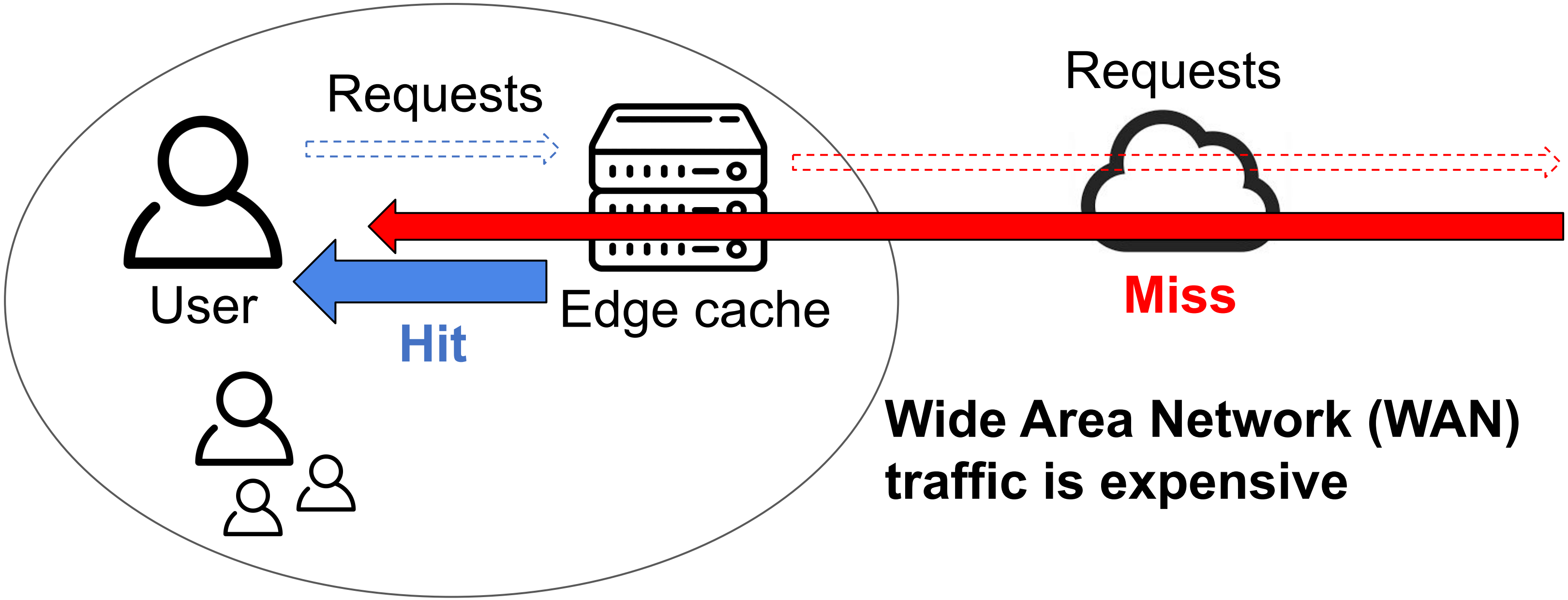
Learning Relaxed Belady for Content Delivery Network Caching

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

NSDI 2020

Microsoft Research

CDN Caching Goal: Minimize WAN Traffic



Wide Area Network (WAN) traffic is expensive

Key metric **byte miss ratio**: $\frac{\textbf{miss bytes}}{\textbf{total bytes}}$

Caching Remains Challenging

Heuristic-based algorithms (1965–): LRU, LRUK, 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

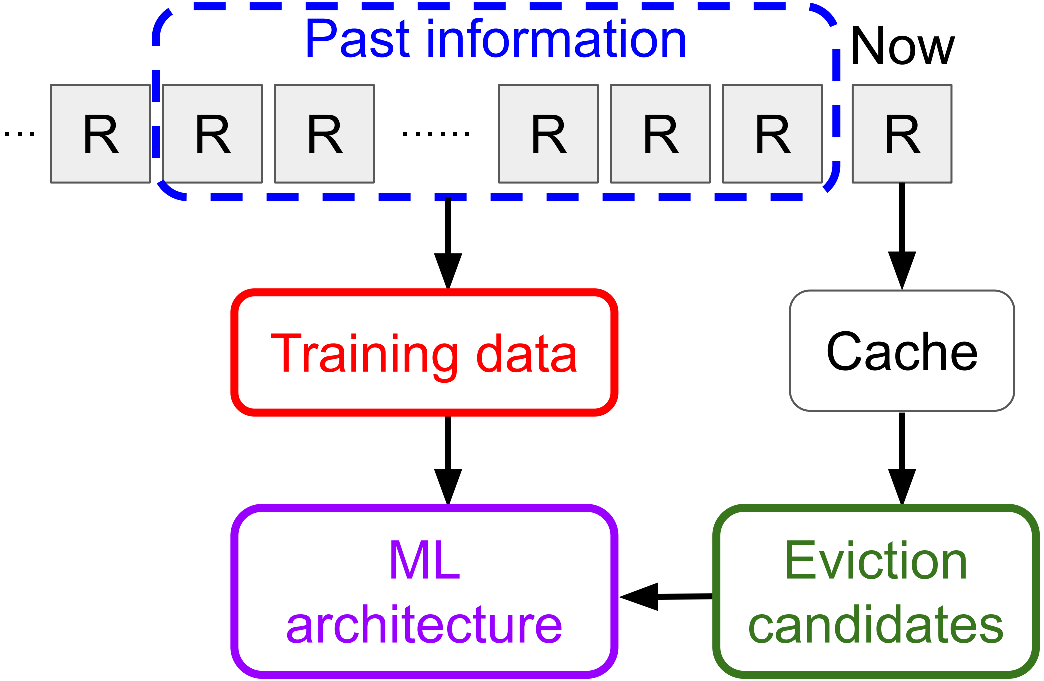
Belady’s MIN algorithm (1966)

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

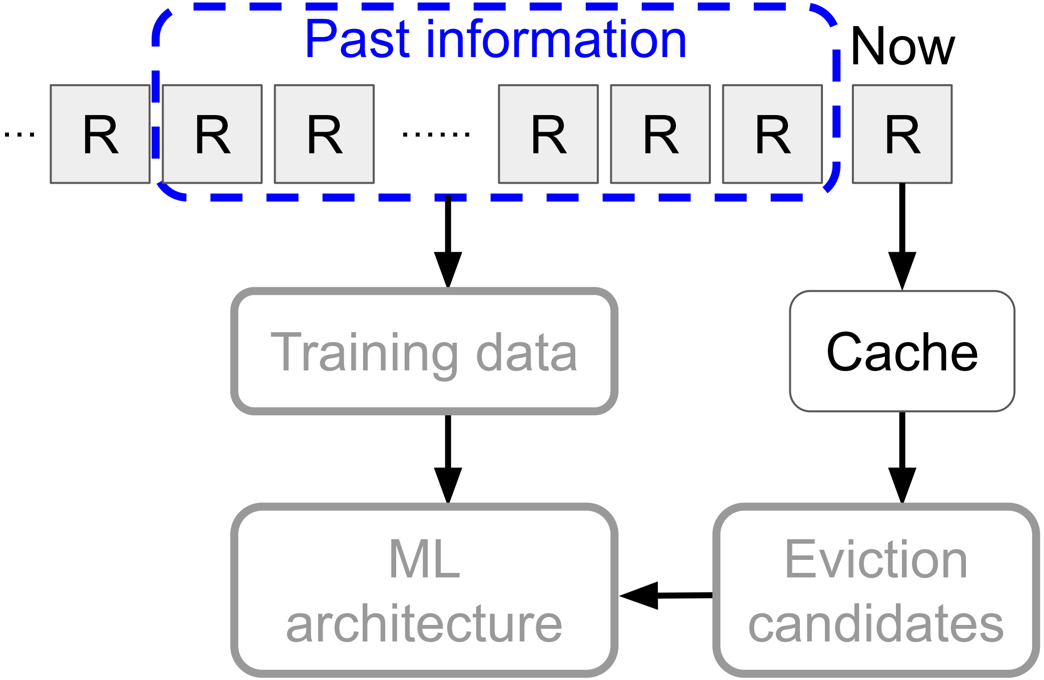
Introducing Learning Relaxed Belady (LRB)

