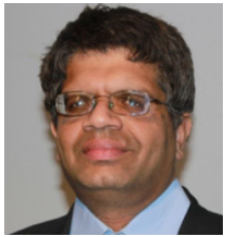
AdaptSize: Orchestrating the Hot Object Memory Cache in a CDN

Daniel S. Berger

Mor Harchol-Balter



Ramesh K. Sitaraman

USENIX NSDI. Boston, March 28, 2017.

disco distributed computer systems

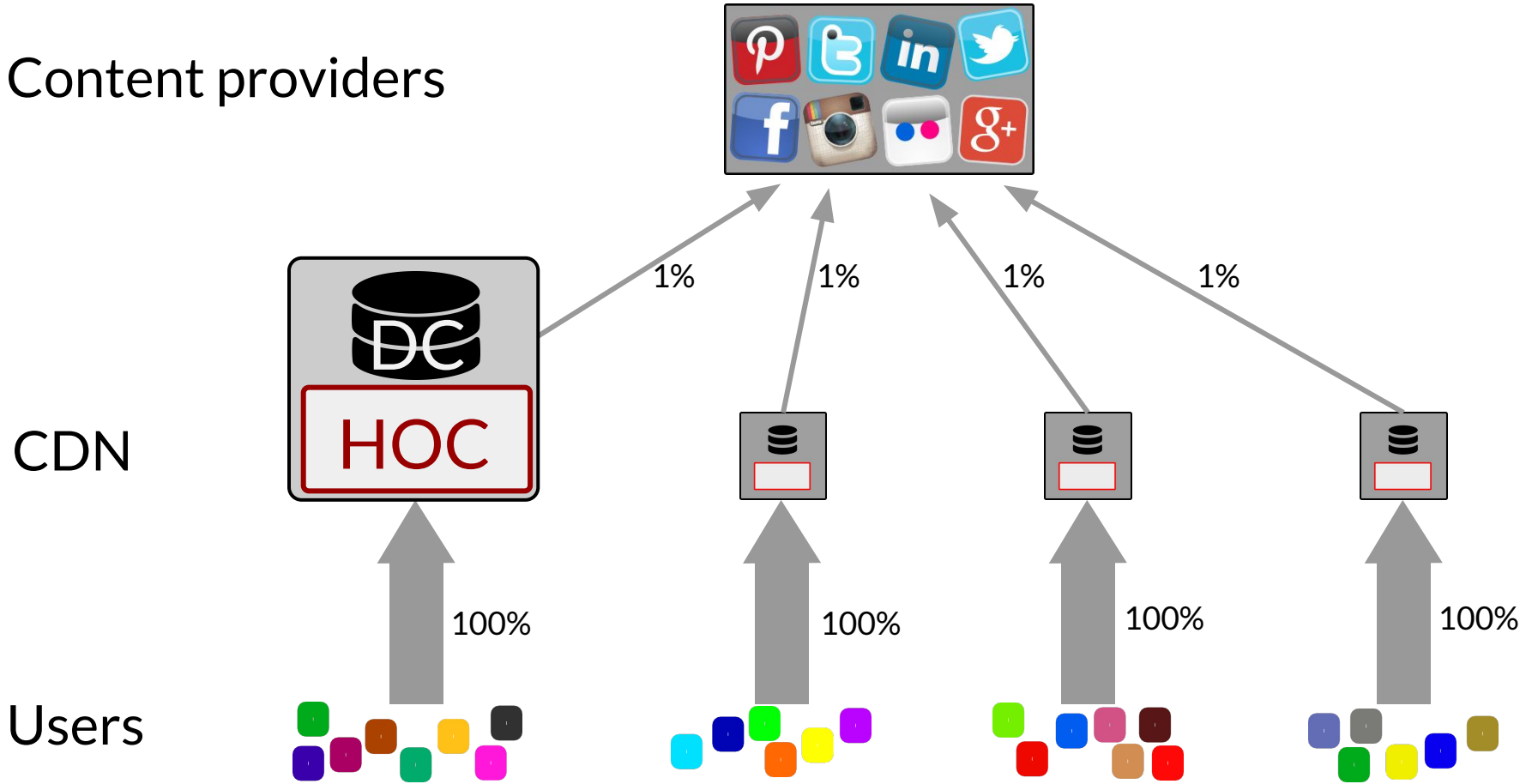
Technische Universität Kaiserslautern

Carnegie Mellon University

UMASS AMHERST

Akamai

CDN Caching Architecture



Optimizing CDN Caches

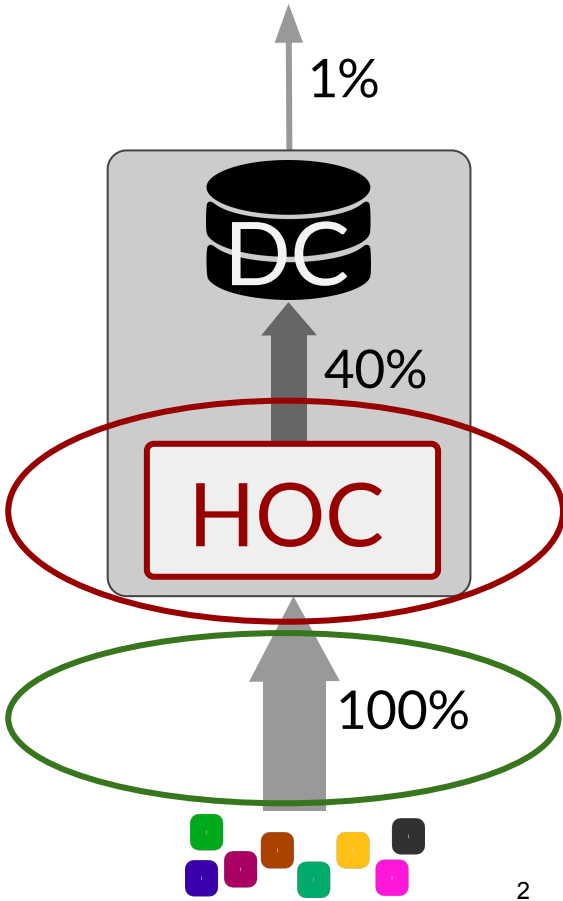
Two caching levels:

- Disk Cache (DC)
- Hot Object Cache (HOC)

HOC performance metric

$$\text{object hit ratio} = \text{OHR} = \frac{\text{\# reqs served by HOC}}{\text{total \# reqs}}$$

