Mongo DB

Table Of Contents

- Overview
- BSON
- Indices
- Write Operations and concern
- Journaling
- Read Operations
- Replication
- Sharding

Overview

- Open Source
- CP (mostly consistent)
- No Sql Database
- Scalable, High-performance and highly available
- Native replication
- Document Oriented storage
- Store Data in BSON format

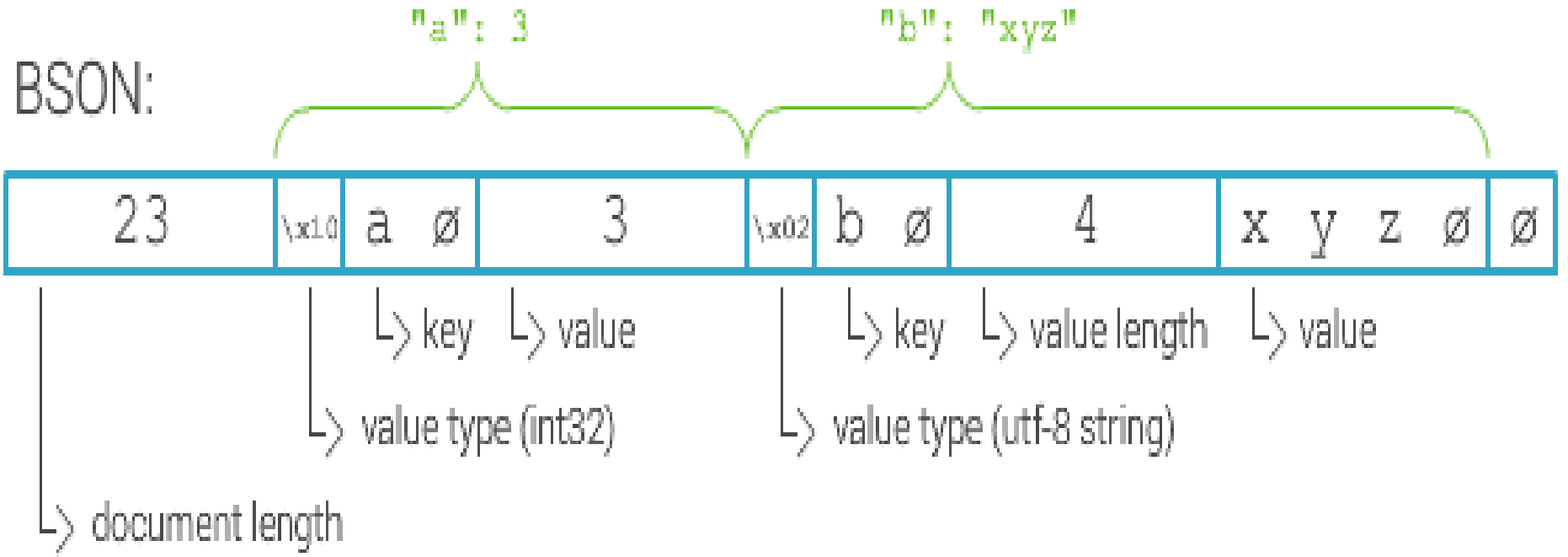
BSON

- Binary JSON: binary-encoded serialization of JSON-like documents which adds two things to the mix, lengths and types
- Lightweight
- Traversable : BSON is designed to be traversed e ily.
- Efficient: Conversion fron JSON to BSON and vice versa is easy and fast

JSON:

```
{
   "a": 3,
   "b": "xyz"
}
```

BSON:



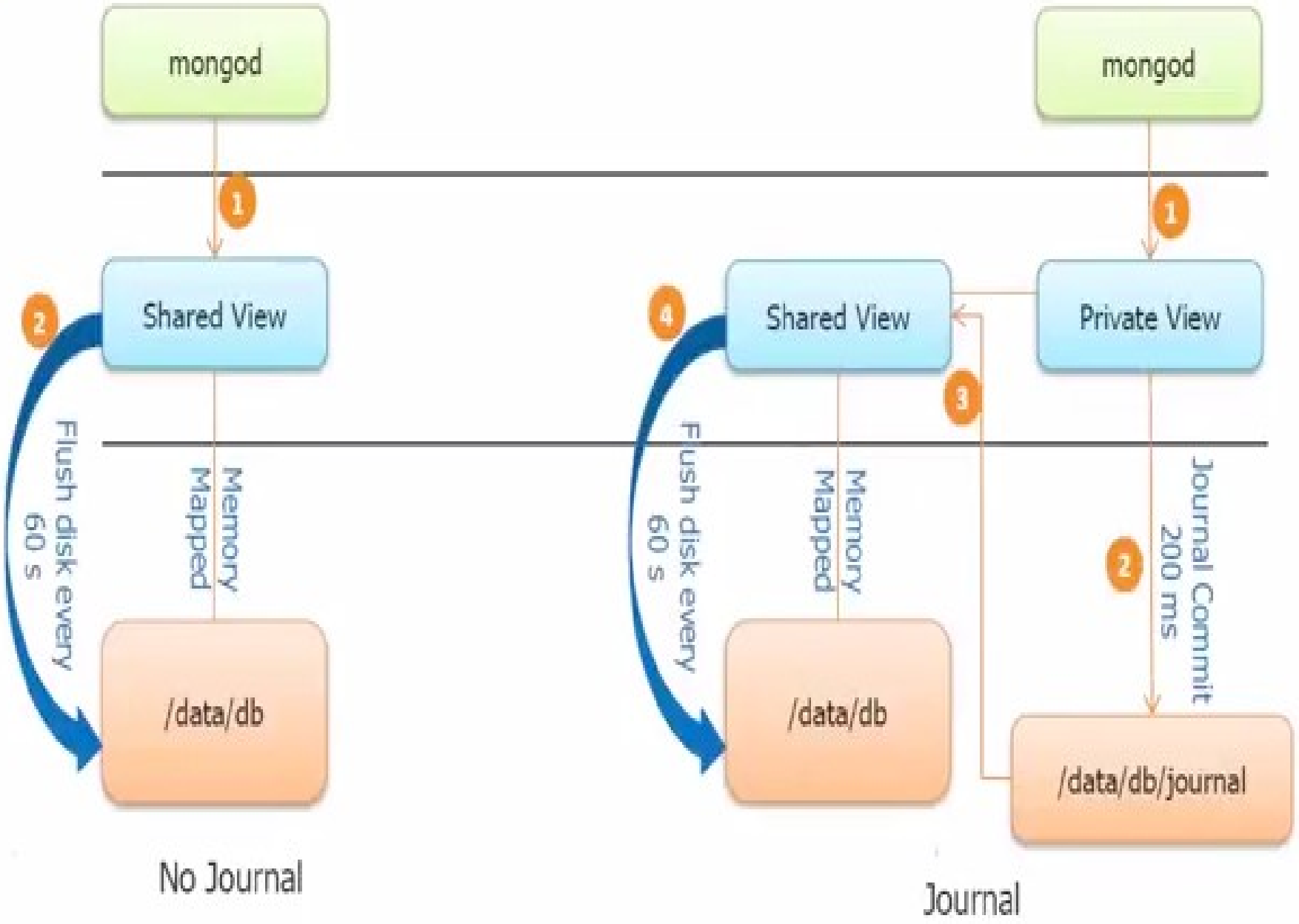
Indices

- Indexs: default _id (Object Id hashcode, can be overwritten but must be unique)
- Some of the Types:
- Single: single json key index (can be inc or dec) (db.collection.createIndex({ name: -1 }))
- Compound Index: on multiple keys ({ userid: 1, score: -1 })(userId as inc then those doc has score dec)
- Multikey Index: if index has value as array multiple entry in index for each value
- **Some limitations**
- Size limitation of 1 kB (with BSON conversion overhead)
- Max 64 Indexes
- no more than 31 fields in a compound index.
- **Some Properties**
- Can add properties to index data to filter documents (unique, Sparse,TTL etc)
- db.collection.createIndex(<key and index type specification>, { unique: true })

Write Operation

- Each index on a collection adds some amount of overhead to the performance of write operations.
- Document gets relocated if outgrows the space limitation and all indexs has to be updated
- the basic unit of atomicity is the single document: you can make transactional writes to a document, but not across documents.
- Max document size 16MB
- Journaling for consistency (journalcommit interval (3-300ms)(only operations))