New approach: mimic Belady using machine learning

General Overview of our Approach



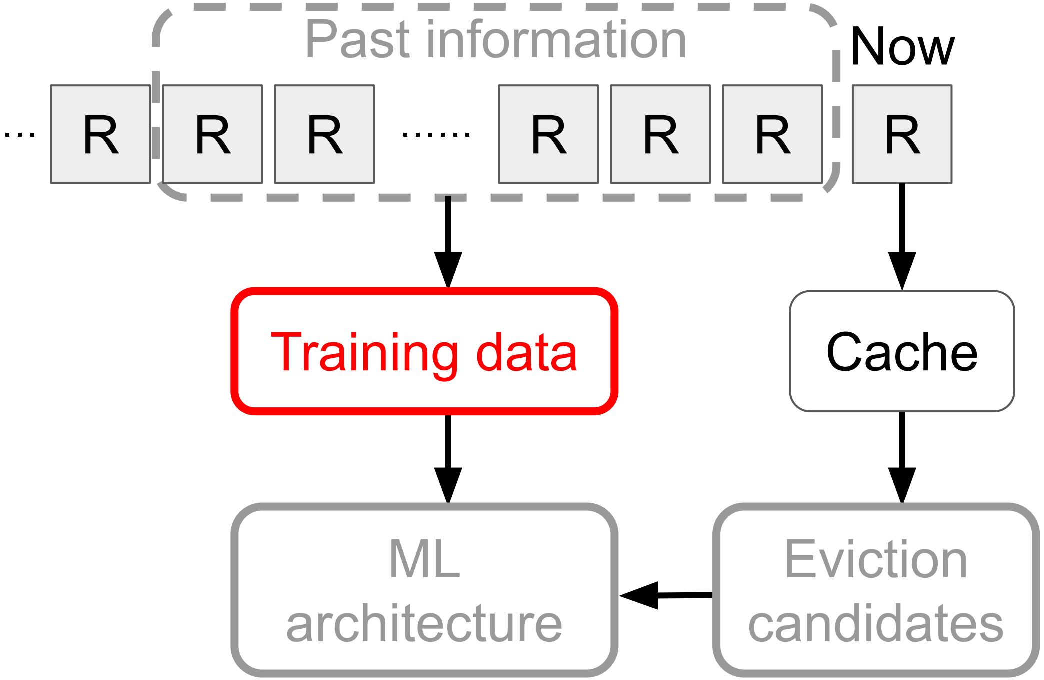
Challenge 1: Past Information



What past information to use?

More data improves training
but increases mem overhead

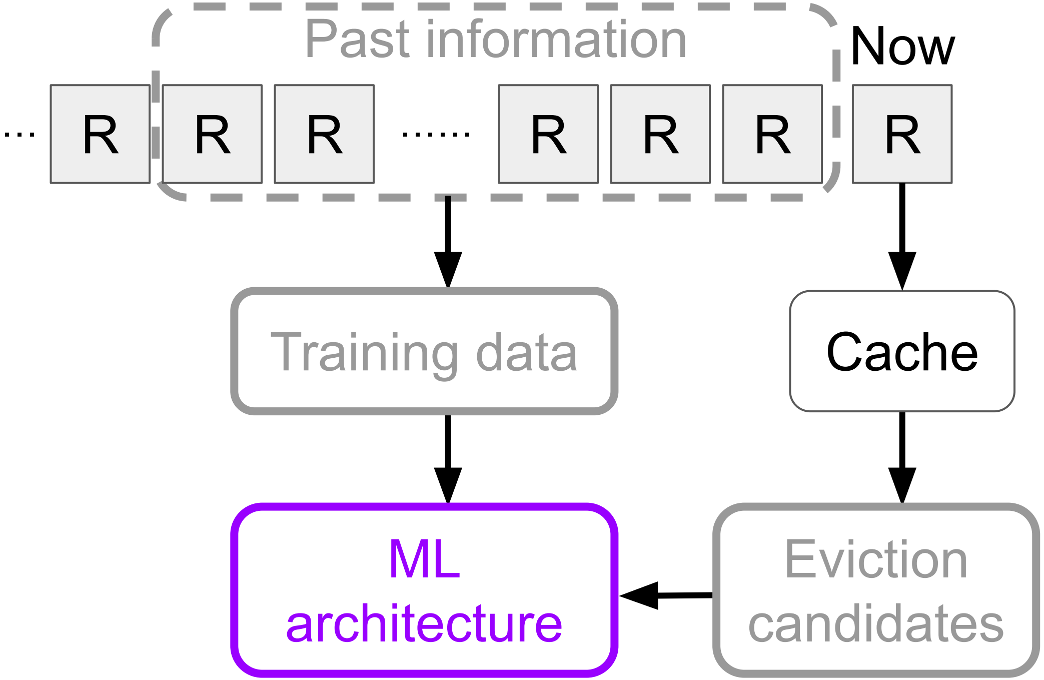
Challenge 2: Generate Online Training Data



What past information to use?

Generate online training data?

Challenge 3: ML Architecture



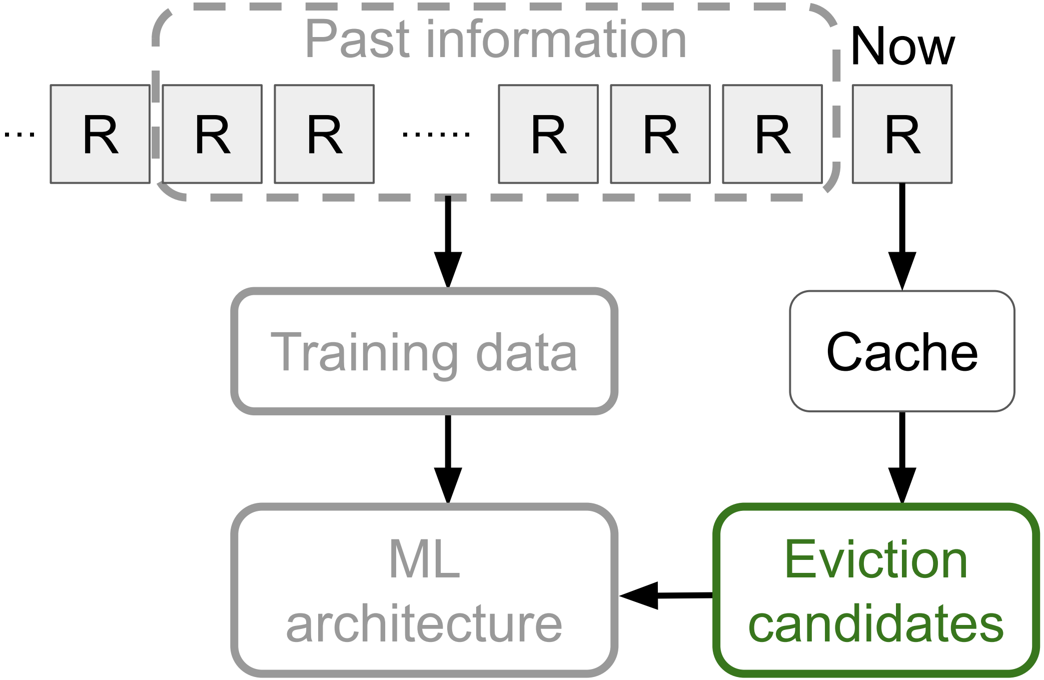
What past information to use?

Generate online training data?

What ML architecture to select?

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

Challenge 4: Eviction Candidates



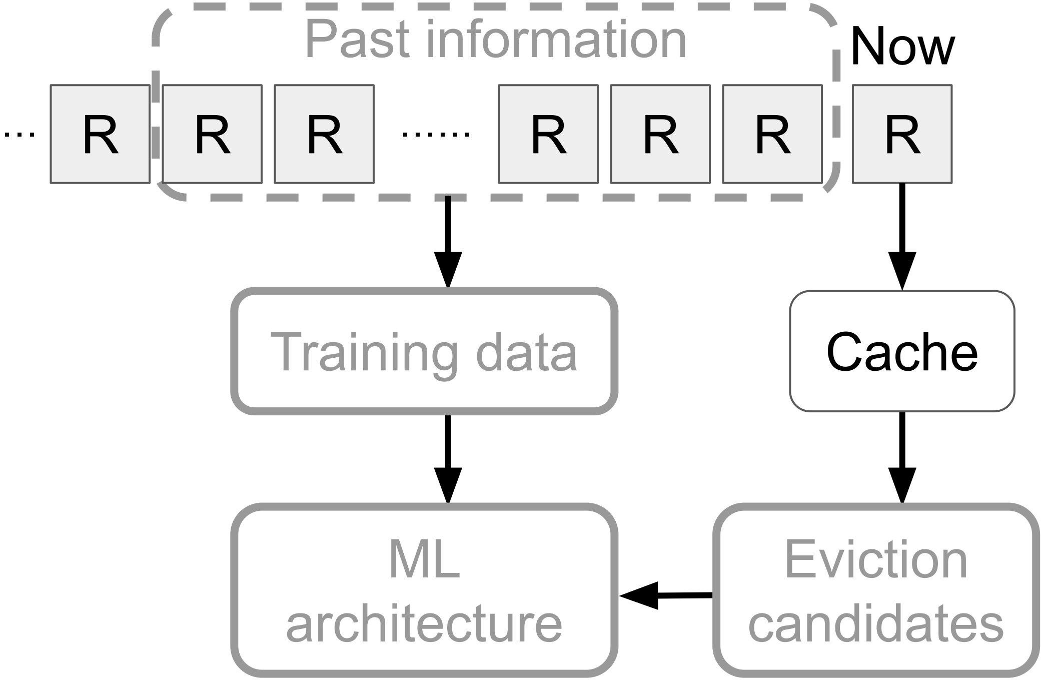
What past information to use?

