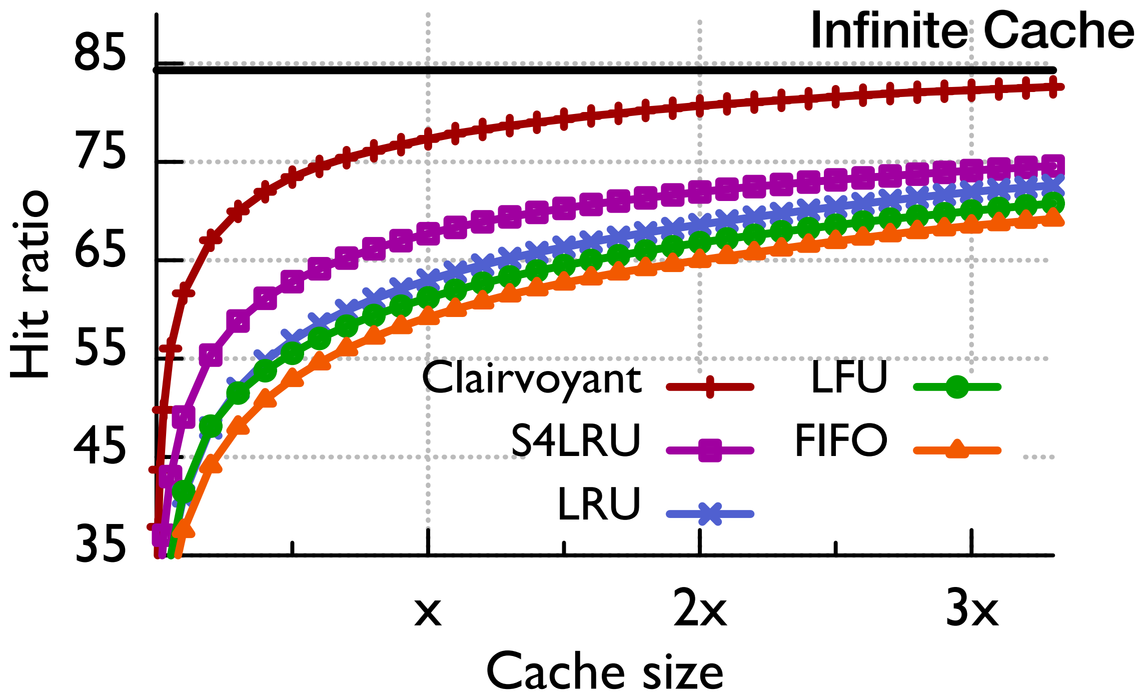
Learning Relaxed Belady for CDN Caching

COS 316: Principles of Computer System Design

Lecture 10

Amit Levy & Wyatt Lloyd

Edge Cache with Different Algos



- **Clairvoyant (Bélády)** shows we can do much better!

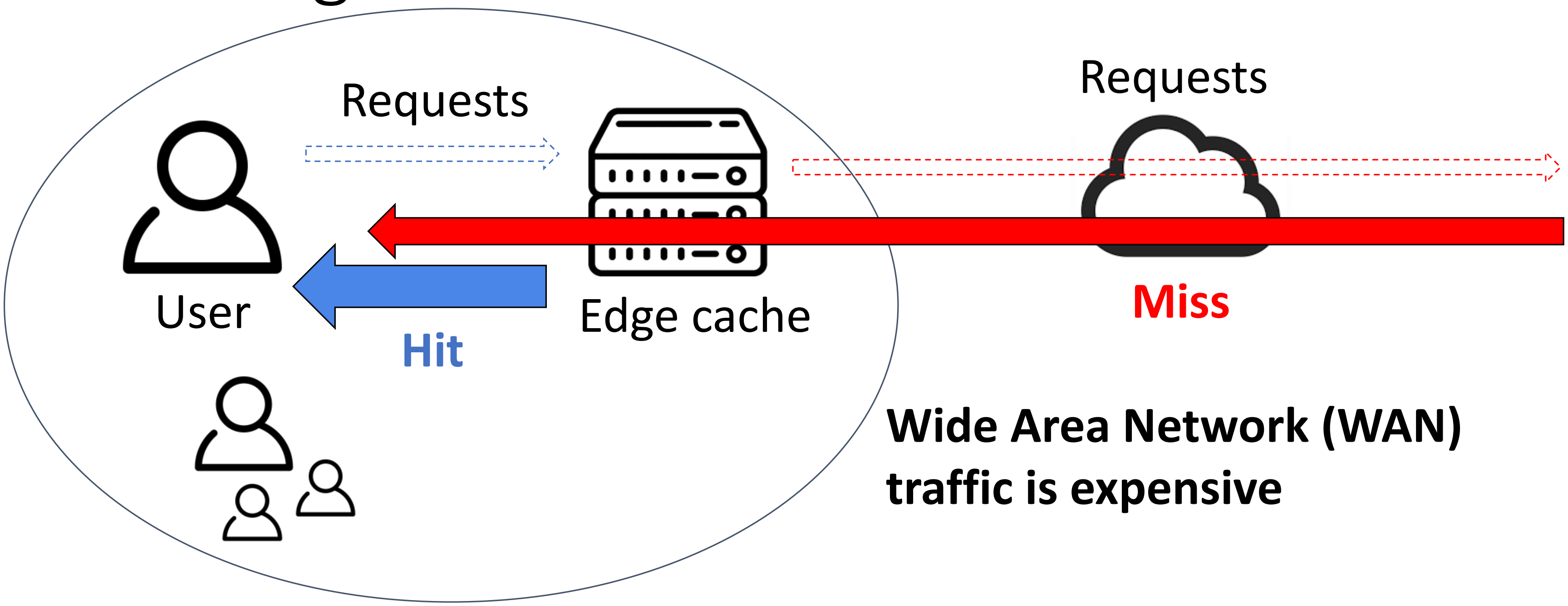
Cutting Edge Research From Princeton!

Learning Relaxed Belady for
Content Distribution Network Caching.

Zhenyu Song, Daniel S. Berger,
Kai Li, and Wyatt Lloyd.

In 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), February 2020.

CDN Caching Goal: Minimize WAN Traffic



Key metric **hit ratio**

Caching Remains Challenging

Heuristic-based algorithms (1965–): **LRU**, **LFU**, GDSF, ARC, ...

- Work well for some workloads, but work poorly for other

ML-based adaptation of heuristics (2017–): UCB, LeCAR, ...

