Riak Use Cases: Dissecting the Solutions to Hard Problems

Andy Gross <@argv0>
Chief Architect
Basho Technologies

Riak

- Dynamo-inspired key value database
  - with full text search, mapreduce, secondary indices, link traversal, commit hooks, HTTP and binary interfaces, pluggable backends
- Written in Erlang and C/C++
- Open Source, Apache 2 licensed
- Enterprise features (multi-datacenter replication) and support available from Basho

Choosing a NoSQL Database

- At small scale, everything works.
- NoSQL DBs trade off traditional features to better support new and emerging use cases
- Knowledge of the underlying system is essential
- A lot of NoSQL marketing is bullshit

Tradeoffs

- If you're evaluating Mongo vs. Riak, or CouchDB vs. Cassandra, you don't understand your problem
- By choosing Riak, you've already made tradeoffs:
  - Consistency for availability in failure scenarios
  - A rich data/query model for a simple, scalable one
  - A mature technology for a young one

Distributed Systems: Desirable Properties

- Highly Available
- Low Latency
- Scalable
- Fault Tolerant
- Ops-Friendly
- Predictable

1000s of Deployments

User/Metadata Store
Comcast

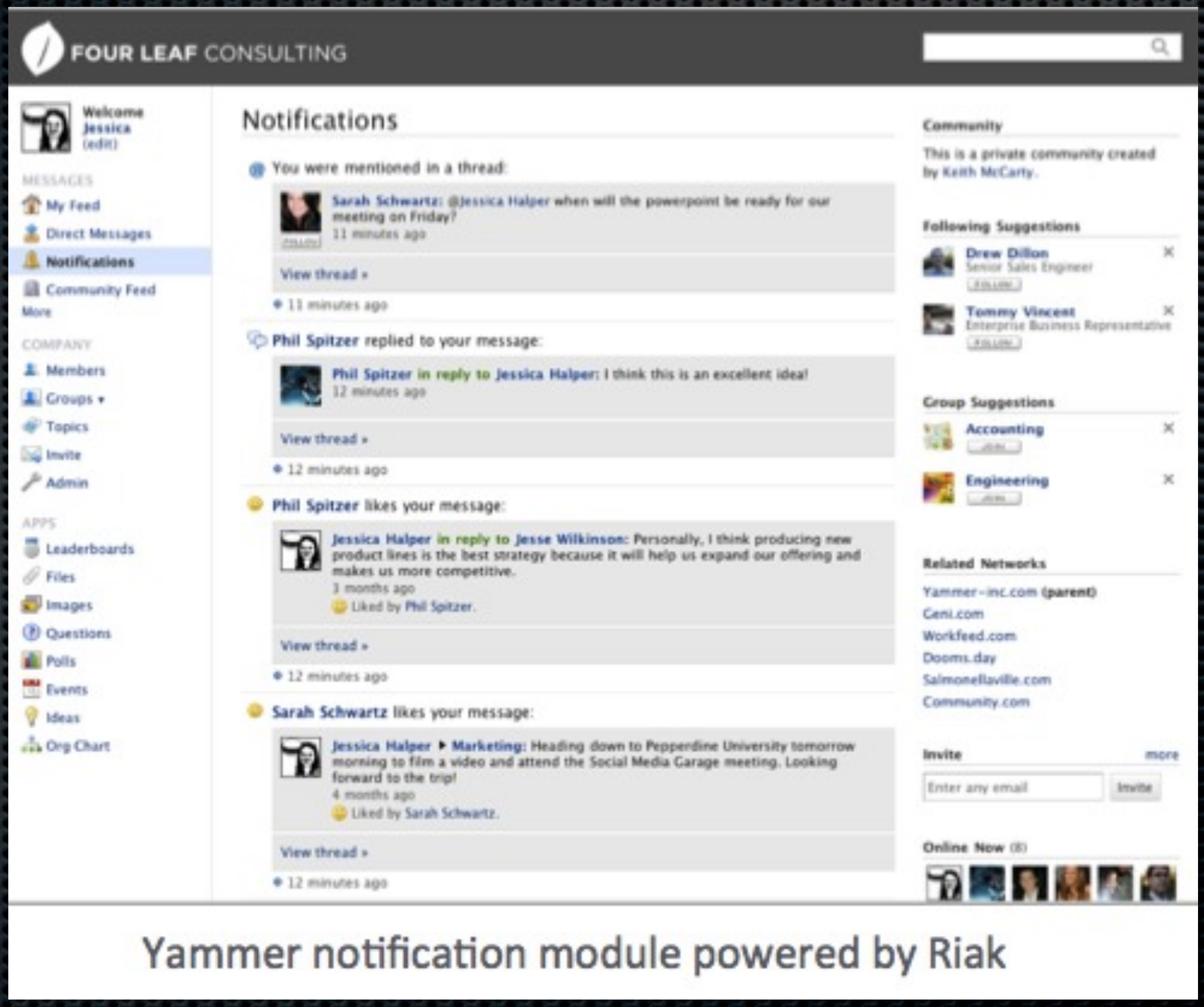
User profile storage for xfinityTV mobile application

