MongoDB Sharded Cluster - How to design your topology

Vinodh Krishnaswamy
Support Engineer - MongoDB & MySQL
Percona

Who am I

- Started as MySQL DB DBA
- Support for MongoDB
- Trainer
- Scripting, Reading, Driving

- *"Guruji"* - who shares knowledge
- Now here I am - Perconian!!!

Agenda

- Sharded Cluster - components
- Ranged Sharding
- Hashed Sharding
- Zones / Tags

Agenda

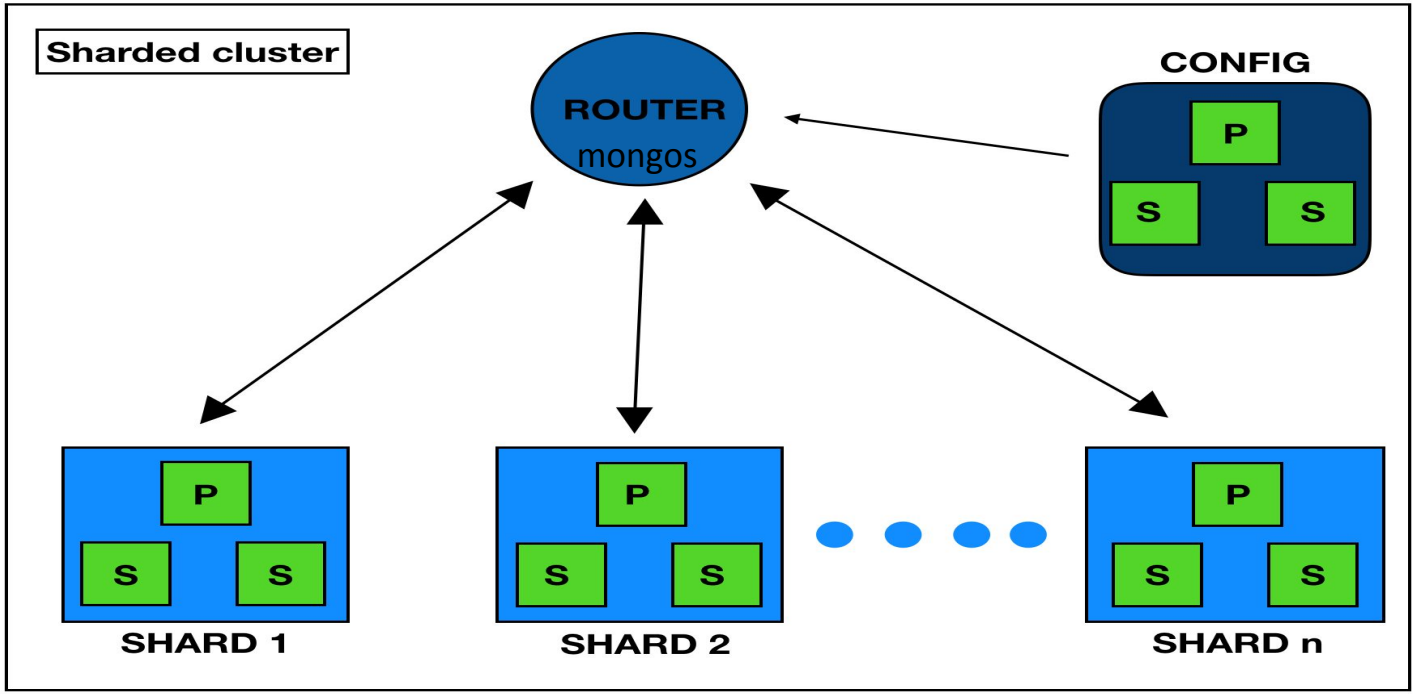
- HA
- ReplicaSet - Key points
- Topology
- Q&A

Sharded Cluster

Sharded Cluster

- Horizontal Scaling
  - *Data across multiple nodes*
  - *Reduce stress for larger working set*
  - *Ability to add more servers*

Architecture



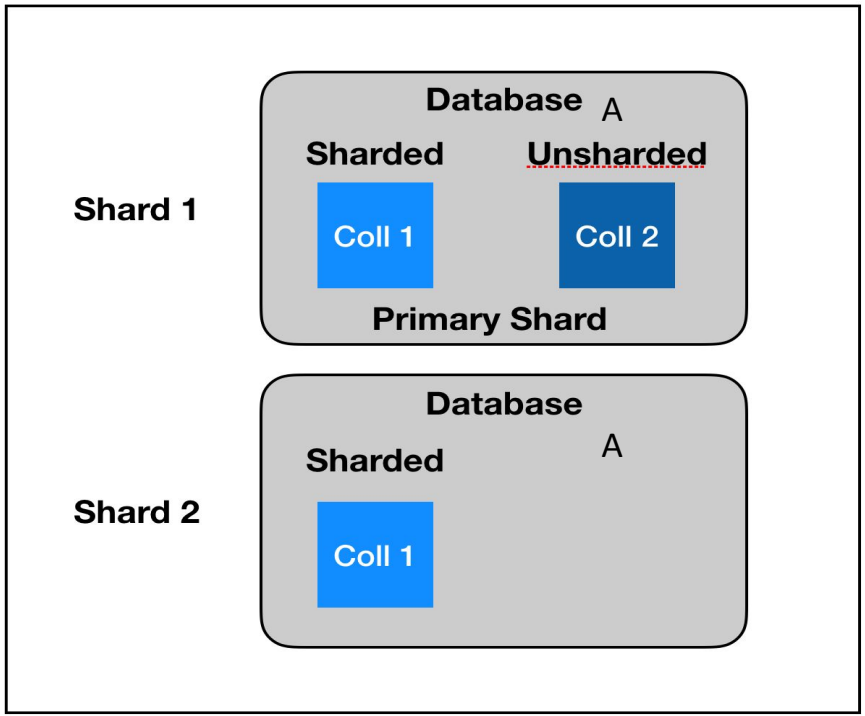
Sharded Cluster

- Shard Key
- Chunks
- Balancer - Chunk distribution

POINTS TO NOTE

TARGET vs BROADCAST

Primary Shard



Primary Shard

sh.status()

```
        config.system.sessions
                shard key: { "_id" : 1 }
                unique: false
                balancing: true
                chunks:
                        shard01 1
                { "_id" : { "$minKey" : 1 } } -->> { "_id" : { "$maxKey" : 1 } } on : shard01 Timestamp(1, 0)
  {  "_id" : "percona",  "primary" : "shard03",  "partitioned" : false }
  {  "_id" : "vinodh",  "primary" : "shard02",  "partitioned" : true }
  _
```

