Riak Core: Dynamo Building Blocks

Andy Gross (@argv0)
Basho Technologies

QCon SF 2010

About Me

- Basho Technologies - Riak, Riak Search, Webmachine, Erlang open source
- Mochi Media - Ad network written in Erlang
- Apple - distributed compilers, filesystems
- Akamai - large distributed systems, worlds first CDN

This Talk

- Background and design philosophy
- Overview of Riak Features
- Riak Core Architecture
- Future Directions

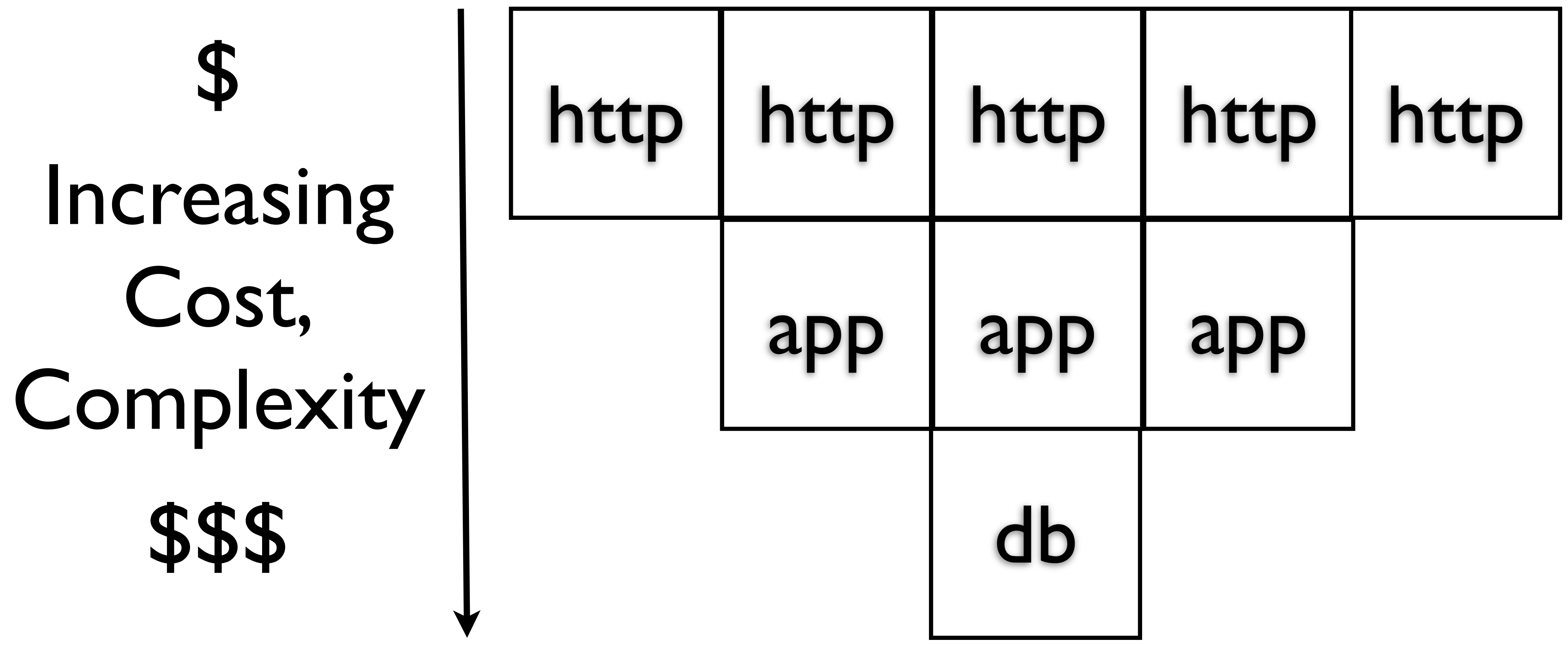
Front Matter

- Dynamo (and NoSQL) are nothing new
- Much of Dynamo was invented > 10 years ago
- Dynamo chooses AP of CAP
- This talk will focus on properties of Dynamo-inspired systems (Riak, Cassandra, Voldemort)

Why Now?

- Changing face of web applications
- Explosion of data beyond our means to store it
- Higher uptime demands
- Cloud computing requires horizontal scaling
- Velocity, volume, variety of data

Scaling Traditional Web Architectures



When to choose Dynamo-style systems

- Cost of scaling traditional DBs becomes prohibitive
- Availability is a primary concern
- You can cope with eventual consistency (not as scary as it seems)

Eventual Consistency

- The real world is eventually consistent and works (mostly) fine
- “Eventual” doesn’t mean minutes, days, or even seconds in non-failure cases
- DNS, HTTP with Expires: header
- How you model the real world matters!

What Is Riak?

- Distributed Key-Value Store, inspired by Amazon's Dynamo
- Eventually consistent, horizontally scalable
- Written in Erlang (and some C)
- Novel features (links, MapReduce)
- HTTP and binary interfaces

Basic Usage: PUT

```
PUT /riak/qcon/foo HTTP/1.1
Content-Type: text/plain
Content-Length: 3

bar
HTTP/1.1 204 No Content
Vary: Accept-Encoding
Server: MochiWeb/1.1 WebMachine/1.7.2 (participate in the frantic)
Date: Tue, 05 Oct 2010 09:43:52 GMT
Content-Type: text/plain
Content-Length: 0
```

Basic Usage: GET

```
GET /riak/qcon/foo HTTP/1.1

HTTP/1.1 200 OK
X-Riak-Vclock: a85hYGBgzGDKBVIsbBXOTzOYEhnzWBki8uWP8WUBAA==
Vary: Accept-Encoding
Server: MochiWeb/1.1 WebMachine/1.7.2 (participate in the frantic)
Link: </riak/qcon>; rel="up"
Last-Modified: Tue, 05 Oct 2010 09:43:52 GMT
ETag: 1vSkKtrE4Fg8VDkke9aL5J
Date: Tue, 05 Oct 2010 09:46:53 GMT
Content-Type: text/plain
Content-Length: 3

bar
```

Basic Usage: POST

```
POST /riak/qcon HTTP/1.1
Content-Type: text/plain
Content-Length: 3

bar
HTTP/1.1 201 Created
Vary: Accept-Encoding
Server: MochiWeb/1.1 WebMachine/1.7.2 (participate in the frantic)
Location: /riak/qcon/NRMNPDGYoW3LPOKmROLqz6o4KO
Date: Tue, 05 Oct 2010 09:48:49 GMT
Content-Type: application/json
Content-Length: 0
```

Basic Usage: DELETE

```
DELETE /riak/qcon/foo HTTP/1.1

HTTP/1.1 204 No Content
Vary: Accept-Encoding
Server: MochiWeb/1.1 WebMachine/1.7.2 (participate in the frantic)
Date: Tue, 05 Oct 2010 09:49:34 GMT
Content-Type: text/html
Content-Length: 0
```

High-Level Dynamo

- Gossip Protocol: membership, partition assignment
- Consistent Hashing: division of labor
- Vector clocks: versioning, conflict resolution
- Read Repair: anti-entropy
- Hinted Handoff: failure masking, data migration

basho

riak

High-Level Dynamo

- Decentralized (no master nodes, no SPOF)
- Homogeneous (all nodes can do anything)
- No reliance on physical time
- No global state