Storage of metadata on content providers, and content licensing info

Strict latency requirements

Notification Service

Yammer



Yammer notification module powered by Riak

Session Store
Mochi Media

First Basho Customer (late 2009)

Every hit to a Mochi web property = 1 read, maybe one write to Riak

Unavailability, high latency = lost ad revenue

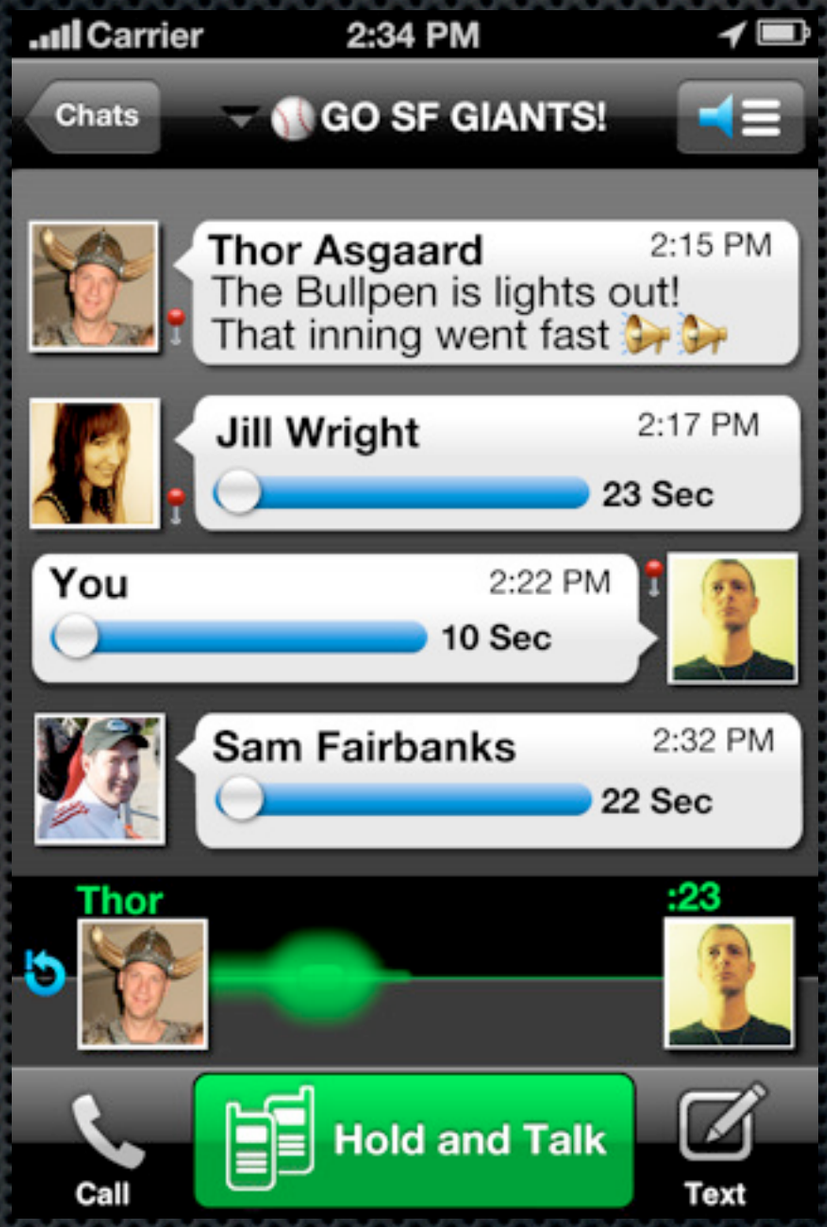
Document Store
Github Pages / Git.io

Riak as a web server for Github Pages

Webmachine is an awesome HTTP server!

Git.io URL shortener

Walkie Talkie
Voxer



Voxer - Initial Stats

- 11 Riak Nodes
- ~500GB dataset
- ~20k peak concurrent users
- ~4MM daily requests

Then something happened...

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

Voxer - Current Stats

- > 100 nodes
- ~1TB data incoming / day
- > 200k concurrent users
- > 2 billion requests / day
- Grew from 11 to 80 nodes Dec - Jan

Distributed Systems: Desirable Properties

- High Availability
- Low Latency
- Horizontal Scalability
- Fault Tolerance
- Ops-Friendliness
- Predictability