Primary Shard

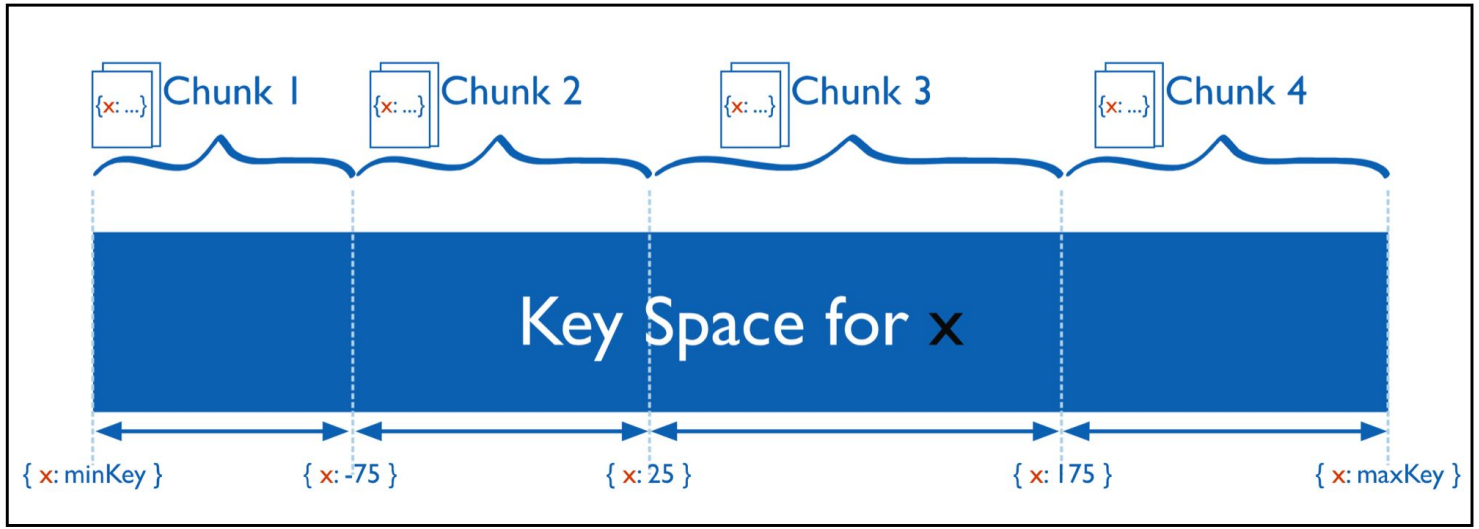
- Routing - sharded vs unsharded
- Keep at right shard

db.adminCommand({ movePrimary: <databaseName>, to: <newPrimaryShard> })

Ranged Sharding

Strategy

- Ranged Sharding



Strategy

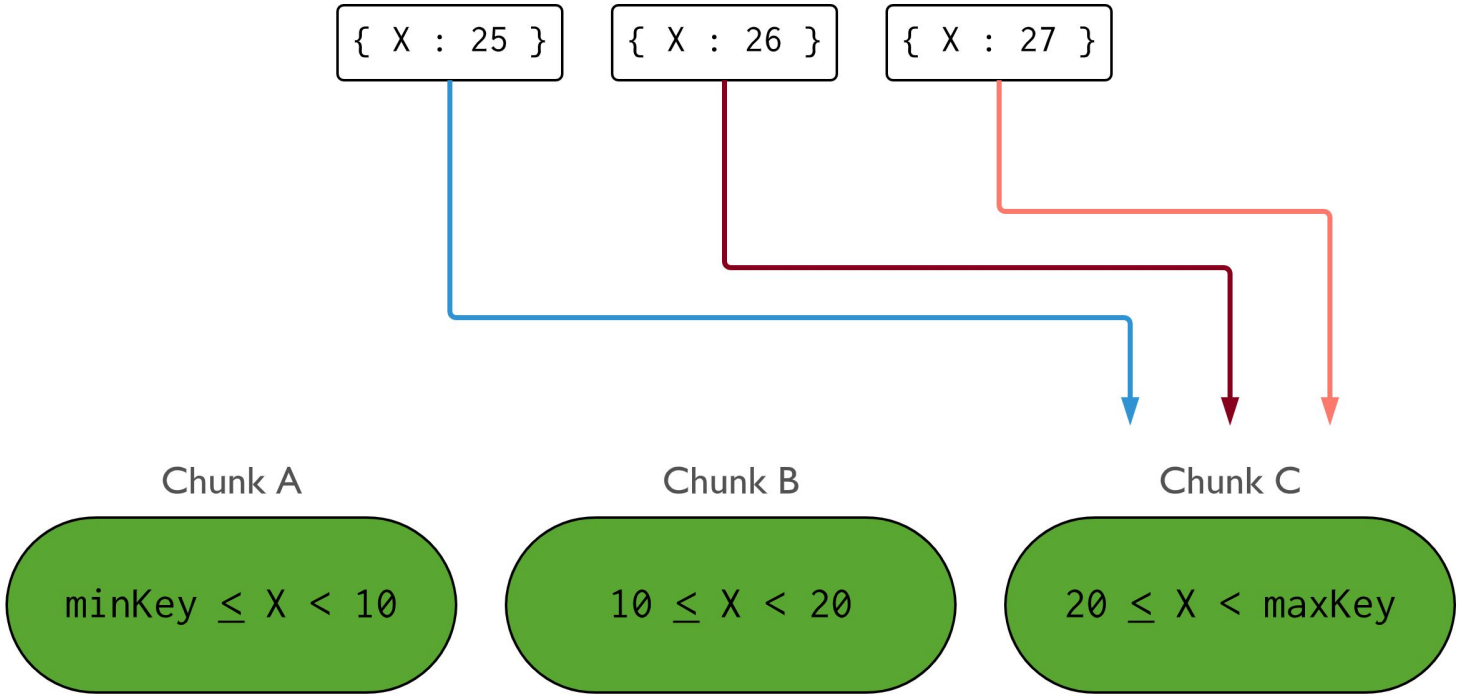
- Ranged Sharding

sh.shardCollection("database.collection", { <shard key> })

Example:

sh.shardCollection("vinodh.testData", { empNo: 1 })

Monotonically Changing Shard Key



```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("5d66c59d291f1f6349c928f6")
  }
  shards:
        {  "_id" : "shard01",  "host" : "shard01/localhost:27019,localhost:27020,localhost:27021",  "state" : 1 }
        {  "_id" : "shard02",  "host" : "shard02/localhost:27022,localhost:27023,localhost:27024",  "state" : 1 }
        {  "_id" : "shard03",  "host" : "shard03/localhost:27025,localhost:27026,localhost:27027",  "state" : 1 }
  active mongoses:
        "3.6.13-3.3" : 2
  autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                2 : Success
  databases:
        {  "_id" : "config",  "primary" : "config",  "partitioned" : true }
                config.system.sessions
                        shard key: { "_id" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shard01 1
                        { "_id" : { "$minKey" : 1 } } -->> { "_id" : { "$maxKey" : 1 } } on : shard01 Timestamp(1, 0)
        {  "_id" : "vinodh",  "primary" : "shard01",  "partitioned" : true }
                vinodh.testData2
                        shard key: { "num" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shard01 1
                                shard02 1
                                shard03 1
                        { "num" : { "$minKey" : 1 } } -->> { "num" : 3001 } on : shard02 Timestamp(2, 0)
                        { "num" : 3001 } -->> { "num" : 7001 } on : shard03 Timestamp(3, 0)
                        { "num" : 7001 } -->> { "num" : { "$maxKey" : 1 } } on : shard01 Timestamp(3, 1)
```

```
mongos> db.testData2.getShardDistribution()

Shard shard01 at shard01/localhost:27019,localhost:27020,localhost:27021
 data : 683KiB docs : 20000 chunks : 1
 estimated data per chunk : 683KiB
 estimated docs per chunk : 20000

Shard shard02 at shard02/localhost:27022,localhos
 data : 102KiB docs : 3000 chunks : 1
 estimated data per chunk : 102KiB
 estimated docs per chunk : 3000

Shard shard03 at shard03/localhost:27025,localhos
 data : 136KiB docs : 4000 chunks : 1
 estimated data per chunk : 136KiB
 estimated docs per chunk : 4000

Totals
 data : 922KiB docs : 27000 chunks : 3
 Shard shard01 contains 74.07% data, 74.07% docs in cluster, avg obj size on shard : 35B
 Shard shard02 contains 11.11% data, 11.11% docs in cluster, avg obj size on shard : 35B
 Shard shard03 contains 14.81% data, 14.81% docs in cluster, avg obj size on shard : 35B
```

```
mongos> use vinodh
switched to db vinodh
mongos> db.testData2.count()
20000
mongos> db.testData2.find({num:{$gt:7000}}).count()
13000
mongos>
```