Generate online training data?

What ML architecture to select?

How to select evict candidates?

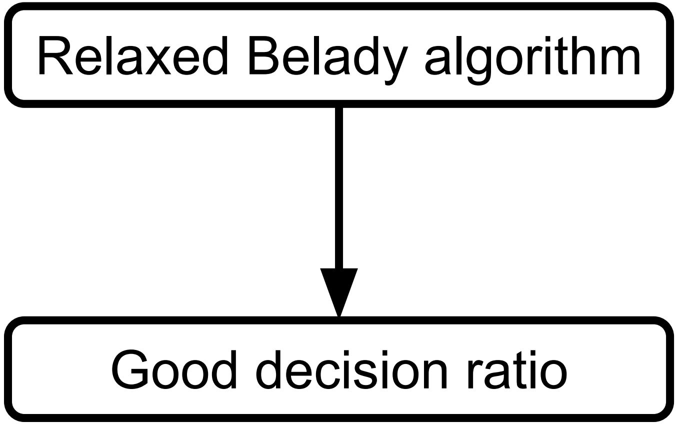
Challenge 5: Quickly Evaluate Design Decisions



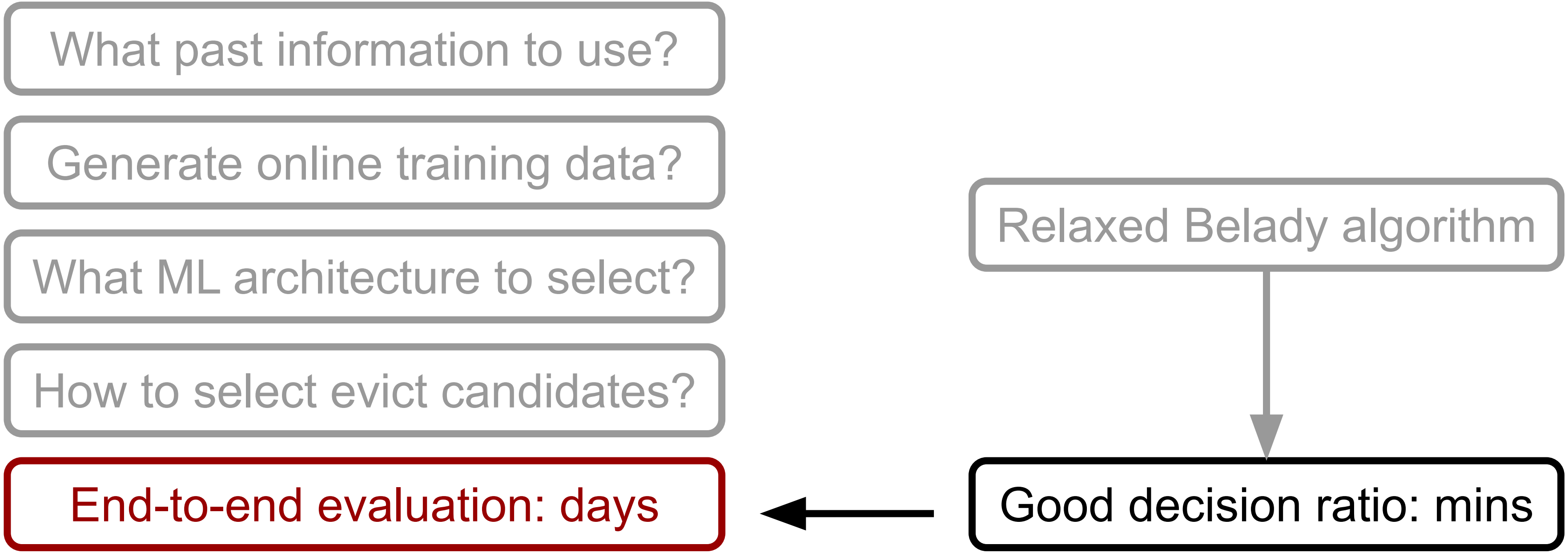
- What past information to use?
- Generate online training data?
- What ML architecture to select?
- How to select evict candidates?
- End-to-end evaluation: days

Solutions: Relaxed Belady Algorithm & Good Decision Ratio

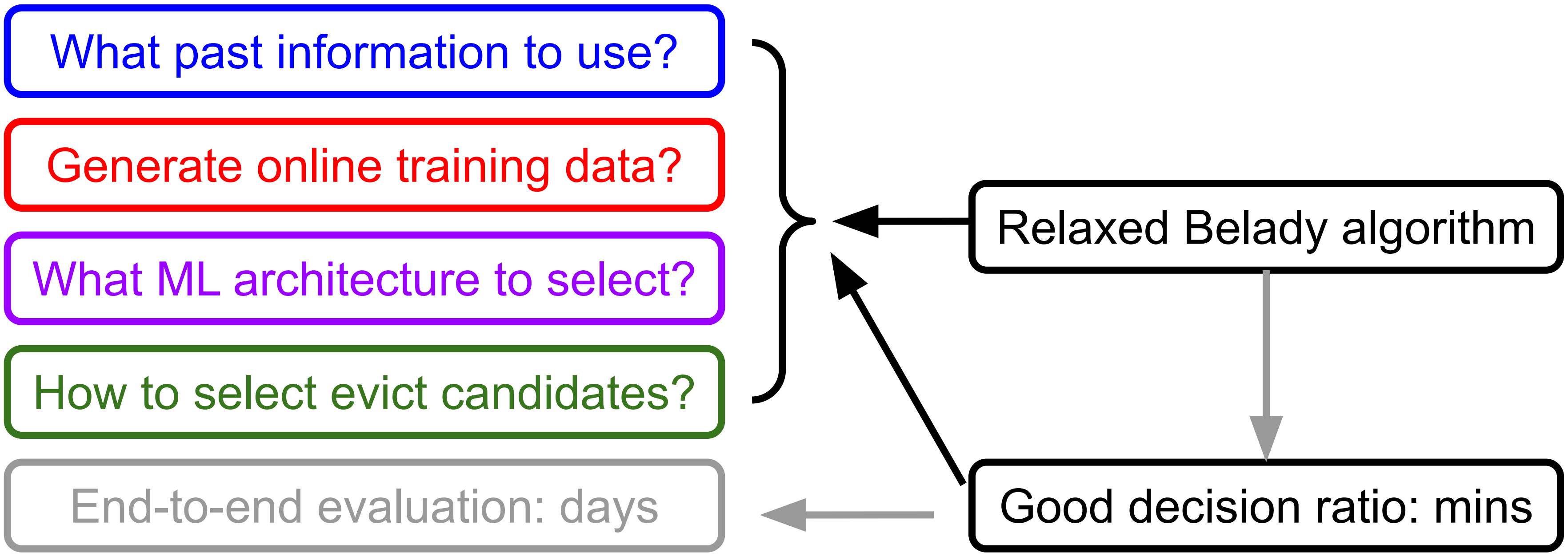
- What past information to use?
- Generate online training data?
- What ML architecture to select?
- How to select evict candidates?
- End-to-end evaluation: days



