Eventual Consistency

In the real world

or

Why You Already Know Eventual Consistency

or

Eventual Consistency is better* than Eventual Availability

*depending on the use case

@roder

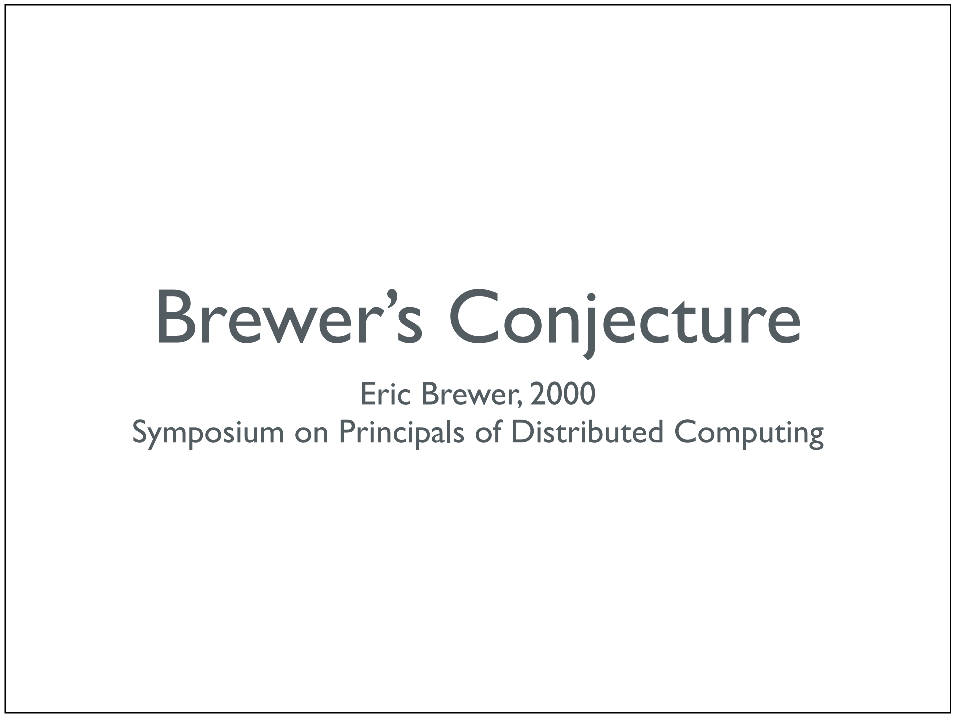
Matt Heitzenroder

basho

riak

We <3 Distributed Systems

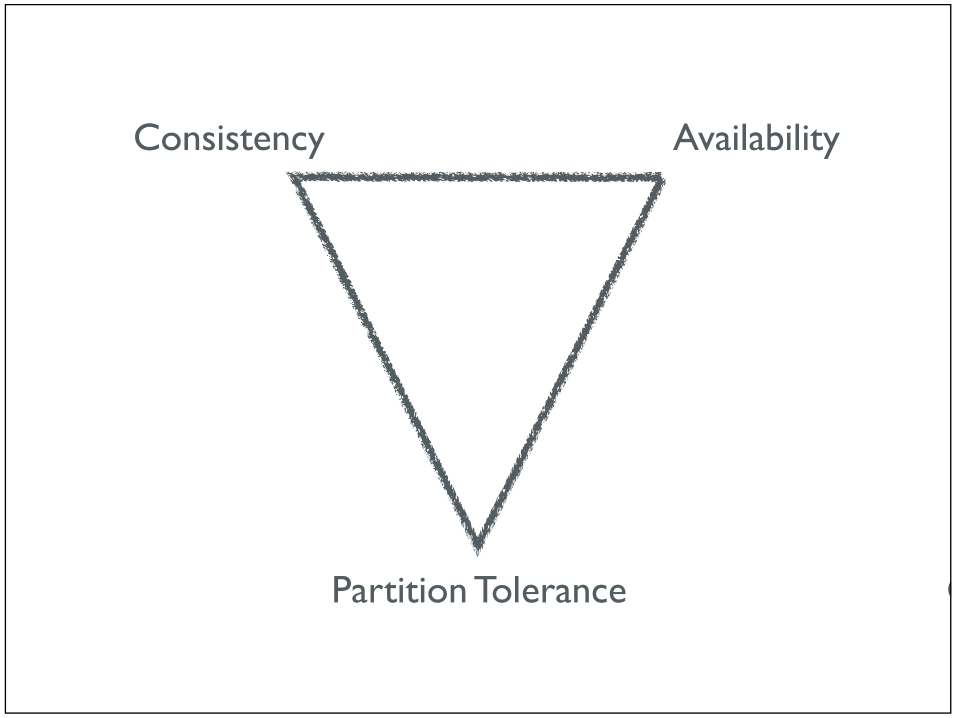
Basho has a Distributed Systems experts on the board – Eric Brewer