- Also work well for some workloads, but poorly for others

The **Belady** algorithm (1966)

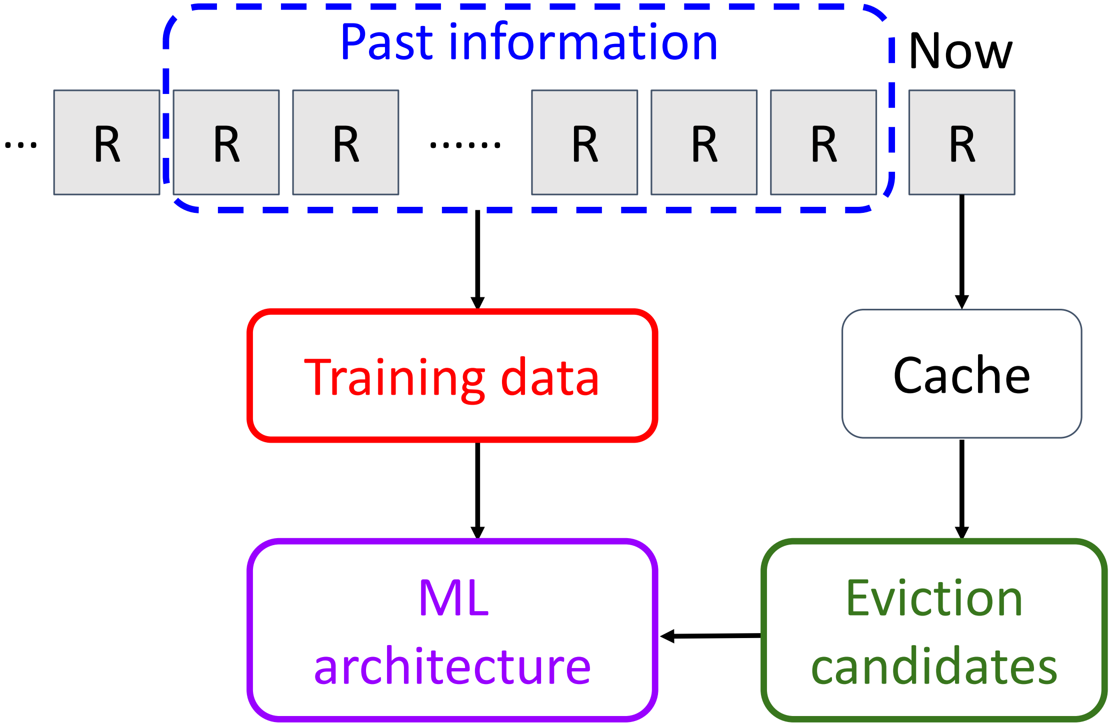
- Offline optimal: requires future knowledge
- Large gap in miss ratio between state-of-the-art and Belady:
- 20–40% on production traces

Introducing Learning Relaxed Belady (LRB)

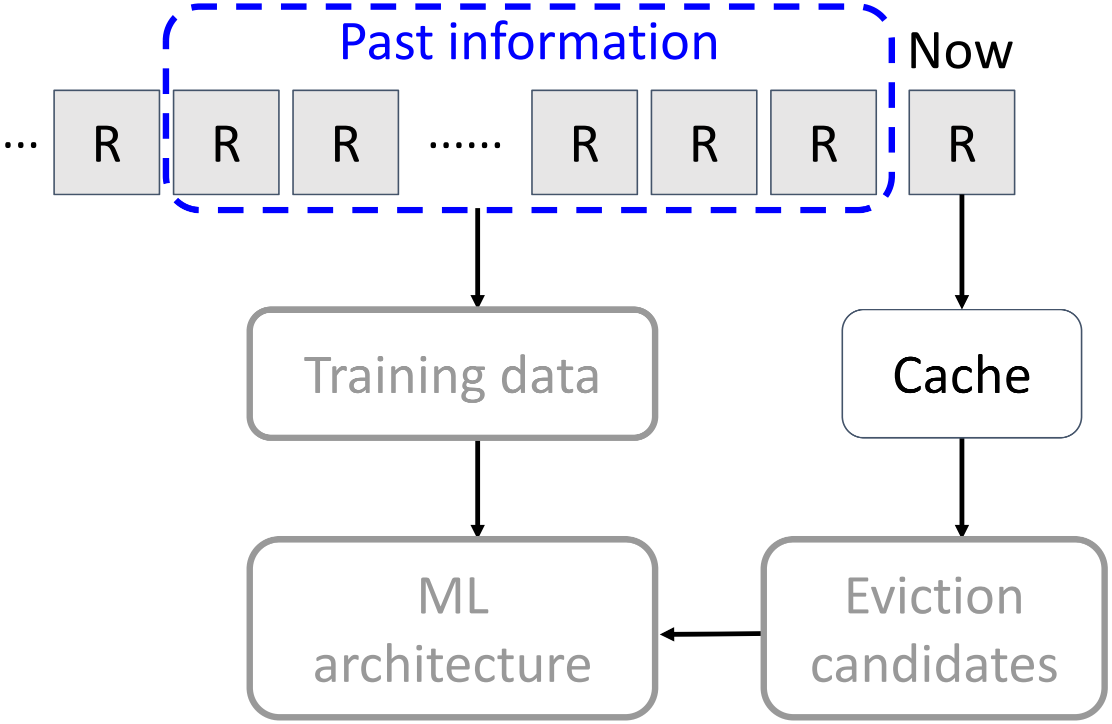
New approach: mimic Belady using machine learning

- Machine-Learning-for-Systems (ML-for-Systems)
  - Enabling technologies
  - When does it make sense?

General Overview of our Approach



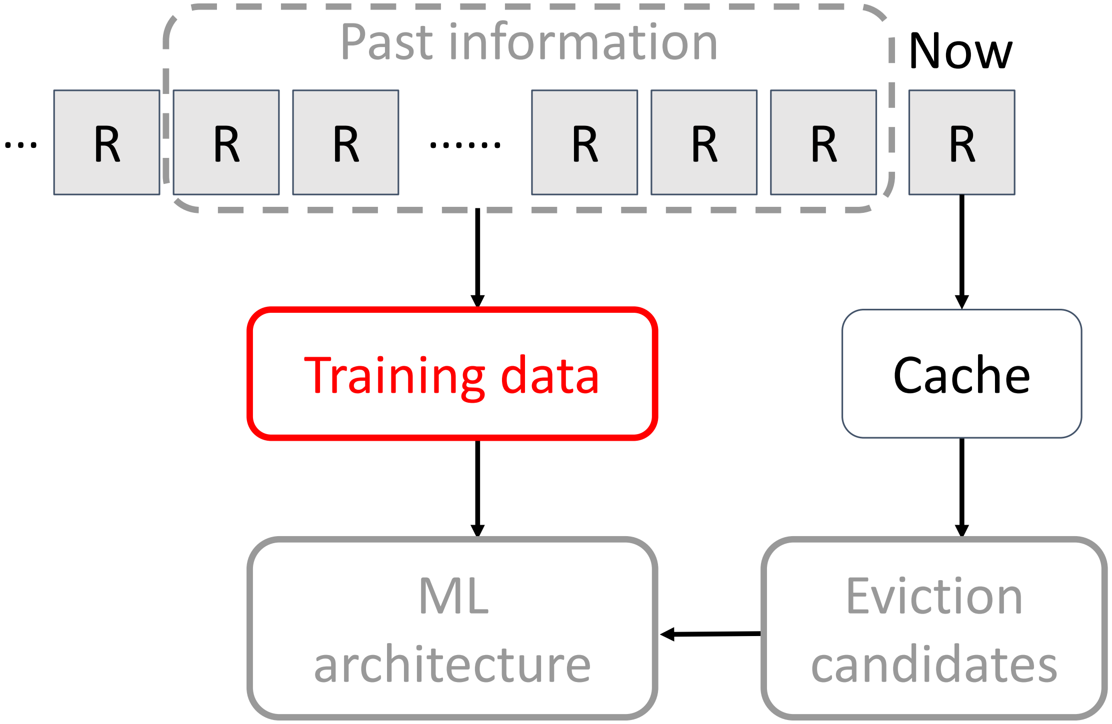
Challenge 1: Past Information



What past information to use?