Solutions: Relaxed Belady Algorithm & Good Decision Ratio

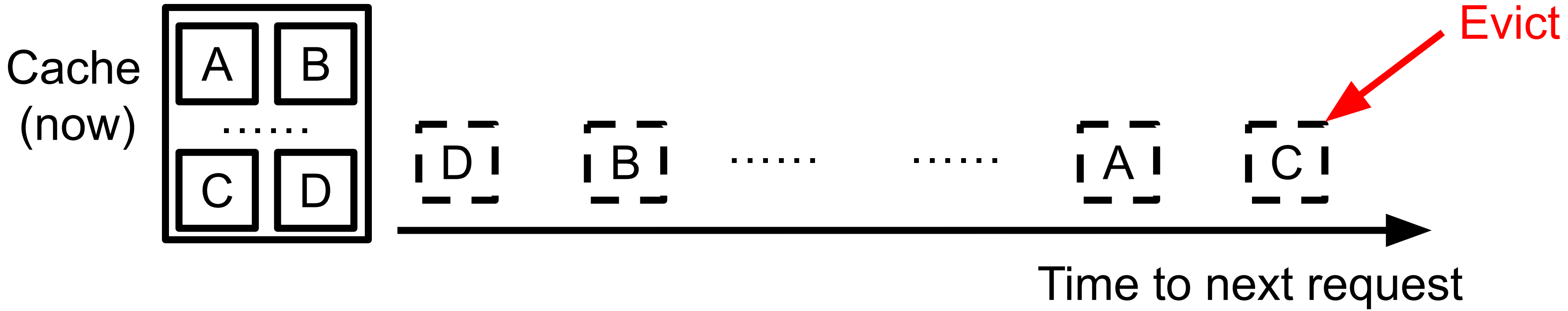


Solutions: Relaxed Belady Algorithm & Good Decision Ratio



Challenge: Hard to Mimic Belady (Oracle) Algorithm

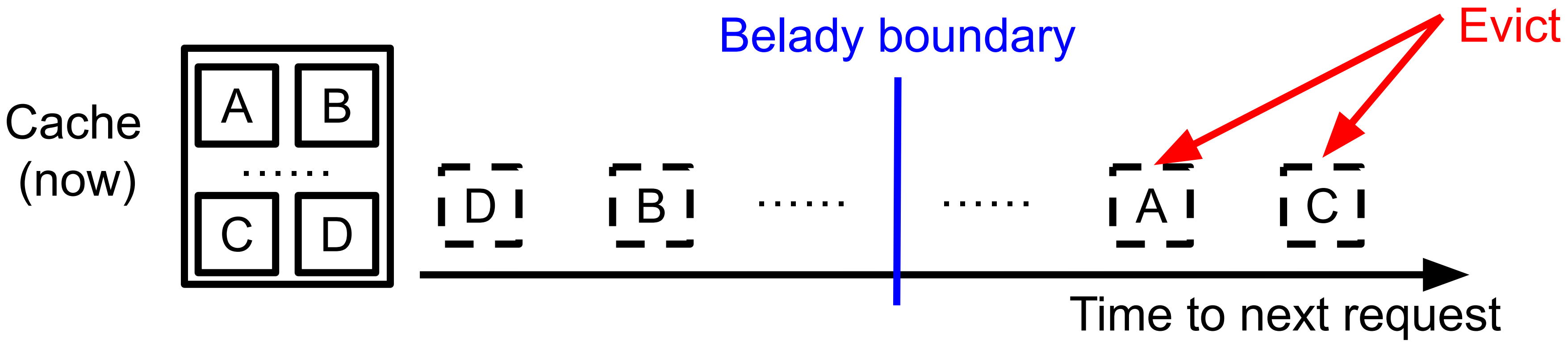
Belady: evict object with next access farthest in the future



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

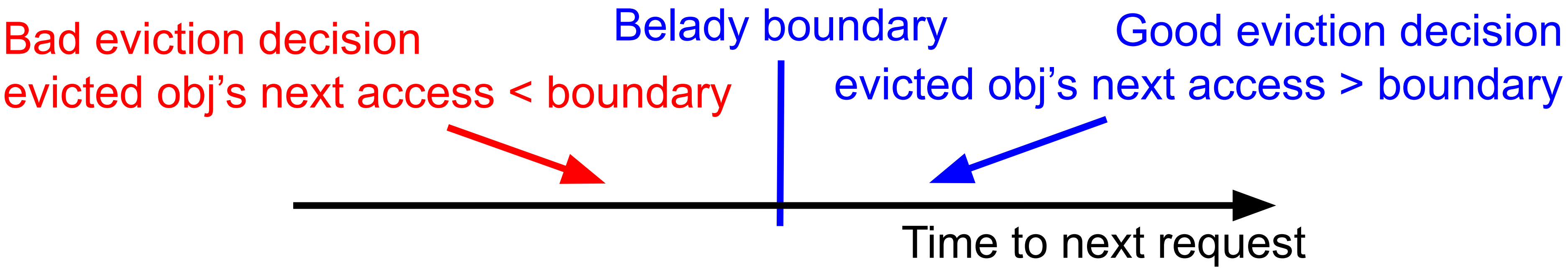


Observation: many objects are good candidates for eviction

Relaxed Belady evicts an objects beyond boundary

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

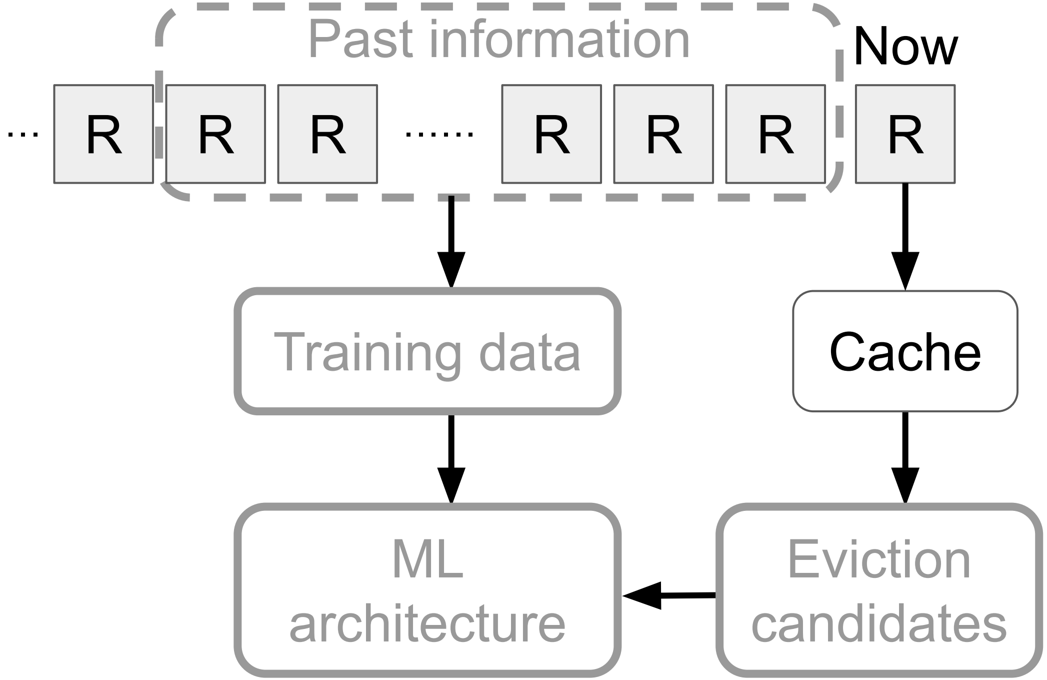
Good Decision Ratio: Directly Measures Eviction Decisions



Insight: relaxed Belady enables evaluating eviction decisions

Good decision ratio: $\dfrac{\text{\# good eviction decisions}}{\text{\# total eviction decisions}}$

Challenge 5: Quickly Evaluate Design Decisions



What past information to use?

