Riak Use Cases: Dissecting the Solutions to Hard Problems

Andy Gross <@argv0>
Principal Architect, Basho Technologies

QCon SF 2011

Riak

- Dynamo-inspired key value database
  - with full text search, map/reduce, secondary indices, link traversal, commit hooks, HTTP and binary interfaces
- 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
- NoSQL marketing is... “confusing”

Tradeoffs

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

Distributed Systems: Desirable Properties

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

Medical Records Store
Danish Health Authorities

Implemented by Trifork A/S

Won the Digitization Prize as one of the “best government IT projects in Denmark”

Stores medical prescription history for all Danish Citizens, replicated in 2 data centers.

Accessed from pharmacies, hospital, mobile devices

Replicated in multiple data centers

User/Metadata Store
Comcast

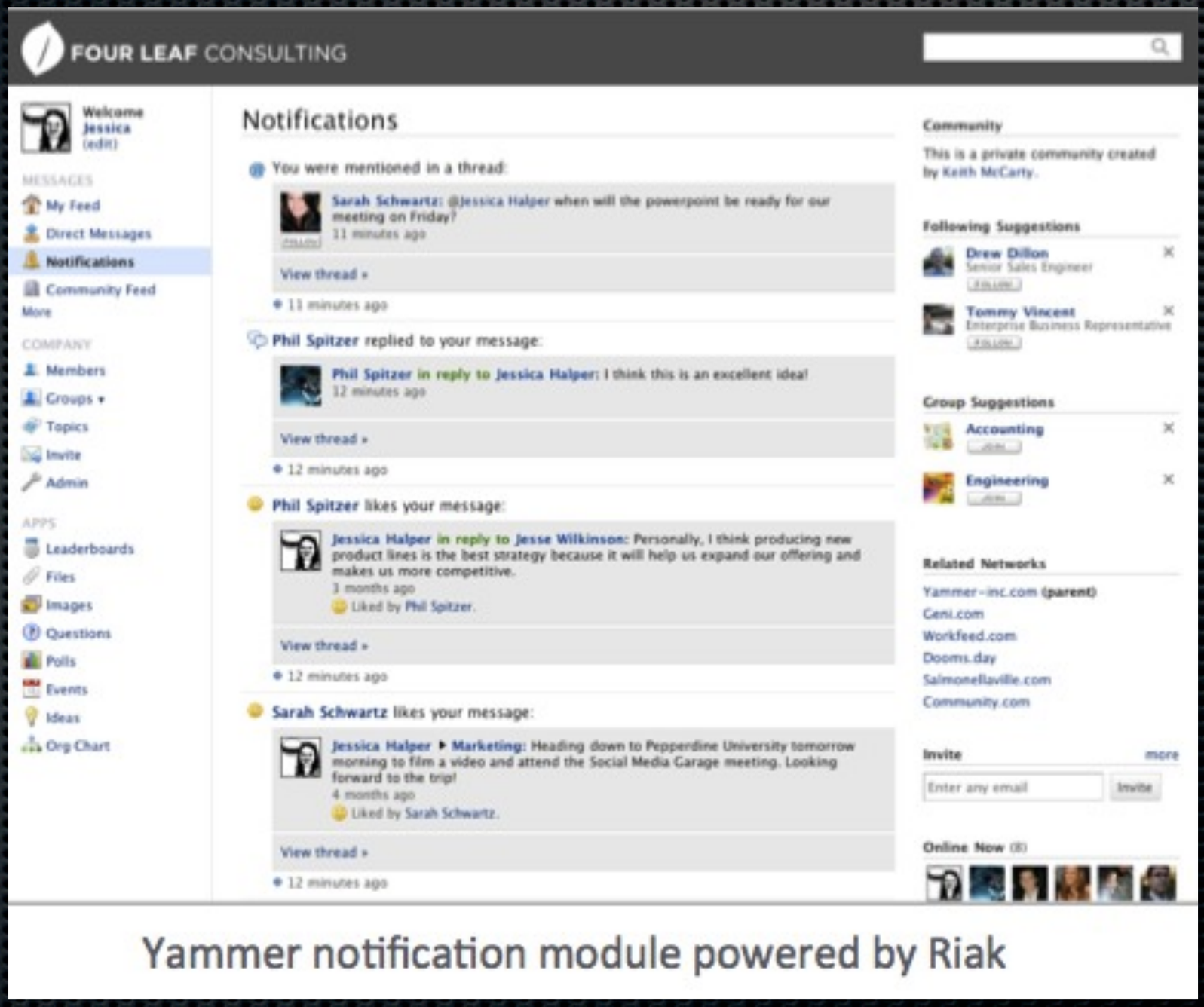
User profile storage for xfinityTV mobile application

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 (in staging)

Webmachine is an awesome HTTP server!

