goto; conference aarhus

DEMYSTIFYING BIG DATA WITH RIAK USE CASES

Martin Schneider
Basho Technologies

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

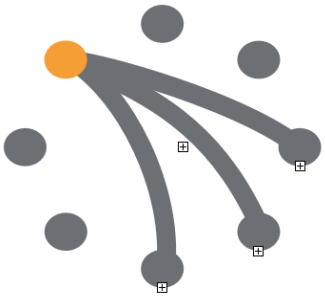
gotocon.com

Agenda

- Defining “Big Data” in Regards to Riak
- A Series of Trade-Offs
- Use Cases
- Q & A

About Basho & Riak

Basho Technologies is the Company behind Riak



Riak is the Distributed Database

riak

Riak:

- Solves problems traditional databases are ill-suited to address - write-intensive, distributed apps, big data, increased uptime.
- Riak makes applications previously thought impossible both easy and inexpensive to realize.
- The world's first commercially available distributed systems management platform (Riak Core) with case-specific data storage & retrieval modules (Riak DSRM)
- Combines principles deployed on two of world's most successful distributed systems:

Defining Characteristics

- **masterless** – no master or slave nodes, subsystems
- **key-value store** -- use a “well known” key to locate values
- **document-oriented** – store anything: text, movies, code
- **clustered for high-availability** – Riak runs on one or many servers, replicates data automatically both within cluster and between data centers
- **industry-standard search** – supports SOLR-based search applications in use throughout industry
- **map/reduce** – uses a version of map/reduce to query data

Basho Riak: the Product

- Open Source and Enterprise versions
- Modules: KV Data Store, Search, Column* & File Store*
- Riak EnterpriseDS – enterprise-focused tools, 24x7 support
- Easy to install, run, scale (<5 minutes to install, configure, query a Basho cluster.)
- Built using same principles as Akamai, leading content delivery network (CDN), by original Akamai team
- Inexpensive alternative to Oracle, MySQL (< 5% TCO)

** these modules available in late Q3 and Q4 of 2010*

basho

Clients and Partners

basho

Big

Volume Data Value

Big

Value

Trade-Off #1

- You Get: Low-Latency, Global Storage
- You Give Up: Granular Query Ability
- Use Case: Purpose-built storage; session stores, file-based storage, etc.

Trade-Off #2

- You Get: Extreme Write Availability
- You Give Up: Strict Consistency of Entire Data Set
- Use Case: E-Commerce shopping cart; social media tools, etc.

Trade-Off #3

- You Get: Ability to Deploy Read/Write Intensive Applications across Data Centers and Device Platforms
- You Give Up: Latency in Offering Some Reads
- Use Case: Hi-Scale Mobile/Social Applications