More data improves training but increases memory overhead

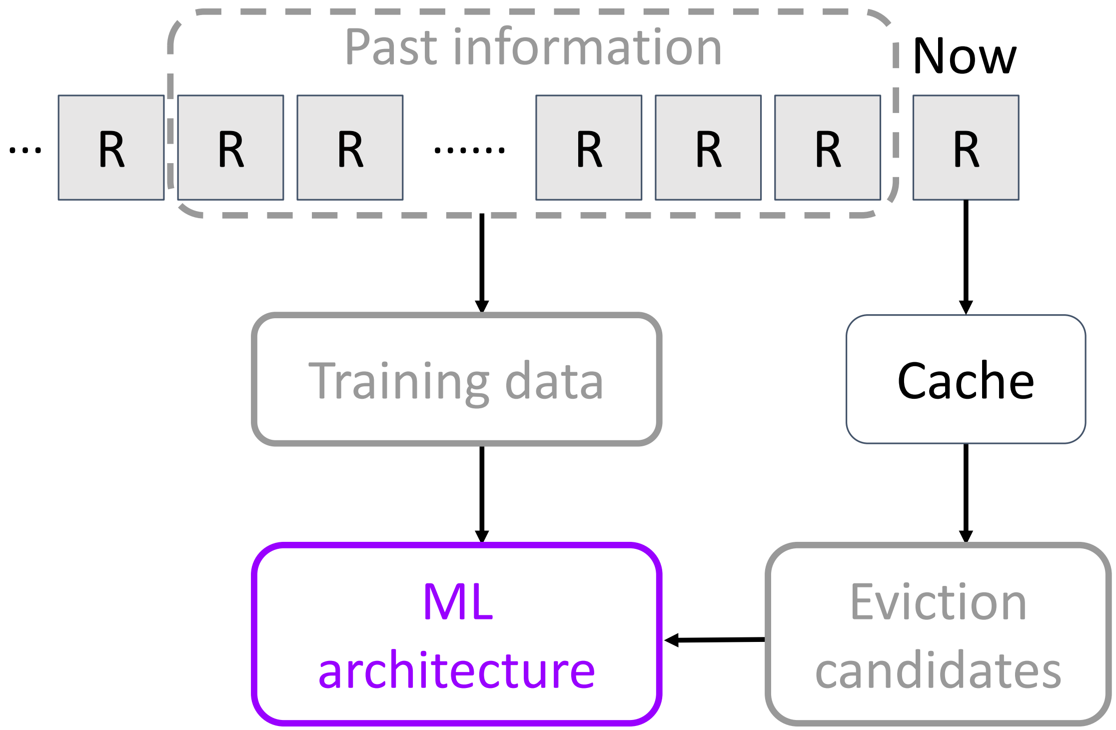
Challenge 2: Generate Online Training Data



What past information to use?

Generate online training data?

Challenge 3: ML Architecture



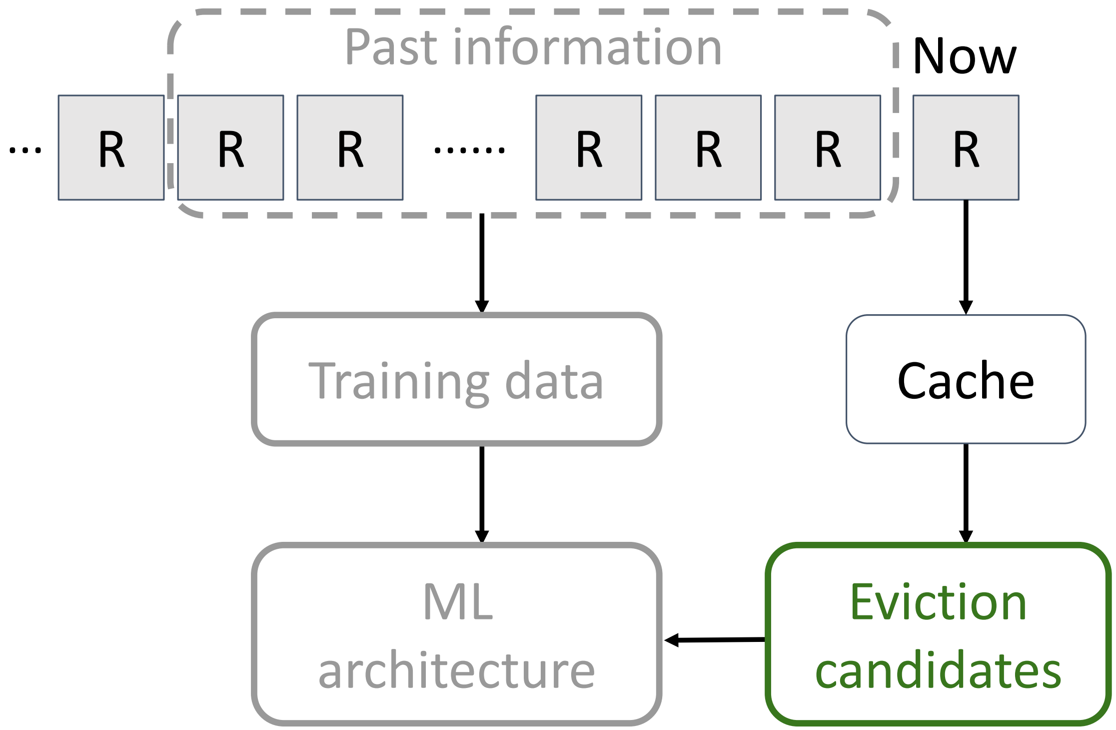
What past information to use?

Generate online training data?

What ML architecture to select?

Large design space:
features, model, prediction target, loss function

Challenge 4: Eviction Candidates



- What past information to use?
- Generate online training data?
- What ML architecture to select?
- **How to select evict candidates?**

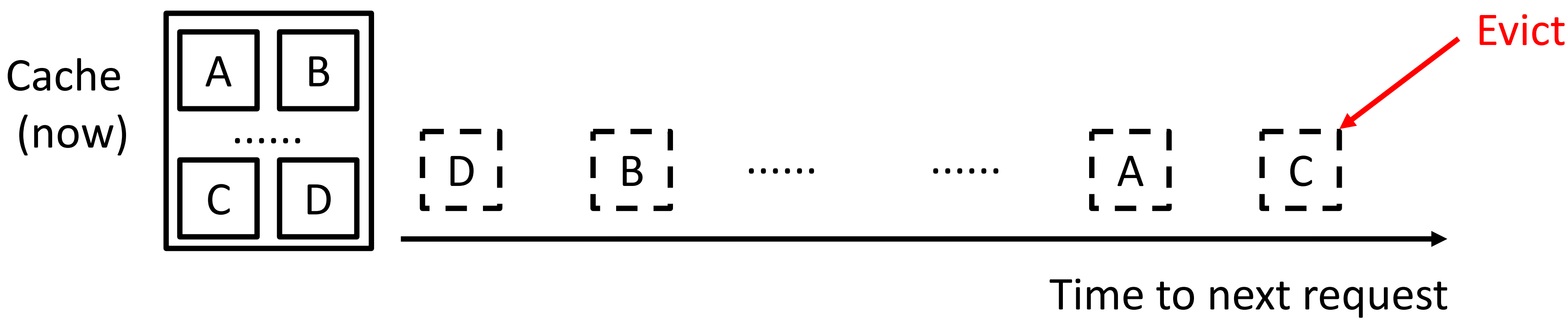
Solution: Relaxed Belady Algorithm

- What past information to use?
- Generate online training data?
- What ML architecture to select?
- How to select evict candidates?