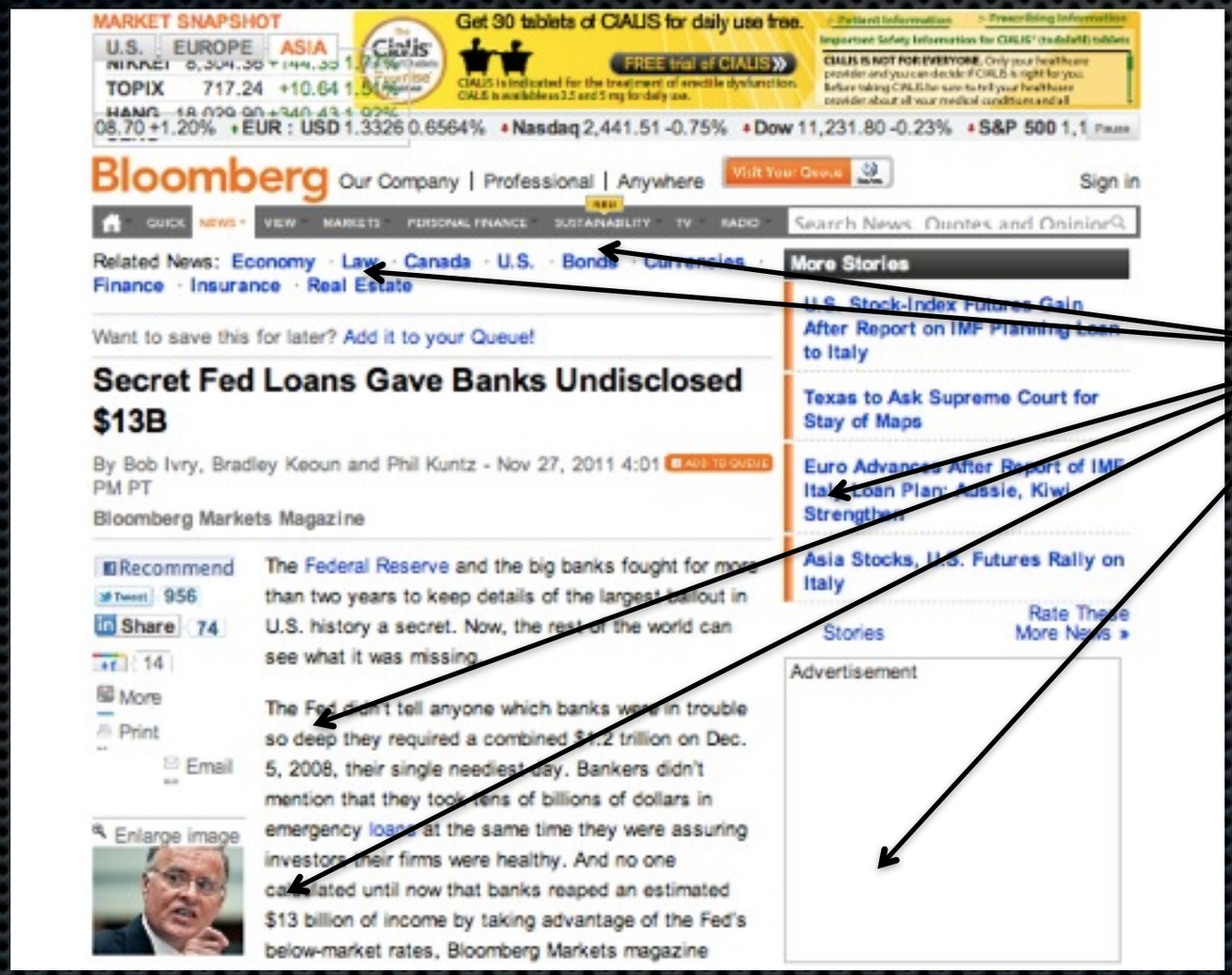
High Availability

- Failure to accept a read/write results in:
  - lost revenue
  - lost users
- Availability and latency are intertwined

