
Riak Intro

Us

- Shanley Kane
  @shanley
  shanley@basho.com

- Mark Phillips
  @pharkmillups
  mark@basho.com


riak

What's in store?

- At a High Level
- For Developers
- Under the Hood
- When and Why
- Some Users
- Commercial Extensions
- 1.2 and Roadmap

At a High Level

Riak

- Dynamo-inspired key/value store
  - with some extras: search, MapReduce, 2i, links, pre- and post-commit hooks, pluggable backends, HTTP and binary interfaces
- Written in Erlang with C/C++
- Open source under Apache 2 License

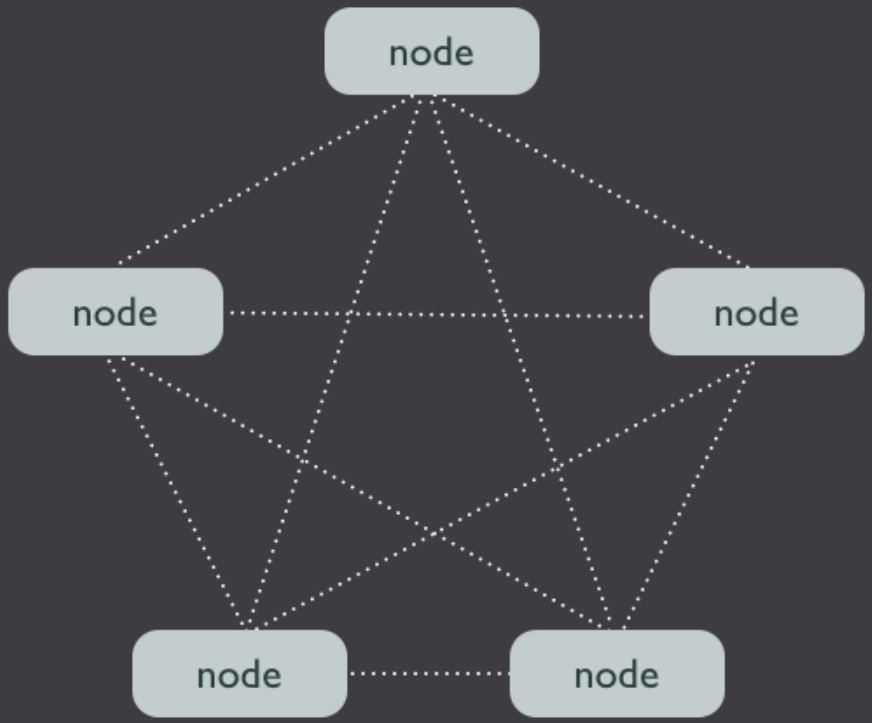
Riak's Design Goals (1)

- High-availability
- Low-latency
- Horizontal Scalability
- Fault Tolerance
- Ops Friendliness
- Predictability

Riak's Design Goals (2)

- Design Informed by Brewer's CAP Theorem and Amazon's Dynamo Paper
- Riak is tuned to offer availability above all else
- Developers can tune for consistency (more on this later)

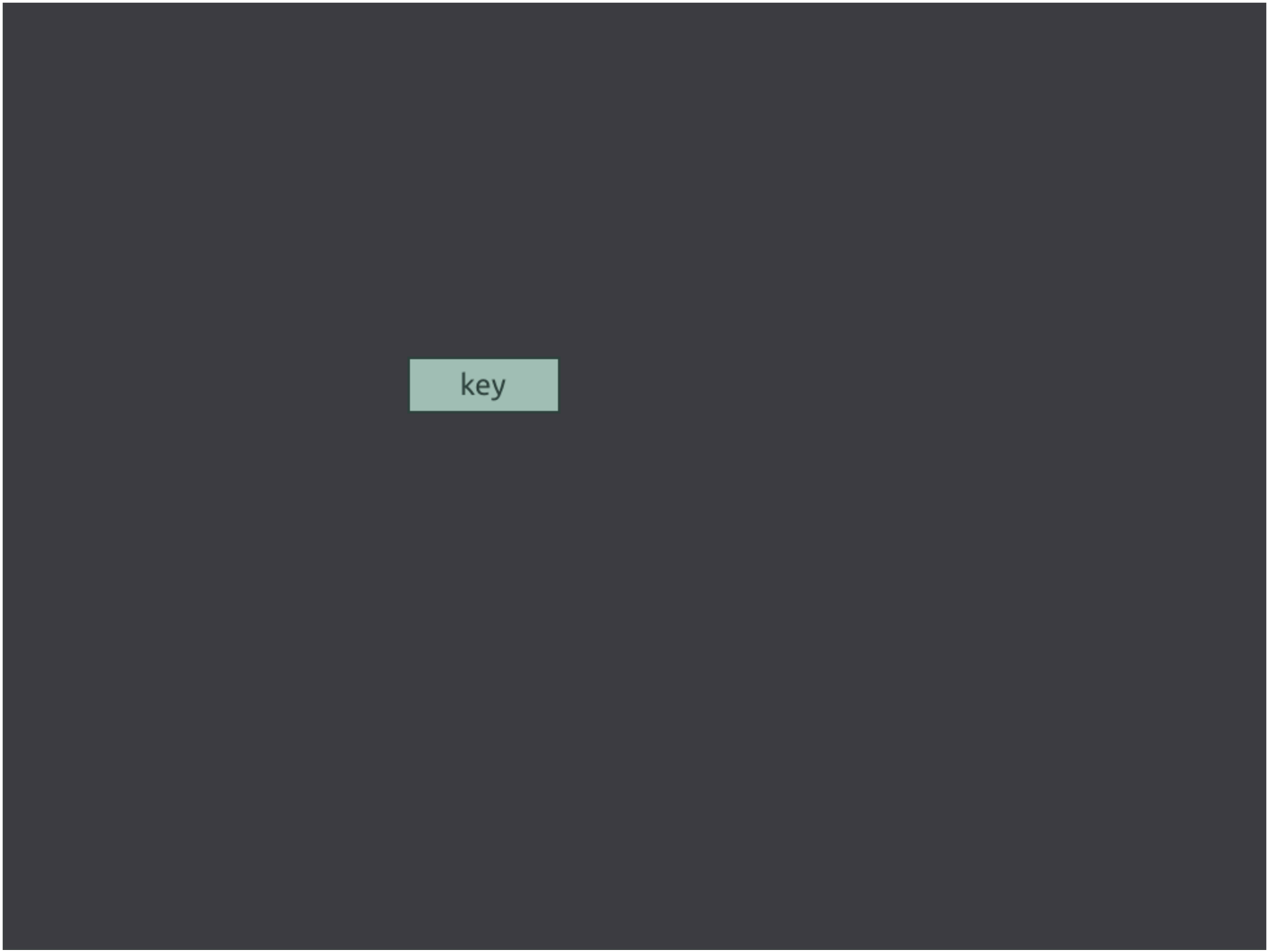
Masterless; deployed as a cluster of nodes



