AdaptSize: Orchestrating the Hot Object Memory Cache in a CDN

Daniel S. Berger
Mor Harchol-Balter
Ramesh K. Sitaraman

CDN Caching Architecture

Content providers

CDN

Users
Optimizing CDN Caches

Two caching levels:
- Disk Cache (DC)
- Hot Object Cache (HOC)

HOC performance metric
object hit ratio = OHR = 

Goal: maximize OHR
Prior Approaches to Cache Management

Frequent **decisions** required

What to admit
What to evict

Today in practice
- e.g., Nginx, Varnish

2000s in academia
- e.g., Modha, Zhang, Kumar

2010s in academia
- e.g., Kaminsky, Lim, Andersen

- **everything**
- **everything**
- **everything**
- **LRU**
- **mixtures of LRU/LFU**
- **concurrent LRU**

DC

HOC

a few GBs capacity

500 GB per hour
We Are Missing a Key Issue

Not all objects are the same

- Should we admit every object? (no, we should favor small objects)
- A few key companies know this (but don’t know how to do it well)
- Academia has not been helpful (almost all theoretical work assumes equal-sized objects)
What’s Hard About Size-Aware Admission

Fixed Size Threshold:
admit if $\text{size} < \text{Threshold } c$

The best threshold changes with traffic mix

How to pick $c$:
pick $c$ to maximize OHR

The best threshold changes with traffic mix
Can we avoid picking a threshold $c$?

Probabilistic admission:

Unfortunately, many curves example: $\exp(c)$ family

Which curve makes big difference

We need to adapt $c$
The AdaptSize Caching System

First system that continuously adapts the parameter of size-aware admission

- Take traffic measurements
- Calculate the best c
- Enforce admission control

Adapt with traffic

Adapt with time
How to Find Best $c$ Within Each $\Delta$ Interval

**Traditional approach**
- Hill climbing
  - Local optima on OHR-vs-c curve

**AdaptSize approach**
- Markov model
  - Enables speedy global optimization
How AdaptSize Gets the OHR-vs-c curve

Markov chain

➢ track IN/OUT for each object

Algorithm

For every $\Delta$ interval and for every value of $c$

❐ use Markov chain to solve for $OHR(c)$

❐ find $c$ to maximize $OHR$

Why hasn’t this been done?

Too slow: exponential state space

New technique: approximation with linear state space
Implementing AdaptSize

Incorporated into Varnish
highly concurrent HOC system, 40+ Gbit/s

Take traffic measurements ➔ Calculate the best c ➔ Enforce admission control
Implementing AdaptSize

Incorporated into Varnish
highly concurrent HOC system, 40+ Gbit/s

Challenges
1) Concurrent write conflicts
2) Locks too slow  [NSDI'13 & 14]

AdaptSize: producer/consumer + ring buffer

Lock-free implementation
Implementing AdaptSize

Incorporated into Varnish

highly concurrent HOC system, 40+ Gbit/s

Take traffic measurements ➔ Calculate the best c ➔ Enforce admission control

AdaptSize: admission is really simple

- given c, and the object size
- admit with P(c, size)

Enables lock free & low overhead implementation
AdaptSize Evaluation Testbed

**Origin:** emulates 100s of web servers
55 million / 8.9 TB unique objects

**DC:** unmodified Varnish
4x 1TB / 7200 Rpm

**HOC systems:**
- unmodified Varnish
- NGINX cache
- AdaptSize

**Clients:** replay Akamai requests trace
440 million / 152 TB total requests
Comparison to Production Systems

<table>
<thead>
<tr>
<th></th>
<th>what to admit</th>
<th>what to evict</th>
</tr>
</thead>
<tbody>
<tr>
<td>Varnish</td>
<td>everything</td>
<td>concurrent LRU</td>
</tr>
<tr>
<td>Nginx</td>
<td>frequency filter</td>
<td>LRU</td>
</tr>
<tr>
<td>AdaptSize</td>
<td>adaptive size-aware</td>
<td>concurrent LRU</td>
</tr>
</tbody>
</table>

![Graph showing Object Hit Ratio]

- **Varnish**: +92%
- **Nginx**: +48%
- **AdaptSize**: Highest Hit Ratio
Comparison to Research-Based Systems

recency and frequency combinations

- LRU: manually tuned parameters
- LRU-K: manually tuned parameters
- SLRU: manually tuned parameters
- 2Q: manually tuned parameters
- TLFU: manually tuned parameters
- AdaptSize: +67%
Robustness of AdaptSize

Size-Aware OPT: offline parameter tuning
AdaptSize: our Markovian tuning model
HillClimb: local-search using shadow queues
Conclusion

Goal: maximize OHR of the Hot Object Cache

Approach: size-based admission control
Conclusion

Goal: maximize OHR of the Hot Object Cache

Approach: size-based admission control

Key insight: need to adapt parameter c

AdaptSize: adapts c via a Markov chain

Result: 48-92% higher OHRs
Conclusion

Goal: maximize OHR of the Hot Object Cache

Approach: size-based admission control

Key insight: need to adapt parameter \( c \)

AdaptSize: adapts \( c \) via a Markov chain

Result: 48-92\% higher OHRs

In our paper

- Throughput
- Disk utilization
- Byte hit ratio
- Request latency

GitHub: /dasebe/AdaptSize

OHR = \( \frac{\text{# reqs served by HOC}}{\text{total # reqs}} \)