RobinHood: Tail Latency-Aware Caching

Dynamically Reallocating from Cache-Rich to Cache-Poor

Daniel S. Berger *Benjamin Berg* *Timothy Zhu*

Carnegie Mellon University *Pennsylvania State University*

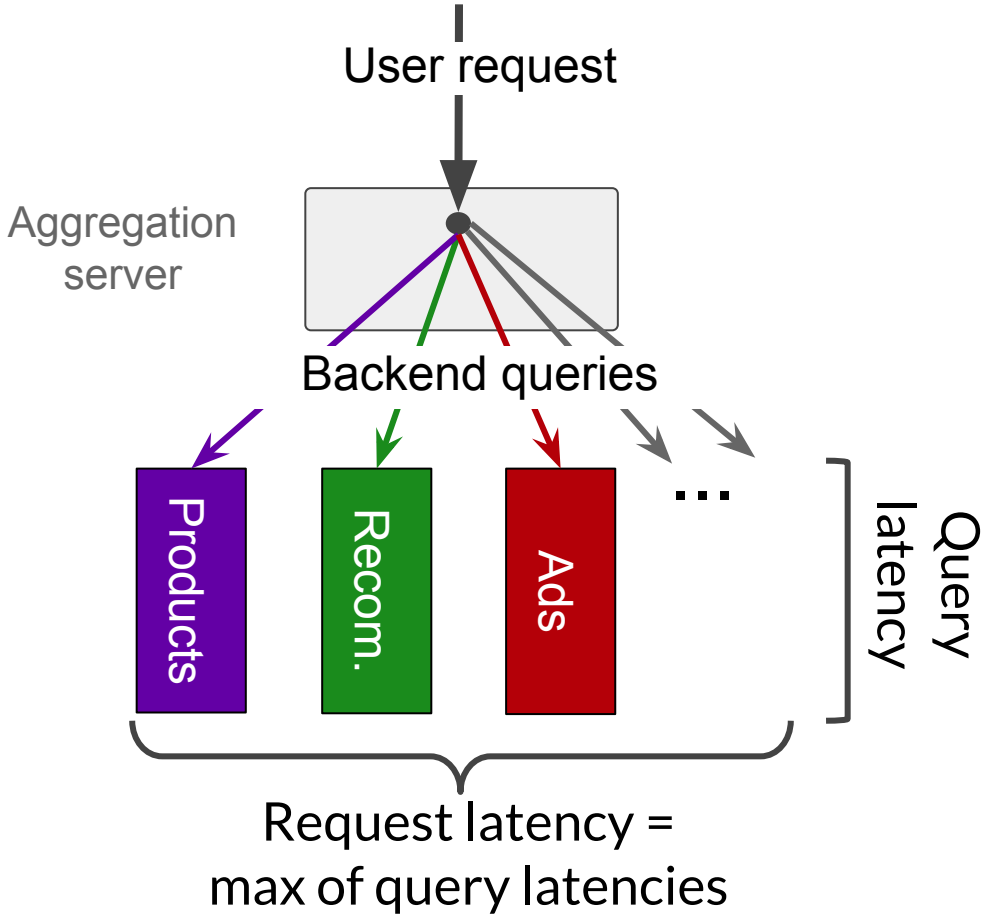
Siddhartha Sen *Mor Harchol-Balter*

Microsoft Research *Carnegie Mellon University*

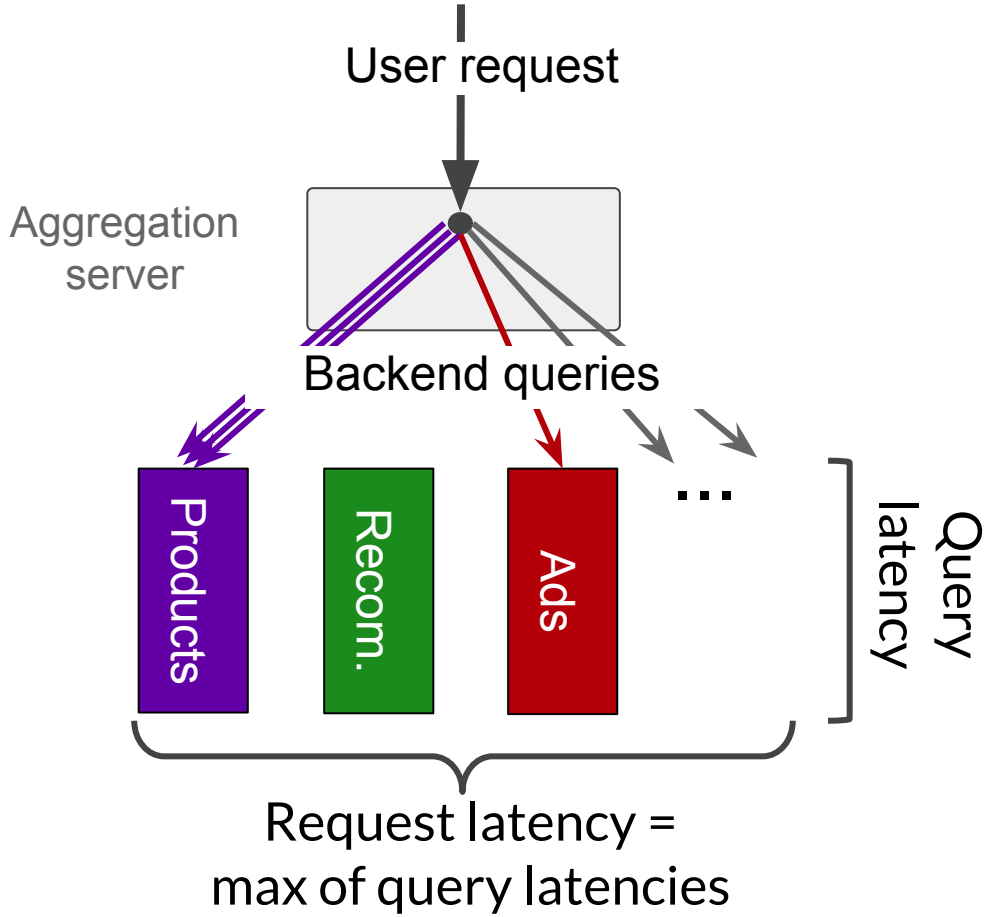
USENIX OSDI, 10/8/18.

