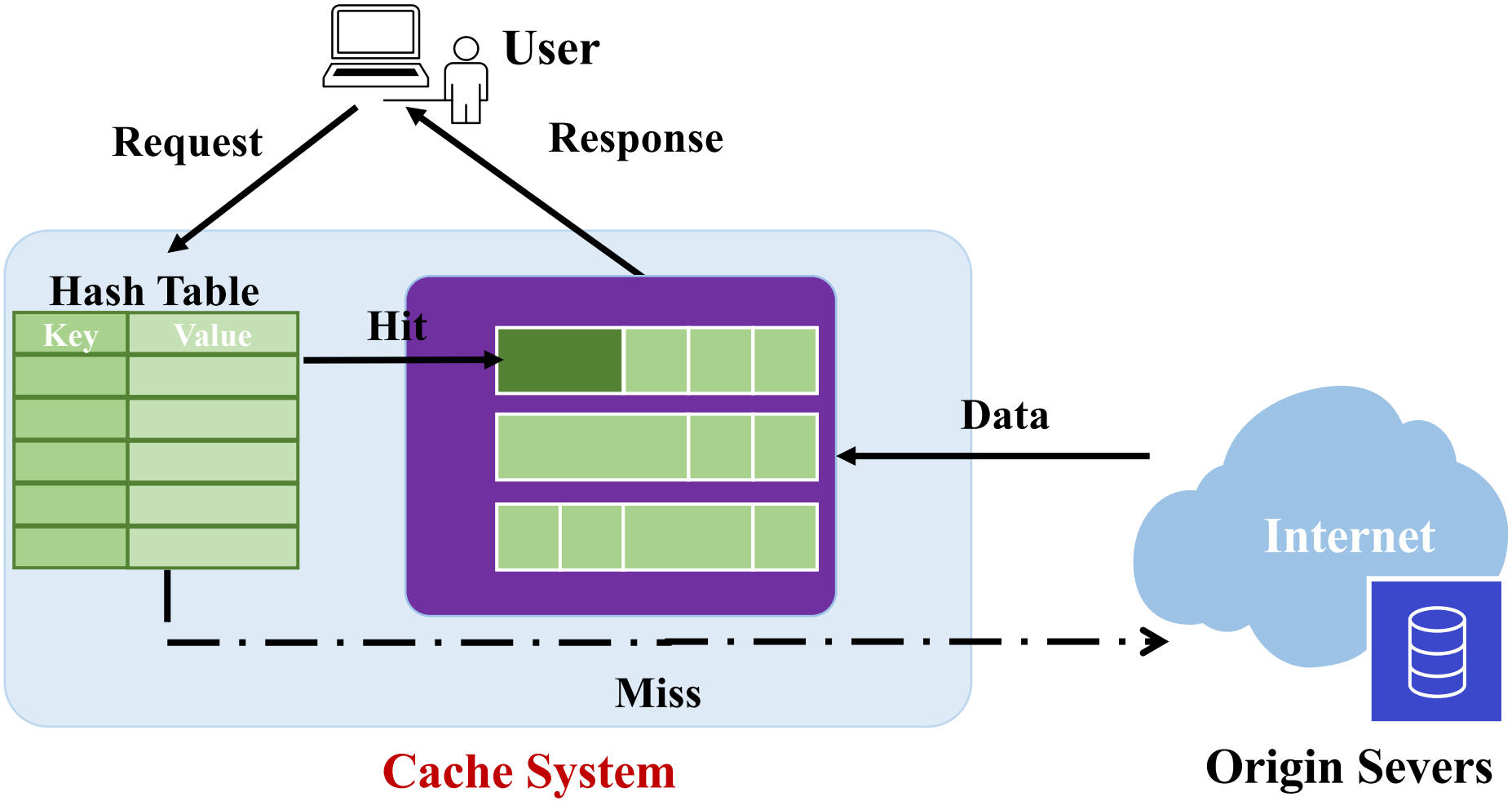
3L-Cache: Low Overhead and Precise Learning-based Eviction Policy for Caches

Wenbin Zhou, Zhixiong Niu, Yongqiang Xiong, Juan Fang, Qian Wang

Beijing University of Technology, Microsoft Research

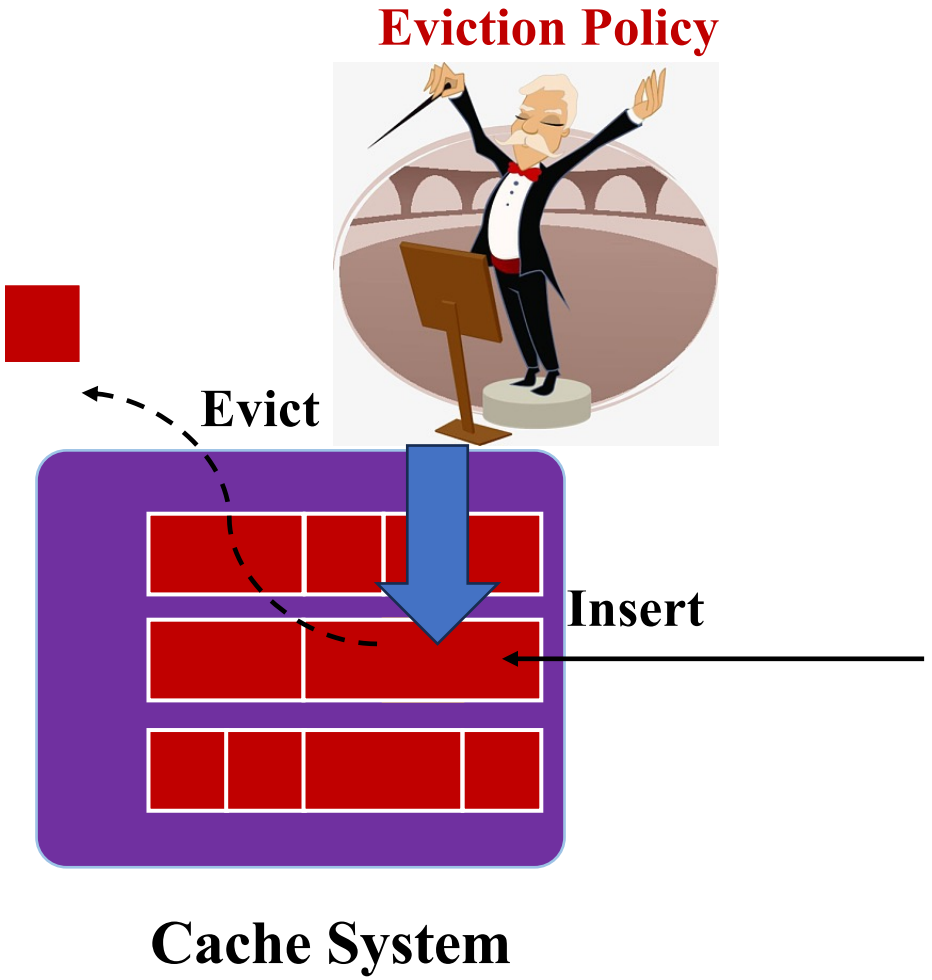
Introduction

Cache system : Accelerating response time & Reducing network traffic.



Introduction

An **Eviction Policy** is a set of rules that determine which data should be removed from the cache when the storage space is full.