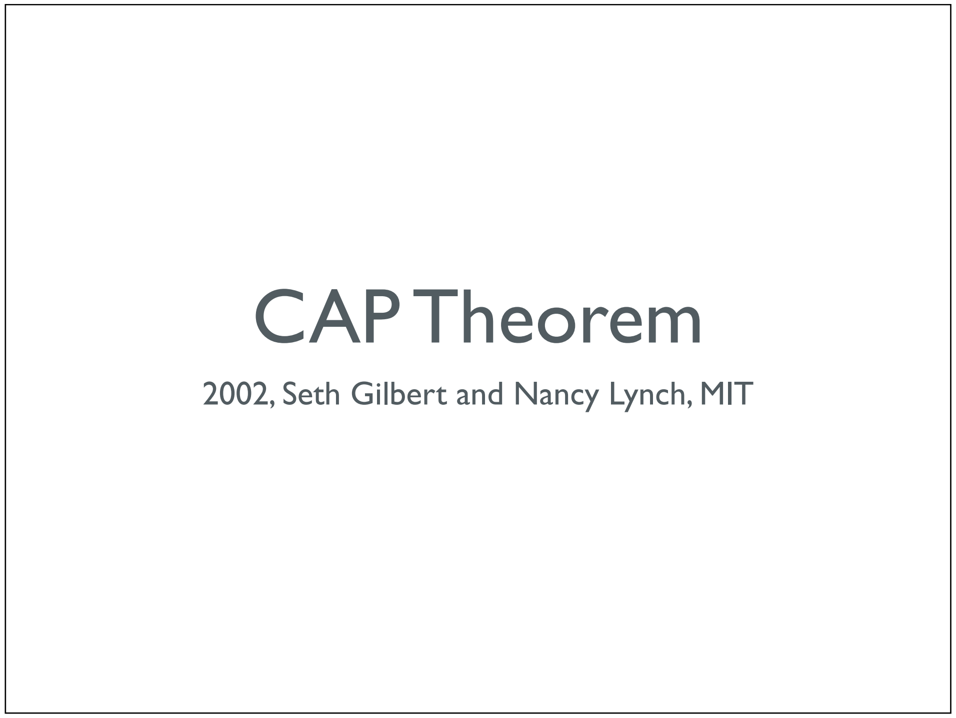
Brewer's Conjecture

Eric Brewer, 2000
Symposium on Principals of Distributed Computing

Eric Brewer, UC Berkley 2000



impossible for a distributed system to all 3 guarantees simultaneously

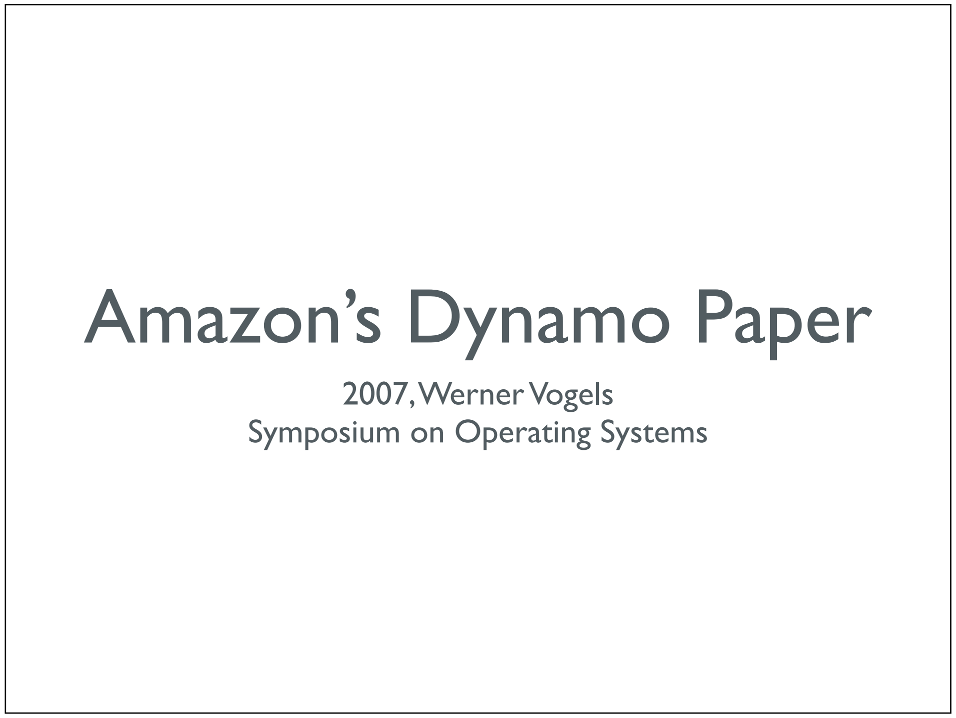


CAP Theorem

2002, Seth Gilbert and Nancy Lynch, MIT