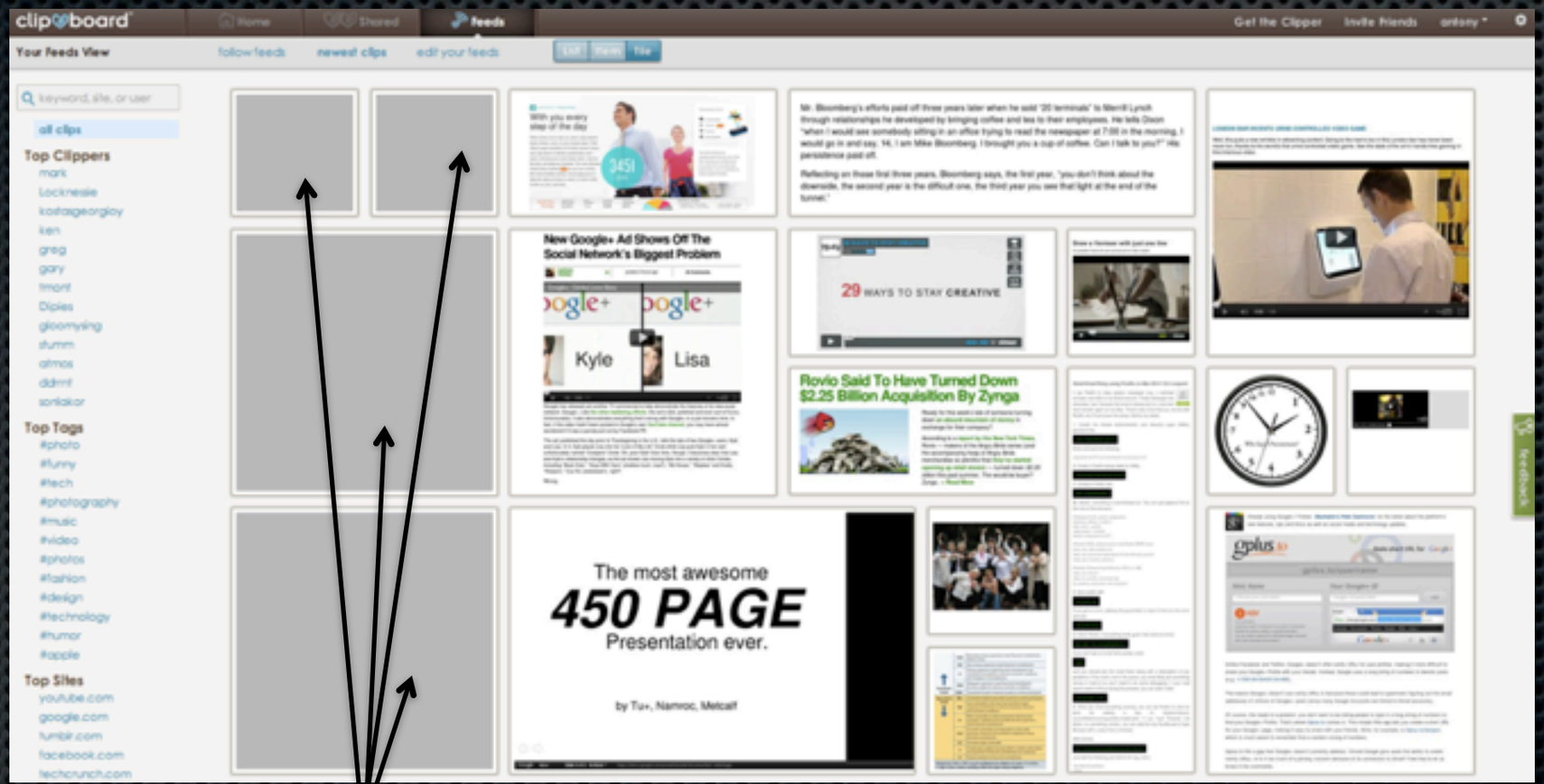
Low Latency

- Sometimes late answer is useless or wrong
- Users perceive slow sites as unavailable
- SLA violations
- SOA approaches magnify SLA failures

Who cares about latency?



Who cares about latency?



Sometimes high latency looks like an outage to the end user.

Fault Tolerance

- Everything fails
  - Especially in the cloud
- When a host/disk/network fails, what is the impact on
  - Availability
  - Latency
  - Operations staff

Predictability

“It’s a piece of plumbing; it has never been a root cause of any of our problems.”

Coda Hale, Yammer

Operational Costs

- Sound familiar?
  - “we chose a bad shard key...”
  - “the master node went down”
  - “the failover script did not run as expected...”
  - “the root cause was traced to a configuration error...”
- ***Staying up all night fighting your database does not make you a hero.***