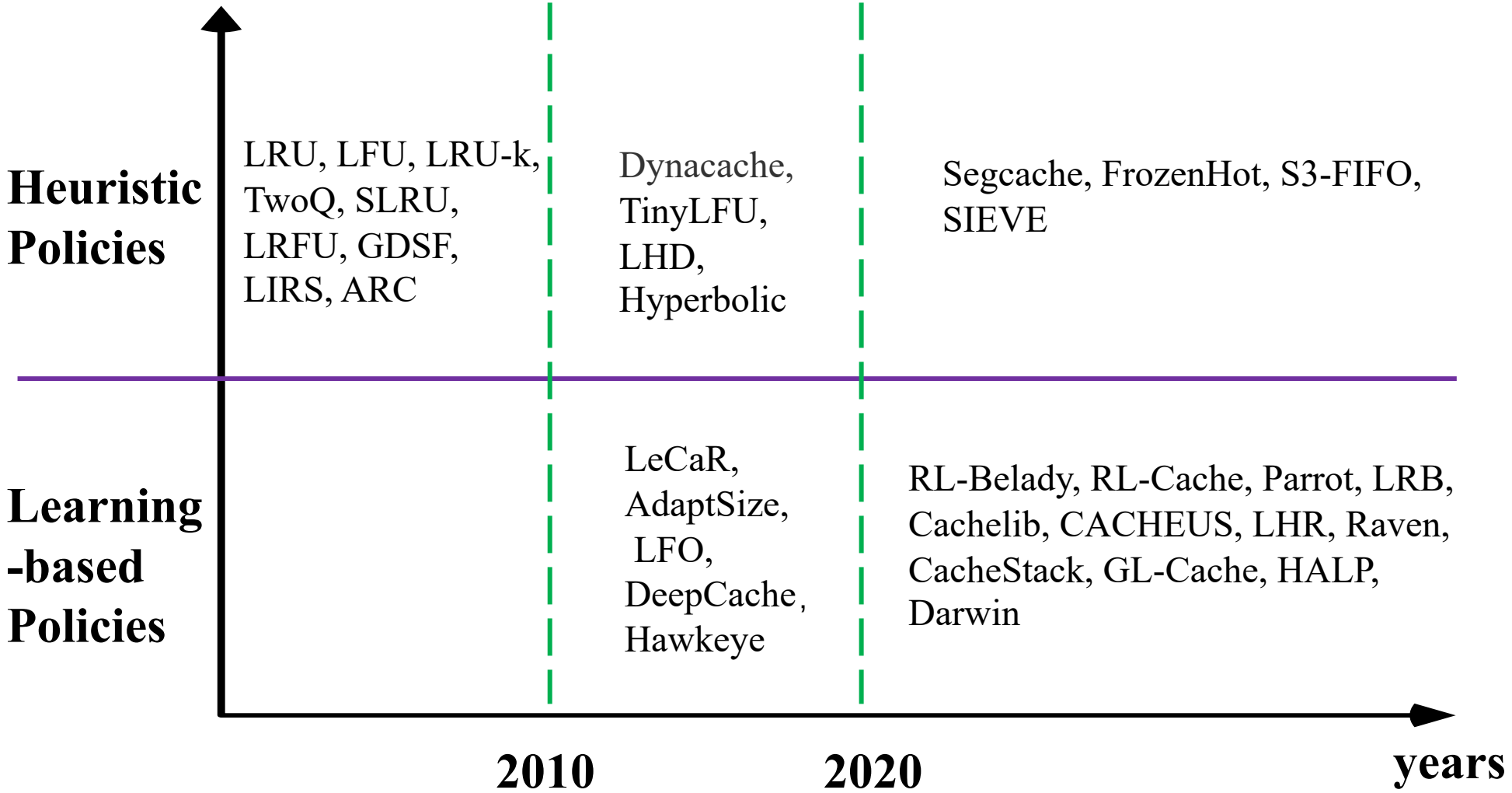


Metrics

- Object miss ratio: response speed to requests
- Byte miss ratio: reduced network traffic

Introduction

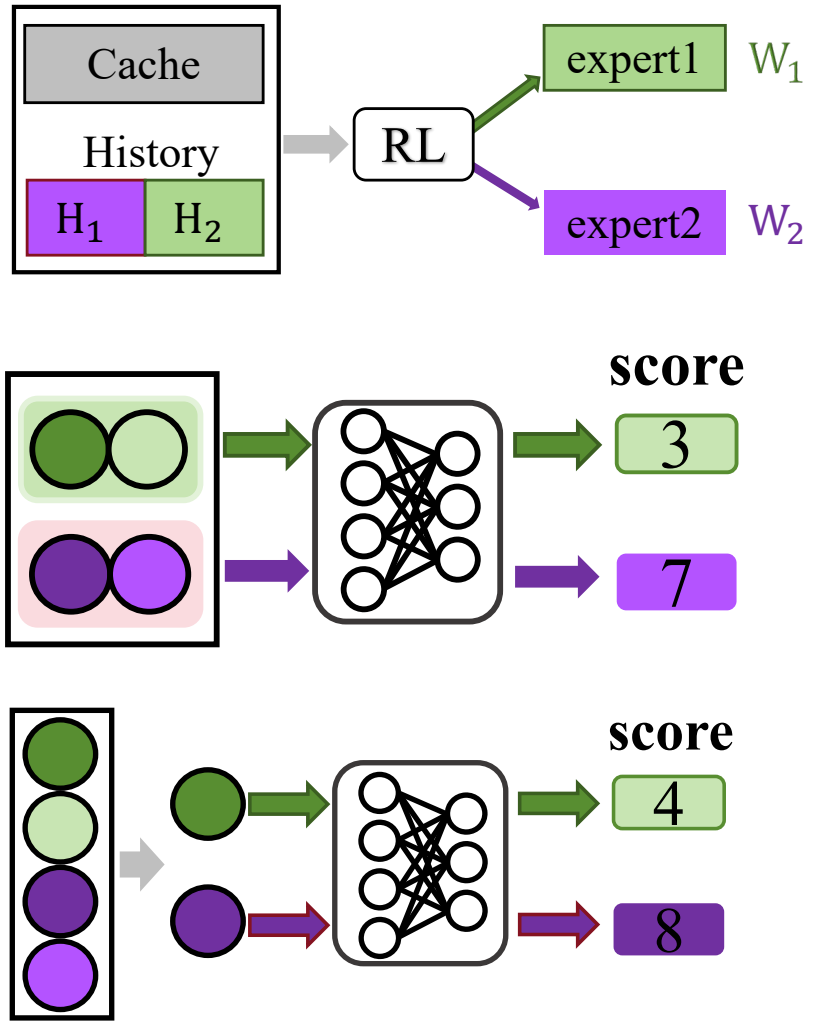
Eviction Policies



Introduction

Three categories of learning-based policies

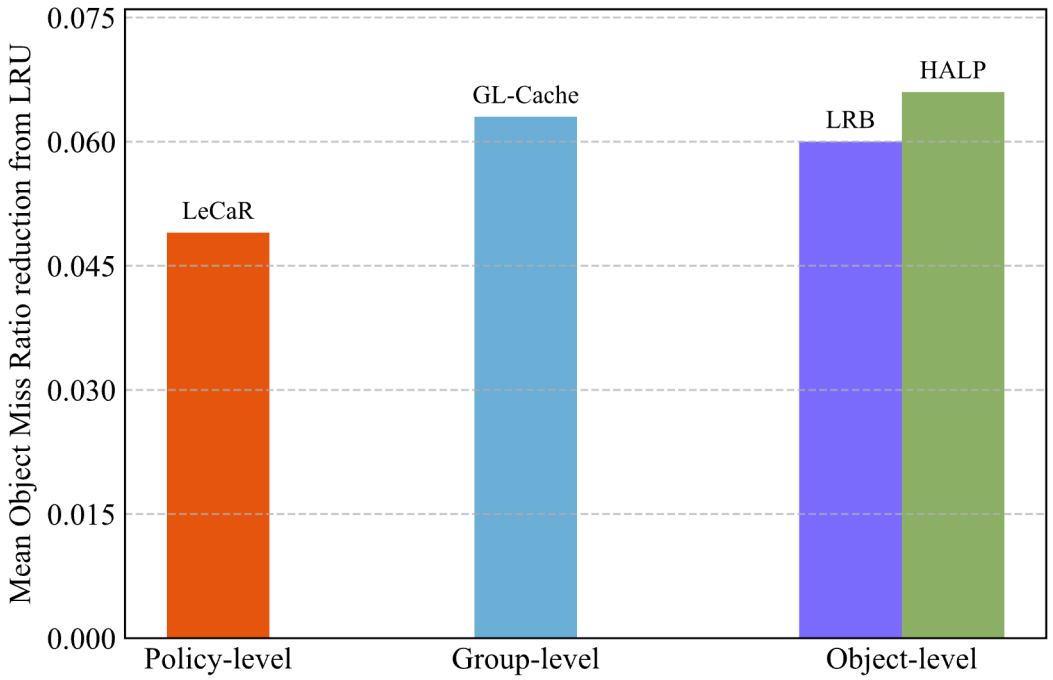
- Policy-level learning
  - Uses RL to select among expert strategies such as LRU and LFU
- Group-level learning
  - Predicts the eviction weight of groups and evicts objects based on group-level eviction weights.
- Object-level learning
  - Samples multiple objects and predicts individual eviction weights to make eviction decisions.