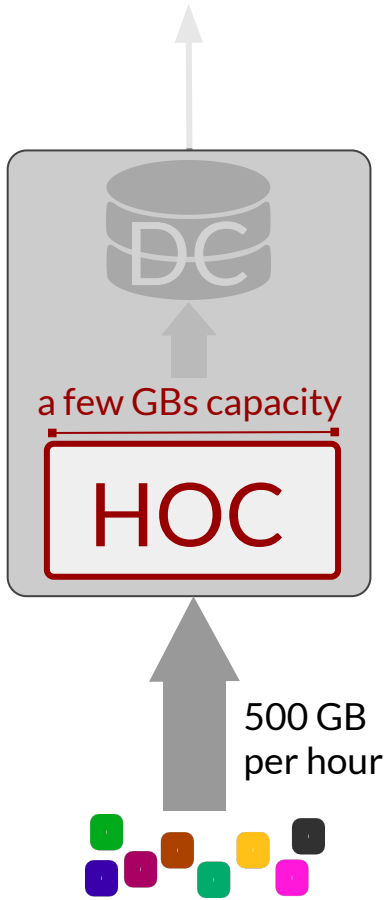
Goal: maximize OHR



Prior Approaches to Cache Management

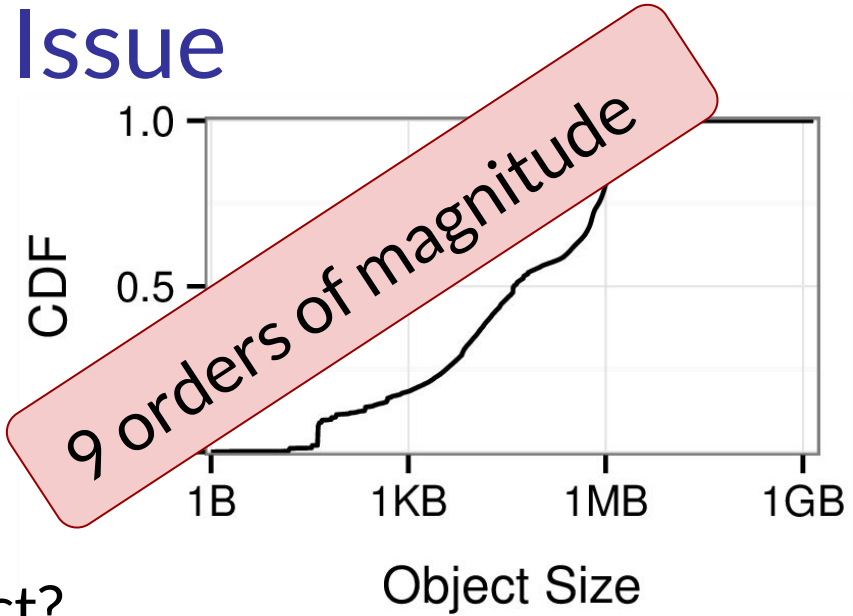
Frequent **decisions** required

	What to admit	What to evict
Today in practice e.g., Nginx, Varnish	everything	LRU
2000s in academia e.g., Modha, Zhang, Kumar	everything	mixtures of LRU/LFU
2010s in academia e.g., Kaminsky, Lim, Andersen	everything	concurrent LRU



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 it well)
- Academia has not been helpful
  (almost all theoretical work assumes equal-sized objects)