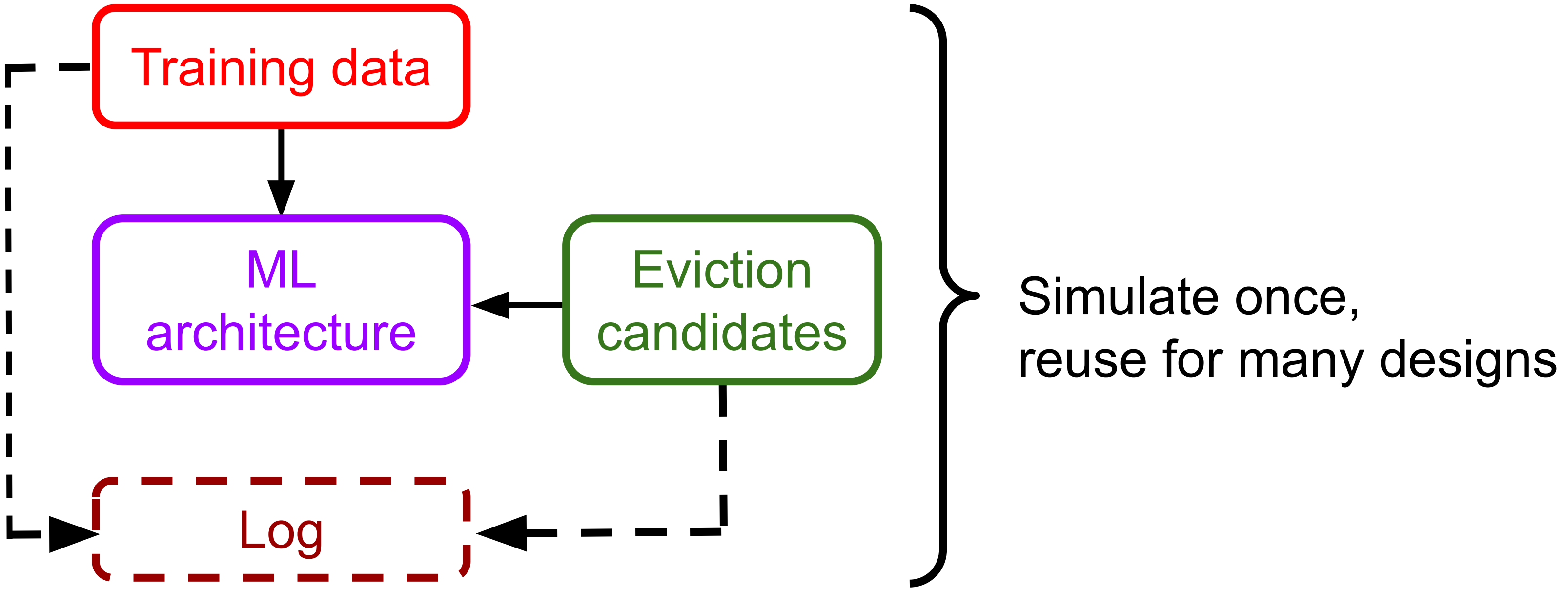
Generate online training data?

What ML architecture to select?

How to select evict candidates?

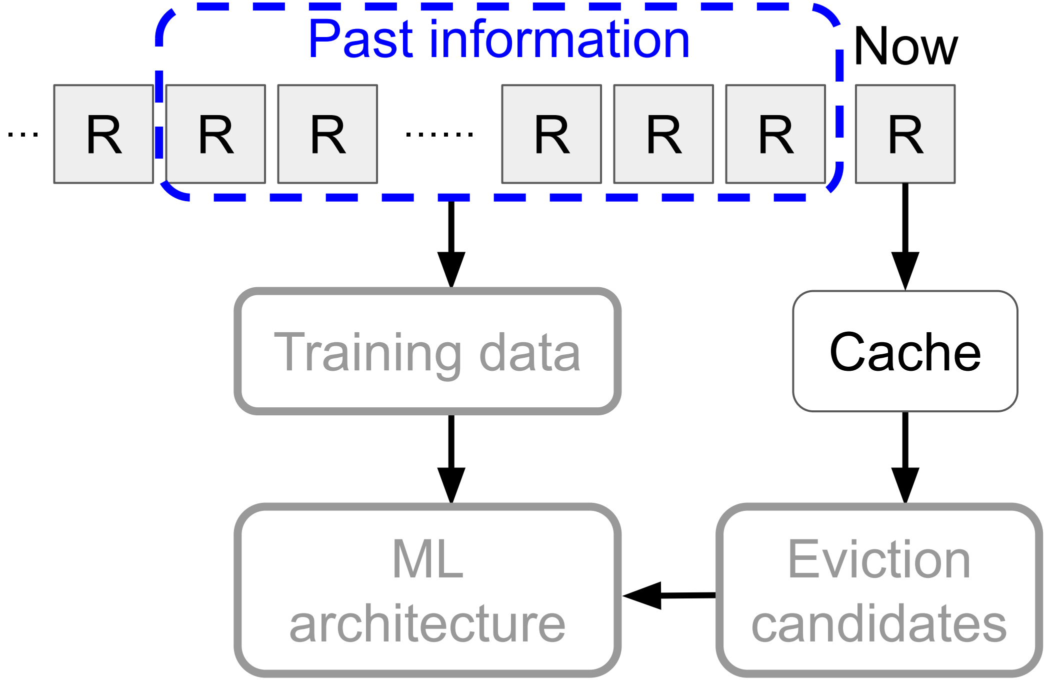
End-to-end evaluation: days

Evaluate Design Decisions w/o Simulation



Evaluate designs on log using good decision ratio in minutes

Challenge 1: Past Information

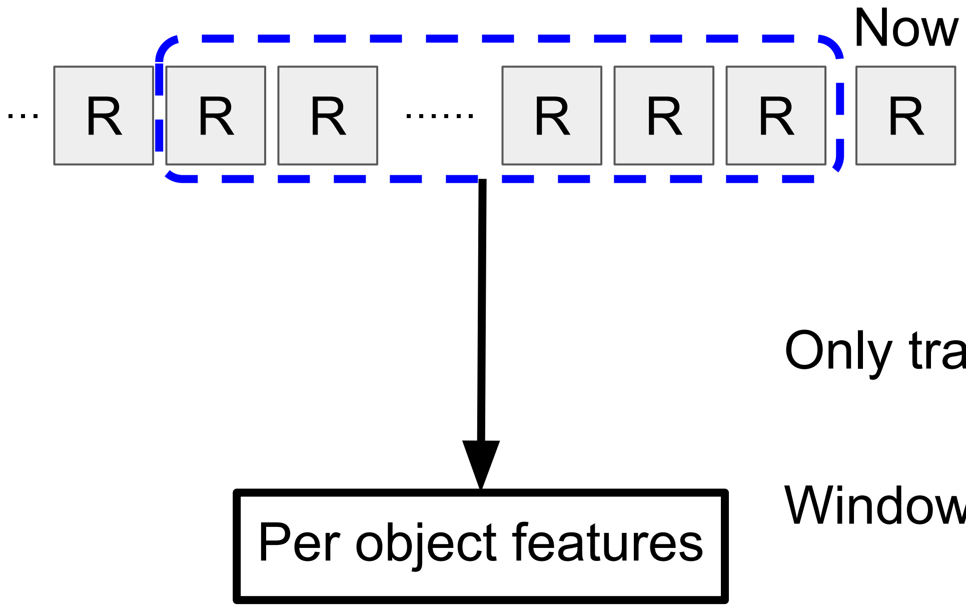


What past information to use?

More data improves training but increases mem overhead

Track Objects within a Sliding Memory Window

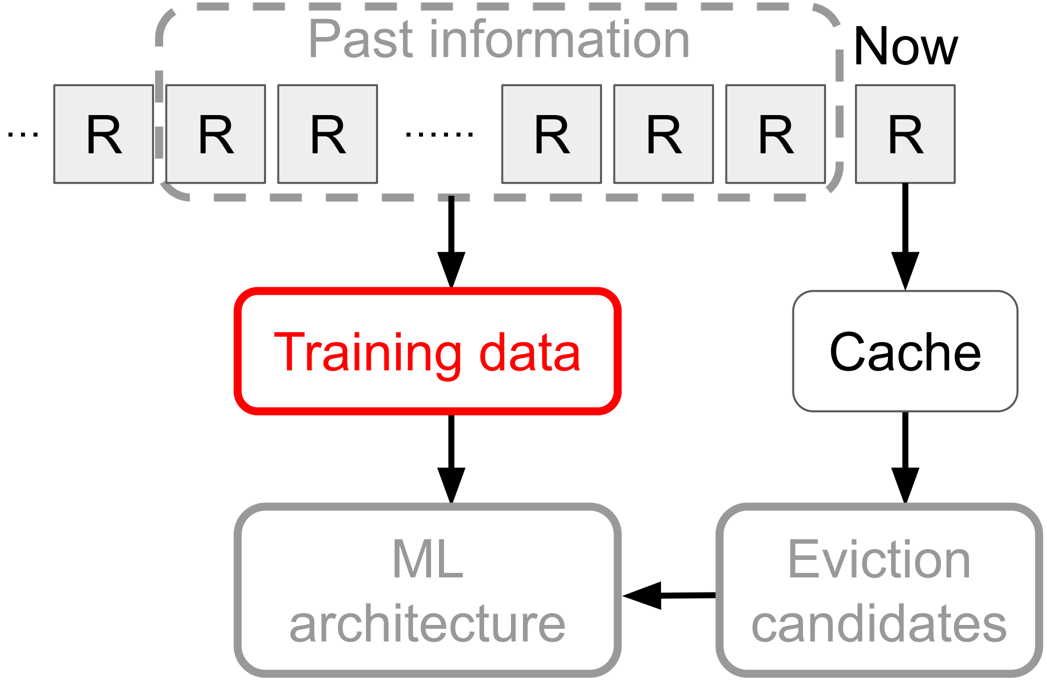
Sliding memory window mimics Belady boundary



