AEROSPIKE NEXTGEN NOW SUMMIT '20

Rebuilding a Real-time Datastore: a story of design, deployment and performance

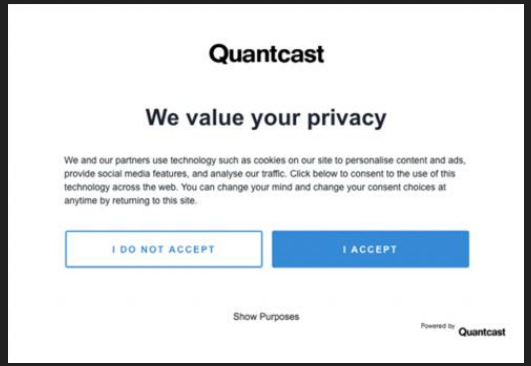
Quantcast

Kristi Tsukida
STAFF SOFTWARE ENGINEER
QUANTCAST

Paul Revere
SENIOR SOFTWARE ENGINEER
QUANTCAST

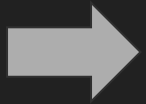
Quantcast overview

- We're an online advertising platform attempting to simplify advertising on the open internet
- We're also an online measurement platform for publishers
- We also make products to help publishers comply with privacy regulation



Real Time Bidding (RTB)

TV, Radio, Newspaper Ads
Static ad targeted at thousands/millions



Customized ad targeted at individual user

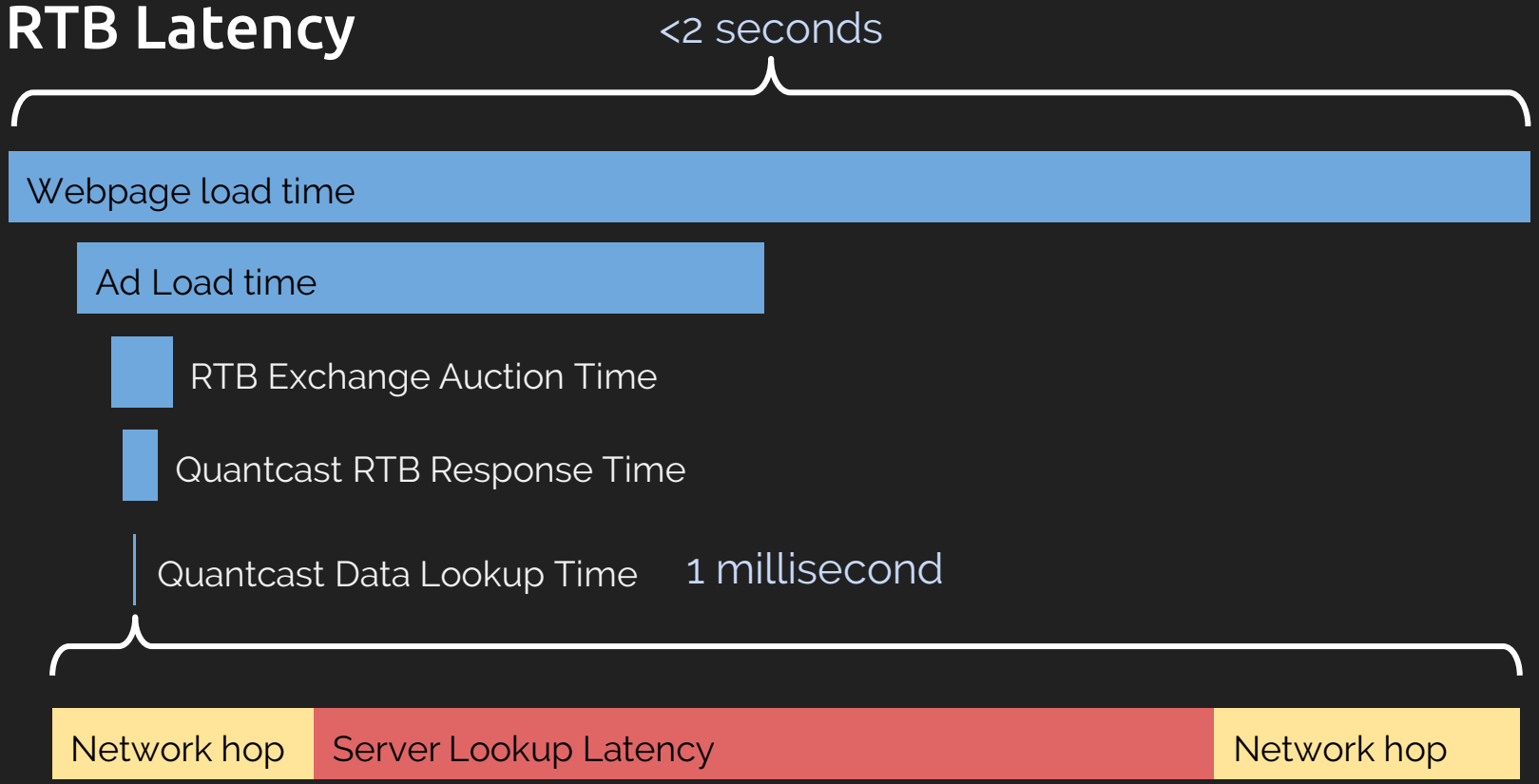
Real Time Bidding (RTB)