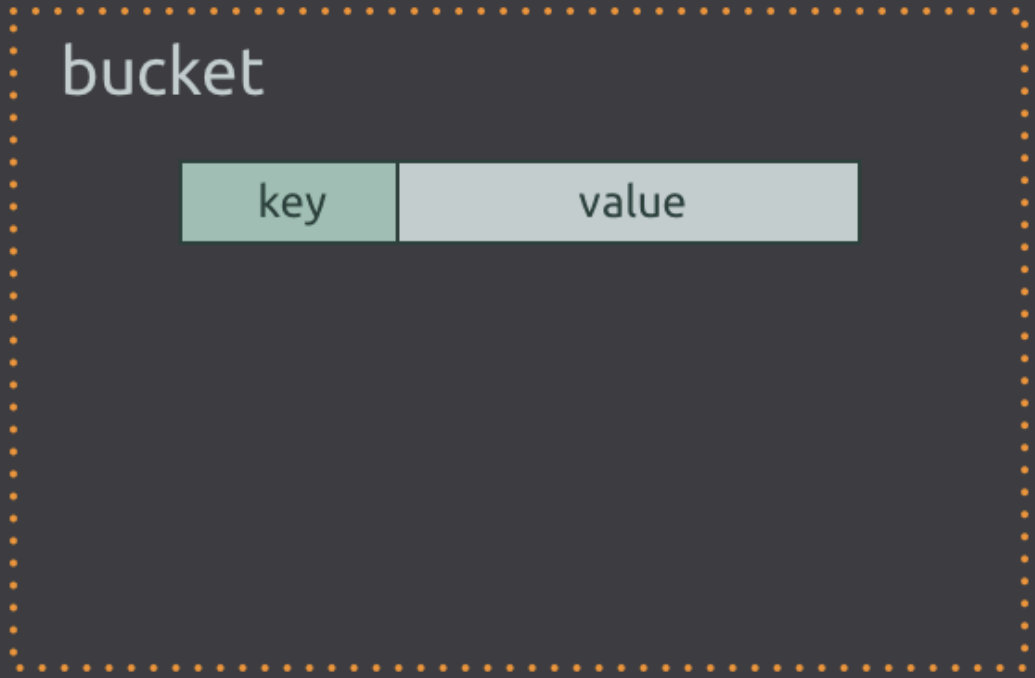
For Developers

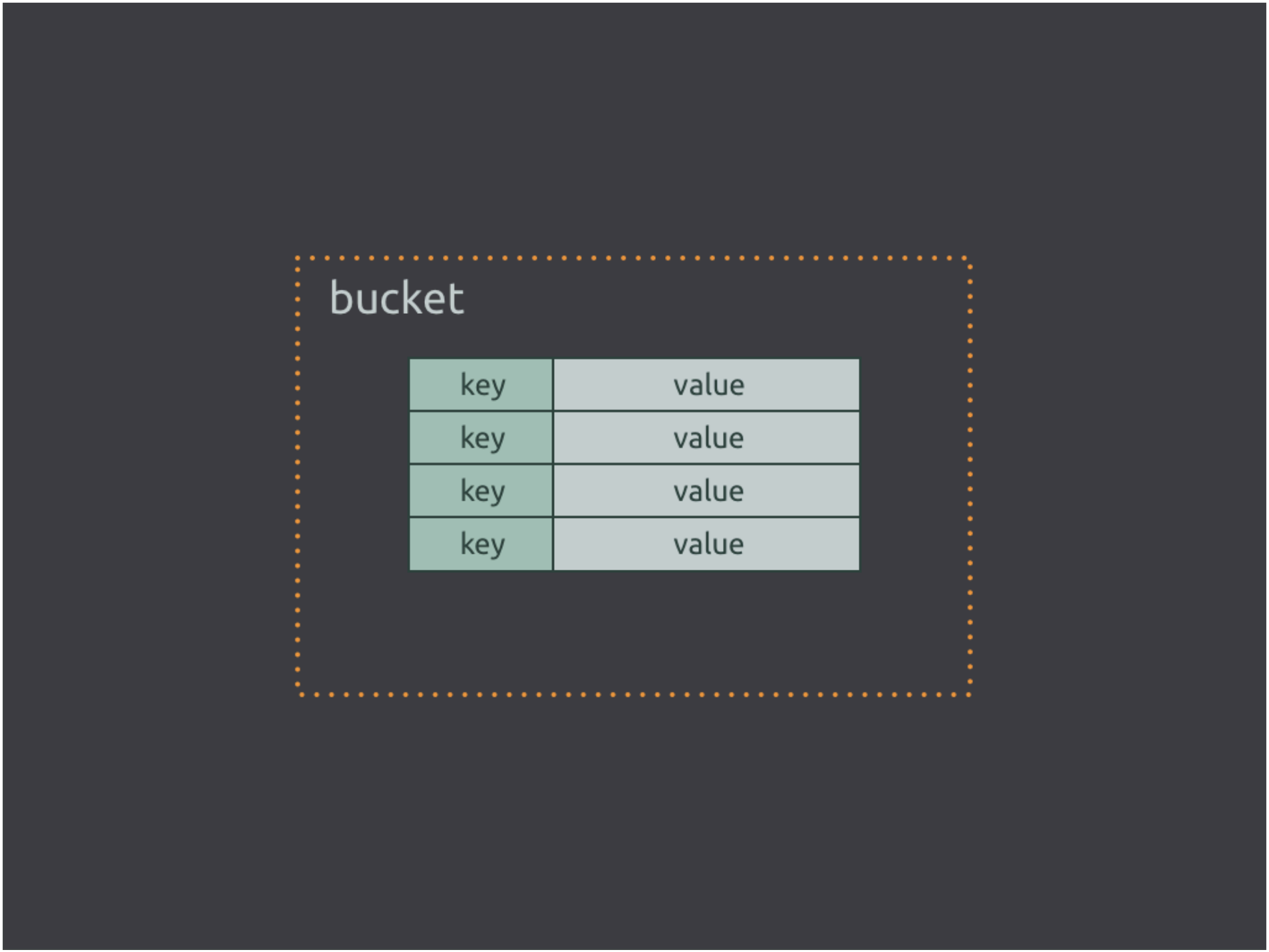
Riak is a database that stores keys against values. Keys are grouped into a higher-level namespace called buckets.

Riak doesn’t care what you store.
It will accept any data type; things
are stored on disk as binaries.


key


key
value


bucket
key
value


bucket
key
value
key
value
key
value
key
value

Two APIs

1. HTTP (just like the web)
2. Protocol Buffers (thank you, Google)

Querying

GET/PUT/DELETE

MapReduce

Full-Text Search

Secondary Indexes (2i)

Tunable Consistency

- n_val - number of replica to store; bucket-level setting. Defaults to “3”.
- w - number of replicas required for a successful write; Defaults to “2”.
- r - number of replica acks required for a successful read. request-level setting. Defaults to “2”.
- Tweak consistency vs. availability

Client Libraries

Ruby, Node.js, Java, Python, Perl, OCaml, Erlang, PHP, C, Squeak, Smalltalk, Pharoah, Clojure, Scala, Haskell, Lisp, Go, .NET, Play, and more (supported by either Basho or the community).