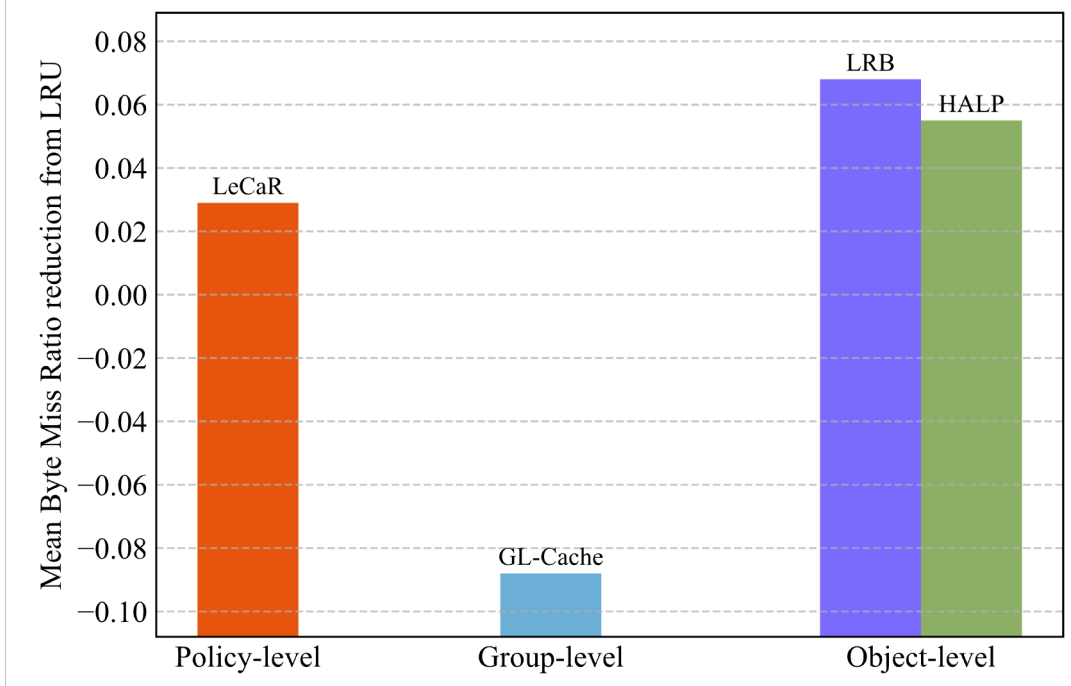
Deployment Bottlenecks

Policy-level learning & Group-level learning: **High miss ratio!**

Object-level learning is stable and performs well on two miss ratios!



Mean object miss ratio reduction from LRU

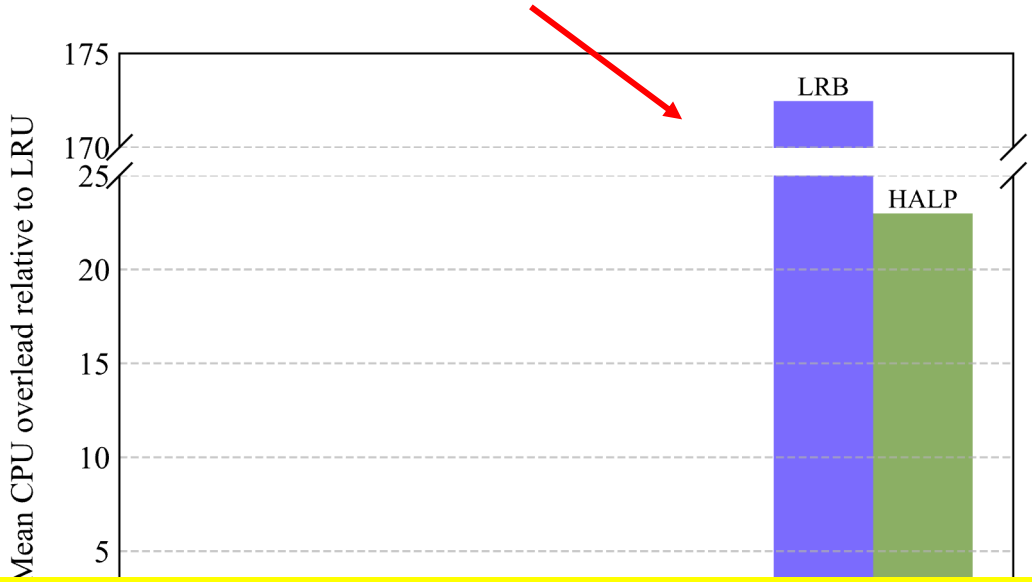


Mean byte miss ratio reduction from LRU

Deployment Bottlenecks

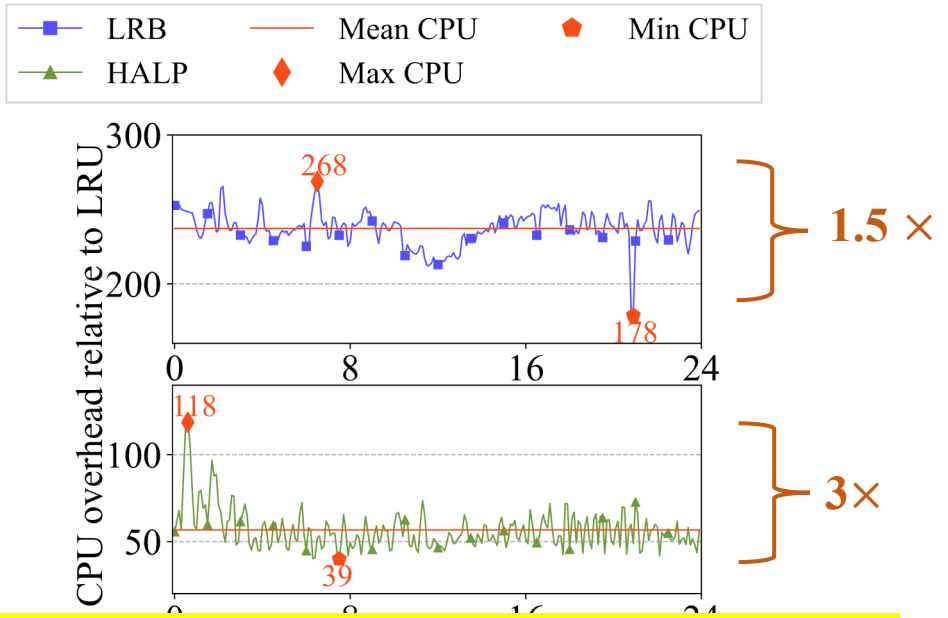
Object-level learning policies encounter deployment bottlenecks

