Hot Swap Your Datastore:
A practical approach and lessons learned

Mehmet Can Kurt
Raj Shekhar
Hello, my name is

Raj Shekhar
@ilunatech
Staff Systems Engineer, Quantcast

Mehmet Kurt
Senior Software Engineer, Quantcast
About Quantcast

Radically simplify advertising and privacy for publishers and brands on the open internet.
Real Time Bidding (RTB)

Buying and selling of online ad impressions through real time auctions

example.com

Ad Exchange
Some Numbers

1.8 Million  Number of bid requests per sec we receive in largest region

3.3 Million  Number of bid requests per sec we receive in all regions

700 Million  Models evaluated by bidders per sec

<80 msec  Total time we have to respond to a bid request
Keebler

- Quancast’s distributed cookie data store since 2010
- A Keebler cluster in every AWS region

Features

- GeoIP
- Bid lockout
- Cookie maps

Segments

- Frequency capping
- Fraud filtering
A single Keebler node

- Sharded

- SSD:
  - Re-computed every day
  - Immutable

- Redis:
  - Realtime updates
Keebler uses caching

memcached

cookie data

merge

SSD

redis

Keebler LRU Hit Percentage
Why replace Keebler?

- Sheer number of machines

<table>
<thead>
<tr>
<th>Region</th>
<th>Keebler Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East</td>
<td>172</td>
</tr>
<tr>
<td>US West</td>
<td>129</td>
</tr>
<tr>
<td>Europe</td>
<td>173</td>
</tr>
<tr>
<td>Asia</td>
<td>130</td>
</tr>
</tbody>
</table>

$1.8$ Million

Infrastructure costs just to run the hardware
Why replace Keebler?

- Operational complexities: manual sharding/resharding
- Unreliable performance
  - Service restart upon new SSD files
  - EBS volumes running out of IOPS credits
LRU Hit Percentage Fluctuation

Keebler LRU Hit Percentage

0 20 40 60 80 100 120
03:00 06:00 09:00 12:00 15:00 18:00 21:00
EBS Volumes running out of IOPS credits
Bad long tail of latencies
Loss of institutional knowledge

Feb 22, 2009 – Feb 25, 2020

Contributions to master, excluding merge commits
Indirect costs

- Running a map-reduce job with 300TB sort every Sunday
Requirements for the new system

● Read/write volumes at peak traffic
  ○ 900K reads, 100K writes

● Reads must be fast (1-2 msec)
  ○ slow writes are more tolerable

● Cost ($$$)
  ○ must scale vertically

● No manual intervention for sharding
● Connectors with data frameworks
● Observability, support, …
Proof of Concept

- Aerospike cluster: 20 c5d.18xlarges, 10 billion records, 16TB data
- Client setup: 864 clients (on idle m4.xlarges) using sync API

594 Million cookies written

6 Billion cookies read

0 number of failures

%99.9 of reads writes less than 5 msec
Design Choices

- Implement the new system as a library

  ![Diagram showing Qfs.jar and Aerospike connection]

- Benefits:
  - Single network hop
  - Less number of machines
  - Think twice before adding yet another use case
Design Choices

- Stay away from cryptic representations as much as possible

- Abandon *edge cookie file* format
  - Only Keebler can understand

- Use standard data structures for storing key value
Requirements when performing the migration
01 At any point switch back to legacy system

02 Ability to run the system in hybrid mode

03 Keep the costs down when running two systems in parallel

04 Verify the correctness of the data in Aerospike against Keebler

05 Get equal or better latency performance from Aerospike

06 No downtime in our bidding service
03

How we met the requirements
Step 1

Clear go/no-go metrics and gradual deployment process
01 Latency
99\textsuperscript{th} percentile latency for the datastore lookup, as well as the bidder service

02 Throughput
Number of bids processed

03 CPU
Any changes (+/-) in CPU usage

04 Performance
Impressions, auction spend
Requirement 1

Ability to quickly switch back to keebler. Rollbacks were our best safety net
Feature Flags

