Building Cloud Services with Riak

Andy Gross, @argv0
Principal Architect, Basho Technologies

Erlang Factory SF 2012

What is a cloud service?

- Web scale (seriously)
  - highly available
  - globally distributed
  - horizontally scalable
  - operationally simple

Riak is Web Scale

- Web scale (seriously)
  - highly available
  - globally distributed
  - horizontally scalable
  - operationally simple

NoSQL Complexity

- Quorum controls: R, W, DW, PW, PR
- Backend choices
- Unfamiliar query model
- Client libraries sometimes immature
- Difficult to sell!

Use Riak as foundation for simpler, “vertical” services (like cloud storage).

Vertical Services

- Abstract away NoSQL complexity
- Provide simpler APIs
- Have fewer configuration knobs
- Have existing client libraries (sometimes)
- Easier to sell!

Riak Service Patterns

- Riak Core Application
- Stateless Proxy
- Clustered Proxy

Riak Extension Points

commit hooks

custom kv backends

custom hash functions

foo_api

foo_fsm

foo_console

Riak Core

foo_vnode

foo_ring|node_watcher

foo_db

Riak Core Application

- Usually consists of a custom vnode implementation and client API
- Usually requires FSM for coordination with vnodes
- Example:
  - https://github.com/jbrisbin/misultin-riak-core-vnode-dispatcher

Stateless Proxy

- Implemented as client of Riak KV
- Deployed in separate VM
- Uses Riak KV for all state storage
- Proxies have no knowledge of each other
- Scale independently from Riak

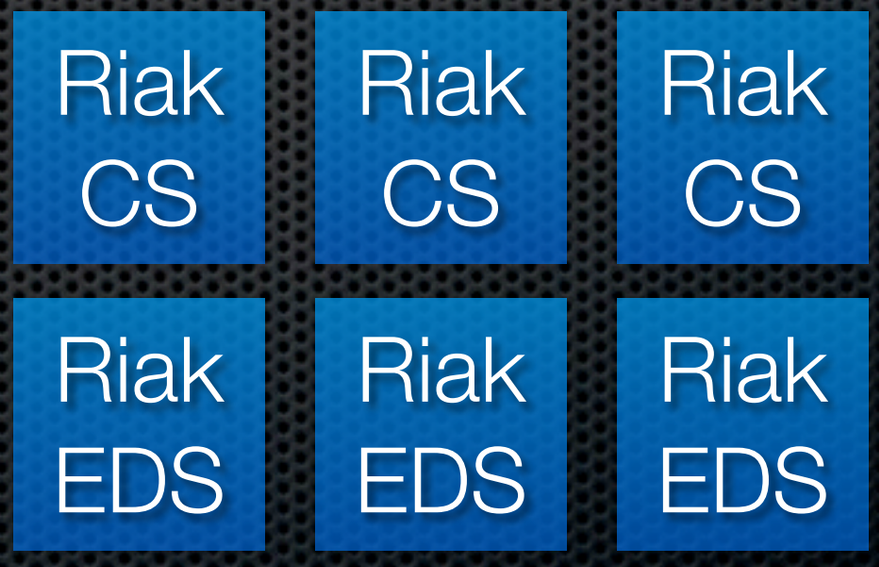
Clustered Proxy

- Use Riak Core at proxy layer for:
  - clustering
  - load balancing
  - distribution of proxy state

Example: riak cs

Riak CS

- Announced on Tuesday
- S3-compatible cloud storage backed by Riak
- Follows “Stateless Proxy” pattern



Riak CS Overview

- Implements S3 API via webmachine
- Large files come in through API, 1+ Riak Objects written:
  - manifest: file metadata
  - chunks: statically sized chunks of large file

What worked well?

Riak KV, Riak Core!

- No code modifications required to build Riak CS
- Resulting service inherits all of Riak's webscale

Tools

- Erlang
- Rebar
- Quickcheck
- Webmachine
- Other Basho Open Source projects

Process

- Started as a prototype
- Iterated quickly with a beta customer
- Shipped frequently
- Small, close team, grown slowly

What sucked?

Connection Pooling

- Just as hard as caching and naming
- # incoming connections > # connection capacity of cluster
  - Started with naive approach
  - Outsourced to proxy software
  - Wrote proper connection pool

Conflict Resolution Is Hard

- Implementation of conflict-handling code can be very tricky
- Required for high availability
- CRDTs may help
- QuickCheck saves the day, as always

Lack Of Strong Consistency

- Some S3 operations need to be atomic
- Riak can’t do this
- Implemented a stopgap solution with less-than-ideal availability properties

Customer Environments

- Everything besides Riak and Riak CS
- Software != Service
  - Planning
  - Provisioning
  - Deployment
  - Monitoring

What may suck soon?

Large Erlang Clusters

- ~150 works well in Riak deployments
- Not sure how much further we can go
- Approaches:
  - shard among many Riak clusters
  - investigate new distribution protocol

Storage Costs

- 3x replication per datacenter gets expensive
- Erasure coding is a possibility
- Smarter global replication
  - notion of “home cluster” with 3 copies, others hav e 1 or 2

Conclusions

- Riak makes a perfect foundation for large scale internet services
- Basho will make more of these
- Lots of work to do on the environments riak/riak cs runs in

Questions?