What's Hard About Size-Aware Admission

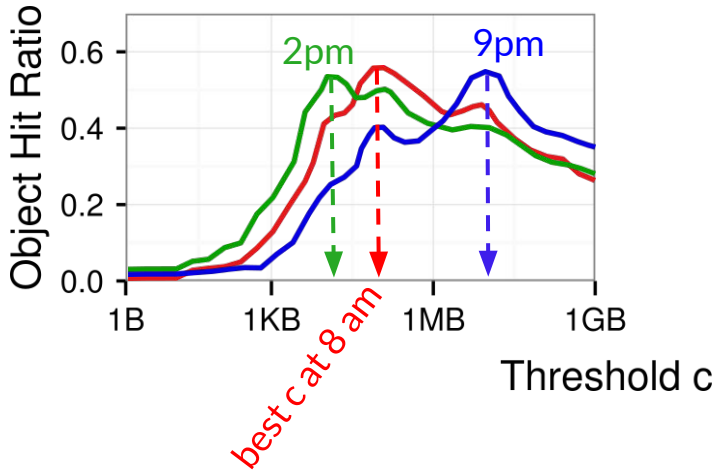
Fixed Size Threshold:

admit if *size* < Threshold c

Threshold c

How to pick c:

pick c to maximize OHR



The best threshold changes with traffic mix

Can we avoid picking a threshold c

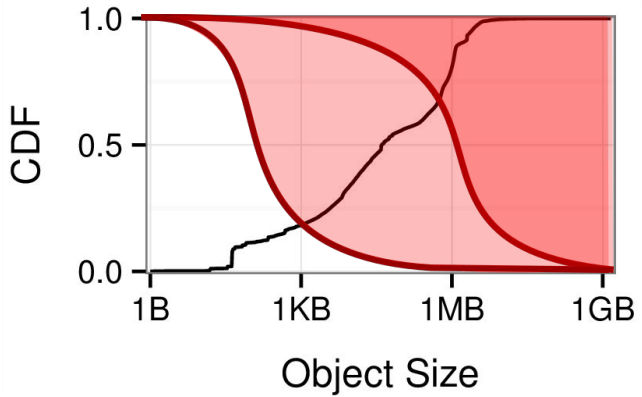
Probabilistic admission:

high admission probability

low admission probability

Unfortunately, many curves

example: exp(c) family



Which curve makes big difference

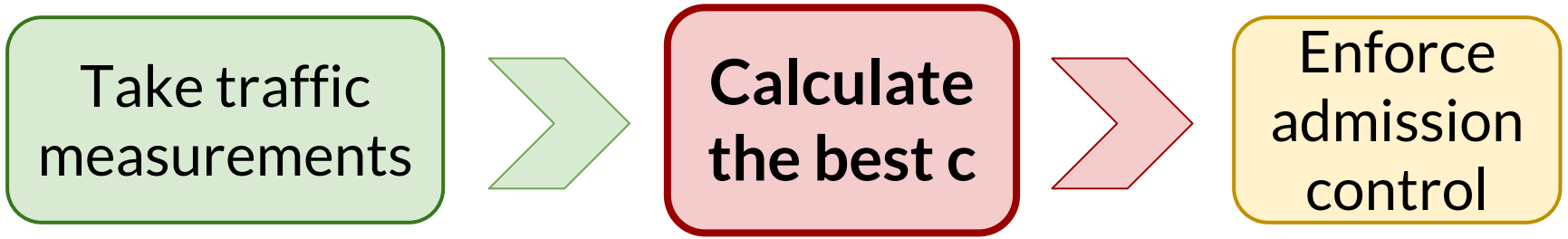
We need to **adapt** c

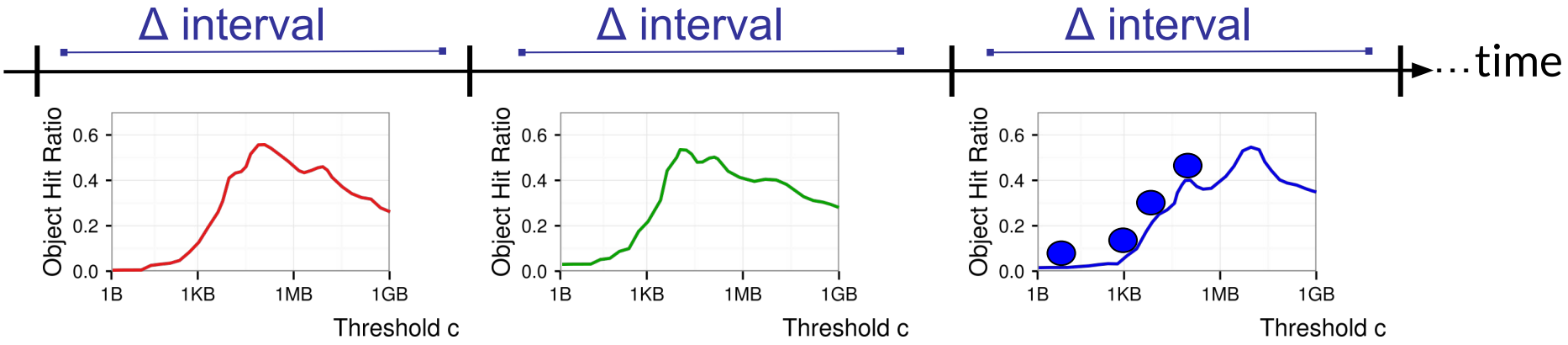
The AdaptSize Caching System

adapt with traffic

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

adapt with time





How to Find Best c Within Each Δ Interval

Traditional approach

Hill climbing

Local optima on OHR-vs-c curve

AdaptSize approach

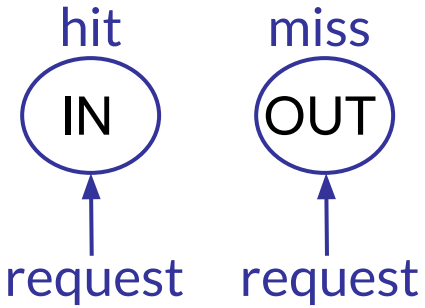
Markov model

Enables speedy global optimization

How AdaptSize Gets the OHR-vs-c curve

Markov chain

- track IN/OUT for each object



Algorithm

For every Δ 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?

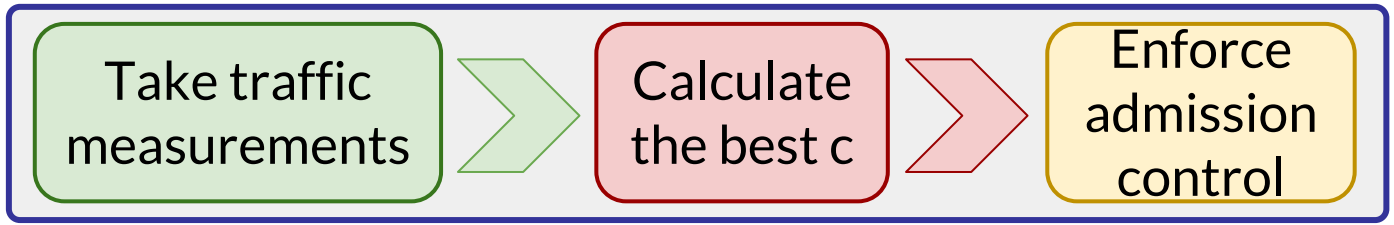
Too slow: exponential state space 需要枚举所有object的in、out的可能，是很大的action space