Relaxed Belady algorithm

Challenge: Hard to Mimic Belady Algorithm

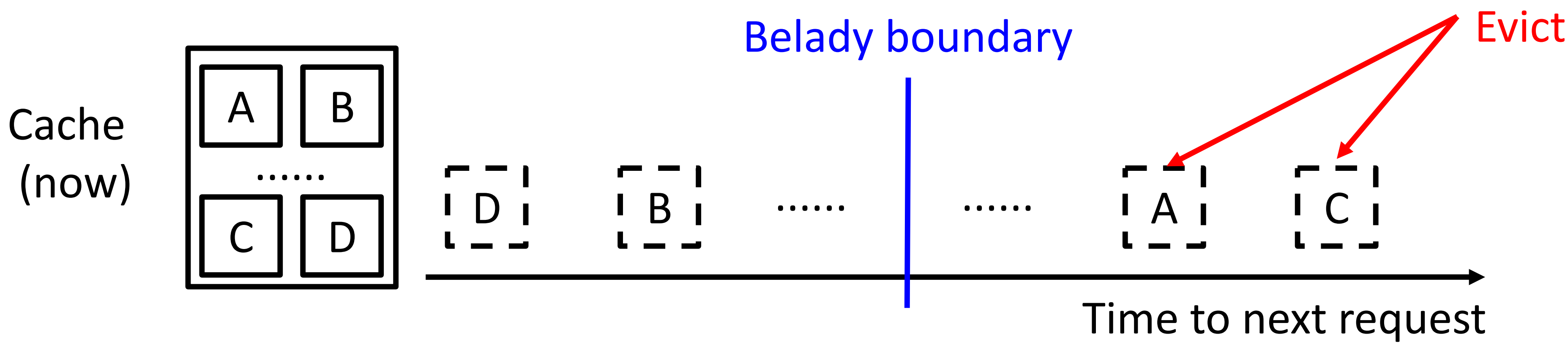
Belady: evict object with next access farthest in the future



Mimicking exact Belady is impractical

- Need predictions for all objects → prohibitive computational cost
- Need exact prediction of next access → further prediction are harder

Introducing the Relaxed Belady Algorithm

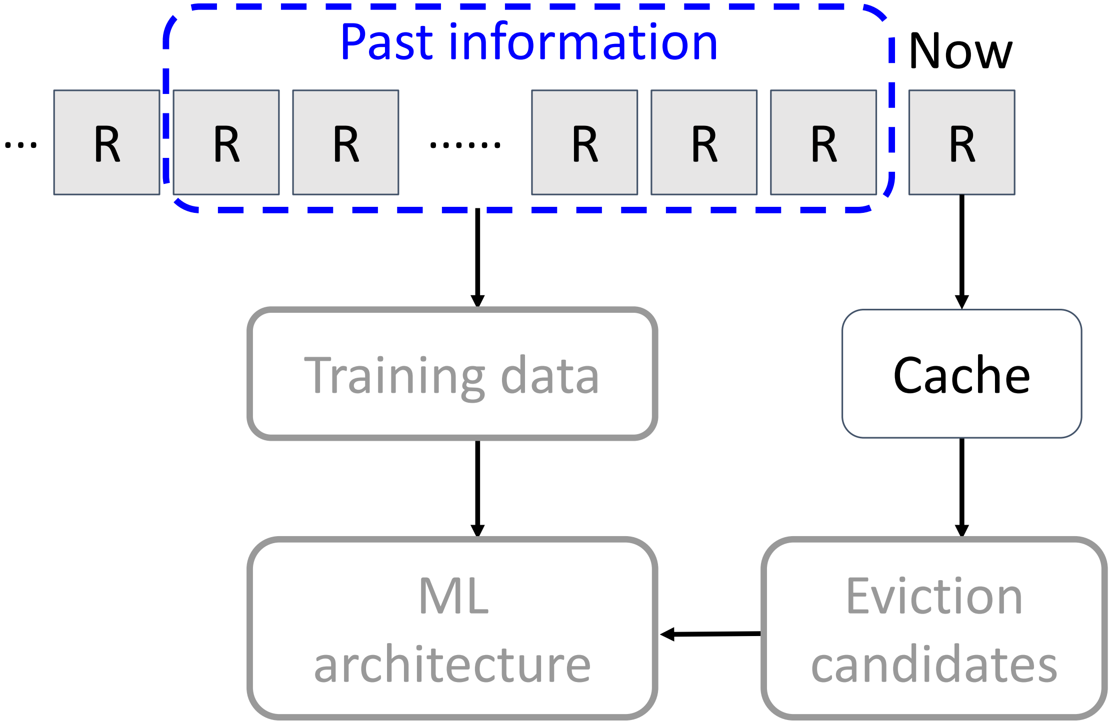


Observation: many objects are good candidates for eviction

Relaxed Belady evicts a random object beyond boundary

- Do not need predictions for all objects → reasonable computation
- No need to differentiate beyond boundary → simplifies the prediction

Challenge 1: Past Information

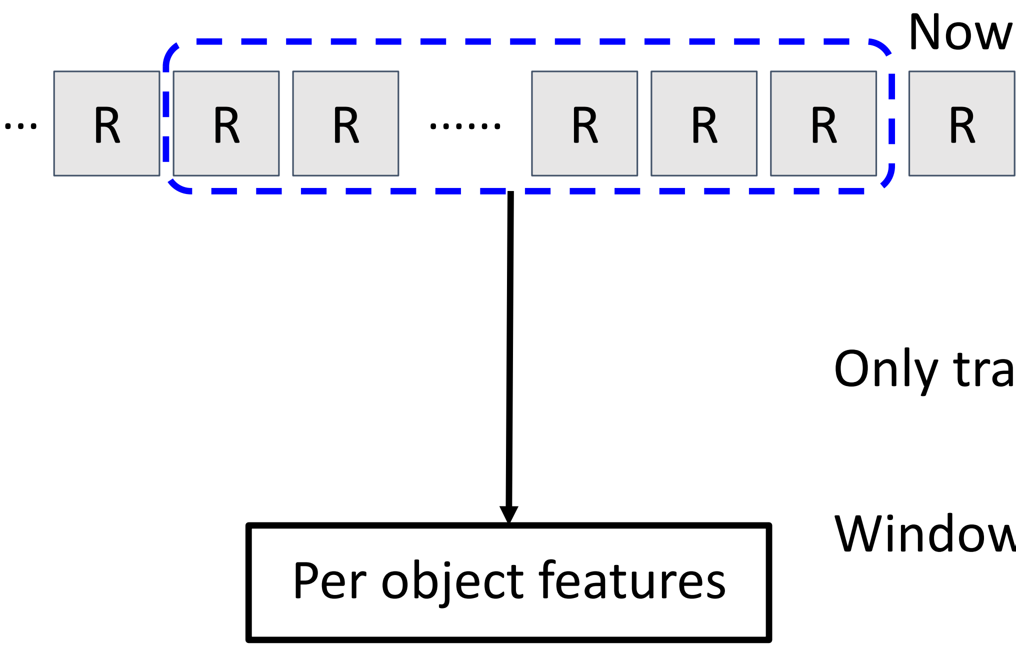


What past information to use?