Carnegie Mellon University

Typical Web Architecture

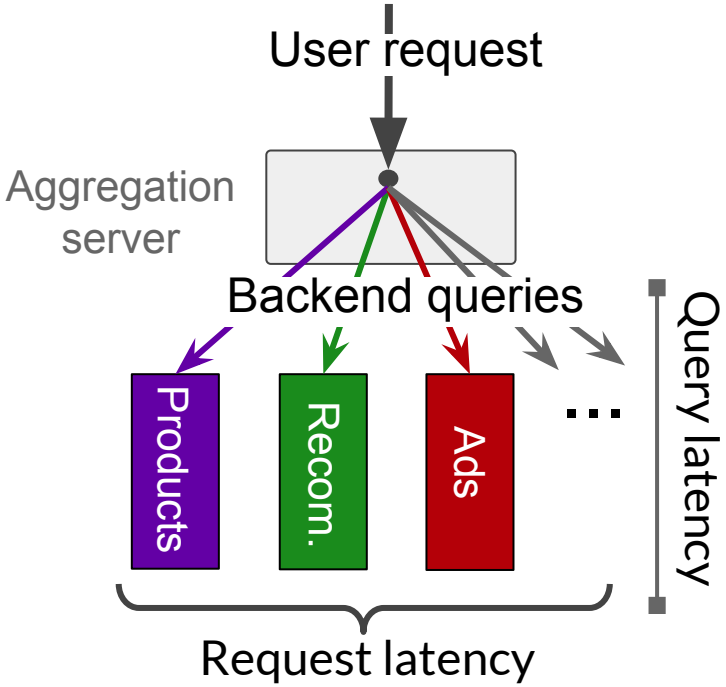


Typical Web Architecture

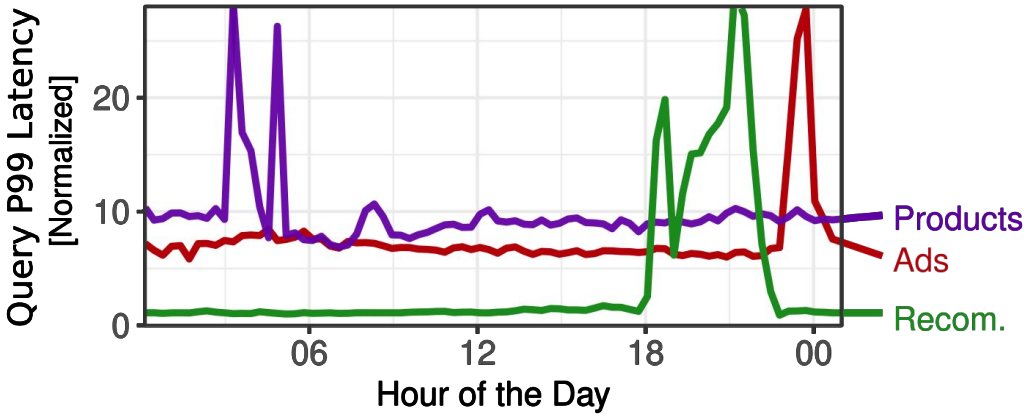


Goal: minimize 99-th percentile (P99) **request** latency

What Causes High P99 Request Latency?



Observations at xbox.com (3/2018):

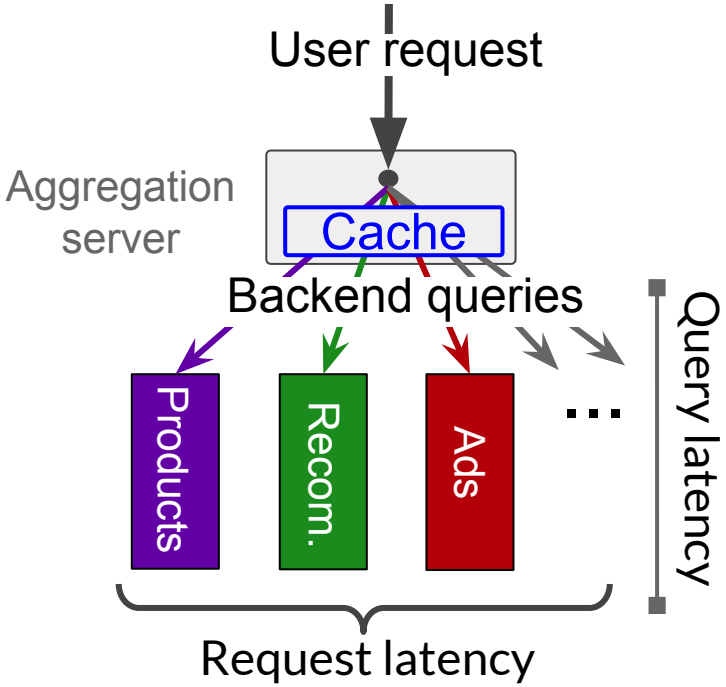


~~Better load balancing?~~

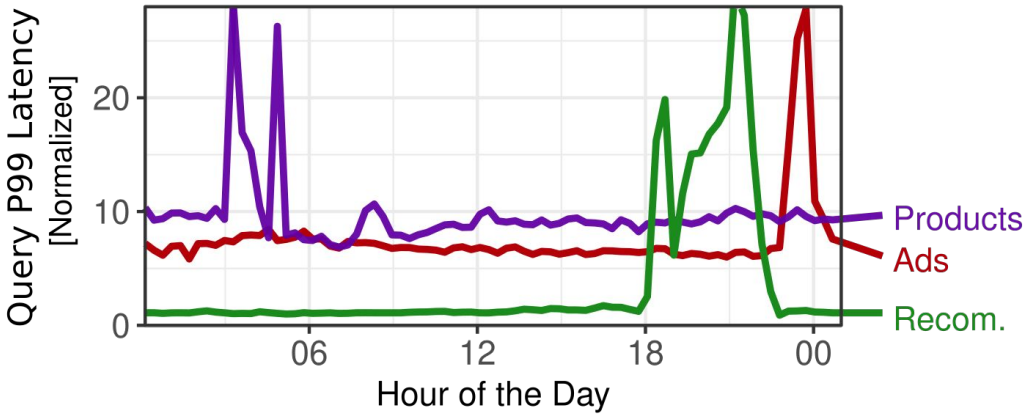
Partially implemented

Elastically scale backends?

What Else Can We Do?