New technique: approximation with linear state space

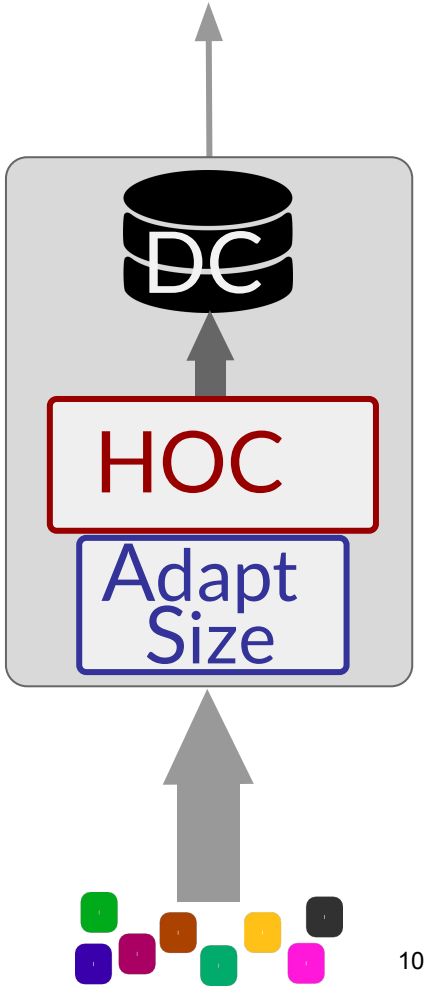
Implementing AdaptSize

Incorporated into Varnish

highly concurrent HOC system, 40+ Gbit/s



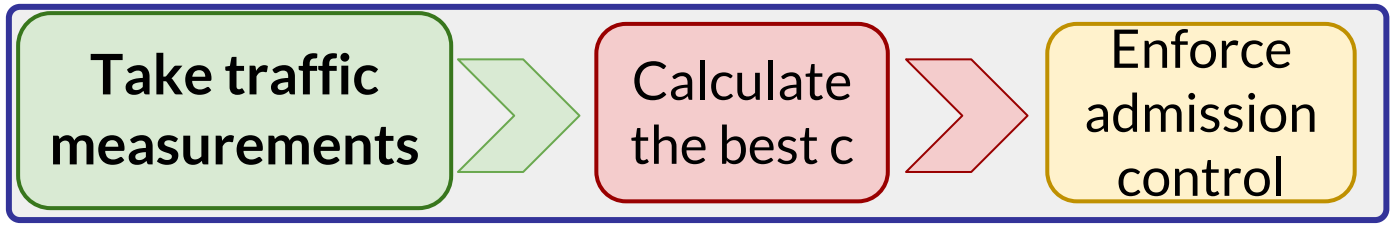
没有考虑latency

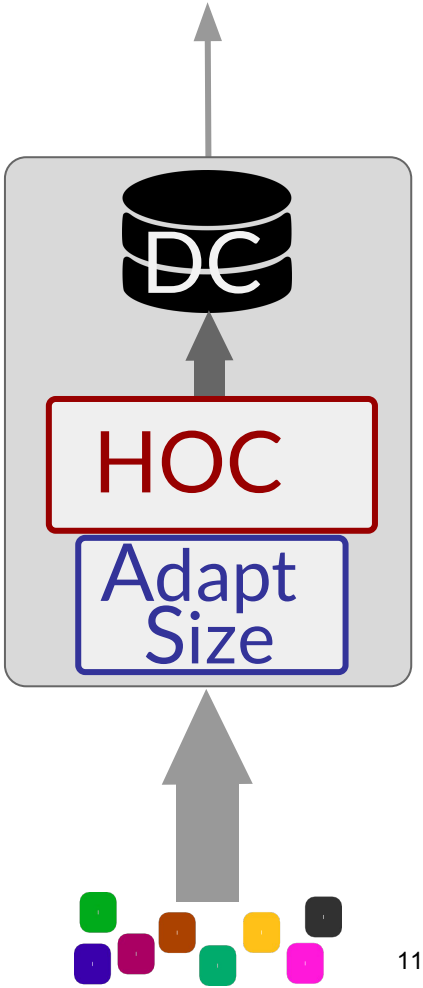


Implementing AdaptSize

Incorporated into Varnish

highly concurrent HOC system, 40+ Gbit/s





Challenges

1) Concurrent write conflicts

40% requests → 1% objects

2) Locks too slow [NSDI'13 & 14]

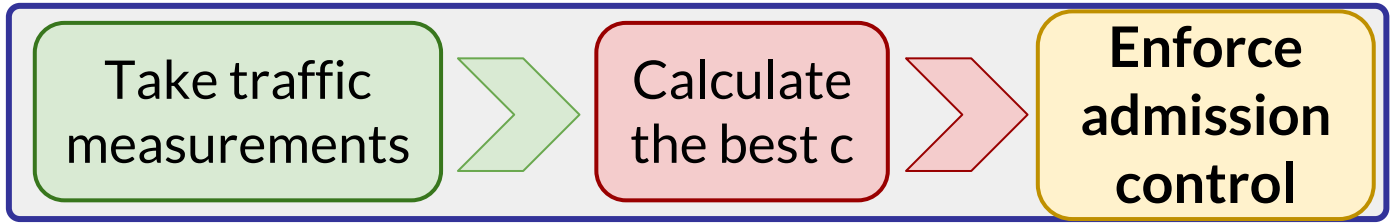
AdaptSize: producer/consumer + ring buffer

