INTRODUCTION TO BIGCOUCH

ROBERT NEWSON

COUCHDB CONF BERLIN
JANUARY 2013
ROBERT NEWSON
CouchDB PMC member
IRC menace

BigCouch
Putting the “C” back in CouchDB

Contact
rnewson@apache.org
rnewson in #cloudant or #couchdb
@rnewson
WHAT WE TALK ABOUT WHEN WE TALK ABOUT SCALING

• Horizontal scaling: more servers creates more capacity

• Transparent to the application: adding more capacity should not affect the business logic of the application.

• No single point of failure.

Pseudo Scalars

http://adam.heroku.com/past/2009/7/6/sql_databases_dont_scale/
BIGCOUCH = COUCH+SCALING

• Horizontal Scalability
  Easily add storage capacity by adding more servers
  Computing power (views, compaction, etc.) scales with more servers

• No single point of failure (SPOF)
  Any node can handle any request
  With quorum, individual nodes can come and go

• Transparent to the Application
  All clustering operations take place “behind the curtain”
  ‘looks’ like a single server instance of Couch, just with more awesome
  asterisks and caveats discussed later
PUT http://rnewson.cloudant.com/dbname/blah?w=2

Load Balancer

hash(blah) = E

N=3
W=2
R=2

• Clustering in a ring (a la Dynamo)
• Any node can handle a request
• O(1) lookup
• Quorum system (N, R, W)
• Views distributed like documents
• Distributed Erlang
• Masterless
• Shopping List

Dependencies
  • Erlang (R13B03+)
  • ICU
  • Spidermonkey
  • LibCurl
  • OpenSSL
  • make
  • Python

brew install erlang icu4c spidermonkey
brew ln icu4c

git clone https://github.com/cloudant/bigcouch.git
cd bigcouch
./configure
make dev
BUILDING YOUR FIRST CLUSTER

dev1
rel/dev1/bin/bigcouch

dev2
rel/dev2/bin/bigcouch

dev3
rel/dev3/bin/bigcouch

Join the cluster

```
curl localhost:15986/nodes/dev2@127.0.0.1 -X PUT -d '{}'
curl localhost:15986/nodes/dev3@127.0.0.1 -X PUT -d '{}'
```

... and verify

```
curl http://localhost:15984/_membership
```
• BigCouch Clusters are governed by 4 parameters

Q: Number of shards per DB
N: Number of redundant copies of each document
R: Read quorum constant
W: Write quorum constant

(NB: Also consider the number of nodes in a cluster)

For the next few examples, consider a 5 node cluster
• Q: The number of shards over which a DB will be spread
  consistent hashing space divided into Q pieces
  Specified at DB creation time
  possible for more than one shard to live on a node
  Documents deterministically mapped to a shard
  More shards = faster view builds
  Less shards = better memory management
- **N**: The number of redundant copies of each document

  Choose \( N > 1 \) for fault-tolerant cluster
  Default specified at DB creation
  Each shard is copied \( N \) times
  Recommend \( N > 2 \)
• \( W \): The number of document copies that must be saved before a document is “written”

\( W \) must be less than or equal to \( N \)

\( W=1 \), maximise throughput

\( W=N \), maximise consistency

Allow for “202” created response

Can be specified at write time
• **R**: The number of identical document copies that must be read before a read request is ok

R must be less than or equal to N
R=1, minimise latency
R=N, maximise consistency
Can be specified at query time
• So far, so good, but what about secondary indexes?
Views are built locally on each node, for each DB shard
Merge sort at query time using exactly one copy of each shard
Run a final re-reduce on each row if the view has a reduce

• _changes feed works similarly, but has no global ordering
Sequence numbers converted to JSON to encode more information
API AND CAVEATS

• **Clustered API**
  By default listens on port 5984
  All single-doc operations and most view operations

• **What’s Different?**
  update_seq value is now opaque JSON
  rereduce=true always called on reduce views
  no temporary views
  no all_or_nothing: true

• **‘Backdoor’ Access**
  Able to reach a single node (i.e. at the shard level)
  By default listens on port 5986
  Allows you to trigger local view updates, compactions, etc.
The BigCouch Stack

- CHTTPD
- Fabric
- REXI
- Mem3

Embedded CouchDB
Mochiweb, Spidermonkey, etc.
CHTTPD / FABRIC

• Chttpd
Cut-n-paste of couch_httpd, but using fabric for all data access

• Fabric
OTP library application (no processes) responsible for clustered versions of CouchDB core API calls
Quorum logic, view merging, etc.
Provides a clean Erlang interface to BigCouch
• Maintains the shard mapping for each clustered database in a node-local CouchDB database

• Changes in the node registration and shard mapping databases are automatically replicated to all cluster nodes
• BigCouch makes a large number of parallel RPCs
• Erlang RPC library not designed for heavy parallelism
  promiscuous spawning of processes  
  responses directed back through single process on remote node  
  requests block until remote ‘rex’ process is monitored
• Rexi removes some of the safeguards in exchange for lower latencies
  no middlemen on the local node  
  remote process responds directly to client  
  remote process monitoring occurs out-of-band
FUTURE
BIGCOUCH HAS NO FUTURE
THE FUTURE IS COUCHDB
WE’RE MERGING
THE MERGE

• Release BigCouch 0.5.0
• Release Apache CouchDB 1.3.0
• Merge them
• Release Apache CouchDB 2.0.0 (couchdb strikes back)
SUMMARY

• BigCouch: putting the ‘C’ back in CouchDB
• Consistent hashing for database sharding (a la Dynamo)
• True horizontal scalability with CouchDB
• Download now and get started
  https://github.com/cloudant/bigcouch.git