Significantly higher CPU overhead (~172x) than LRU



Mean CPU overhead relative to LRU

Frequent and significant CPU fluctuations



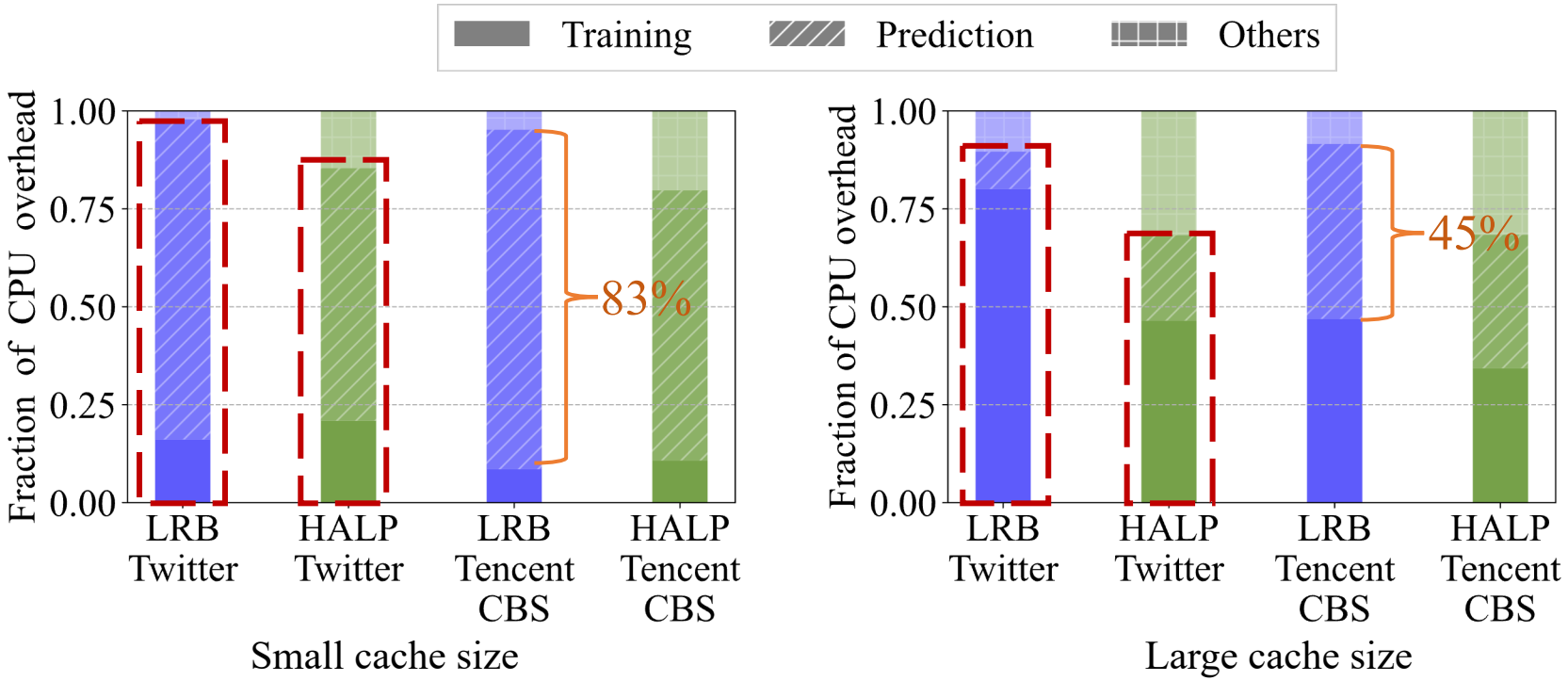
Real-time CPU overhead relative to LRU

Question: Can we significantly reduce the CPU overhead of object-level learning while maintaining its miss ratio advantage?

Breaking Down the Overhead

Training and Prediction

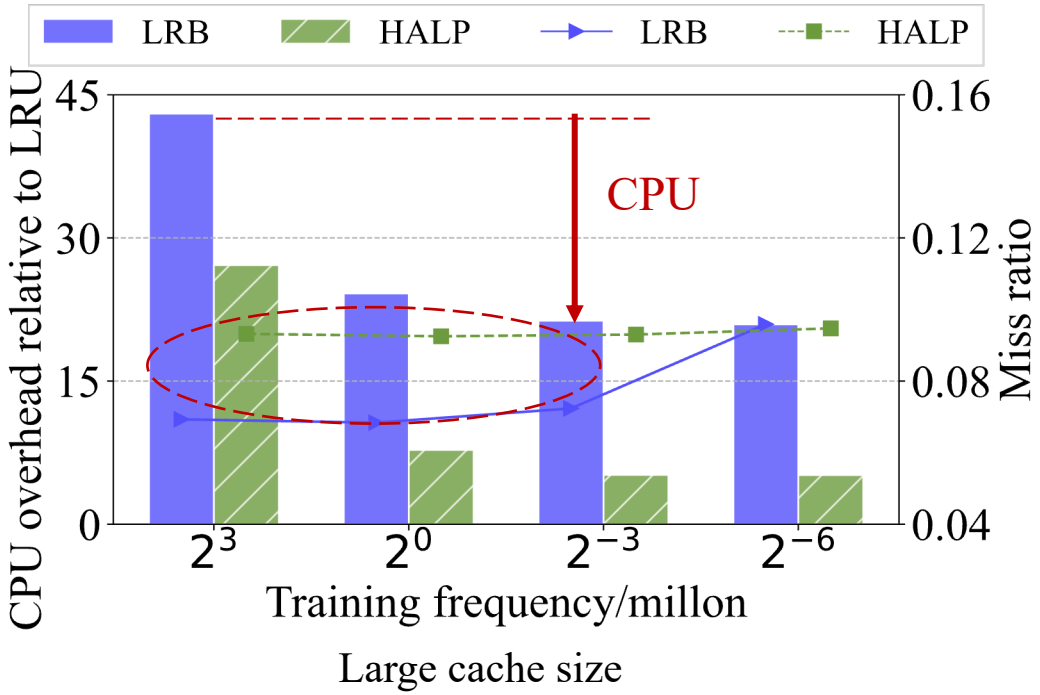
- Both occupy the majority of computation resources;
- Proportions vary with cache size.



Breaking Down the Overhead

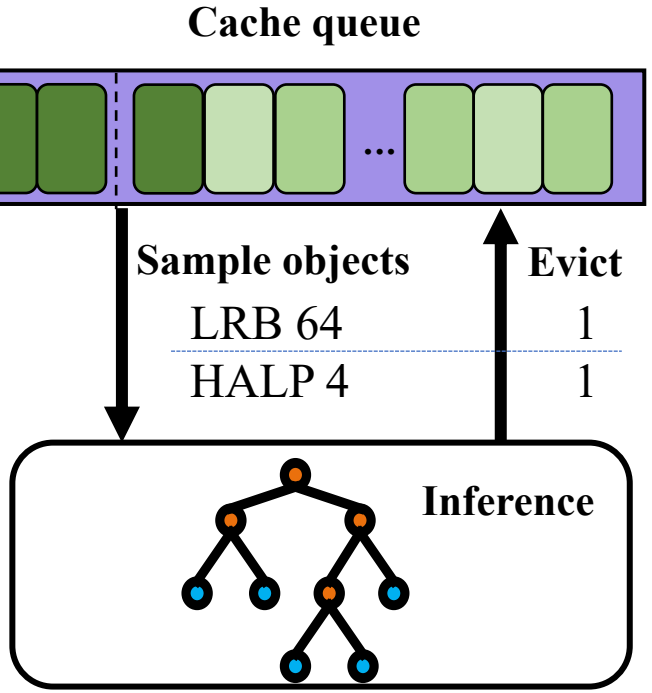
Training

- High training frequency
  - Lower training frequency → reduced CPU overhead
  - Miss ratio only increases slightly



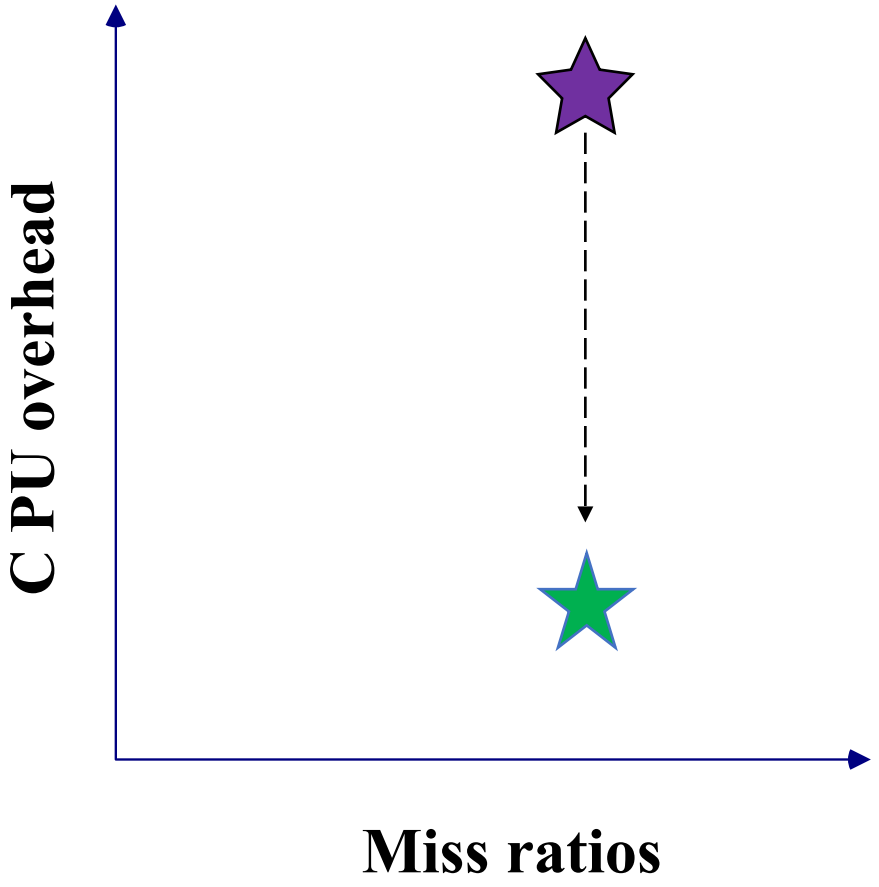
Prediction

- Low eviction ratio
  - LRB eviction ratio is only **1.56%**;
  - HALP eviction ratio is only **25%**;

