Always-On NoSQL: Speed, Scale and Flexibility

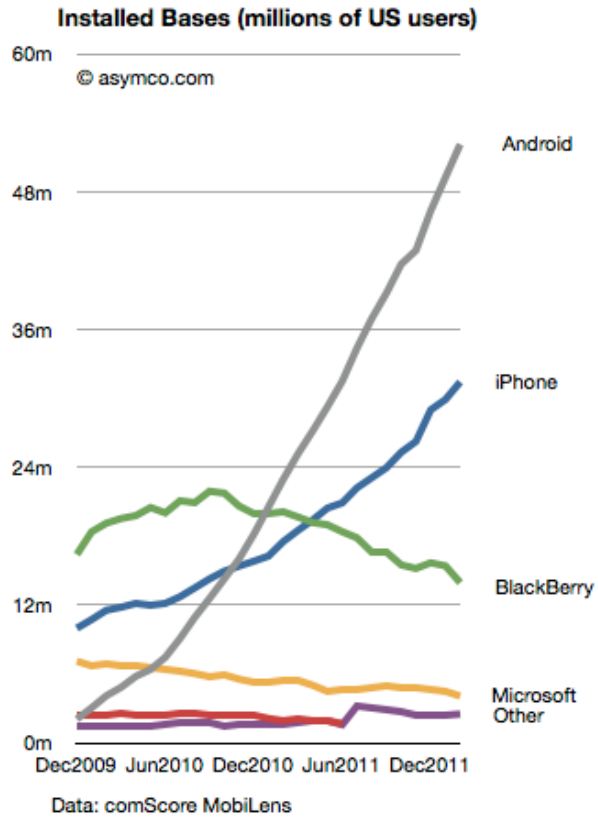
Chris Anderson
@jchris

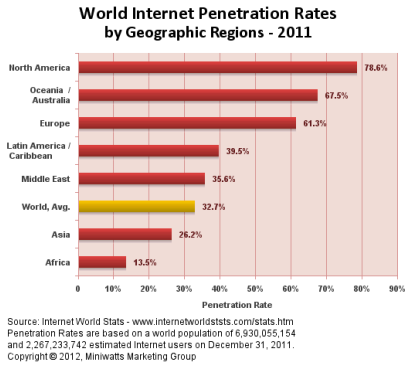


CouchBase

Growth is the New Reality

- 2.2 Billion internet users
- 50% Americans use smartphones
- Your app can grow overnight
- Are you ready?



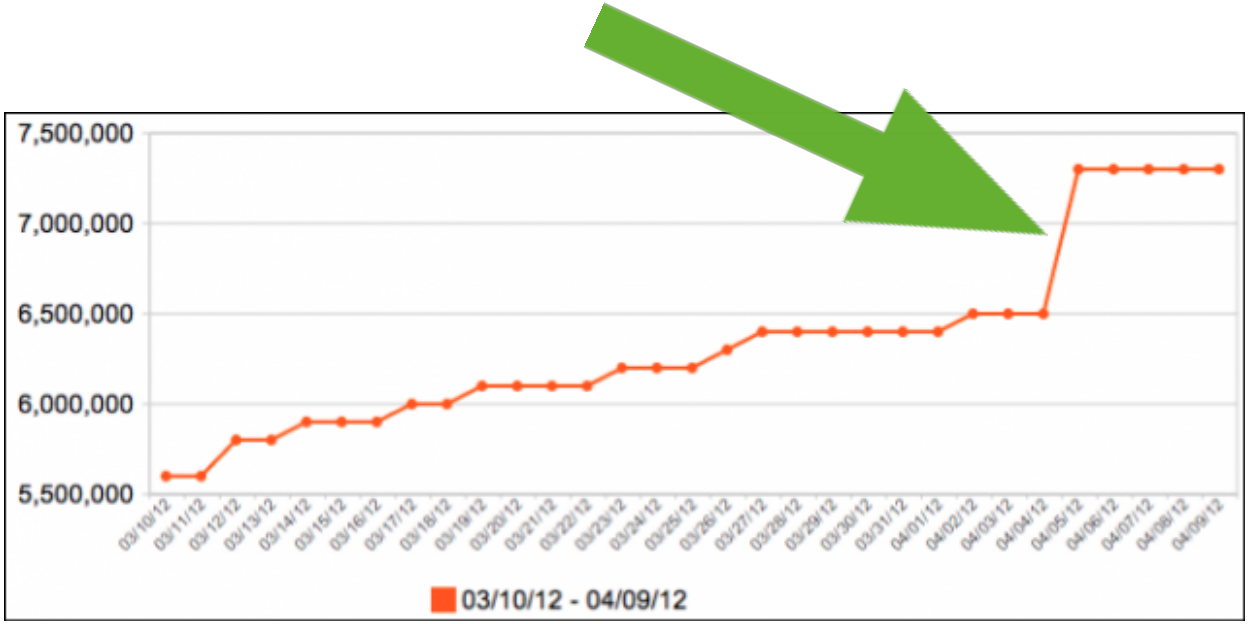




Instagrowth: Android Launch

Example

- Instagram gained nearly 1 million users overnight when they expanded to Android



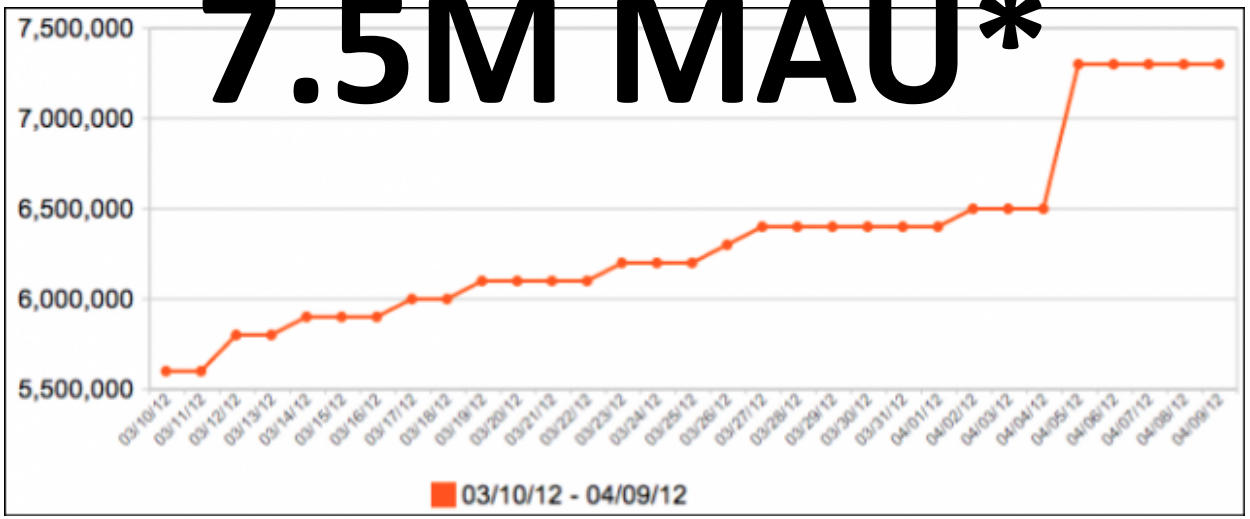
Instagrowth: Android Launch

Example

1 Instagram

=

7.5M MAU*

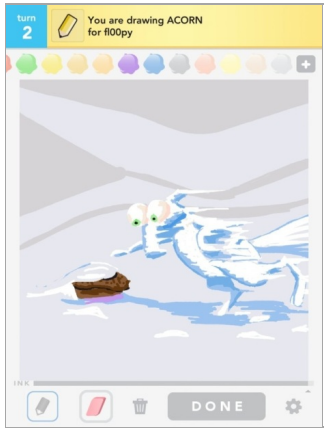


Draw Something - Social Game









35 million monthly active users in 1 month

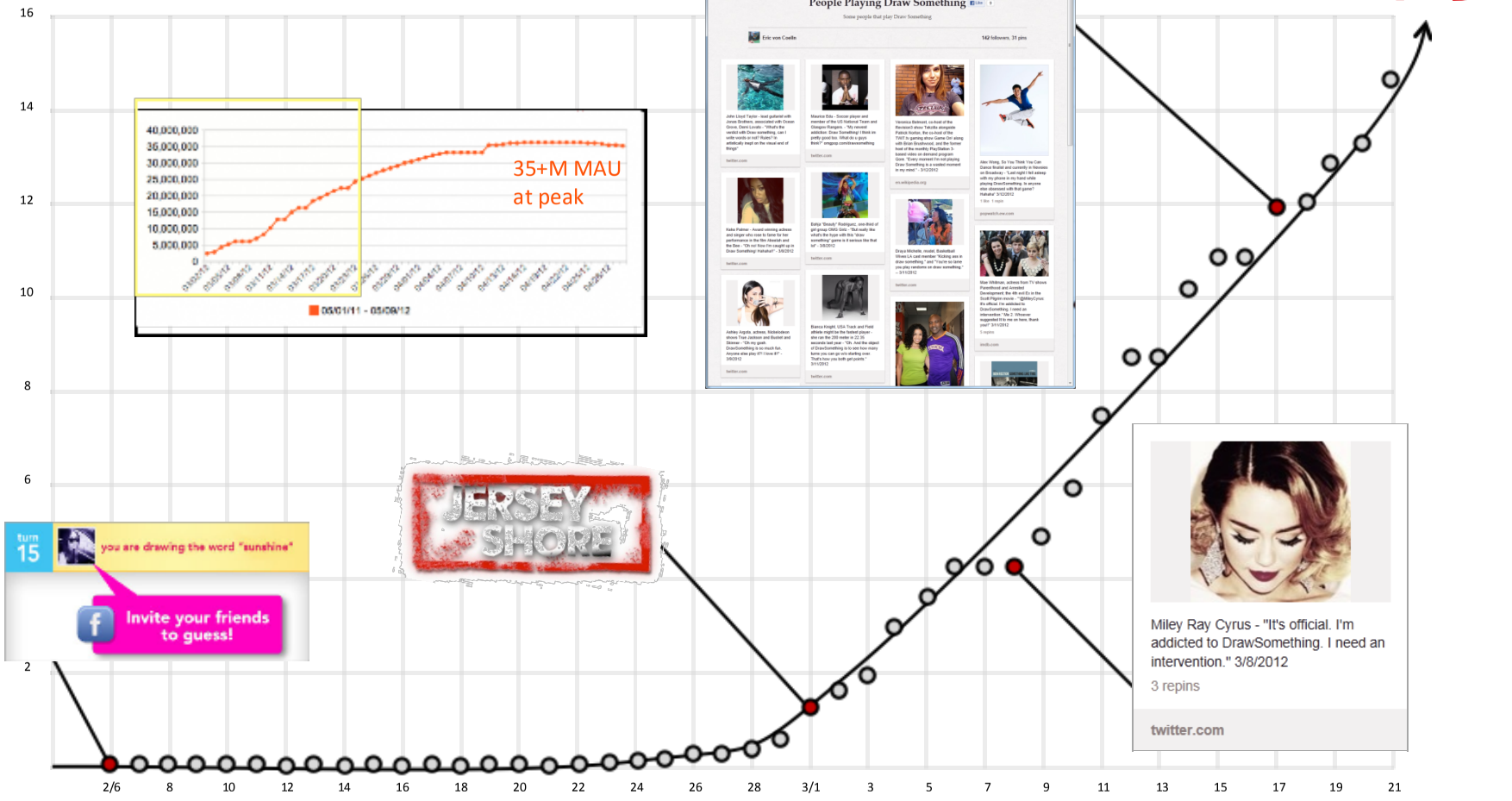
about 5 Instagrams

(Instagram today is waaaay more than 1 Instagram)