Under the Hood

Consistent Hashing and Replicas

Virtual Nodes

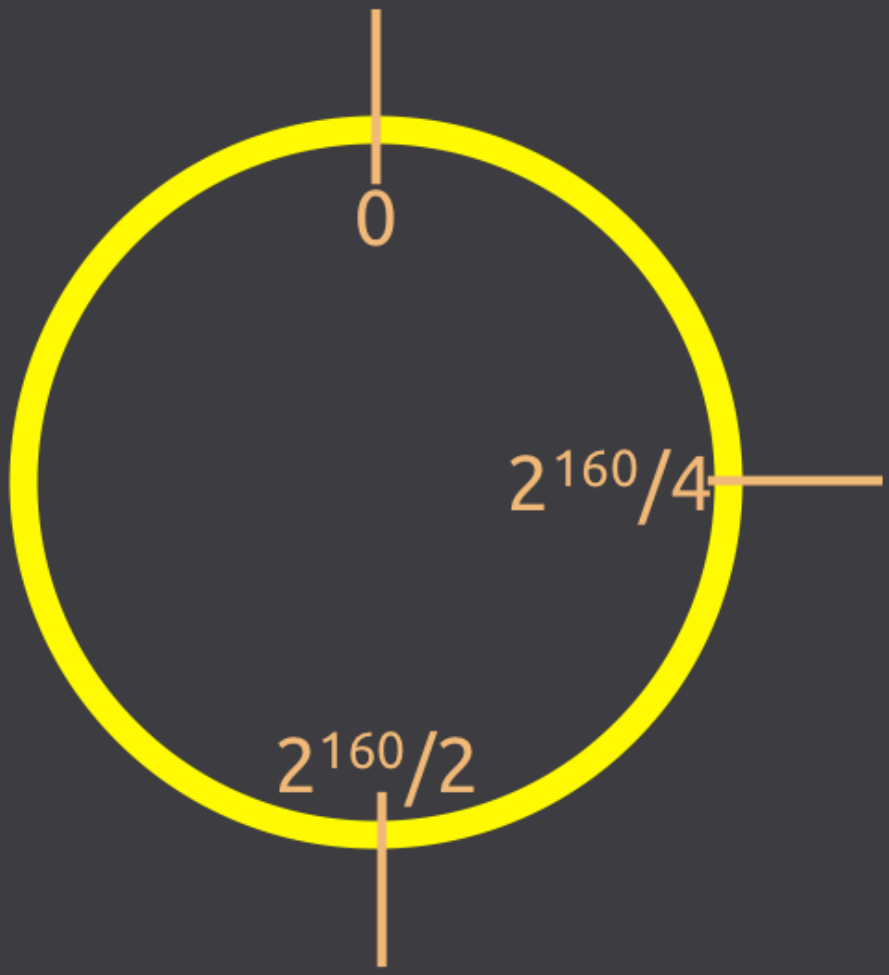
Vector Clocks

Handoff and Rebalancing

Gossiping

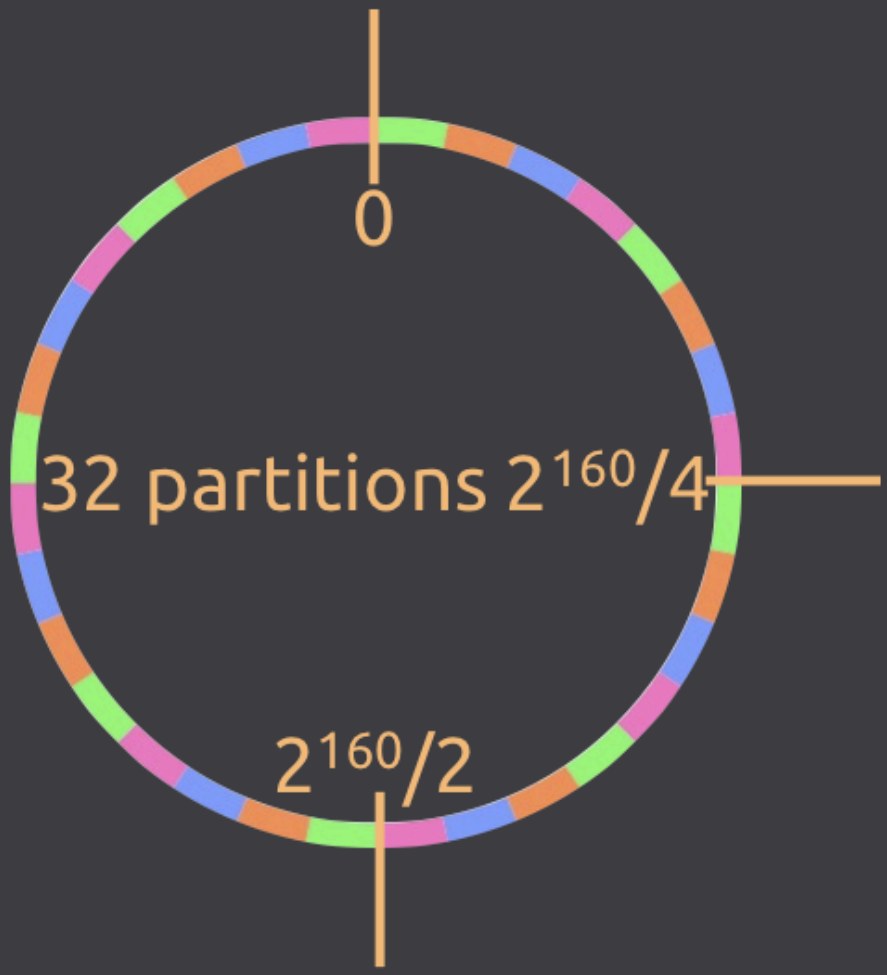
Consistent Hashing

- 160-bit integer keyspace



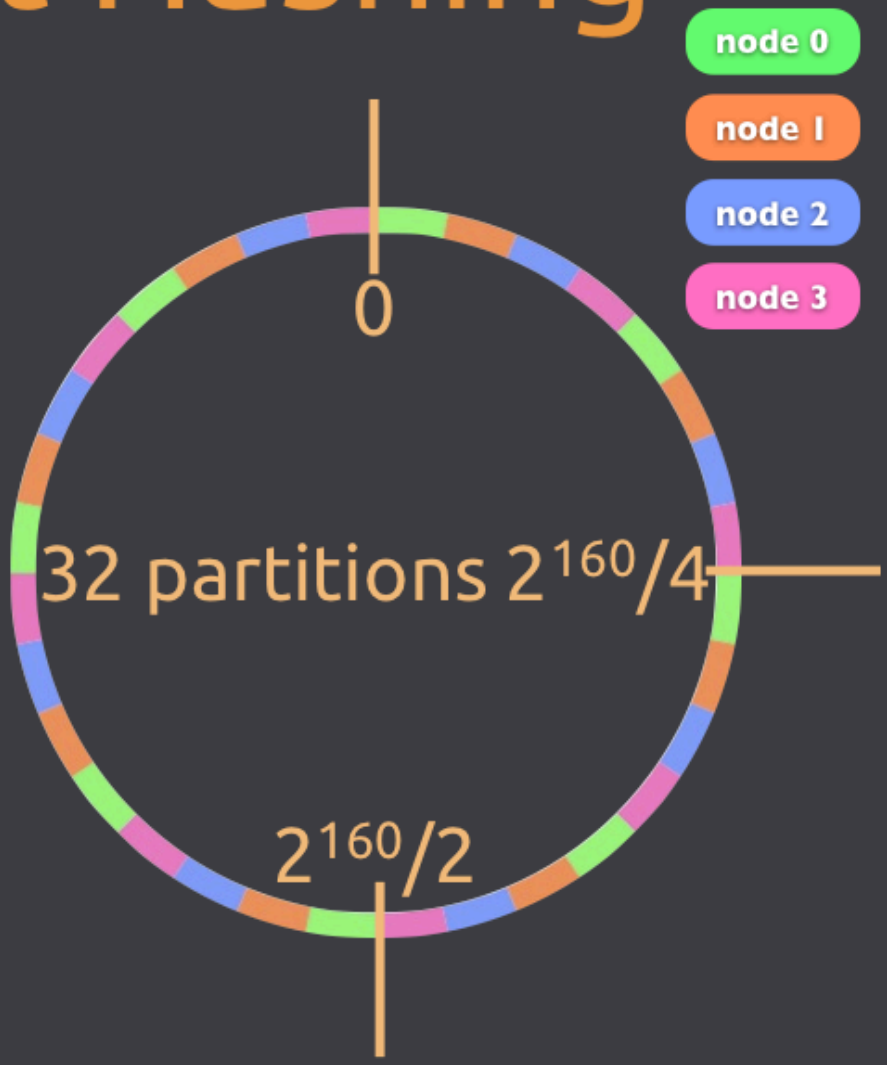
Consistent Hashing

- 160-bit integer keyspace
- divided into fixed number of evenly-sized partitions



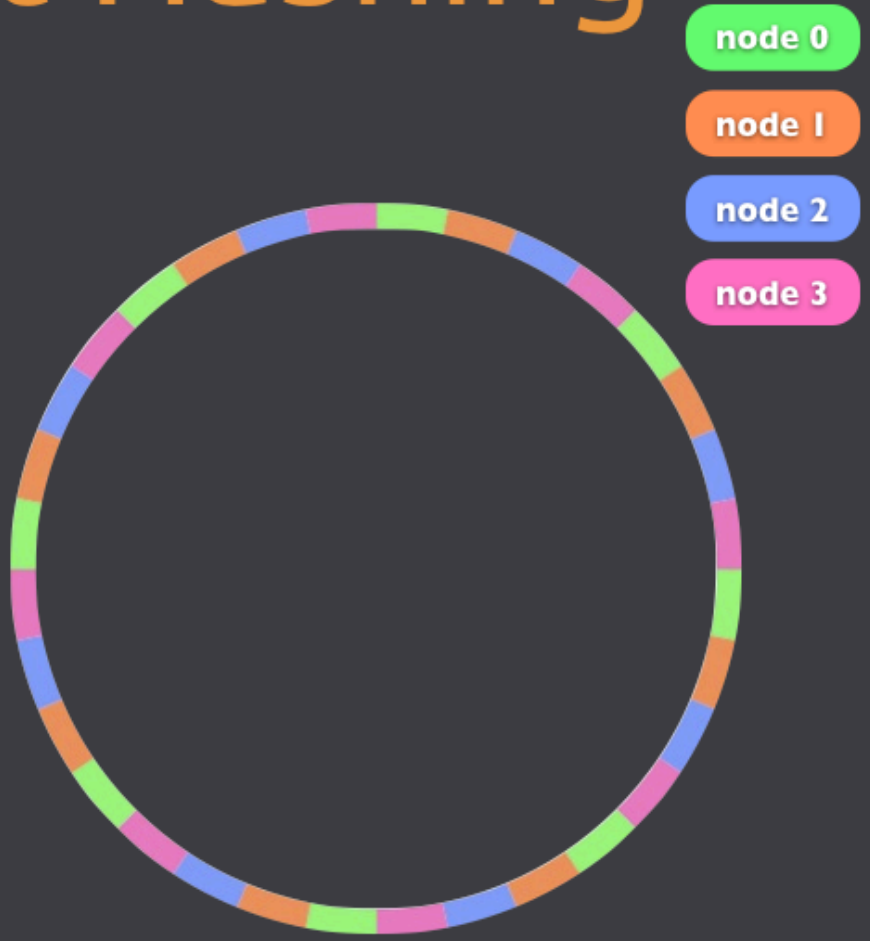