Journaling Mechanics

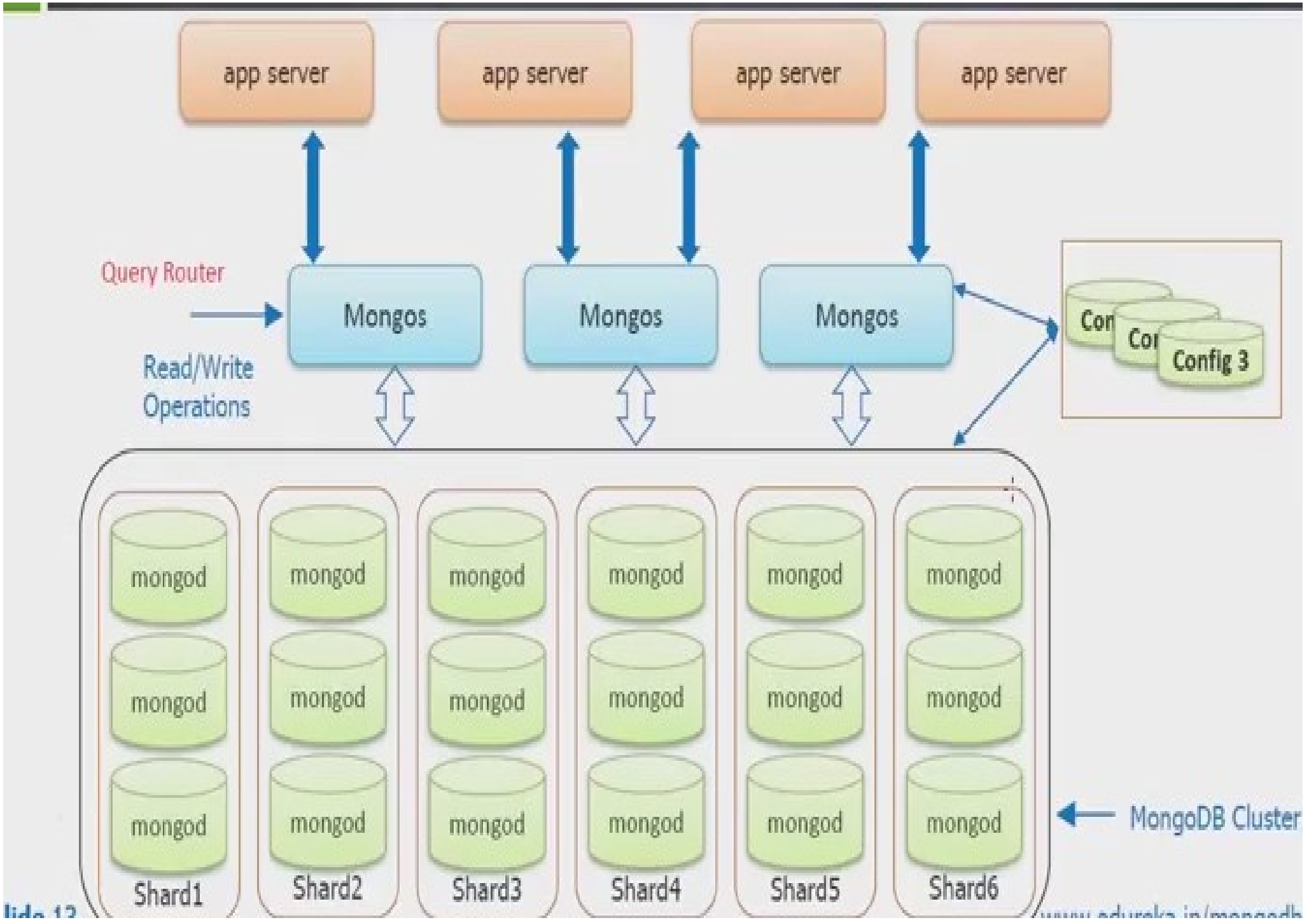


Read operations

- Based on query mongo will parse if fields are index or can have intersection of indexs or compound index.
- If keys are not indexed the all documents will be parsed in the array and match the conditions
- Reads may miss matching documents that are updated during the course of the read operation.
- Mongo also provide wide functionality for crud operations as well as multiple operational procedure to aggreate (group) the result and perform operations on the data

Read Operations

- **Aggregation Pipeline**
- Example : db.collection.aggregate(pipeline, options)
- db.orders.aggregate([{ $match: { status: "A" } }, { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }])
- **Single Purpose Aggregation Operations**
- Example: db.collections.distinct(“user_id”);
- **MapReduce**
- db.collection.mapReduce(map, reduce, {<out>, <query>, <sort>, <limit>, <finalize>, <scope>, <jsMode>, <verbose>})
  db.collection.mapReduce(
  function(){emit(this.key,this.value);},
  function(key,value){return Array.sum(value);},
  {query:{status:1},out:”total”})