With great targeting power comes great engineering problems

- Scale
  - Huge request volume
  - Store data on as many internet users as possible
  - Make updates to that data in real time
- Latency
  - Retrieve and evaluate that data as a webpage is loading

Customized ad targeted at individual user

RTB Latency


<2 seconds
Webpage load time
Ad Load time
RTB Exchange Auction Time
Quantcast RTB Response Time
Quantcast Data Lookup Time
1 millisecond
Network hop
Server Lookup Latency
Network hop

Our beloved villain

A legacy custom-built key-value store system

Impressively kept the business running for the last 8 years

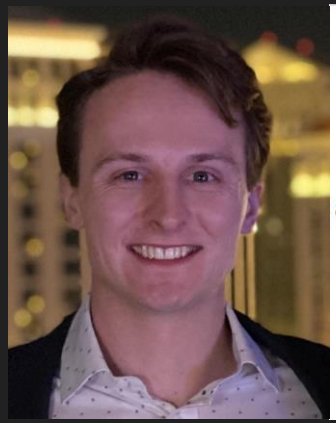
Great Destroyer of New Product Ideas

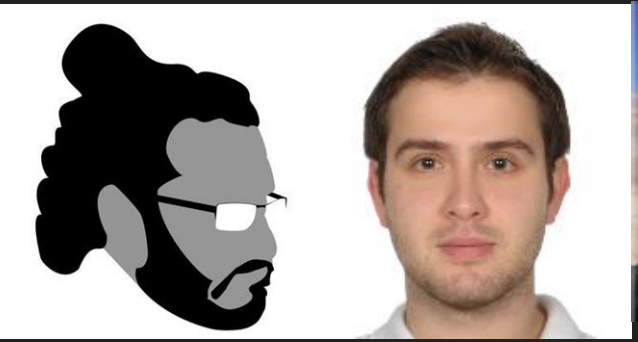
Invokes Fear Of Change

The Heroes

This story has a lot of heroes

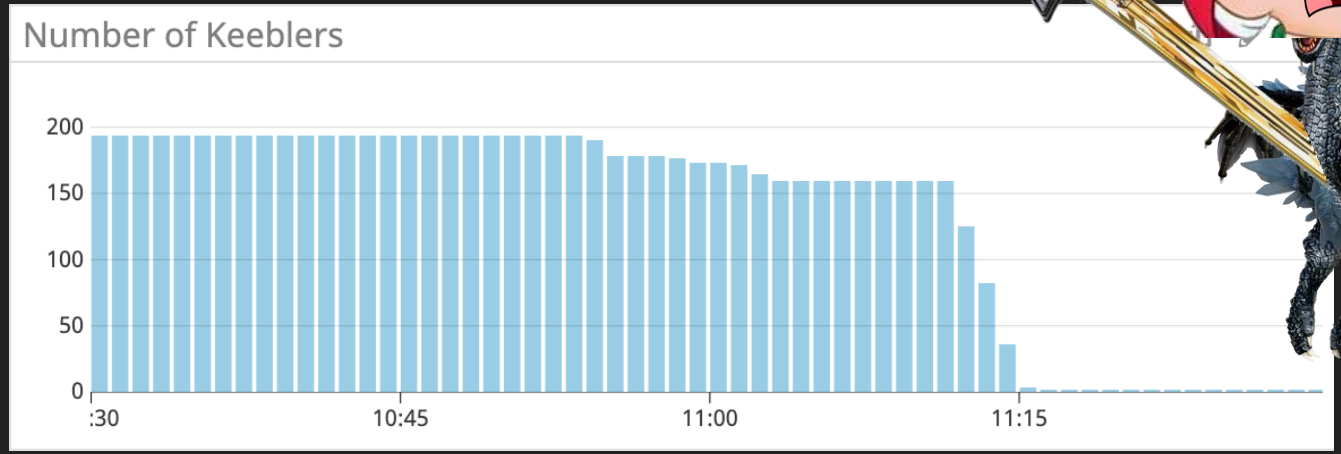
Couldn't have been done without teamwork!