Consistent Hashing

- 160-bit integer keyspace
- divided into fixed number of evenly-sized partitions
- partitions are claimed by nodes in the cluster



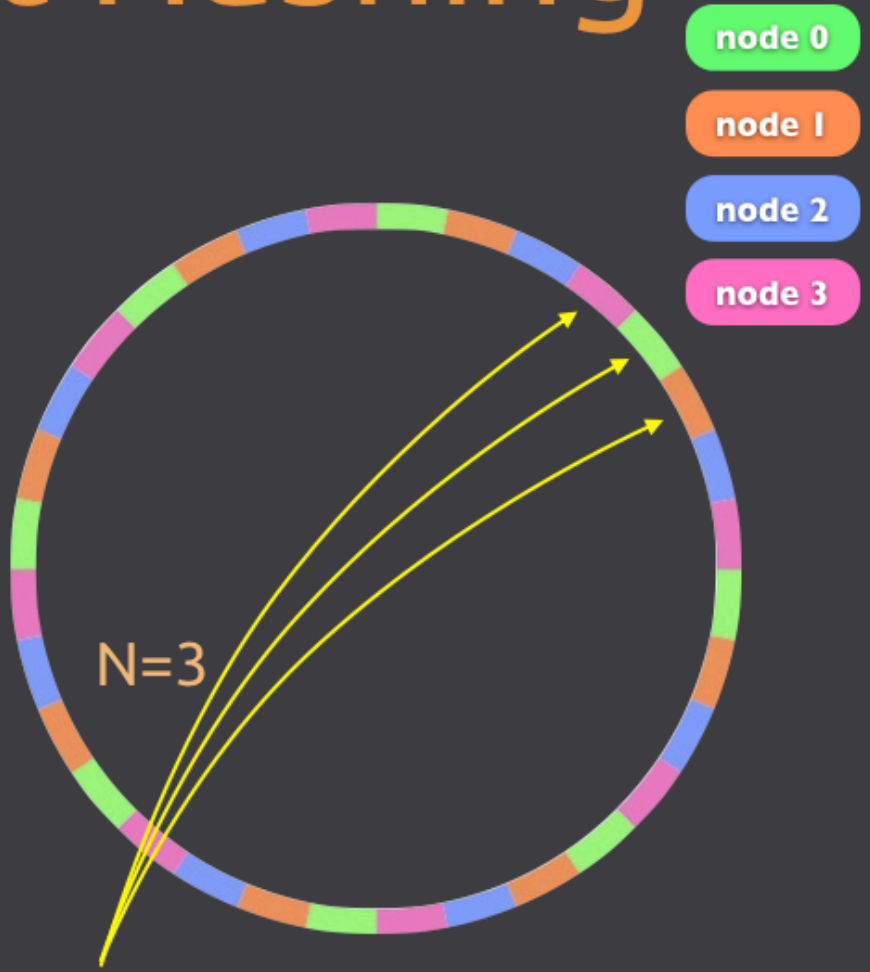
Consistent Hashing

- 160-bit integer keyspace
- divided into fixed number of evenly-sized partitions
- partitions are claimed by nodes in the cluster
- replicas go to the N partitions following the key