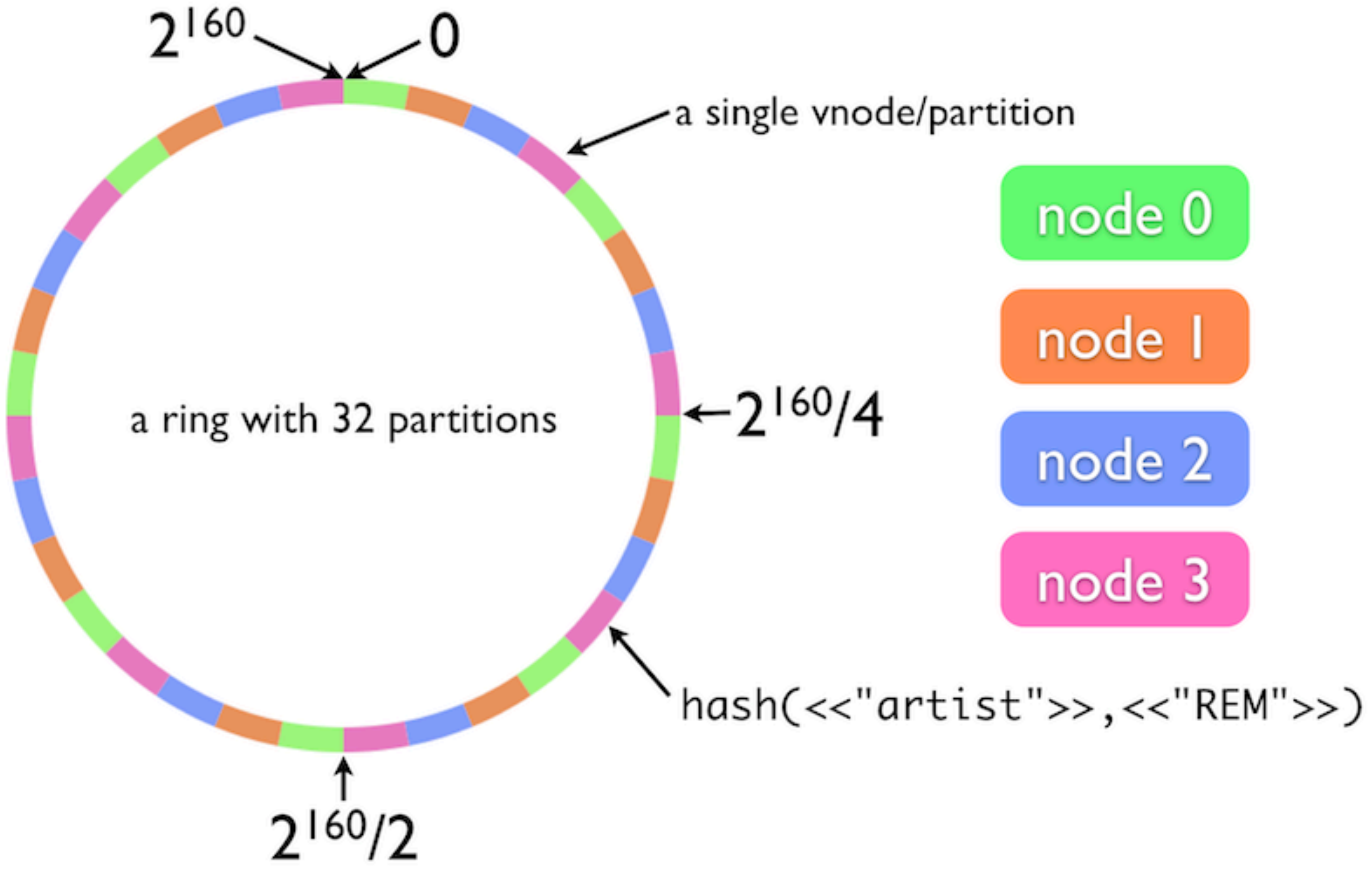
Gossip Protocol

- Handles cluster membership, partition assignment
- Works just how it sounds:
  - Change local state, send to random peer
  - When receiving gossip, merge with local state, send to random peer
- Converges quickly, but *not immediately.*

Consistent Hashing

- Modulus-based hashing: great until adding/ removing machines causes complete reshuffle.
- Consistent hashing: optimally minimal resource reassignment when # buckets changes
- Any node can calculate replica locations using gossiped partition map