Goes Viral 3 Weeks after Launch

Draw Something by OMGPOP

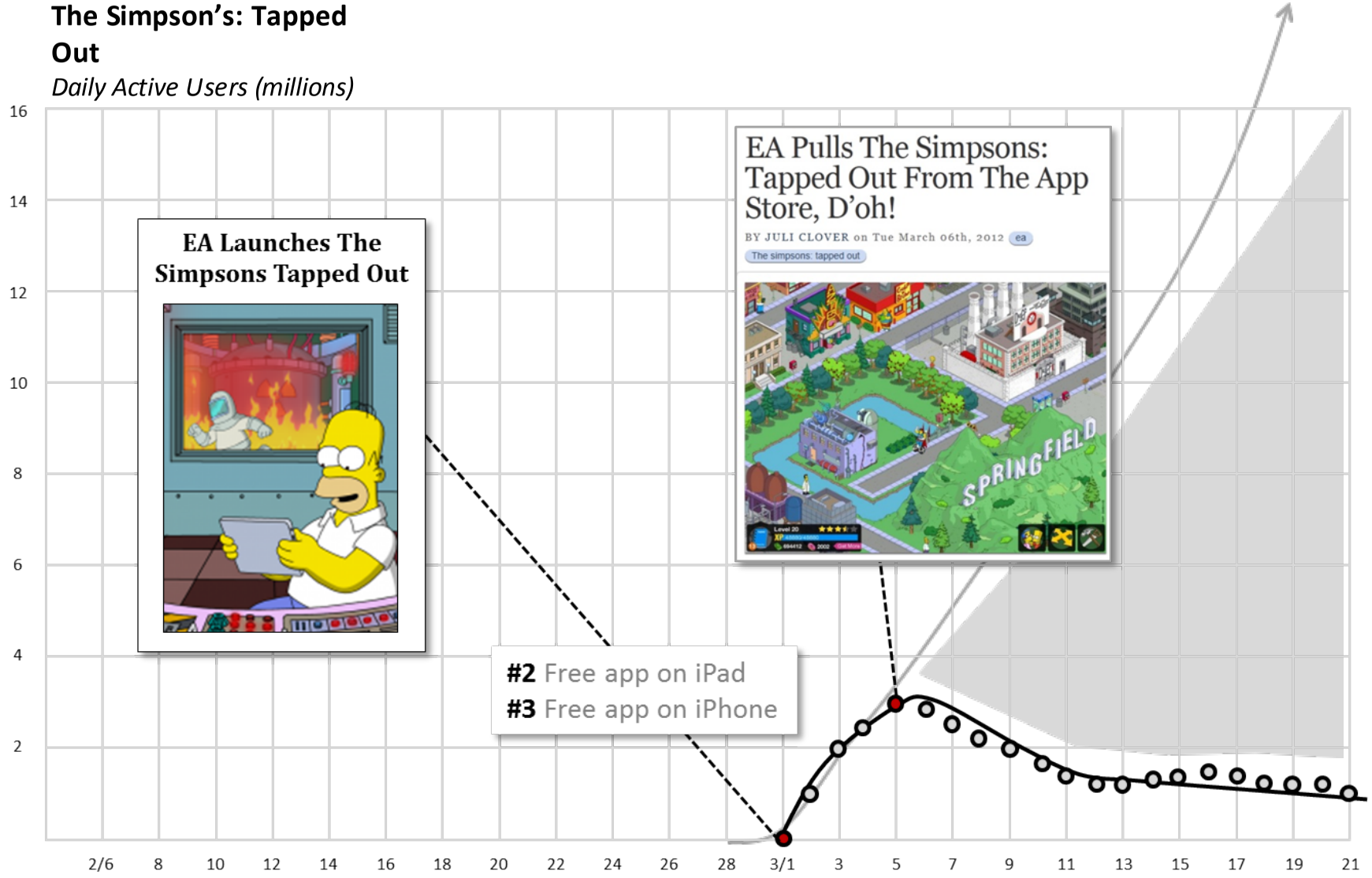
Daily Active Users (millions)



By Contrast, at 1/2 an Instagram

The Simpson's: Tapped Out

Daily Active Users (millions)



Couchbase handles real world scale



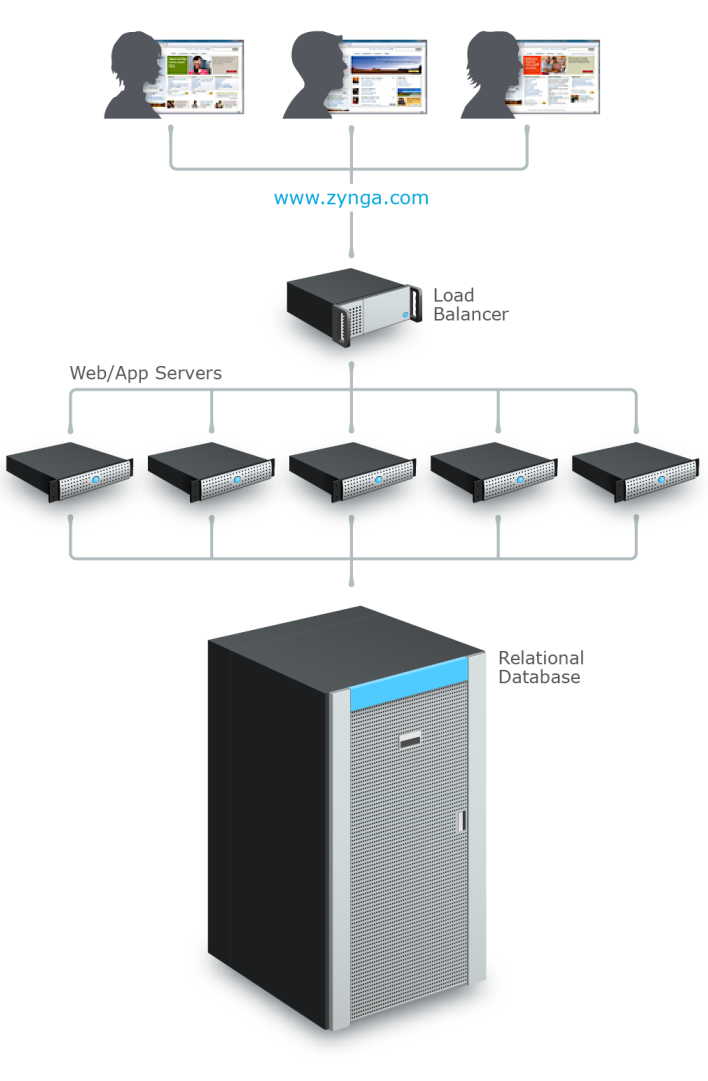
GETTING IT RIGHT

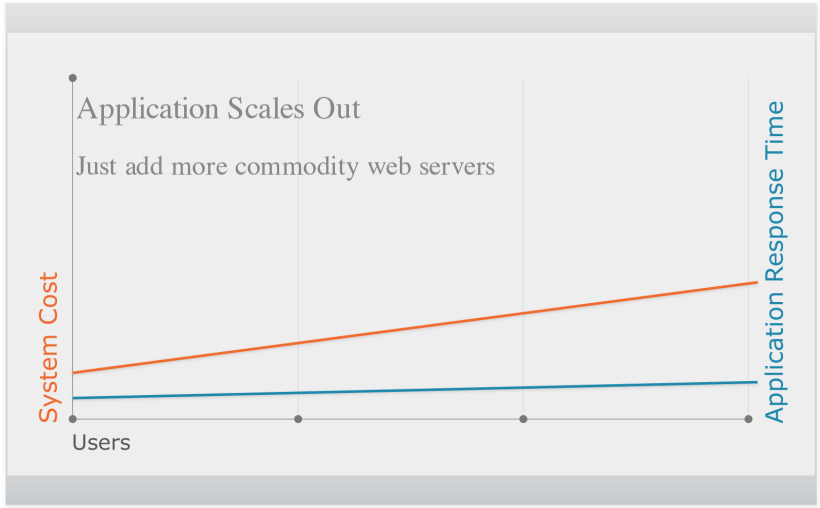
Scalable Data Layer

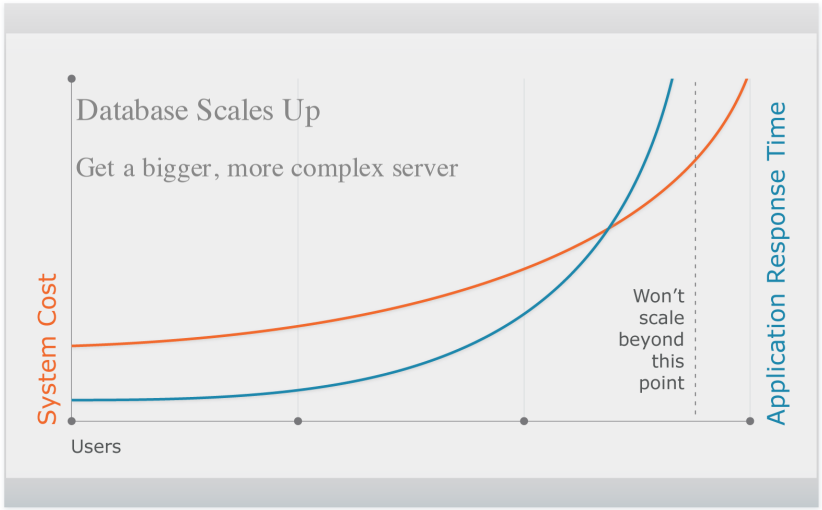


- On-demand cluster sizing
  - Grow *or shrink* with workload
- Easy node provisioning
  - All nodes are the same
- Multi-master Cross-Datacenter Replication
  - For a fast and reliable user experience worldwide
- Effective Auto-sharding
  - Should avoid cluster hot spots

