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

SAFARI
SAFARI Research Group
safari.ethz.ch

ETH Zürich

TU/e Eindhoven University of Technology
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**

[https://github.com/CMU-SAFARI/Sibyl](https://github.com/CMU-SAFARI/Sibyl)
Talk Outline

Key Shortcomings of Prior Data Placement Techniques

Formulating Data Placement as Reinforcement Learning

Sybil: Overview

Evaluation of Sybil and Key Results

Conclusion
Hybrid Storage System Basics

Address Space (Application/File System View)

Storage Management Layer

Logical Pages

- Read
- Write

Promotion

Eviction

Fast Device

Slow Device

Hybrid Storage System
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

Workload Changes

Prior data placement techniques consider only a few workload characteristics that are statically tuned.
Lack of Adaptivity

Workload Changes

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

![Graph showing normalized average request latency for different workloads: hm_1, prn_1, usr_0, wdev_2, AVG. The graph compares CDE, RNN-HSS, and Oracle.]
Lack of Adaptivity

Workload Changes

Prior data placement techniques consider only a few workload characteristics that are statically tuned.
Lack of Adaptivity

Workload Changes

Prior data placement techniques consider only a few workload characteristics that are statically tuned.
Lack of Adaptivity

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 Adaptivity

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)

![Graph showing normalized average request latency for different configurations](image)
Lack of Adaptivity

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

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

Change in storage configuration

Dual-HSS
Lack of Extensibility

**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

Sybil 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

Sybil: Overview

Evaluation of Sybil and Key Results

Conclusion
Basics of Reinforcement Learning (RL)

Agent

Environment

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

Agent

State ($S_t$)

Reward ($R_{t+1}$)

Action ($A_t$)

Environment

Sibyl

Features of the current request and system

Request latency (of last served request)

Select storage device to place the current page

Hybrid Storage System

Features of the current request and system

Request latency (of last served request)

Select storage device to place the current page
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
Sibyl Execution

RL Decision Thread

State, Reward, and Action Information → Periodic Policy Weight Update

Storage Request (from OS) → Data Placement Decision

RL Training Thread

Asynchronous Execution

Sibyl
Sibyl Design: Overview

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

- **Inference Network**
  - Max

- **Observation Vector**
  - State

- **State**

- **Experience Buffer (in host DRAM)**
  - Batch
  - RL Training Thread

- **RL Decision Thread**
  - Reward
  - Collect Experiences

- **Sibyl Policy**
  - Action

- **HSS**
  - Storage Request (from OS)

- **Training Dataset**
  - Batch

- **Periodic Policy Update**
  - RL Training Network

- **RL Decision**
  - Periodic Policy Weight Update
RL Decision Thread

**Inference Network**

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

**RL Training Thread**

**RL Decision Thread**

- **Observation Vector**
  - **State**
- **Inference Network**
  - **Max**
  - **Sibyl Policy**
- **Experience Buffer (in host DRAM)**
  - **Action**
  - **Reward**
  - **Collect Experiences**
- **HSS**

**Storage Request (from OS)**

**SAFARI**
RL Decision Thread

- **Observation Vector**
  - Storage Request (from OS)
  - States

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

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

- **Training Network**
  - Training Dataset
  - Batch

- **Periodic Policy Weight Update**

- **RL Training Thread**

- **RL Decision Thread**

(SAFARI)
RL Decision Thread

- RL Training Thread
  - Training Network
  - Periodic Policy Weight Update
  - Training Dataset
  - Batch

- RL Decision Thread
  - State
  - Observation Vector
  - Inference Network
  - Sibyl Policy
  - Max
  - Action
  - HSS
  - Reward
  - Experience Buffer (in host DRAM)
  - Collect Experiences
  - Storage Request (from OS)
RL Decision Thread

- **Observation Vector**
- **State**
- **Inference Network**
- **Max**
- **Action**
- **HSS**
- **Reward**
- **Collect Experiences**
- **Experience Buffer (in host DRAM)**
- **Periodic Policy Weight Update**
- **Training Network**
- **Training Dataset**
- **Batch**

**SAFARI**
RL Decision Thread

Observation Vector

State

Training Network

Periodic Policy Weight Update

Training Dataset

Batch

Experience Buffer (in host DRAM)

