Dynamo, Five Years Later

Andy Gross

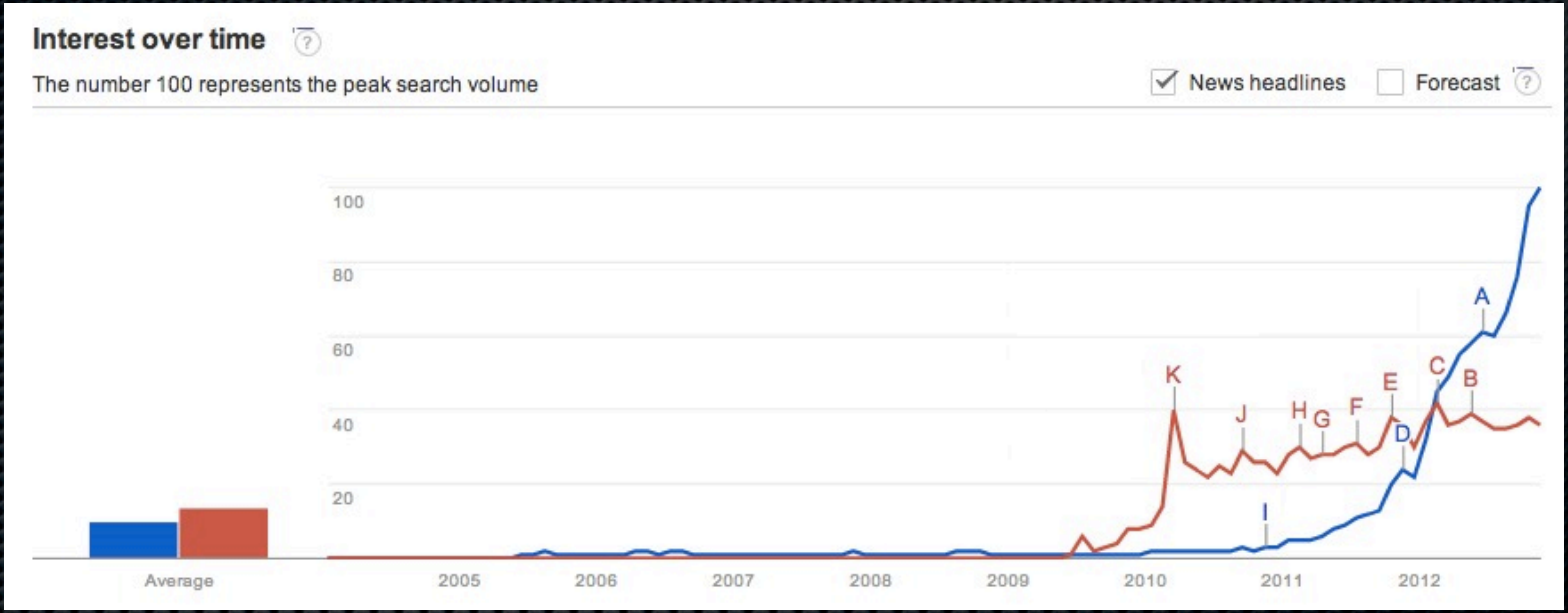
Chief Architect, Basho Technologies

QCon SF 2012

Dynamo

- Published October 2007 @ SOSP
- Describes a collection of distributed systems techniques applied to low-latency key-value storage
- Spawned (along with BigTable) many imitators, an industry (LinkedIn -> Voldemort, Facebook -> Cassandra)
- Authors nearly got fired from Amazon for publishing

NoSQL and Big Data



Interest over time

The number 100 represents the peak search volume

☑ News headlines □ Forecast

Riak

- First lines of first prototype written in Fall 2007 on a plane on the way to my Basho interview
- My excuse to learn Erlang while reading the Dynamo paper
- A huge example of NIH Syndrome
- “Technical Debt” is another term we use at Basho for this code
- 1.0 in September 2011, 1.3 coming this year