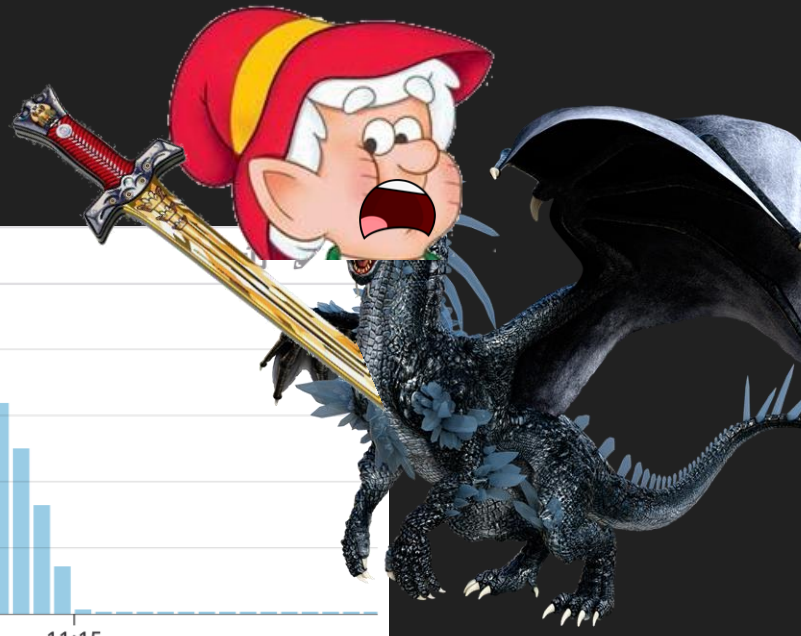




Spoiler alert: We win

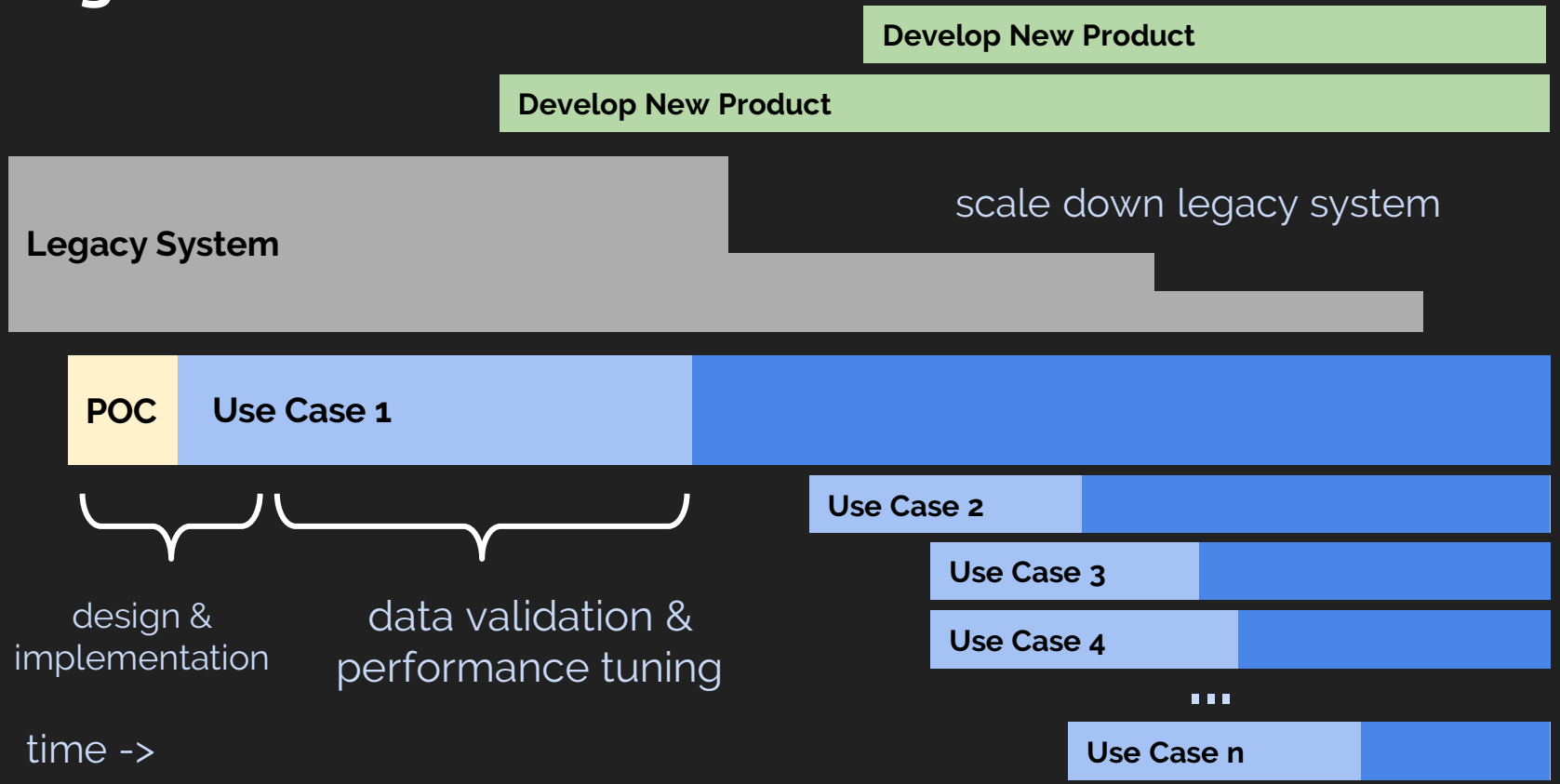
On March 5th we shut off Keebler





Spoiler alert: We win

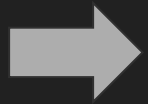
Migration Timeline



Migration: Featurestore use case

Proof of Concept

- synthetic benchmark
- realtime & batched data loading
- no data validation



Productionization

- production integration
  - performance evaluation
- data validation

Aerospike Hacks: Mesh Seed Address

Using an AWS ELB-backed DNS entry for the Aerospike Mesh Seed Address

- Downside
  - A bit finicky when creating a new cluster
    - we don't create new clusters often
  - Technically not recommended by Aerospike Support
- Upside
  - Works great with Auto Scaling groups
  - Simpler operation: We don't have to have a separate script/service updating the Aerospike configuration with ip addresses
  - Works with Aerospike client too

Aerospike Support

A couple space squeezing tricks

- Memory (index) space
  - Single set with multiple bins
  - Increase Memory High Water Mark
  - Allocate new namespaces based on **Unallocated** disk/memory
    - NOT based on the **Unutilized** disk/memory
- Disk space
  - Increase Disk High Water Mark
  - Increase defrag-lwm-pct
- Data layout
  - Single characters for bin names
  - Using lists instead of maps
  - Setting newer “epoch” to get smaller timestamp integers

AWS Issues

- Placement groups have not worked out for us
- Capacity issues
- Reservation juggling
- AWS sometimes puts many of our instances on the same physical machine
    - When that machine dies, we lose multiple nodes at once

Disaster recovery

- Have a (tested) plan
  - You'll end up using it

Client Performance Debugging

Client performance is just as important as Server performance.

Aerospike Client monitoring

`AerospikeClient.getClusterStats()`

- Cluster stats
  - # connections used
  - # connections in pool
- Event Loop stats
  - # commands in process
  - # commands in queue

Connection spikes

Client timeouts causes socket to close

-> Client needs to make new connections

times millions of requests per second

-> hit proto-fd-max limit on number of connections very quickly

- timeoutDelay
- socketTimeout
- totalTimeout

Async reads & Java Garbage Collection

High volume Async reads can cause large backlog on the EventLoop delay queue

-> Can cause Java Garbage Collection issues

maybe also exacerbated by connection churn?