Consistent Hashing



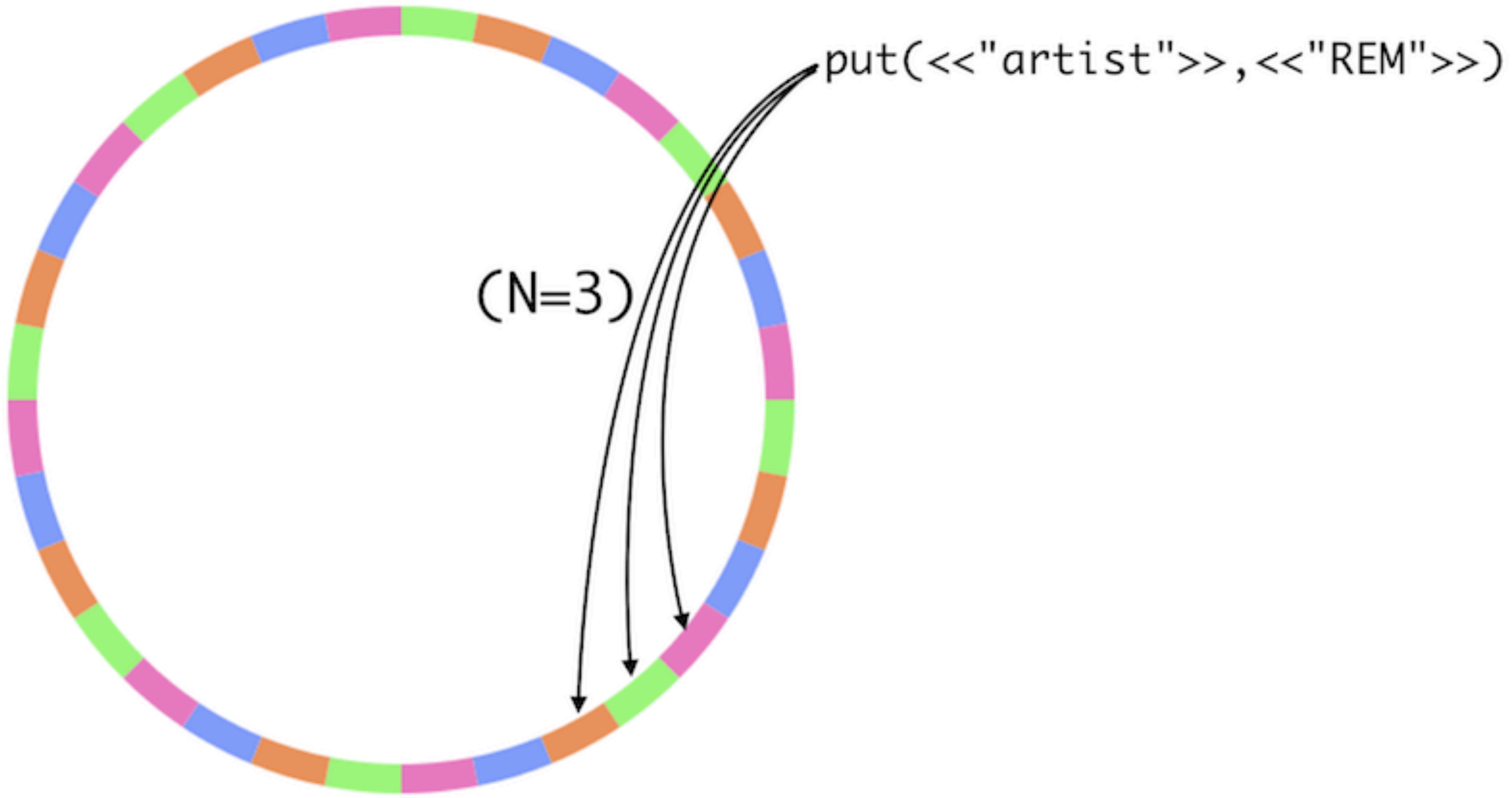
basho

riak

N,R,W Values

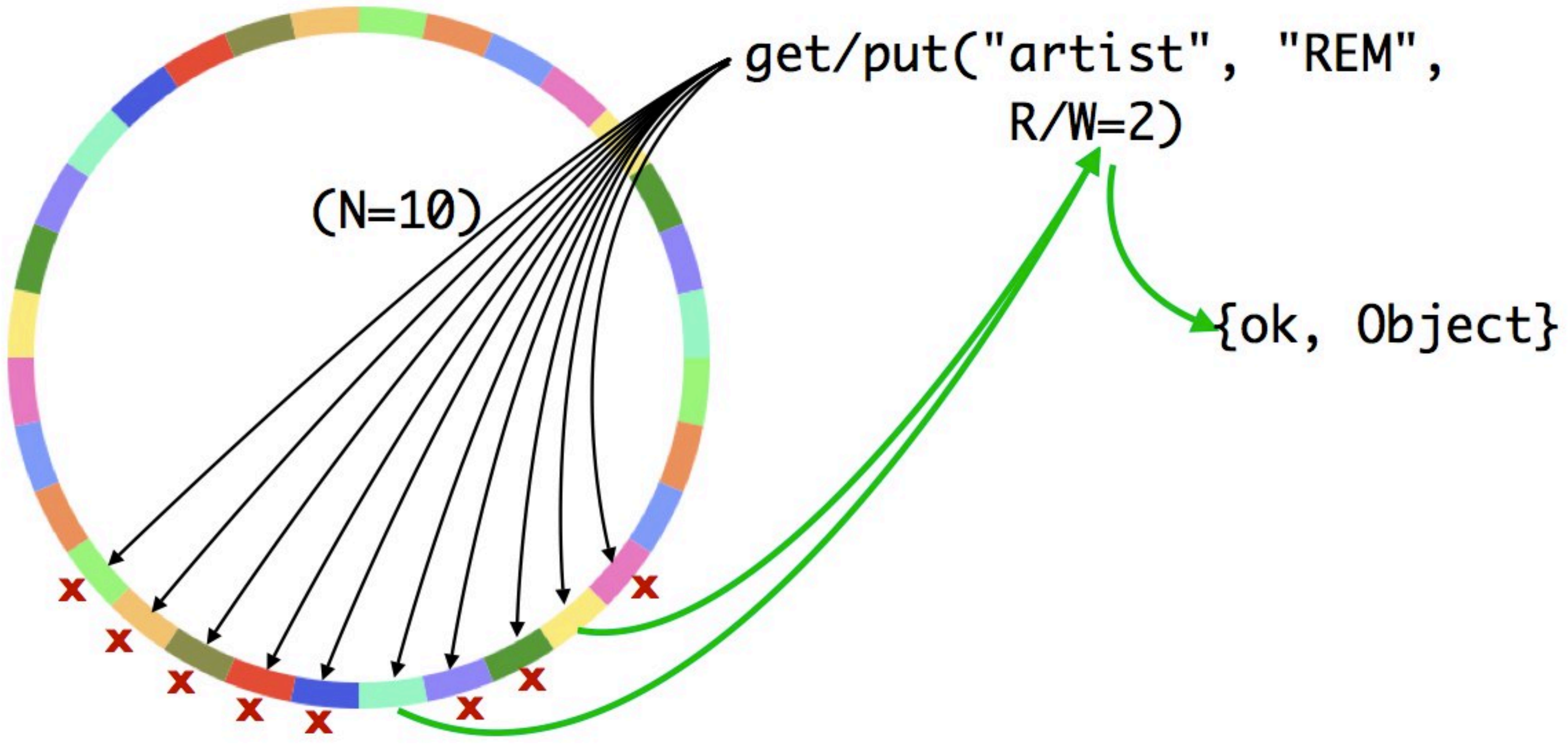
- N = number of replicas to store (on distinct nodes)
- R = number of replica responses needed for a successful read (specified per-request)
- W = number of replica responses needed for a successful write (specified per-request)

N,R,W Values



basho riak

N,R,W Values



Hinted Handoff

- Any node can handle data for any logical partition (virtual node)
- Virtual nodes continually try to reach "home"
- When machines re-join, data is handed off
- Used for both failure recovery and node addition/removal

Read Repair

- When reading values, opportunistically repair stale data
- “Stale” is determined by vector clock comparisons
- Occurs asynchronously

Adding/Removing Nodes

- "riak start && riak-admin join"
- Riak scales *down* to 1 node and up to hundreds or thousands.
- Developers often run many nodes on a single laptop
- Data is re-distributed using hinted handoff

Vector Clocks

- Reasoning about time and causality is *fundamentally hard.*
- Ask a physicist!
- Integer timestamps an insufficient model of time - don't capture causality
- Vector clocks provide a *happens-before* relationship between two events

Vector Clocks

- Simple data structure: [(ActorID,Counter)]
- Objects keep a vector clock in metadata, actors update their entry when making changes
- ActorID needs to reflect potential concurrency - early Riak used server names - too coarse!

Link Walking

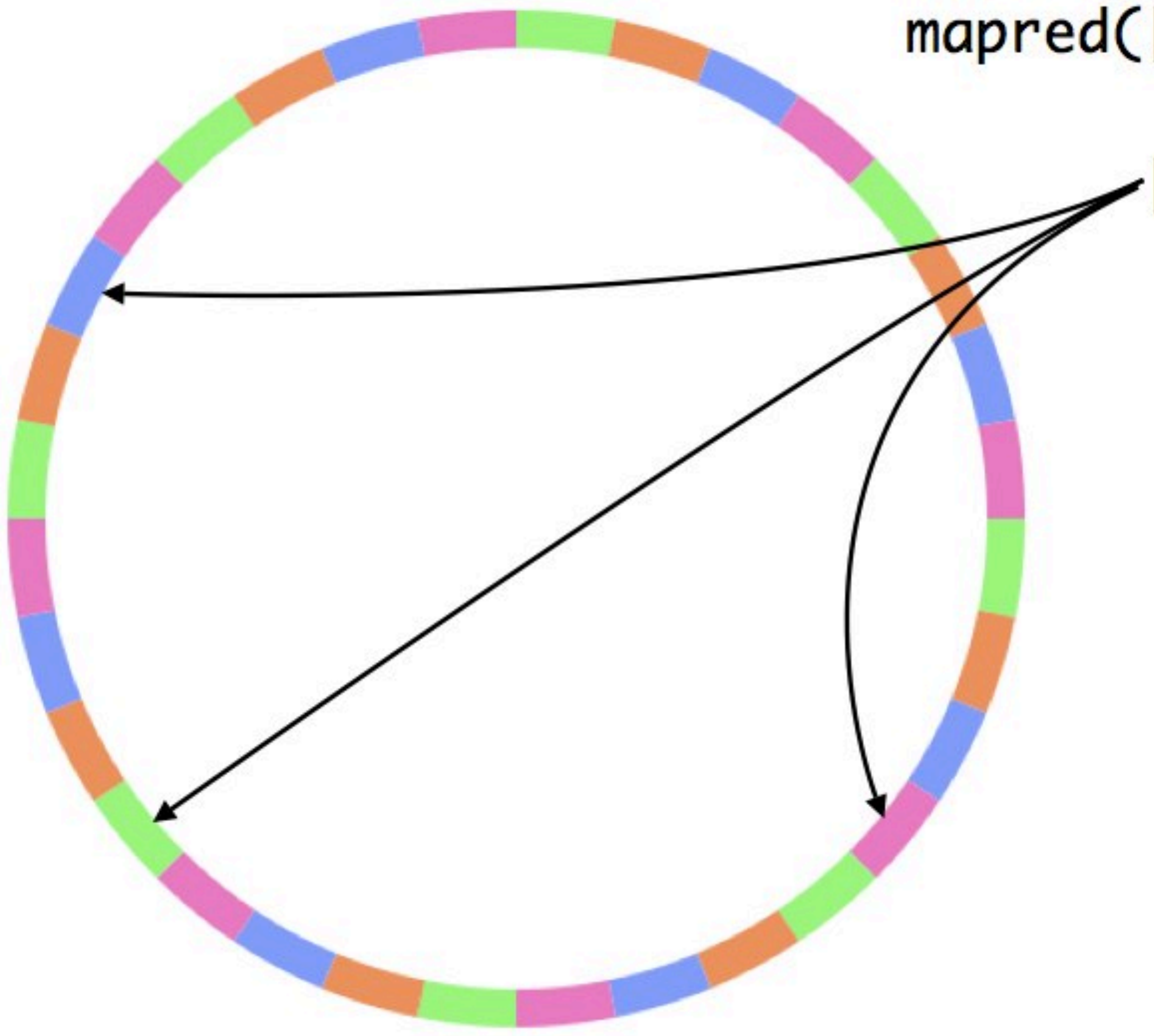
- Lightweight, flexible object relationships
- Works like the web
- Structure: (Bucket, Key, Tag)
- http://host/riak/conferences/qcon/talks,_,nosql/

“Fetch the “qcon” object from the “conferences” bucket and give me all linked “talk” objects tagged “nosql”

Map/Reduce

- M/R functions can be implemented in Erlang or Javascript
- Scope: pre-defined set of keys or entire buckets
- Functions are shipped to the data
- Phases can be arbitrarily chained

Map/Reduce



```
mapred([{<<"artist">>,<<"REM">>},
        {<<"artist">>,...},...],
       [{map,
         {modfun,artist,member_count},
         none,false},
        {reduce,
         {qfun,fun(L,_,_) ->
                    lists:unique(L)
               end},
         none, true}]).
```