Principles

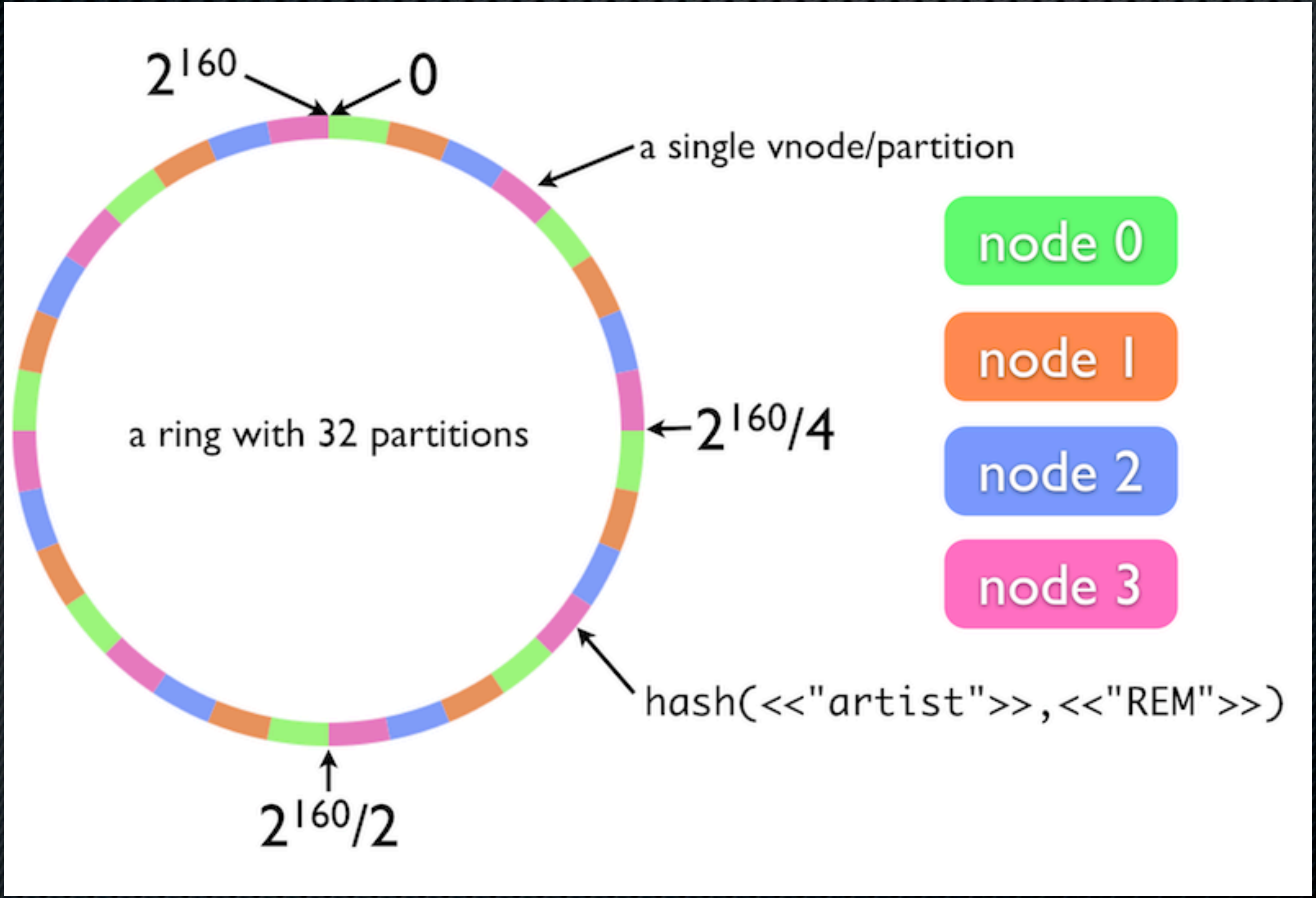
- Always-writable
- Incrementally scalable
- Symmetrical
- Decentralized
- Heterogenous
- Focus on SLAs, tail latency

Techniques

- Consistent Hashing
- Vector Clocks
- Read Repair
- Anti-Entropy
- Hinted Handoff
- Gossip Protocol

Consistent Hashing

- Invented by Danny Lewin and others @ MIT/Akamai
- Minimizes remapping of keys when number of hash slots changes
- Originally applied to CDNs, used in Dynamo for replica placement
- Enables incremental scalability, even spread
- Minimizes hot spots


2^{160}
0
a single vnode/partition
node 0
node 1
node 2
node 3
a ring with 32 partitions
$2^{160}/4$
hash(<<"artist">>,<<"REM">>)
$2^{160}/2$

Vector Clocks

- Introduced by Mattern et al, in 1988
- Extends Lamport’s timestamps (1978)
- Each value in Dynamo tagged with vector clock
- Allows detection of stale values, logical siblings

Read Repair

- Update stale versions opportunistically on reads (instead of writes)
- Pushes system toward consistency, after returning value to client
- Reflects focus on a cheap, always-available write path

Hinted Handoff

- Any node can accept writes for other nodes if they’re down
- All messages include a destination
- Data accepted by node other than destination is handed off when node recovers
- As long as a single node is alive the cluster can accept a write

