actions
  
  - Reward needs to be different for each policy
  
  - How can we make these policies work harmoniously for the overall improvement of the HSS?
    - An RL agent to optimize each policy in HSS
    - Choose the input features and reward for coordination between the agents
Discussion (3/3)

• Security vulnerability in ML-based approaches?
  - Adversarial attacks to cause denial of service
  - Execute malicious workloads in parallel with real workloads
  - Mislead Sibyl to learn access patterns that are different from the real access patterns
  - Sibyl makes inaccurate decisions
  - Implications on performance and storage device lifetimes

• Solutions??
  - Use techniques such as Proximal Policy Optimization (PPO) and Gradient Masking
  - Improves training stability by limiting the changes to the policy at each training epoch
  - May not work for workloads with large variations in access patterns
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
Seminar in Computer Architecture
02.05.2024
BACKUP
Workloads: Analysis

Figure 3: Randomness and hotness characteristics of real-world MSRC workloads [91]

Figure 4: Timeline of accessed logical addresses and request sizes during the execution of rsrch_0 workload
**RL State**

- *k-dimensional* tuple of features
- Feature selection is performed to select only the most correlated features that affect data placement

\[ O_t = (size_t, type_t, intr_t, cnt_t, cap_t, curr_t) \]

- Divide the states into a small number of bins to reduce the state space

<table>
<thead>
<tr>
<th>Features</th>
<th>Description</th>
<th># of bins</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_t )</td>
<td>Size of the incoming request</td>
<td>8</td>
</tr>
<tr>
<td>( x_t )</td>
<td>Type of the incoming request</td>
<td>2</td>
</tr>
<tr>
<td>( r_t )</td>
<td>Access interval of the page</td>
<td>64</td>
</tr>
<tr>
<td>( f_t )</td>
<td>Access frequency of the page</td>
<td>64</td>
</tr>
<tr>
<td>( c_t )</td>
<td>Remaining capacity in the fast device</td>
<td>8</td>
</tr>
<tr>
<td>( p_t )</td>
<td>Current page placement</td>
<td>2</td>
</tr>
</tbody>
</table>
Reward

• For every action at time-step \( t \), Sibyl gets a reward from the environment at time-step \( t + 1 \)
• **Reward** acts as a **feedback** to the agent’s past action
• **Request latency** faithfully captures the status of the hybrid storage system
• **Penalty** value is chosen to prevent the agent from aggressively servicing all the requests from the faster device

\[
R = \begin{cases} 
  \frac{1}{L_t} & \text{if no eviction} \\
  \max(0, \frac{1}{L_t} - R_p) & \text{if an eviction happens}
\end{cases}
\]

\( L_t \) = latency of the request
\( R_p \) = eviction penalty

SAFARI
Training and Inference Network

• Training and inference network **allow parallel execution**

• Observation vector as the input

• Produces probability distribution of Q-values
Hyper-parameter Tuning

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

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Design Space</th>
<th>Chosen Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discount factor ($\gamma$)</td>
<td>0-1</td>
<td>0.9</td>
</tr>
<tr>
<td>Learning rate ($\alpha$)</td>
<td>$1e^{-5} - 1e^0$</td>
<td>$1e^{-4}$</td>
</tr>
<tr>
<td>Epsilon factor decay rate ($\epsilon$)</td>
<td>0-1</td>
<td>0.001</td>
</tr>
<tr>
<td>Batch size</td>
<td>64-256</td>
<td>128</td>
</tr>
<tr>
<td>Buffer size</td>
<td>10-10000</td>
<td>1000</td>
</tr>
<tr>
<td>Initial collect steps</td>
<td>100-10000</td>
<td>1000</td>
</tr>
<tr>
<td>Hidden_fc_layer1_params</td>
<td>1-100</td>
<td>20</td>
</tr>
<tr>
<td>Hidden_fc_layer2_params</td>
<td>1-100</td>
<td>30</td>
</tr>
<tr>
<td># of atoms for C51 probability</td>
<td>2-100</td>
<td>51</td>
</tr>
<tr>
<td>n-step update</td>
<td>2-64</td>
<td>2</td>
</tr>
<tr>
<td>Activation functions</td>
<td>ReLU, swish</td>
<td>swish</td>
</tr>
</tbody>
</table>
Effect of Buffer Sizes on Latency

