HALP: Heuristic Aided Learned Preference Eviction Policy for YouTube Content Delivery Network

Zhenyu Song, Princeton University; Kevin Chen, Nikhil Sarda, Deniz Altınbüken, Eugene Brevdo, Jimmy Coleman, Xiao Ju, Pawel Jurczyk, Richard Schooler, and Ramki Gummadi, Google

Presented by Allen Jue

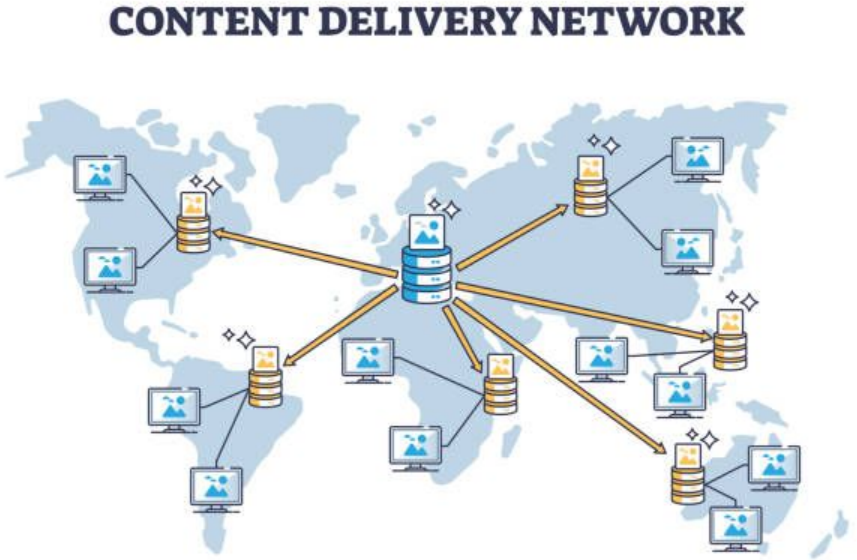
Background: YouTube

- YouTube: A large-scale video streaming service at Google
- In 2021, YouTube served ~15% of the world's traffic
- It serves content to over:
  - 200 countries
  - 2 billion people
  - This number is increasing!
- It needs to serve videos efficiently for quality user experience

Background: CDNs

CDNs

- Caching content in proxy servers
- Multi-level caching
- Evaluated with Byte-Miss Ratio (Bytes missed / Bytes requested)



Related Work

- Learning Relaxed Belady - LRB trains a gradient boosted model to approximate Belady's decisions online using request history.[1]
- ML-based LeCaR - Use ML to switch between common cache eviction policy (LRU and LFU)[2]

[1] Zhenyu Song, Daniel S Berger, Kai Li, and Wyatt Lloyd. Learning relaxed belady for content distribution network caching. In 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), pages 529–544, 2020

[2] Giuseppe Vietri, Liana V Rodriguez, Wendy A Martinez, Steven Lyons, Jason Liu, Raju Rangaswami, Ming Zhao, and Giri Narasimhan. Driving cache replacement with ML-based LeCaR. In USENIX HotStorage, 2018.