Consistency, Availability, Latency

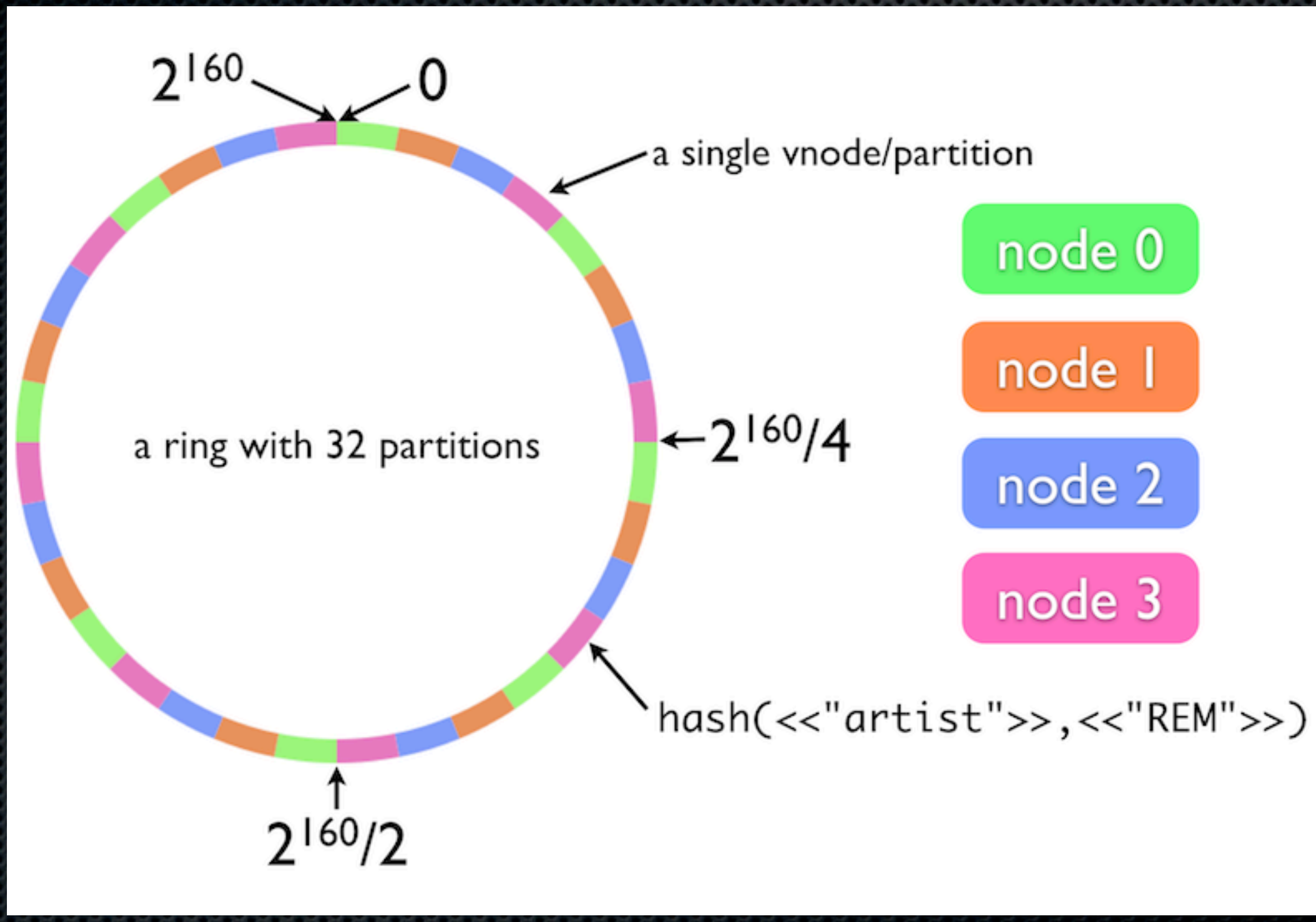
CAP

- The fundamental, most-discussed tradeoff
- When a network partition (message loss) occurs, laws of physics make you choose:
  - Consistency OR
  - Availability
- No system can “beat the CAP theorem”

Data Distribution

- Location of data is determined based on a hash of the key
- Provides even distribution of storage and query load
- Trades off advantages gained from locality
  - range queries
  - aggregates

Consistent Hashing



Virtual Nodes

- Unit of addressing, concurrency in Riak
- Each host manages many vnodes
- Riak *could* manage all host-local storage as a unit and gain efficiency, but would lose
  - simplicity in cluster resizing
  - failure isolation