PERCONA

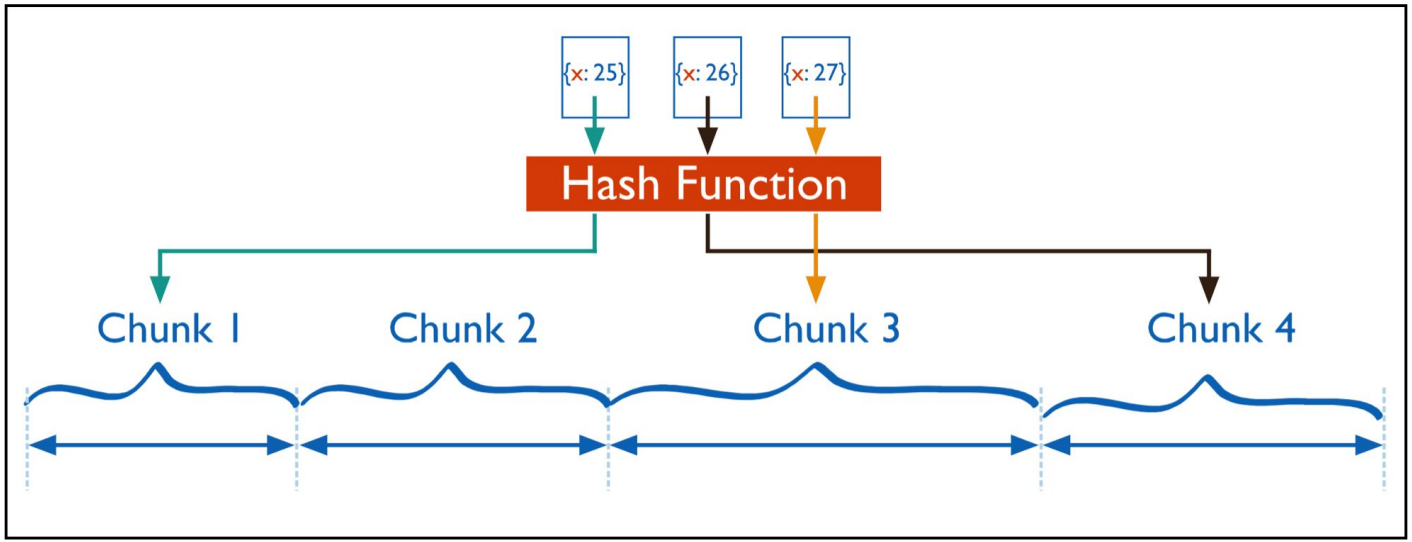
Ranged Sharding

- Contiguous ranges determined by Shard Key
- Default method
- Efficient when
  - ***High cardinality***
  - ***Low Shard Key frequency***
  - ***Non-Monotonically Changing Shard key***

Hashed Sharding

Strategy

- Hashed Sharding



Strategy

- Hashed Sharding uses hashed index

sh.shardCollection("database.collection", {

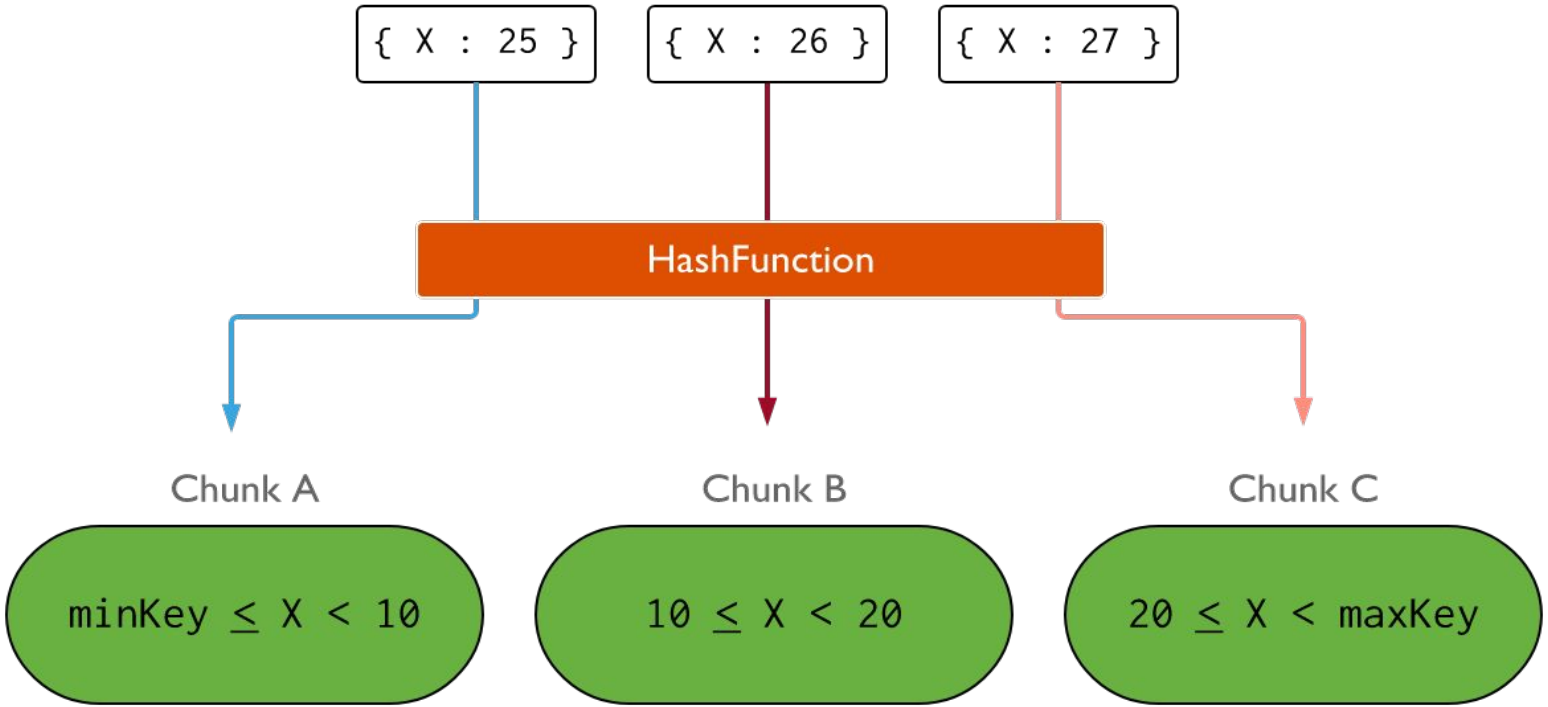
<field> : “hashed” })

<u>Example:</u>

sh.shardCollection("vinodh.testData",

{ UUIDfield: “hashed” })

Monotonically Changing Shard Key - Hash



Hashed Sharding

- Provides equal distribution
- Target vs Broadcast operations
- Ideal for fields that changes monotonically like ObjectID, timestamp etc

Zones

Why Zones?