Git.io URL shortener

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

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

Cost

@moonpolysoft
Cliff Moon

Amortize the cost of an database across it's entire life. Turns out the only thing that matters is operational cost.

6 Nov via TweetDeck ☆ Favorite ↻ Retweet ↩ Reply

Retweeted by murf and 22 others

Operational Costs

- Sound familiar?
  - “we chose a bad shard key...”
  - “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.***

High Availability: Erlang

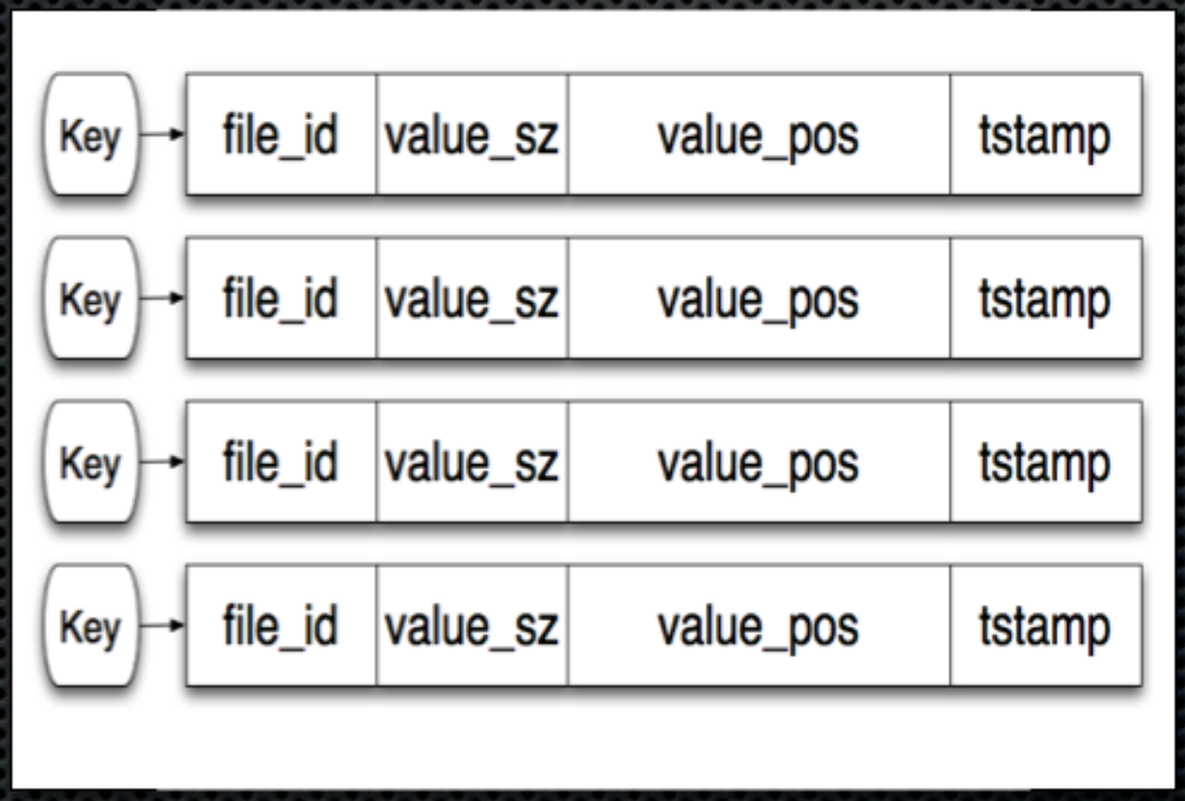
- Ericcson AXD-301: 99.9999999% uptime (31ms/year)
- Shared-nothing, immutable, message-passing, functional, concurrent
- Distributed systems primitives in core language
- OTP (Open Telecom Platform)