More data improves training but increases memory overhead

Track Objects within a Sliding Memory Window

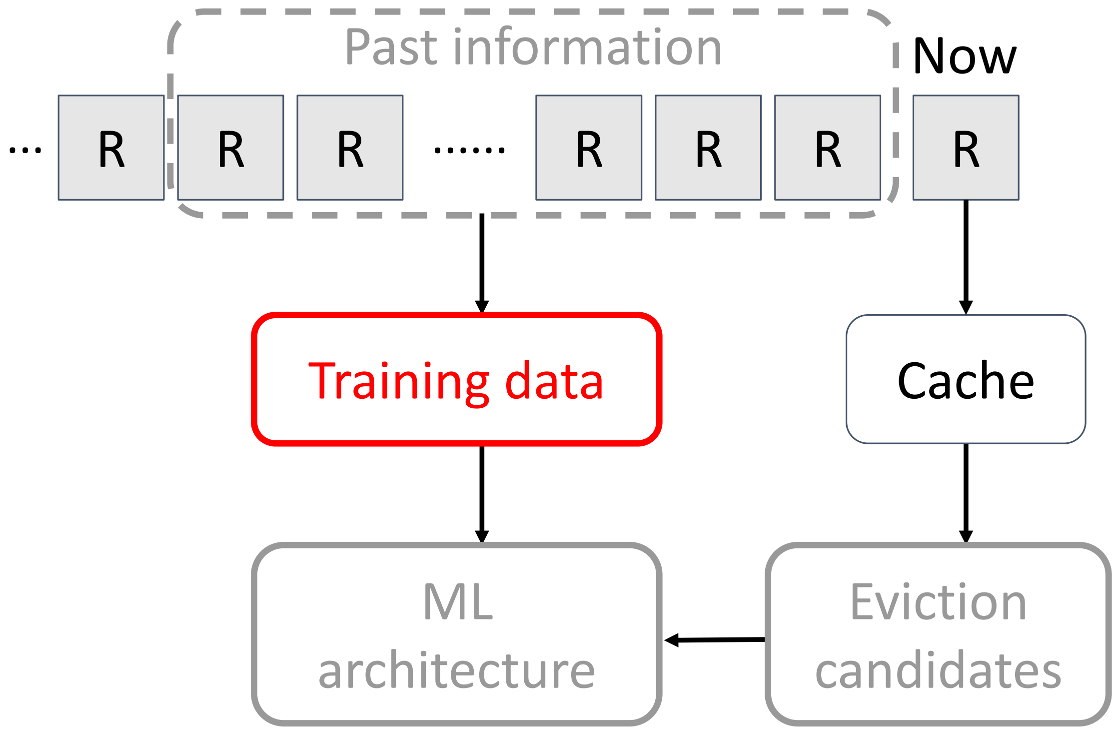
Sliding memory window mimics Belady boundary



Only track objects within memory window

Window size is LRB's main hyperparameter

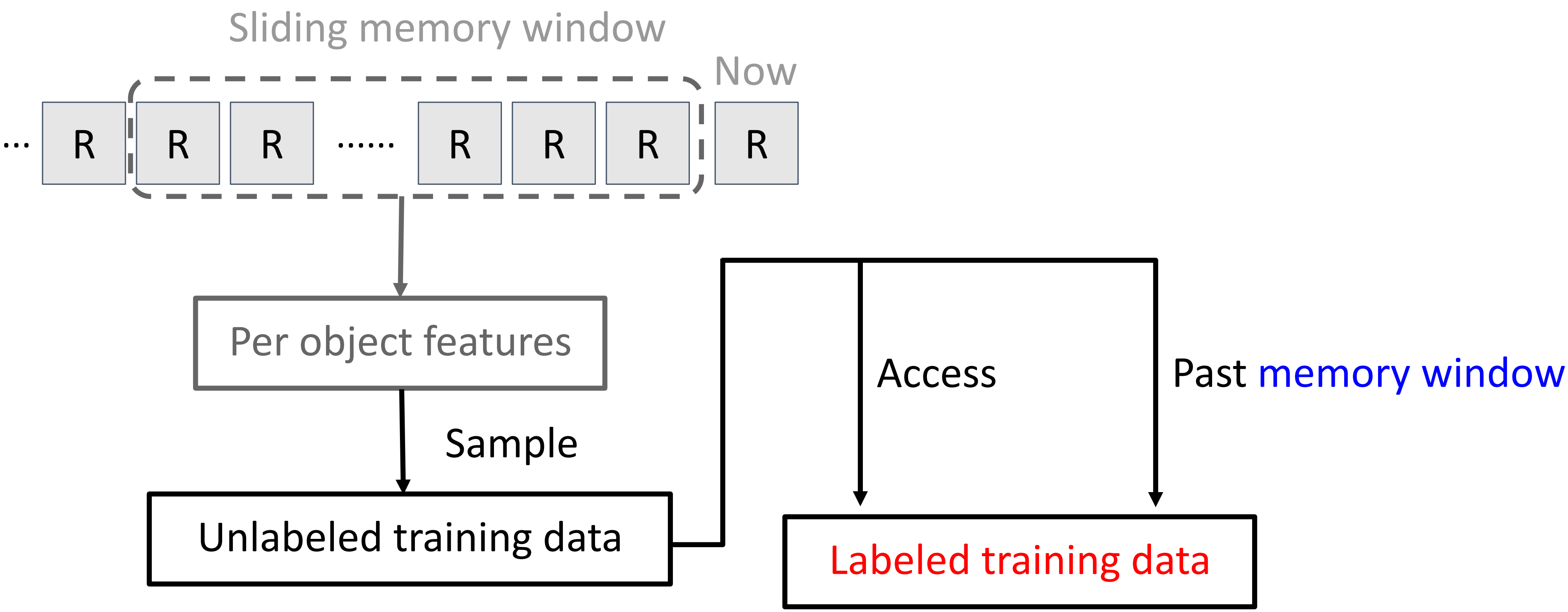
Challenge 2: Training Data



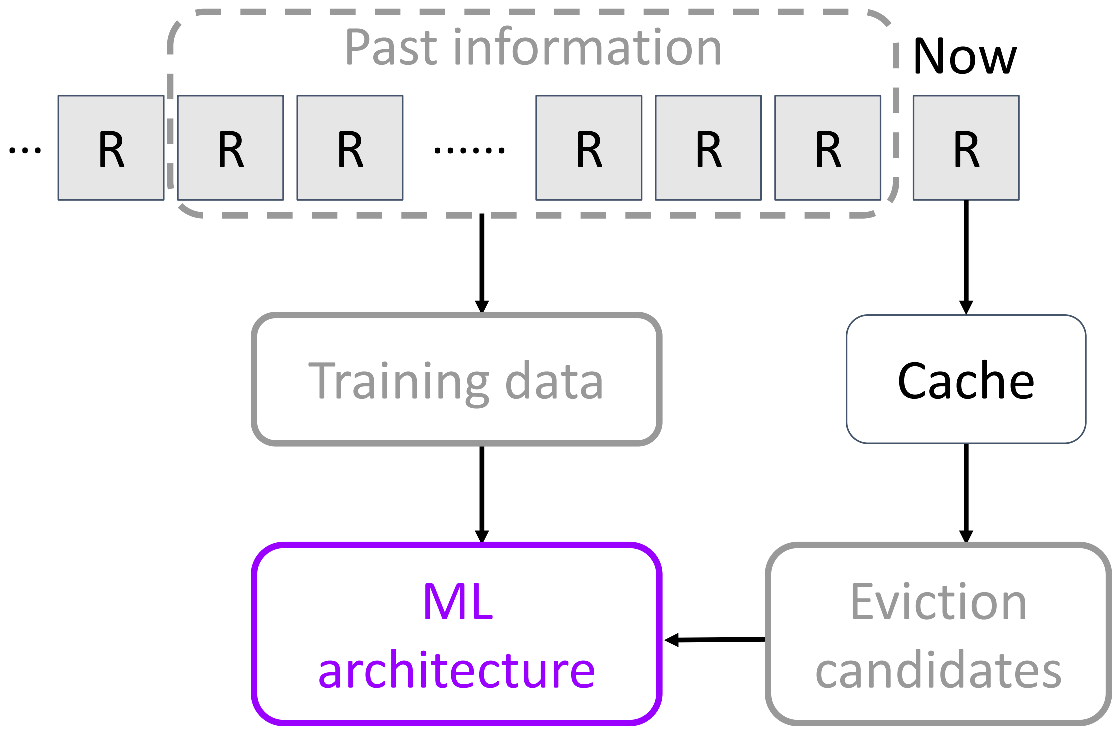
What past information to use?