Normalized Latency

1    10    100    1000    10000    100000

0    5     10     15     20     25
Sensitivity to Hyper-parameters

![Graphs showing sensitivity](image)

**Figure 14:** Sensitivity of Sibyl throughput to: (a) the discount factor \( \gamma \), (b) the learning rate \( \alpha \), (c) the exploration rate \( \epsilon \), averaged across 14 workloads (normalized to Fast-Only)
# Evaluation Methodology

<table>
<thead>
<tr>
<th>Host System</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD Ryzen 3 2200G with Radeon Vega Graphics, 4-core@3GHz, 128 KiB L1-I/D, 2 MiB L2, 4 MiB L3, 16GiB RDIMM DDR4 2666 MHz</td>
<td></td>
</tr>
<tr>
<td><strong>Storage Device</strong></td>
<td><strong>Characteristics</strong></td>
</tr>
<tr>
<td>H: Intel Optane SSD P4800X [93]</td>
<td>375GB, PCIe 3.0 NVMe, SLC, R/W: 2.4/2 GB/s, random R/W: 550000/500000 IOPS</td>
</tr>
<tr>
<td>L: Seagate HDD ST1000DM010 [95]</td>
<td>1TB, SATA 6Gb/s 7200rpm</td>
</tr>
<tr>
<td></td>
<td>Max. Sustained Transfer Rate: 210MB/s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Config.</th>
<th>Fast Device</th>
<th>Slow Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>H&amp;M (Performance-optimized (PO))</td>
<td>high-end (H)</td>
<td>middle-end (M)</td>
</tr>
<tr>
<td>H&amp;L (Cost-optimized (CO))</td>
<td>high-end (H)</td>
<td>low-end (L)</td>
</tr>
</tbody>
</table>
# Workload Characteristics

## Table 4: Characteristics of 14 evaluated workloads

<table>
<thead>
<tr>
<th>Workload</th>
<th>Write %</th>
<th>Read %</th>
<th>Avg. request size</th>
<th>Avg. access count</th>
<th>No. of unique requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>hm_1</td>
<td>4.7%</td>
<td>95.3%</td>
<td>15.2</td>
<td>44.5</td>
<td>6265</td>
</tr>
<tr>
<td>mds_0</td>
<td>88.1%</td>
<td>11.9%</td>
<td>9.6</td>
<td>3.5</td>
<td>31933</td>
</tr>
<tr>
<td>prn_1</td>
<td>24.7%</td>
<td>75.3%</td>
<td>20.0</td>
<td>2.6</td>
<td>6891</td>
</tr>
<tr>
<td>proj_0</td>
<td>87.5%</td>
<td>12.5%</td>
<td>38.0</td>
<td>48.3</td>
<td>1381</td>
</tr>
<tr>
<td>proj_2</td>
<td>12.4%</td>
<td>87.6%</td>
<td>42.4</td>
<td>2.9</td>
<td>27967</td>
</tr>
<tr>
<td>proj_3</td>
<td>5.2%</td>
<td>94.8%</td>
<td>9.6</td>
<td>3.6</td>
<td>12939</td>
</tr>
<tr>
<td>prxy_0</td>
<td>96.9%</td>
<td>3.1%</td>
<td>7.2</td>
<td>95.7</td>
<td>525</td>
</tr>
<tr>
<td>prxy_1</td>
<td>34.5%</td>
<td>65.5%</td>
<td>12.8</td>
<td>150.1</td>
<td>6845</td>
</tr>
<tr>
<td>rsrch_0</td>
<td>90.7%</td>
<td>9.3%</td>
<td>9.2</td>
<td>34.7</td>
<td>5504</td>
</tr>
<tr>
<td>src1_0</td>
<td>43.6%</td>
<td>56.4%</td>
<td>43.2</td>
<td>12.7</td>
<td>13640</td>
</tr>
<tr>
<td>stg_1</td>
<td>36.3%</td>
<td>63.7%</td>
<td>40.8</td>
<td>1.1</td>
<td>3787</td>
</tr>
<tr>
<td>usr_0</td>
<td>59.6%</td>
<td>40.4%</td>
<td>22.8</td>
<td>19.7</td>
<td>2138</td>
</tr>
<tr>
<td>wdev_2</td>
<td>99.9%</td>
<td>0.1%</td>
<td>8.0</td>
<td>17.7</td>
<td>4270</td>
</tr>
<tr>
<td>web_1</td>
<td>45.9%</td>
<td>54.1%</td>
<td>29.6</td>
<td>1.2</td>
<td>6095</td>
</tr>
</tbody>
</table>
Throughput Analysis