Non-Aerospike improvements to the client's operating environment improved the Aerospike client's performance

Migration: Frequency Capping use case

What is Frequency Capping

- Use case: provide advertising clients controls on how often individual users are shown ads
- To support this use case we need to store data about the ads shown for each user

Initial designs… Which we threw away

- We investigated how the storage for this data was working
- We found it was using a custom storage structure inside keebler
  - List of key-value pairs on each record
  - Only data type was integers
  - No delete operation, set values to `“0xFFFFFFFFFFFFFFFF”` instead
  - Strange update semantics
- We came up with a design to recreate it in Aerospike
- But we decided to throw out the design
- Back to the drawing board…

With a clean slate we made something better

- We understood the use case for our customers
- We decided build a better solution
  - Came up with a clean interface and simple data layout
  - Left room to develop future functionality
- Established a correctness metric to evaluate the new system
- Rolled out in stages, validated it, and fixed issues until it was rolled out globally

Key takeaways from the Frequency Capping migration

- If you deeply understand the use cases during a migration you'll have a chance to make a better solution in the process

- Being able to run two systems in parallel, measure them with objective metrics, and experiment on them was critical for the success of this migration

Other things we've done: Supporting new use cases

- We have already launched new products, with customer adoption, relying on new datasets stored in Aerospike
- We have improved the quality of data provided to batch processing use cases
- We're fielding a good number of requests for development of new datasets

Other things we've done: Move to cheaper, more efficient instances

1. We've moved to the i3en instance class in AWS
2. We're seeing good performance and they're way cheaper

The Journey

- The keebler dragon has been slain
  - We've won riches and feature velocity
  - Living happily ever after in operational stability?
  - Everlasting fame at Aerospike Summit?

- What was important
  - Deeply understanding use cases
  - Client performance is critical
    - Especially tricky in resource-constrained environments
  - Being able to tune in production
  - Metrics driven validation

Future Aerospike developments we're excited about

- Improved client performance
- Improved XDR

Q&A