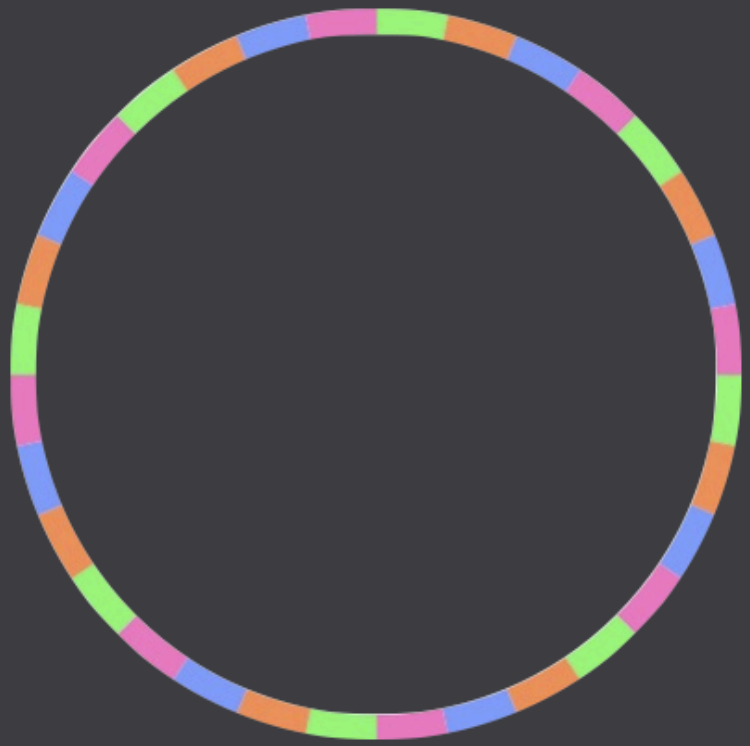
Consistent Hashing

- 160-bit integer keyspace
- divided into fixed number of evenly-sized partitions
- partitions are claimed by nodes in the cluster
- replicas go to the N partitions following the key



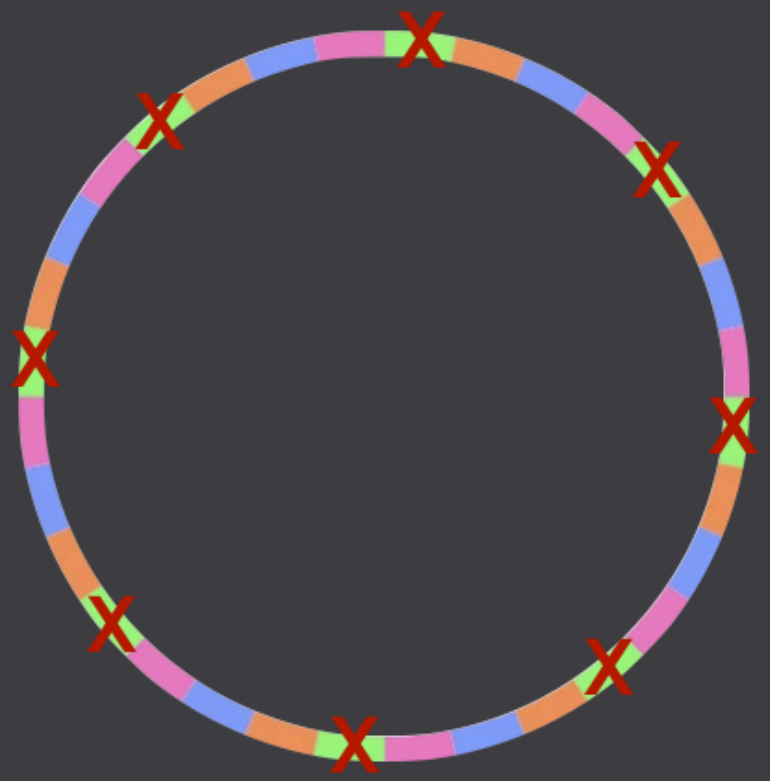
hash(“meetups/nycdevops”)

Disaster Scenario



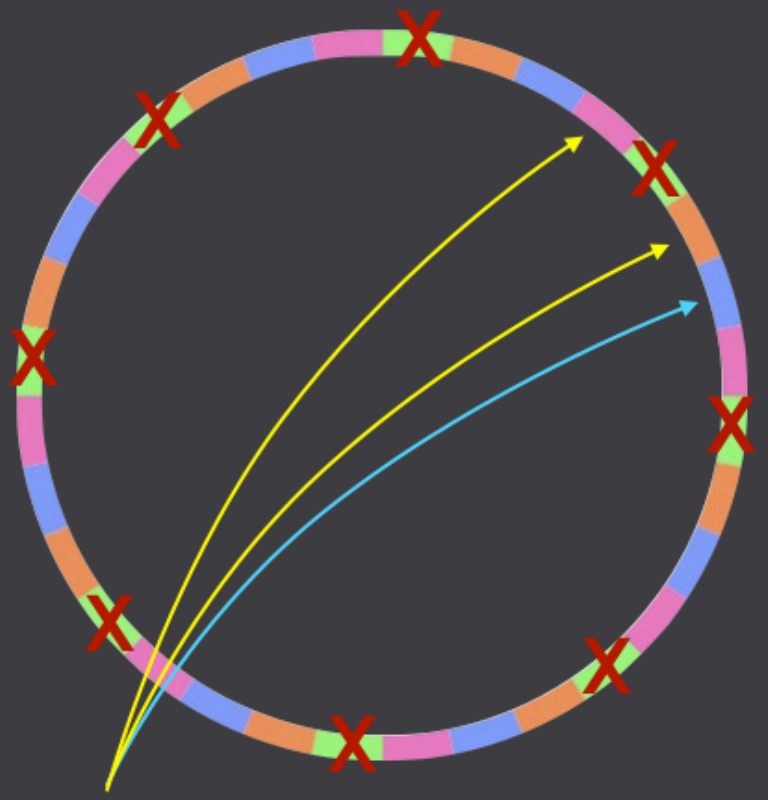
Disaster Scenario

- node fails



Disaster Scenario

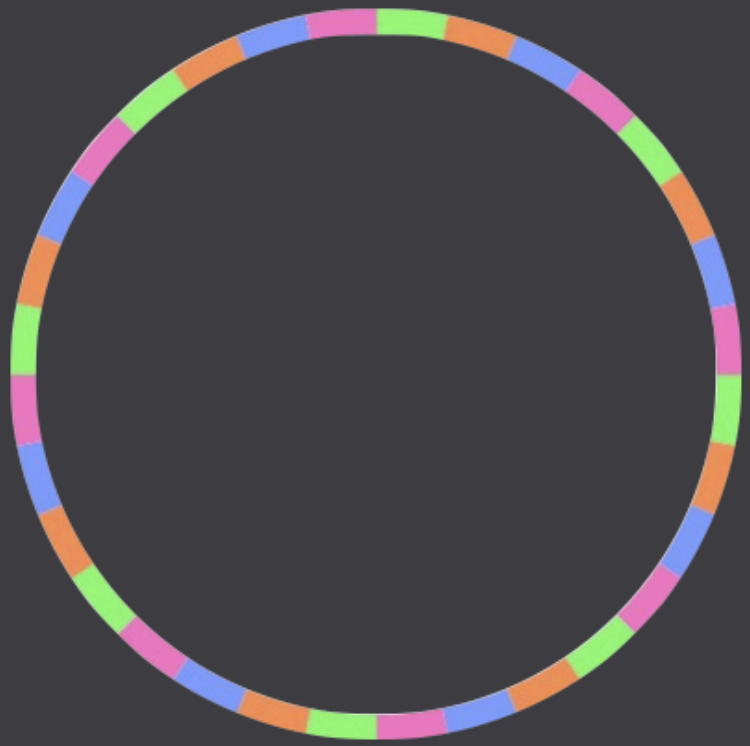
- node fails
- requests go to fallback



hash(“meetups/nycdevops”)