- Isolate data set
- Geographically closest
- Route data to desired Shard h/w
- Distribute Writes based on DC

How to…

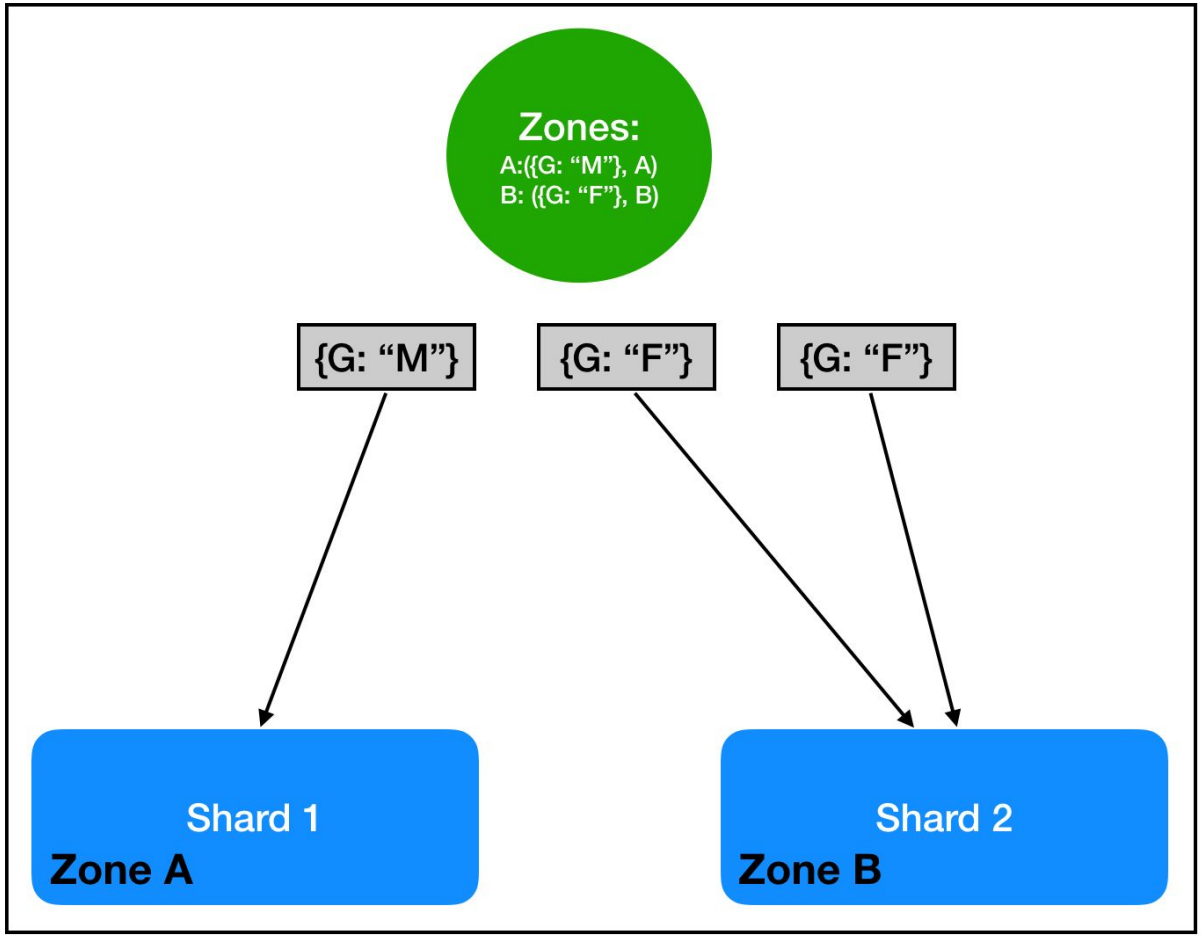
- First create **tagRange** with Shards

```
sh.addShardTag("shard01", "tag1")
sh.addShardTag("shard02", "tag2")
```

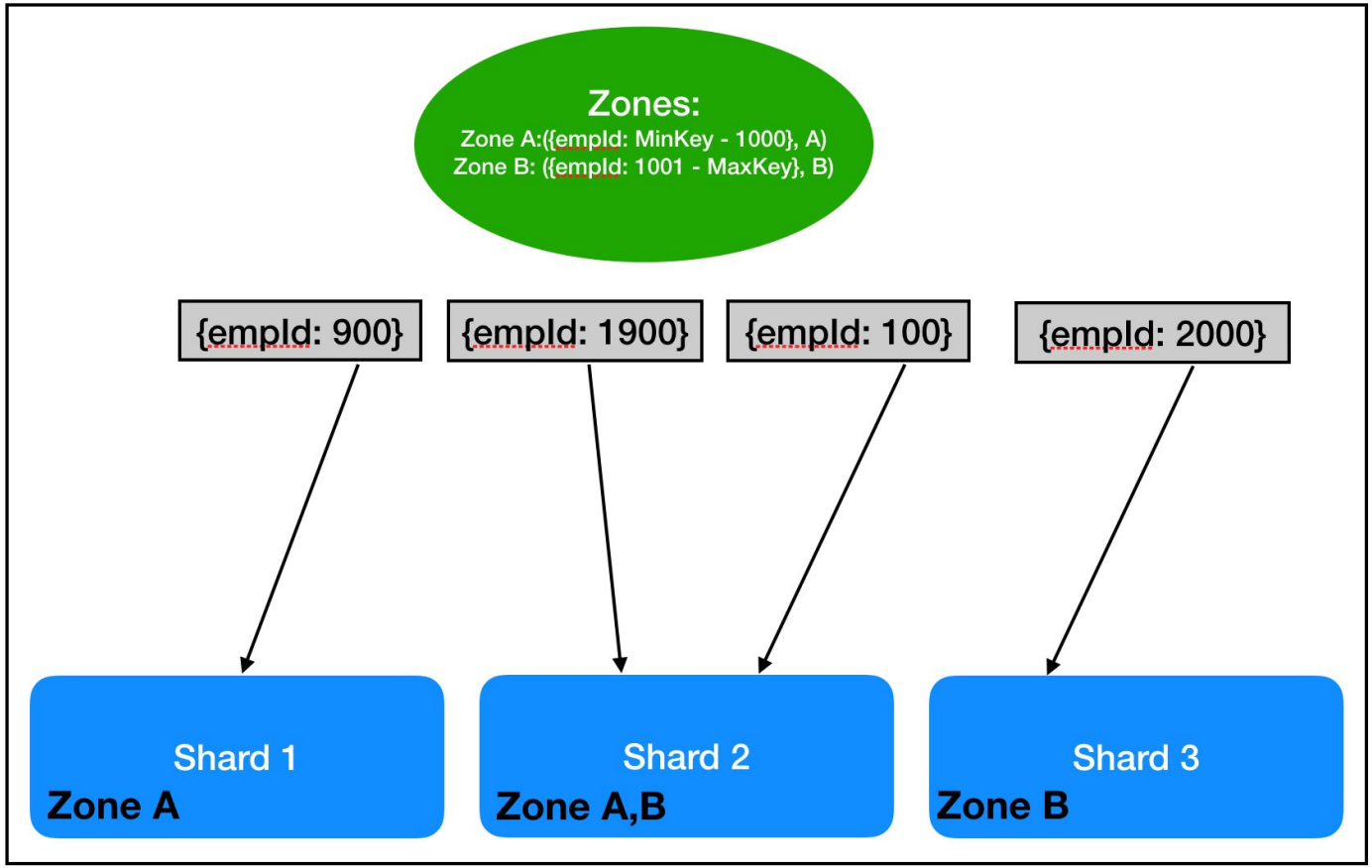
- Assign **tagRanges** to collections

```
sh.addTagRange(  "<database>.<collection>",
{ "gender" : "Male", "userid" : MinKey },
{ "gender" : "Male", "userid" : MaxKey },
"tag1")
```