Append-Only Stores, Bitcask

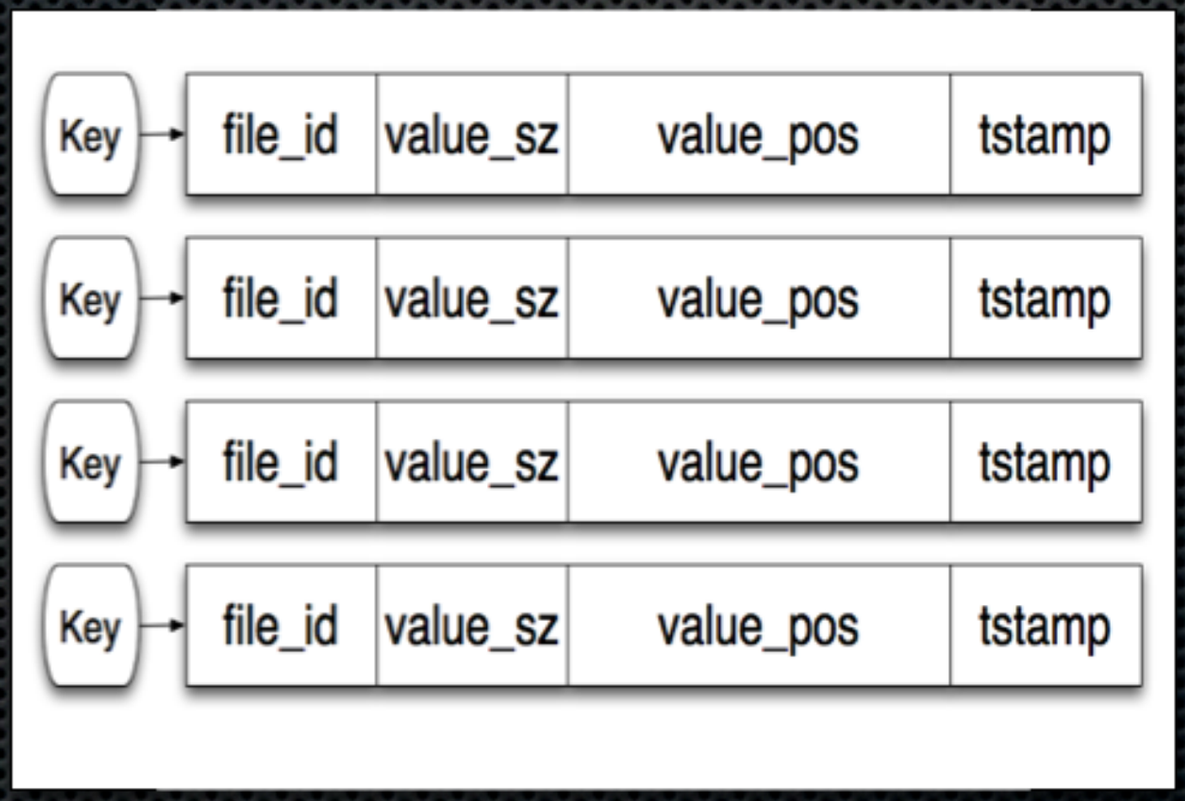
Append-Only Stores

- All writes are appends to a file
- This provides crash-safety, fast writes
- Tradeoff: must periodically compact/merge files to reclaim space
  - Causes periodic pauses while compaction occurs that must be masked/mitigated