Old School Hits a Scale Wall



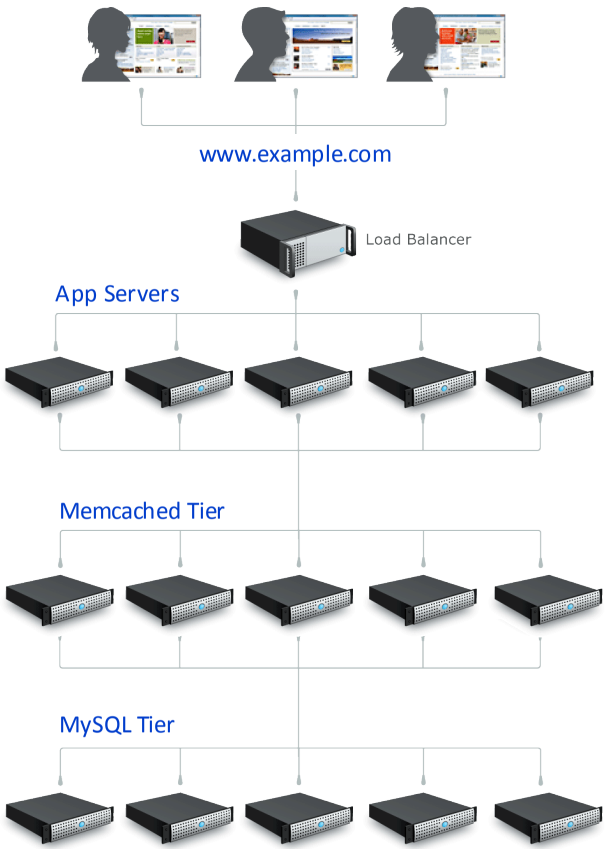




- Expensive & disruptive sharding, doesn't perform at web scale

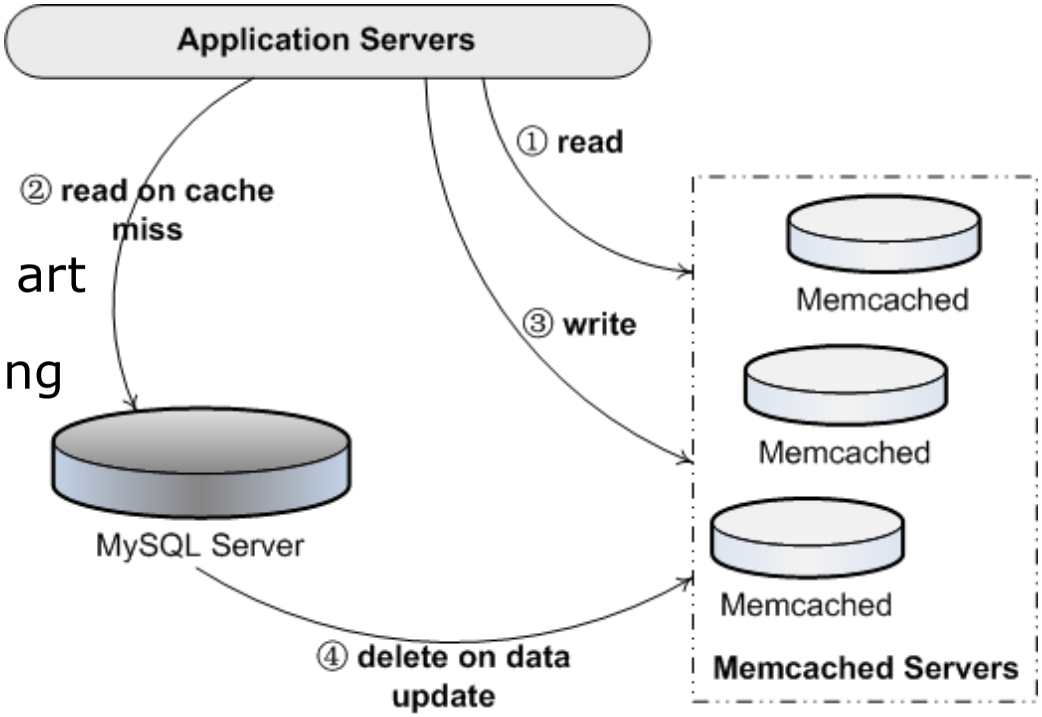
Traditional MySQL + Memcached Architecture

- Run as many MySQL machines as you need
- Data sharded evenly across the machines using client code
- Memcached used to provide faster response time for users and reduce load on the database



Limitations of MySQL + Memcached

- To scale you need to start using MySQL more simply
- Scale by hand
- Replication / Sharding is a black art
- Code overhead to manage keeping memcache and mysql in sync
- Lots of components to deploy



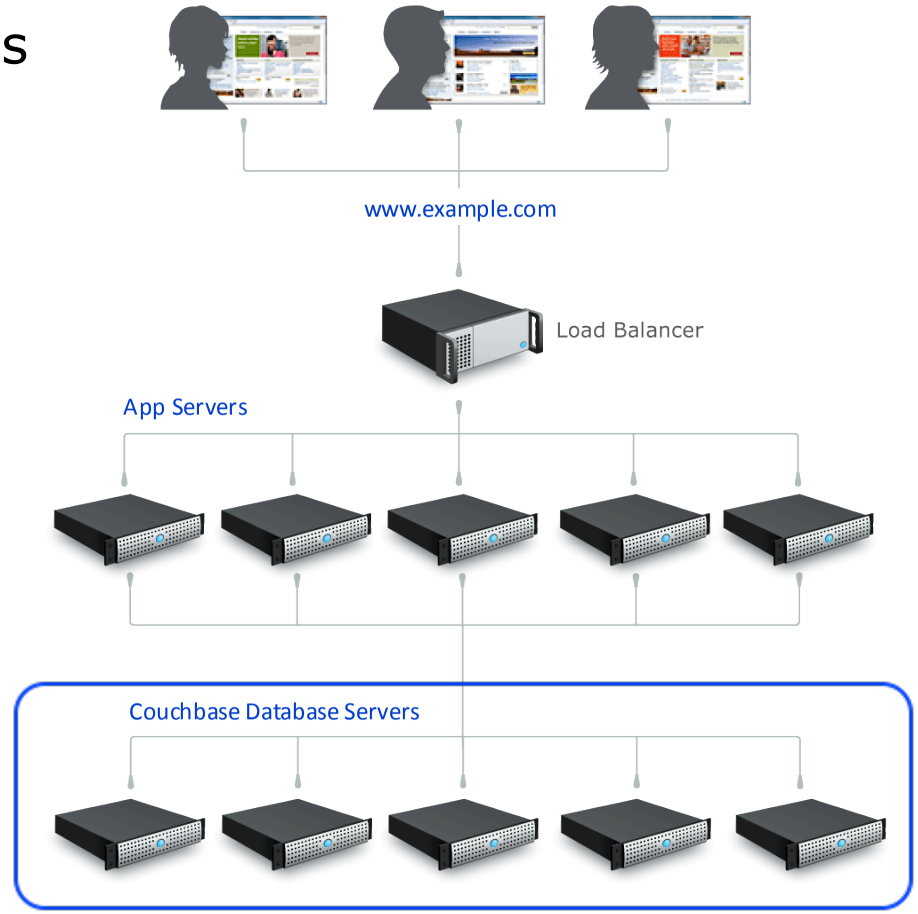


Learn From Others - This Scenario Costs Time and Money. Scaling SQL is potentially disastrous when going Viral: very risky time for major code changes and migrations... you have no Time when skyrocketing up.

NoSQL Architectural Promise

- High Performance data access
- Scale Up/Down Horizontally
- 24x7x365 Always-On Availability
- Flexible Schema Document Model





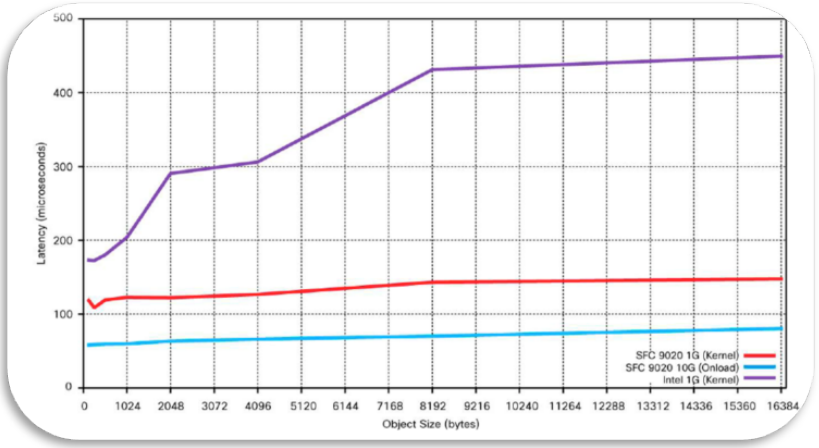
HELLO COUCHBASE

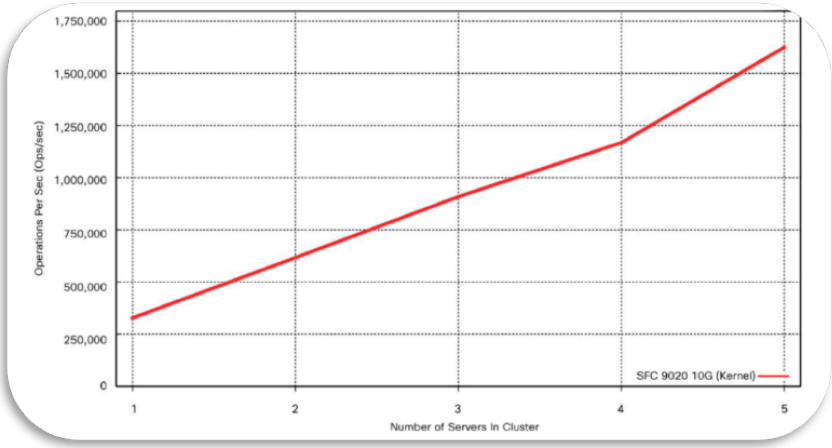
Couchbase Server 2.0 beta



(Really) High Performance

Cisco and Solarflare benchmark of Couchbase Server





Latency

less than 1/2 ms

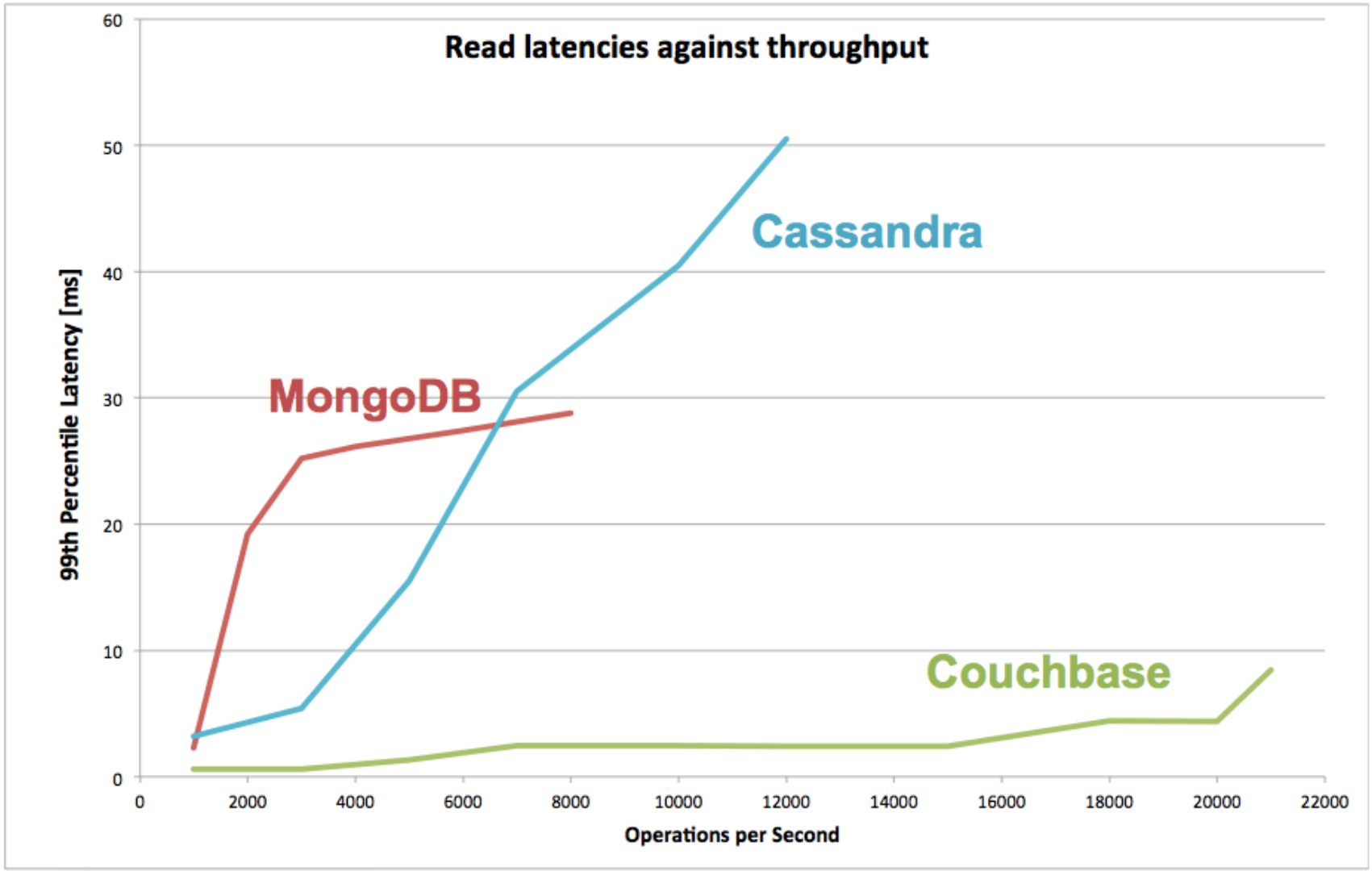
Throughput

grows linearly with cluster size

5 Nodes -- 1.75M operations per second

How fast?