Generate online training data?

Sample Training Data & Label on Access or Boundary



Challenge 3: ML Architecture



What past information to use?

Generate online training data?

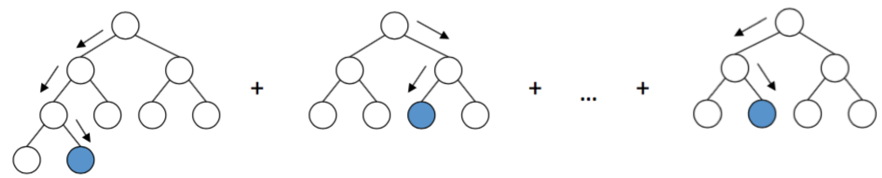
What ML architecture to select?

Large potential design space

Solution 3: Feature & Model Selection

Use good decision ratio to evaluate new designs

Features
Object size
Object type
Inter-request distances (recency)
Exponential decay counters (long-term frequencies)

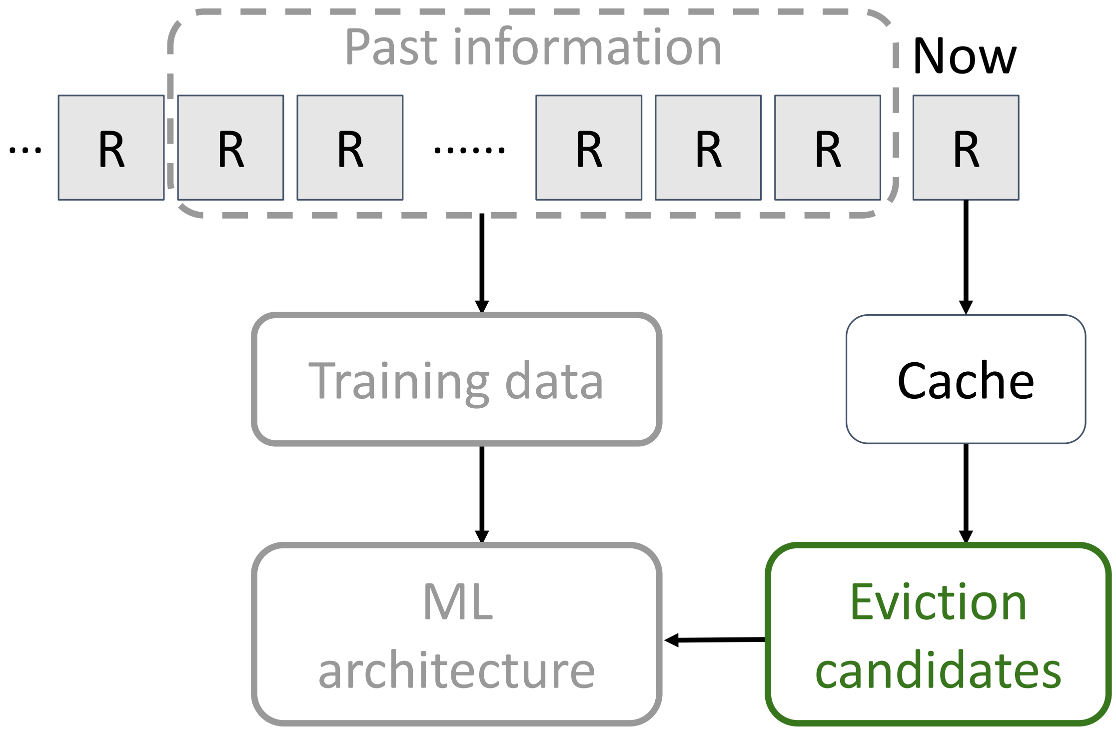


Gradient boosting decision trees

Lightweight & **high good decision ratio**

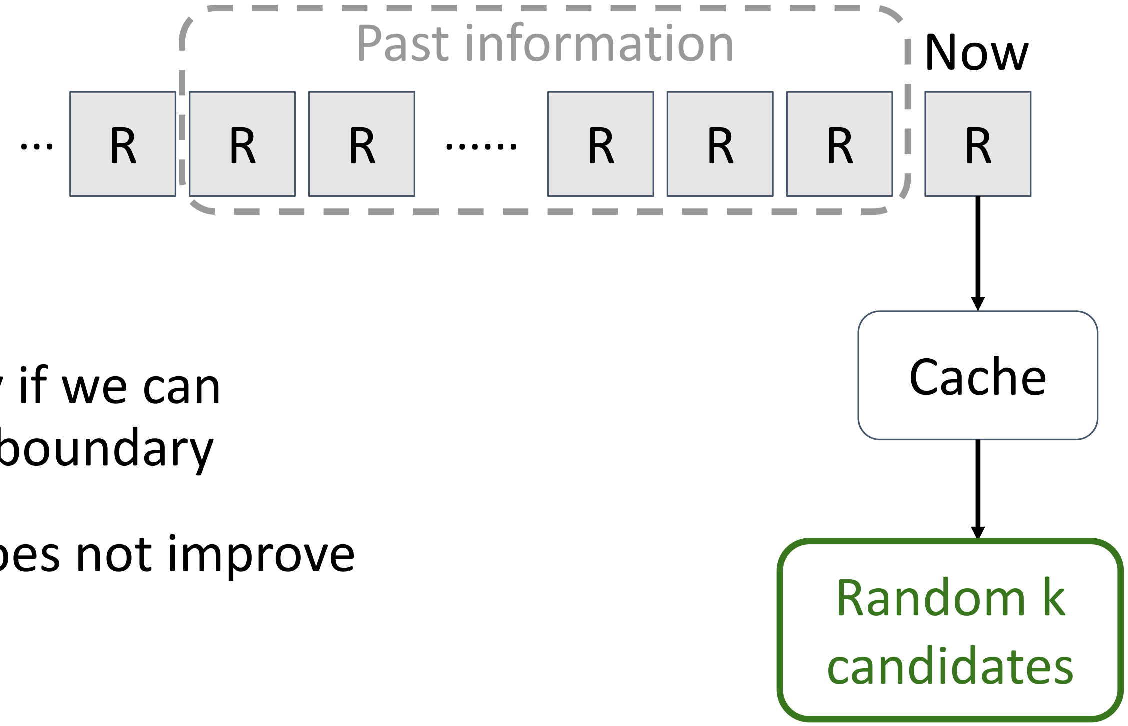
Training ~300 ms, prediction ~30 us

Challenge 4: Eviction Candidates



- What past information to use?
- Generate online training data?
- What ML architecture to select?
- **How to select evict candidates?**