Disaster Scenario

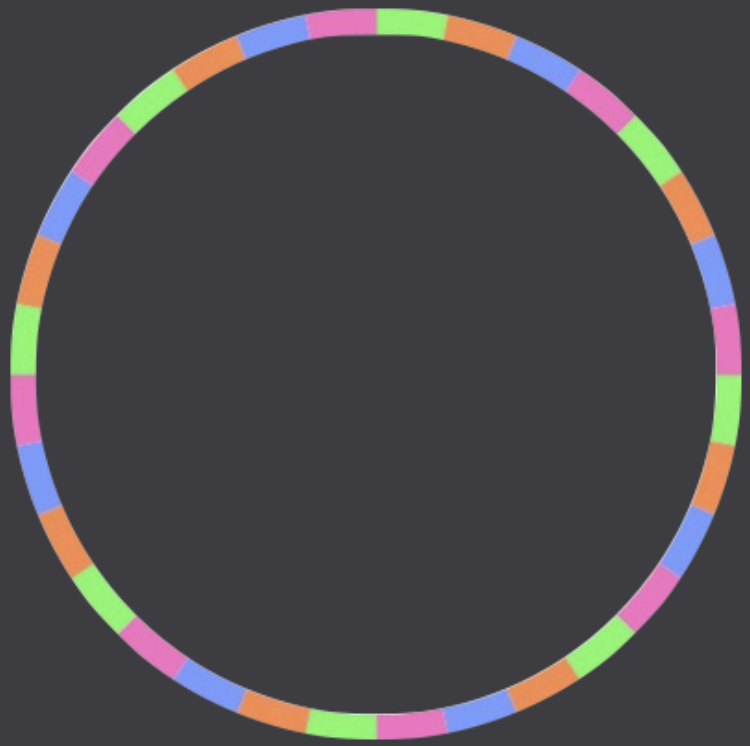
- node fails
- requests go to fallback
- node comes back



hash(“meetups/nycdevops”)

Disaster Scenario

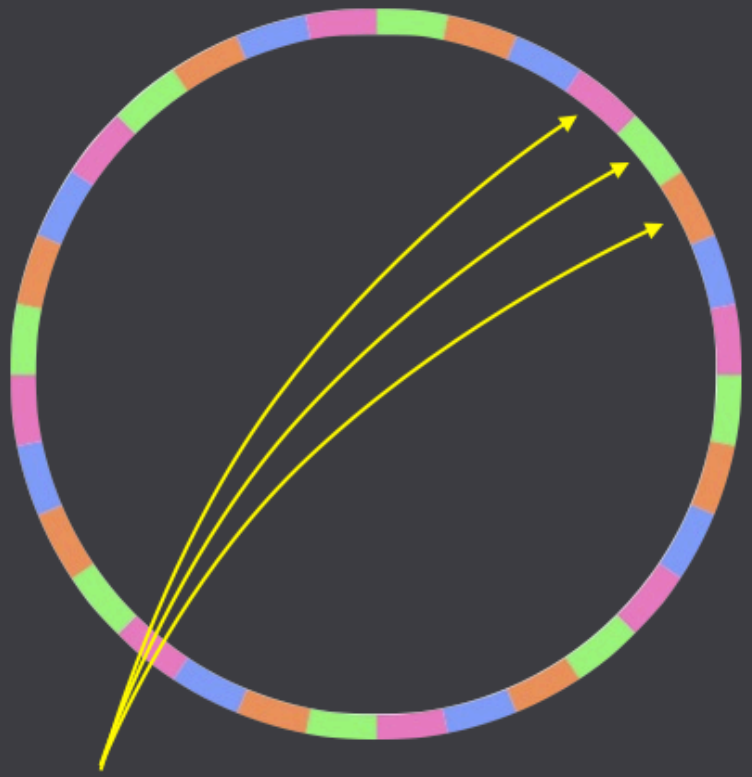
- node fails
- requests go to fallback
- node comes back
- “Handoff” - data returns to recovered node



hash(“meetups/nycdevops”)

Disaster Scenario

- node fails
- requests go to fallback
- node comes back
- “Handoff” - data returns to recovered node
- normal operations resume



hash(“meetups/nycdevops”)

Virtual Nodes

- Each physical machine runs a certain number of Vnodes
- Unit of addressing, concurrency in Riak
- Storage not tied to physical assets
- Enables dynamic rebalancing of data when cluster topology changes

Vector Clocks

- Data structure used to reason about causality at the object level
- Provides happened-before relationship between events
- Each object in Riak has a vector clock*
- Trade off space, speed, complexity for safety

Handoff and Rebalancing

- When cluster topology changes, data must be rebalanced
- Handoff and rebalancing happen in the background; no manual intervention required*
- Trade off speed of convergence vs. effects on cluster performance

Gossip Protocol

- Nodes “gossip” their view of cluster state
- Enables nodes to store minimal cluster state
- Can lead to network chatiness; in OTP, all nodes are fully-connected

Riak: when and why

When Might Riak Make Sense

When you have enough data to require >1 physical machine (preferably >5)