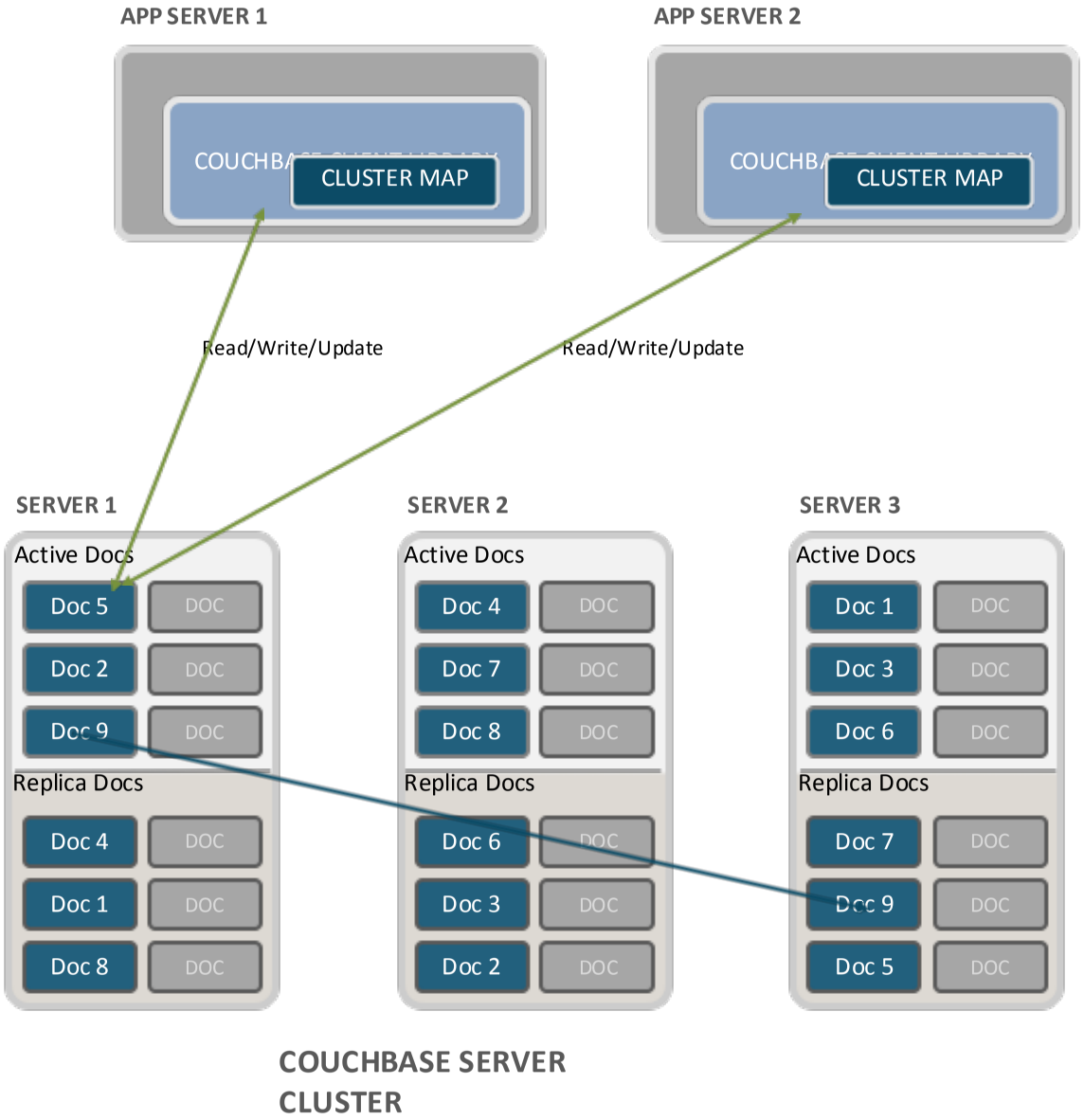
Reads (99th percentile)



ALTOROS SYSTEMS

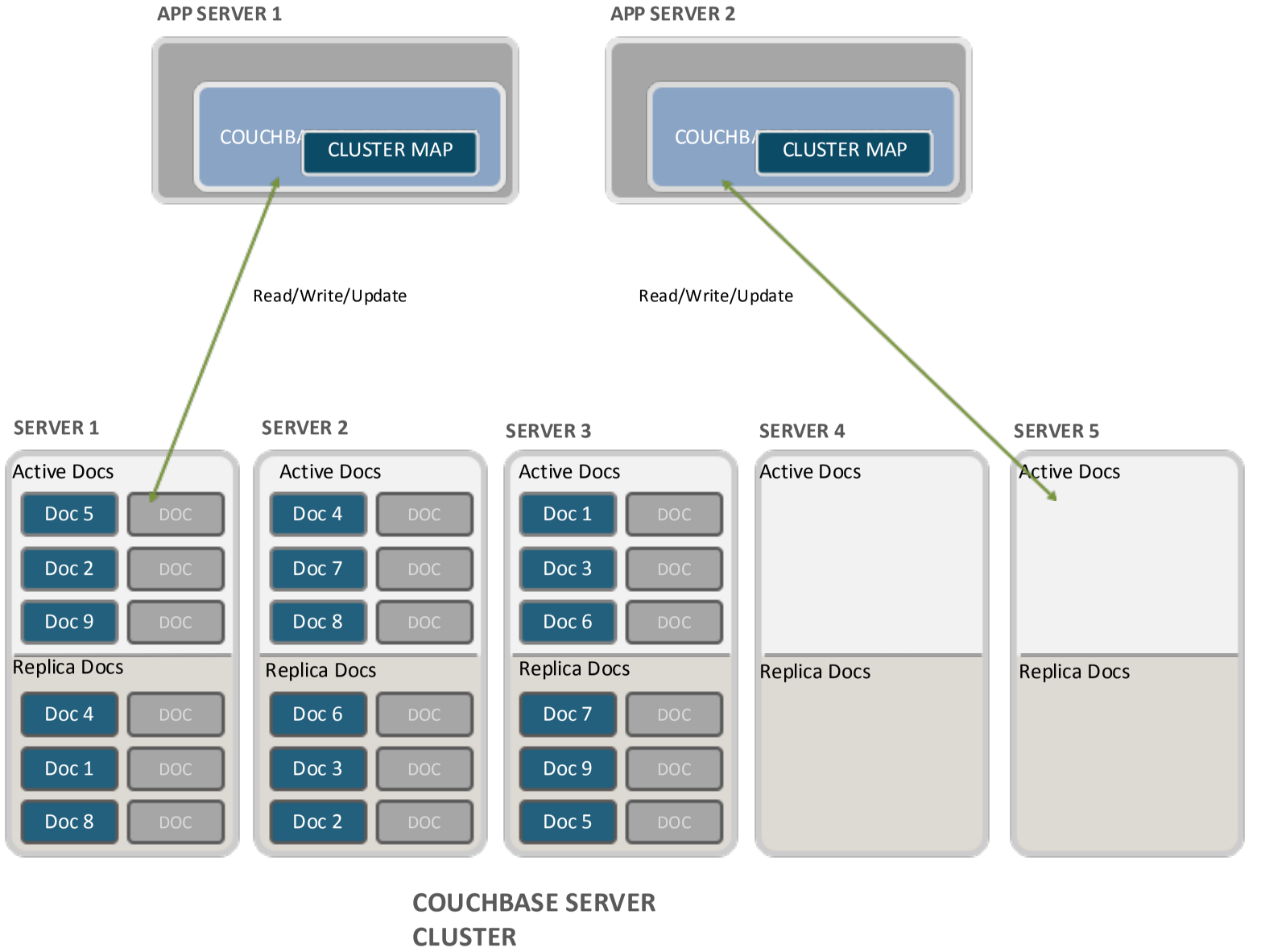
http://www.slideshare.net/renatko/couchbase-performance-benchmarking

Couchbase Server Basic Operation



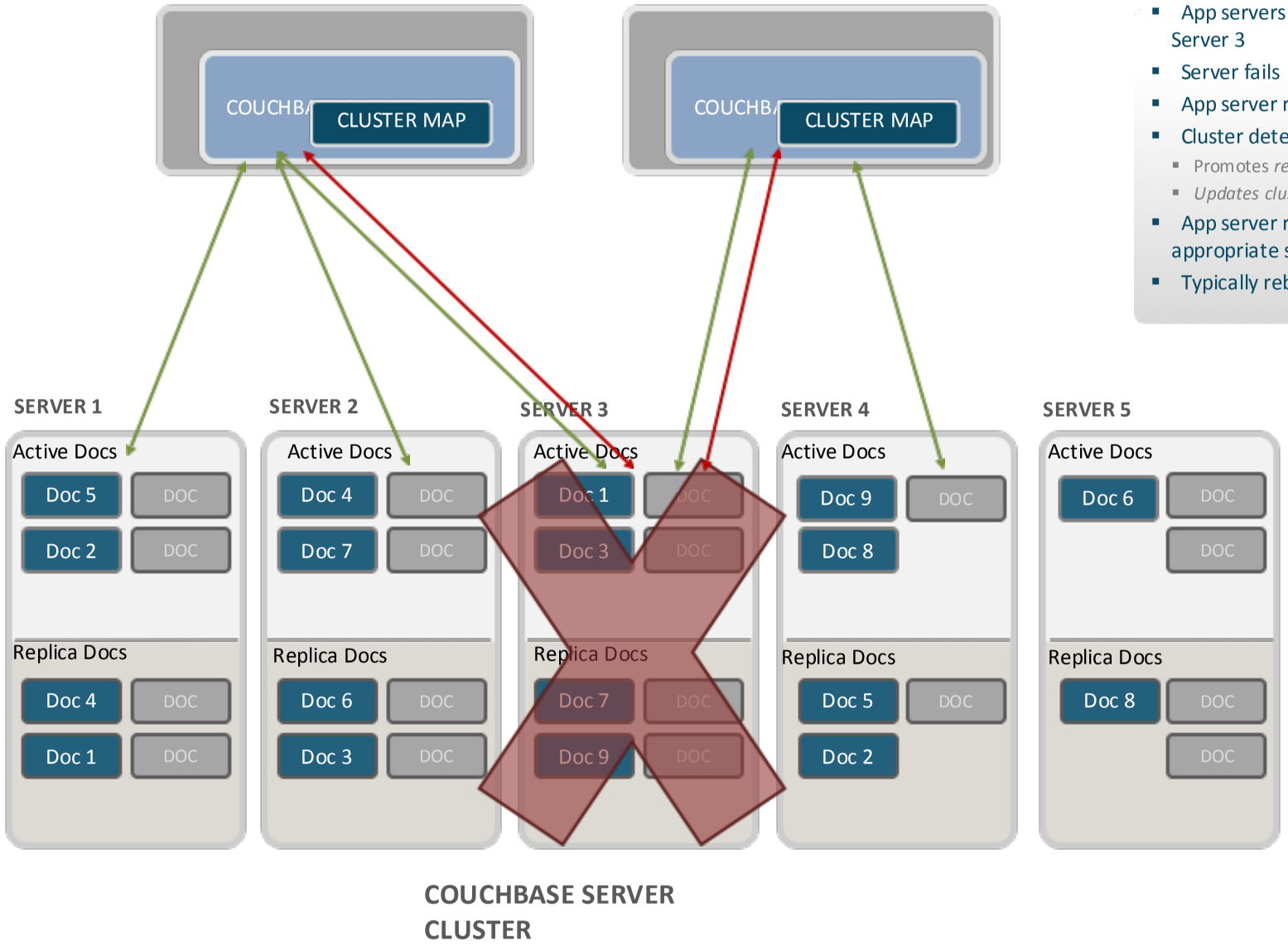
- Docs distributed evenly across servers in the cluster
- Each server stores both *active* & *replica* docs
  - Only one server active at a time
- Client library provides app with simple interface to database
- Cluster map provides map to which server doc is on
  - App never needs to know
- App reads, writes, updates docs
- Multiple App Servers can access same document at same time