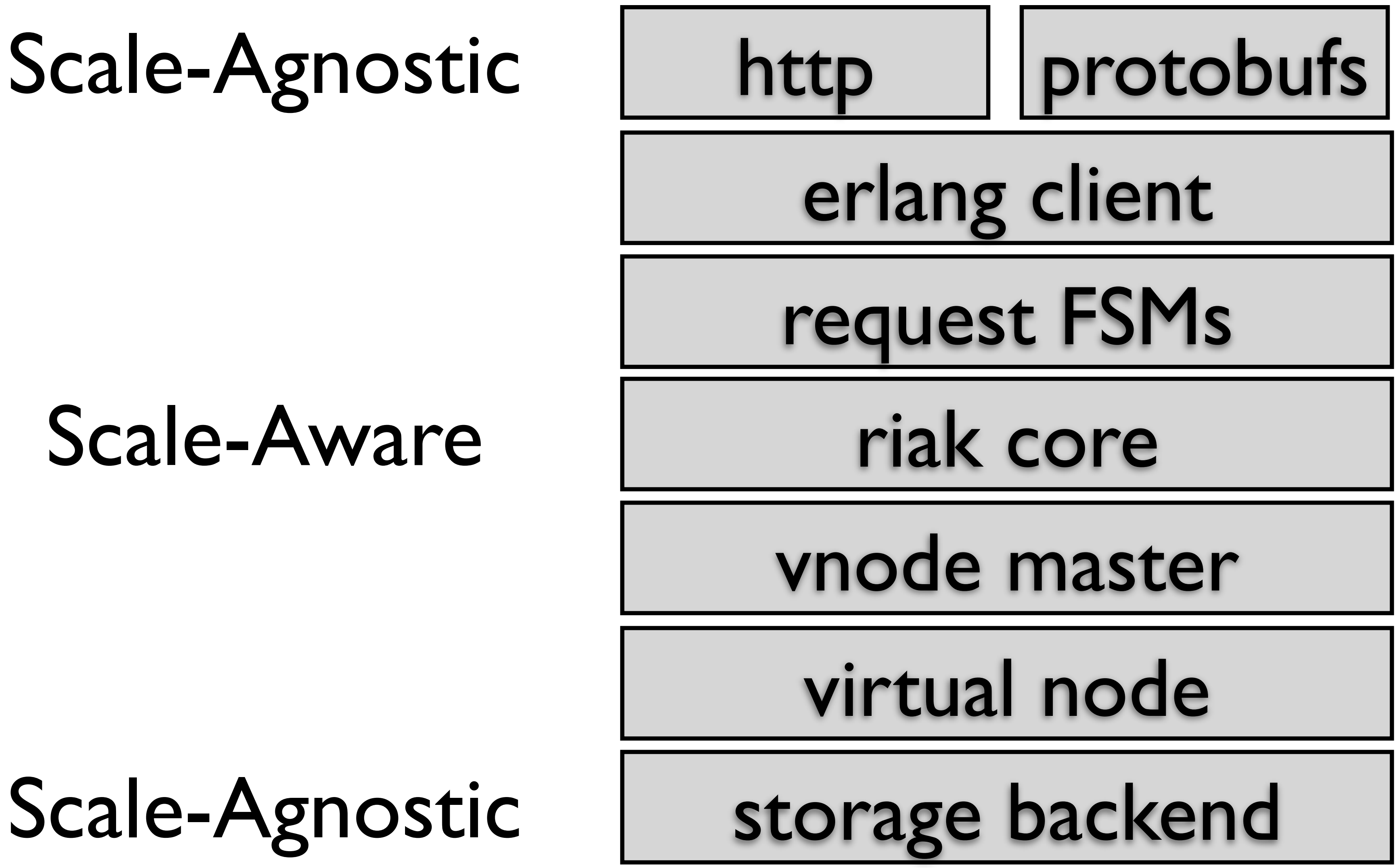
Commit Hooks

- Similar to triggers in traditional databases
- Pre-commit hooks: Executed synchronously, can fail updates, modify data
- Post-commit hooks: Executed asynchronously, used for integration with other systems

Harvesting A Framework

- We noticed that Riak code fell into one of two categories
    - Code specific to K/V storage
    - “generic” distributed systems code
- So we split Riak into K/V and Core
- Useful outside of Riak