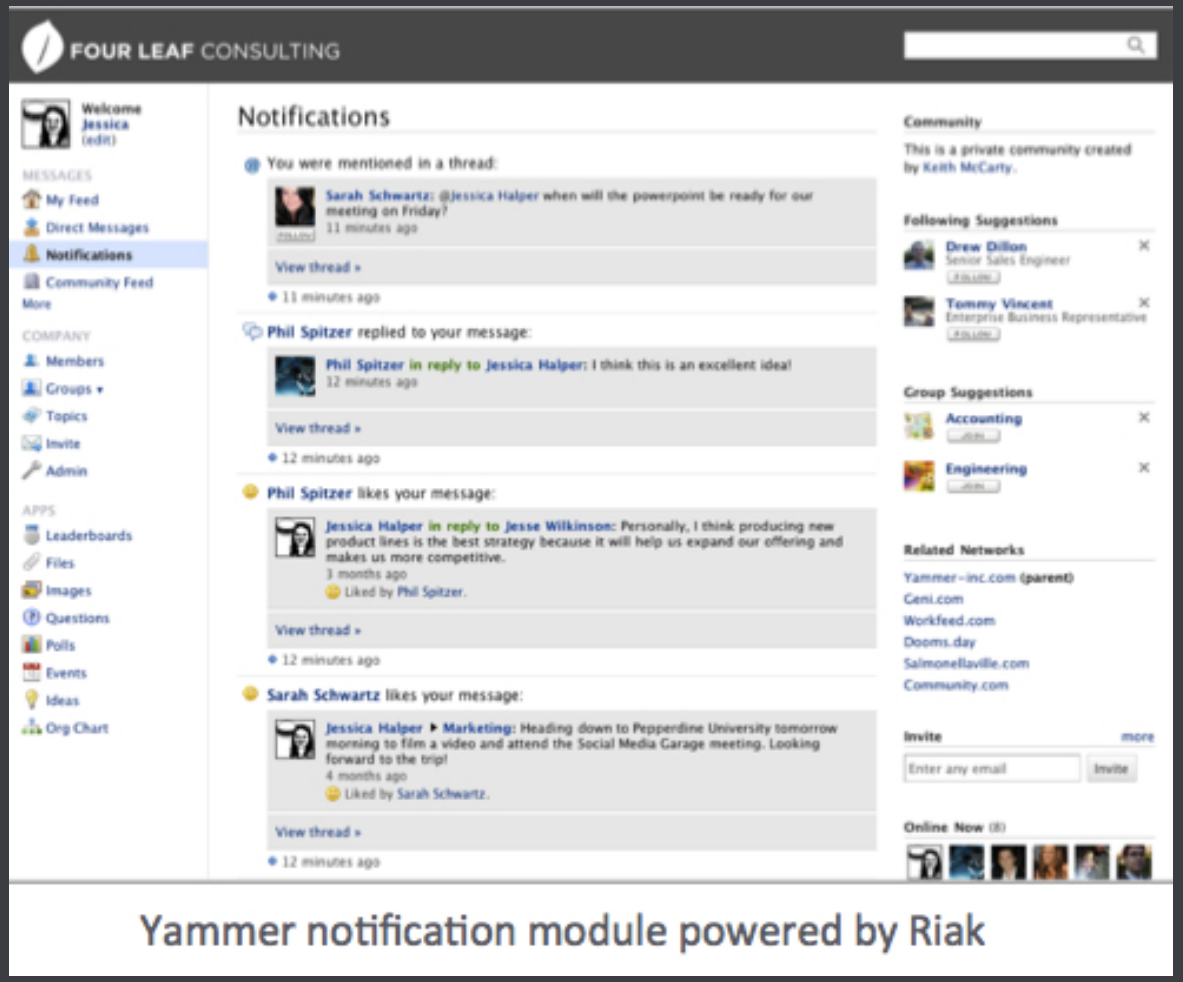
When availability is more important than consistency (think “critical data”on “big data”)

When your data can be modeled as keys and values; don’t be afraid to denormalize

User/MetaData Store

- User profile storage for xfinityTV Mobile app
- Storage of metadata on content providers and licensing
- Strict Latency requirements

Notifications



Yammer notification module powered by Riak

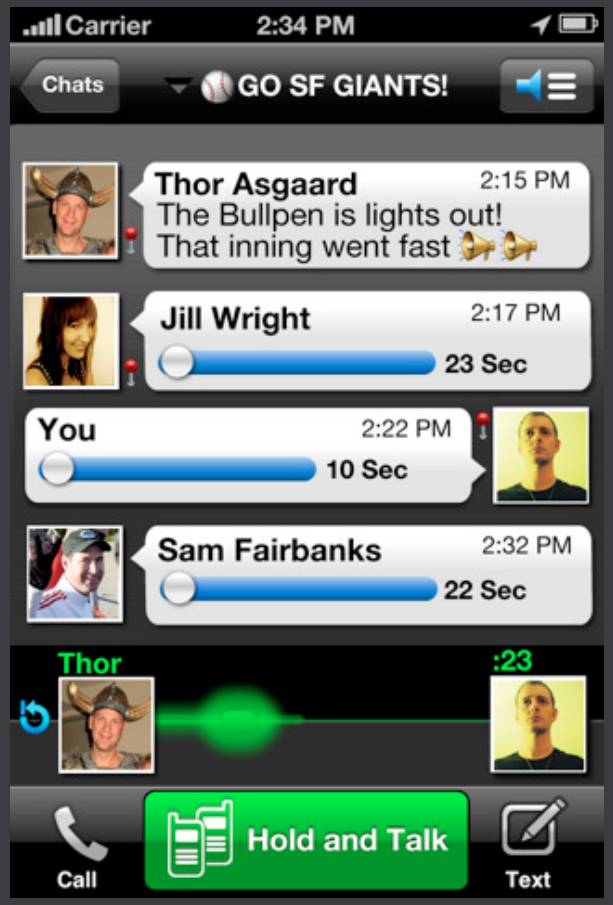
Session Storage

- First Basho customer in 2009
- Every hit to a Mochi web property results in at least one read, maybe write to Riak
- Unavailability or high latency = lost ad revenue

Ad Serving

- OpenX will serve ~4T ad in 2012
- Started with CouchDB and Cassandra for various parts of infrastructure
- Now consolidating on Riak and Riak Core

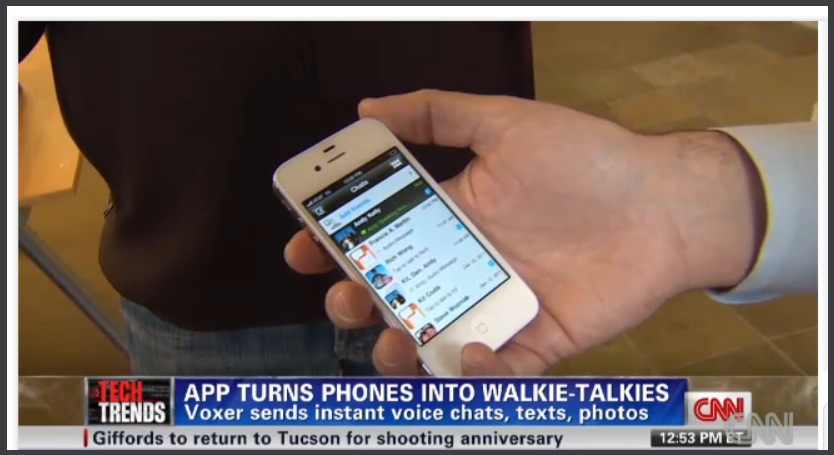
Riak for All Storage: Voxer



Voxer: Initial Stats

- 11 Riak nodes (switched from CouchDB)
- 100s of GBs
- ~20k Peak Concurrent Users
- ~4MM Daily Request

Walkie Talkie App Voxer Is Going Viral On iPhones And Androids, Trending On Twitter



Voxer: Post Growth

- ~60 Nodes total in prod
- 100s of TBs of data (>1TB daily)
- ~400k Concurrent Users
- Billions of daily Requests