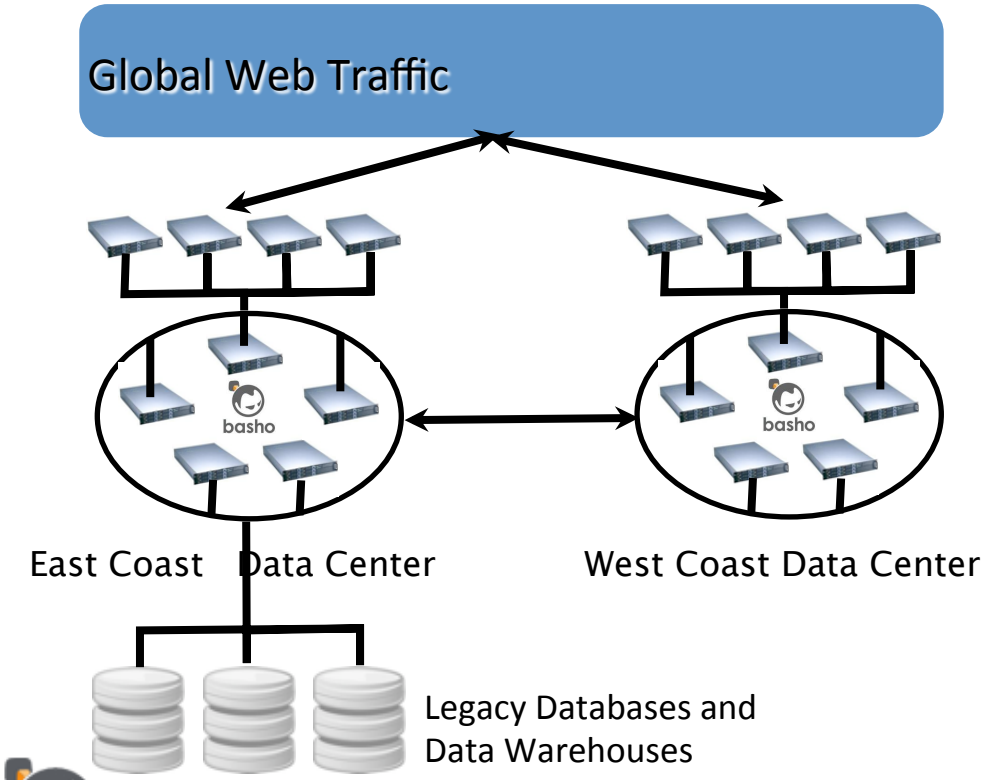
Users and Customers

Riak User: 

- Issue: Multiple data stores for content; subscriber info
- Solution: Riak as a distributed caching layer
- Benefit: Real-time access to premium content on handheld devices

Use Case: Comcast-NBC Interactive

RDBMS can't scale with traffic. Interactive media properties can't use Oracle data. Chose Riak over Couch and Cassandra.



Riak used to unlock the data for high-volume interactive properties.

With Riak the online properties experiences sub-10 ms latencies with reduced error rates.

Riak User: 

- Issue: Lots of unstructured voice data
- Solution: Riak as distributed social graph and content meta data store
- Benefit: Scalable voice texting solution connecting users in real-time

Riak User: yammer

- Issue: Millions of user interactions daily
- Solution: User notification engine built with Riak
- Benefit: Better sort, manage, and prioritize social interactions

Riak User: National Health Service, Denmark

- Issue: Providing personal health records
- Solution: Distributed network of records; available on any device
- Benefit: Reliable system of record for critical personal medical data

Benefits of Building Distributed Systems with Riak:

- **Dynamo-based** – a faithful adaptation of Amazon's Dynamo model
- **Cloud-Ready** – elastic architecture means you can grow clusters dynamically without downtime
- **Master-less** – no single point of failure
- **Fault Tolerant** – survive outages with no data loss
- **Multi-Data Center** – write-available, master-less replication
- **Linearly Scalable** – adding 10% more nodes means 10% more capacity
- **No-Sharding** – consistent hashing means 0% downtime

Summary

- Huge Opportunity: data creation has outstripped storage. New solutions needed.
- Simple, inexpensive, yet powerful modular platform replaces mass of expensive point solutions in enterprise.
- Built using same principles as Akamai, leading content delivery network (CDN).
- In the market with clients and revenue today.

Riak Use Case Resources:

- http://blog.basho.com/2011/03/28/Riak-and-Scala-at-Yammer/
- http://wiki.basho.com/Who-is-Using-Riak.html
- http://www.infoq.com/interviews/sheehy-riak
- http://blog.basho.com/2011/08/08/Riak-at-Formspring-Video-from-SF-Riak-Meetup/
- http://lanyrd.com/2011/erlang-factory-london/sgwxw/ (Riak Mobile/Danish Health Service)