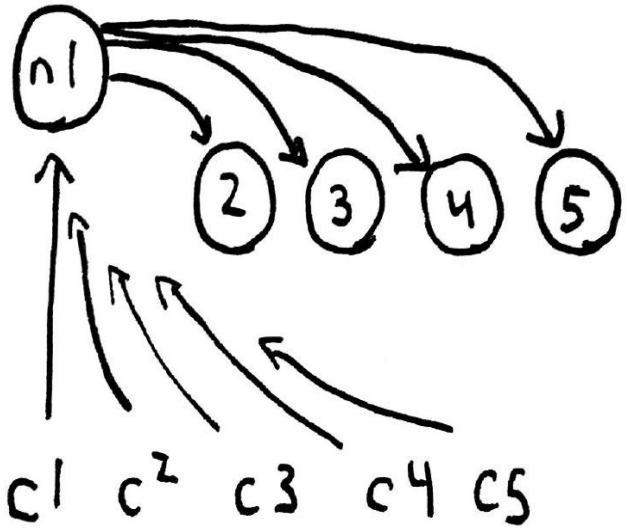

app server
app server
app server
app server
Query Router
Mongos
Mongos
Mongos
Read/Write
Operations
Config 3
mongod
Shard1
Shard2
Shard3
Shard4
Shard5
Shard6
MongoDB Cluster