Only track objects within memory window

Window size is LRB’s main hyperparameter

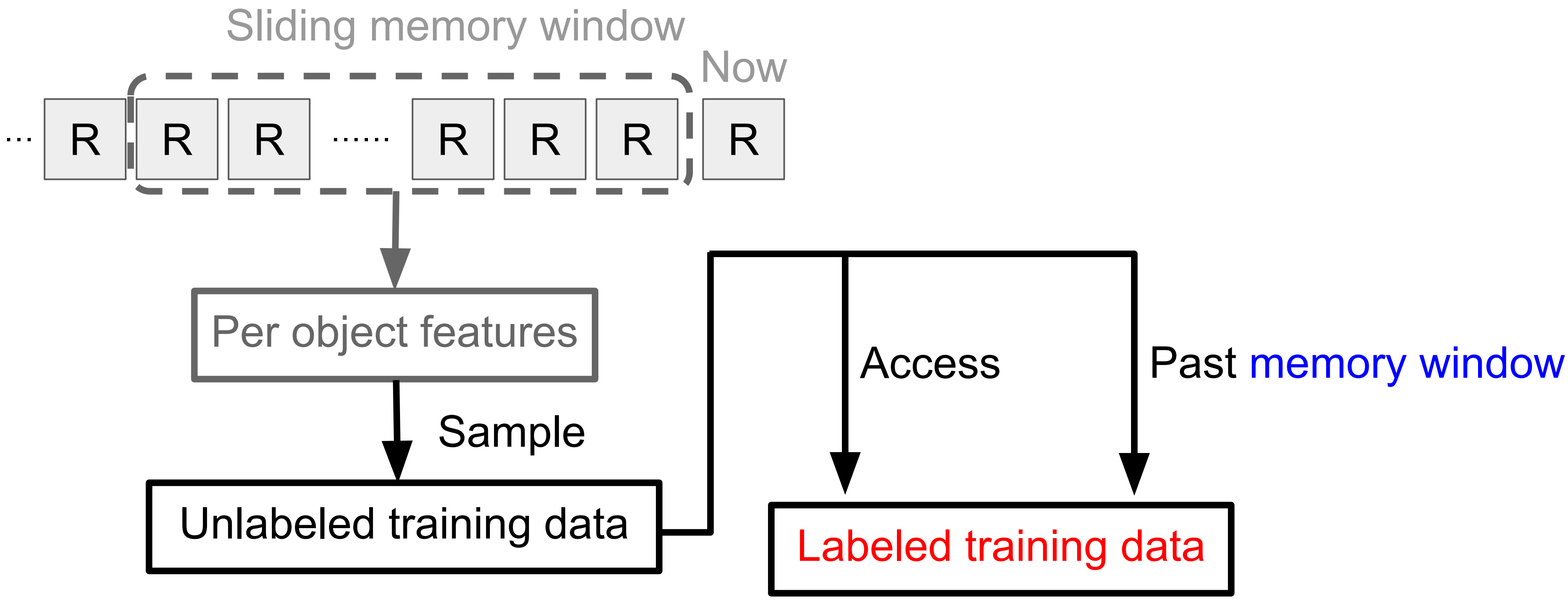
Challenge 2: Training Data



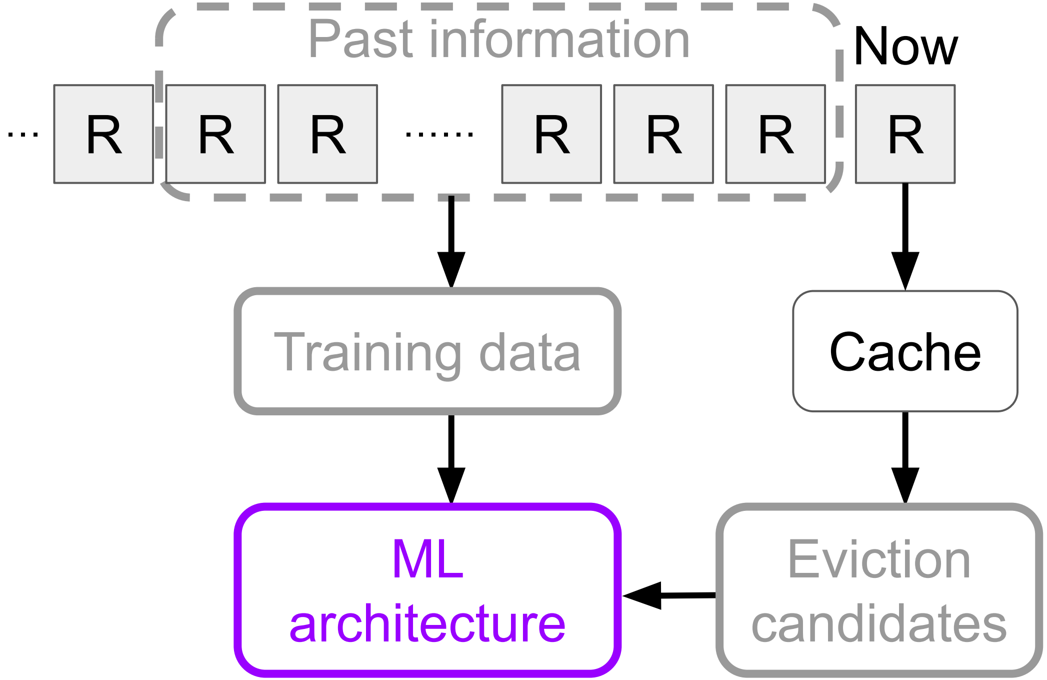
What past information to use?

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?

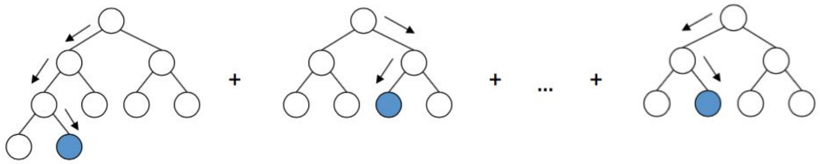
What ML architecture to select?

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)