Thank You

martin@basho.com

www.basho.com

www.wiki.basho.com

@mschneider718

Technical Appendix

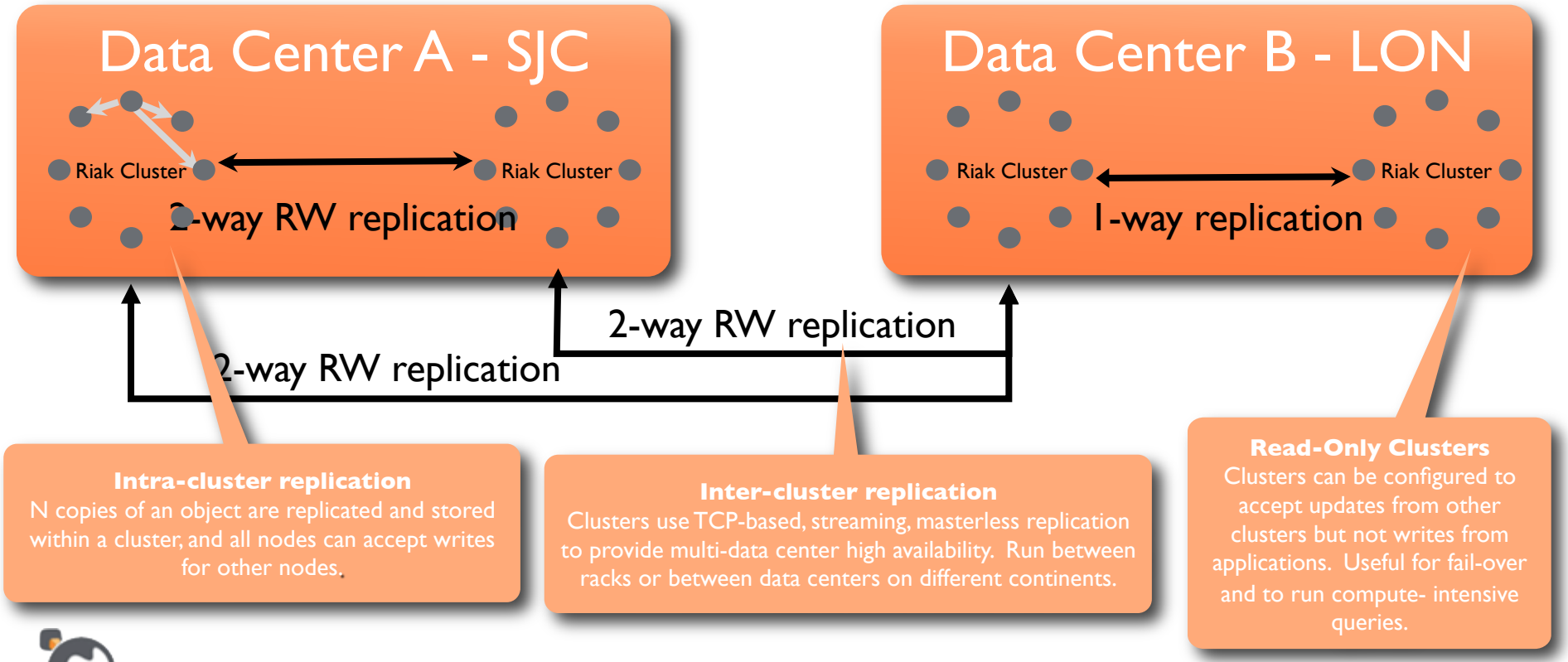
Consistent hashing and anti-entropy subsystems achieve high read-***and*** write-availability in and between data centers.

Basic Concepts

- Simple operations: **get**, **put** & **delete**
- Riak stores values against keys.
- Encode your data how you like it.
- Keys organized into buckets (one-level namespaces)
- Object relationships expressed with links
- Consistent hashing, not sharding.
- Many **vnodes** hashed across fewer physical nodes

Replication within and between clusters

Lose nodes, racks, data centers, or clusters with no downtime or data loss.



basho

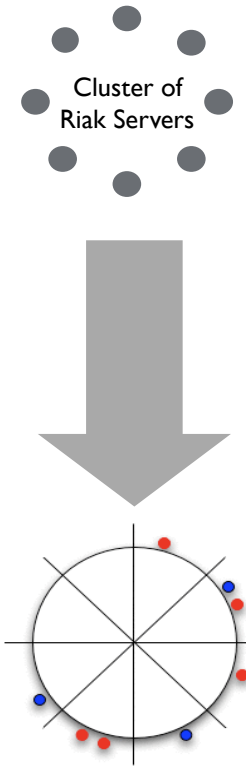
Design Goals

- Always remain write-available
- Ensure Operational Simplicity
  - consistent hashing distributes multiple copies across cluster
  - CAP controls exposed: **N, R, W** values
    - **N**umber of replicas to store,
    - # of **R**eads of **N** replicas or **W**rites to **N** nodes for request to be considered successful
- Scale out, not up – add commodity servers to increase compute capacity, fault-tolerance

Replication and Consistency

vnodes, vector clocks and gossip protocol

- Riak uses the technique of consistent hashing to organize data storage.
- Physical nodes run many vnodes (virtual nodes
- data mapped to vnodes on N # of hosts
- Gossip protocol for vnode ownership distributes load dynamically.
- Adding servers is easy and does not cause interruption.
- Consistency maintained using logical versioning
  - causality and version preserved during availability and partitioning events
  - vector clocks expose conflict for automatic resolution



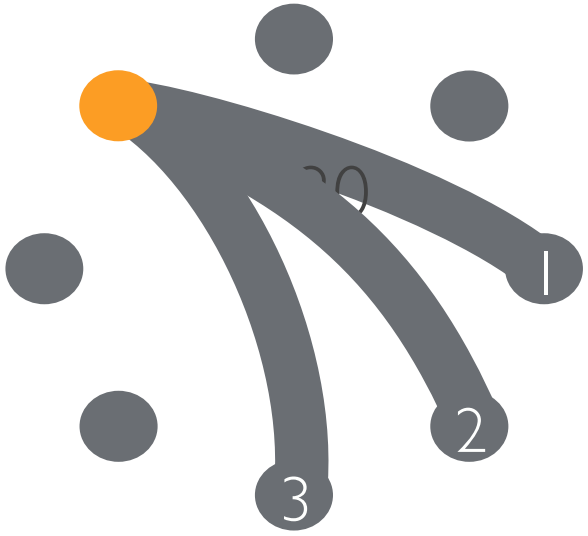
A logical representation of a Basho cluster. Each partition is a vnode. Copies of data are distributed in multiple locations on the ring.