Feature flags

The datastore to be used is controlled through config files. The config files are managed by puppet

Code enabled using feature flags

At startup, read the config file and decide what to use

$unifiedfeaturestore_enable_optout = extlookup("mux_unifiedfeaturestore_enable_optout", "false")
$unifiedfeaturestore_lookups_enabled = extlookup("mux_unifiedfeaturestore_lookups_enabled", "false")
$unifiedfeaturestore_use_features_for_bidding = extlookup("mux_unifiedfeaturestore_use_features_for_bidding", "false")
$unifiedfeaturestore_use_filter_segments = extlookup("mux_unifiedfeaturestore_use_filter_segments", "false")
Requirement 2

Run system in hybrid mode. Have consistent response no matter which datastore the clients connected to
Keep both datastores running

● We kept the data across both the datastores
● We maintained the data pipelines that update both datastores simultaneously
● We provided on-call support for both the datastore systems (no second-class citizens)
Benefits of being able to run in hybrid mode

• Allowed us to not care which datastore was used by bidding systems – we knew upstream bidding client will have cookie features
• Migrate multiple regions sequentially, instead of trying to parallelize. Reduced complexity load
Requirement 3

Verify the correctness of the data in Aerospike against keebler. We wanted to track key lookup bugs or missing data.
Parallel lookups for all, log a sample

• During this phase, lookup from both datastores and log 1% of responses. Have an offline job to compare response
• We found bugs in our data pipeline, key lookups and verified data consistency
Requirement 4

Keep the costs down when running two systems in parallel
Keep costs down

- Opportunistically kill Aerospike clusters. Terminate cluster during codefreeze and over weekends. **Bonus**: we refined our complete restore process
- Reclaim instances from keebler cluster. As we increased aerospike cluster size, we reduced keebler cluster.
Requirement 5

Get equal or better latency performance as keebler.
## Equal or better latency performance

<table>
<thead>
<tr>
<th>Verify capacity</th>
<th>Track p99 latency</th>
<th>Verification</th>
<th>Release at peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-blocking dark reads from Aerospike in one region (while using Keebler)</td>
<td>Datastore lookup latency, as well as full bidding stack latency</td>
<td>Monitor go/no-go dashboard during release</td>
<td>Any latency bottlenecks would show up during release, instead of waiting a day</td>
</tr>
</tbody>
</table>
Requirement 6

No downtime in services using the datastore
No downtime in dependent services

01
Gradual changes: host, canary rack, region

02
Keep operational support for legacy, as well as new datastore

03
Well tested rollback runbooks
What went well
Reduced infrastructure footprint

- From 500 to 90 hosts: lower operational load
- Reclaimed reserved instances quota from reducing cluster size
- Hard to trace latency spikes due to iops slowness are gone
More deployments and experiments

- Better documented APIs
- Better integration with Jupyter notebooks and spark
  - New models from data scientists
  - New features using UFS
Retire keebler data pipeline

- Killed off 300Tb sort job, no more spikes over weekends
- Scaled down cluster supporting keebler
Unpleasant surprises
Extended deadlines

• Higher latency when using the non-blocking aerospike api. During prototype, we had used sync api.
• Teams were using keebler to store multi-region data
  • Hard to discover small use cases
  • Had to re-discover why some data was being used through keebler
Latency spikes due to too fast cluster reduction

Teams had been using keebler as a multi-region store. We found this when we would reduce cluster size and there would be reports latency spikes.
Audience Takeaways
Audience takeaways

Replacing a major component of your distributed system is feasible. However, there is no out-of-box solution. Be prepared for a cycle of deploy, find a bug, rollback, fix

Avoid synthetic benchmarks
For proof of concept, use production software under production loads

Have safety nets
Reliable rollbacks, controlled changes, feature flags

Support hybrid mode
Clients running in hybrid mode helps to smooth the migration

Plan rollbacks first
Before big infrastructure changes, plan how to rollback and resurrect
Thank you

• Q&A – on Twitter: @ilunatech
• https://www.quantcast.com/blog/category/engineering/