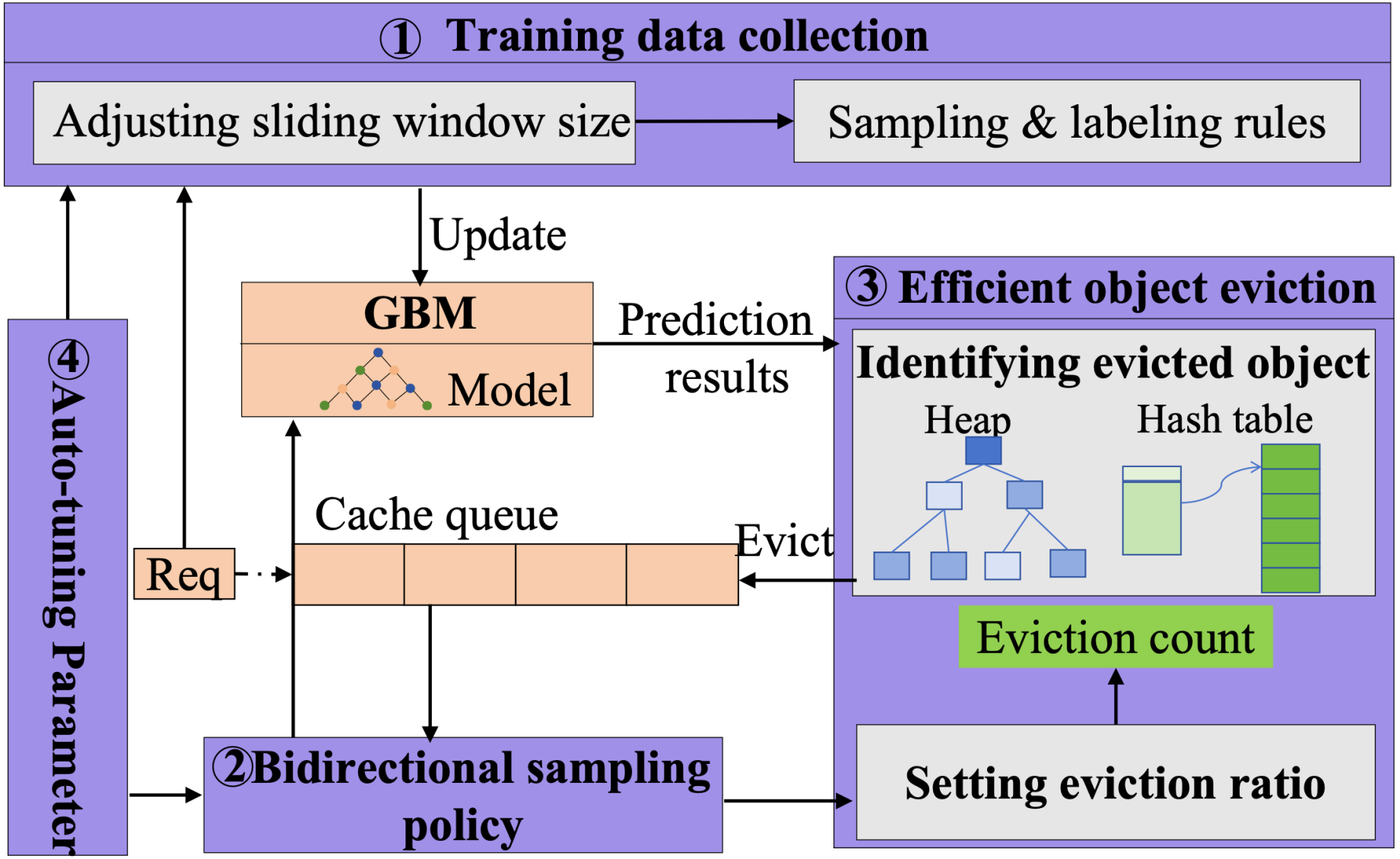


Key Challenges

- C-1: How to reduce overhead waste in training without reducing model accuracy?
- C-2: How to reduce the prediction overhead without sacrificing miss ratios?
- C-3: How to improve generalizability across traces?



3L-Cache Design Overview



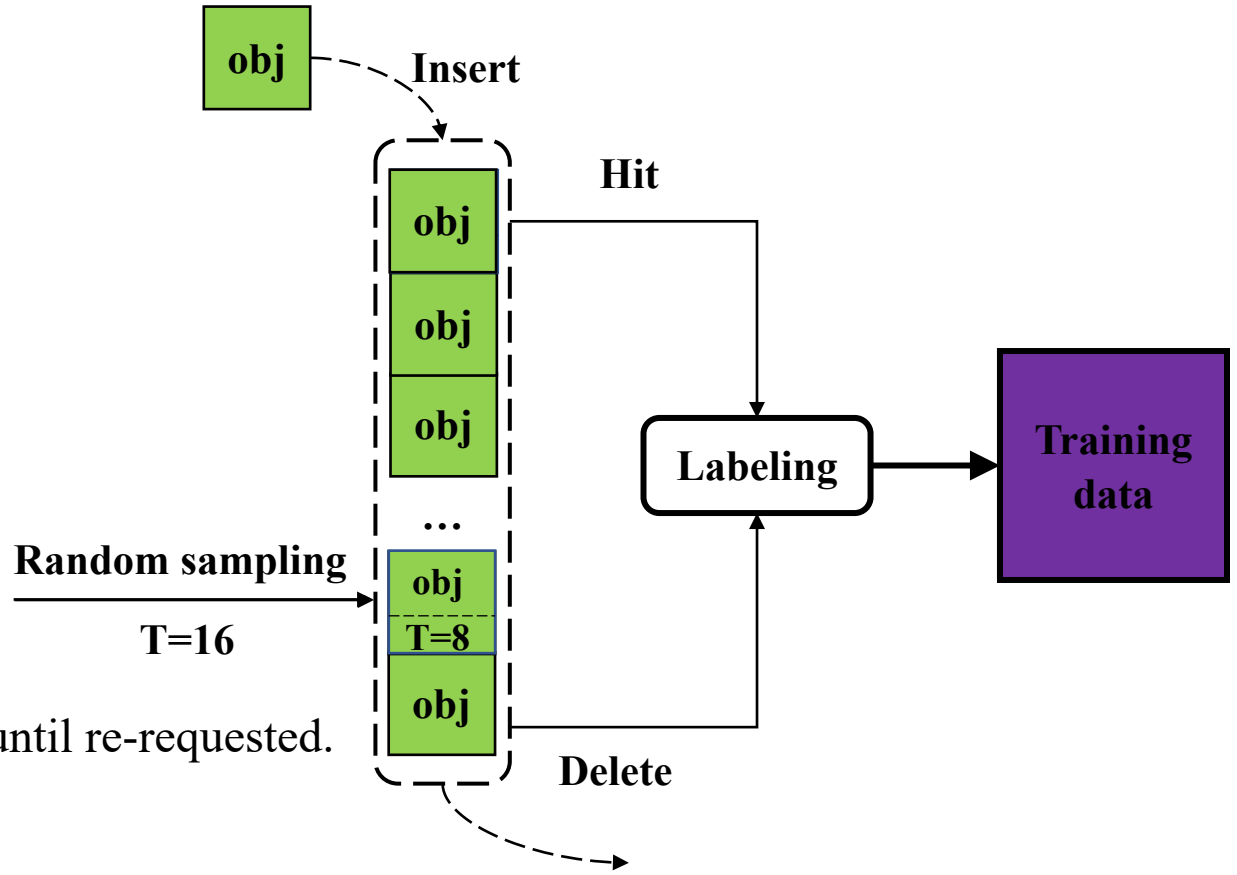
S-1: Online Training Design

Sliding Window Adjustment

- Using h_{sw} to enable coarse-grained, rapid adjustment;
- Window size = h_{sw} × cache queue size;

Efficient Sampling & Labeling

- Sampling: Randomly sample one object per request.
- Recording: Store sampling time; no duplicate recording until re-requested.
- Labeling: Mark re-requested objects and record interval.
- Updating: Update every M labeled entries (M ∈ [32K, 128K]).