Observations at xbox.com (3/2018):



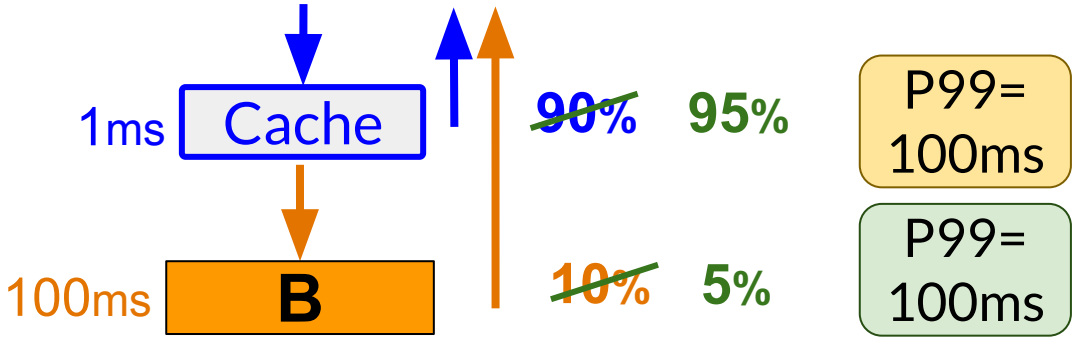
Aggregation Cache:

Currently shared among queries to all backends

Can we use the aggregation cache to reduce the P99 request latency?

Can We Use Caching to Reduce the P99?

Belief: No



BY JEFFREY DEAN AND LUIZ ANDRÉ BARROSO

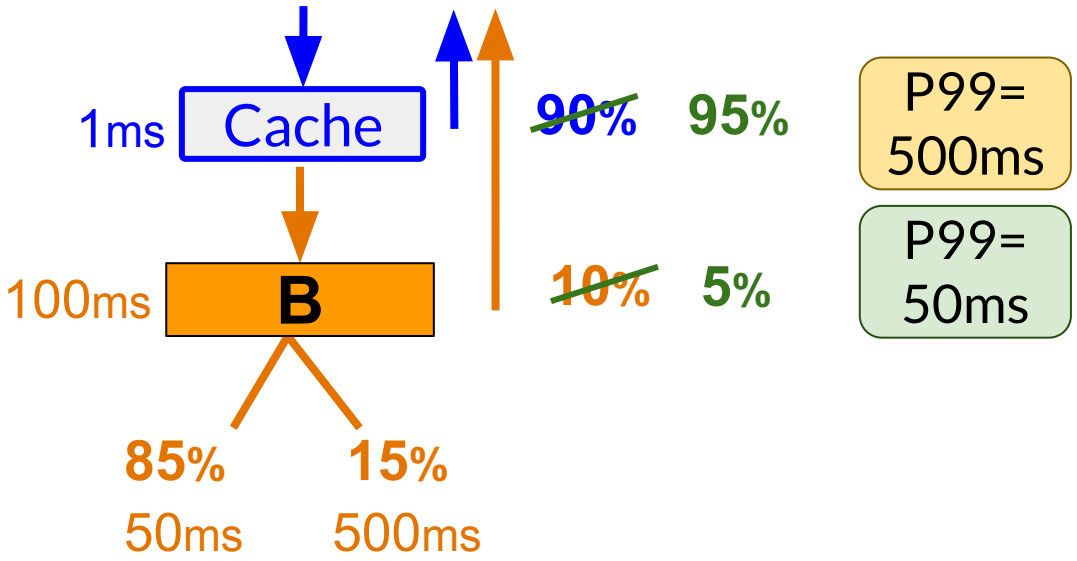
The Tail at Scale

"Caching layers do not directly address tail latency, aside from configurations where the entire working set can reside in a cache."

State-of-the-art caching systems focus on hit ratio, fairness — **not the P99**

Can We Use Caching to Reduce the P99?

Belief: No

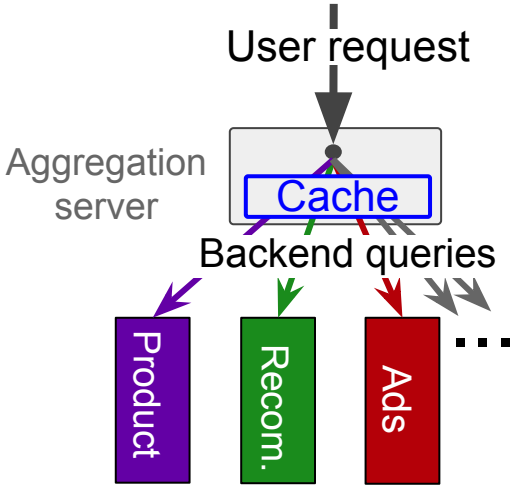


But: latency is not a constant

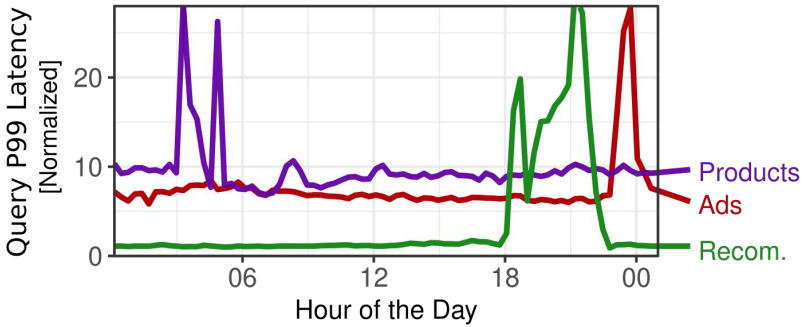
Caching can reduce P99 request latency!

Effectiveness in web architecture?