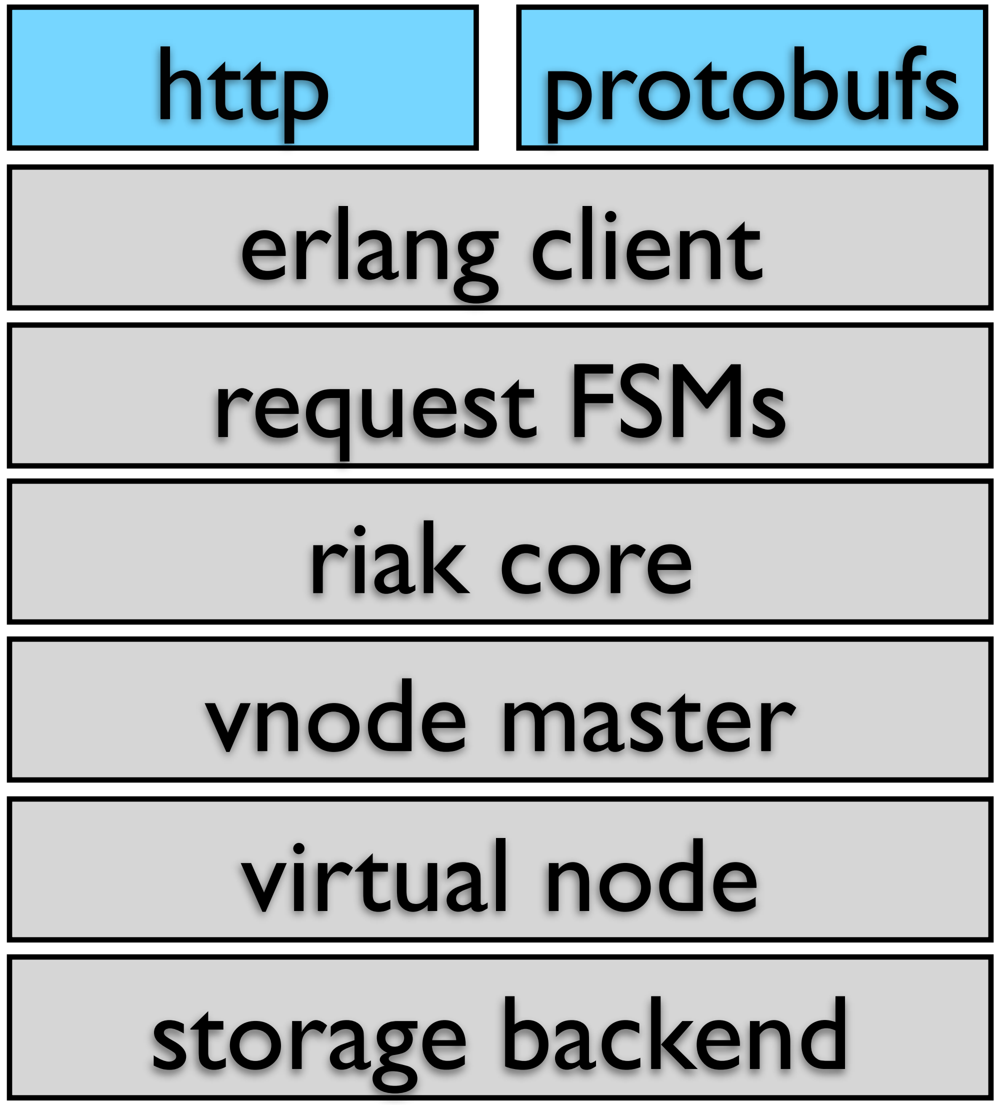
Riak Core: The Stack



Client Interfaces

<u>HTTP</u>
Rich semantics
Cacheable
Easy Integration

<u>Protocol Buffers</u>
Fast
Compact

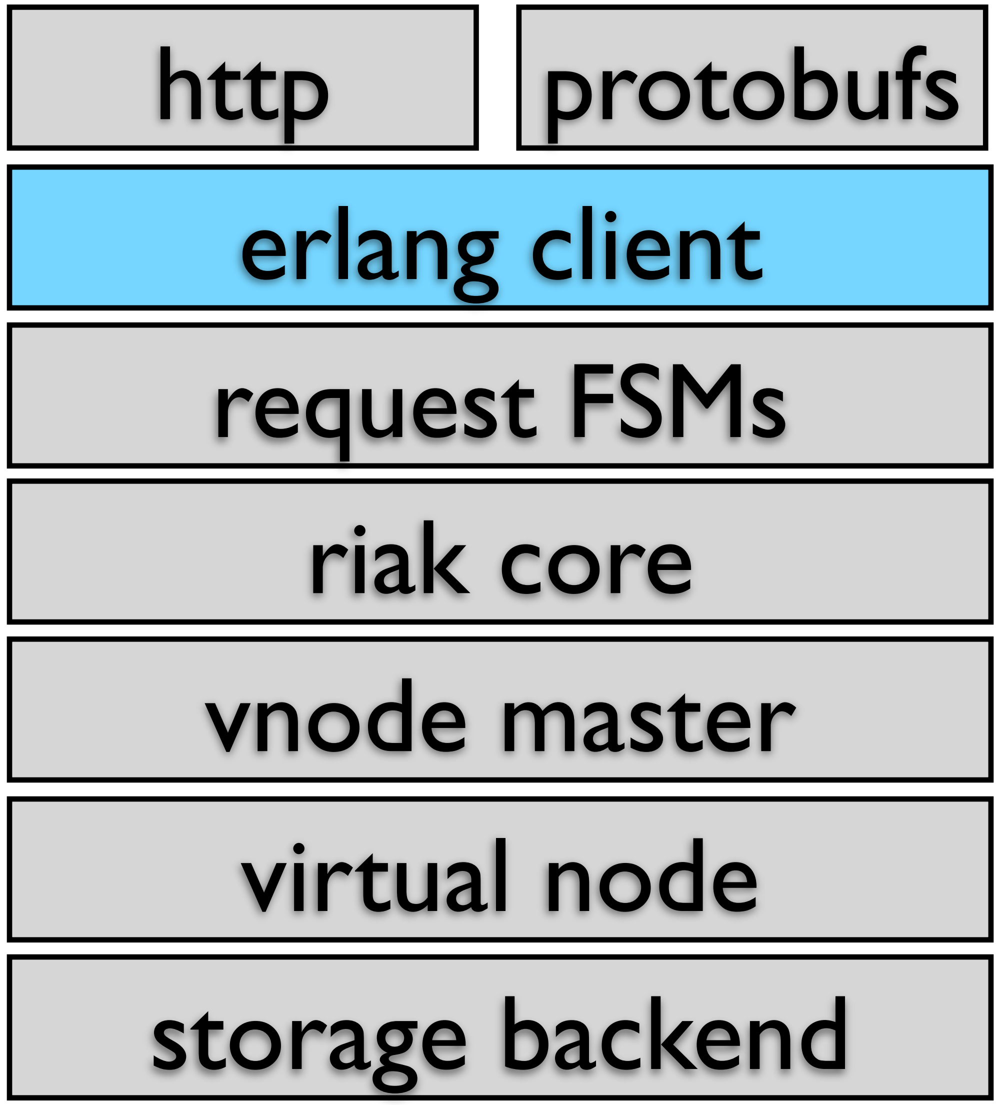


basho

riak

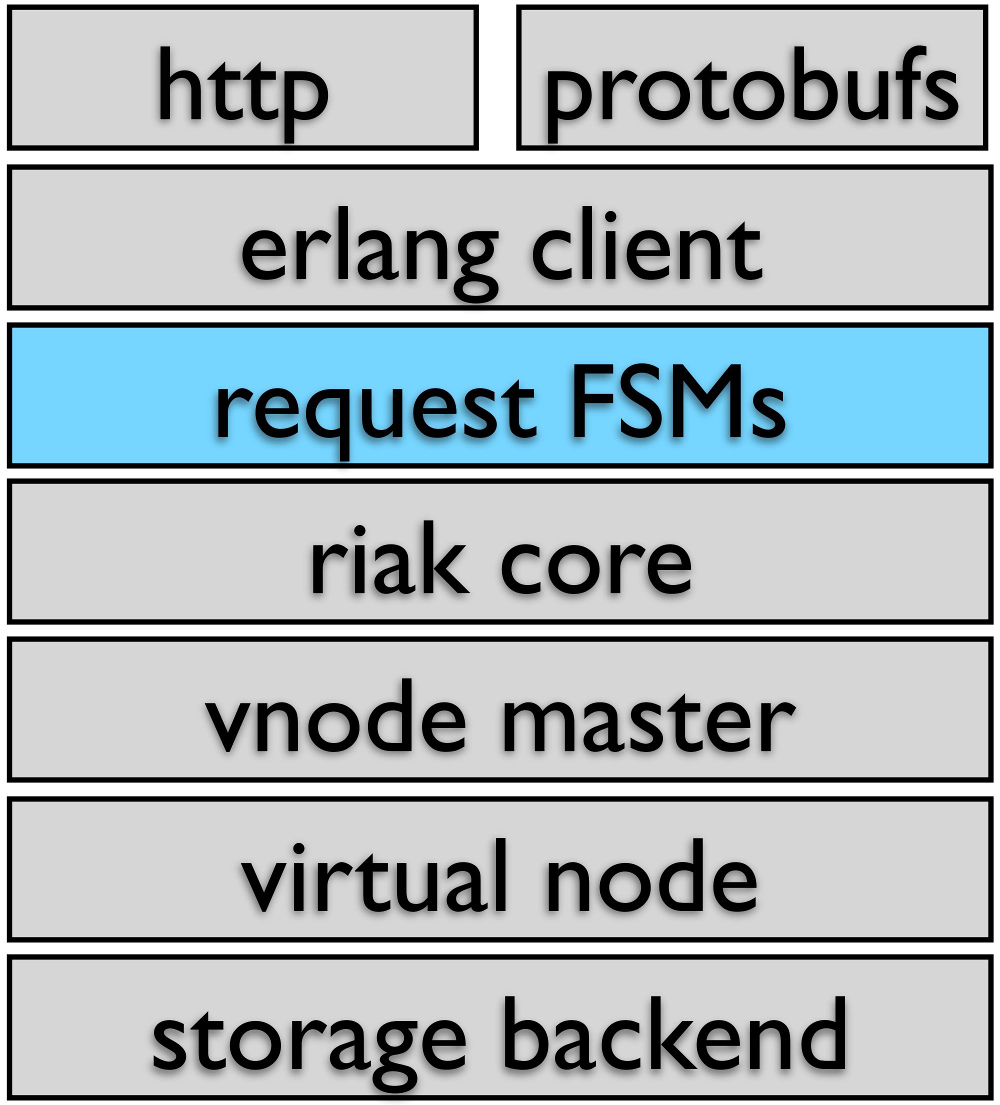
Client Implementation

All front-end client interfaces implemented against the Erlang low-level client API.