Figure 10: Request throughput (IOPS) under two different hybrid storage configurations (normalized to Fast-Only)
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.

SAFARI
Performance Analysis

Performance-Oriented HSS Configuration

Baseline policies are ineffective for many workloads even when compared to Slow-Only
Performance on Mixed Workloads

![Graphs showing performance-oriented and cost-oriented normalized average request latency for various workloads and solutions.]

Table 5: Characteristics of mixed workloads

<table>
<thead>
<tr>
<th>Mix</th>
<th>Workloads</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mix1</td>
<td>prxy_0 [91] and ntrx_rw [92]</td>
<td>Both prxy_0 and ntrx_rw are write-intensive</td>
</tr>
<tr>
<td>mix2</td>
<td>rsrch_0 [91] and oltp_rw [92]</td>
<td>rsrch_0 is write-intensive and oltp_rw is read-intensive</td>
</tr>
<tr>
<td>mix3</td>
<td>proj_3 [91] and YCSB_C [147]</td>
<td>Both proj_3 and YCSB_C are read-intensive</td>
</tr>
<tr>
<td>mix4</td>
<td>src1_0 [91] and fileserver [92]</td>
<td>Both src1_0 and fileserver have nearly equal numbers of reads and writes</td>
</tr>
<tr>
<td>mix5</td>
<td>prxy_0 [91], oltp_rw [92] and fileserver [92]</td>
<td>prxy_0 is write-intensive, oltp_rw is read-intensive, and fileserver has nearly equal numbers of reads and writes</td>
</tr>
<tr>
<td>mix6</td>
<td>src1_0 [91], YCSB_C [147] and fileserver [92]</td>
<td>src1_0 and fileserver have nearly equal numbers of reads and writes while YCSB_C is read-intensive</td>
</tr>
</tbody>
</table>
Performance on Mixed Workloads

Sibyl\textsubscript{Def} \textbf{outperforms} baseline data placement techniques by up to \textbf{27.9\%}
Performance on Mixed Workloads

Sibyl\textsubscript{Def} outperforms baseline data placement techniques by up to 27.9%

Sibyl\textsubscript{Opt} provides 7.2% higher average performance than Sibyl\textsubscript{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

(a) H&M

(b) H&L

Available capacity in fast storage
Explainability Analysis

![Bar chart showing preference for fast storage with categories such as 'hm_1', 'mds_0', 'prn_1', 'proj_0', 'proj_2', 'proj_3', 'prxy_0', 'prxy_1', 'rsrch_0', 'src1_0', 'stg_1', 'usr_0', 'wdev_2', 'web_1'].

Legend:
- H&M
- H&L

The chart compares the preference for fast storage between H&M and H&L across several categories.