Learning Relaxed Belady for Content Delivery Network Caching

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

NSDI 2020
CDN Caching Goal: Minimize WAN Traffic

Wide Area Network (WAN) traffic is expensive

Key metric byte miss ratio: \( \frac{\text{miss bytes}}{\text{total bytes}} \)
Caching Remains Challenging

Heuristic-based algorithms (1965–): LRU, LRU\textsubscript{K}, GDSF, ARC, ...
\begin{itemize}
\item Work well for some workloads, but work poorly for others
\end{itemize}

ML-based adaptation of heuristics (2017–): UCB, LeCAR, ...
\begin{itemize}
\item Also work well for some workloads, but poorly for others
\end{itemize}

Belady’s MIN algorithm (1966)
\begin{itemize}
\item Oracle: requires future knowledge
\item Large gap in byte miss ratio between state-of-the-art and Belady:
\item 20–40\% on production traces
\end{itemize}
Introducing Learning Relaxed Belady (LRB)

New approach: mimic Belady using machine learning
General Overview of our Approach

Past information

... R R R ..... R R R R R

Now

Training data

ML architecture

Cache

Eviction candidates
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?

Past information

Now

Training data

Cache

ML architecture

Eviction candidates
Challenge 3: ML Architecture

Past information

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

Relaxed Belady algorithm

Good decision ratio
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

Relaxed Belady algorithm

Good decision ratio: mins
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

Relaxed Belady algorithm
Good decision ratio: mins
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 $\rightarrow$ reasonable computation
- No need to differentiate beyond boundary $\rightarrow$ simplifies the prediction
Good Decision Ratio: Directly Measures Eviction Decisions

**Insight:** relaxed Belady enables evaluating eviction decisions

**Good decision ratio:** \[rac{\text{# good eviction decisions}}{\text{# total eviction decisions}}\]
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

Training data

ML architecture

Eviction candidates

Log

Simulate once, reuse for many designs

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?

Past information

Training data

ML architecture

Cache

Eviction candidates
Sample Training Data & Label on Access or Boundary

Sliding memory window

... R R R ...... R R R R R

Per object features

Sample

Unlabeled training data

Access

Past memory window

Labeled training data
Challenge 3: ML Architecture

Large potential design space

Past information

Now

What past information to use?
Generate online training data?
What ML architecture to select?
Large potential design space

Training data

Cache

ML architecture

Eviction candidates
Solution 3: Feature & Model Selection

Use good decision ratio to evaluate new designs

<table>
<thead>
<tr>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object size</td>
</tr>
<tr>
<td>Object type</td>
</tr>
<tr>
<td>Inter-request distances (recency)</td>
</tr>
<tr>
<td>Exponential decay counters (long-term frequencies)</td>
</tr>
</tbody>
</table>

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

Label

Labeled dataset

Unlabeled dataset

Memory window

Now

Sample

Cache

Train

Model

Predict

Eviction Candidates

Sample

Evict
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

Traffic Reduction to B-LRU

LRB (Ours)
LFUDA
LRU4
Adaptive-TinyLFU
LeCaR
B-LRU
LRU

Wikipedia trace

Log Cache Size (GB)
64 128 256 512 1024
LRB Consistently Improves on the State of the Art

![Graphs showing traffic reduction to B-LRU across different cache sizes for LRB (Ours), LFUDA, LRU4, TinyLFU, LeCaR, B-LRU, and LRU.](image)

- **Wikipedia**
- **CDN-A1**
- **CDN-A2**
- **CDN-B1**
- **CDN-B2**
- **CDN-B3**
LRB Overhead Is Modest

Throughput: 11.7 Gbps vs 11.7 Gbps (unmodified)

Memory overhead = 1–3% cache size

Peak CPU: 16% vs 9% (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