formal proof in 2002

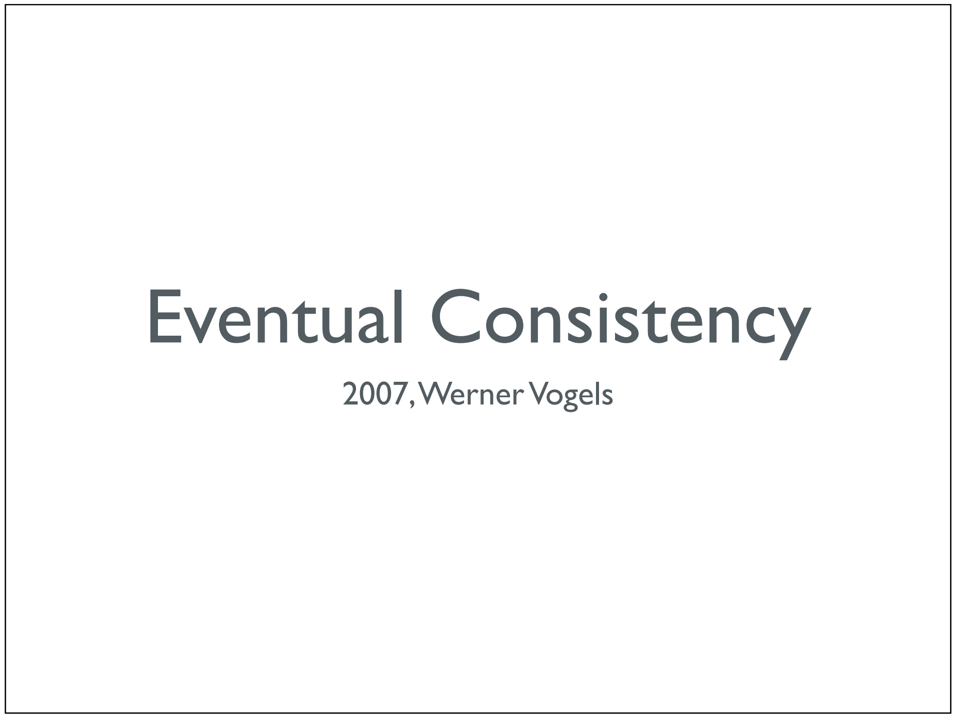
Life is full of Tradeoffs



Amazon’s Dynamo Paper

2007, Werner Vogels
Symposium on Operating Systems

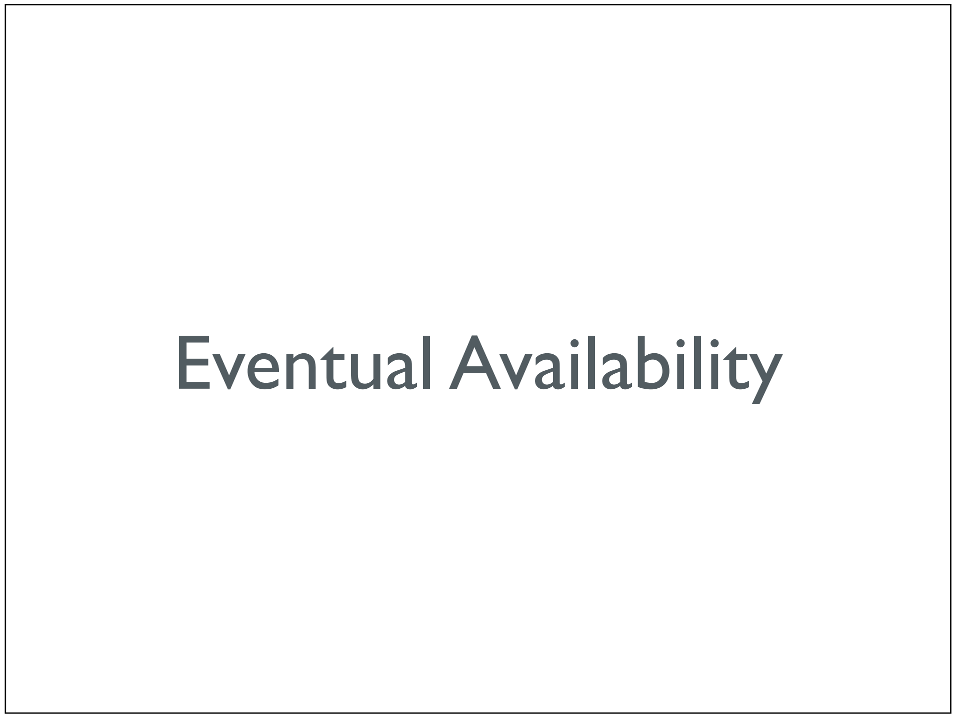
addresses need for incrementally scalable, highly-available key-value storage system



