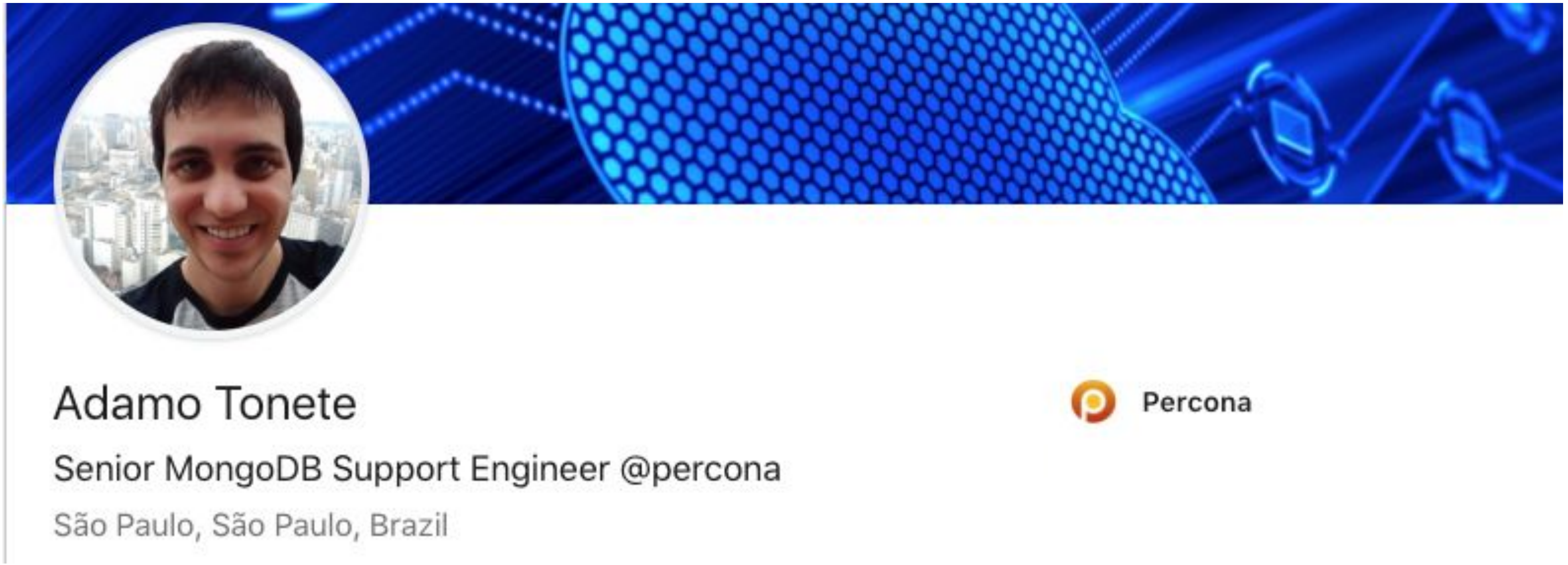
MongoDB Sharding 101

Percona Webinar

July 19th, 2018, at 12:30 PM PDT (UTC-7) / 3:30 PM EDT (UTC-4).

PERCONA

Me - @adamotonete



Adamo Tonete

Senior MongoDB Support Engineer @percona

São Paulo, São Paulo, Brazil

Agenda

- What is MongoDB?
- Single Instances
- Replica-set architecture
- Shard architecture
- Q&A

MongoDB

- Open source document-oriented database
- Made to run in the cloud, easily scalable
- Quick installation and configuration

PERCONA

Single Instance

Single Instance

- Commonly used for development/tests
- Embedded systems

PERCONA

Replica-set

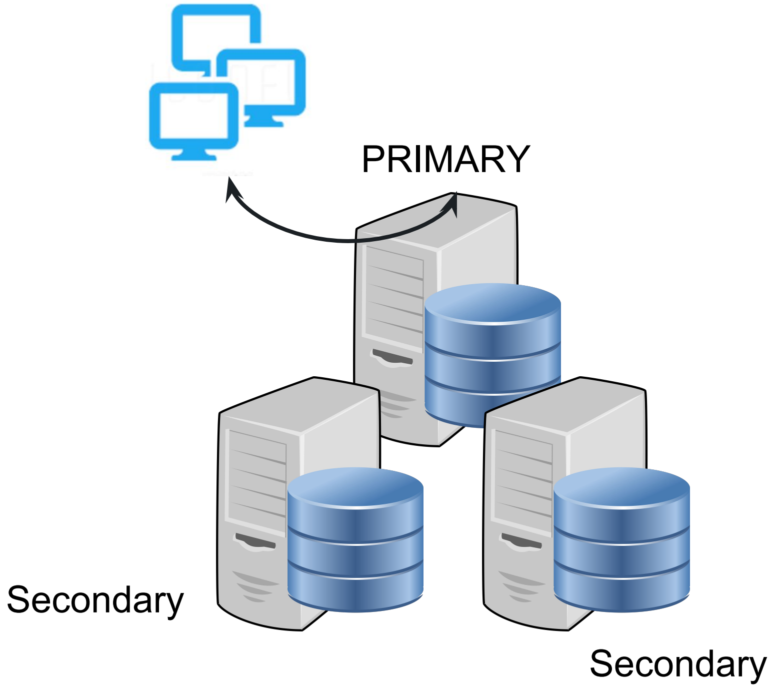
- Scale out
- Ability to elect a new primary in case of failure (auto HA)
- Data are the same in the replicas, asynchronous replication.
- Single master - PRIMARY