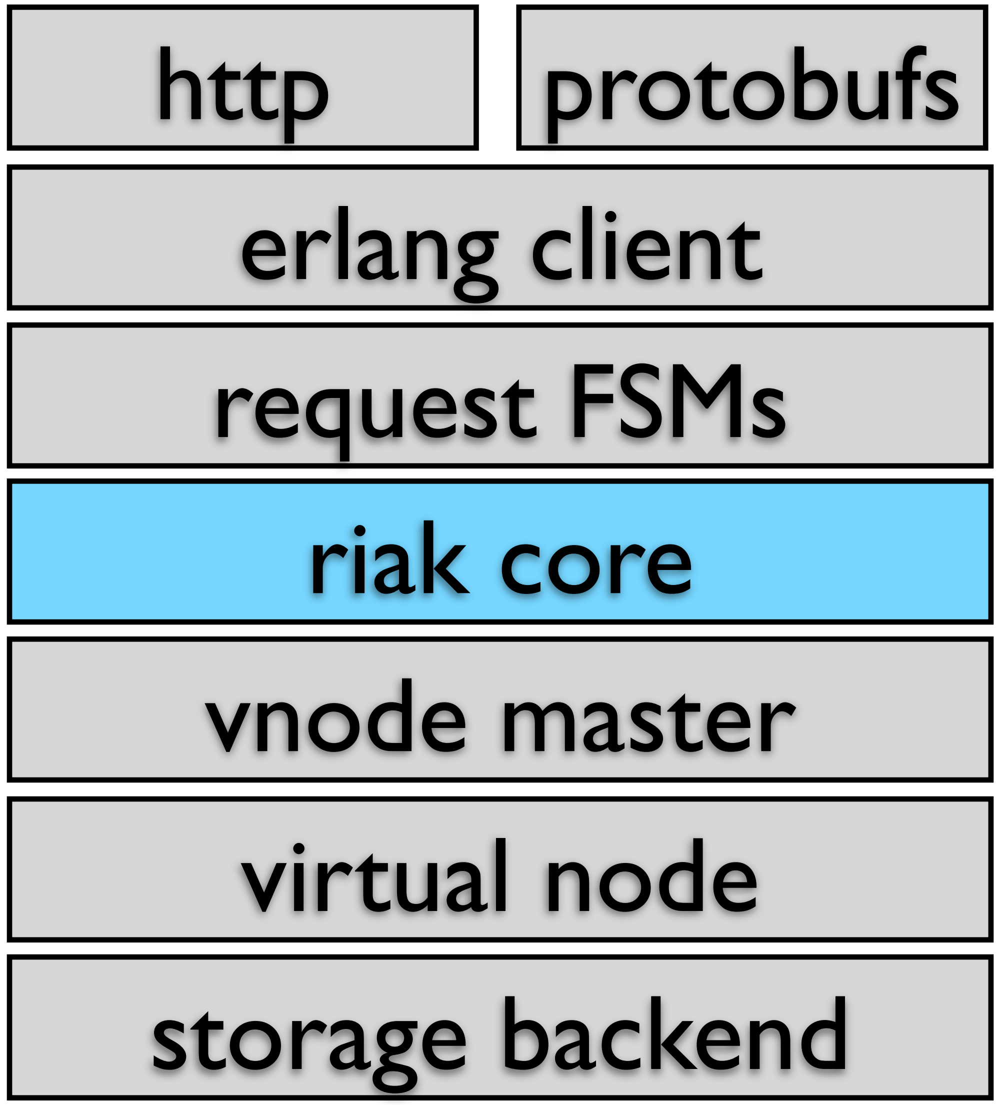
Modeling Requests

Requests are modeled as finite state machines, each in its own Erlang process



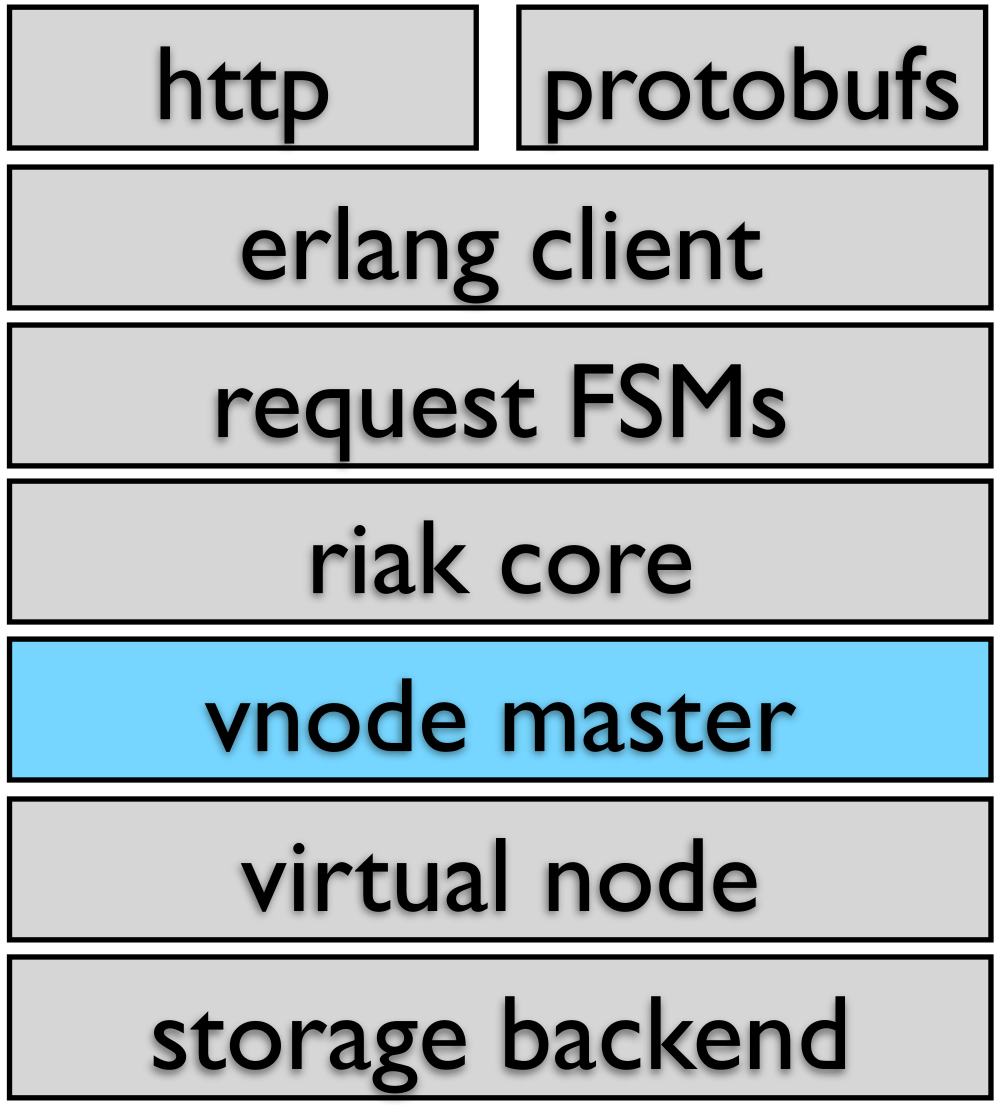
Riak Core: The Hard Stuff

Vector Clocks
Consistent Hashing
Merkle Trees
Virtual Node
Handoff
Failure Detection
Gossip