Eventual Consistency

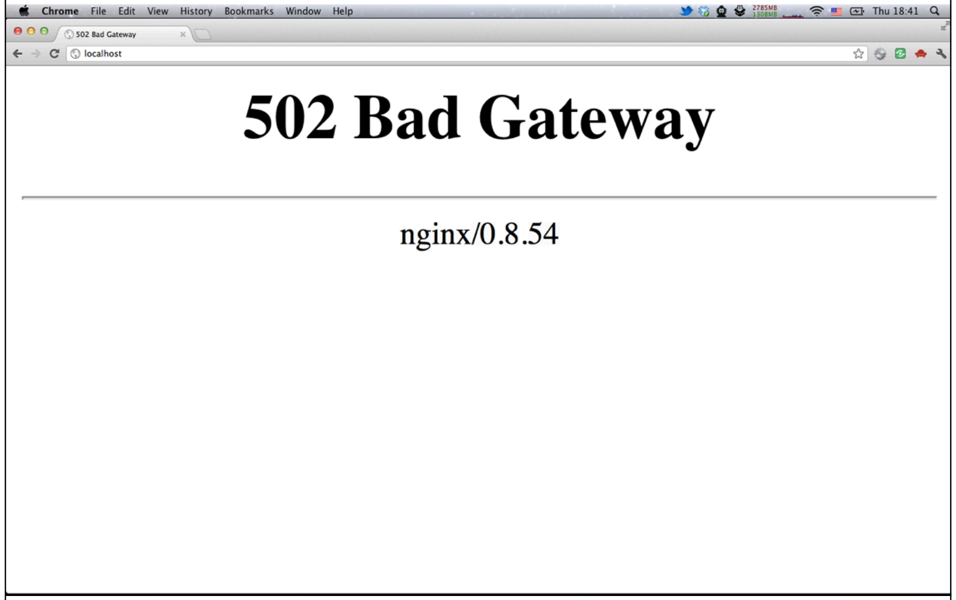
2007, Werner Vogels

Shopping cart is the defacto example of eventual consistency



Eventual Availability

We don’t talk about this enough – what does it look like?