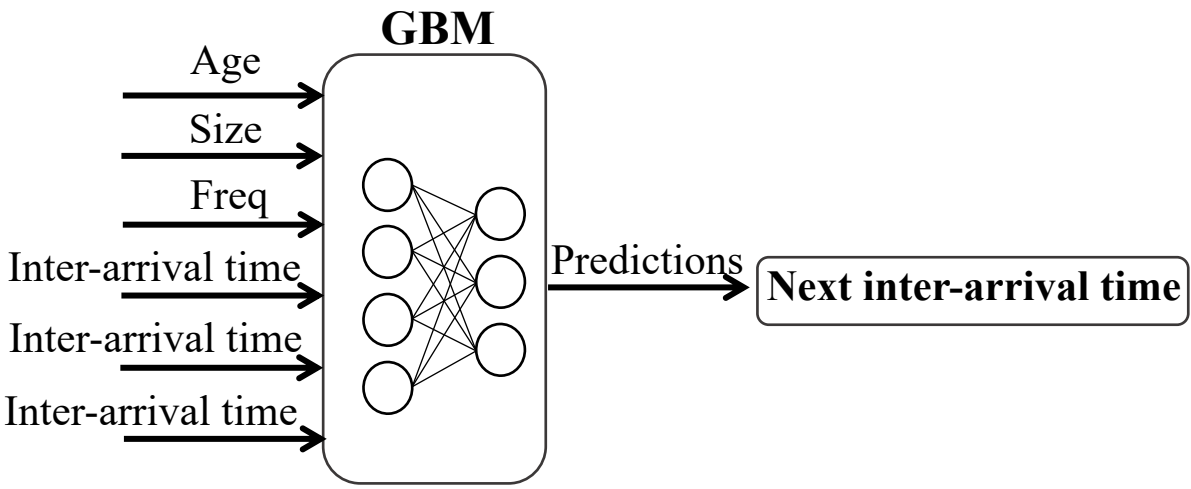
S-1: Online Training Design

Model: Gradient Boosting Machine (GBM)

Loss: Cross entropy

Features

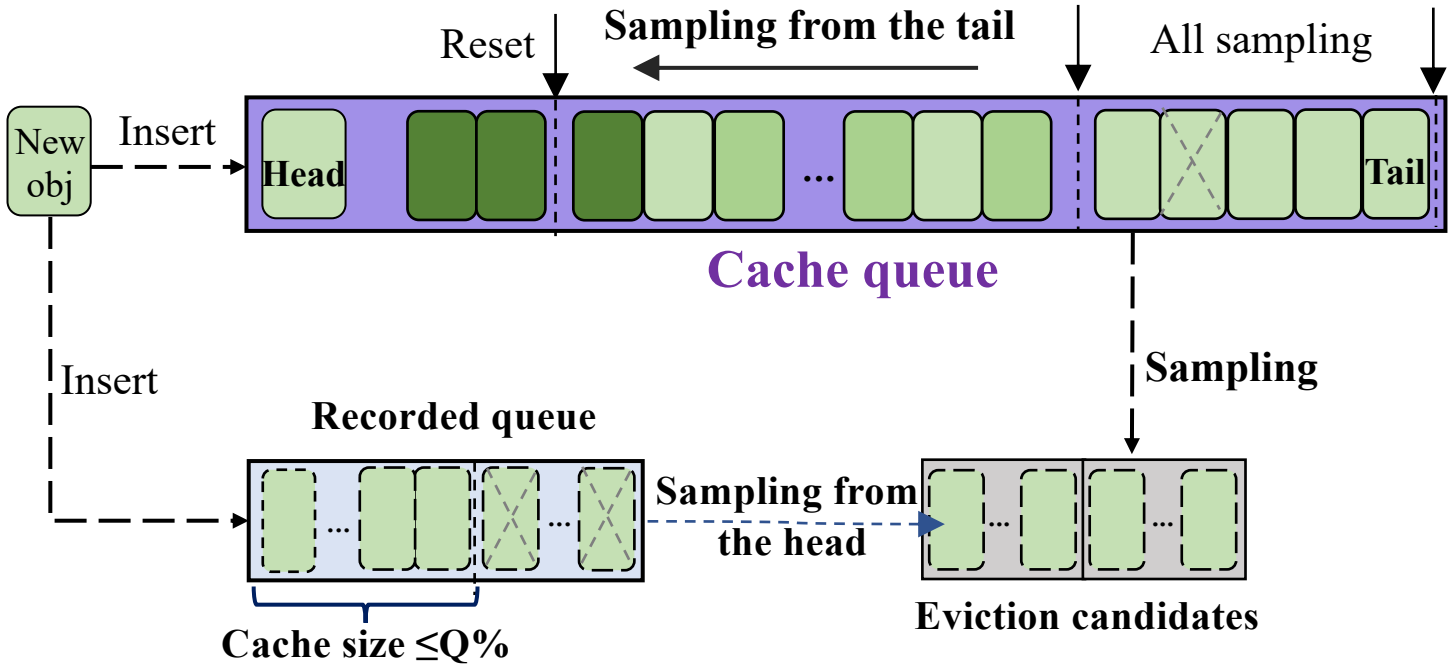
- Age: Time since last access
- Size: Object size
- Freq: Number of access in cache
- 3 latest inter-arrival times: Time between access



S-2.1: Cached Object Eviction Design- Sampling

Bidirectional Sampling Policy

- Sampling from the tail
- Sampling from the head



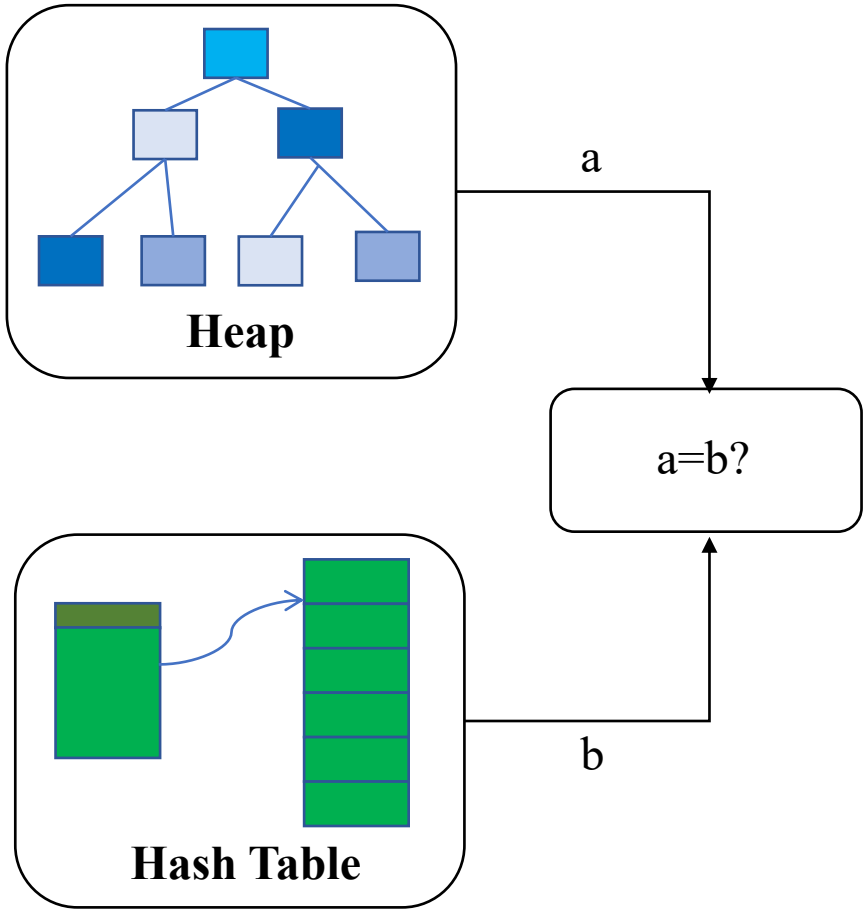
S-2.2: Cached Object Eviction Design- Object Eviction

Use Accumulate Prediction Results

Efficient Management

- Uses a heap structure to select eviction objects
- Uses a hash table to maintain data validity in the heap

Achieves a 50% Eviction Ratio



S-3: Parameter Auto-Tuning

Name	Description	Initial value	Trigger event	Action
h_{sw}	Coarse-grained adjustment of the sliding window to quickly find the appropriate size	2	Model update during training	h_{sw} += 1
f	Reduce the probability of sampling popular objects in targeted reset sampling.	1	End of a round of sampling from the tail	Update to the 99th percentile of the access times of the evicted objects in the round.
x	Determine the proportion of unpopular objects at the head of the cache queue.	1	End of a round of sampling from the tail	x += 1, or x −= 1
Q	Determine the proportion of newly arrived objects in the cache queue.	2	End of a round of sampling from the tail	Q += 1, or Q/= 2
n	The proportion of newly arrived objects in the eviction process does not deviate significantly from Q%.	2	End of a round of sampling from the tail	min (1024, 1% · L + 2)

Evaluation

Datasets: 8 different sources, 4,855 traces

Datasets	Approx time	# Traces	Cache Type	# Request (million)	# Object (million)
CloudPhysics	2015	106	Block	2214	492
TencentPhoto	2018	2	Object	5650	1038
Wikipedia CDN	2016-19	3	Object	8400	126
Tencent CBS	2020	4030	Block	33690	551
Twitter	2020	54	KV	195441	10650
Alibaba	2020	652	Block	19676	1702
Meta KV	2022	5	KV	1644	82
Meta CDN	2023	3	Object	231	76