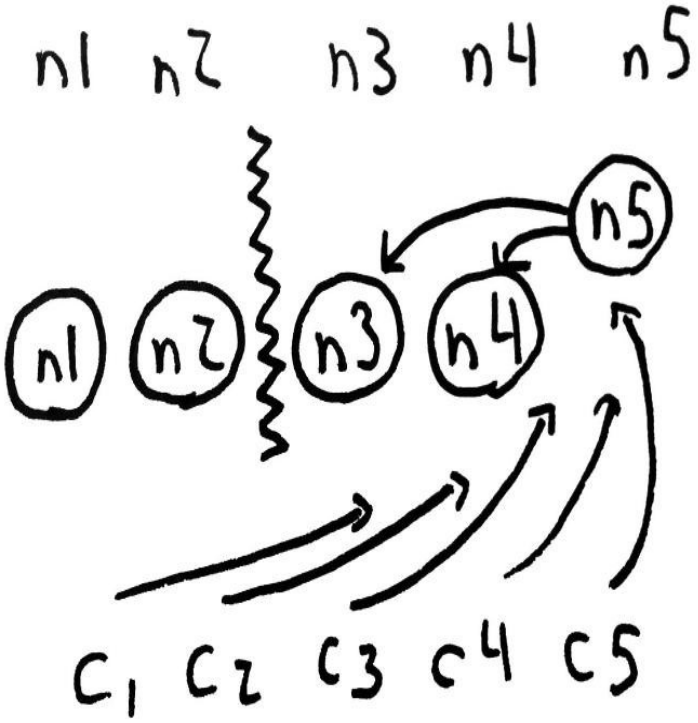
Replication

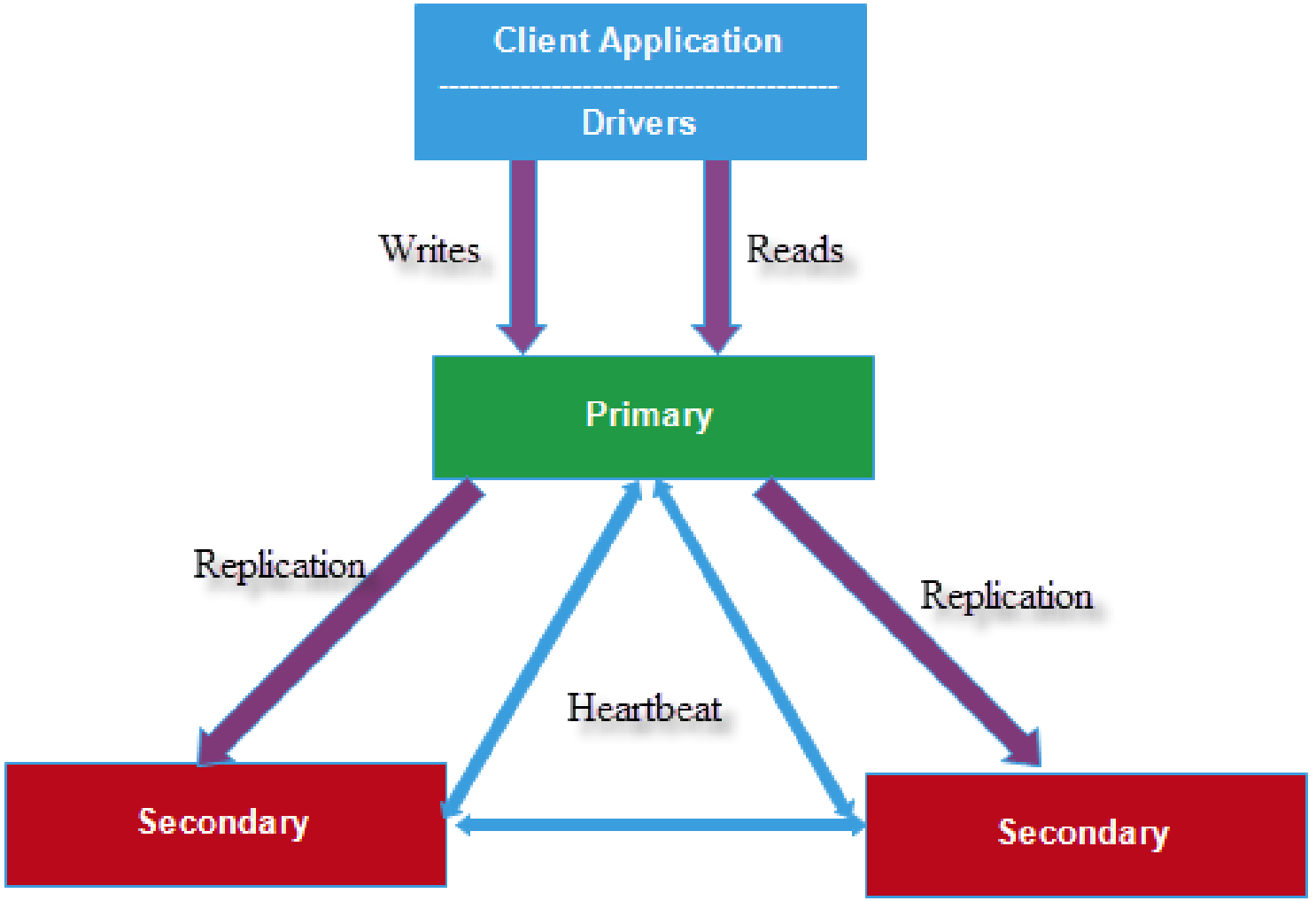
- A replica set is a group of mongod instances that maintain the same data set. A replica set contains several data bearing nodes and optionally one arbiter node.
- The primary node receives all write operations. A replica set can have only one primary capable of confirming writes
- The primary records all changes to its data sets in its operation log
- The secondaries replicate the primary's oplog and apply the operations to their data sets such that the secondaries' data sets reflect the primary's data set. If the primary is unavailable, an eligible secondary will hold an election to elect itself the new primary.
- Arbiters do not maintain a data set. The purpose of an arbiter is to maintain a quorum in a replica set by responding to heartbeat and election requests by other replica set members. (can be usedful in tie scenario)
- When a primary does not communicate with the other members of the set for more than 10 seconds, an eligible secondary will hold an election to elect itself the new primary. The first secondary to hold an election and receive a majority of the members' votes becomes primary.

- { w: <value>, j: <boolean>, wtimeout: <number> }
- W: “majority”,j:true, wtimeout timelimit for auto fail. If not specified can block indefinelity if write concern is unachieveable
- Useful in replication enviornment
- example

db.products.insert(
{ item: "envelopes", qty : 100, type: "Clasp" },
- { writeConcern: { w: 2, wtimeout: 5000 } })