High Availability: Riak Core

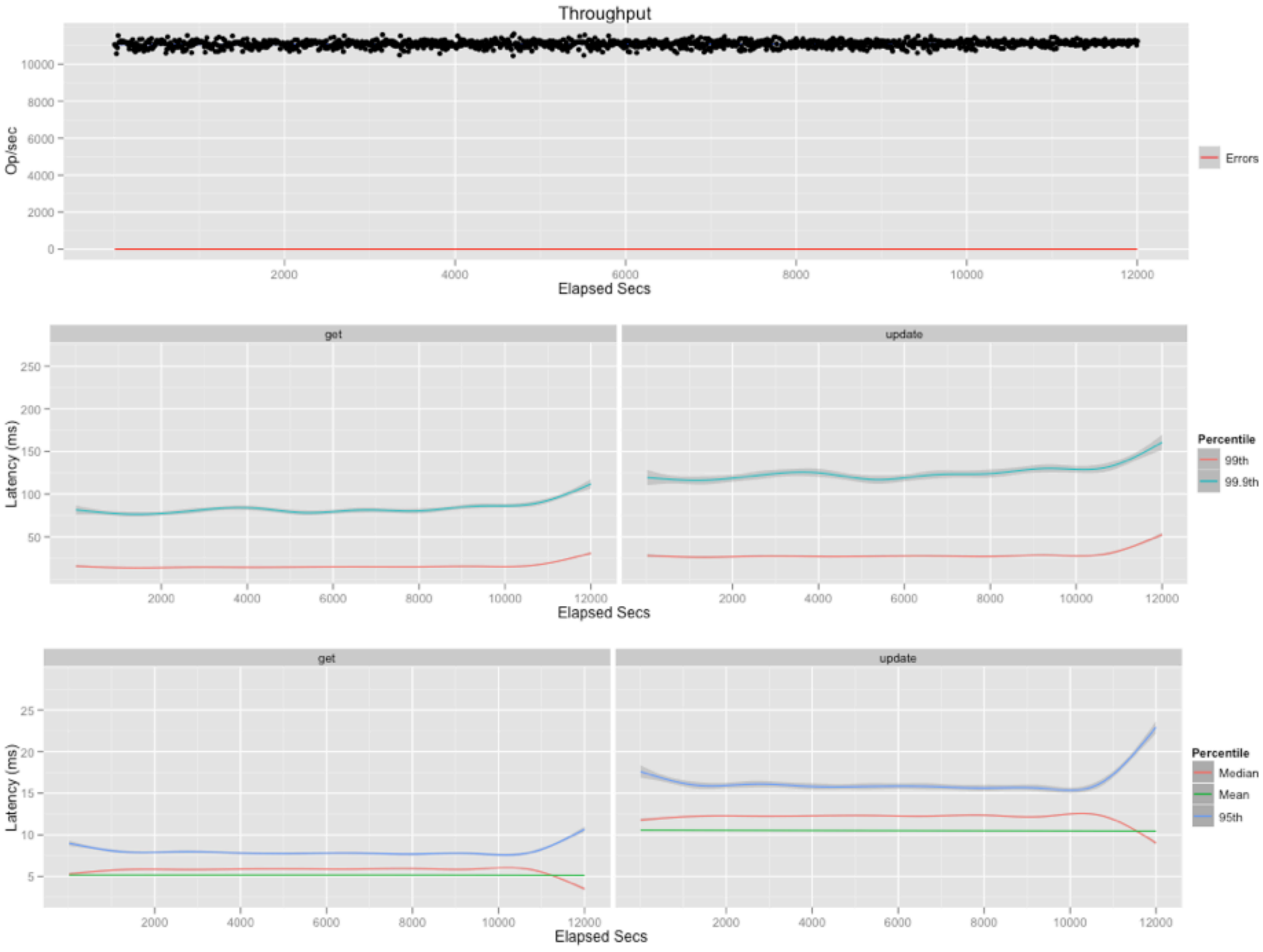
- Dynamo abstracted: distributed systems toolkit
- Exhaustively tested
- In production use at AOL, Yahoo, others
- Insulates local storage and client API code from the hard problems