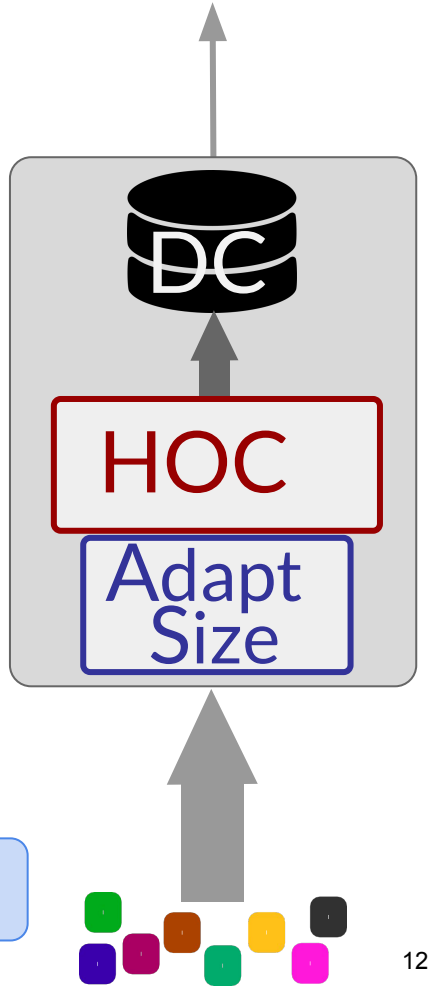
Lock-free implementation

Implementing AdaptSize

Incorporated into Varnish

highly concurrent HOC system, 40+ Gbit/s





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

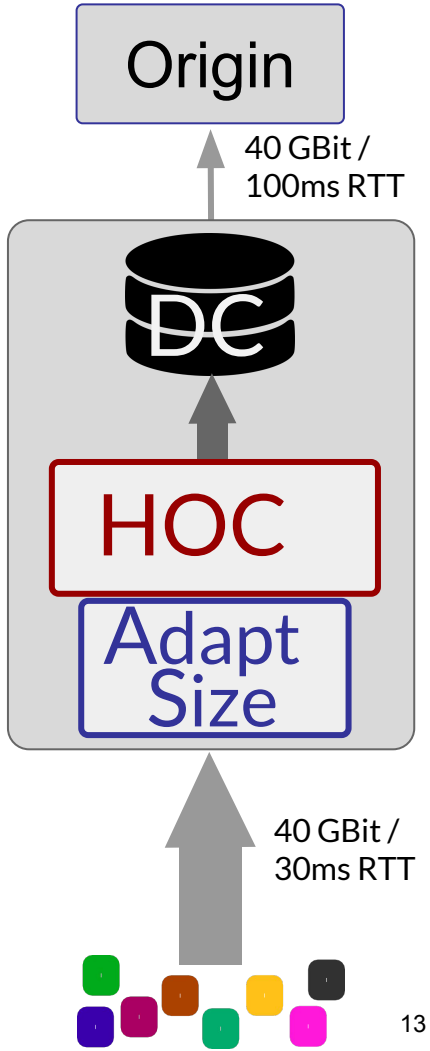
Origin: emulates 100s of web servers
55 million / 8.9 TB unique objects

DC: unmodified Varnish
4x 1TB/ 7200 Rpm

HOC systems:
1.2 GB
16 threads

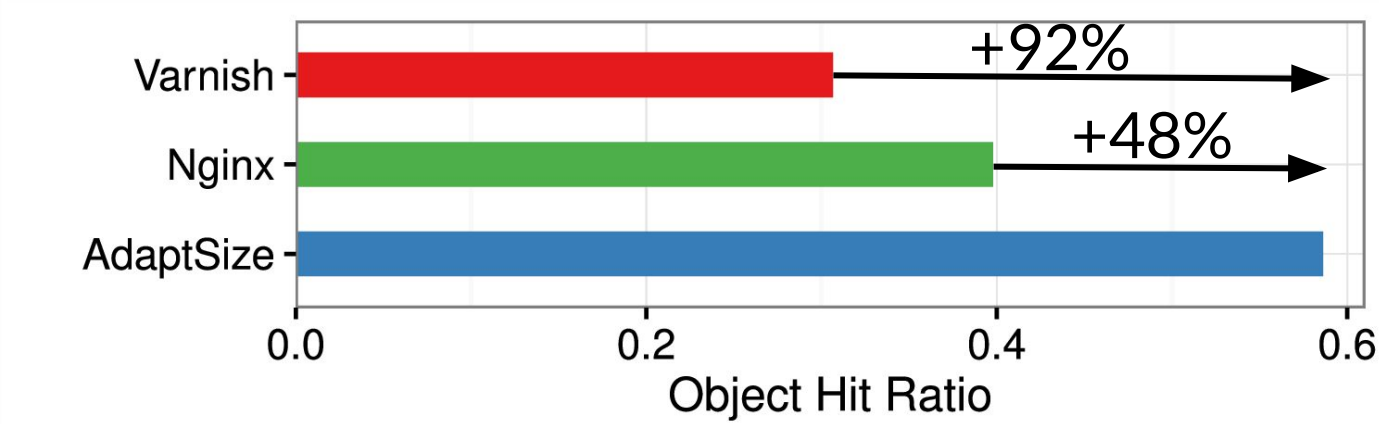
- unmodified Varnish
- NGINX cache
- AdaptSize

