Document oriented

15 November 2017

Ignacio Coterillo
Computer Engineer, CERN

Document oriented databases

Documents are data structures:

- Self describing
- Hierarchical
- Can contain scalar values, maps, collections, ...
- Can be in different formats XML, JSON, BSON, ...
- Freely structures. Typically similar between them, but it's not required

Document oriented databases

Documents are stored in the Value part of a KV store. The biggest difference is that the value is examinable and its internal attributes can be examined and acted upon

The documents structure can vary inside the same collection or set of documents -> RBMS Table with *Null* columns

The documents can be nested for easy access and optimized performance

Examples: **MongoDB**, CouchDB, OrientDB

MongoDB Concepts

```
RDBMS                           MongoDB

Instance ------------------------ Instance
Schema   ------------------------ Database
Table    ------------------------ Collection
Row      ------------------------ document
rowid    ------------------------ _id
join     ------------------------ DBRef
```

MongoDB Consistency

Transactions on single documents are atomic and consistent

Warning: A write operation is acknowledged by the server when it's received by the server, not when it its commited to disk. Can be controlled with the *writeconcern* setting:

```
{ w: <value>, j: <boolean>, wtimeout: <number> }
```

- **w**: Number of nodes to wait for confirmation to
- **j**: Require confirmation of the operation being commited to the Journal
- **wtimeout**: Maximum time to wait for the **w** confirmations

MongoDB Availabilty

Based in the concept of *Replica Sets*, formed by **N** servers.

The system is configured for **Writes** to be considered confirmed after **W** <= **N** nodes have acknowledged the operation

One node, the **master** receives writes and forward the change log (journal) to the slaves. The slaves can be used for scaling read operations.

W can be specified per operation

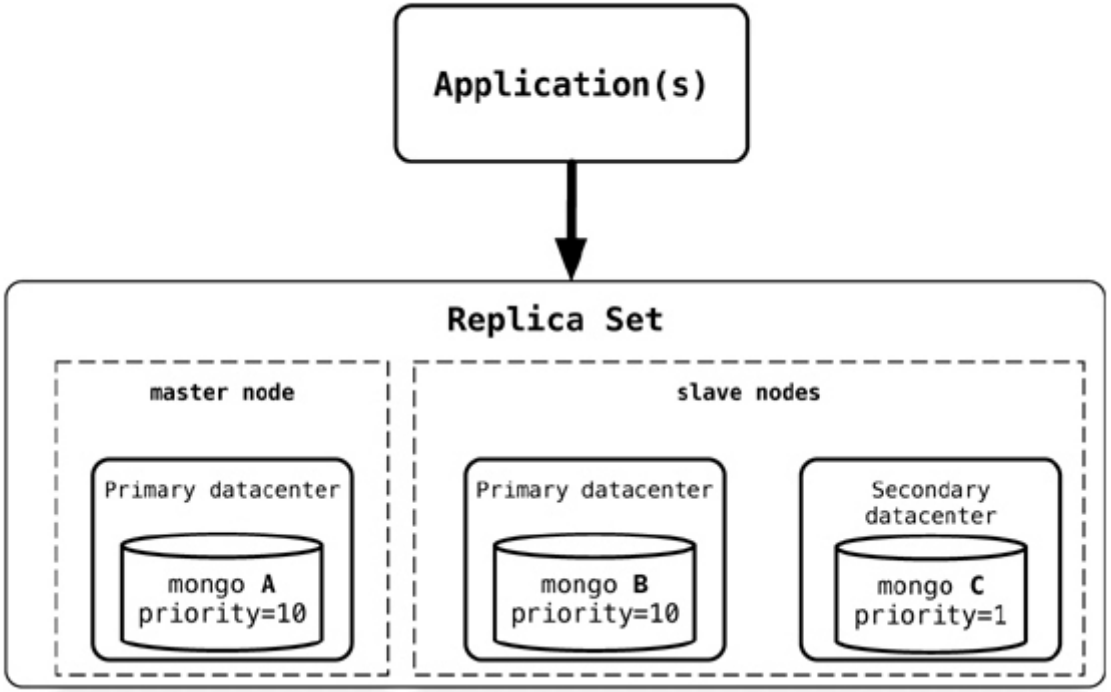
MongoDB Availabilty

Replica sets require **N** >= 2 servers

Asynchronous replication between the Master and Slaves

The nodes elect the master via a voting procedure but the election can be rigged by using weights to prefer certain nodes over other the rest

MongoDB Availabilty



MongoDB Query capabilites

```
SELECT * FROM order;

db.orders.find()

SELECT * FROM order WHERE id = "1bad84f5677";

db.orders.find( { "id" : "1bad84f5677" } )
```

MongoDB Query capabilites

```
SELECT orderId, ordeDate FROM order WHERE id = "1bad84f5677";

db.orders.find( { "id" : "1bad84f5677" }, { orderId : 1, orderDate : 1} )

SELECT * from order where requester like 'John%';

db.orders.find({"requester":/^John/)
```

MongoDB Query capabilites

- Indexing

Typical use cases

- Schema-Free friendly: Web, CMS, blogging, e-commerce
- Event Logging

ElasticSearch Example

Thank you

Ignacio Coterillo
Computer Engineer, CERN
ignacio.coterillo.coz@cern.ch (mailto:ignacio.coterillo.coz@cern.ch)