502 Bad Gateway

nginx/0.8.54

Eventual Consistency

In the real world

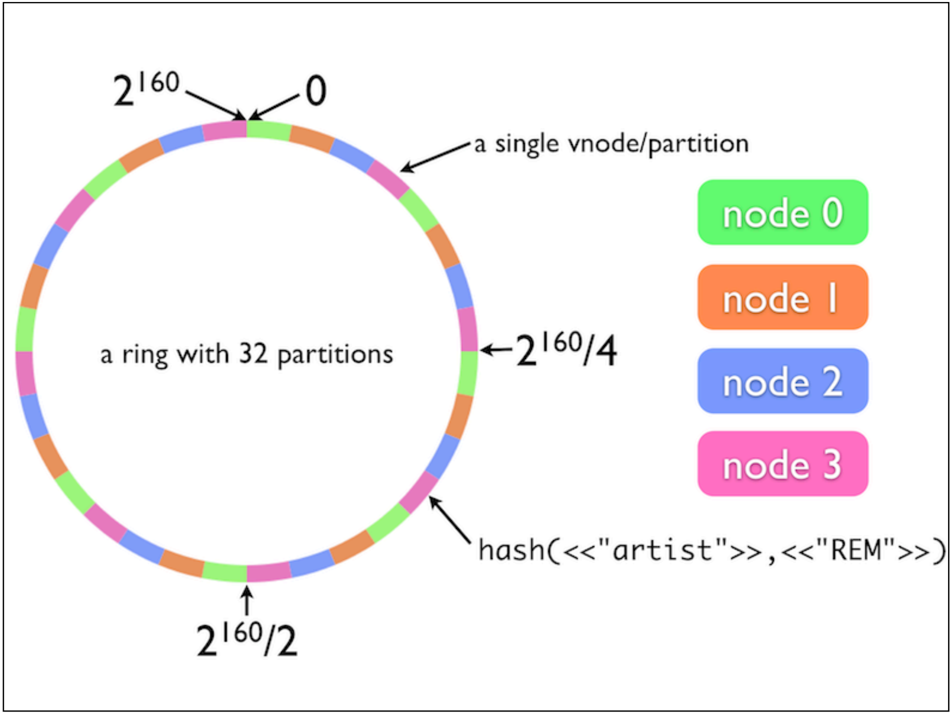
HSBC

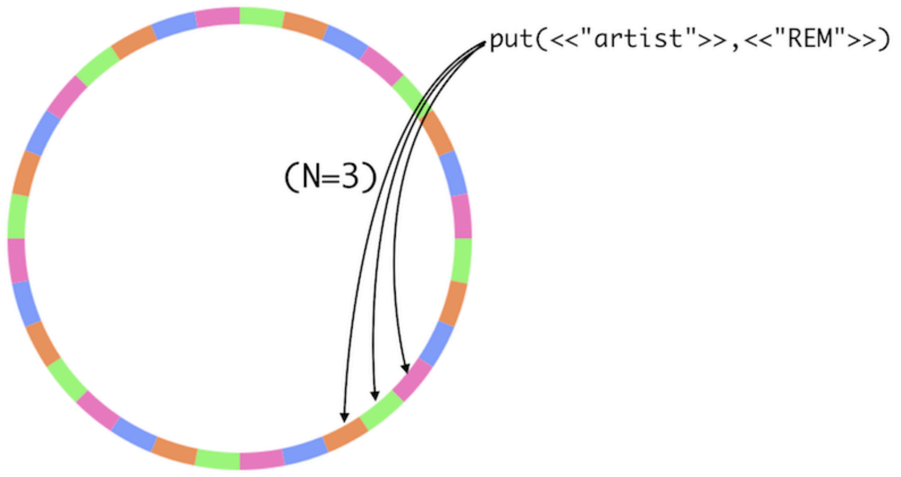
HSBC
DIEBOLD
PLUS FROM HSBC
MORE THAN A CHECKING ACCOUNT
STOP IN AND ASK US ABOUT
THE NEW MATH OF BANKING
FEE NOTICE

JCB
nuttall

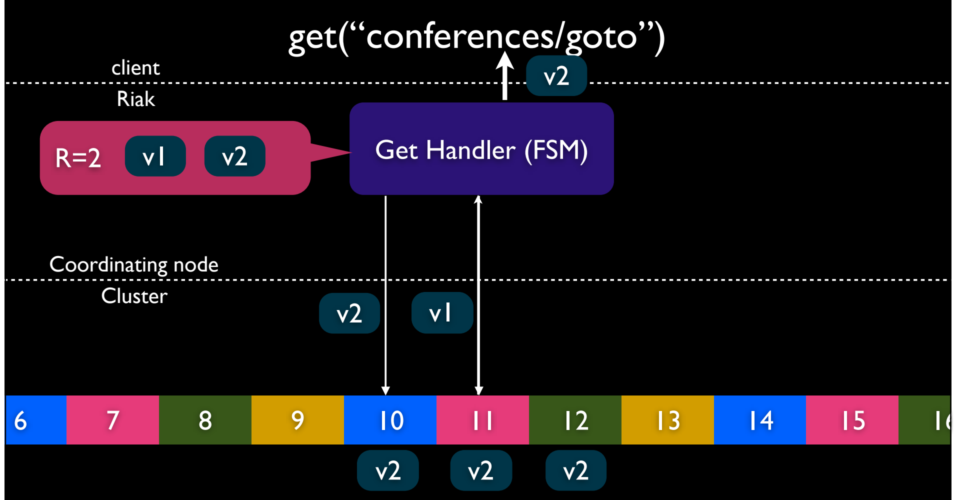
Eventual Consistency

In Riak


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


put(<<"artist">>,<<"REM">>)
(N=3)