Riak : Hybrid Solutions

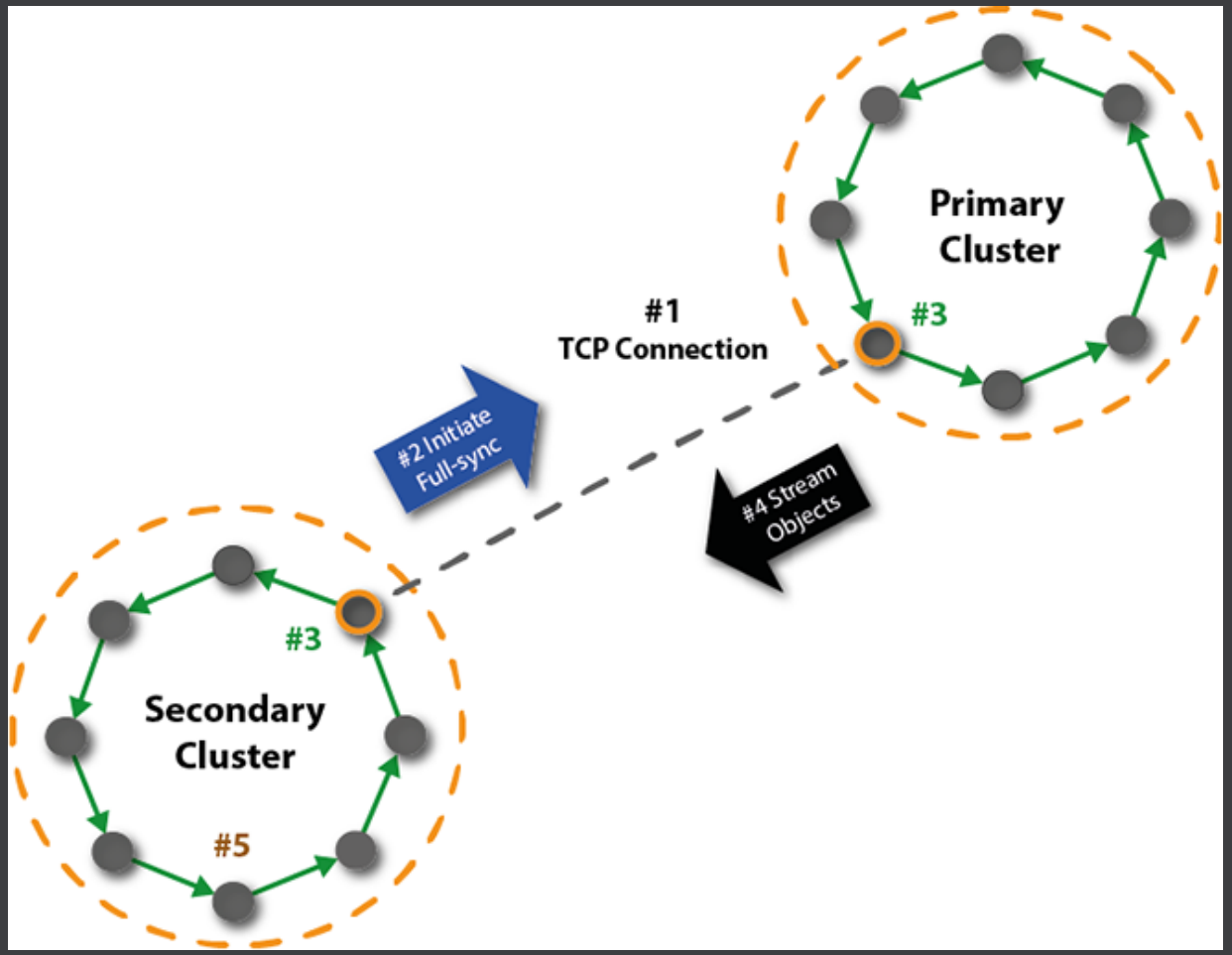
- Riak with Postgres
- Riak with Elastic Search
- Riak with Hadoop
- Secondary analytics clusters

Buy Some Software...

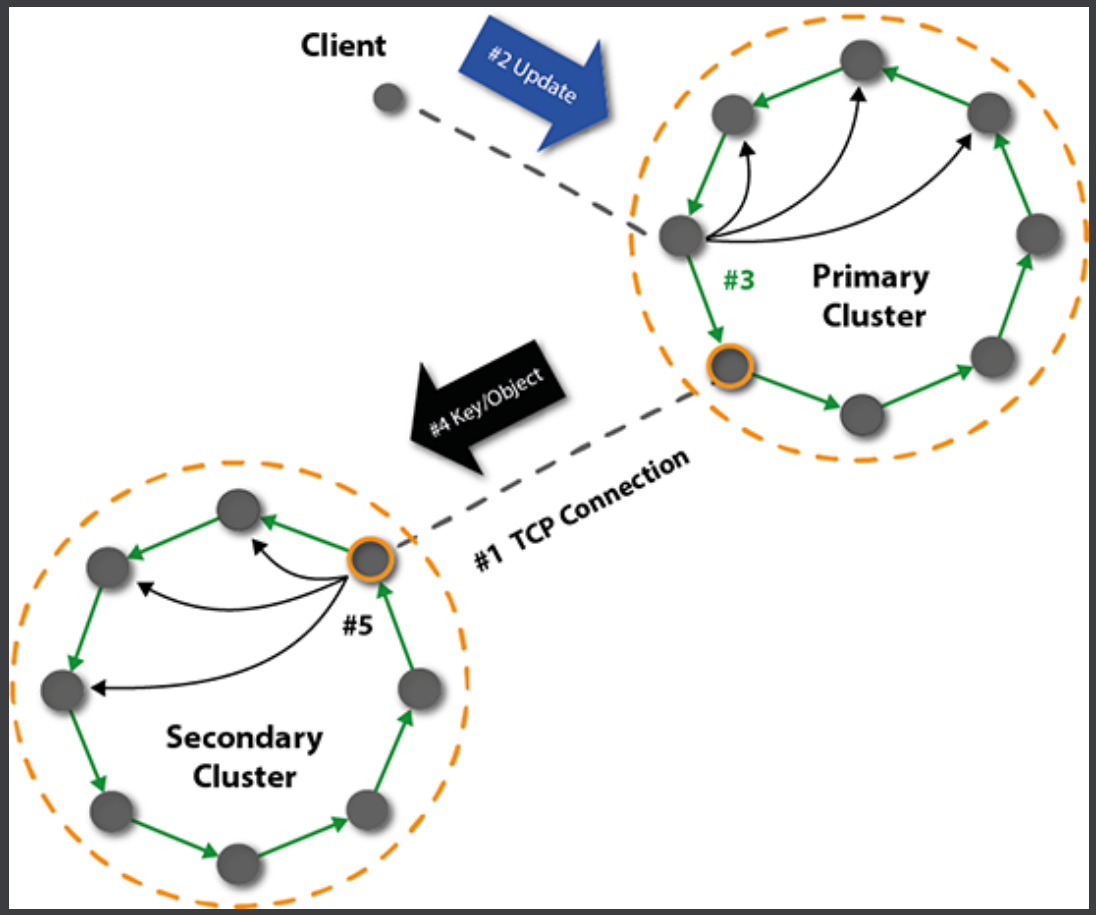
Riak Enterprise

- Multi-data center replication
- Real-time or full-time sync

Riak Enterprise: Full Sync



Riak Enterprise: Real-Time Sync



Riak Cloud Storage

- Large object support
- S3-compatible API
- Multi-tenancy
- Reporting on usage

Roadmap Stuff...

New in Riak 1.2

- LevelDB Improvements
- FreeBSD Support
- New Cluster Admin Tools
- Folsom for Stats
- KV and Search Repair work
- Much much more

Future Work

- Active Anti Entropy
- CRDTs
- Tight Solr integration
- Greater consistency
- Lots of other hotness

Riak

- docs.basho.com
- @basho
- github.com/basho