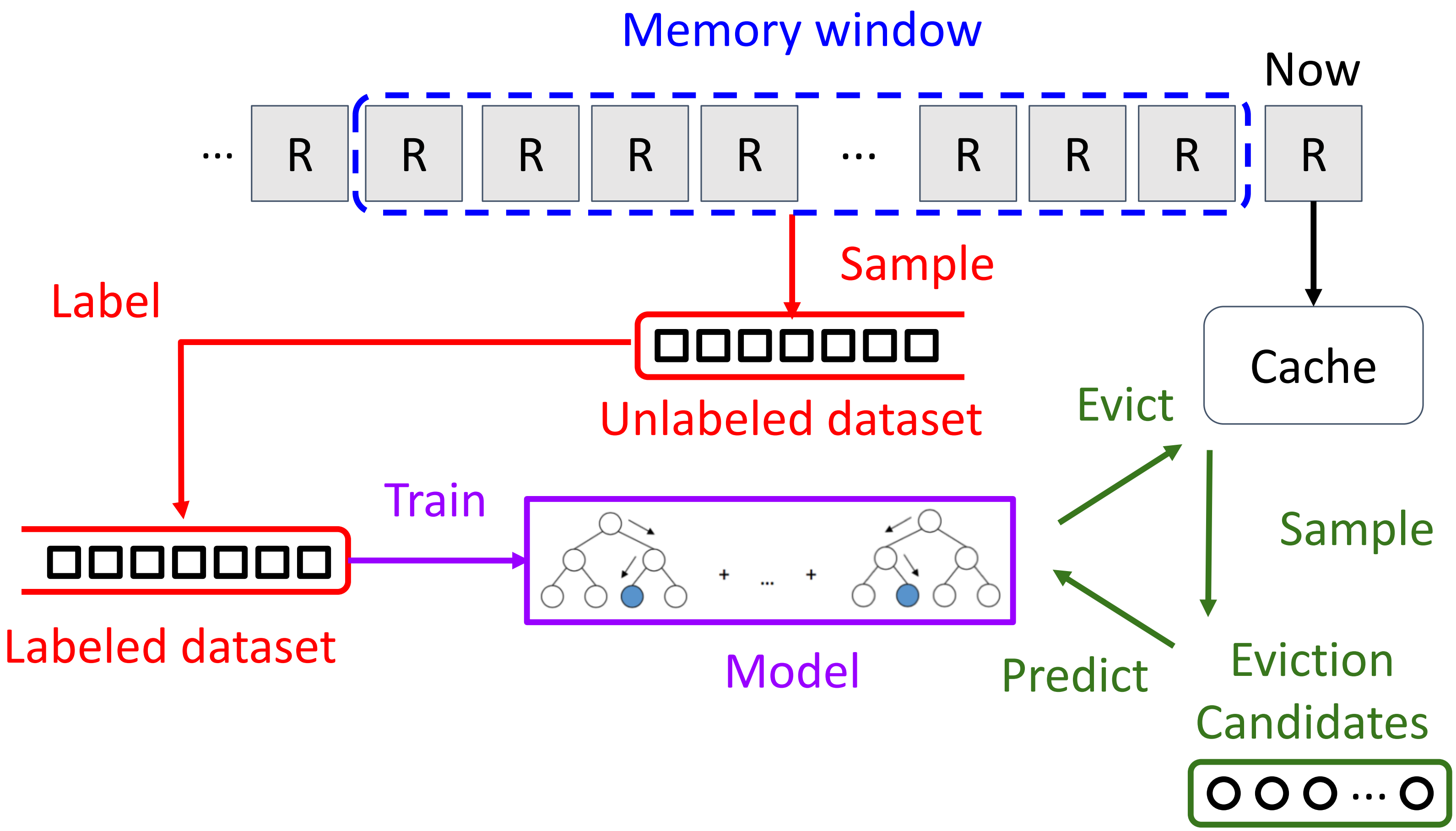
Solution 4: Random Sampling for Eviction



Can mimic relaxed Belady if we can find 1 object beyond the boundary

k=64 candidates; more does not improve good decision ratio

Learning Relaxed Belady



Implementation

- Simulator implementation
  - LRB + 14 other algorithms

- Prototype implementation
  - C++ on top of production system (Apache Traffic Server)
  - Many optimizations

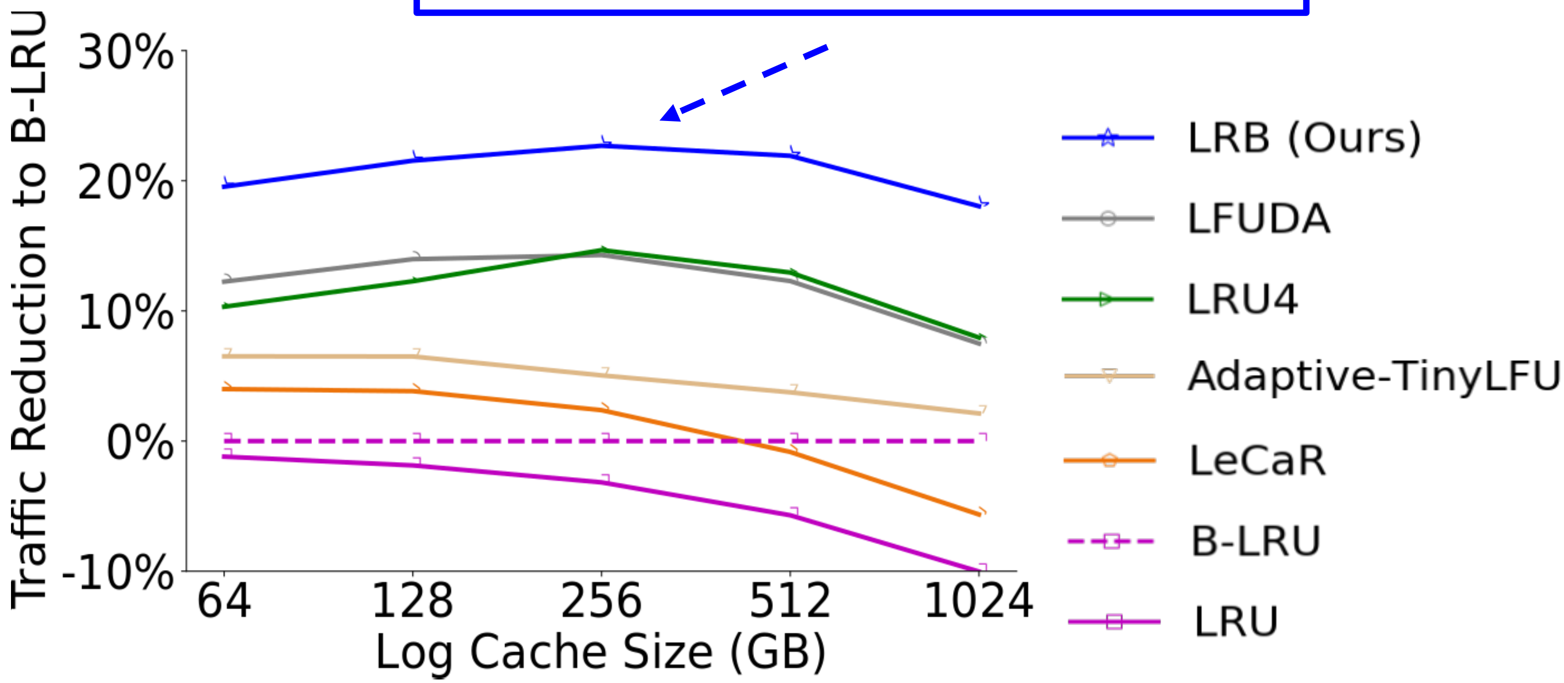
Evaluation Setup

- Q1: Learning Relaxed Belady (LRB) traffic reduction vs state-of-the-art
- Q2: overhead of LRB vs CDN production system
- Traces: 6 production traces from 3 CDNs
- Hyperparameter (memory window/model/...) tuned on 20% of trace