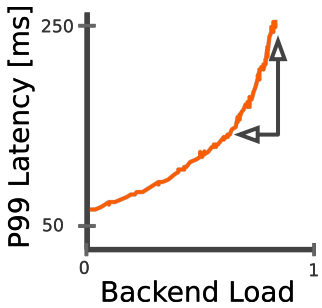
RobinHood: Key Idea



Observations for xbox.com (3/2018):

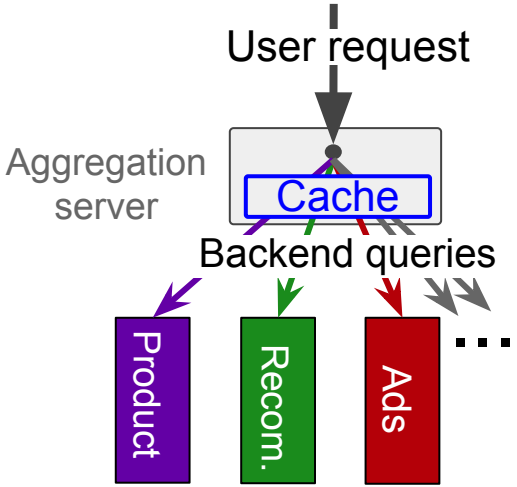


During load spike:

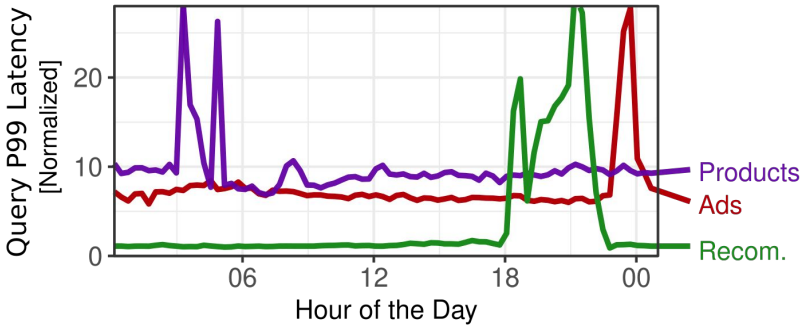


RobinHood: more cache ⇒ less load ⇒ much lower P99

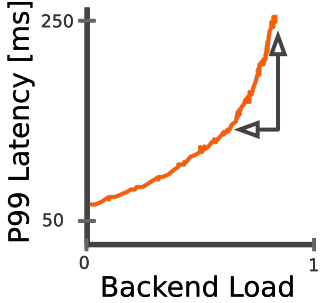
RobinHood: Key Idea



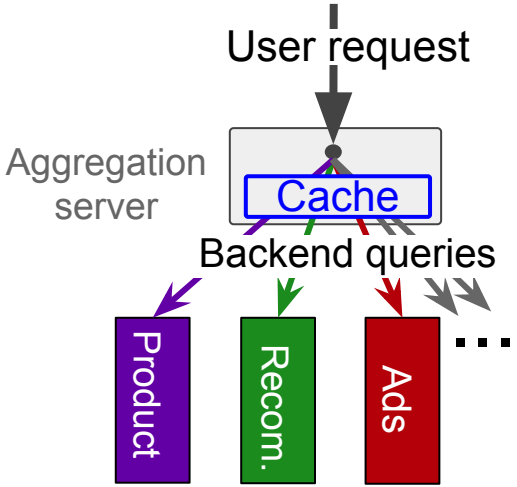
Observations for xbox.com (3/2018):



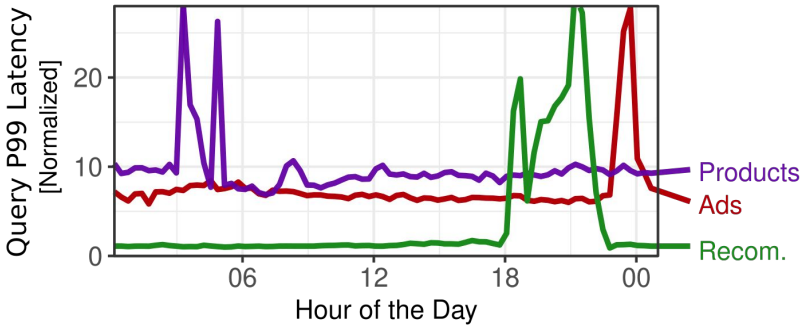
During load spike:



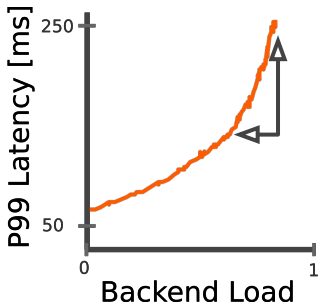
RobinHood: Key Idea



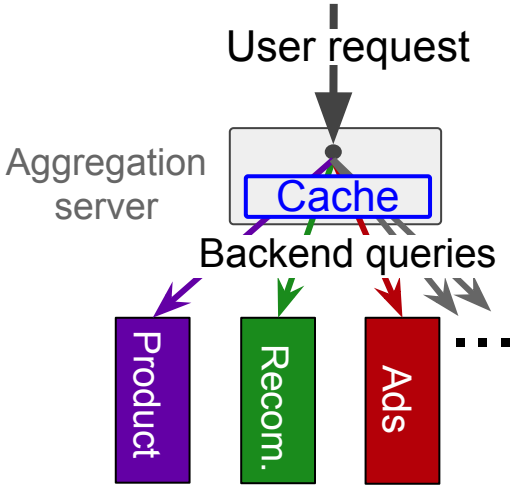
Observations for xbox.com (3/2018):



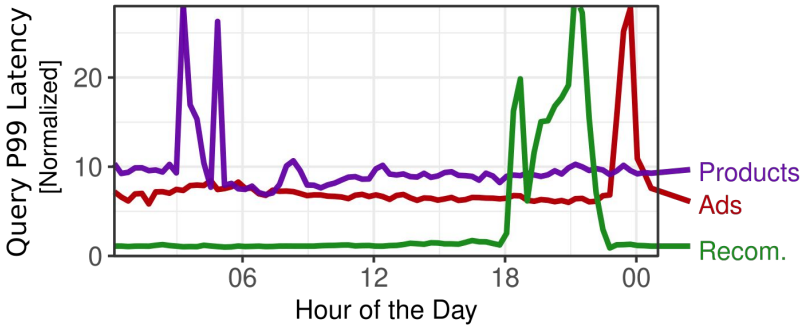
During load spike:



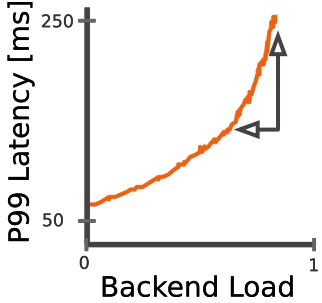
RobinHood: Key Idea

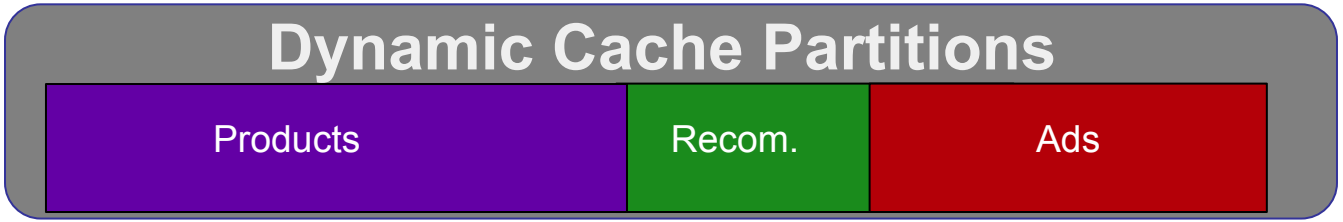


Observations for xbox.com (3/2018):

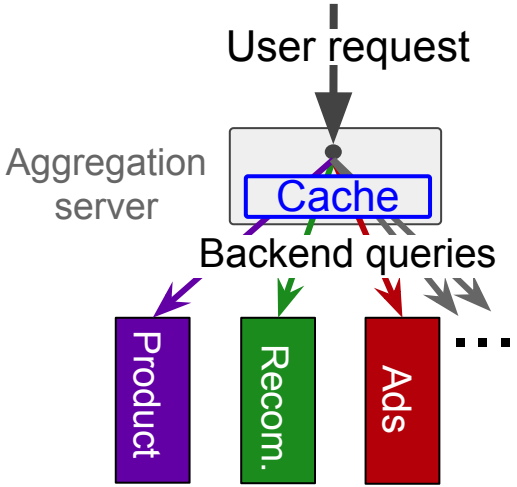


During load spike:

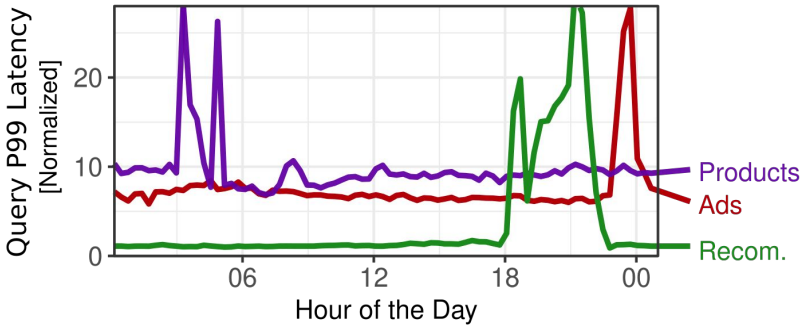




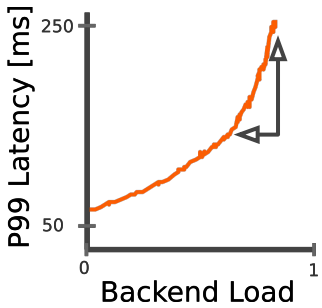
RobinHood: Key Idea

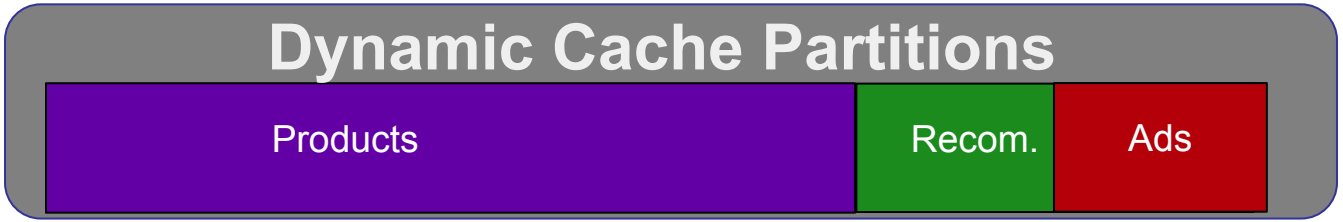


Observations for xbox.com (3/2018):

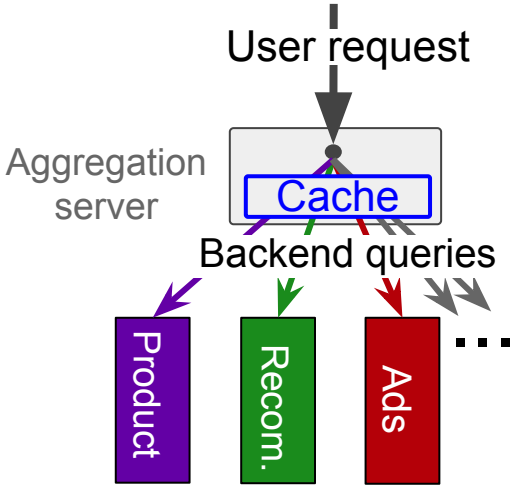


During load spike:

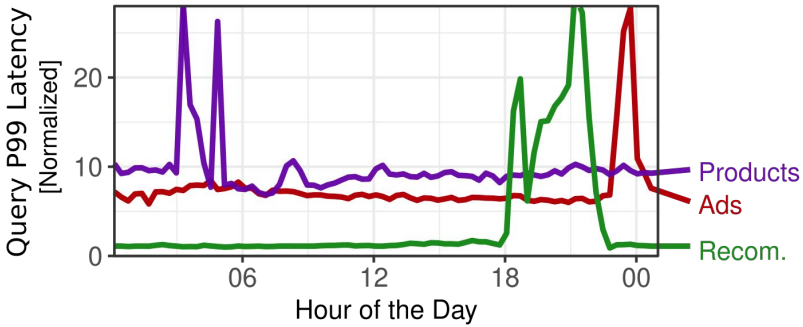




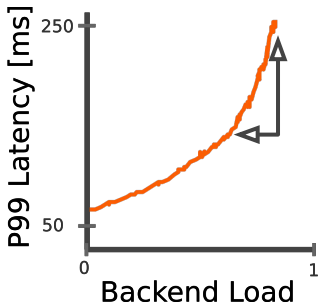
RobinHood: Key Idea

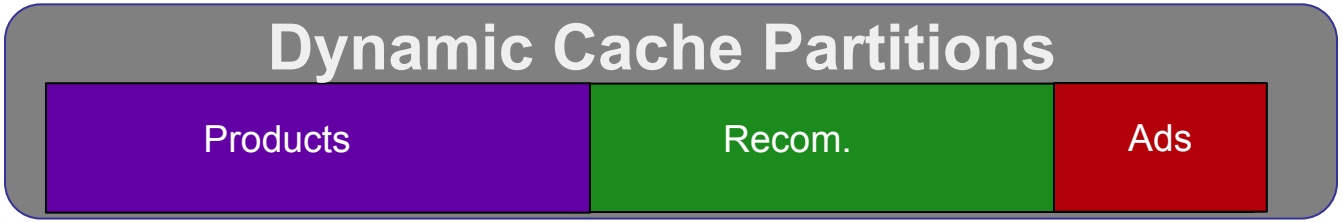


Observations for xbox.com (3/2018):