Anti-Entropy

- Replicas maintain a Merkle Tree of keys and their versions/hashes
- Trees periodically exchanged with peer vnodes
- Merkle tree enables cheap comparison
- Only values with different hashes are exchanged
- Pushes system toward consistency

Gossip Protocol

- Decentralized approach to managing global state
- Trades off atomicity of state changes for a decentralized approach
- Volume of gossip can overwhelm networks without care

Problems with Dynamo

- Eventual Consistency is harsh mistress
  - Pushes conflict resolution to clients
- Key/value data types limited in use
- Random replica placement destroys locality
- Gossip protocol can limit cluster size
- R+W > N is **NOT** more consistent
- TCP Incast

Key-Value
Conflict Resolution

- Forcing clients to resolve consistency issues on read is a pain for developers
- Most end up choosing the server-enforced last-write-wins policy
- With many language clients, logic must be implemented many times
- One solution: https://github.com/bumptech/montage
- Another: Make everything immutable
- Another: CRDTs

Optimize for Immutability

- “Mutability, scalability are generally at odds” - Ben Black
- Eventual consistency is *great* for immutable data
- Conflicts become a non-issue if data never changes
  - don’t need full quorums, vector clocks
  - backend optimizations are possible
- Problem space shifts to distributed GC
- See Pat Helland’s Talk @ http://ricon2012.com

CRDTs

- Conflict-free, Replicated Data Types
- Lots of math - see Sean Cribbs and Russell Brown's RICON presentation
- A server side structure and conflict-resolution policy for richer datatypes like counters and sets
- Prototype here: http://github.com/basho/riak_dt

Random Placement and Locality

- By default, keys are randomly placed on different replicas
- But we have buckets!
- Containers imply cheap iteration/enumeration, but with random placement it becomes an expensive full-scan
- Partial Solution: hash function defined per-bucket can increase locality
- Lots of work done to minimize impact of bucket listings

(R+W>N) != Consistency

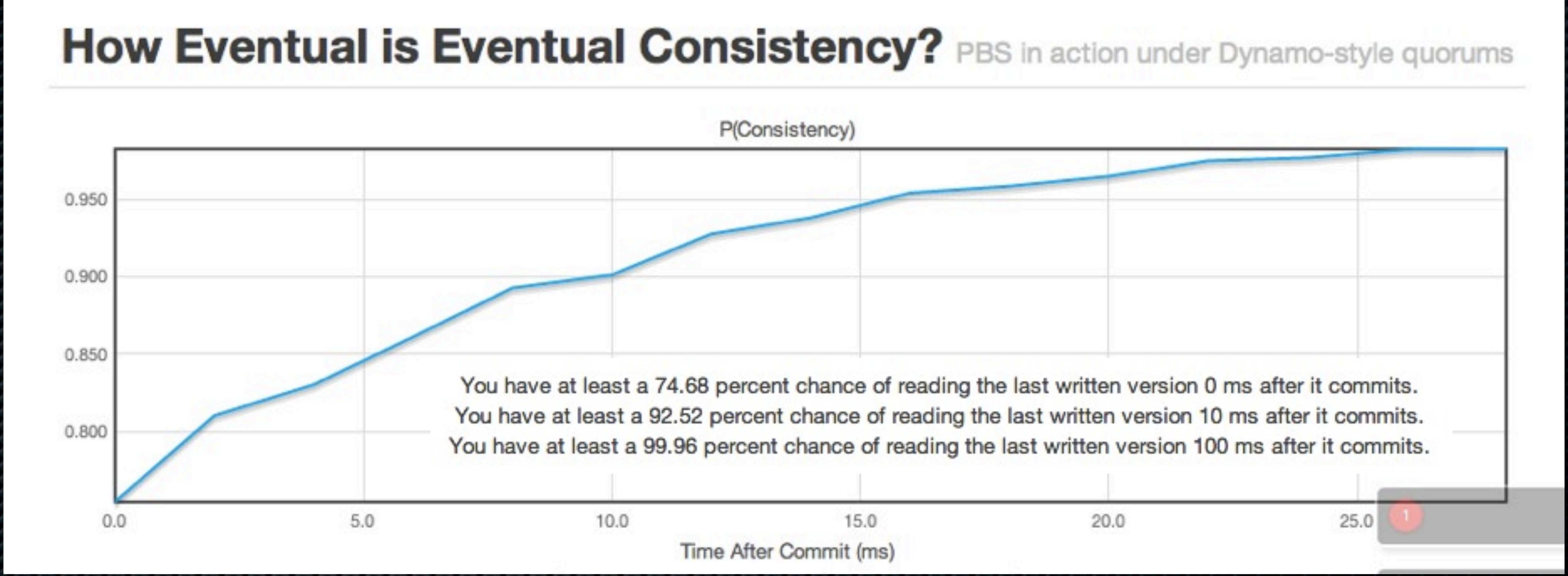
- R+W described in Dynamo paper as “consistency knobs”
  - Some Basho/Riak docs still say this too! :(
- Even if R=W=N, sloppy quorums and partial writes make reading old values possible
- “Read your own writes if your writes succeed but otherwise you have no idea what you’re going to read consistency (RYOWIWSBOYHNIWYGTRC)” - Joe Blomstedt
- Solution: actual “strong” consistency