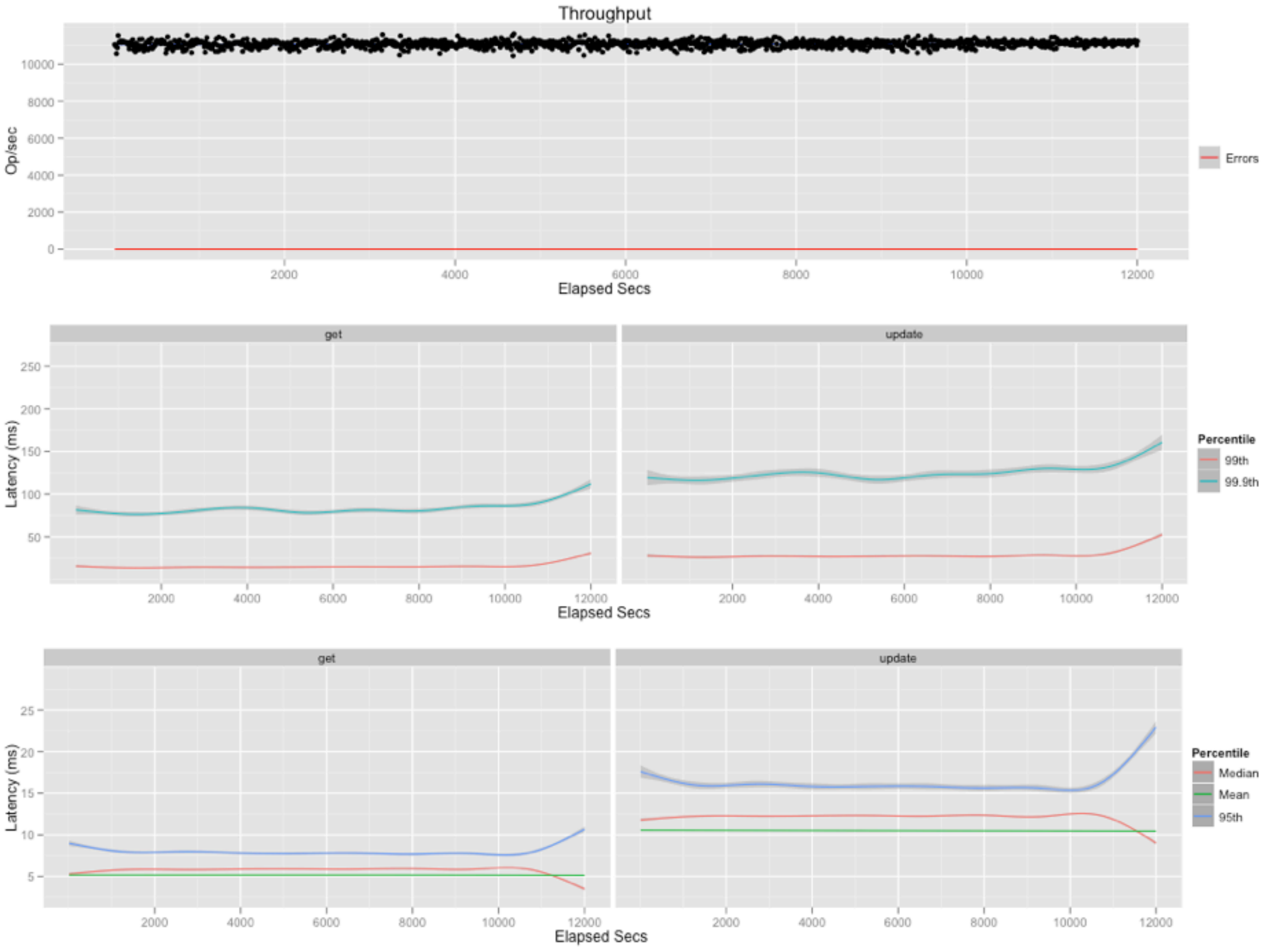
Bitcask

Low Latency: All reads = hash lookup + 1 seek

All writes = append to file



Tradeoff: Index must fit in memory


Throughput
Op/sec
10000
8000
6000
4000
2000
0
2000
4000
6000
8000
10000
12000
Elapsed Secs
Errors
get
update
Latency (ms)
250
200
150
100
50
Percentile
99th
99.9th
Elapsed Secs
get
update
Latency (ms)
25
20
15
10
5
Percentile
Median
Mean
95th
Elapsed Secs

Handoff and Rebalancing

- When nodes are added to a cluster, data must be rebalanced
- Rebalancing causes disk, network load
- Tradeoff: speed of convergence vs. effects on cluster performance

Vector Clocks

- Provide happened-before relationship between events
- Riak tags each object with vector clock
- Tradeoff: space, speed, complexity for safety

Gossip Protocol

- Nodes “gossip” their view of cluster state to each other
- Tradeoffs:
  - atomic modifications of cluster state for no SPOF
  - complexity for fault tolerance

Sane Defaults

- Speed vs. Safety
- Riak ships with N=3, R=W=2
  - Bad for microbenchmarks, good for production use, durability
- Mongo ships with W=0
  - Good for benchmarks, horrible and insane for durability, production use.

Erlang

- Best language ever:
  - for distributed systems glue code
  - for safety, fault tolerance
- Sometimes you want:
  - Destructive operations
  - Shared memory

NIFs to the rescue?

- Use NIFs for speed, interfacing with native code, but:
  - You make the Erlang VM only as reliable as your C code
  - NIFs block the scheduler

Conclusions

- Over time, operational costs dominate
- Predictability in:
  - Latency
  - Scalability
  - Failure scenarios
- ...is essential for managing operational costs
- When choosing a database, raw throughput is often the *least* important metric.

Thanks!

- Visit us at http://www.basho.com
- Check out our open source code at http://github.com/basho
- Follow us on Twitter: @basho
- We're hiring!