Add Nodes to the Cluster



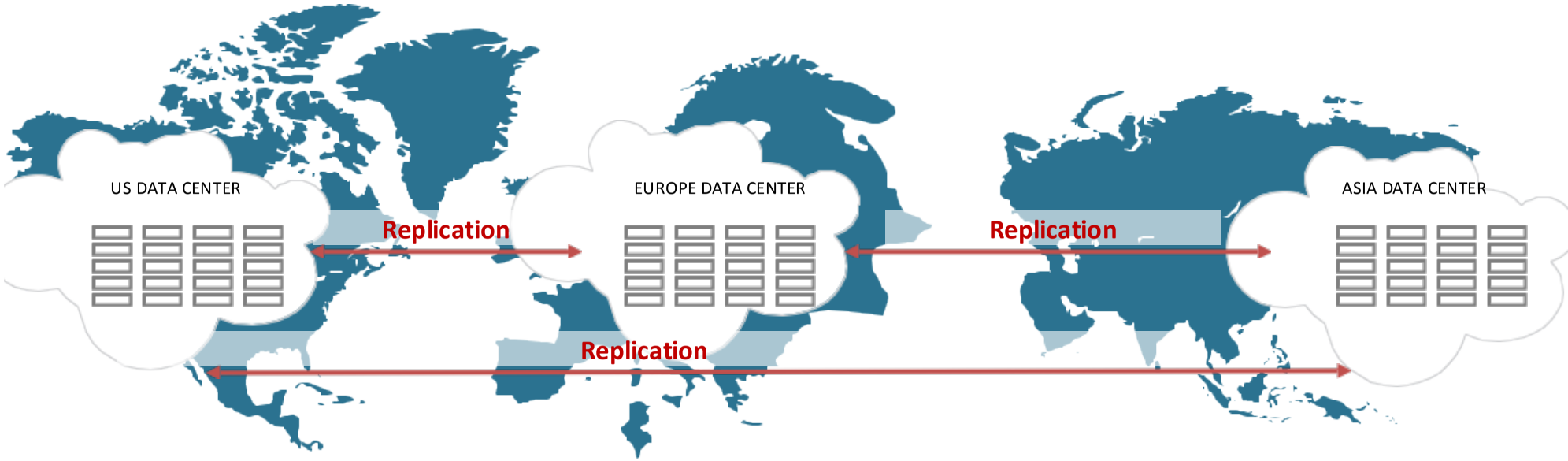
- Two servers added to cluster
  - One-click operation
- Docs automatically rebalanced across cluster
  - Even distribution of docs
  - Minimum doc movement
- Cluster map updated
- App database calls now distributed over larger # of servers

Fail Over Node



- App servers happily accessing docs on Server 3
- Server fails
- App server requests to server 3 fail
- Cluster detects server has failed
  - Promotes *replicas* of docs to *active*
  - *Updates cluster map*
- App server requests for docs now go to appropriate server
- Typically rebalance would follow

Cross Data Center Replication



- Data close to users
- Multiple locations for disaster recovery
- Independently managed clusters serving local data

Built for Production



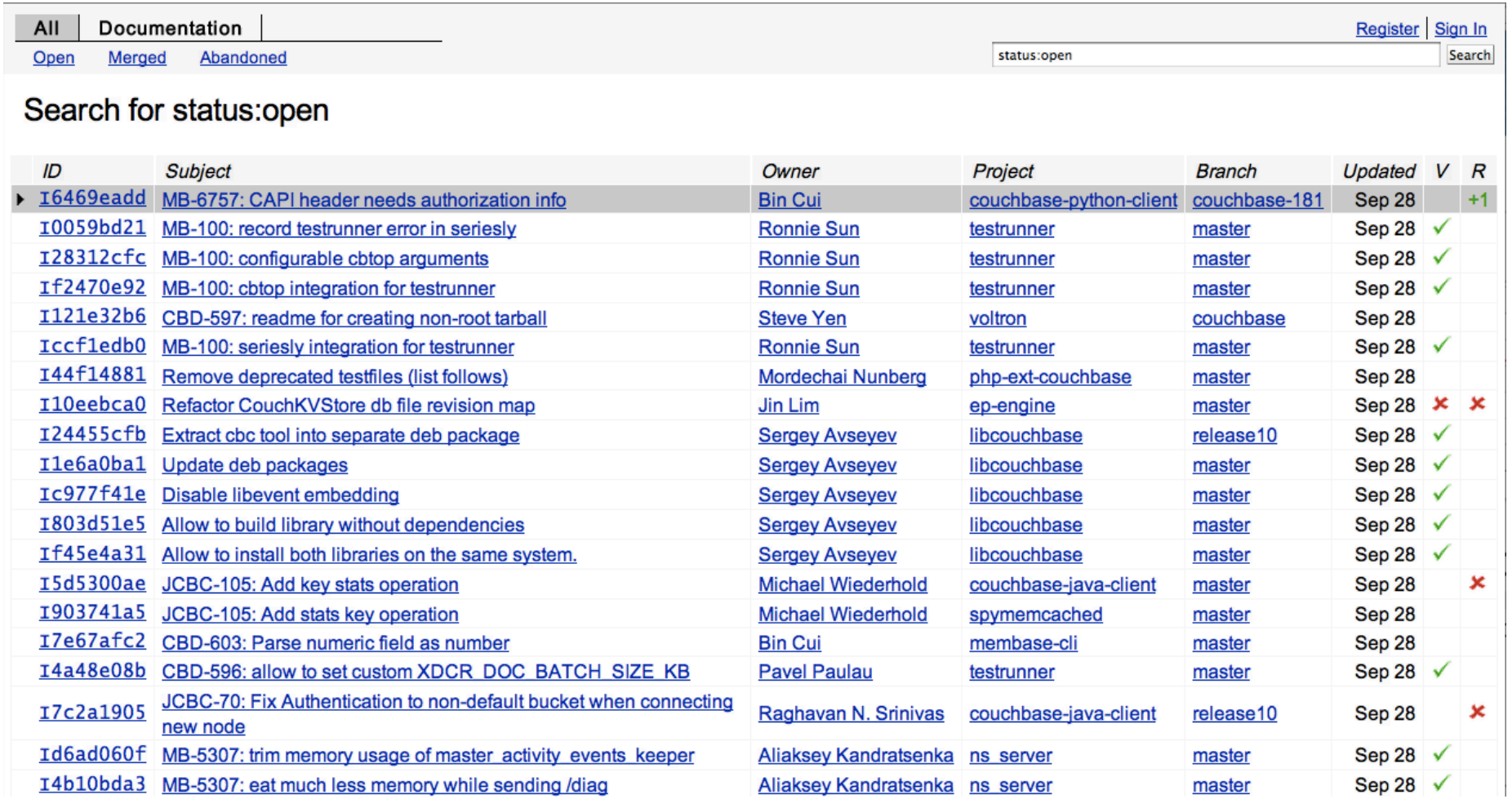
War Story: EBS Outage



- Suddenly, disk writes all began to time out
- Many services experienced outages:
  - FourSquare, Reddit, Quora, among others
- With memory buffered writes, a scalable data layer keeps working
  - When EBS came back online, Couchbase wrote all the updated data to disk without missing a beat.

Open Source (Apache 2.0 License)

& open development: http://review.couchbase.org/



All | Documentation

Open Merged Abandoned

Register | Sign In

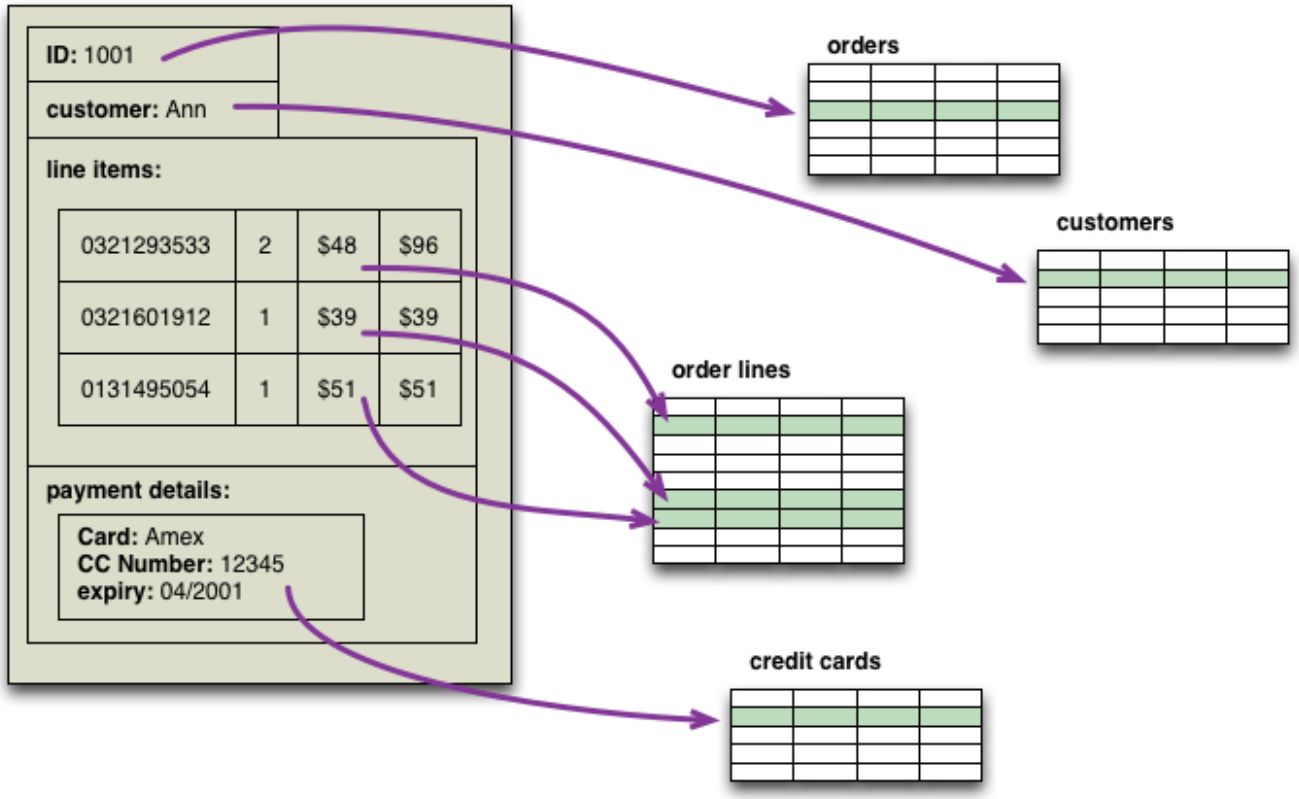
status:open Search

Search for status:open

	ID	Subject	Owner	Project	Branch	Updated	V	R
▸	I6469eadd	MB-6757: CAPI header needs authorization info	Bin Cui	couchbase-python-client	couchbase-181	Sep 28		+1
	I0059bd21	MB-100: record testrunner error in seriesly	Ronnie Sun	testrunner	master	Sep 28	✓	
	I28312cfc	MB-100: configurable cbtop arguments	Ronnie Sun	testrunner	master	Sep 28	✓	
	If2470e92	MB-100: cbtop integration for testrunner	Ronnie Sun	testrunner	master	Sep 28	✓	
	I121e32b6	CBD-597: readme for creating non-root tarball	Steve Yen	voltron	couchbase	Sep 28		
	Iccf1edb0	MB-100: seriesly integration for testrunner	Ronnie Sun	testrunner	master	Sep 28	✓	
	I44f14881	Remove deprecated testfiles (list follows)	Mordechai Nunberg	php-ext-couchbase	master	Sep 28		
	I10eebca0	Refactor CouchKVStore db file revision map	Jin Lim	ep-engine	master	Sep 28	✗	✗
	I24455cfb	Extract cbc tool into separate deb package	Sergey Avseyev	libcouchbase	release10	Sep 28	✓	
	I1e6a0ba1	Update deb packages	Sergey Avseyev	libcouchbase	master	Sep 28	✓	
	Ic977f41e	Disable libevent embedding	Sergey Avseyev	libcouchbase	master	Sep 28	✓	
	I803d51e5	Allow to build library without dependencies	Sergey Avseyev	libcouchbase	master	Sep 28	✓	
	If45e4a31	Allow to install both libraries on the same system.	Sergey Avseyev	libcouchbase	master	Sep 28	✓	
	I5d5300ae	JCBC-105: Add key stats operation	Michael Wiederhold	couchbase-java-client	master	Sep 28		✗
	I903741a5	JCBC-105: Add stats key operation	Michael Wiederhold	spymemcached	master	Sep 28		
	I7e67afc2	CBD-603: Parse numeric field as number	Bin Cui	membase-cli	master	Sep 28		
	I4a48e08b	CBD-596: allow to set custom XDCR_DOC_BATCH_SIZE_KB	Pavel Paulau	testrunner	master	Sep 28	✓	
	I7c2a1905	JCBC-70: Fix Authentication to non-default bucket when connecting new node	Raghavan N. Srinivas	couchbase-java-client	release10	Sep 28		✗
	Id6ad060f	MB-5307: trim memory usage of master_activity_events_keeper	Aliaksey Kandratsenka	ns_server	master	Sep 28	✓	
	I4b10bda3	MB-5307: eat much less memory while sending /diag	Aliaksey Kandratsenka	ns_server	master	Sep 28	✓	