Clients: replay Akamai requests trace
440 million / 152 TB total requests

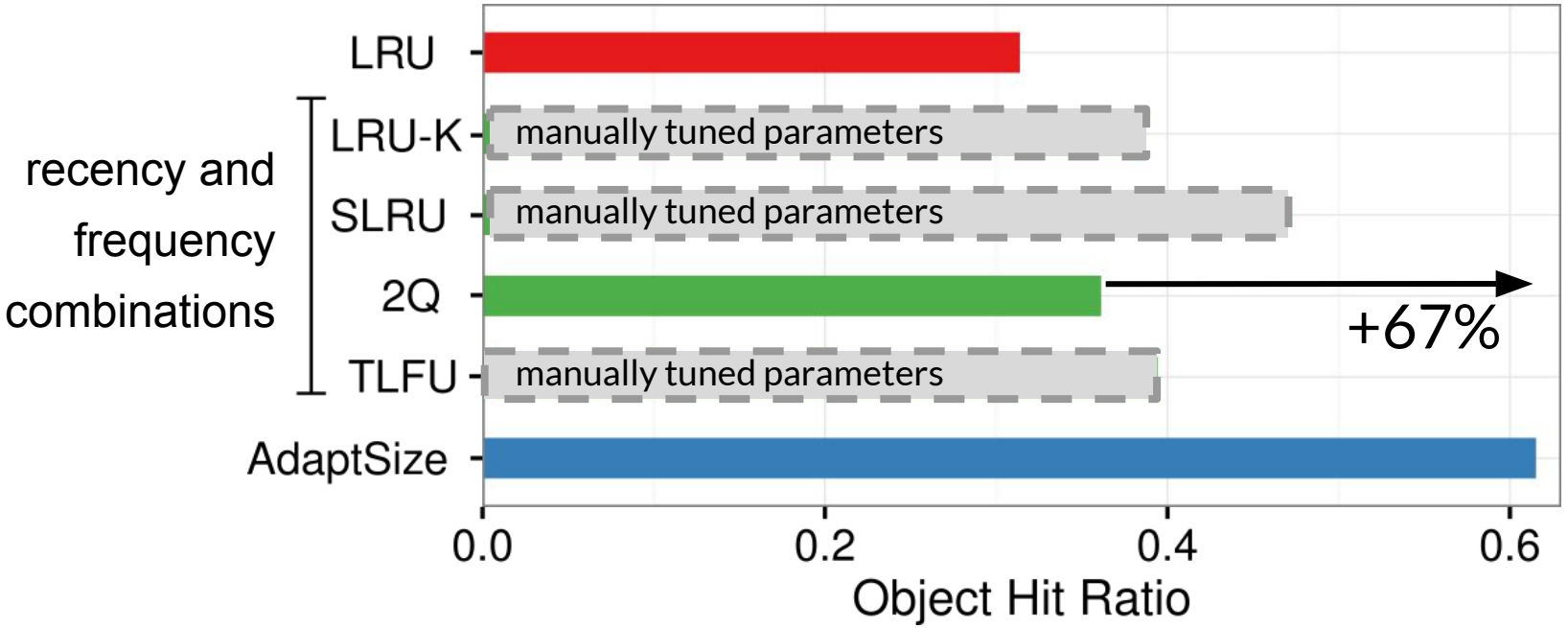


Comparison to Production Systems

	what to admit	what to evict
Varnish	everything	concurrent LRU
Nginx	frequency filter	LRU
AdaptSize	adaptive size-aware	concurrent LRU