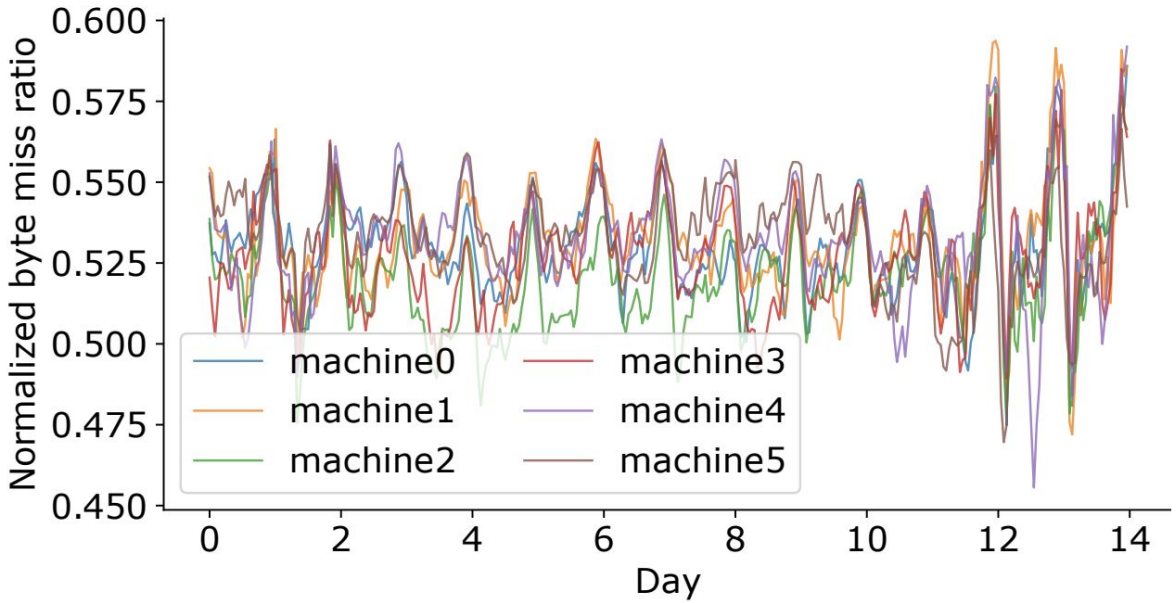
Motivation

- Cache eviction is hard and byte-miss ratio can’t explain everything
- Challenging to deploy ML to make more informed decisions

Challenges:

1. Computation overhead for learning
2. Robust byte miss ratio improvement
3. Measuring impact under production noise

Noisy Machines



Contributions

To combat these challenges, the paper introduces:

- HALP policy: augment heuristics with ML to maintain robustness and limit overheads
  - Recall *Classic Meets Modern: a Pragmatic Learning-Based Congestion Control for the Internet* - The ORCA model does something similar
- Create an impact distribution analysis to evaluate caching algorithms in noisy production environments
- Deploy HALP in YouTube

YouTube CDN Architecture

- Each rack has homogeneous cache servers
- Different racks can have different hardware
- Pretty standard caching + eviction
- DRAM is valuable, want better eviction algorithms

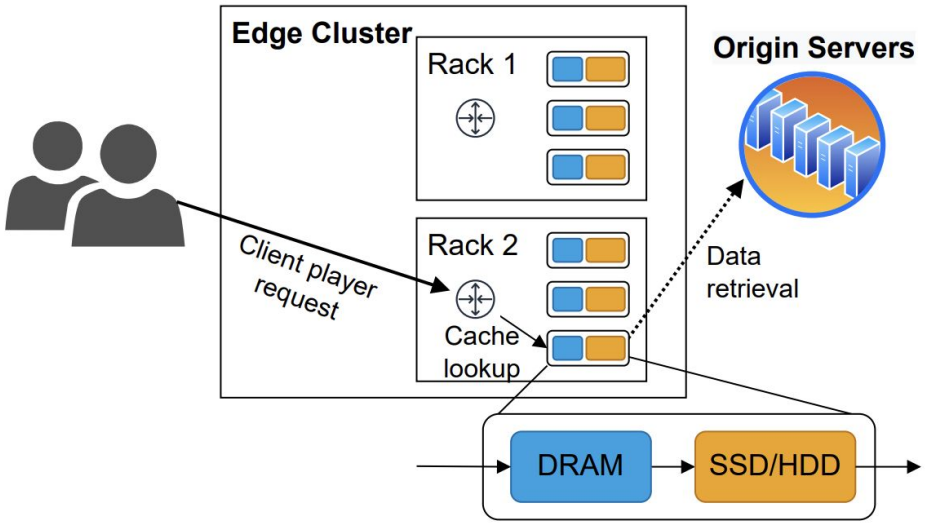


Figure 2: A YouTube CDN edge cluster contains multiple racks of servers. Machines in a rack are of the same type.