Zone



Zone



HA overview

HA

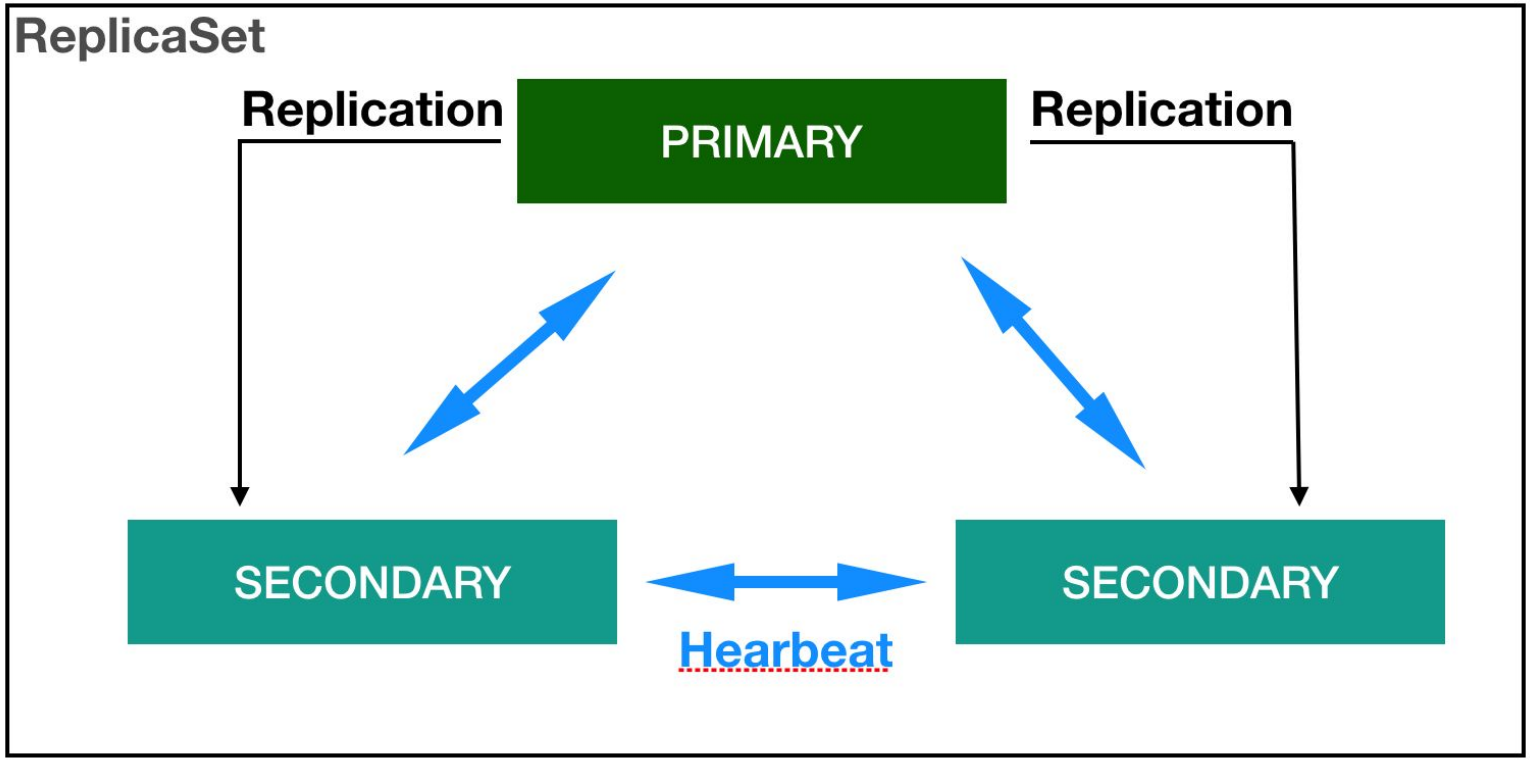
- *Instance down*
- *Loss of DC*
- *Network Partition*
- *Backup*
- *Etc.*

HA

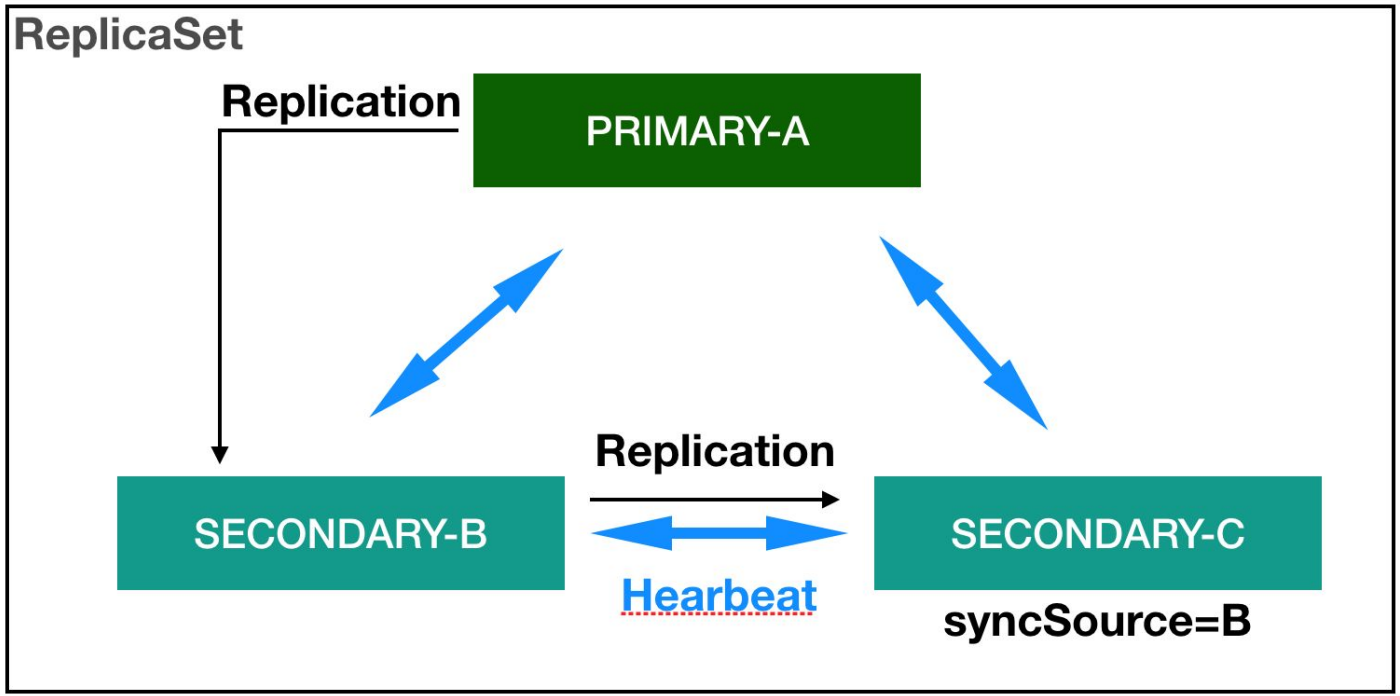
- Sharded Cluster itself is **NOT** HA
- SC + RS = Complete HA
- Failover Scenarios - Do your Exercise!