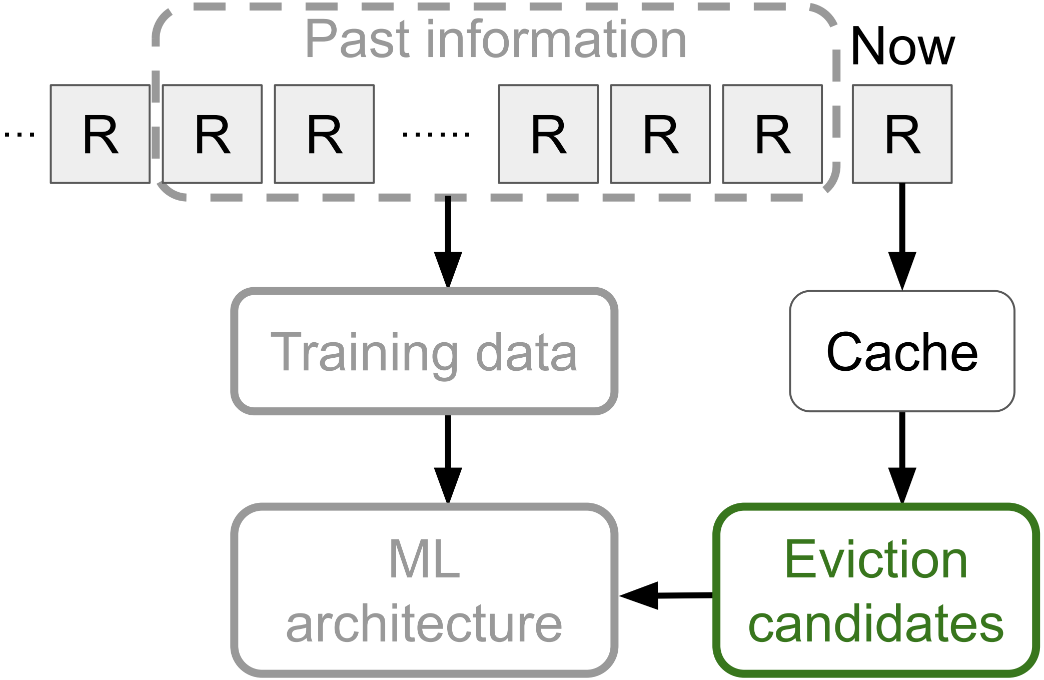


Gradient boosting decision trees

Lightweight & **high good decision ratio**

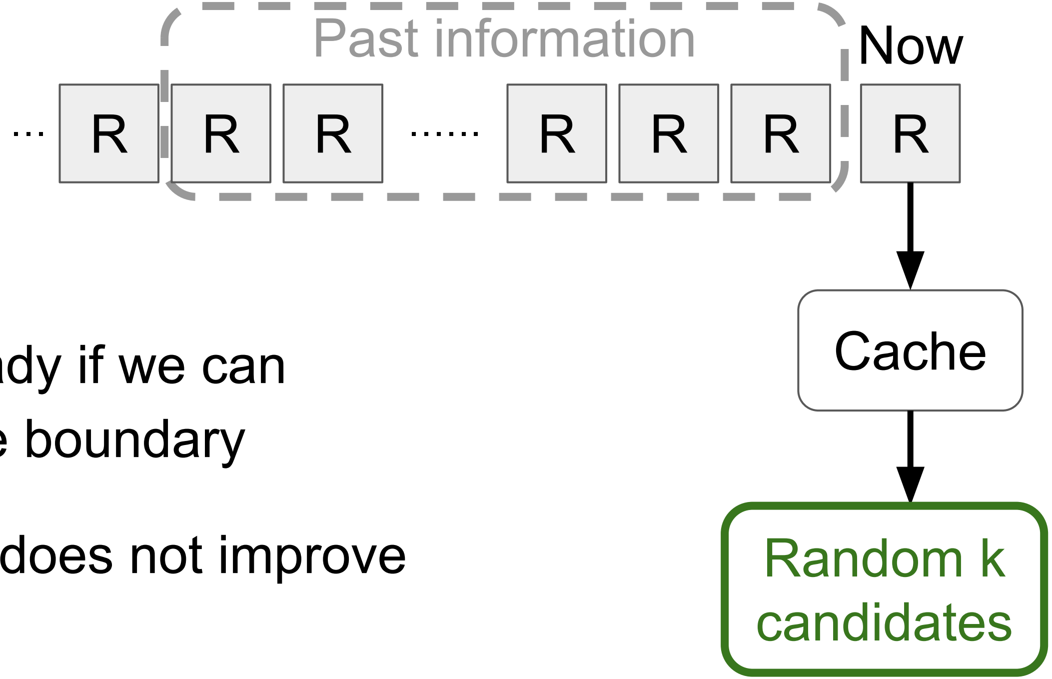
Training ~300 ms, prediction ~30 us

Challenge 4: Eviction Candidates



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

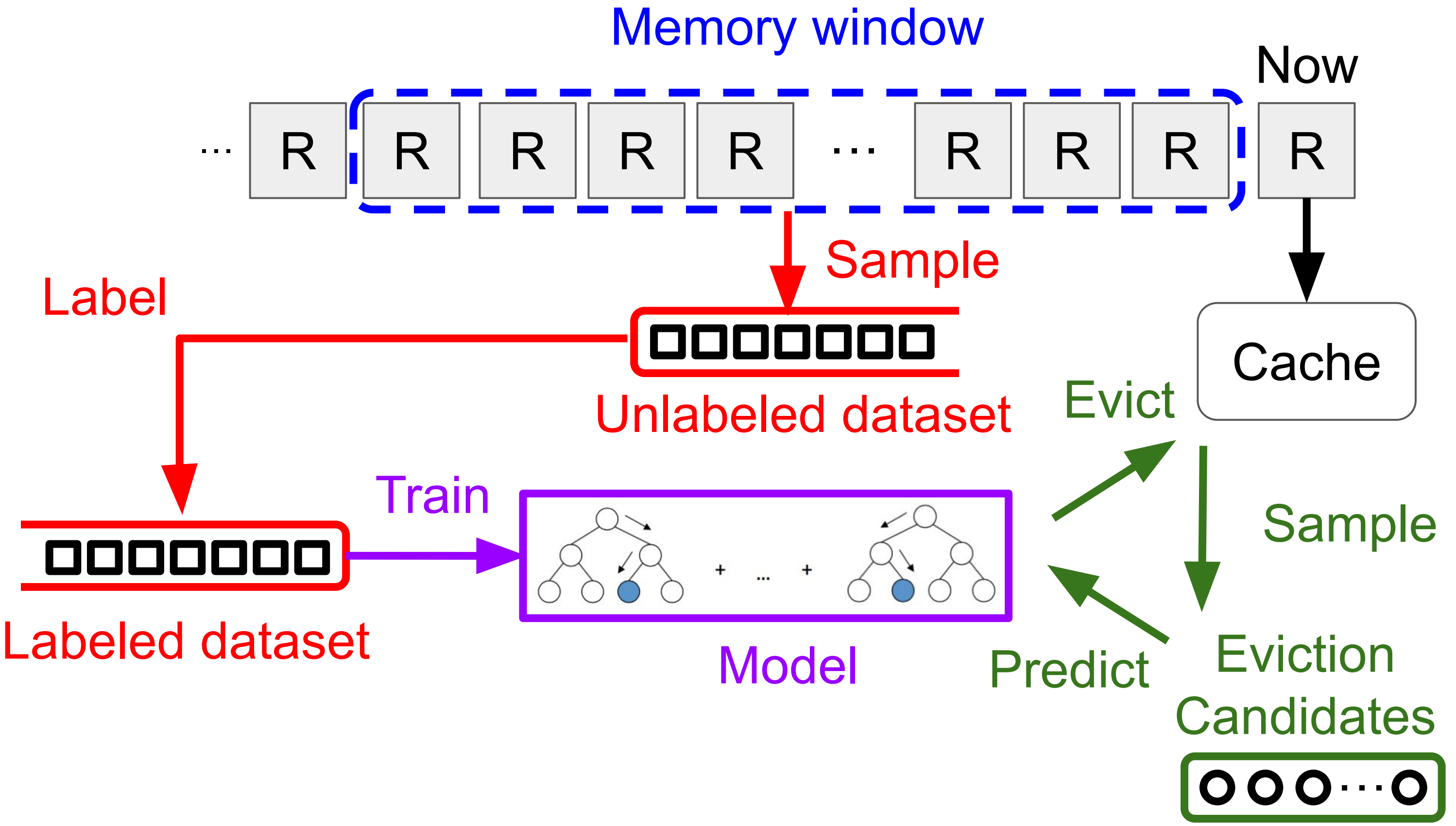
Solution 4: Random Sampling for Eviction