Concurrency and Bookkeeping

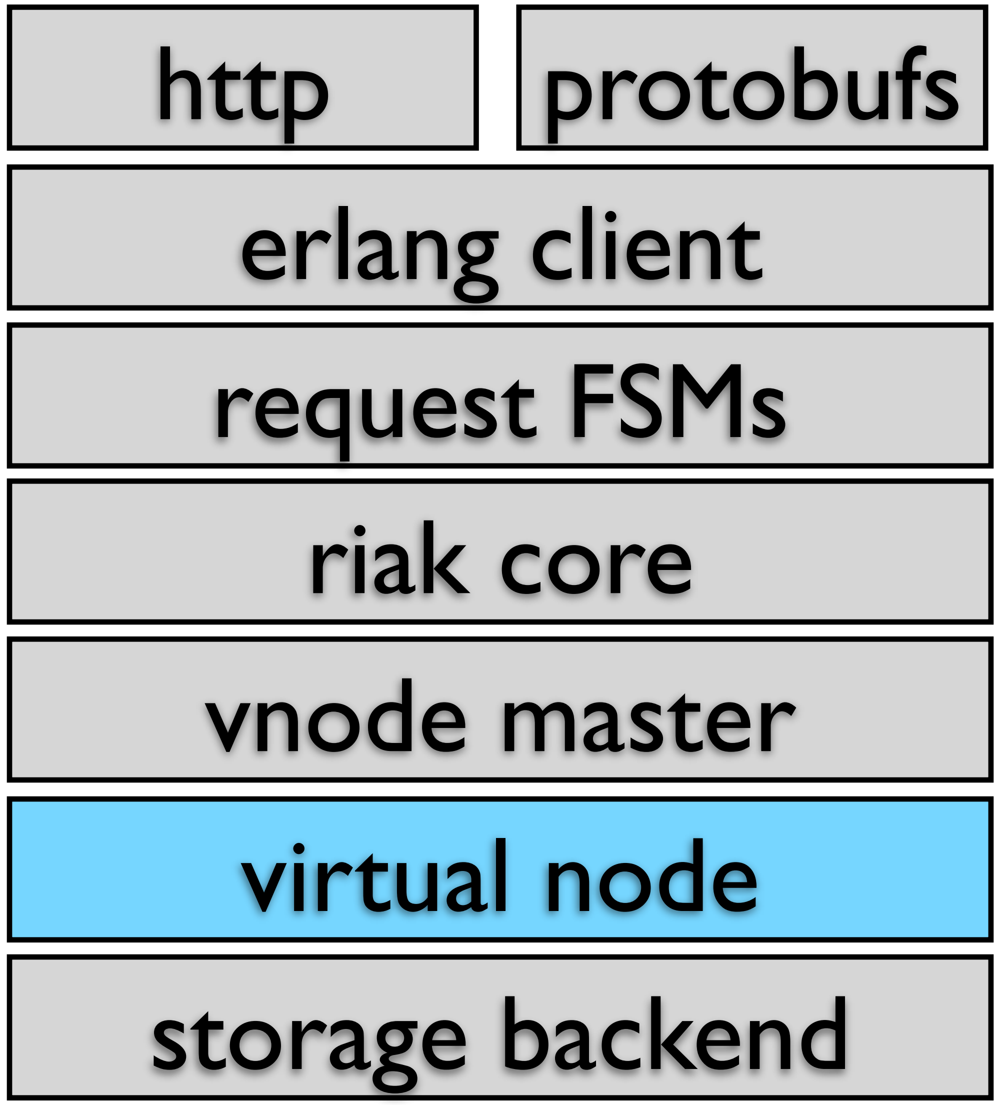
Request dispatching
Book-keeping



Virtual Nodes

disposable, per-partition actor for access to local data

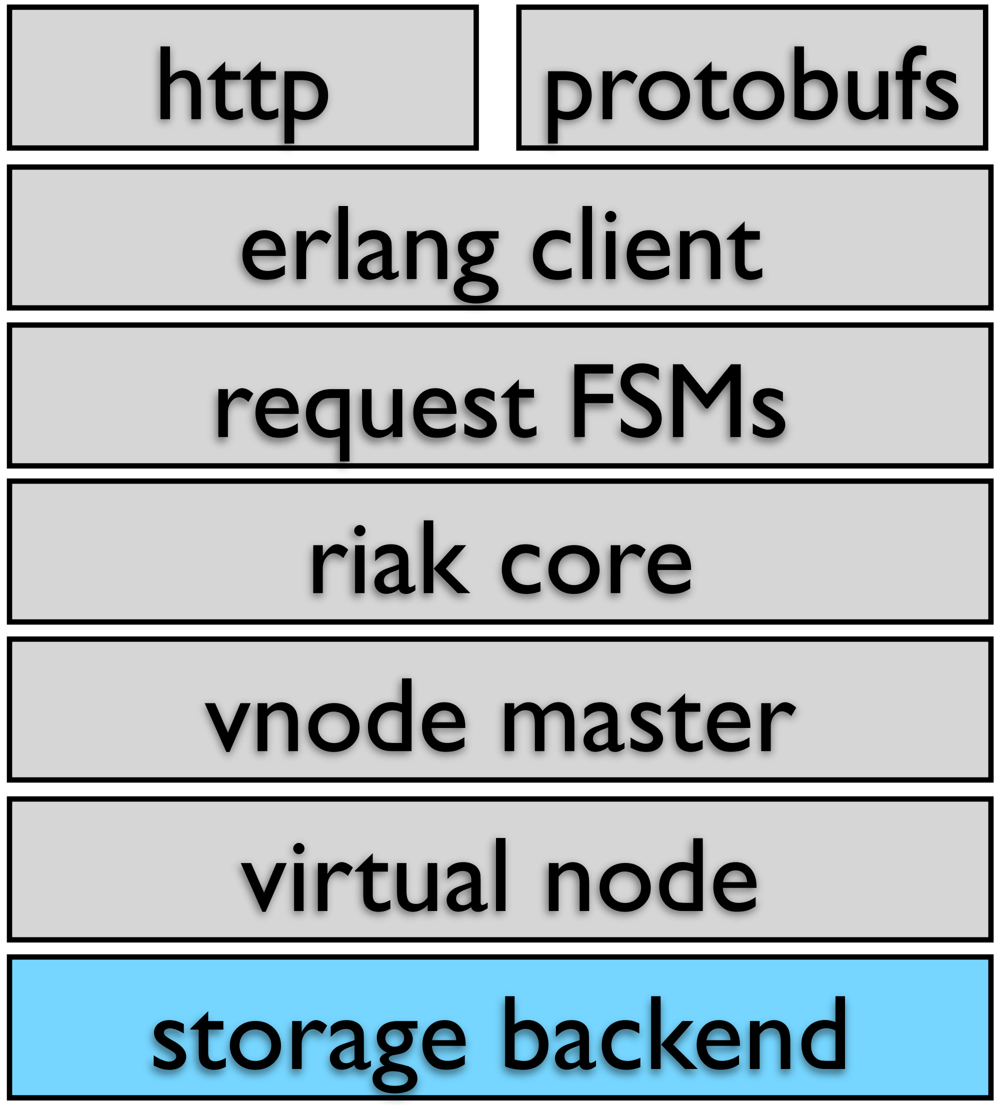
node-local abstraction for storage



Storage Backends

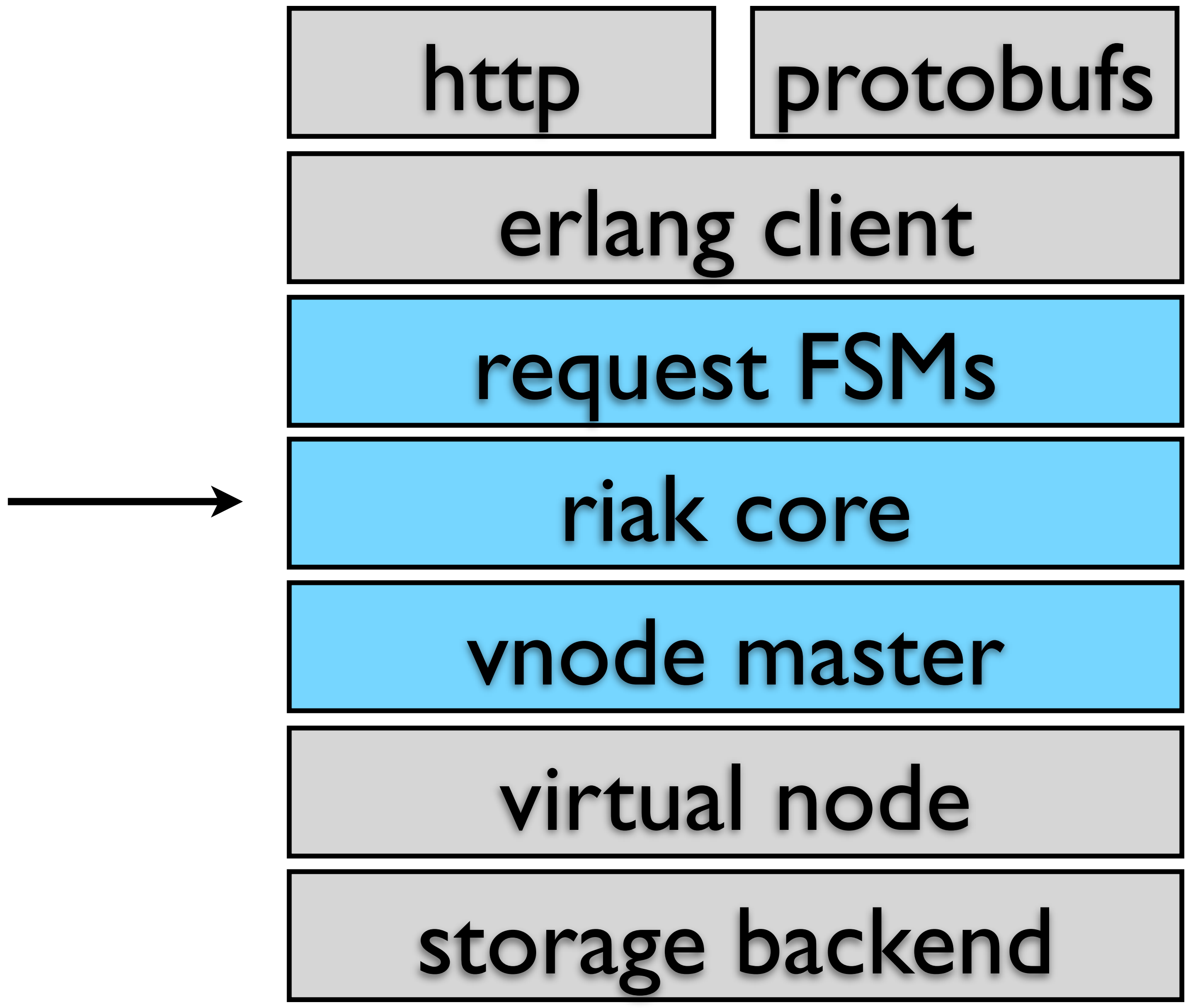
Conform to a common interface, defined by clients and virtual nodes