PERCONA

Replica-set

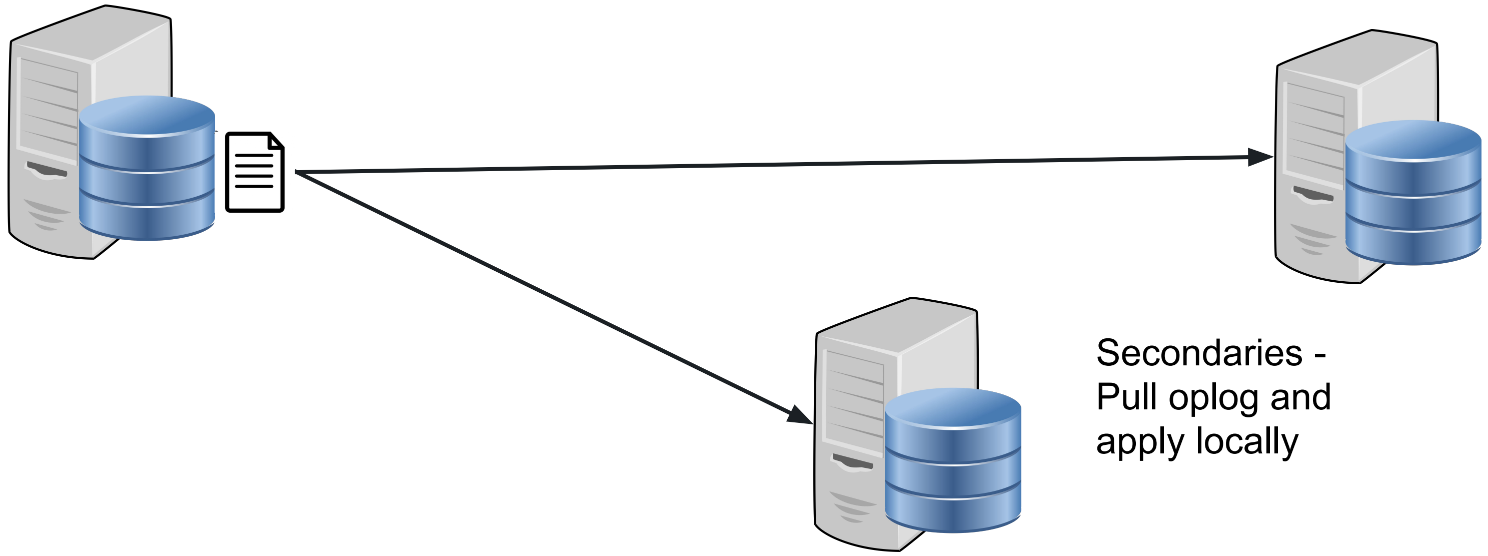
How does a replica-set work?

Main collection when running replica-set is oplog.rs;



Replica-set - oplogs

How does a replica-set work?