Current Implementation

During that client request:

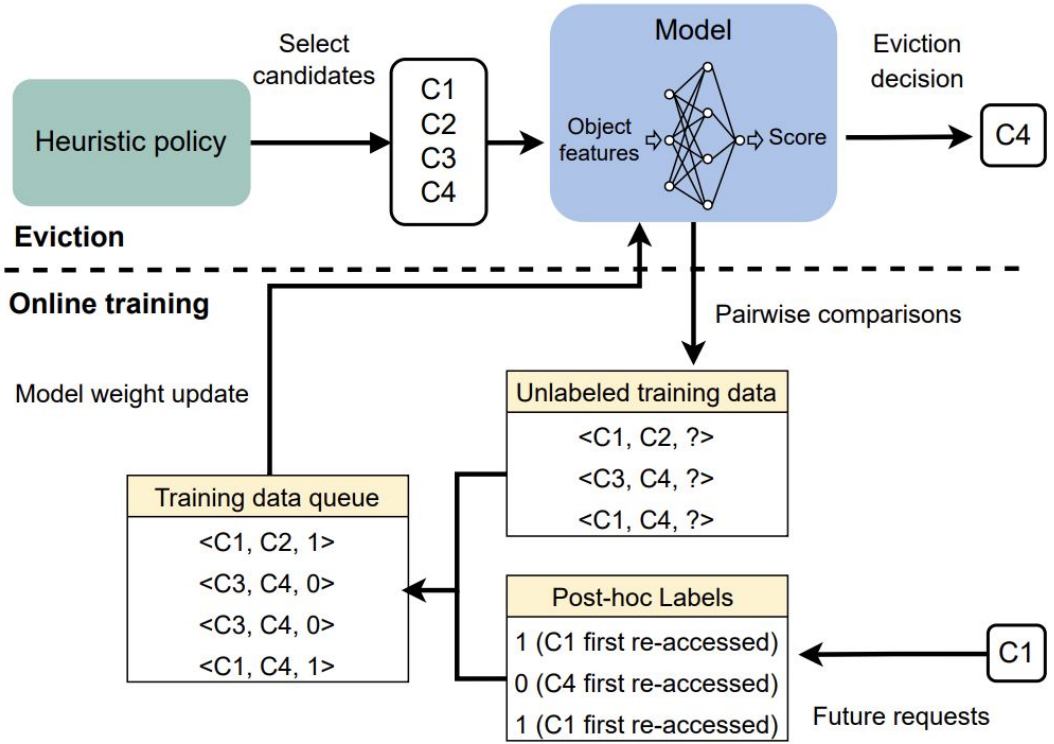
1. Want to minimize p95 byte miss ratio during peak hours
2. Can’t use QoE because it’s too noisy
3. Previous Google production heuristics calculates a score with an emphasis on the last chunk
4. Learning algorithms good but expensive

HALP Architecture

1. Heuristic policy to narrow down the number of candidates
2. Use NN model to predict from the candidates

Heuristic policy lessens weaknesses of learned models.

Model utilizes the strengths of learned cache eviction.



Challenge 1: Heuristic Policy

1. To lower overhead, notice that cheap heuristics can be used to remove obvious bad candidates
2. Number of candidates is a hyperparameter (empirically n=4)

Challenge 2: Ranking-based Learned Eviction

- Frame problem of “which to evict” to “between 2 items, which is worse to keep?”
- Do a tournament style, so worse items get compared with each other until the “worst” is evicted. (if 4 elements, only 3 comparisons are done)
- When a decision is made no truth known yet, so to train:
  - Save comparisons as unlabeled data
  - Wait for future access to label
  - Dropped after a while if no access
- Spinlocks and RCUs easier to reason about correctness?