12 Cache eviction policies

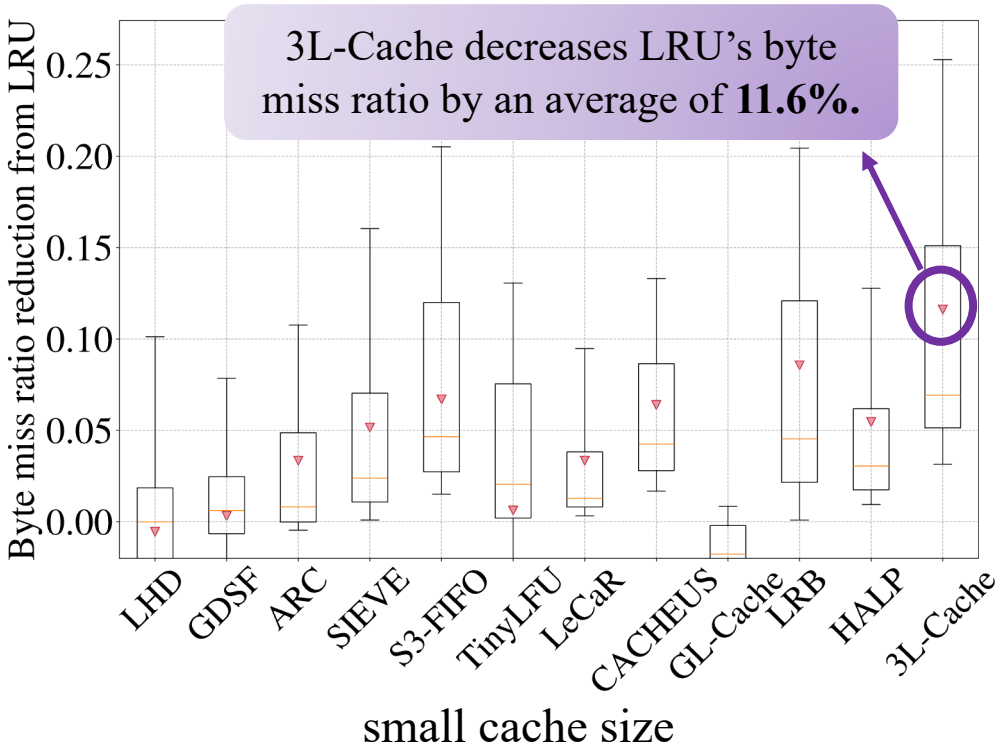
- Heuristic Policy
  - LHD
  - GDSF
  - ARC
  - SIEVE
  - S3-FIFO
  - TinyLFU
- Learning-based Policy
  - LeCaR
  - CACHEUS
  - GL-Cache
  - LRB
  - HALP
  - 3L-Cache (**Ours**)

Byte Miss Ratio

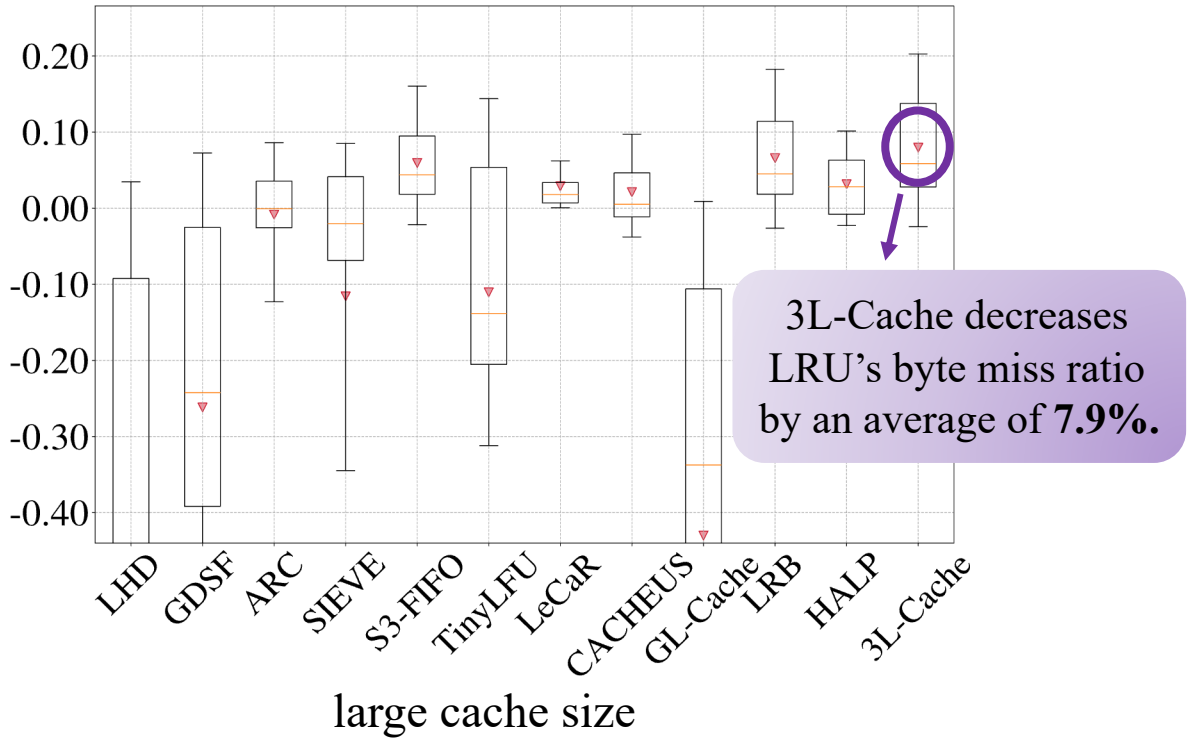
3L-Cache achieves the lowest byte miss ratio for **69.8%** of the traces with small cache sizes.

3L-Cache achieves the lowest byte miss ratio for **41.2%** of the traces with large cache sizes.

Better



small cache size



large cache size

Tencent CBS, 4030 traces (33690 million requests)

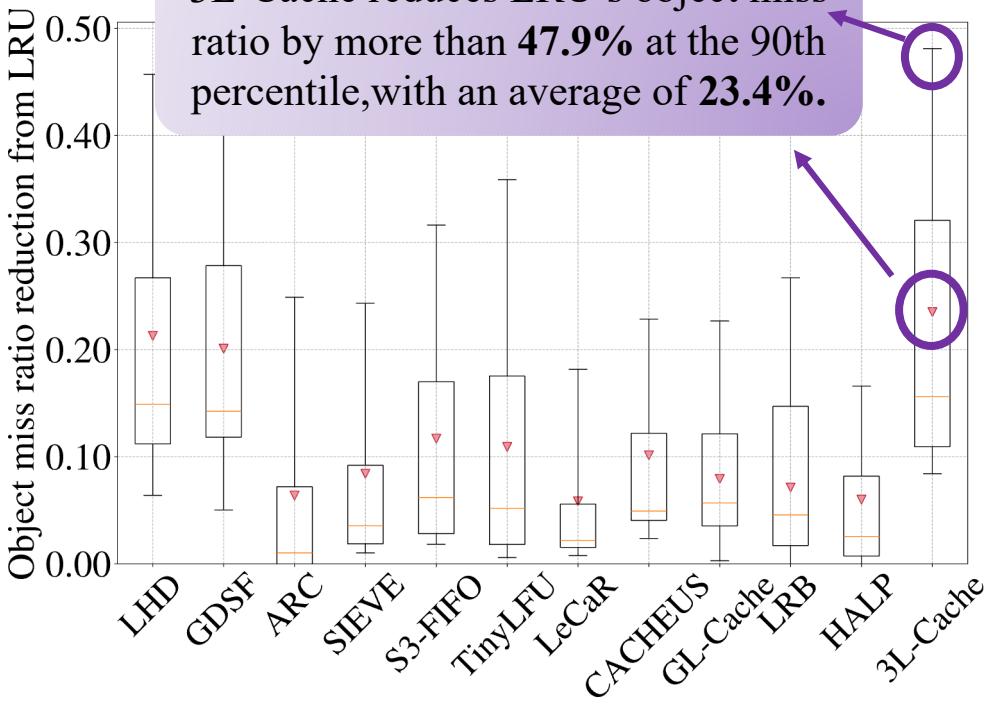
Object Miss Ratio

3L-Cache achieves the lowest object miss ratio for **66.3%** of the traces with small cache sizes.