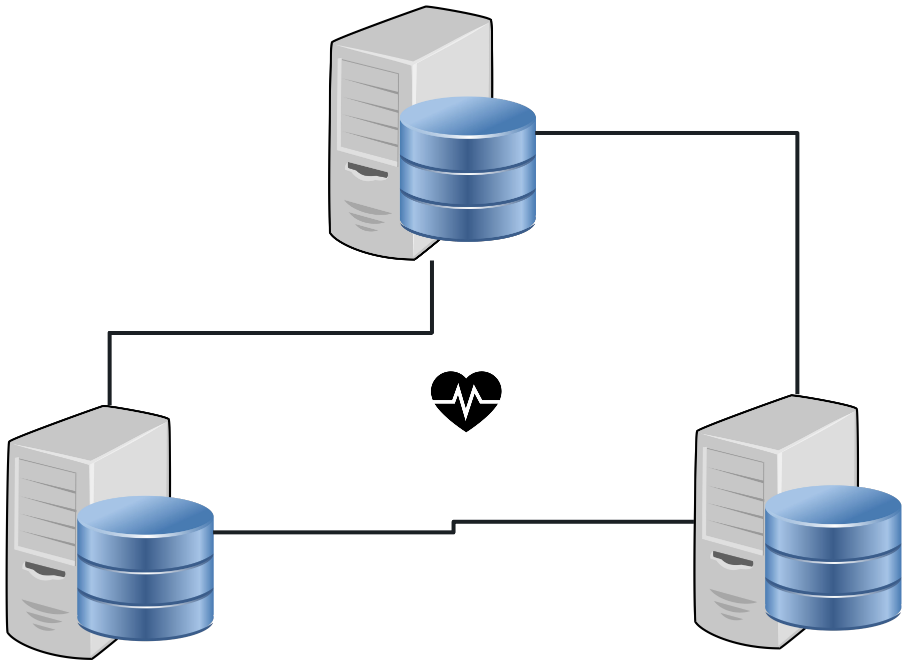


Secondaries -
Pull oplog and
apply locally

Replica-set

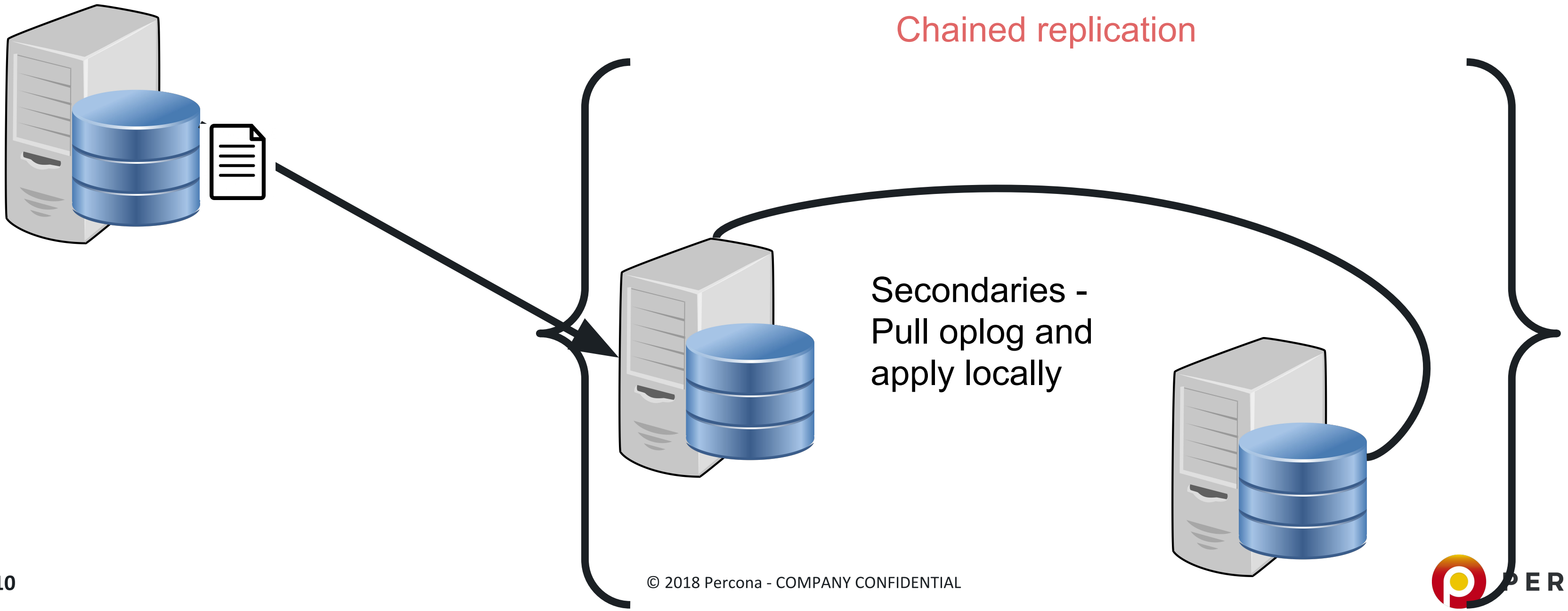
Replica-set

- Heartbeat
- Votes
- Priority
- Arbiter
- Hidden



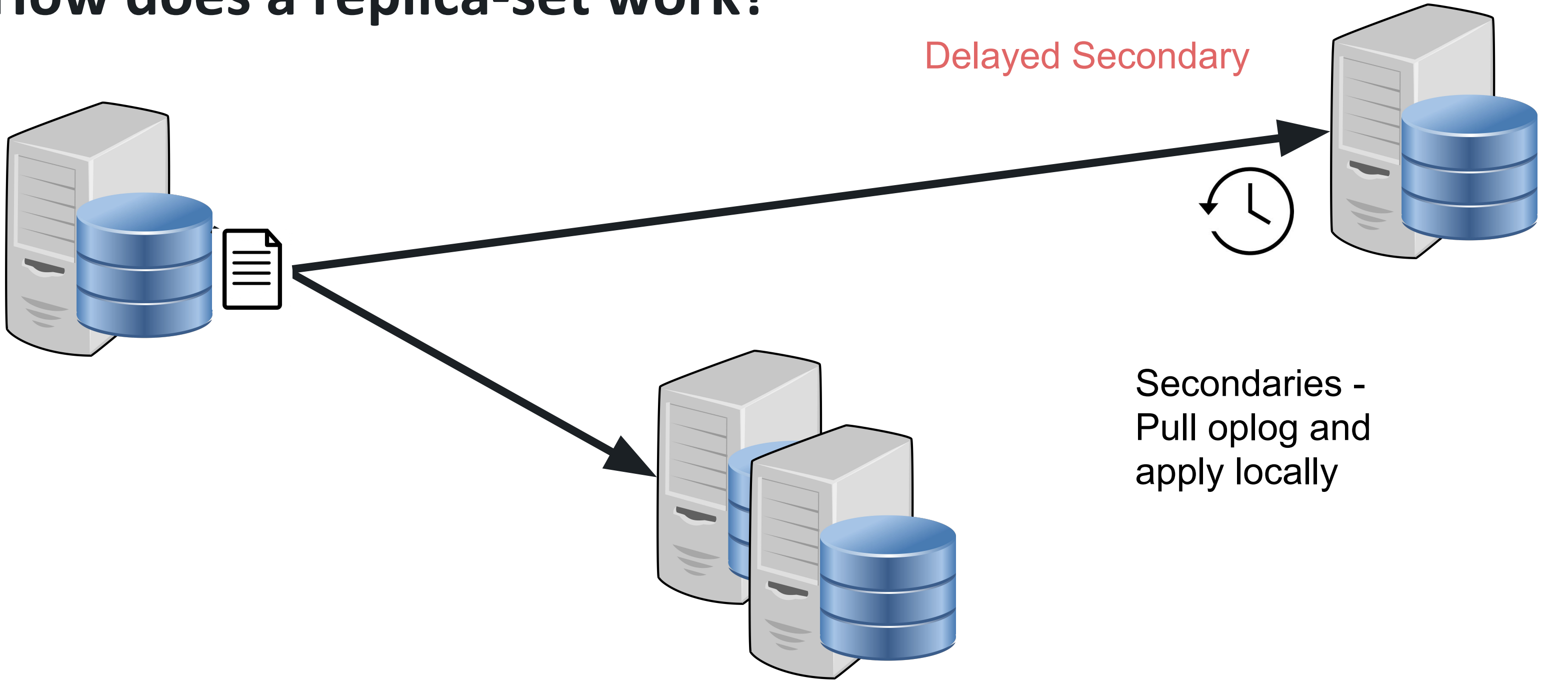
Replica-set

How does a replica-set work?



Replica-set

How does a replica-set work?



Secondaries -
Pull oplog and
apply locally

PERCONA

Replica-set

What if a heartbeat fails?

An election process will promote a secondary to primary when the primary is no longer available.

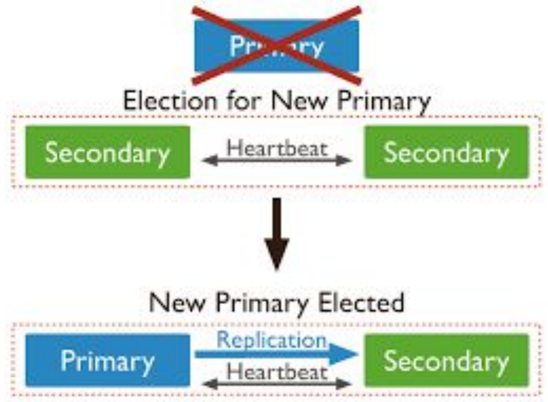
The most up to date instance has preference to become a primary, although it is not always true.

Replica-set

What if a heartbeat fails?

Each instance can have different priorities, in sum the higher the priority, the greater the probability of the instance becoming a primary.

Arbiters do not hold data, they're only used to break ties in elections.
Use an arbiter if a replica set has an even number of members.



Replica-set

- **Tweaking the consistency**

- readPreference
- writeConcern

PERCONA

Replica-set - writeConcern

The default write concern value is *"1",* which means that once the primary receives the operation, it is considered complete.

If an election is triggered, we can lose this operation as it might not be replicated.

It is possible to specify a different writeConcern per query or per connection.