Strong Consistency in Riak

- CAP says you must choose C vs. A, but only during failures
- There's no reason we can't implement both models, with different tradeoffs
- Enable strong consistency on a per-bucket basis
- See Joe Blomstedt's talk at RICON 2012: http://ricon2012.com, earlier work at: http://github.com/jtuple/riak_zab

An Aside: Probabalistically Bounded Staleness

R=W=1, .1ms latency at all hops



How Eventual is Eventual Consistency? PBS in action under Dynamo-style quorums

You have at least a 74.68 percent chance of reading the last written version 0 ms after it commits.
You have at least a 92.52 percent chance of reading the last written version 10 ms after it commits.
You have at least a 99.96 percent chance of reading the last written version 100 ms after it commits.

Bailis et al. : http://pbs.cs.berkeley.edu

TCP Incast

- “You can’t pour two buckets of manure into one bucket” - Scott Fritchie’s Grandfather
- “microbursts” of traffic sent to one cluster member
    - Coordinator sends request to three replicas
    - All respond with large-ish result at roughly the same time
    - Switch has to either buffer or drop packets
- Cassandra tries to mitigate: 1 replica sends data, others send hashes. We should do this in Riak.

What Riak Did Differently (or wrong)

- Screwed up vector clock implementation
  - Actor IDs in vector clocks were *client* ids, therefore potentially unbounded
  - Size explosion resulted in huge objects, caused OOM crashes
  - Vector clock pruning resulted in false siblings
  - Fixed by forwarding to node in preflist circa 1.0

What Riak Did Differently

- No active anti-entropy
  - Early versions had slow, unstable AAE
  - Node loss required reading all objects and repopulating replicas via read repair
    - Ok for objects that are read often
    - Rarely-read objects N value decreases over time
- Will be fixed in Riak 1.3

What Riak Did Differently

- Initial versions had an unavailability window during topology changes
  - Nodes would claim partitions immediately, before data had been handed off
  - New versions don’t change request preflist until all data has been handed off
  - Implemented as 2PC-ish commit over gossip

Riak, Beyond Dynamo

- MapReduce
- Search
- Secondary Indexes
- Pre/post-commit hooks
- Multi-DC replication
- Riak Pipe distributed computation
- Riak CS

Riak CS

- Amazon S3 clone implemented as a proxy in front of Riak
- Handles eventual consistency issues, object chunking, multitenancy, and API for a much narrower use case
- Forced us to eat our own dogfood and get serious about fixing long-standing warts
- Drives feature development

Riak the Product vs. Dynamo the Service

- Dynamo had luxury of being a service while Riak is a product
  - Screwing things up with Riak can not be fixed with an emergency deploy
  - Multiple platforms, packaging are challenges
  - Testing distributed systems is another talk entirely (QuickCheck FTW)
    - http://www.erlang-factory.com/upload/presentations/514/TestFirstConstructionDistributedSystems.pdf

Riak Core

- Some of our best work!
- Dynamo abstracted
- Implements all Dynamo techniques without prescribing a use case
- Examples of Riak Core apps:
  - Riak KV!
  - Riak Search
  - Riak Pipe

Riak Core

- Production deployments
    - OpenX: several 100+-node clusters of custom Riak Core systems
    - StackMob: proxy for mobile services implemented with Riak Core
- Needs to be *much* easier to use and better documented

Erlang

- Still the best language for this stuff, but
  - We mix data and control messages over Erlang message passing. Switch to TCP (or uTP/UDT) for data
  - NIFs are problematic
  - VM tuning can be a dark art
- ~90 public repos of mostly-Erlang, mostly-awesome open source: https://github.com/basho

Other Future Directions

- Security was not a factor in Dynamo’s or Riak’s design
  - Isolating Riak increases operational complexity, cost
- Statically sized ring is a pain
- Explore possibilities with smarter clients
- Support larger clusters
- Multitenancy, tenant isolation
- More vertical products like Riak CS

Questions?

@argv0

We’re hiring!

http://www.basho.com