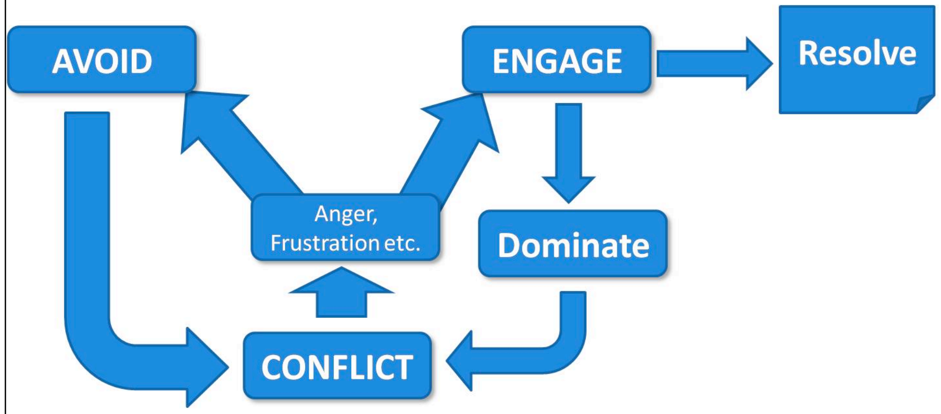
Read Repair



get("conferences/goto")
client
Riak
v2
R=2
v1
v2
Get Handler (FSM)
Coordinating node
Cluster
v2
v1
6
7
8
9
10
11
12
13
14
15
v2
v2
v2

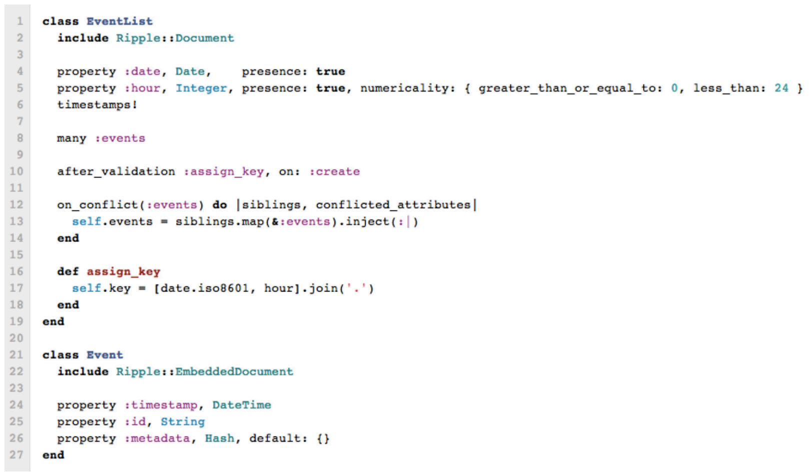
Divergent Object Versions

aka “Siblings”


AVOID
ENGAGE
Resolve
Anger,
Frustration etc.
Dominate
CONFLICT

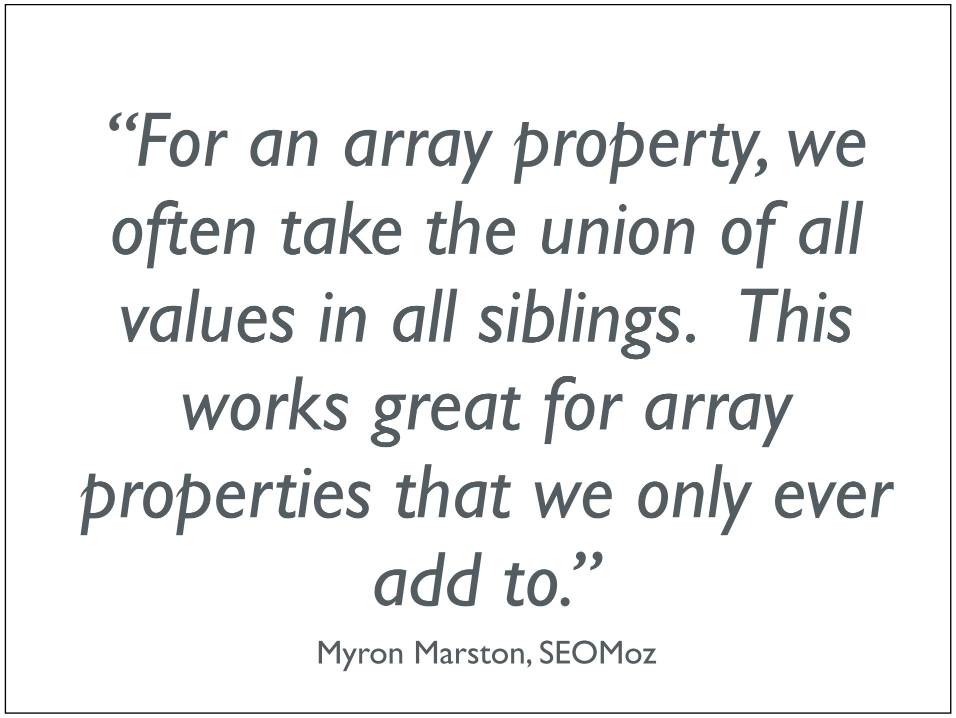
"We don't ever do conflict resolution by picking a particular sibling."