ML Model

- Binary classification task
- Small model
- Fast training (ms)
- Fast inference (ns)

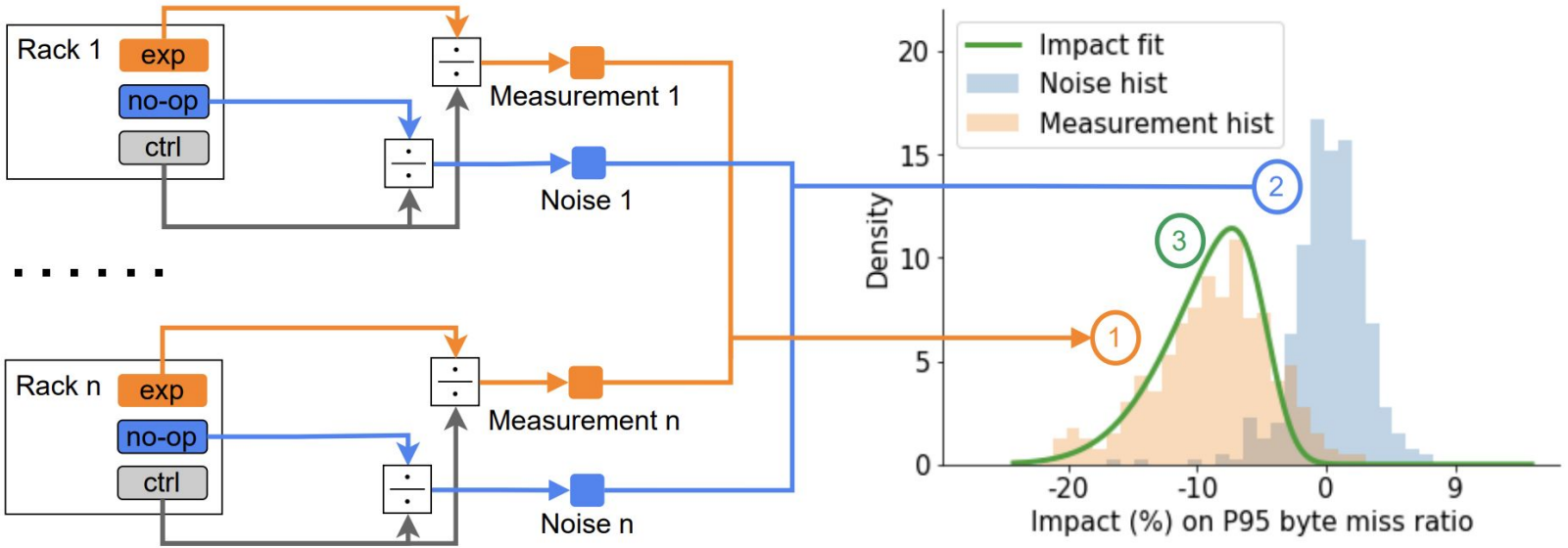
Feature name	Dimension
Access-based	
Time between accesses	32
Exponential decay counters	10
Number of accesses	1
Average time between accesses	1
Time since last access	1
Video-specific	
End of chunk	1

Table 1: Features used by HALP.

Challenge: Impact Distribution Analysis

- Operating conditions are diverse and noisy
- Naively A/B test on every rack = not enough data
- Model as ***M = I + N***
  - ***M*** = Observed Impact
  - ***I*** = Real Impact
  - ***N*** = Noise
- You can isolate ***I*** by performing a deconvolution of ***M***