ReplicaSet - Key Points

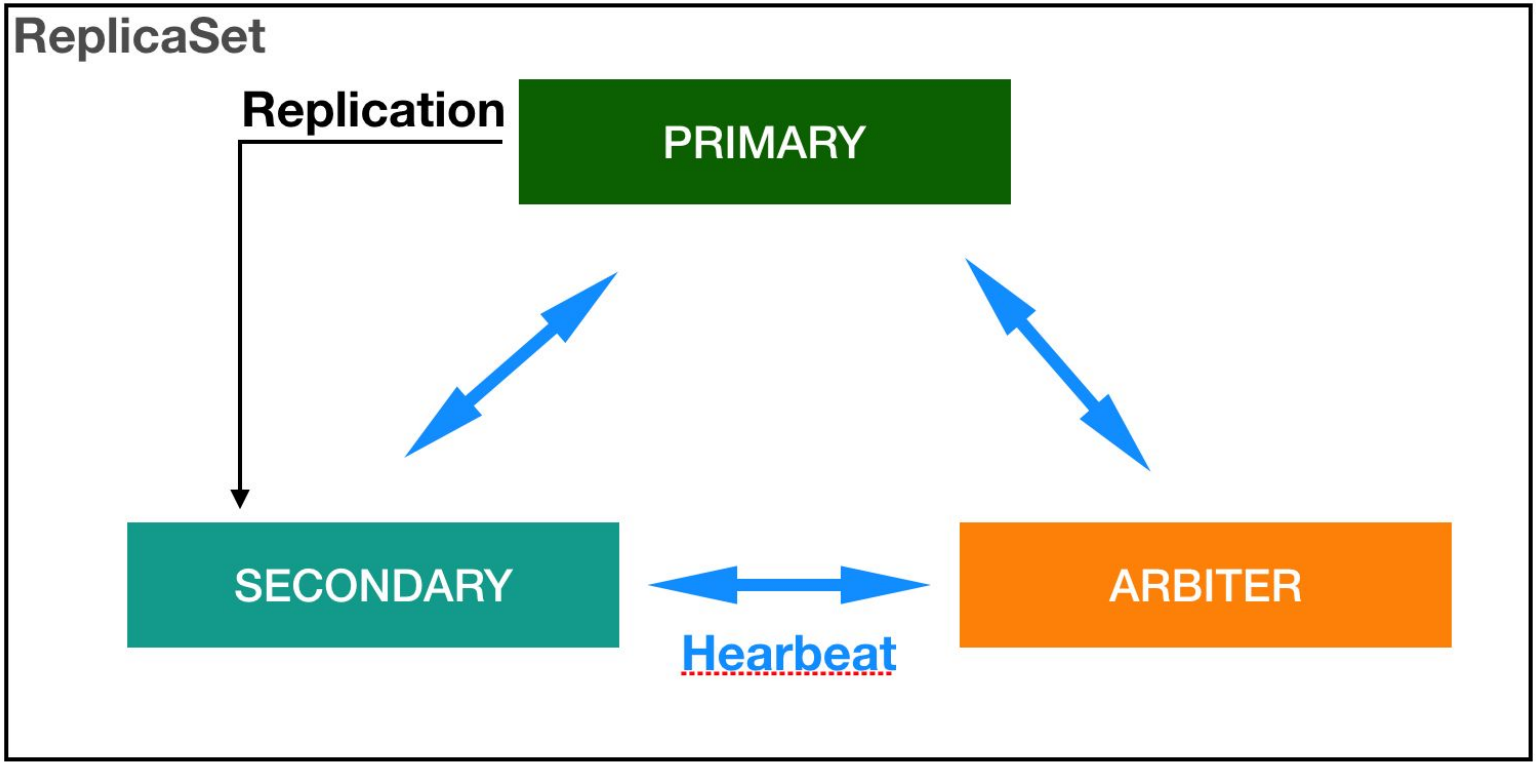
Replicaset


ReplicaSet
Replication
PRIMARY
Replication
SECONDARY
SECONDARY
Hearbeat

Replicaset



Replicaset - ARBITER



replicaSet Configuration

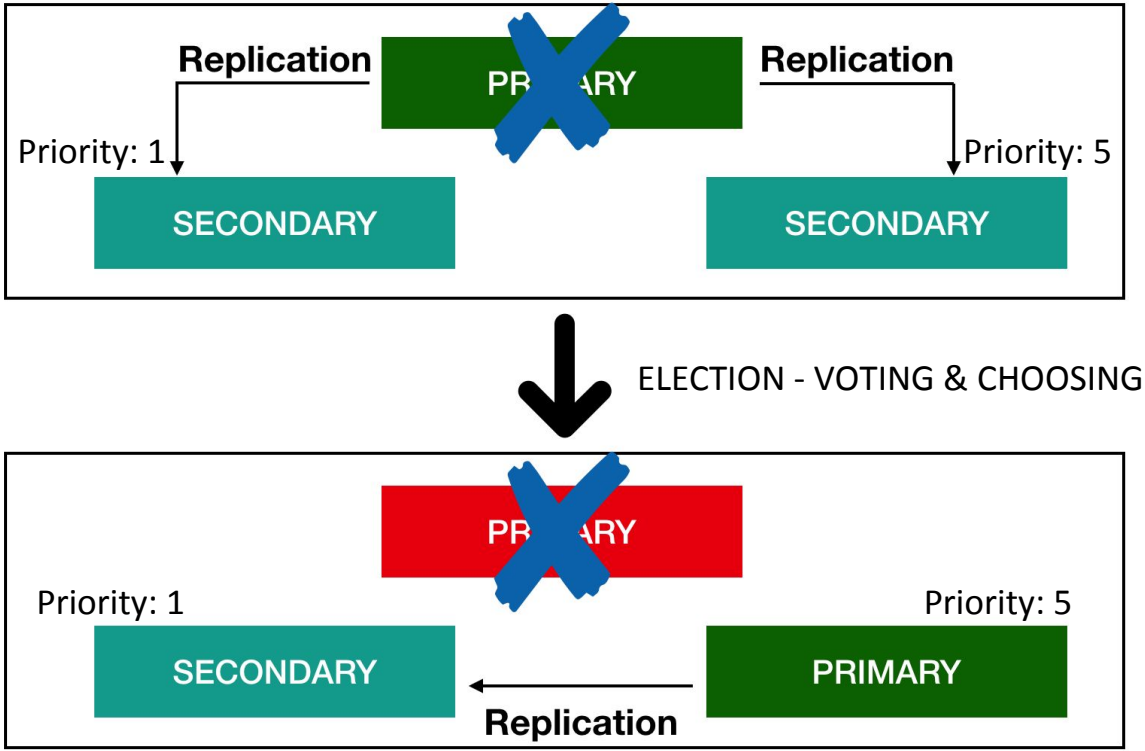
- Priority
- Hidden
- Delayed
- Arbiter

```
shard01:PRIMARY> rs.conf()
{
        "_id" : "shard01",
        "version" : 1,
        "protocolVersion" : NumberLong(1),
        "members" : [
                {
                        "_id" : 0,
                        "host" : "localhost:27019",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1,
                        "tags" : {

                        },
                        "slaveDelay" : NumberLong(0),
                        "votes" : 1
                },
                {
                        "_id" : 1,
                        "host" : "localhost:27020",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1,
                        "tags" : {

                        },
                        "slaveDelay" : NumberLong(0),
                        "votes" : 1
                },
                {
                        "_id" : 2,
                        "host" : "localhost:27021",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1,
                        "tags" : {

                        },
                        "slaveDelay" : NumberLong(0),
                        "votes" : 1
                }
        ],
```

```
{
        "_id" : 1,
        "host" : "localhost:27020",
        "arbiterOnly" : false,
        "buildIndexes" : true,
        "hidden" : false,
        "priority" : 1,
        "tags" : {

        },
        "slaveDelay" : NumberLong(0),
        "votes" : 1
},
```