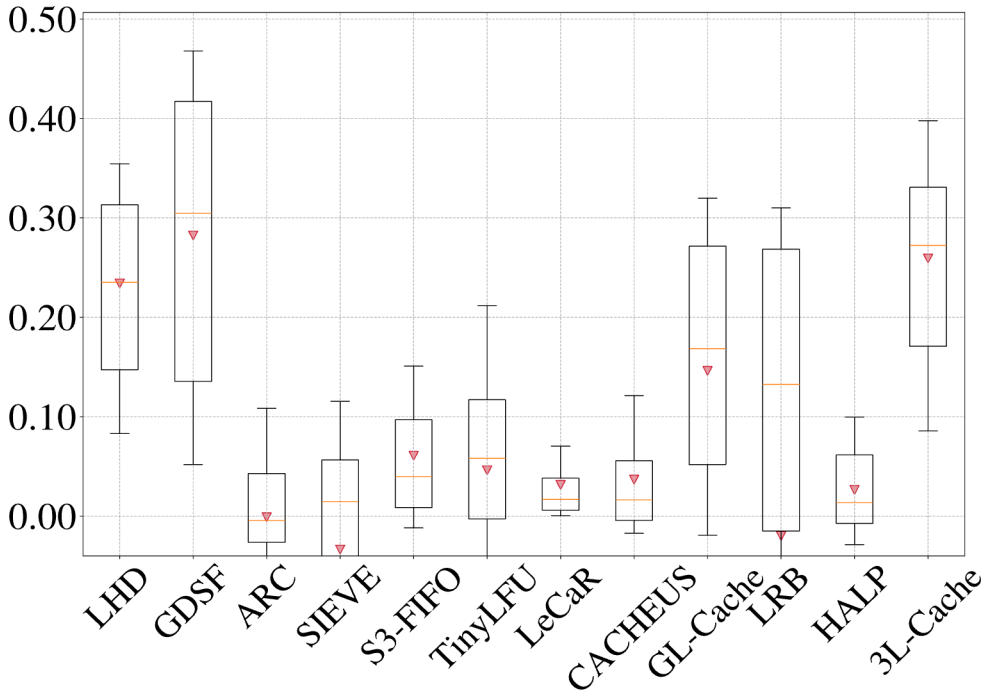
3L-Cache achieves the lowest object miss ratio for **29.3%** of the traces with large cache sizes.

Better

3L-Cache reduces LRU's object miss ratio by more than **47.9%** at the 90th percentile,with an average of **23.4%.**



small cache size



large cache size

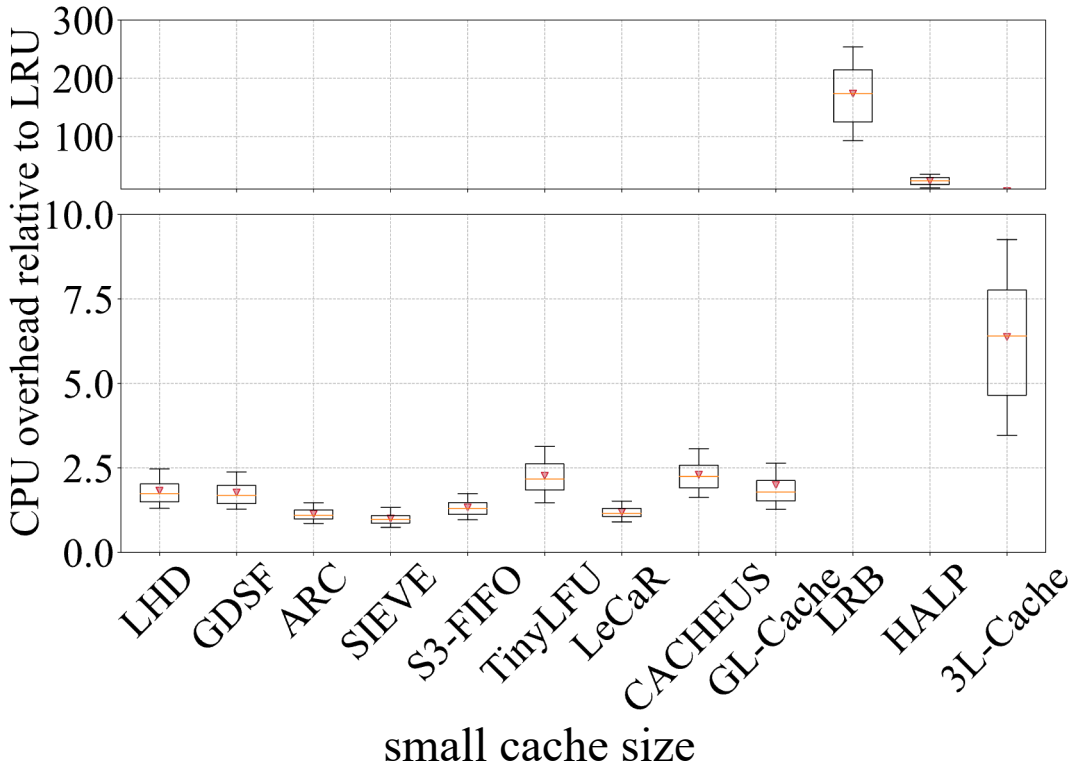
Tencent CBS, 4030 traces (33690 million requests)

CPU Overhead

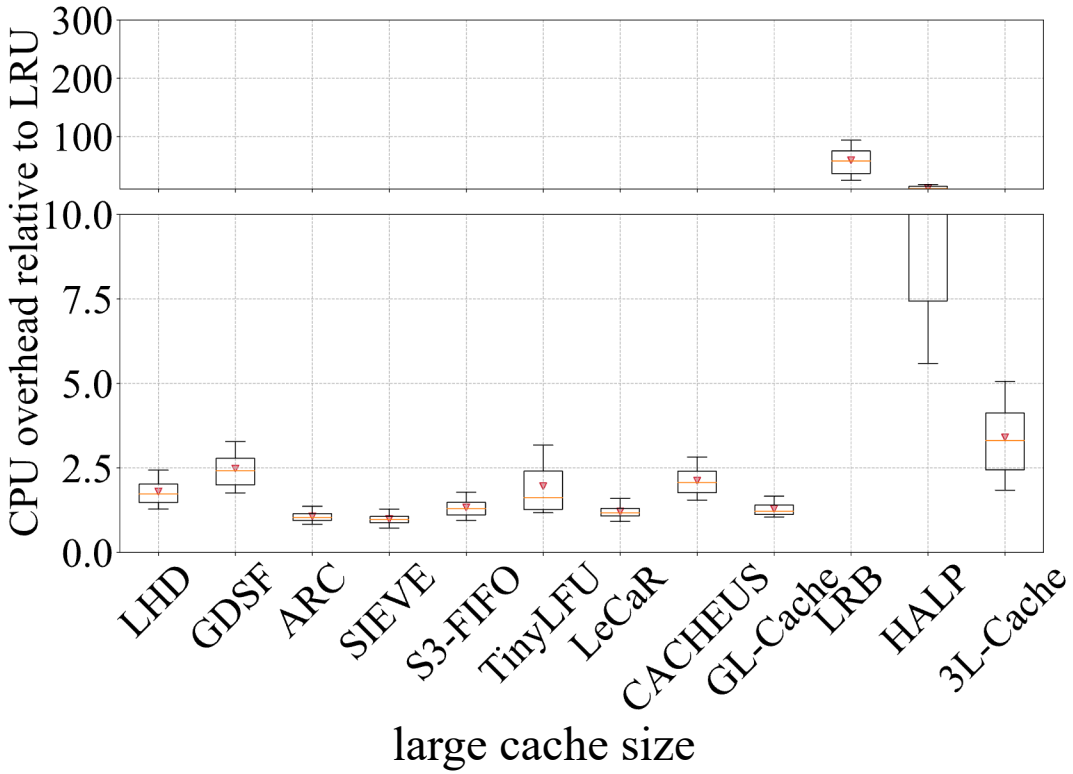
The mean CPU overhead of 3L-Cache is only **2.6×** higher than that of LHD and **6.4×** higher than that of LRU on average.

The mean CPU overhead of 3L-Cache is only is **3.4×** that of LRU.

Better



small cache size



large cache size

Summary

3L-Cache: Low Overhead and Precise Learning-based Eviction Policy for Caches

- **Low-overhead Online Training Design**
  - Sliding Window Adjustment
  - Efficient Sampling & Labeling
- **Low-overhead Cached Object Eviction Design**
  - Bidirectional Sampling Policy
  - Efficient Object Eviction
- **Achieves low object cache miss and low byte cache miss**
  - Outperforms state-of-the-art eviction policies in evaluations on 4,855 traces across diverse datasets.

Open-sourced on GitHub: https://github.com/optiq-lab/3L-Cache