RL Decision Thread

Inference Network

Max

Sibyl Policy

Action

HSS

Reward

Collect Experiences

Storage Request (from OS)
RL Training Thread

Periodic Policy
Weight Update

Training Network

Training Dataset

Batch

Experience Buffer
(in host DRAM)

RL Training Thread

RL Decision Thread

Observation Vector

Inference Network

Max

Sibyl Policy

State

Action

Reward

State

Collect Experiences

HSS
Periodic Weight Transfer

Training Network

Periodic Policy Weight Update

Observation Vector

State

Inference Network

Max

Sibyl Policy

RL Decision Thread

Experience Buffer (in host DRAM)

HSS

Collect Experiences

Reward

Action

Batch

Training Dataset

Periodic Weights update

Training Network

RL Training Thread

Storage Request (from OS)
Training and Inference Network

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

• Observation vector as the input

• Produces probability distribution of Q-values
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
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]  \(\text{Heuristic-based}\)
  - HPS [Meswani+, HPCA’15]  \(\text{Heuristic-based}\)
  - Archivist [Ren+, ICCD’19]  \(\text{Learning-based}\)
  - RNN-HSS [Doudali+, HPDC’19]  \(\text{Learning-based}\)
Performance Analysis

Cost-Oriented HSS Configuration

Normalized Average Request Latency

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

Data showing performance analysis for various configurations and workloads, with metrics such as Slow-Only, CDE, HPS, Archivist, RNN-HSS, Sibyl, and Oracle.
Sibyl consistently outperforms all the baselines for all the workloads
Performance Analysis

Performance-Oriented HSS Configuration

Normalized Average Request Latency

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

[Graph showing performance analysis with various labels and normalized latency values]
Performance Analysis

Performance-Oriented HSS Configuration

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

**SAFARI**
Performance Analysis

Performance-Oriented HSS Configuration

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

Normalized Average Request Latency

Data categories include:
- Slow-Only
- CDE
- HPS
- Archivist
- RNN-HSS
- Sibyl
- Oracle

(Network Performance Metrics and Evaluations)
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 Chart]

- **Heuristic\textsubscript{Tri-hybrid}**
- **Sibyl\textsubscript{Tri-hybrid}**

**Normalized Average Request Latency**

<table>
<thead>
<tr>
<th>Device</th>
<th>hm_1</th>
<th>mds_0</th>
<th>prn_1</th>
<th>proj_0</th>
<th>proj_2</th>
<th>proj_3</th>
<th>prxy_0</th>
<th>prxy_1</th>
<th>rsrch_0</th>
<th>src1_0</th>
<th>stg_1</th>
<th>usr_0</th>
<th>wdev_2</th>
<th>web_1</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 (1/2)

• **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
• 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 Sybil's decision making
  - Explain Sibyl’s actions for different workload characteristics and device configurations
More in the Paper (2/2)

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

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


https://github.com/CMU-SAFARI/Sibyl
Conclusion

• **We introduced Sibyl**, the first reinforcement learning-based data placement technique in hybrid storage systems that provides
  - Adaptivity
  - Easily extensibility
  - Ease of design and implementation

• **We evaluated Sibyl on real systems** using many different workloads
  - Sibyl **improves performance by 21.6%** compared to the best prior data placement policy in a dual-HSS configuration
  - In a tri-HSS configuration, Sibyl **outperforms** the state-of-the-art data placement policy by **48.2%**
  - Sibyl achieves **80% of the performance** of an oracle policy with a storage overhead of only **124.4 KiB**

[https://github.com/CMU-SAFARI/Sibyl](https://github.com/CMU-SAFARI/Sibyl)
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
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.
Baseline policies are ineffective for many workloads even when compared to Slow-Only.
Performance on Mixed Workloads

![Graph showing performance on mixed workloads with normalized average request latency. The graph is divided into Performance-Oriented and Cost-Oriented sections. The bars represent different workloads (mix1 to mix6) and systems (Slow-Only, CDE, HPS, Archivist, RNN-HSS, Sibyl, Sibyl\_Def, Sibyl\_Opt, Oracle).]

SAFARI
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.
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
Explainability Analysis

Preference for Fast Storage

- 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

H&M
H&L

SAFARI
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

49th ISCA 2022, New York, USA