Impact Distribution Analysis



Fitting Algorithm

Algorithm 1 Algorithm for fitting impact distribution

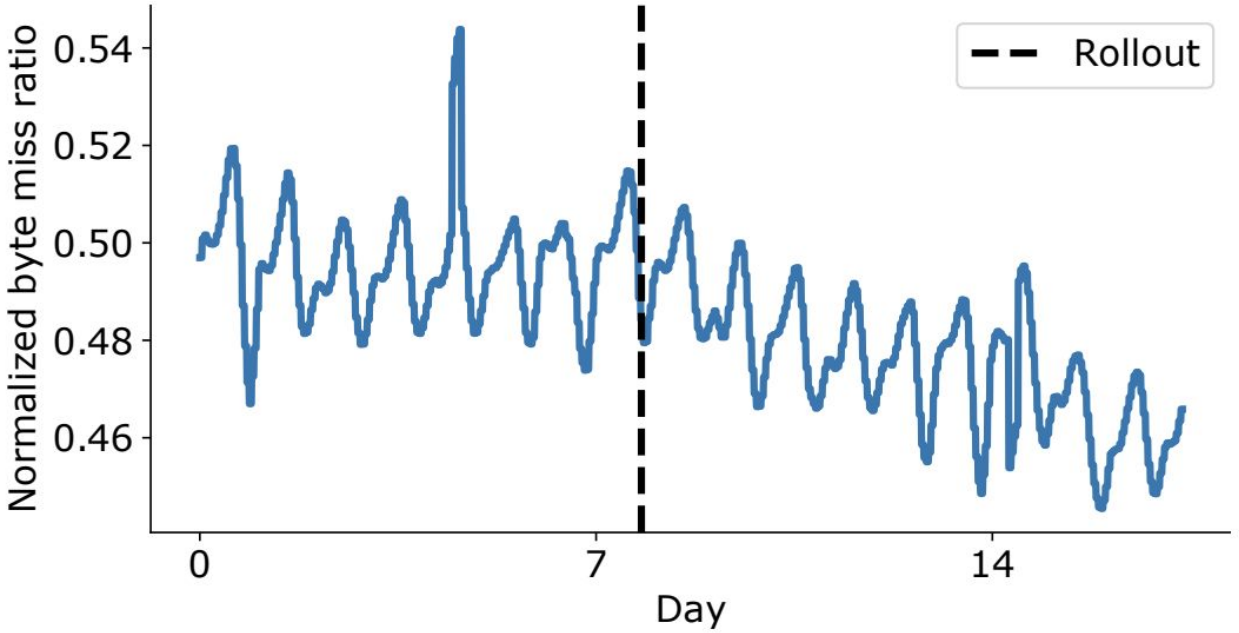
Input: measurement samples M, noise samples N, and distribution candidates.

Output: measurement distribution P_M, impact distribution P_I, noise distribution P_N.

1: $P_N = FitByMLE(dist=\text{“t-dist”}, N)$
2: **for** candidate_dist in candidate_distributions **do**
3: // Approximate P_M by discretized grid G.
4: $P_I = FitByMLE(dist=\text{candidate\_dist}, M, P_N)$, with $P_M(m) \approx \sum_{v \in G} P_N(v) P_I(m-v)$
5: **end for**
6: **return** P_I with the highest likelihood

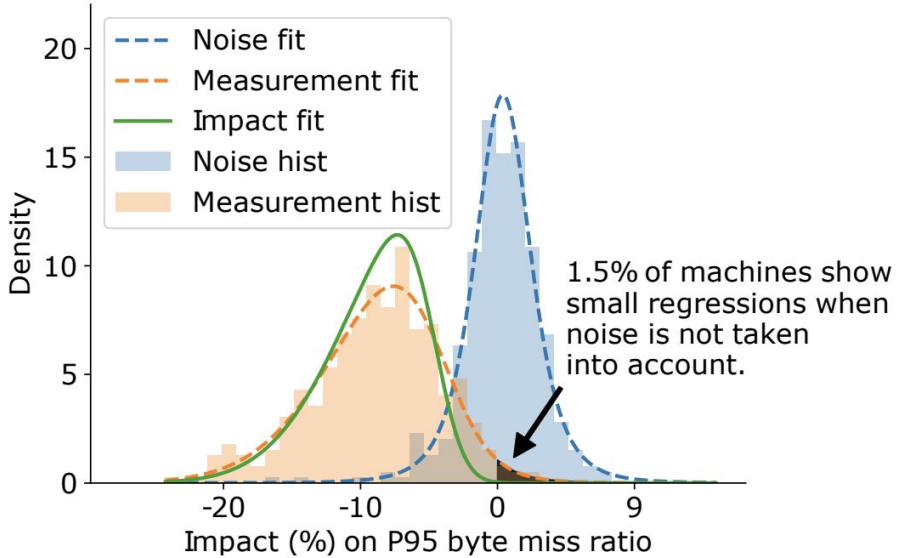
- This is only possible because they have large amount of samples from 200+ countries.