PERCONA

Replicaset - Election+Priority



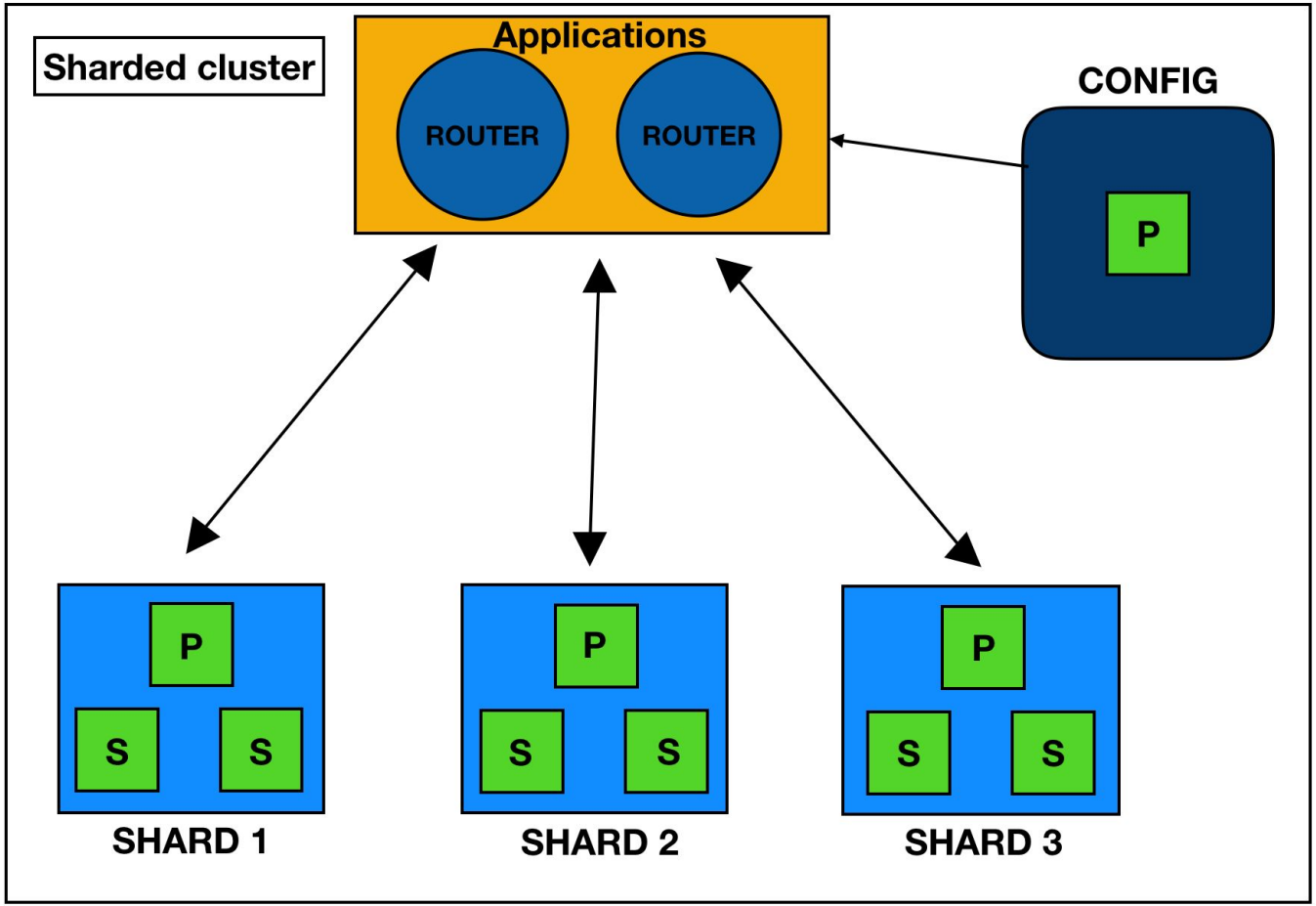
PERCONA

Points to Note

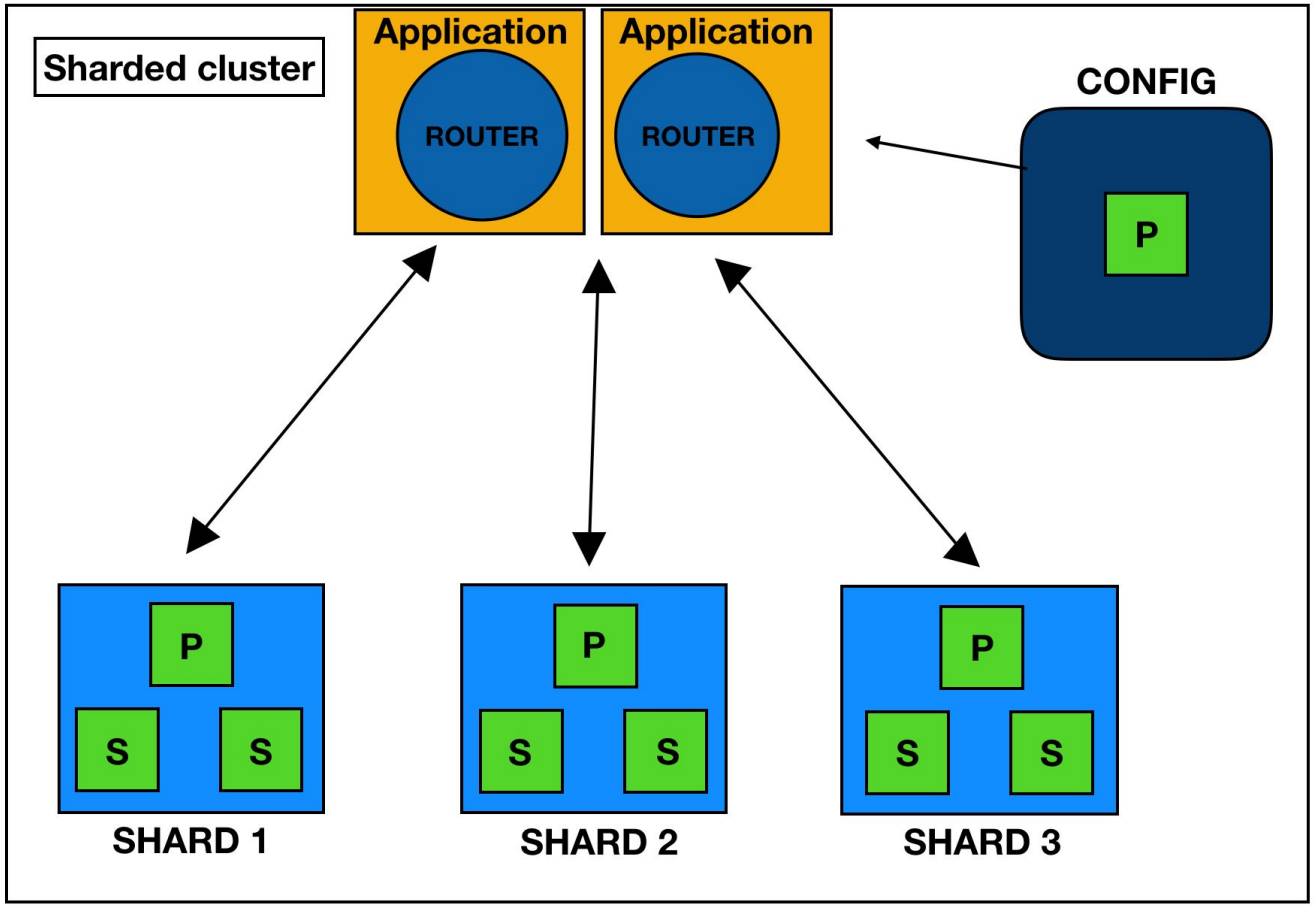
- Set proper **{Priority:n}** to elect a member as Primary
  - *Avoid setting **Priority** for all members*
- Set **{writeConcern:majority}** to avoid data loss
- Set **readPreference** for reads