LRB Reduces WAN Traffic

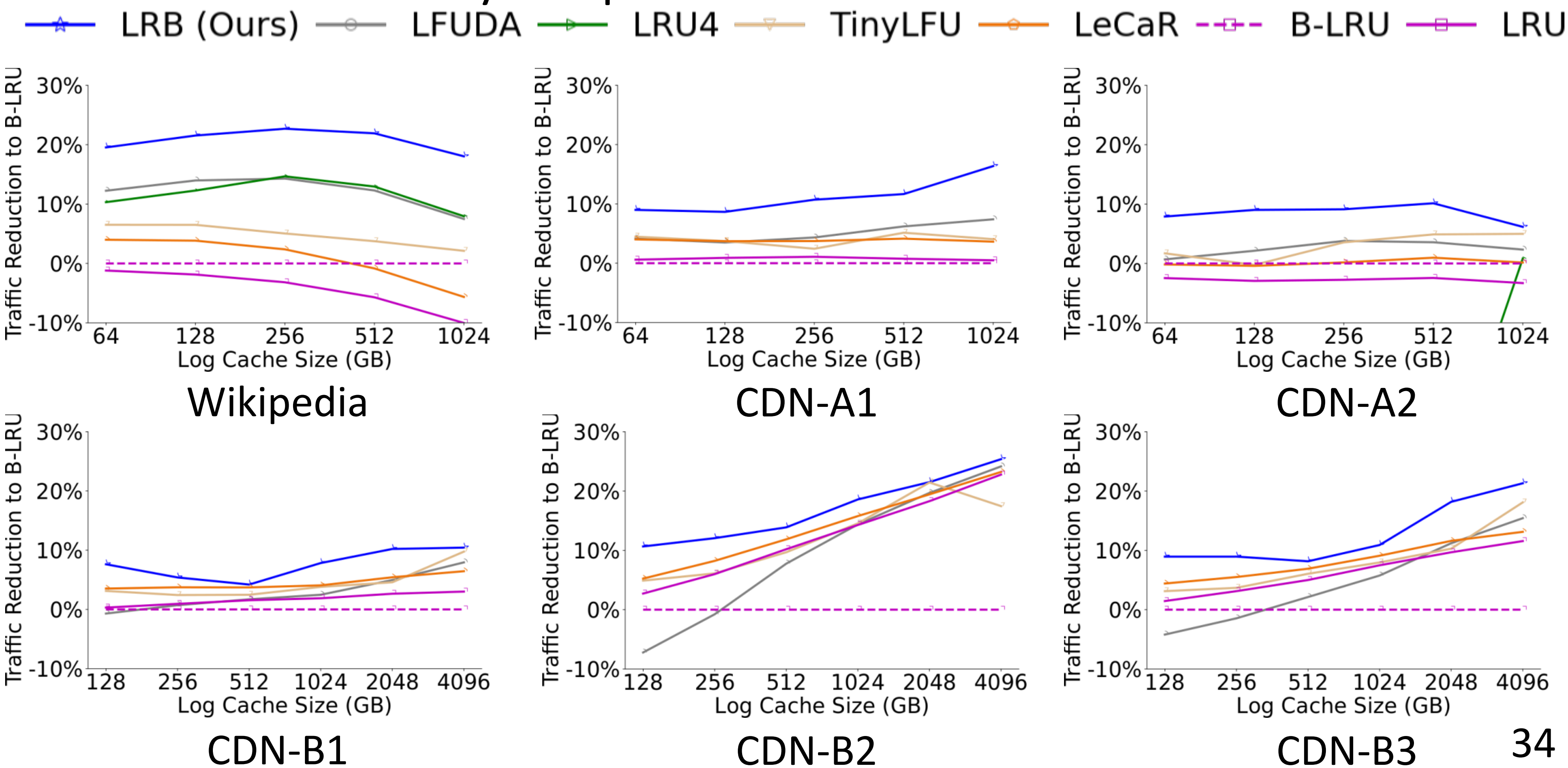
Industry standard

20% traffic reduction over B-LRU
10% reduction over the best SOA

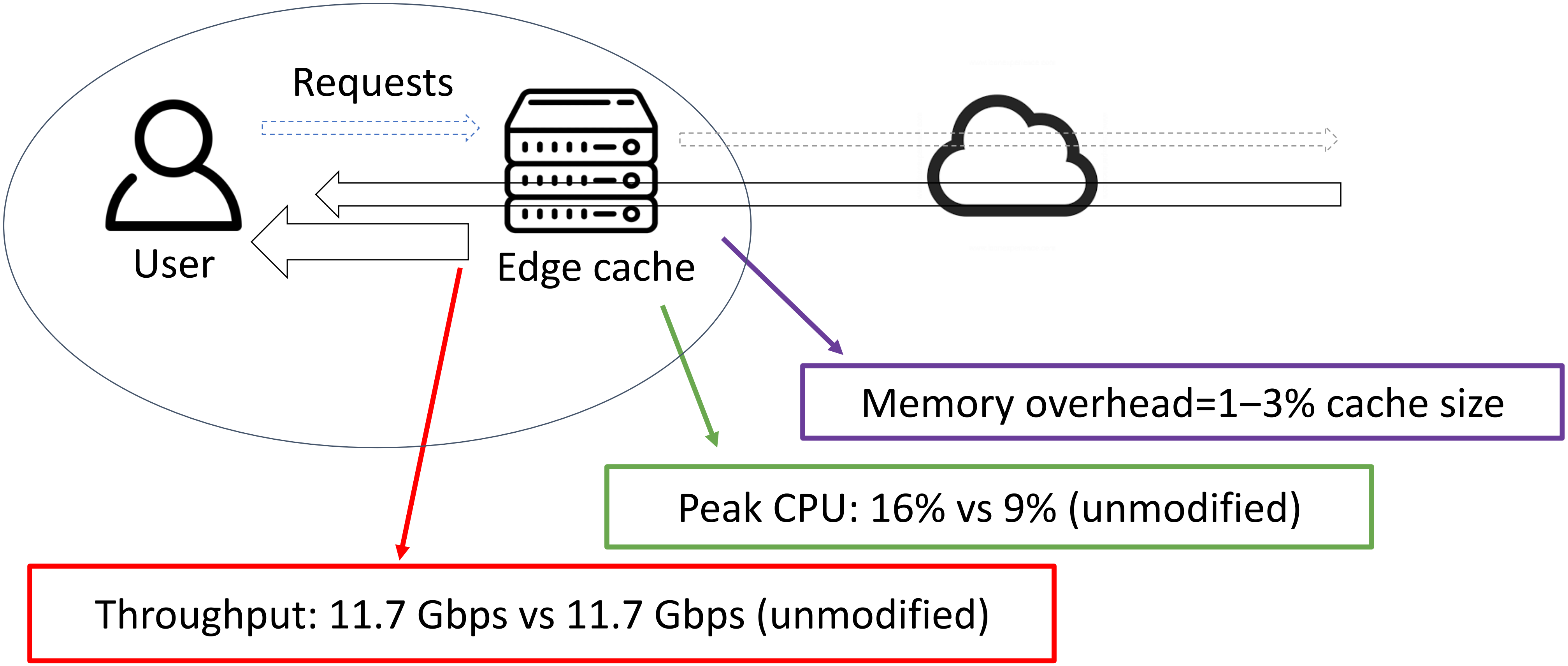


Wikipedia trace

LRB Consistently Improves on the State of the Art



LRB Overhead Is Modest



Conclusion

- LRB reduces WAN traffic with modest overhead

- ML-for-systems generally promising to replace heuristics

- Key insight: **relaxed Belady**

  → Simplifies machine learning & reduces system overhead

Microsoft Research