Results: Production



Results: Impact Distribution Analysis

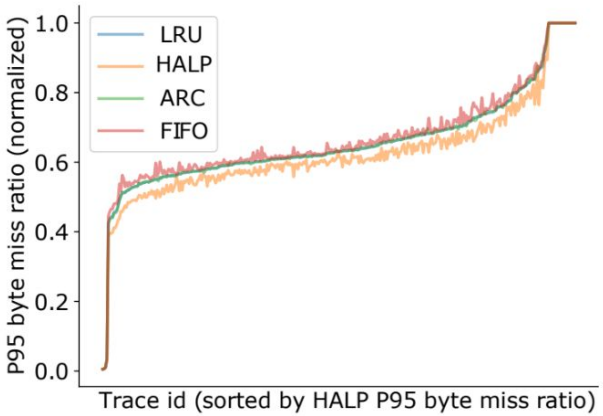
- HALP can reduce regressions and has an average reduction of 9.1% byte miss ratio



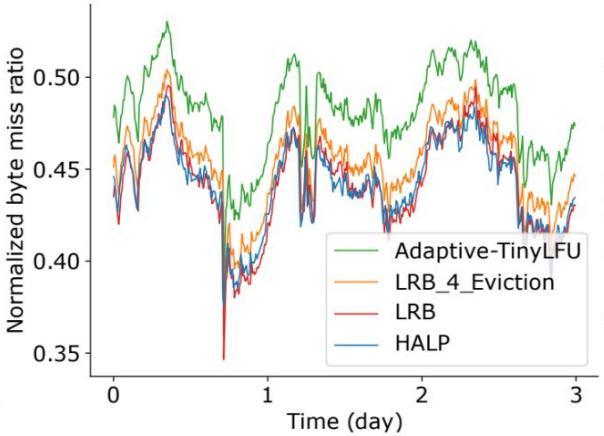
Results

- **Disk first byte latency** - 13% reduction to a 5% *increase,* avg 3.8% decrease
- **Join latency** - 1.03% to 1.41% reduction, with an average reduction of 1.22%
- Consistently low overhead (1.8%) - roughly linear

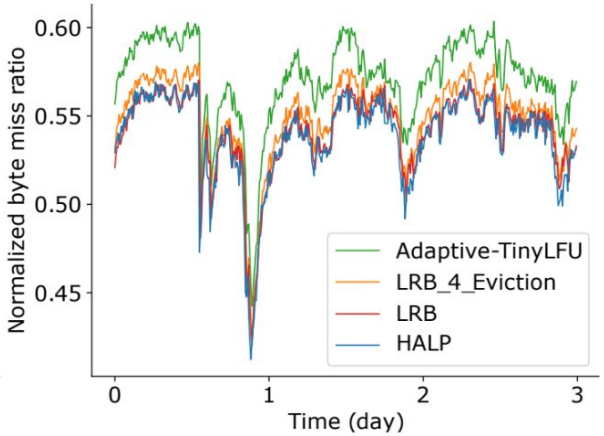
Compared to Classic Cache Algorithms



(a)