The possible values are:
1, 2, 3, N / "majority" / "tag_name"

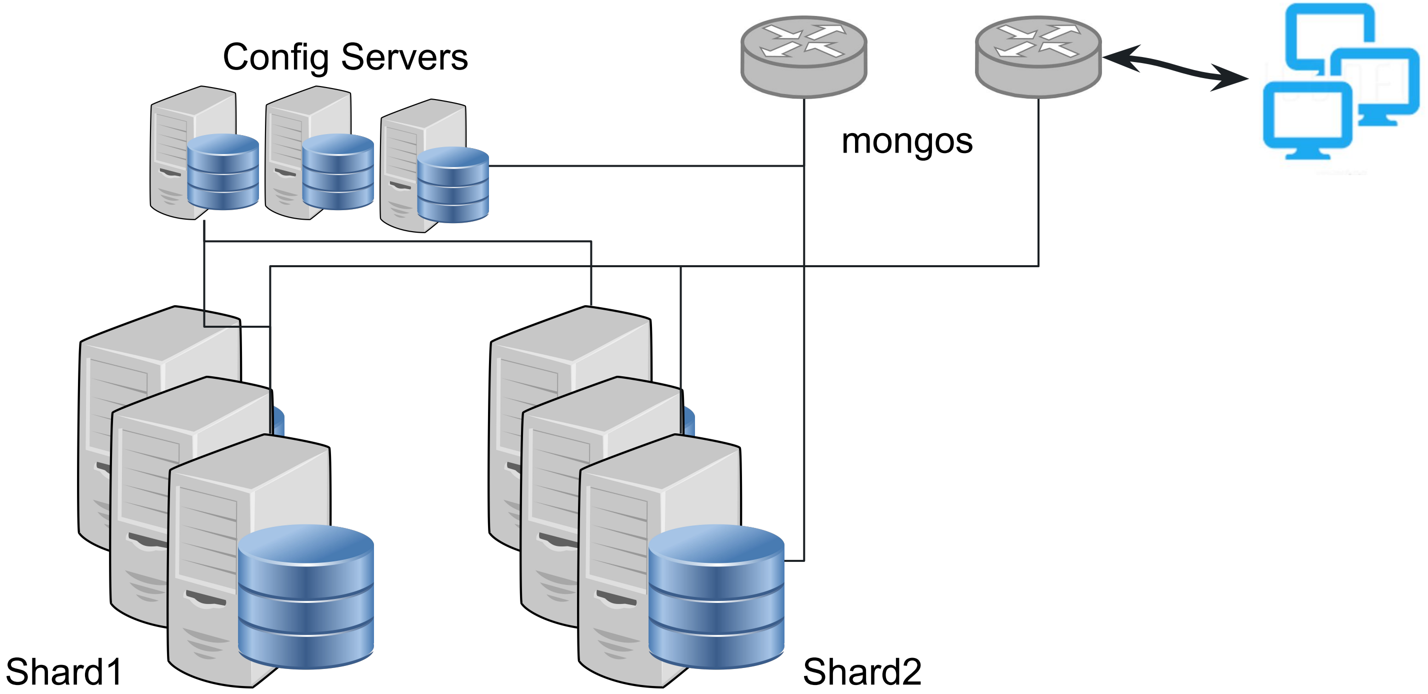
Replica-set - readPreference

Read Preference is primary.

Every single read will come from the primary if we don't specify a parameter for the driver to read from secondaries.

Important! Reading from secondaries may return outdated, stall data.. However, this is a very common way to scale out read intensive applications.

Sharded Cluster



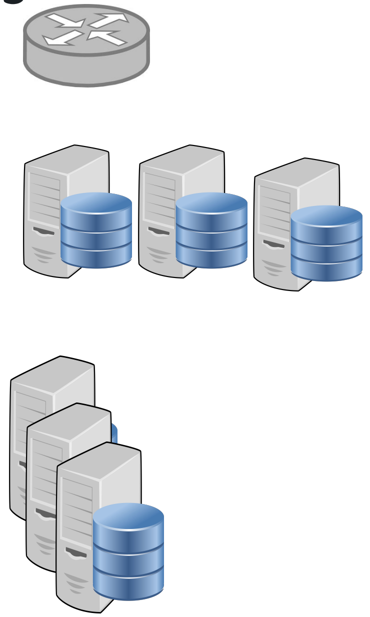
PERCONA

Sharded Cluster architecture

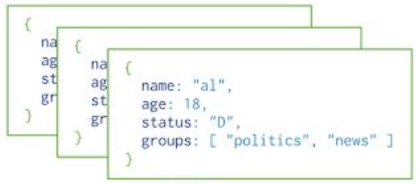
Mongos

Config

Shards

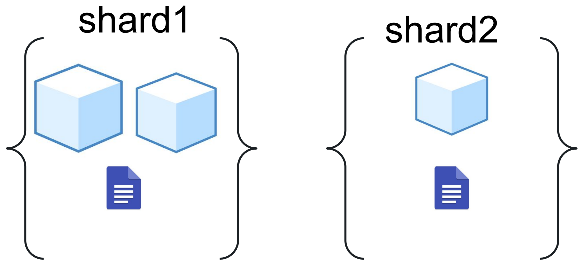


Sharded Cluster - chunks



```
{
  name: "al",
  age: 18,
  status: "D",
  groups: [ "politics", "news" ]
}
```