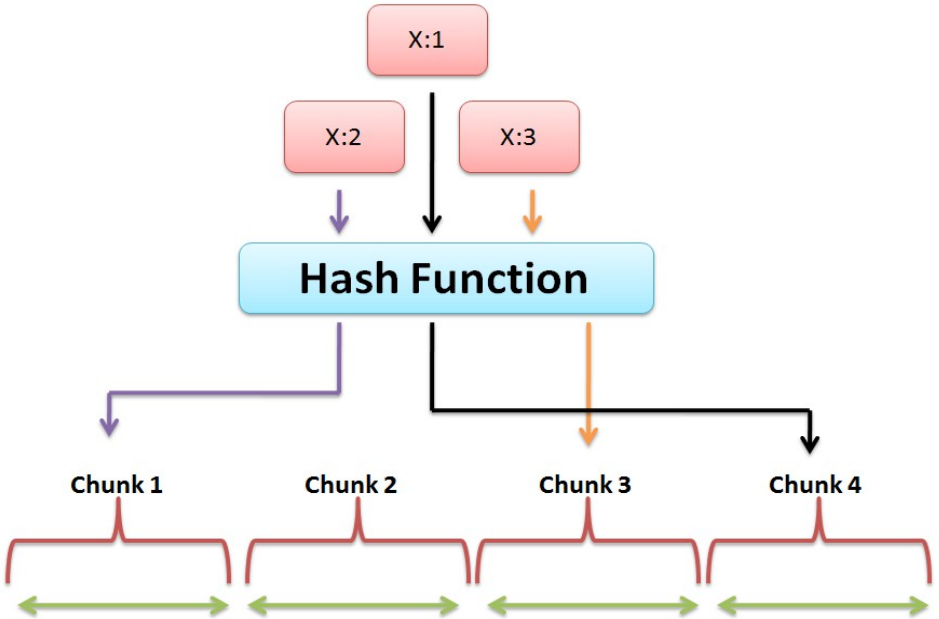


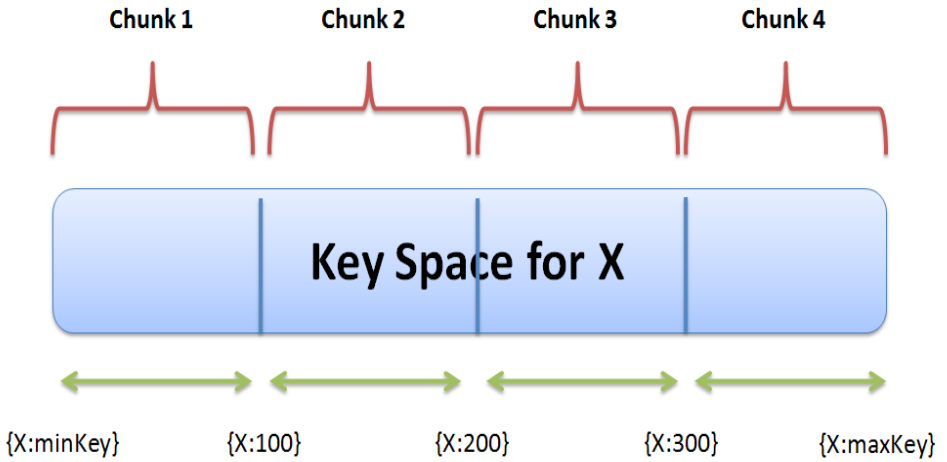

Client Application
Drivers
Writes
Reads
Primary
Replication
Replication
Heartbeat
Secondary
Secondary

Sharding

- Components: shards(replica set), mongos(router) and config servers (as replica set since 3.4)
- The shard key consists of an immutable field or fields that exist in every document in the target collection.
- You choose the shard key when sharding a collection. The choice of shard key cannot be changed after sharding. A sharded collection can have only one shard key
- By default if a collection have data then max size data collection which can be sharded is based on average size of all shard key values, and the configured chunk size.
- maxSplits = 16777216 (bytes) / <average size of shard key values in bytes>, chunkSize =64 default can inc to 1024MB
- maxCollectionSize (MB) = maxSplits * (chunkSize / 2)
- MongoDB does not support unique indexes across shards, except when the unique index contains the full shard key as a prefix of the index. In these situations MongoDB will enforce uniqueness across the full key, not a single field.

- A shard key cannot exceed 512 bytes
- use a hashed shard key or select a field that does not increase or decrease monotonically as shard key otherwise can cause insert throughput bottleneck while inserting a chunk. Works fine for read and update.
- sh.shardCollection("<your-db>", {{ "foo" : 1, "bar" : 1, "_id" : 1 }:"hashed"})
- Types of sharding: hashed or ranged





DATA DOTZ

Refrences

https://docs.mongodb.com/manual/

https://aphyr.com/posts/284-jepsen-mongodb

https://blog.meteor.com/mongodb-queries-dont-always-return-all-matching-documents-654b6594a827