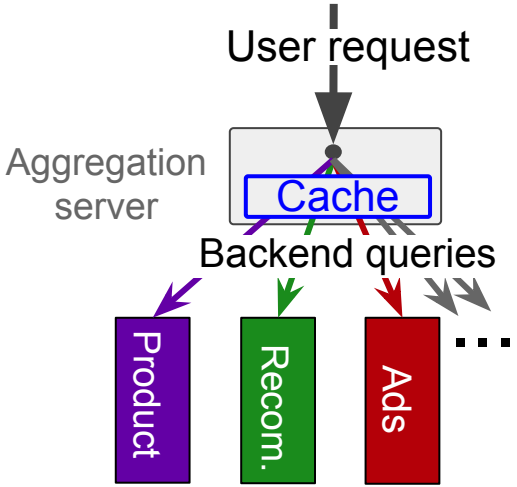


During load spike:

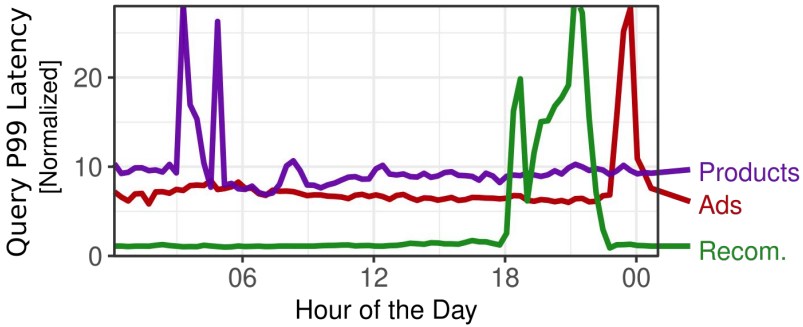




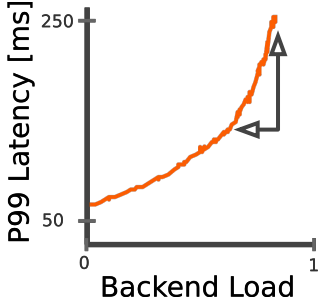
RobinHood: Key Idea



Observations for xbox.com (3/2018):



During load spike:



The RobinHood Caching System

Dynamically partition the aggregation cache

First caching system to minimize **request** P99

RobinHood Cache

Deployable on off-the-shelf software stack

Scalable in # backends, # aggregation servers

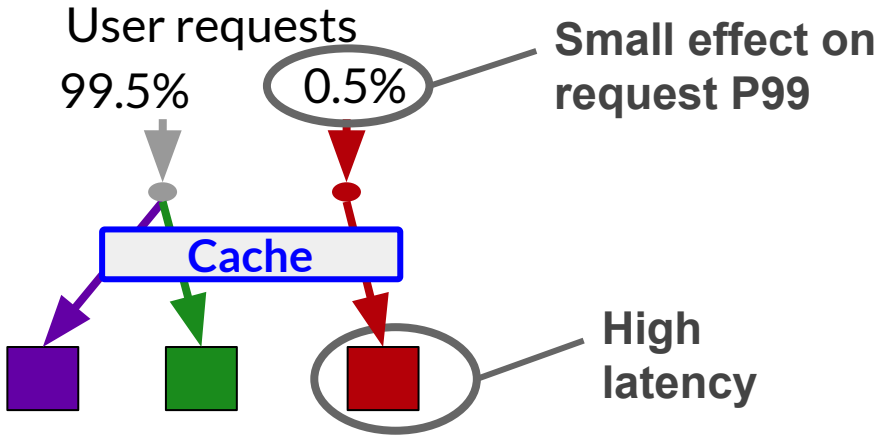
How to Repartition the Cache?

Every 5 seconds: RobinHood taxes everyone 1%

How to redistribute the tax?

First idea: give cache to high-latency backends

Recall: not all requests are the same



How to Repartition the Cache?

Every 5 seconds: RobinHood taxes everyone 1%

How to redistribute the tax?

RobinHood: find the cause of high request P99

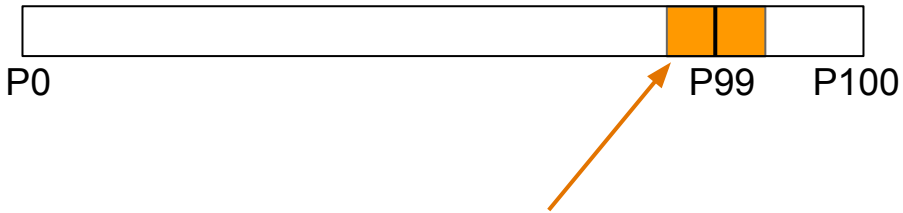
Who “blocked” this request?

How to Repartition the Cache?

Every 5 seconds: RobinHood taxes everyone 1%

How to redistribute the tax?

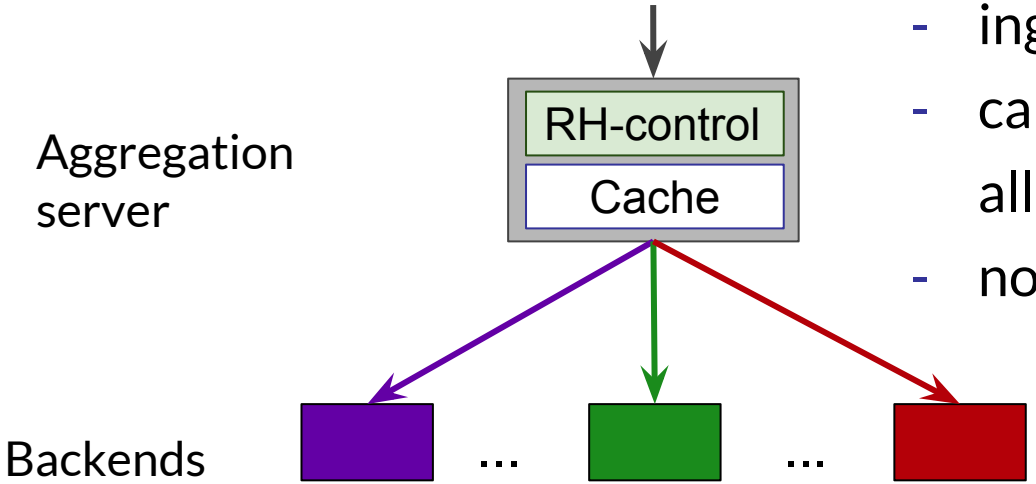
RobinHood: find the cause of high request P99



Who “blocked” these requests?