Consistent Hashing, Not Sharding

Consistent Hashing, Superior to Sharding

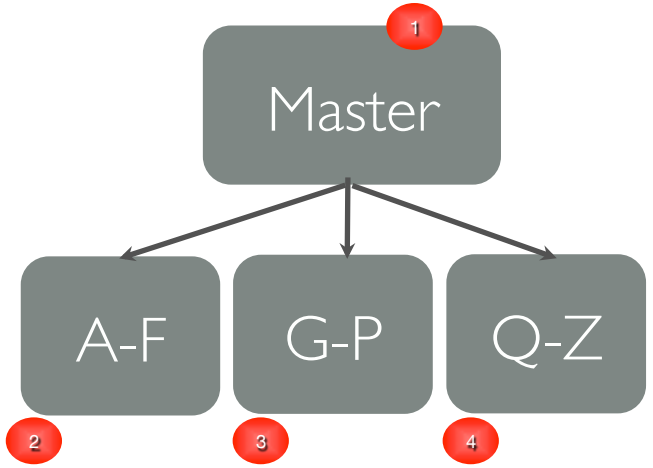
Riak uses consistent hashing to cluster. There is no master and no single replica of data. This technology is what we used at Akamai and what Amazon now uses for their shopping platform.

0 Single Points of Failure



With Riak, data is distributed between N nodes.
In this case, N=3, or 3 replicas store a write.

4 Single Points of Failure

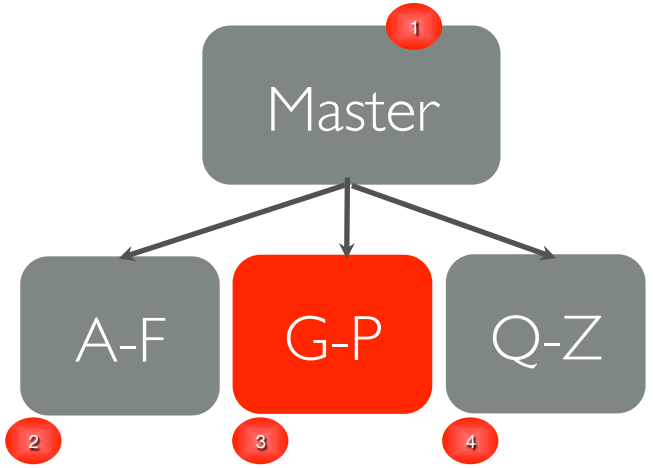


With sharded solutions like Oracle or MongoDB or CouchDB, each shard stores a primary copy of data. (Maybe those shards are replicated, maybe not.)

Consistent Hashing, Superior to Sharding

Riak uses consistent hashing to cluster. There is no master and no primary replica of data. This technology is what we used at Akamai and what Amazon now uses for their shopping platform.

Even though node 3 fails, there are two copies still stored and any node will accept a write for the failed node.

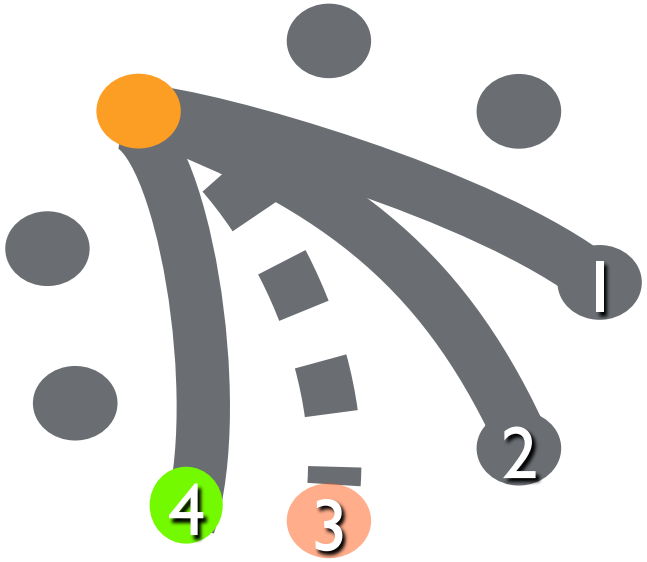


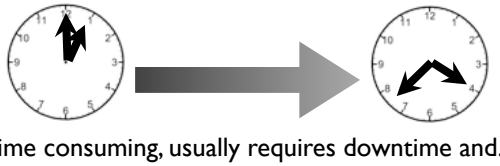
4 SPOF

With sharding, if a node fails, the data range stored on that shard is no longer available.