Collection



A database can have multiple collections, each collection can be sharded differently. Each shard has chunks and their documents.

Sharded Cluster

Primary Shard

Data are split among shards in small chunks by the shard key.

Each chunk has 64MB data (default).

Chunks are distributed among the shards but can also live in a single one.

Sharded Cluster - Shard key

Shard key:

Field(s) that will be used to distribute the data among the shards. Once data is partitioned there is no way to change the shard key.

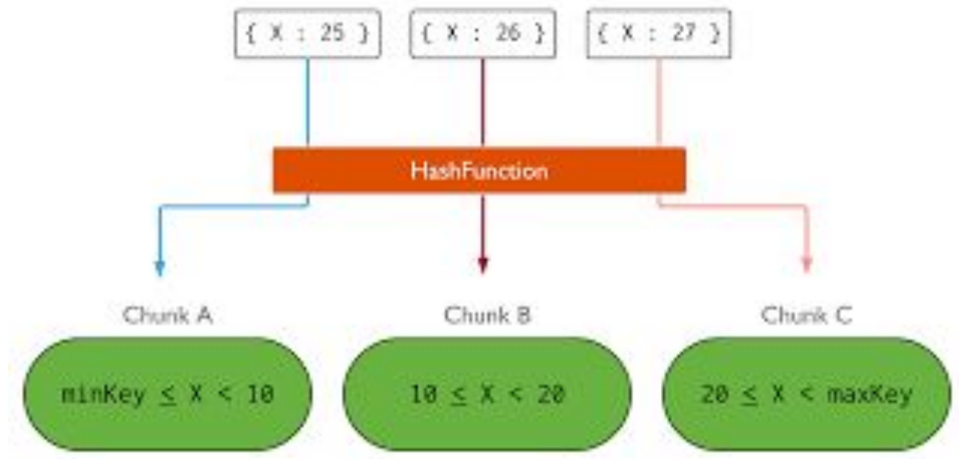
(No other key except _id/not part of shard key columns could be unique - you can explain about this)

A shard key can be used to distribute data in:

- Hashed
- Range
- Zones

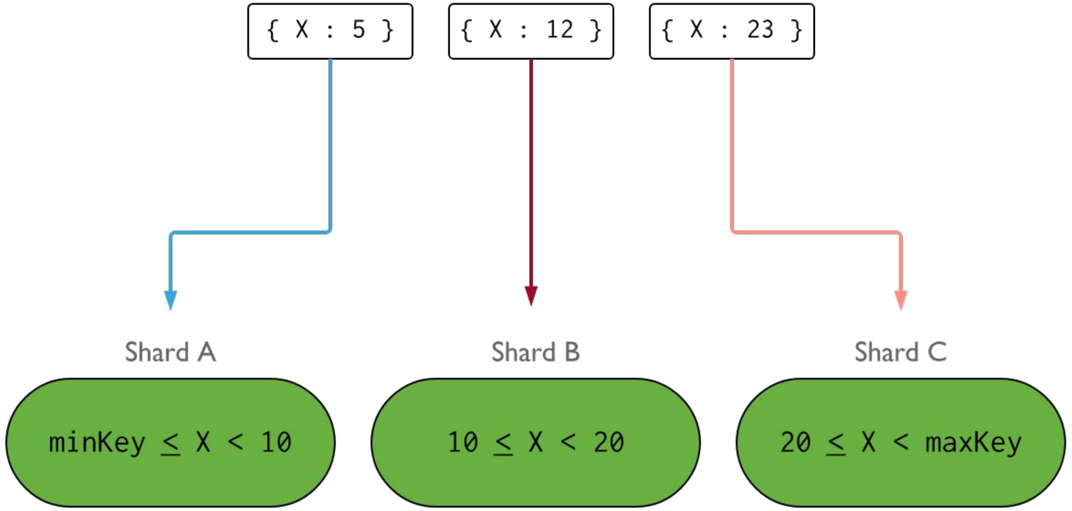
Sharded Cluster

Hashed Shard key:



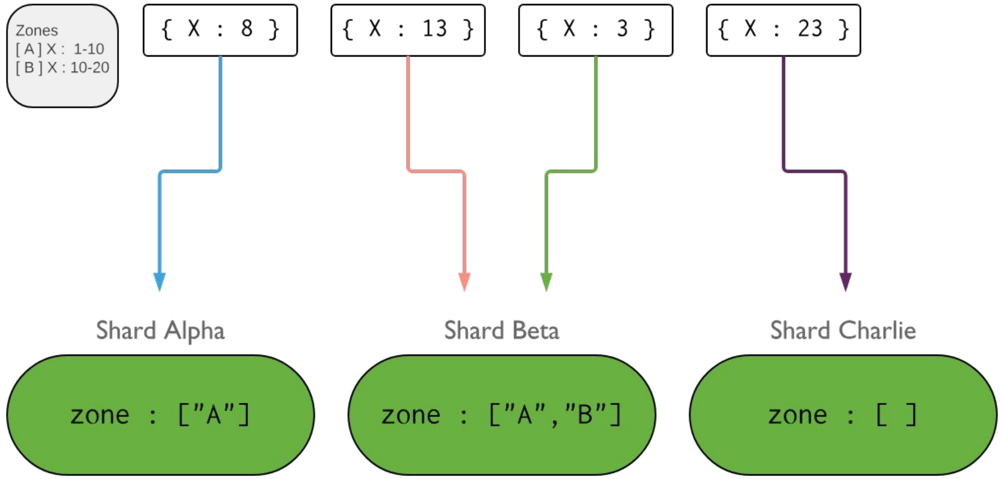
Sharded Cluster

Range Shard key:



Sharded Cluster

TAG Shard key:



Sharded Cluster

Clustered shard parts: The config servers

- All cluster metadata is there
- Partitions, collections, databases configuration
- Migrations

Sharded Cluster

Clustered shard parts: The mongos

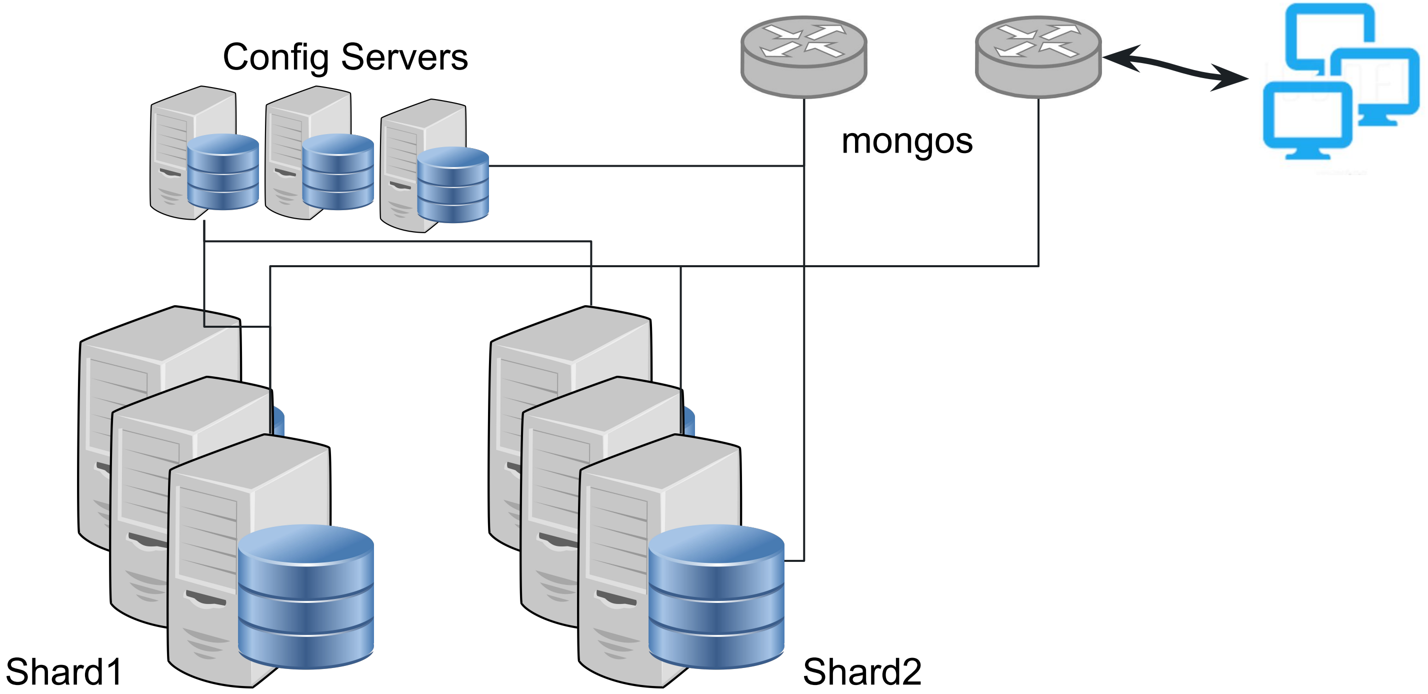
- Responsible for routing all the queries
- Act like a proxy
- Merge results to send to the client
- Clients think it is a single instance

Sharded Cluster

Clustered shard parts: The shard

- It is a replica-set after all.
- Members of a shard do have the same data.
- Only part of the data is saved on a shard.