Pluggable, interchangeable

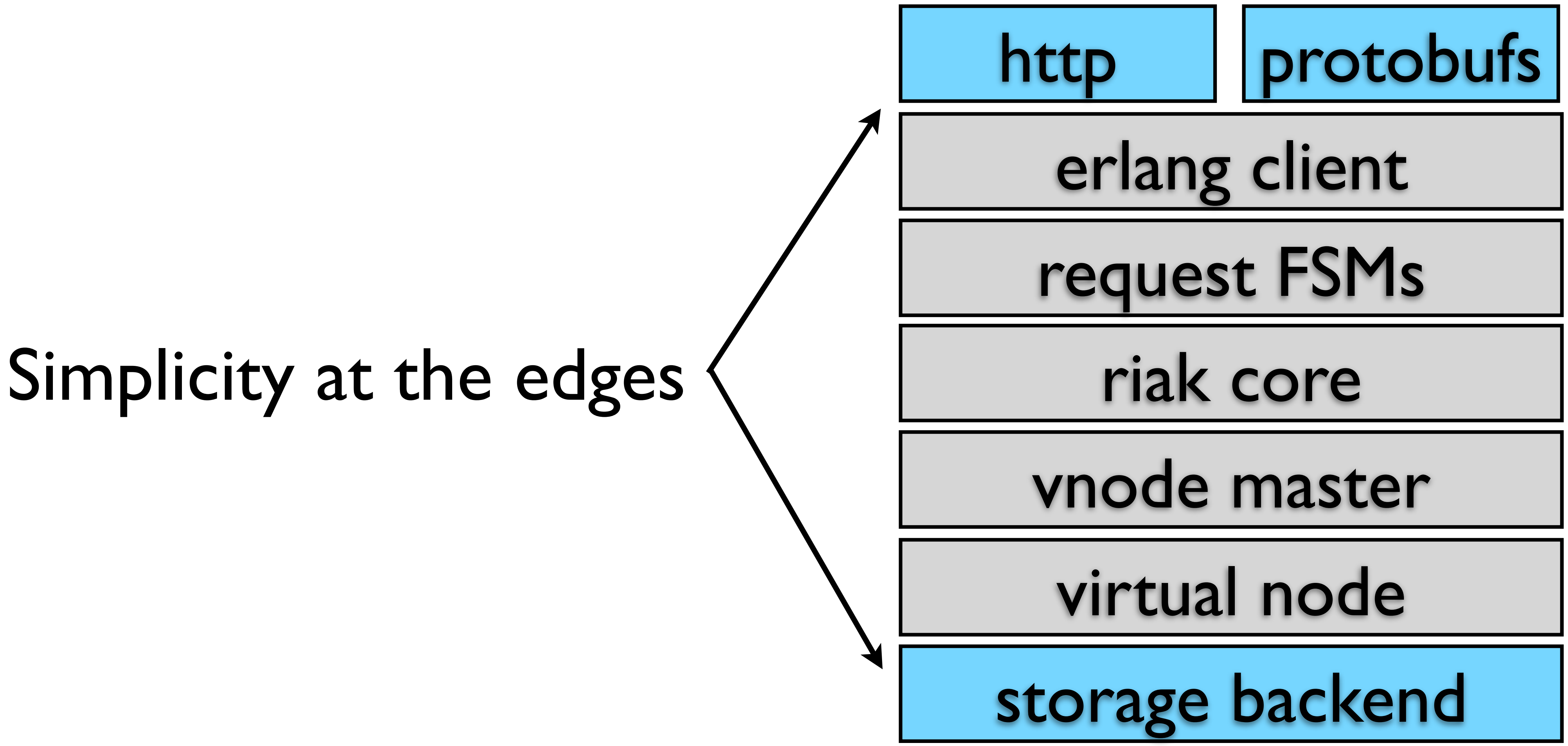


Riak Core

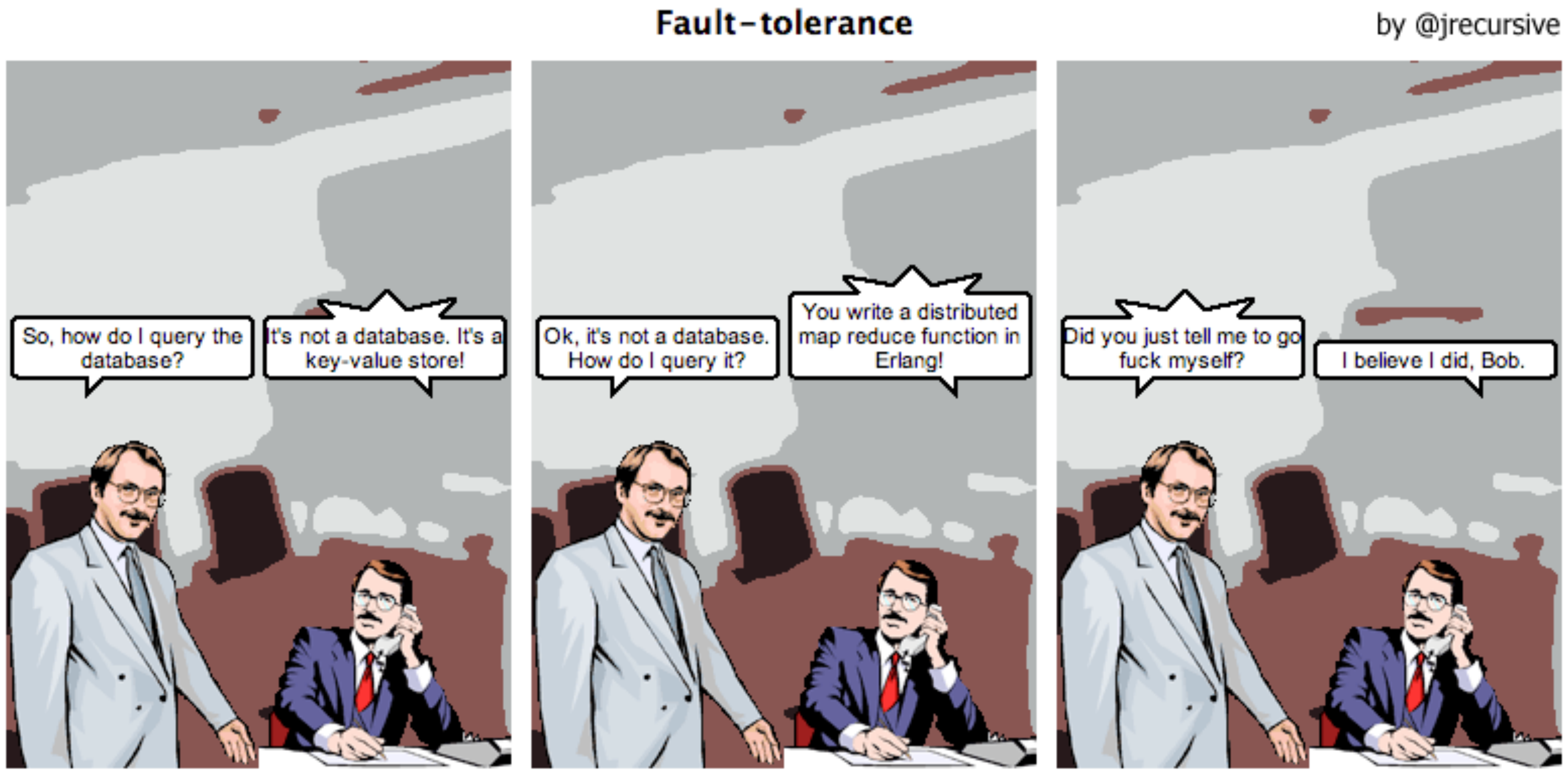
Complexity in the middle



Riak Core


Simplicity at the edges
http
protobufs
erlang client
request FSMs
riak core
vnode master
virtual node
storage backend

Riak Search


Fault-tolerance
by @jrecursive
So, how do I query the database?
It's not a database. It's a key-value store!
Ok, it's not a database. How do I query it?
You write a distributed map reduce function in Erlang!
Did you just tell me to go fuck myself?
I believe I did, Bob.

Little known fact: A Riak engineer drew this cartoon

The key/value access model doesn't satisfy all use cases

Riak Search

- Sometimes key-value isn’t enough
- Search data with Lucene query syntax
- Built on Riak Core
- Stores documents in Riak-KV
- New Map/Reduce type: Search Phase

Future Directions

- Analytical/column store?
- Graph Database?
- Continued work on Riak Core
- Make distributed systems experimentation easier!

Thank You!

@argv0

@basho/team

http://basho.com

http://github.com/basho