⇒ Track “request blocking count” (RBC) for each backend

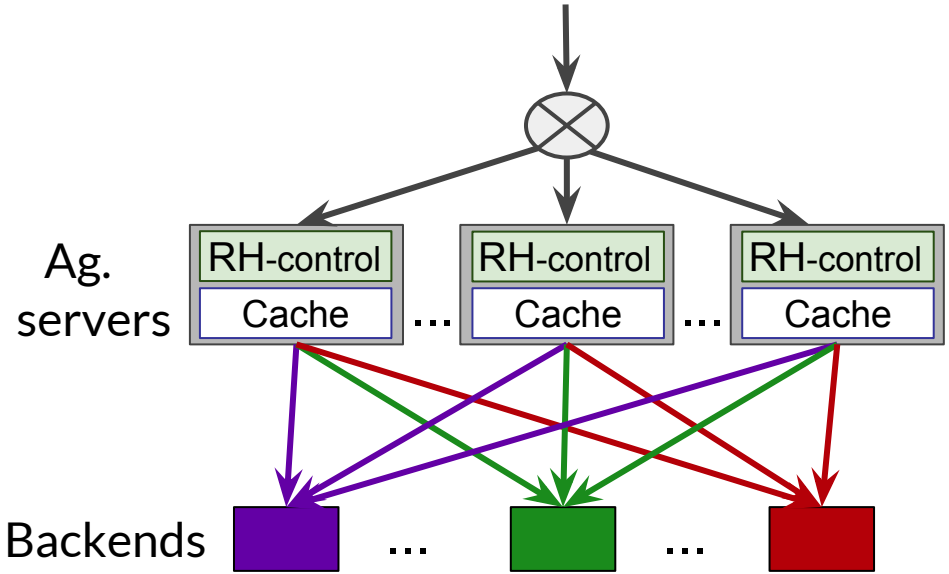
RobinHood Architecture



RobinHood Controller

- ingests RBC
- calculates / enforces cache allocation
- not on request path

RobinHood Architecture



In practice many Ag. servers

⇒ RH-control / Ag. server

Distributed RobinHood:

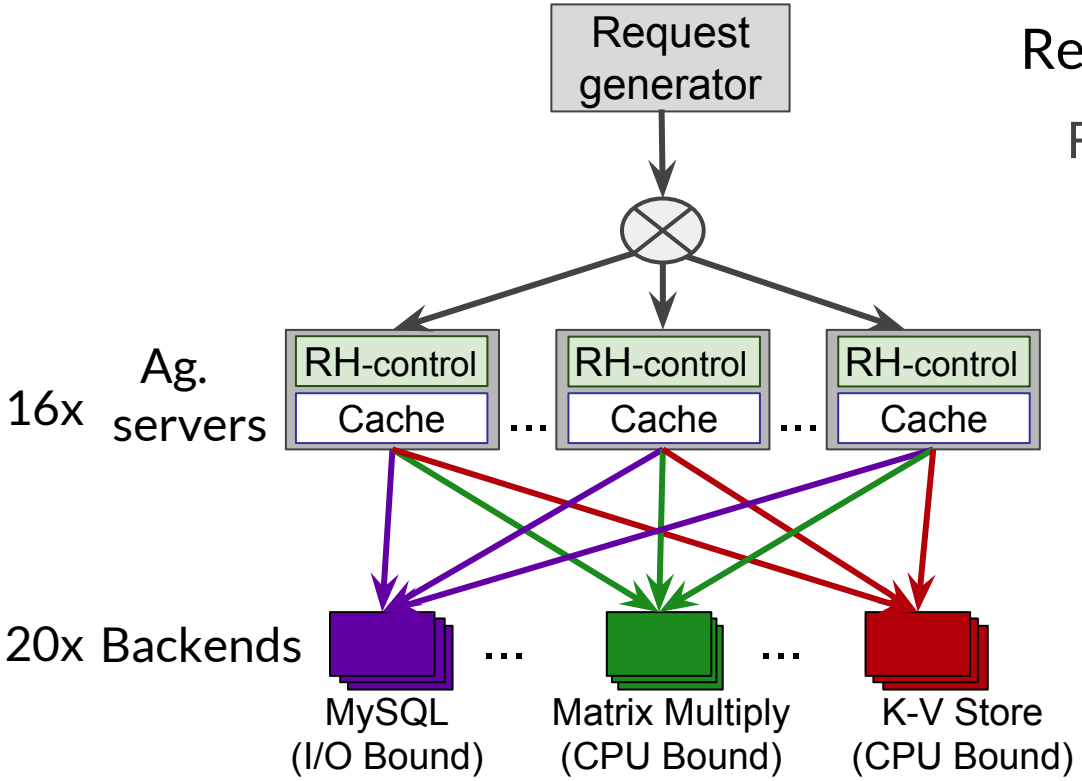
- Local measurements

Poor measurements

Challenge: insufficient # tail data points

- Local decisions

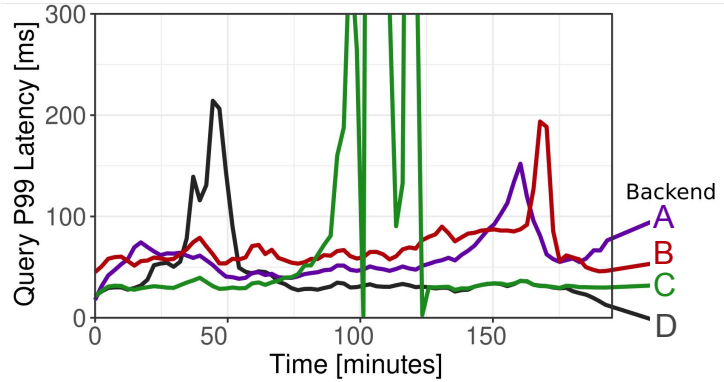
Experimental Setup



Replay production trace

For 4 hours, 200k queries/second

32 GB cache size



⇒ Emulate query latency spikes

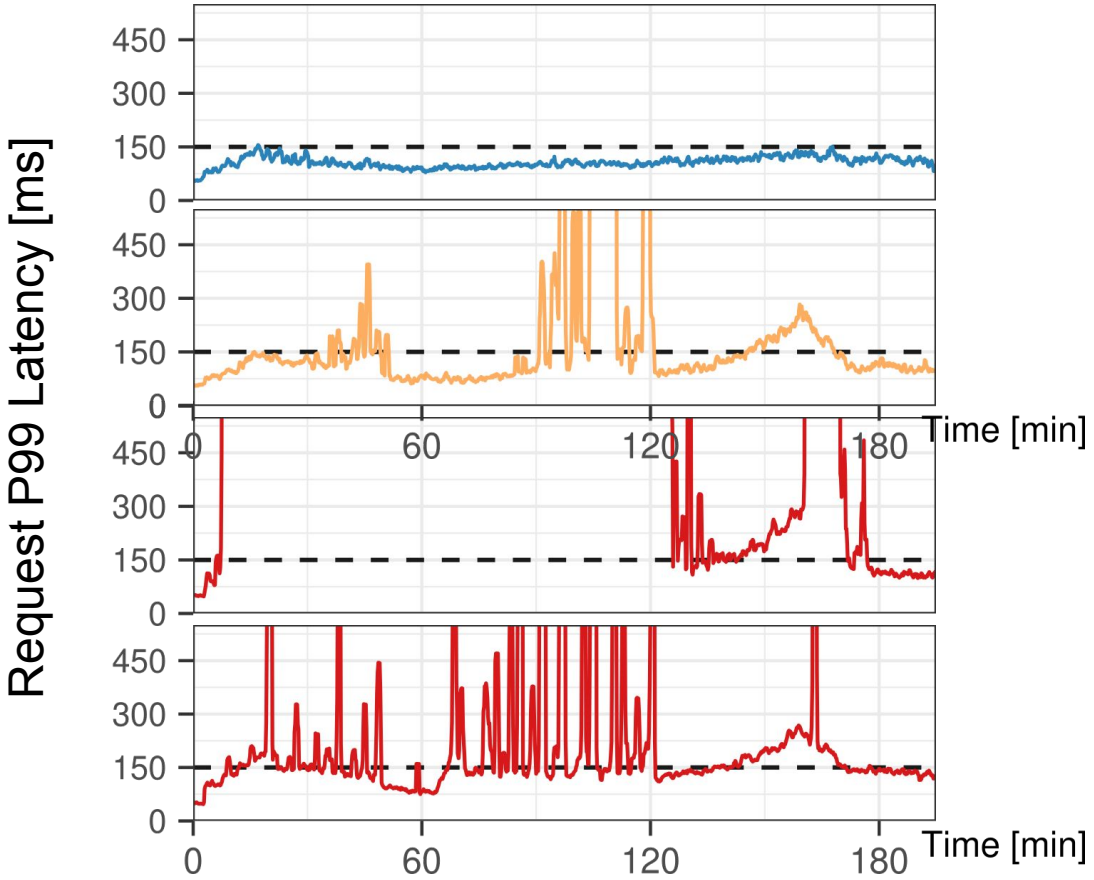
Evaluation Results: P99 Request Latency

RobinHood
[our proposal]

Original MS System
[OneRF]

Maximize Overall Hit Ratio
[Cliffhanger, NSDI'16]

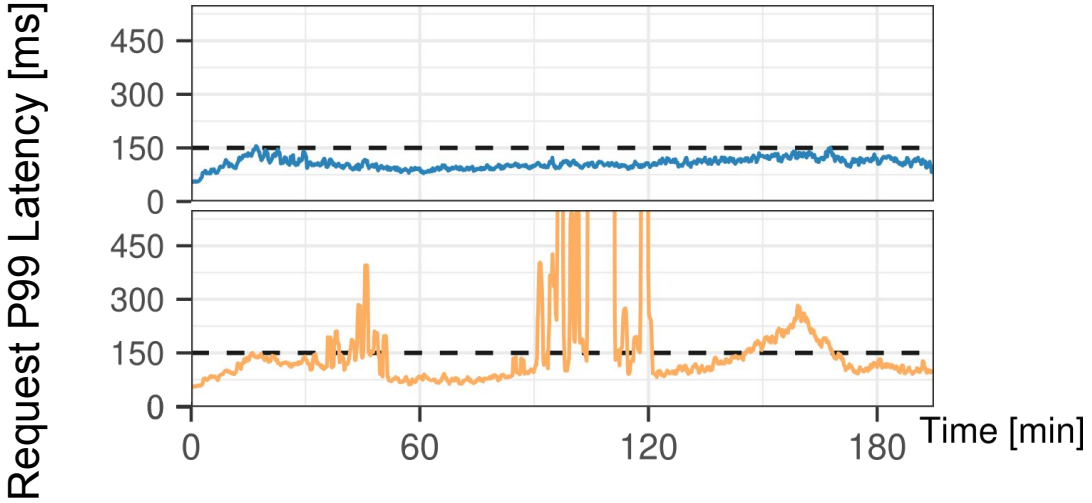
Balance Query Latencies
[Hyberbolic, ATC'17]



What Makes RobinHood so Effective?

RobinHood
[our proposal]

Original MS System
[OneRF]



The RobinHood tradeoff:

- Sacrifice performance of some backends → up to 2.5x higher latency
- Reduce latency of bottleneck backends → typically 4x lower latency

⇒ Reduced **request** latency

Conclusions

Poster #31

Is it possible to use caches to improve the request P99?

Yes! Huge reduction in P99 spikes and SLO violations.
⇒ Use cache as load balancers: “RBC load metric”.

Feasibility in production systems?

Yes! Built using off-the-shelf software stack. Works orthogonally to existing load balancing and data/quality tradeoffs.

Is this the optimal solution? End of this project?

No! There’s a lot to do. Need to consider the effect of other request structures.