Sharded Cluster



Sharded Cluster - Internals

Clustered shard parts, configuration and processes

- Balancer
- Chunk Migration
- Orphan documents
- Auto split
- Jumbo chunk

Sharded Cluster - Internals

Clustered shard processes: Balancer

Balancer is the process responsible for moving chunks along instances. We highly suggest keeping balancer on to distribute the data.

This process directly changes data in the config database.

- Schedule “balancer window”
- Chunk size

PERCONA

Sharded Cluster - Internals

Clustered shard processes: Chunk Migration

In case there are too many chunks in a specific shard, the balancer will move those data to different shards.

Each migration will migrate an entire chunk (64mb) to a different shard.

If the migration fails we may end up with data in a shard that do not belong to the shard range. Such documents are called orphan documents.

Sharded Cluster - Internals

Clustered shard processes: Split

It is possible to split a chunk manually or wait until the balancer figures out whether there is a chunk to be split or not.

The auto split process will divide this chunk in two and might migrate part of the old chunk to another shard.

Sharded Cluster - Internals

Clustered shard processes: Jumbo Chunk

Sometimes it is not possible to split a chunk and we see a chunk with the "jumbo:true".

This means this chunk is bigger than 64MB and it is not possible to split it automatically, mainly because the shard key doesn't have enough selectivity.

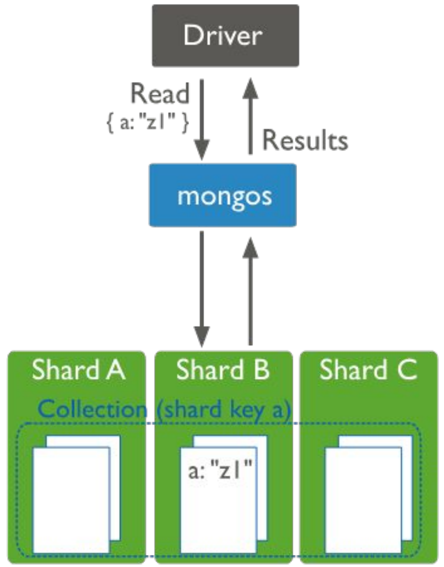
PERCONA

Sharded Cluster - Queries

Querying into a shard

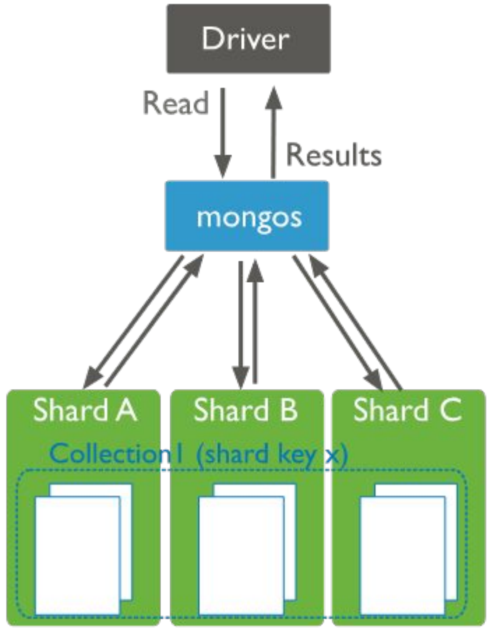
- Mongos is the process that in fact "talks" to the shards. All the queries go through the MongoS process before returning results to the client.
- Target query, scatter gather, collection scans.

Sharded Cluster - Queries



- If querying by the shard key, it is very likely that the *mongos* "knows" where to retrieve data.

Sharded Cluster - Queries



- If querying by something different than the shard key, all the shards will be requested to fetch data and mongos will combine the results to the client, returning one single result.

PERCONA

Wrapping up

- Applications don't know they are talking to a shard. Mongos acts as if it were a proxy.
- Shards are basically a few replica-sets sharing data among them.
- The config database is really, really important.

Do not...

- Write directly into the Shard;
- Make changes in config database, unless you really know what you are doing.
- Run a collection scan in Shard.
- Backup separately without consistency state

Questions?