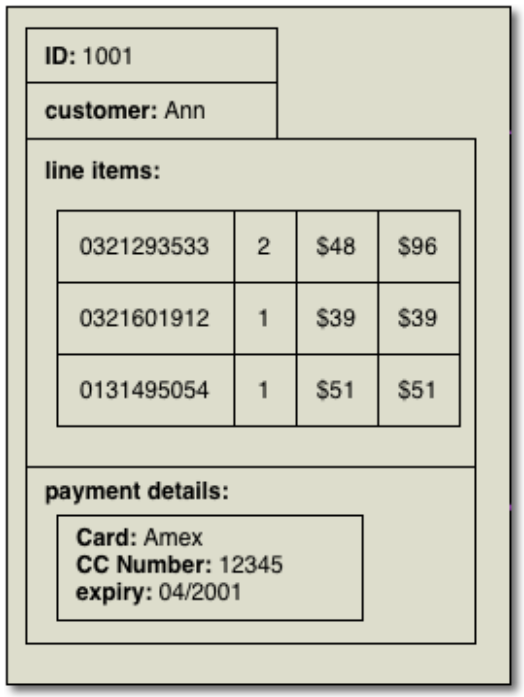
JSON DOCUMENT DATABASE

Couchbase Server is a Document Database



http://martinfowler.com/bliki/AggregateOrientedDatabase.html

Document Database



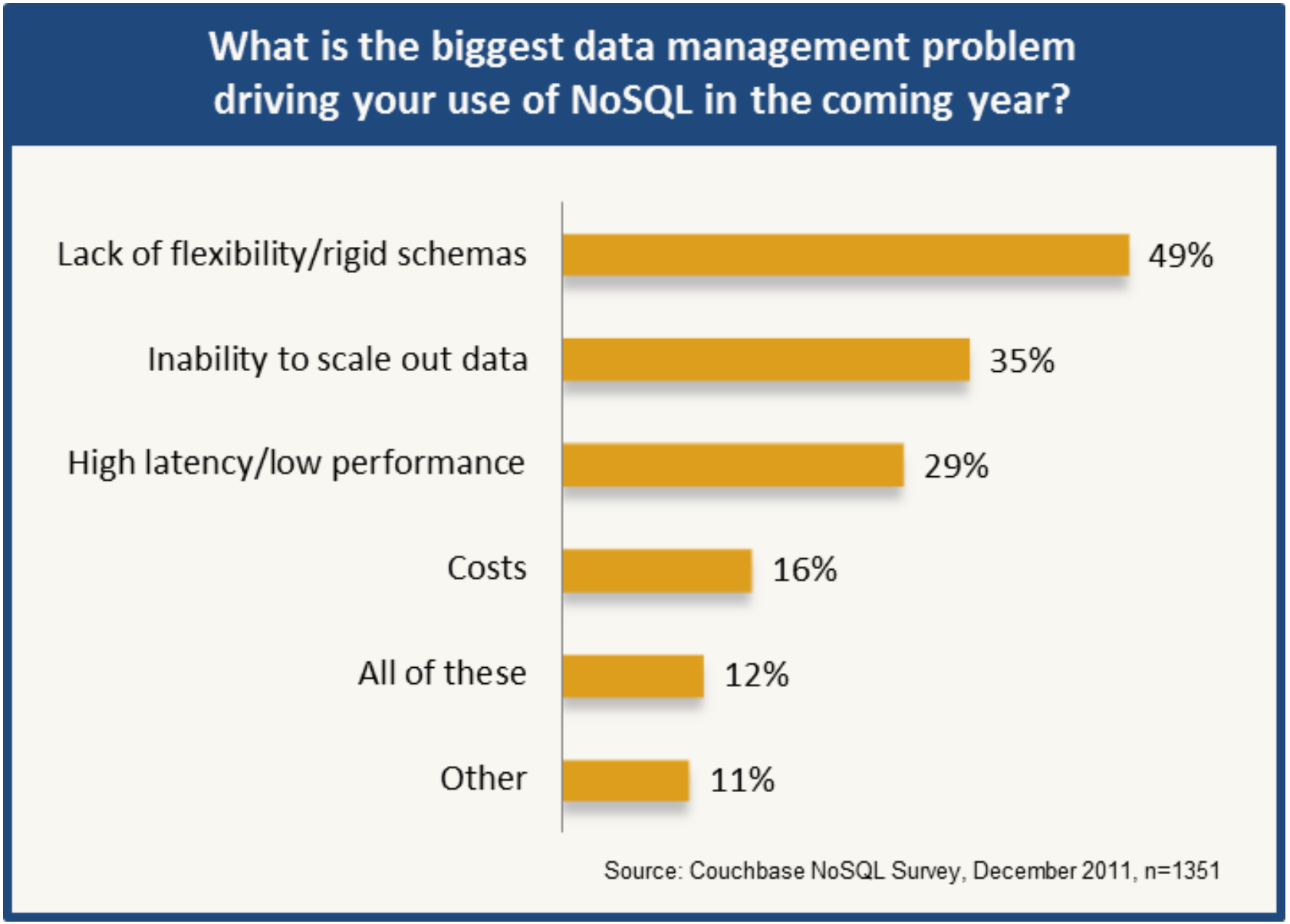


```
o::1001
{
uid: ji22jd,
customer: Ann,
line_items: [
{ sku: 0321293533, quan: 3,  unit_price: 48.0 },
{ sku: 0321601912, quan: 1, unit_price: 39.0 },
{ sku: 0131495054, quan: 1, unit_price: 51.0 }
],
payment: { type: Amex, expiry: 04/2001,
last5: 12345 }
}
```

This synergy between the programming model and the distribution model is very valuable. It **allows the database to use its knowledge of how the application programmer clusters the data** to help performance across the cluster.

http://martinfowler.com/bliki/AggregateOrientedDatabase.html

Developers <3 JSON



LET'S GET POST-RELATIONAL!

JSON Documents

- Maps more closely to external API
- CRUD Operations, lightweight schema

```
{
 “fields” : [“with basic types”, 3.14159, true],
 “like” : “your favorite language”
}
```

- Stored under an identifier key

```
client.set(“mydocumentid”, myDocument);
mySavedDocument = client.get(“mydocumentid”);
```

Meta + Document Body


```
{
  "brewery": "New Belgium Brewing",
  "name": "1554 Enlightened Black Ale",
  "abv": 5.5,
  "description": "Born of a flood...",
  "category": "Belgian and French Ale",
  "style": "Other Belgian-Style Ales",
  "updated": "2010-07-22 20:00:20"
}
```


Document
user data,
can be anything

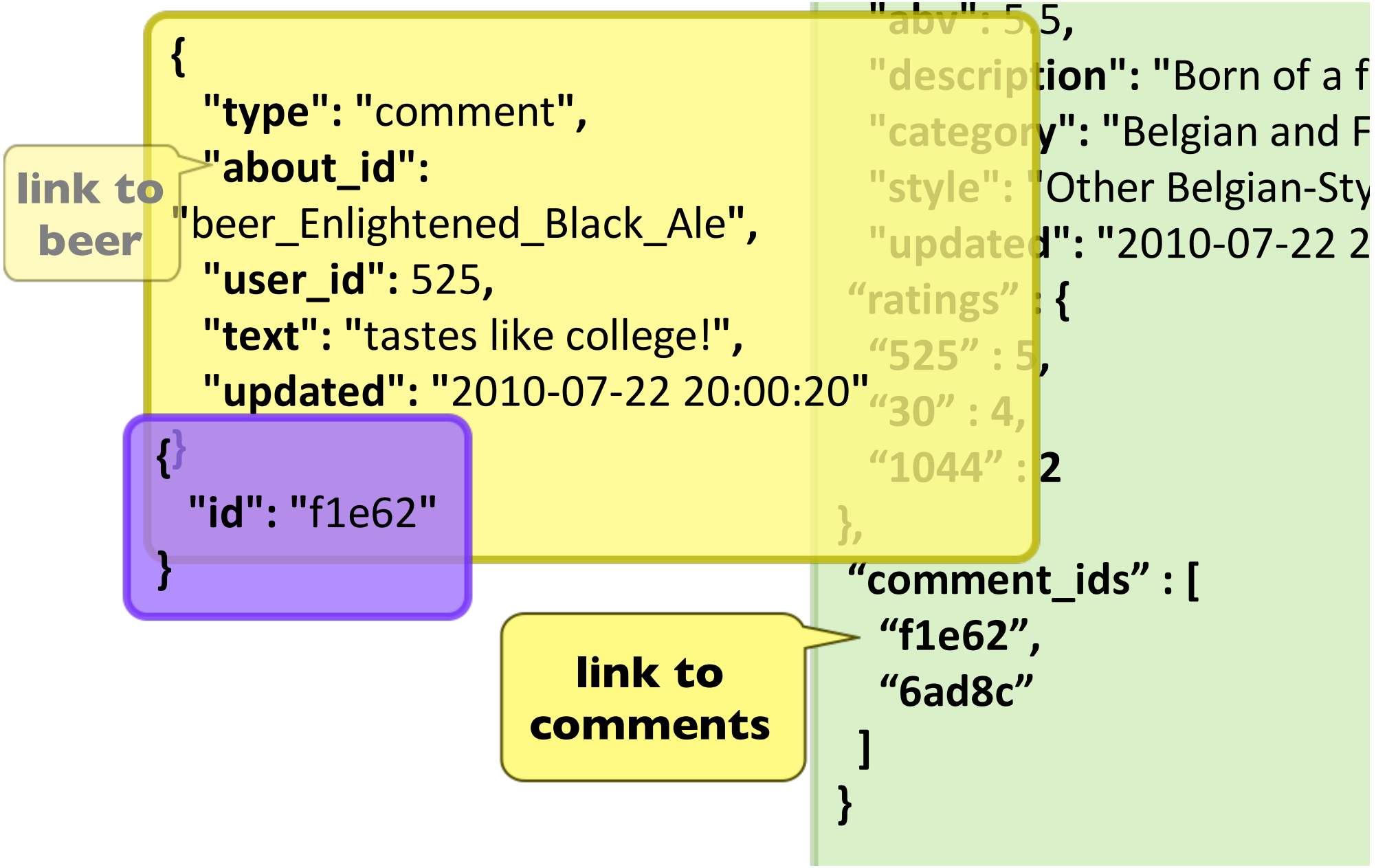
"vintage" date format from an SQL dump
>_<

unique ID


```
{
  "id" : "beer_Enlightened_Black_Ale",
   ...
{
```


Metadata
identifier,
expiration, etc

Add comments to the beer



How to: look up comments from a beer

- SERIALIZED LOOP

```
beer = client.get("beer:A_cold_one");
beer.comment_ids.each { |id|
 comments.push(client.get(id));
}
```