Low Latency: Bitcask

Low Latency: All reads = hash lookup + 1 seek

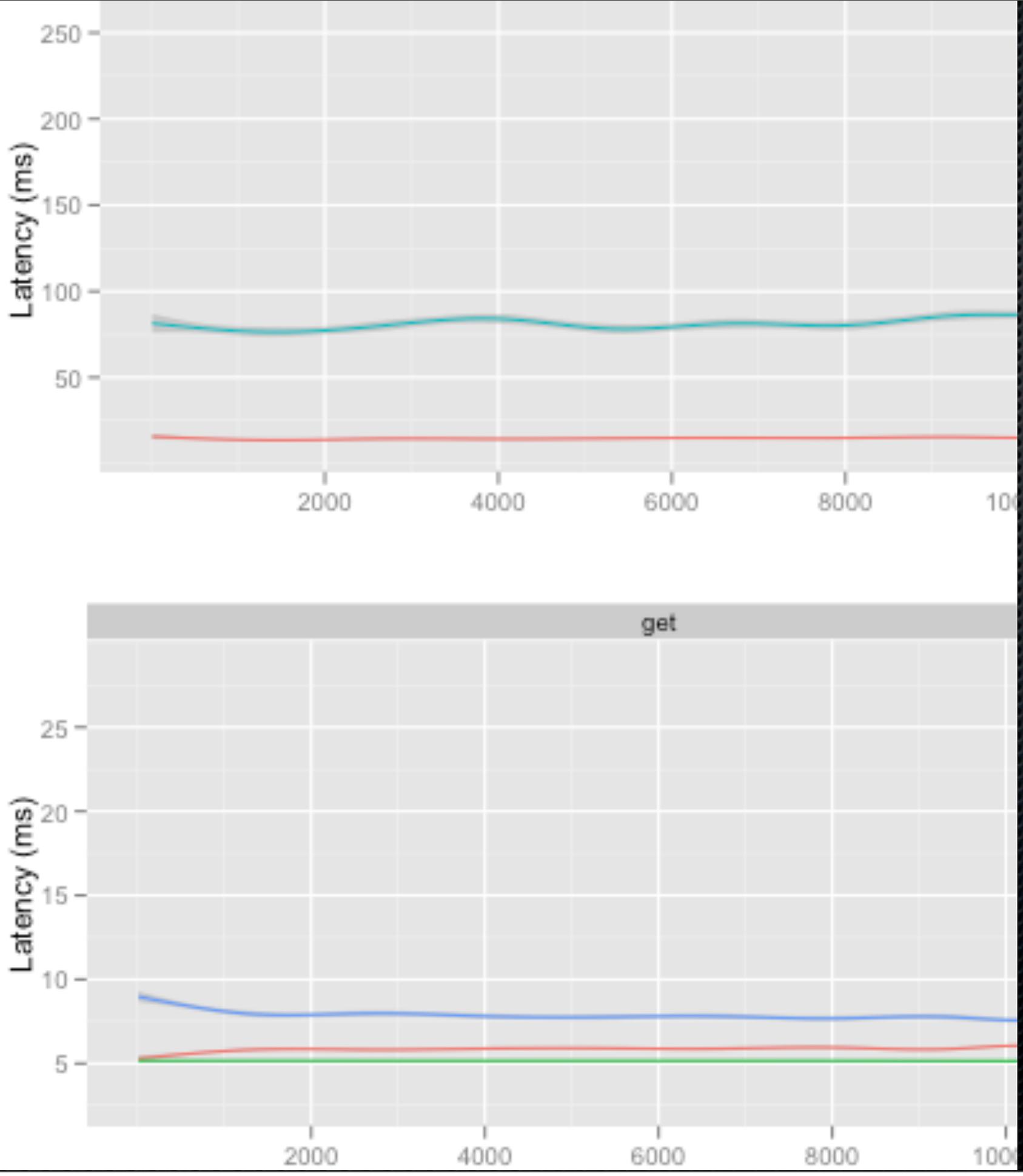


Tradeoff: Index must fit in memory


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

Low Latency: Erlang VM

- Erlang VM was designed for soft-realtime apps
  - Preemptively scheduled lightweight threads
  - GC is per-thread, not stop-the-world
  - Sophisticated scheduler + message passing = effective use of multicore machines.


Latency (ms)
250
200
150
100
50
2000
4000
6000
8000
get
Latency (ms)
25
20
15
10
5
2000
4000
6000
8000

Questions?