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

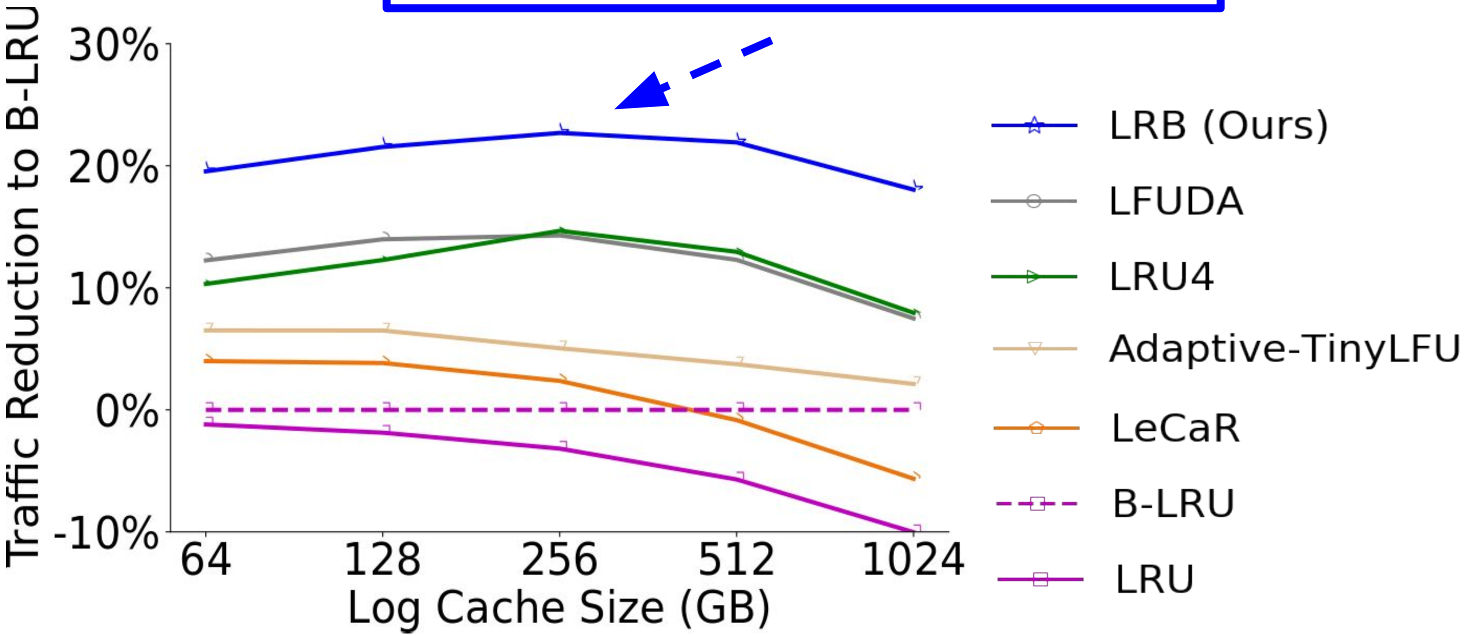
Evaluation Setup

- 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

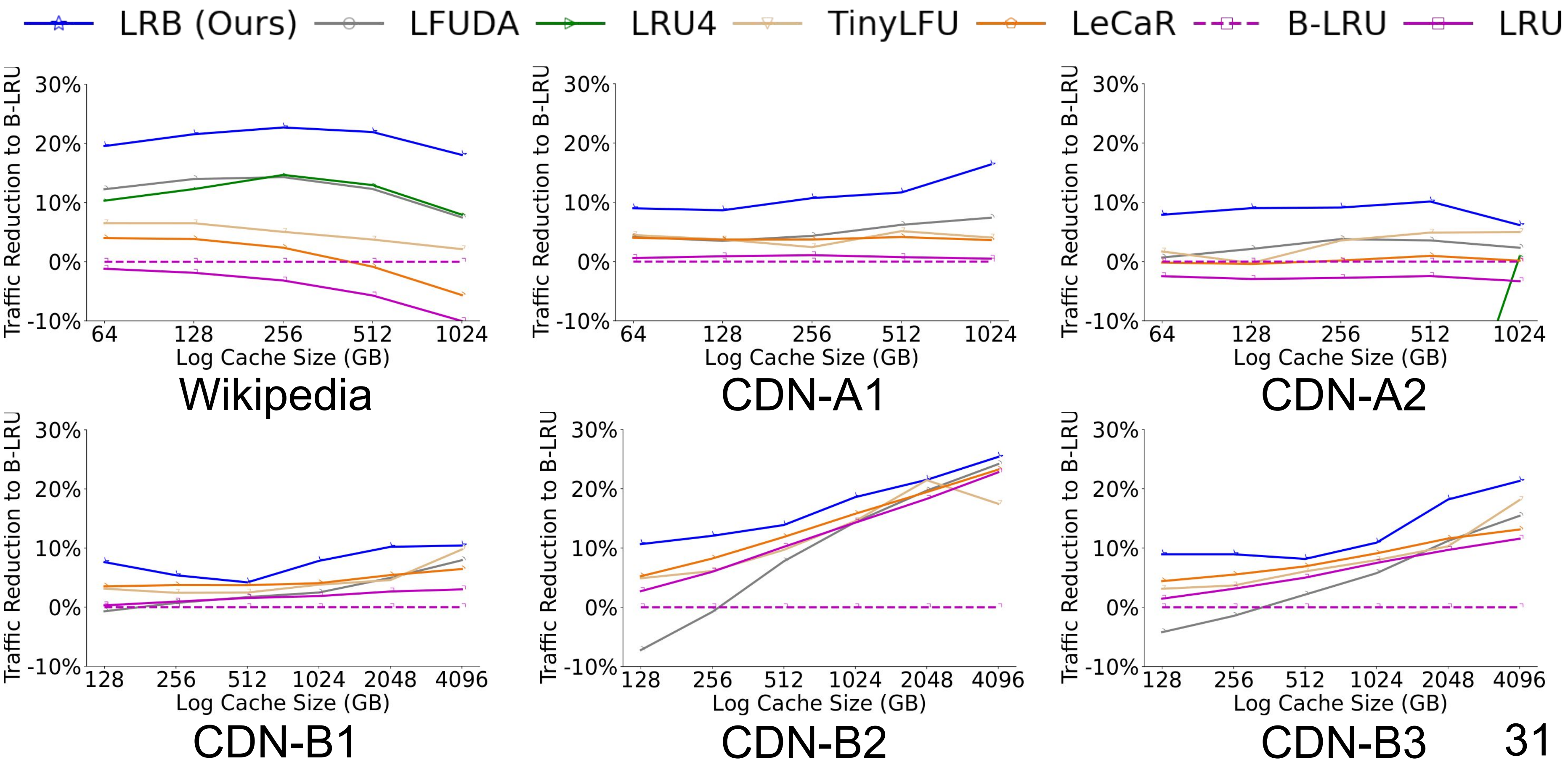
Industry standard

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

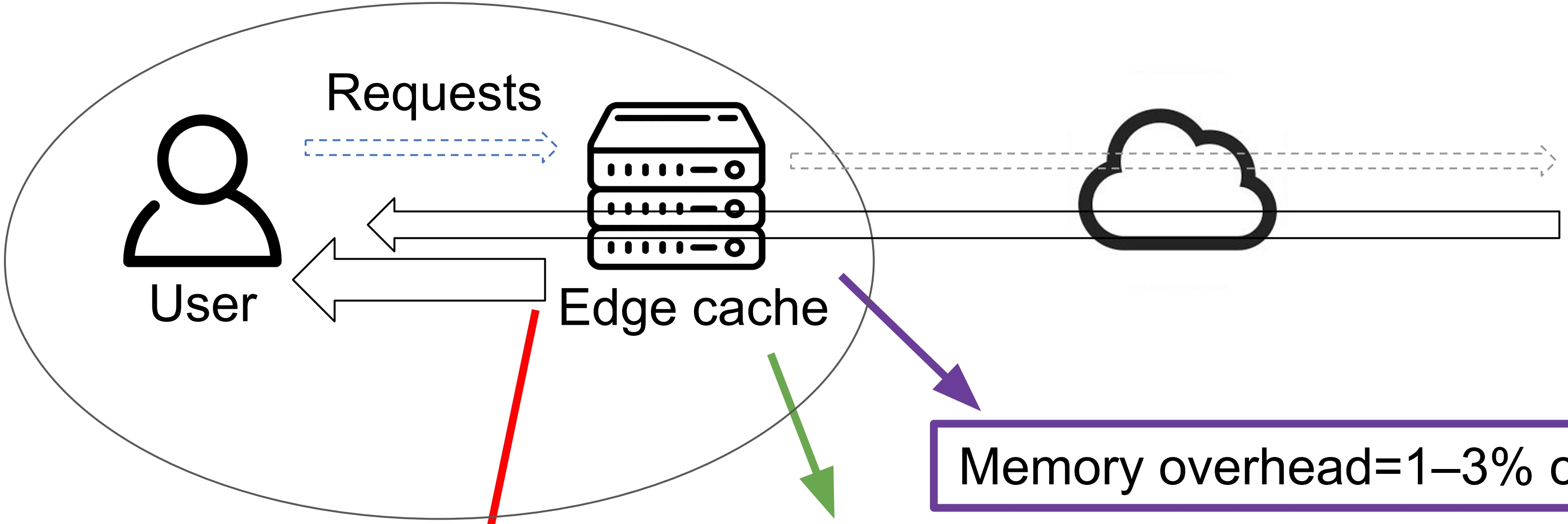


Wikipedia trace

LRB Consistently Improves on the State of the Art



LRB Overhead Is Modest



Memory overhead=1–3% cache size

Peak CPU: 16% vs 9% (unmodified)

Throughput: 11.7 Gbps vs 11.7 Gbps (unmodified)

Conclusion

- LRB reduces WAN traffic with modest overhead
- Key insight: **relaxed Belady**

  → Simplifies machine learning & reduces system overhead

  → **Good decision ratio** enables fast design evaluation & design iteration

Code & Wikipedia trace:

https://github.com/sunnyszy/lrb

Microsoft Research