- **FAST MULTI-KEY LOOKUP**

```
beer = client.get("beer:A_cold_one");
comments = client.multiGet(beer.comment_ids)
```

- ASYNC VIEW QUERY

```
comments = client.query("myapp","by_comment_on",
 {:key => "beer:A_cold_one"});
```

figure http://www.ibm.com/developerworks/webservices/library/ws-sdoarch/

Emergent Schema

- The database can handle it
- Your app controls the schema

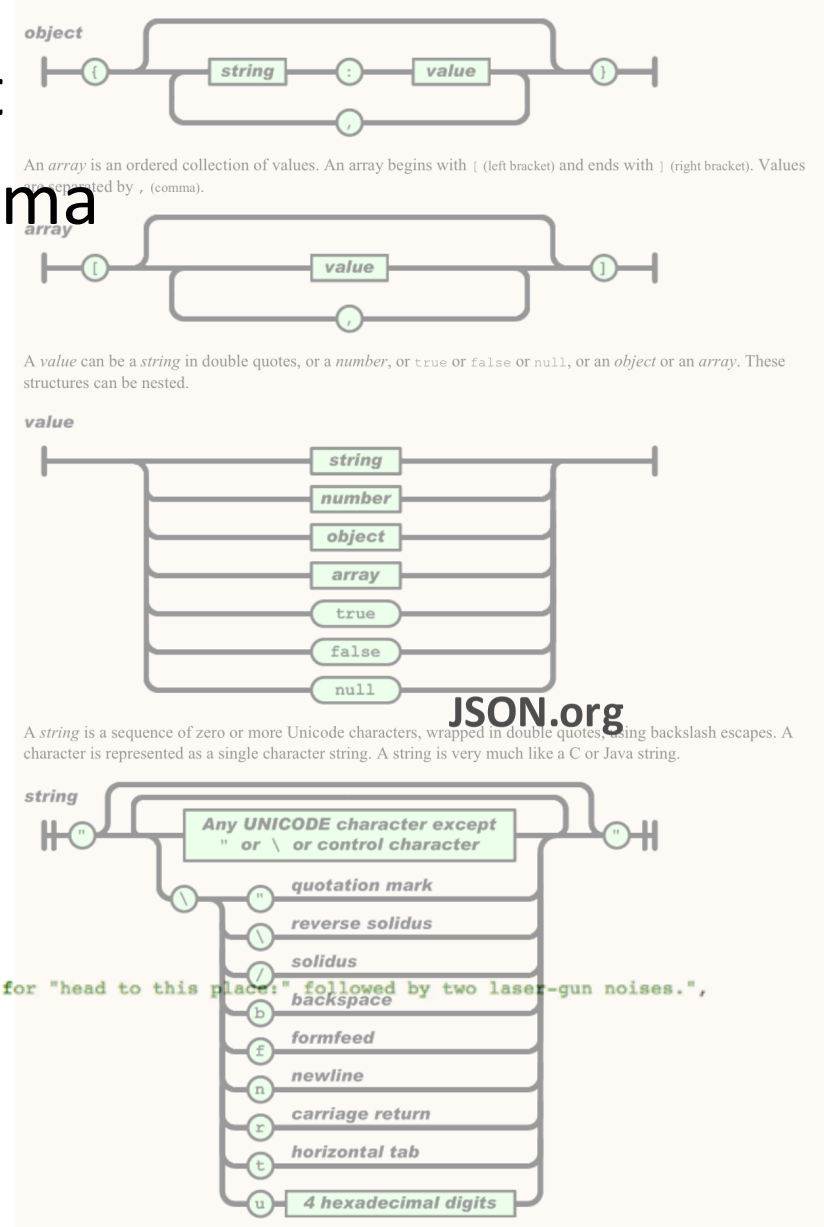
Capture the user's intent"


```
"repo": {
  "url": "https://api.github.com/repos/octocat/Hello-World",
  "git_url": "git://github.com/octocat/Hello-World.git",
  "ssh_url": "git@github.com:octocat/Hello-World.git",
  "svn_url": "https://svn.github.com/octocat/Hello-World",
  "id": 1296269,
  "owner": {
    "login": "octocat",
    "id": 1,
    "avatar_url": "https://github.com/images/error/octocat_happy.gif",
    "gravatar_id": "somehexcode",
    "url": "https://api.github.com/users/octocat"
  },
  "name": "Hello-World",
  "full_name": "octocat/Hello-World",
  "description": "This your first repo!",
  "private": false,
  "fork": false,
  "forks": 9,
  "watchers": 80,
  "size": 108,
  "master_branch": "master",
  "open_issues": 0,
  "pushed_at": "2011-01-26T19:06:43Z",
  "created_at": "2011-01-26T19:01:12Z",
  "updated_at": "2011-01-26T19:14:43Z"
}
},
```


Github API


```
{
  created_at: "Tue Feb 21 21:29:07 +0000 2012",
  id: 172070369035956220,
  id_str: "172070369035956224",
  text: "The "http://" at the beginning of URLs is a command to the browser. It stands for "head to this place:" followed by two laser-gun noises.",
  source: "web",
  truncated: false,
  in_reply_to_status_id: null,
  in_reply_to_status_id_str: null,
  in_reply_to_user_id: null,
  in_reply_to_user_id_str: null,
  in_reply_to_screen_name: null,
- user: {
    id: 63846421,
    id_str: "63846421",
    name: "Brian Sutorius",
```


Twitter API



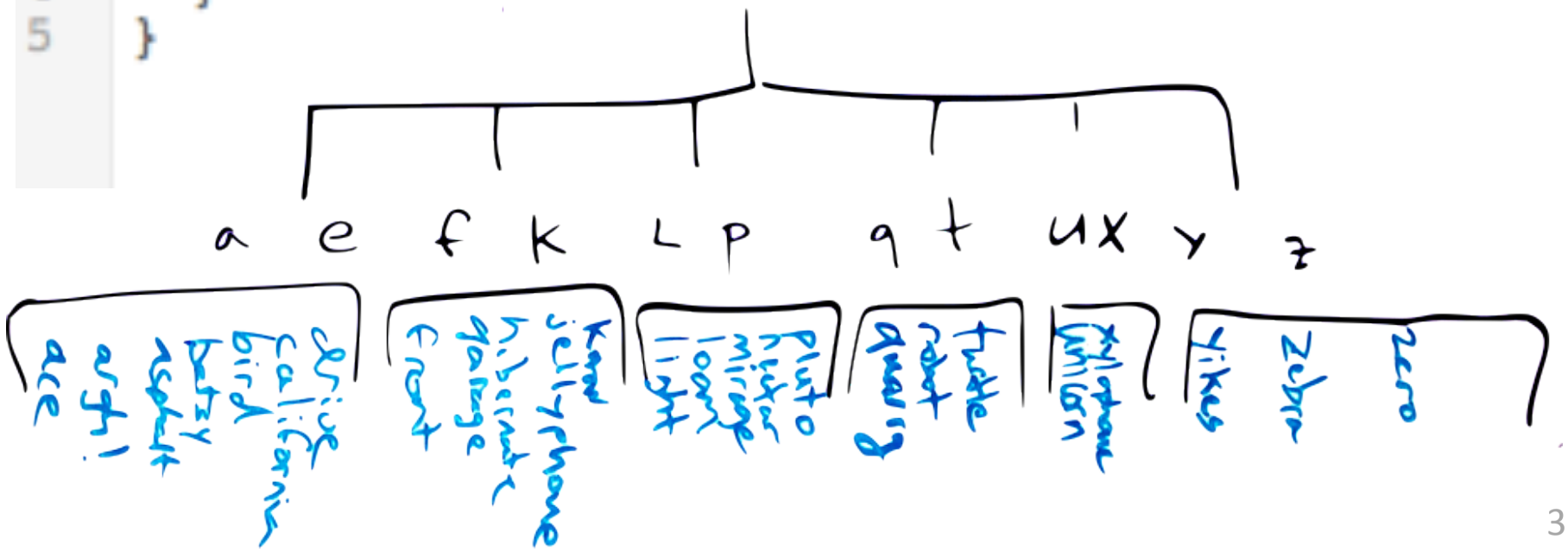
INCREMENTAL MAP-REDUCE FOR REALTIME ANALYTICS

What do you mean "**Incremental**?"

like:

CREATE INDEX city **ON** brewery city;

```
1  function (doc, meta) {
2    if (doc.type == "brewery" && doc.city) {
3      emit(doc.city, null);
4    }
5  }
```



QUERY PATTERN:
FIND BY ATTRIBUTE

Find documents by a specific attribute

- Lets find beers by brewery_id!



The index definition

VIEW CODE

Map

```
1  function (doc, meta) {
2    if (doc.type == "beer" && doc.brewery_id) {
3      emit(doc.brewery_id, doc.abv);
4    }
5  }
```

The result set: beers keyed by brewery_id

Development Time Subset | Full Cluster Data Set

Key
"110f0013c9" 110fdd305e
"110f0013c9" 110fdd3d0b
"110f0013c9" 110fdd4296

QUERY PATTERN:
BASIC AGGREGATIONS

Use a built-in reduce function with a group query

- Lets find **average abv** for each brewery!

VIEW CODE

Save As... Save

Map

```
1  function (doc, meta) {
2    if (doc.type == "beer" && doc.brewery_id) {
3      emit(doc.brewery_id, doc.abv);
4    }
5  }
```

Reduce (built in: _count, _sum, _stats)

```
1  _stats
```

Filter Results ?group=true&reduce=true&stale=false&connection_timeout=60000&limit=10&skip=0

Show Results

Development Time Subset | Full Cluster Data Set

Key	Value
"110f0013c9" undefined	{ "sum": 54.59999999999999, "count": 8, "min": 5.2, "max": 8.2, "sumsqr": 380.92 }
"110f001bbe" undefined	{ "sum": 63.70000000000001, "count": 11, "min": 3.6, "max": 9.8, "sumsqr": 393.8700000000001 }
"110f002955" undefined	{ "sum": 11, "count": 2, "min": 5, "max": 6, "sumsqr": 61 }
"110f0032cc" undefined	{ "sum": 22.8, "count": 5, "min": 0, "max": 5.9, "sumsqr": 130.12 }