Comparison to Research-Based Systems

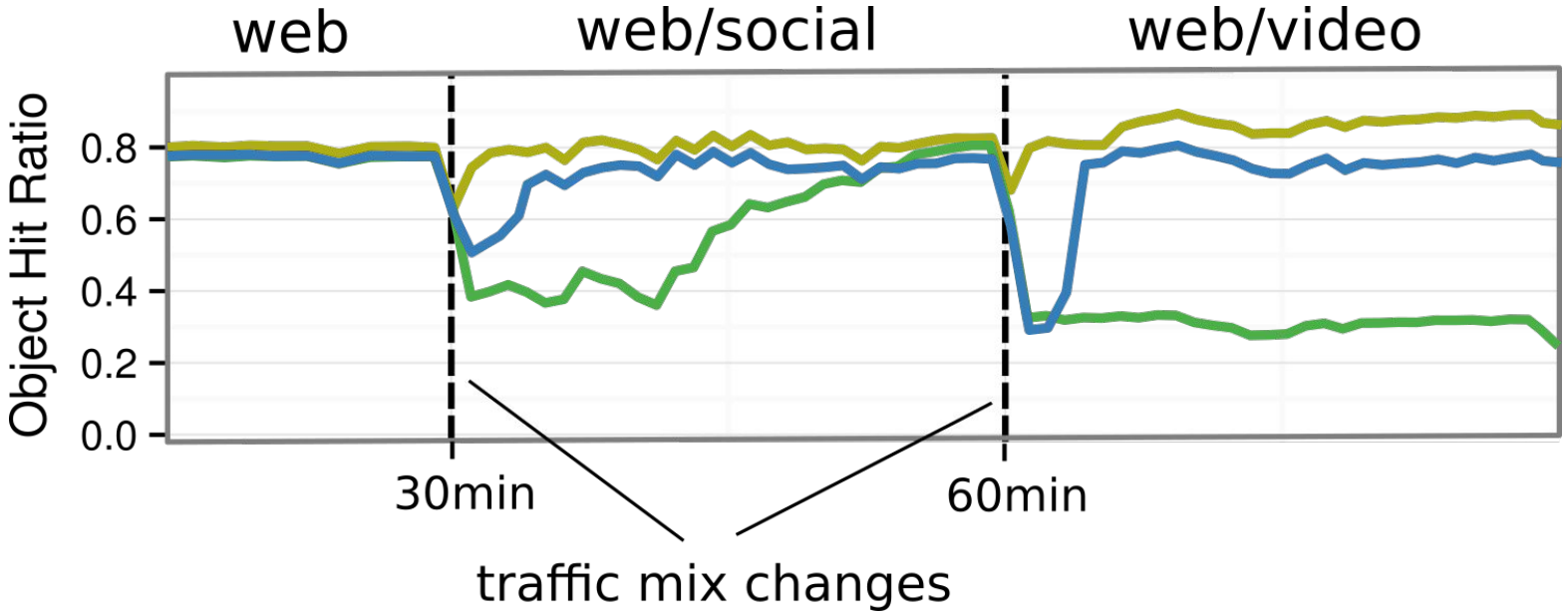


Robustness of AdaptSize

Size-Aware OPT: offline parameter tuning

AdaptSize: our Markovian tuning model

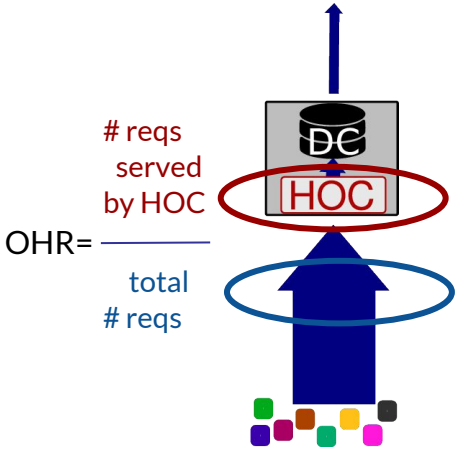
HillClimb: local-search using shadow queues



Conclusion

Goal: maximize OHR of the Hot Object Cache

Approach: size-based admission control



Conclusion

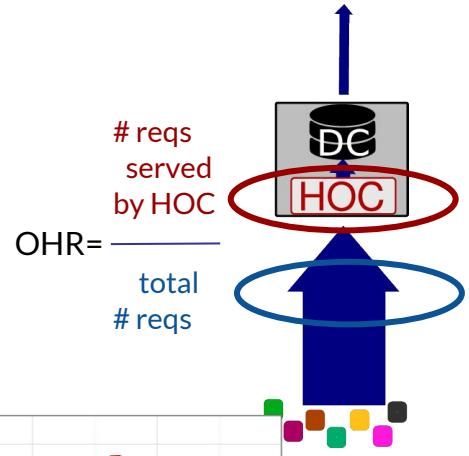
Goal: maximize OHR of the Hot Object Cache

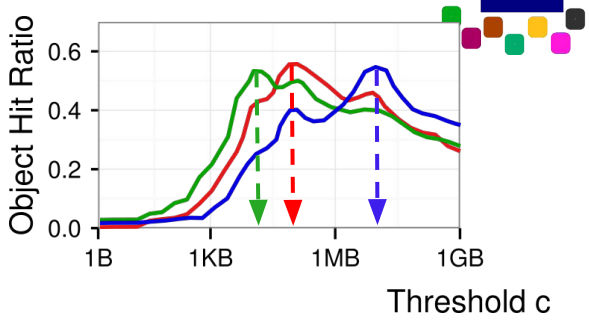
Approach: size-based admission control

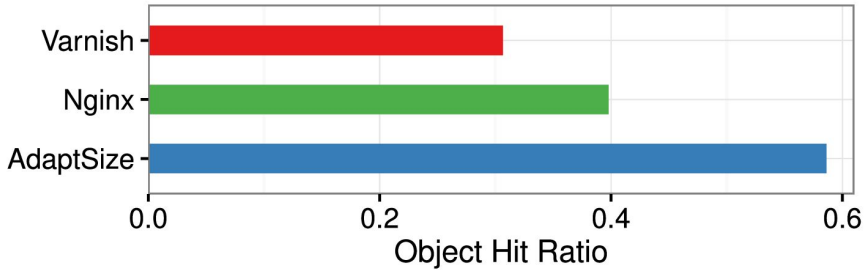
Key insight: need to adapt parameter c

AdaptSize: adapts c via a Markov chain

Result: 48-92% higher OHRs







Conclusion

Goal: maximize OHR of the Hot Object Cache

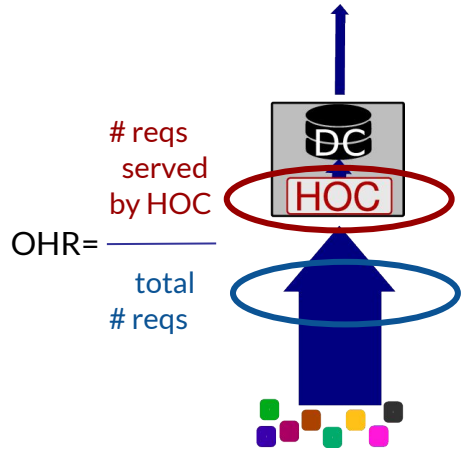
Approach: size-based admission control

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

BSD-2-Clause