Riak KV
Introduction

- What is Riak?
- What is the problem?
  
  low latency at scaling
Replication and Partitions

- Replication High availability
- What is the trade-off?

- Partitioning
- How we divide a set of keys onto separate physical servers
Naive Hashing
Naive Hashing
Consistent Hashing

- Partition remains constant
- Key always map to the same partition
- Nodes own partitions
- Partitions contain keys
Consistent Hashing

- Evenly divides keyspace
- Logical partitioning separated from physical partitioning
- Uniform hash gives uniform distribution
Consistent Hashing
The Ring
The Ring
The Ring
Riak takes shape - The Ring

$2^{160}$

$2^{160}/2$

$2^{160}/4$

hash("artist", "REM")

node 0
node 1
node 2
node 3
Dynamo Paper Influences

- \( N = \) The number of replicas
- \( R = \) The number of Replicas needed for a successful read
- \( W = \) The number of replicas needed for a successful write
R value

get(<<"artist">>, <<"REM">>, R=2)

{ok, Object}
W value

put("artist", "REM", W=2)

(N=3)
N=10, R/W=2

get/put("artist", "REM", R/W=2)

{ok, Object}
Riak Components

- Riak is a key/value database

`hashtable["5124"] = "Bob"
Retrieving Bob is as easy as going to his house
`bob = hashtable["5124"]`

- Let’s say poor old Bob dies, and Clair moves into this
  `hashtable["5124"] = "Clair"

- Successive requests for 5124 will now return Claire
Riak Components

Buckets

- Like street. There could be another 5124
- Logical namespaces so that identical
  main[“5112”] = “Alice”
  bagshot[“5112”] = “Gas”
Riak Components

Types

- Groups of bucket with a similar set of properties
  
  ```
  places["main"]["5122"] = "Alice"
  places["bagshot"]["5122"] = "Gas"
  ```

- The benefit here is that a group of distinct buckets can share properties.

```python
places.props = {
    "search_index" : "anyplace"
}
```

Why???
Eventually Consistent

- Reads can be stale
- Concurrent writes can cause siblings
- Eventually Values converges
Eventually Healing

- When focusing on AP
- Read Repair
- Active Anti - Entropy
AAE - MERKLE TREES
AAE - MERKLE TREES
AAE - MERKLE TREES

HASH OF HASHES IN SEGMENT
AAE - MERKLE TREES

HASH OF HASHES OF HASHES OF HASHES :)
AAE - EXCHANGE
Vector Clocks

- Logical rather than temporal
- How to compare between elements
Riak Types

- Help to resolve conflicts
- 4 types - flag, set, map and counter

Type: set
Bucket: cart
Key: ponies4evr
Vclock: {Node_B : 1}
Value: {“MYPFIM-S2-DVD”}
Basic Operations

- PUT /types/<type>/buckets/<bucket>/keys/<key>
- GET /types/<type>/buckets/<bucket>/keys/<key>
- DELETE /types/<type>/buckets/<bucket>/keys/<key>
DATABASE

- Key/Value model
- Basic secondary index support
- Map/Reduce
- Search
YOKOZUNA

- Integration of riak and Solr
- Index Riak Data with Solr
- Distribute solr with riak
- Together do what each alone cannot