We are reducing **doc.abv** with **_stats**

VIEW CODE

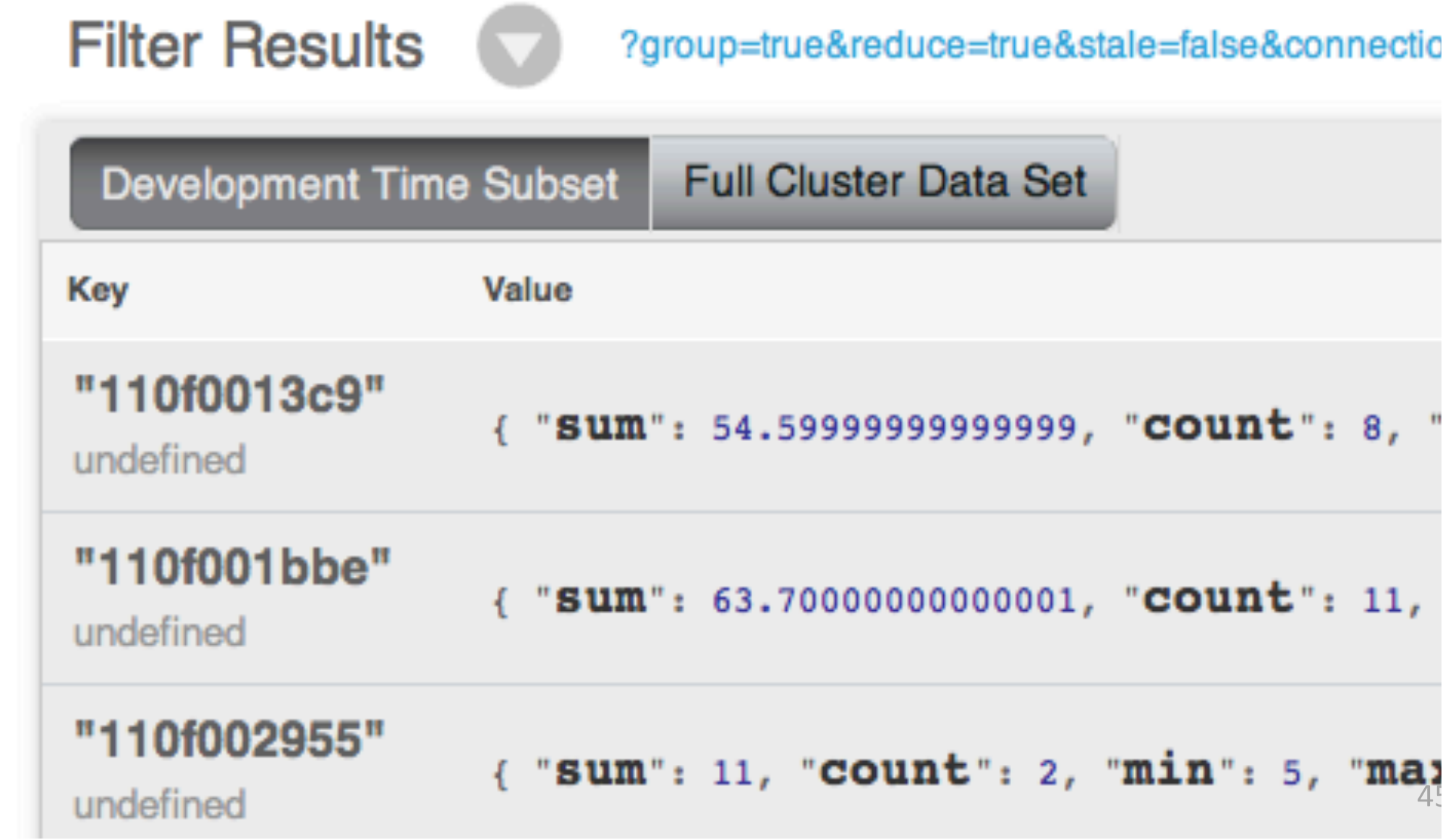
Map

```
1  function (doc, meta) {
2    if (doc.type == "beer" && doc.brewery_id) {
3      emit(doc.brewery_id, doc.abv);
4    }
5  }
```

Reduce (built in: _count, _sum, _stats)

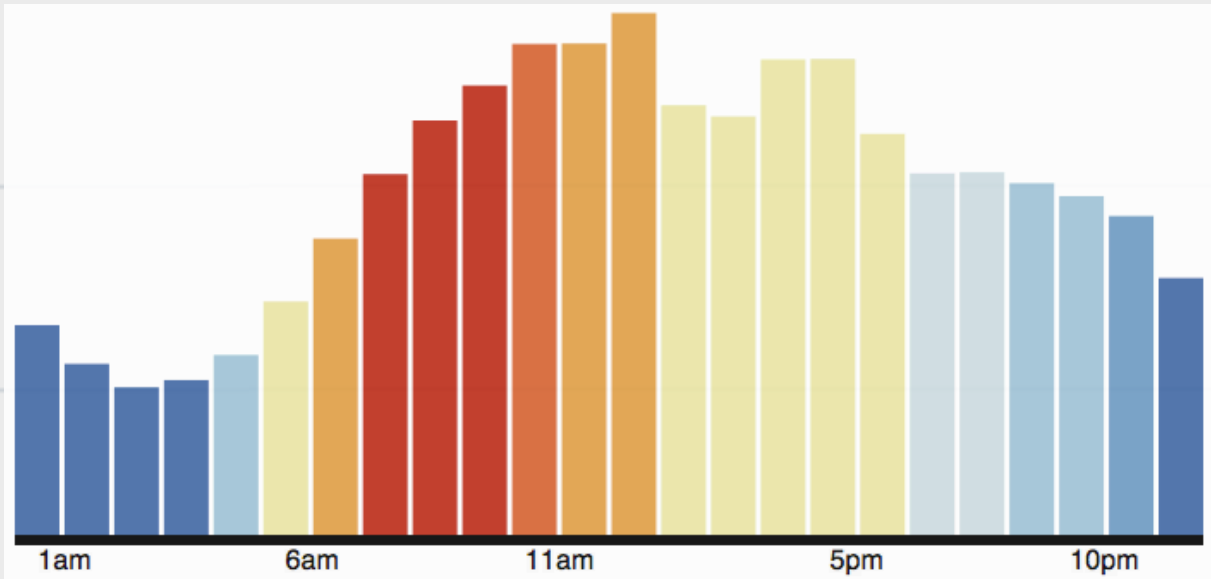
```
1  _stats
```

Group reduce (reduce by unique key)



QUERY PATTERN: TIME-BASED ROLLUPS WITH KEY GROUPING

group_level=3 - daily results - great for graphing

[2010,7,23] undefined		2
[2010,7,29] undefined		2
[2010,10,24] undefined		1
[2010,11,8] undefined		2

- Daily, hourly, minute or second rollup all possible with the same index.
- http://crate.im/posts/couchbase-views-reddit-data/

GEO INDEX &
FULL TEXT INTEGRATION

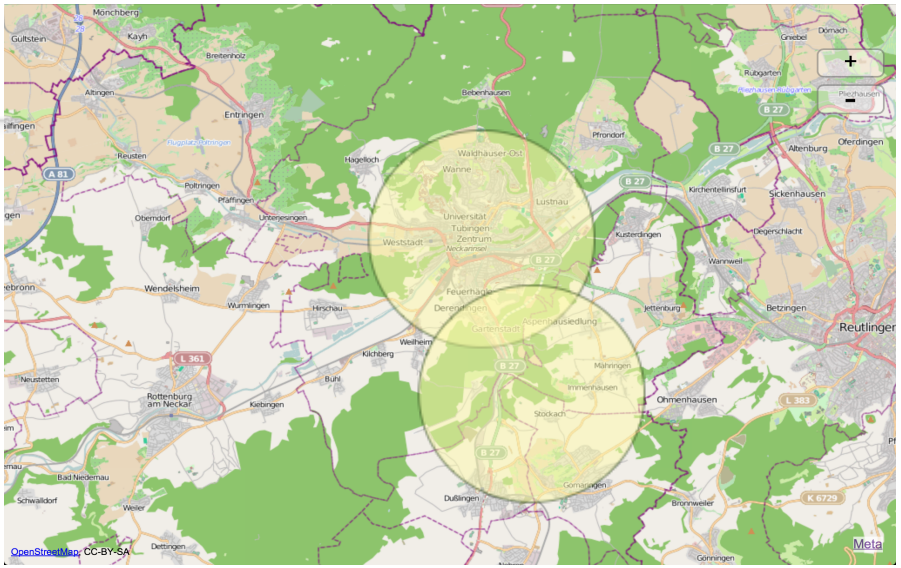
GeoCouch R-Tree Index

- Optimized for bulk loading of large data sets
- Simplified query model (bounding box, nearest

SPATIAL CODE

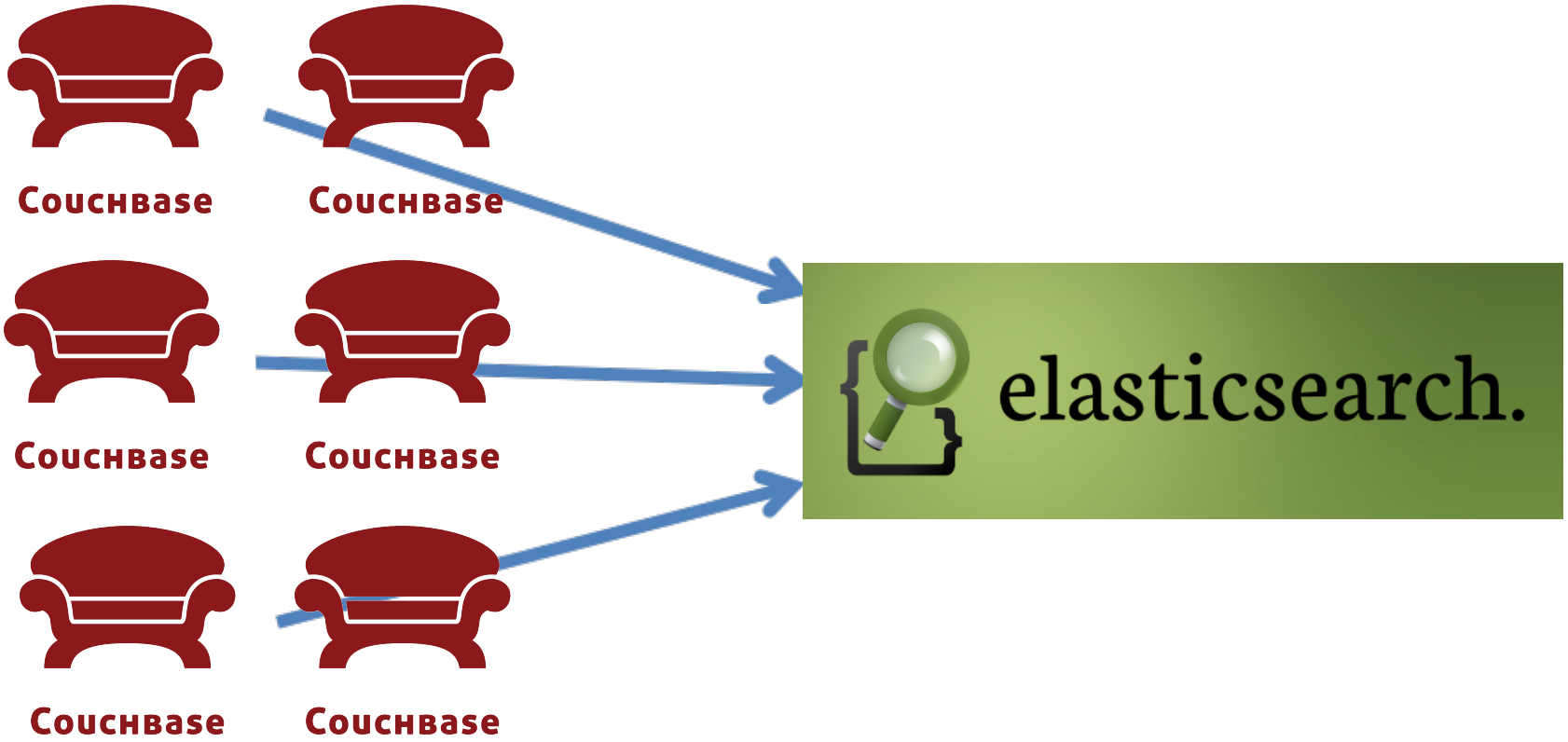
Spatial

```
1  function (doc) {
2    if (doc.geo) {
3      emit({type: "Point", coordinates: [doc.geo.lng, doc.geo.lat]}, doc.name);
4    }
5  }
```



Elastic Search Adapter

- Elastic Search is good for ad-hoc queries and faceted browsing
- Our adapter is aware of changing Couchbase topology
- Indexed by Elastic Search after stored to disk in Couchbase



Questions?

Chris Anderson
@jchris
http://www.couchbase.com/



Couchbase