Myron Marston, SEOMoz



```ruby
class EventList
  include Ripple::Document

  property :date, Date,    presence: true
  property :hour, Integer, presence: true, numericality: { greater_than_or_equal_to: 0, less_than: 24 }
  timestamps!

  many :events

  after_validation :assign_key, on: :create

  on_conflict(:events) do |siblings, conflicted_attributes|
    self.events = siblings.map(&:events).inject(:|)
  end

  def assign_key
    self.key = [date.iso8601, hour].join('.')
  end
end

class Event
  include Ripple::EmbeddedDocument

  property :timestamp, DateTime
  property :id, String
  property :metadata, Hash, default: {}
end
```

courtesy of Myron Marston, SEOMoz



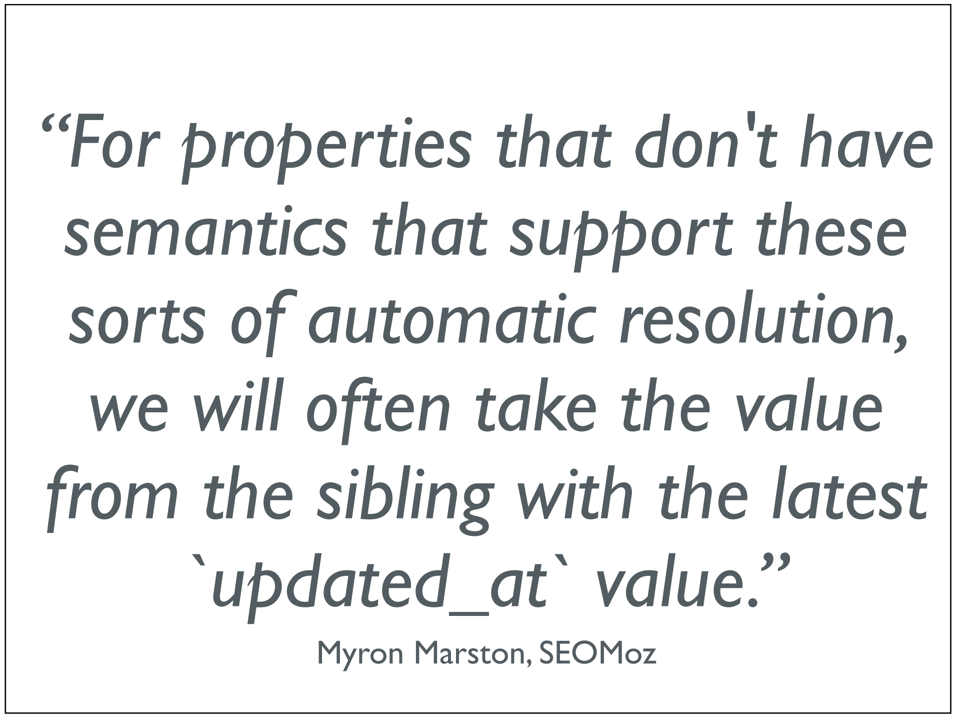
"For an array property, we often take the union of all values in all siblings. This works great for array properties that we only ever add to."

Myron Marston, SEOMoz

set union

“For a timestamp property, we often take the maximum sibling value or the minimum sibling value, depending on the semantics of that attribute.”