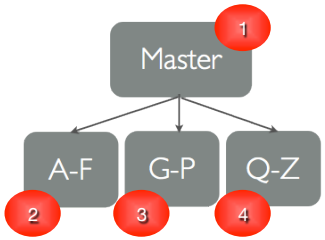
Consistent Hashing Makes Scaling Easy

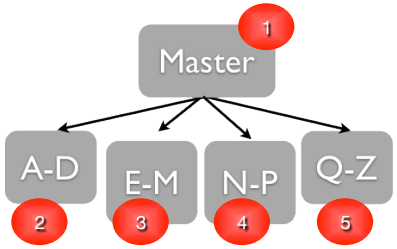
Unlike with Riak, databases that use sharding also incur an operational cost. You cannot dynamically scale a sharded database. You must shut down your application and re-shard.





Time consuming, usually requires downtime and...





4 SPOF vs. **5** SPOF

...you end up less fault-tolerant!

With Riak, you can add another node at will. You can remove a node just as easily.

```
< riak join #adds a node
< riak leave #removes a node
```

Sharded databases require an intensive operational intervention to shut down the database and re-distribute data.

And the mean time between machine failures is no shorter with more shards. **YOU ARE LESS FAULT TOLERANT.**

Write Availability and Scaling

- **No Master** DB means no single point of failure
- **Fast horizontal scaling** using commodity servers
  - Add nodes dynamically, as well as scale down at will to save money
  - No downtime to deploy more servers
  - Add servers for capacity, for redundancy, or both
- **User-controlled replication** - Data replicated around ring according to developer specifications thanks to tunable CAP controls
- **Scale Write Capacity** - Increase write capacity by simply adding servers
- **No row-locking** or resource contention for write-intensive applications

Simplified Operations

- **Simple scaling up/down** – two word commands to add servers
  - a key benefit of masterless system/hinted handoff/gossip
- **Fungible Server set-up** – no master anything
- **Flexible Schema** – no downtime for schema changes
- **Downtime isn't Downtime** – take any host out of action (for repair, accident, or upgrade) and just keep on going!

Simplified Operations

- **Simple scaling up/down** – two word commands to add servers
  - a key benefit of masterless system/hinted handoff/gossip
- **Fungible Server set-up** – no master anything
- **Flexible Schema** – no downtime for schema changes
- **Downtime isn't Downtime** – take any host out of action (for repair, accident, or upgrade) and just keep on going!

Interfaces and Query Methods

- **Ruby, Java, PHP, python, HTTP, Protocol Buffers, JSON, Erlang interfaces**
- **HTTP – main interface**
  - Riak uses simple, standard HTTP methods to transfer data between client and server.

→ If you can process JSON and issue an HTTP request, you can easily use Riak.

- **Riak stores data in "Buckets",**
  - namespaces for documents with loose schemas.
  - new buckets (and more importantly new schemas) are created on demand
  - add new data types/alter structure for Riak without taking your app out of production
- **Query via SOLR Search MapReduce and Linked Data** – built in map/reduce function
  - designed to find and connect disparate documents containing links to one another, giving developers the ability to query large sets of data across entire clusters.

Interfaces and Query Methods

- **Ruby, Java, PHP, python, HTTP, Protocol Buffers, JSON, Erlang interfaces**
- **HTTP – main interface**
  - Riak uses simple, standard HTTP methods to transfer data between client and server.

→ If you can process JSON and issue an HTTP request, you can easily use Riak.

- **Riak stores data in "Buckets",**
  - namespaces for documents with loose schemas.
  - new buckets (and more importantly new schemas) are created on demand
  - add new data types/alter structure for Riak without taking your app out of production
- **Query via SOLR Search MapReduce and Linked Data** – built in map/reduce function
  - designed to find and connect disparate documents containing links to one another, giving developers the ability to query large sets of data across entire clusters.