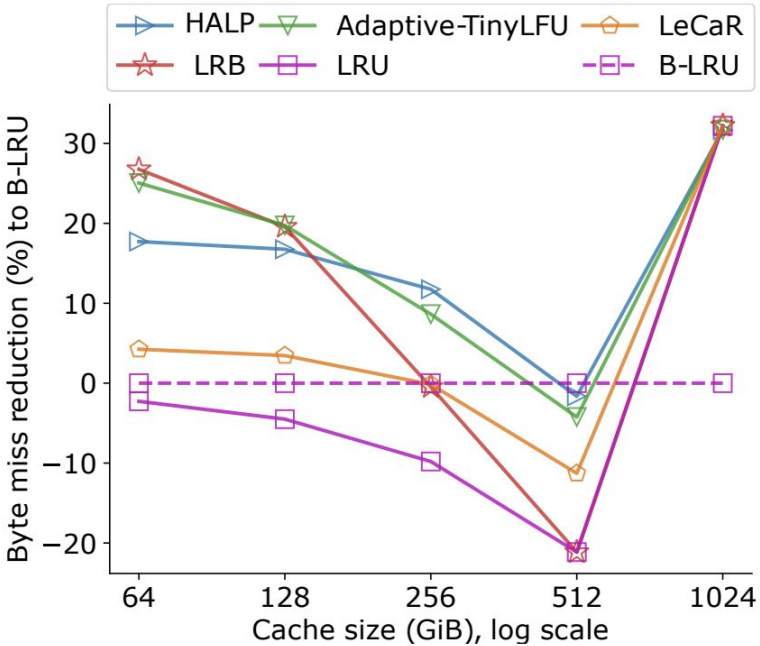


(b) Developed market

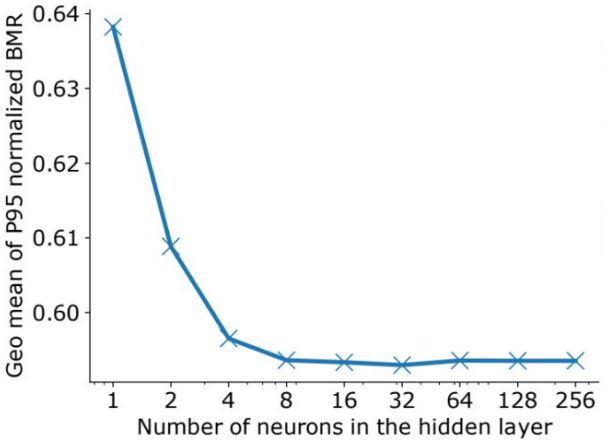


(c) Emerging market

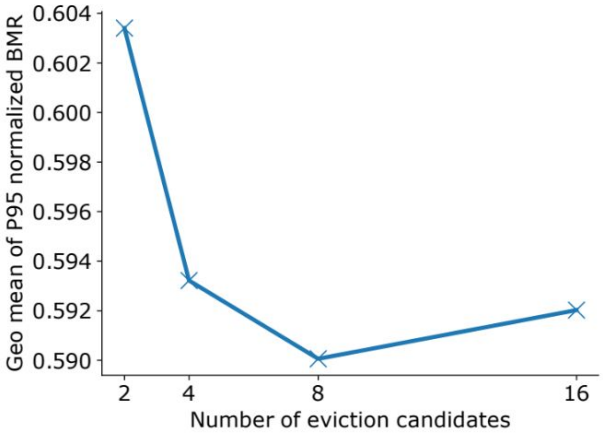
Compared to SOTA Cache Algorithms



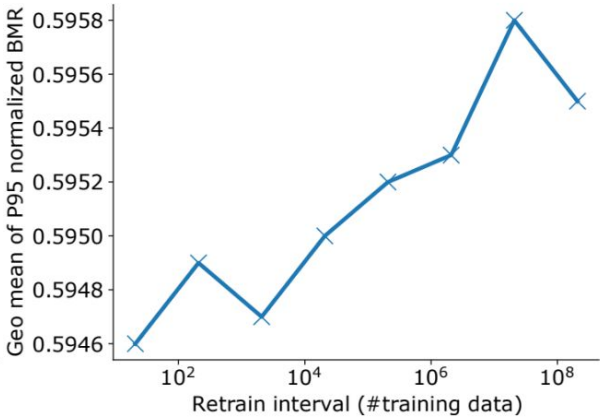
Hyperparameter Selection



(a)



(b)



(c)

Strengths

- Production-ready deployment with low overheads and conservative constants
- Robustness of heuristics mixed with adaptability of learned systems
- Impact distribution analysis is detailed and systematically removes noise

Critiques

- They empirically find that LRU is useful. Other cache heuristics?
- Only Google can really use the impact distribution fitting because they have large samples from 200+ countries
- The effectiveness is workload specific (YouTube has spatial locality)

Future Work

- Explore different heuristics strategies
- Apply different configurations to see if it generalizes to other workloads
- Modify to work with smaller sample sizes
- Is learned cache eviction the way to go?
    - SIEVE (Zhang et al., NSDI ‘24) - an algorithm that is simpler than LRU and provides better than state-of-the-art efficiency and scalability for web cache workloads.
    - 3L-Cache (Zhou et al., FAST ‘25) - lowers overheads, byte miss ratio, and object miss ratio

Key Takeaways