Myron Marston, SEOMoz

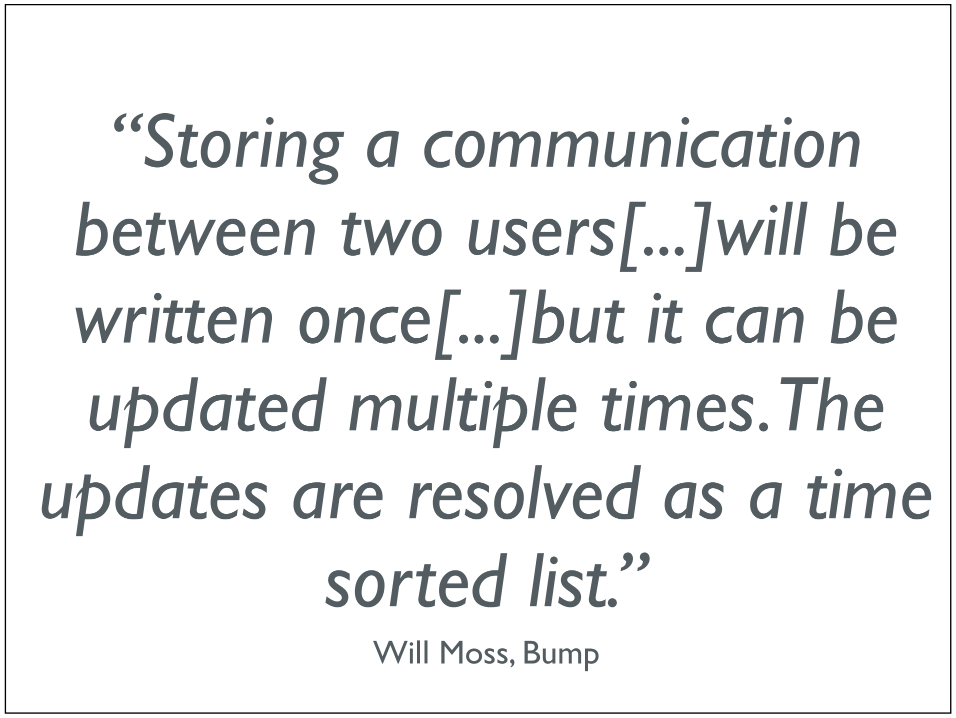


“For properties that don't have semantics that support these sorts of automatic resolution, we will often take the value from the sibling with the latest `updated_at` value.”

Myron Marston, SEOMoz

timestamp in the body of the data object

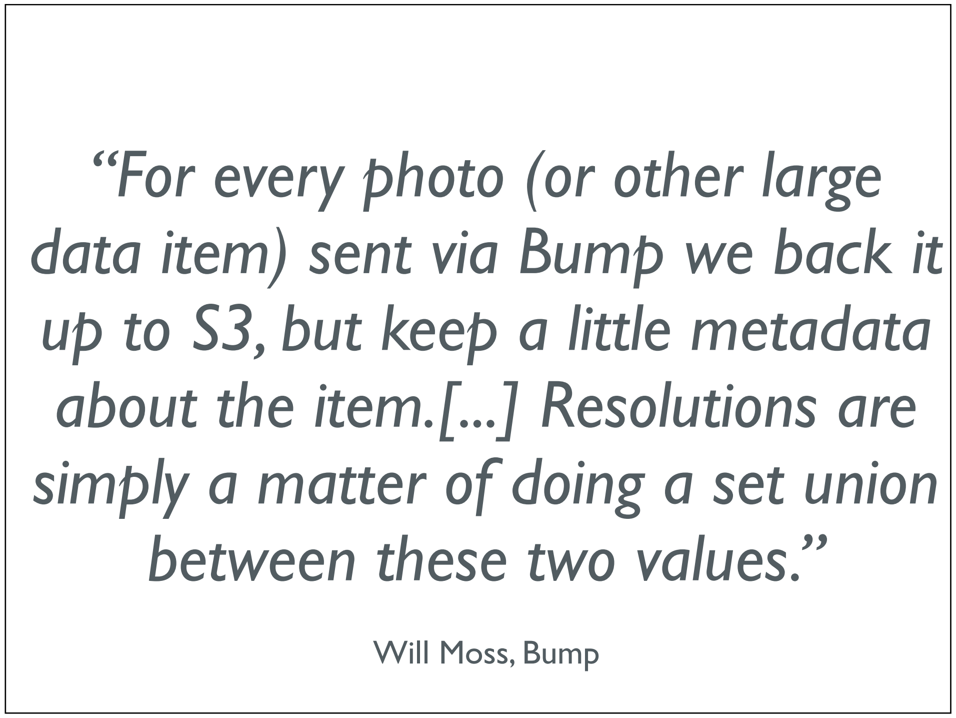
bump
App Store
Android
Scaling to
80 million users
and beyond!
Powered by
basho
CBS



"Storing a communication between two users[...]will be written once[...]but it can be updated multiple times. The updates are resolved as a time sorted list."

Will Moss, Bump

set union sorted by timestamp that is part of the metadata



"For every photo (or other large data item) sent via Bump we back it up to S3, but keep a little metadata about the item.[...] Resolutions are simply a matter of doing a set union between these two values."

Will Moss, Bump

set union based on the metadata

in the real word, events happen concurrently.
We have ways of dealing with it and we must encode them.

http://pbs.cs.berkeley.edu/

quantitatively demonstrate why eventual consistency is "good enough" for many users

Matt Heitzenroder

@roder