Topology- Application Servers

Applications and mongos in one server

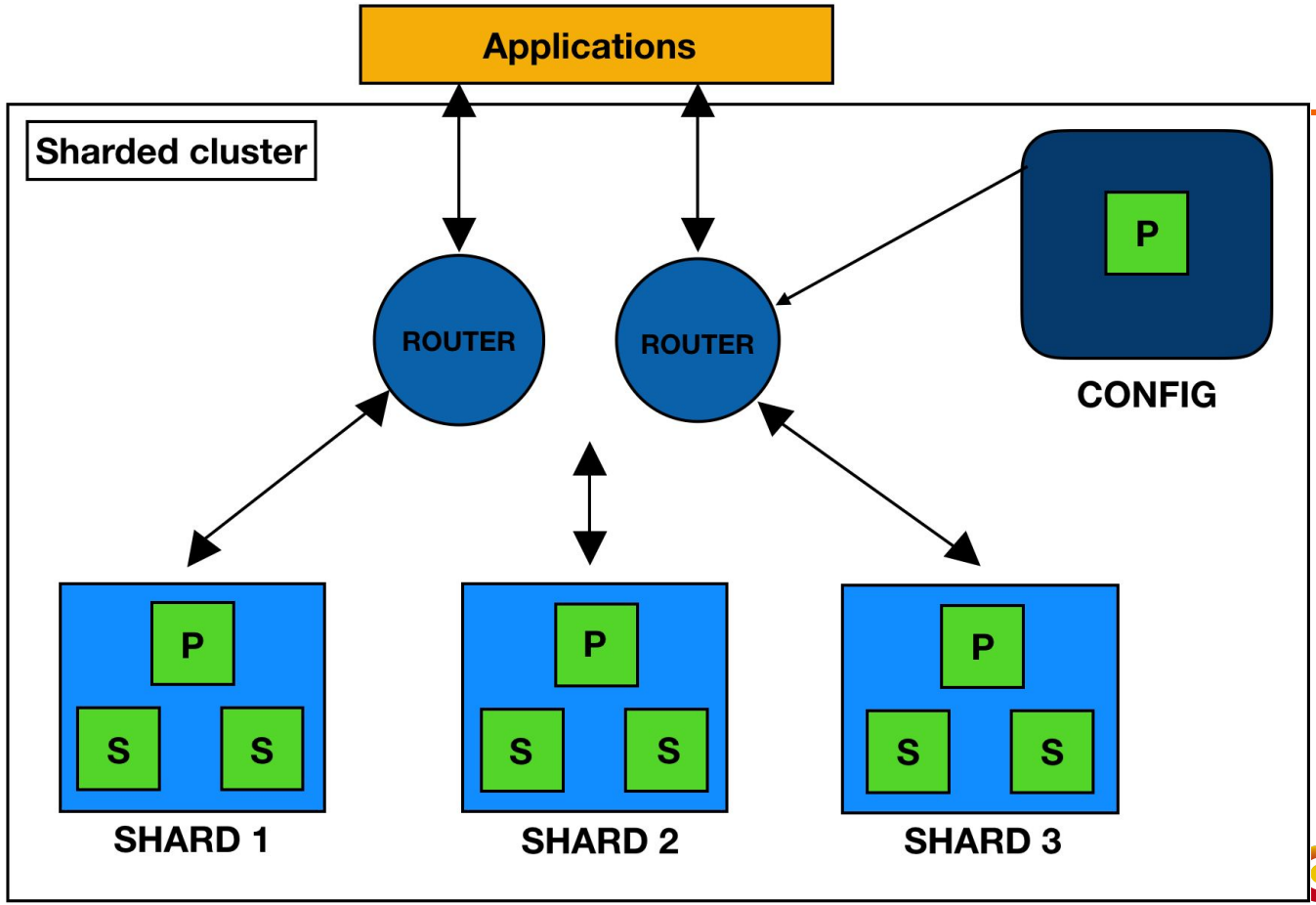


mongos/router in each Application servers



PERCONA

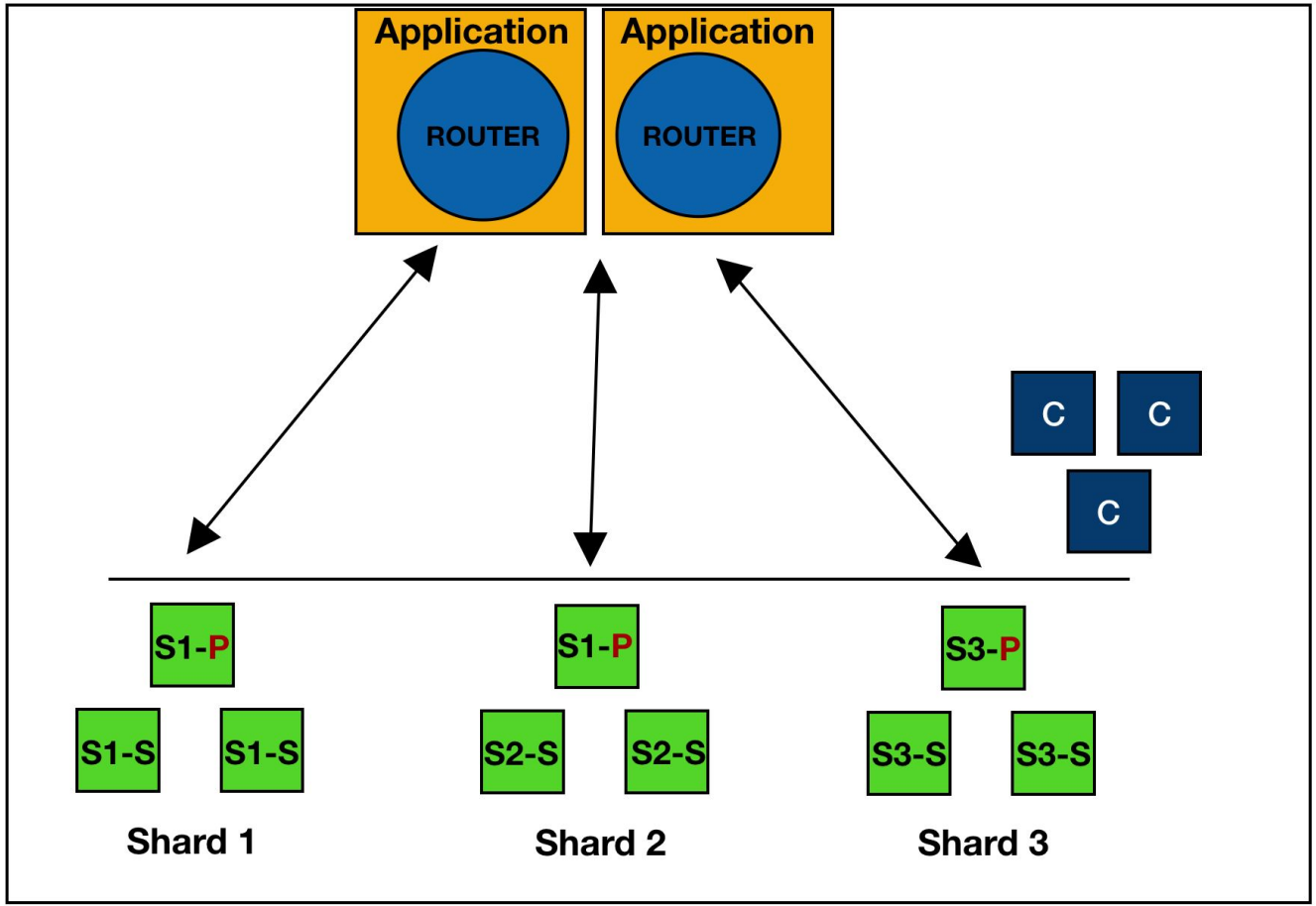
Sharded Cluster in Private network



PERCONA

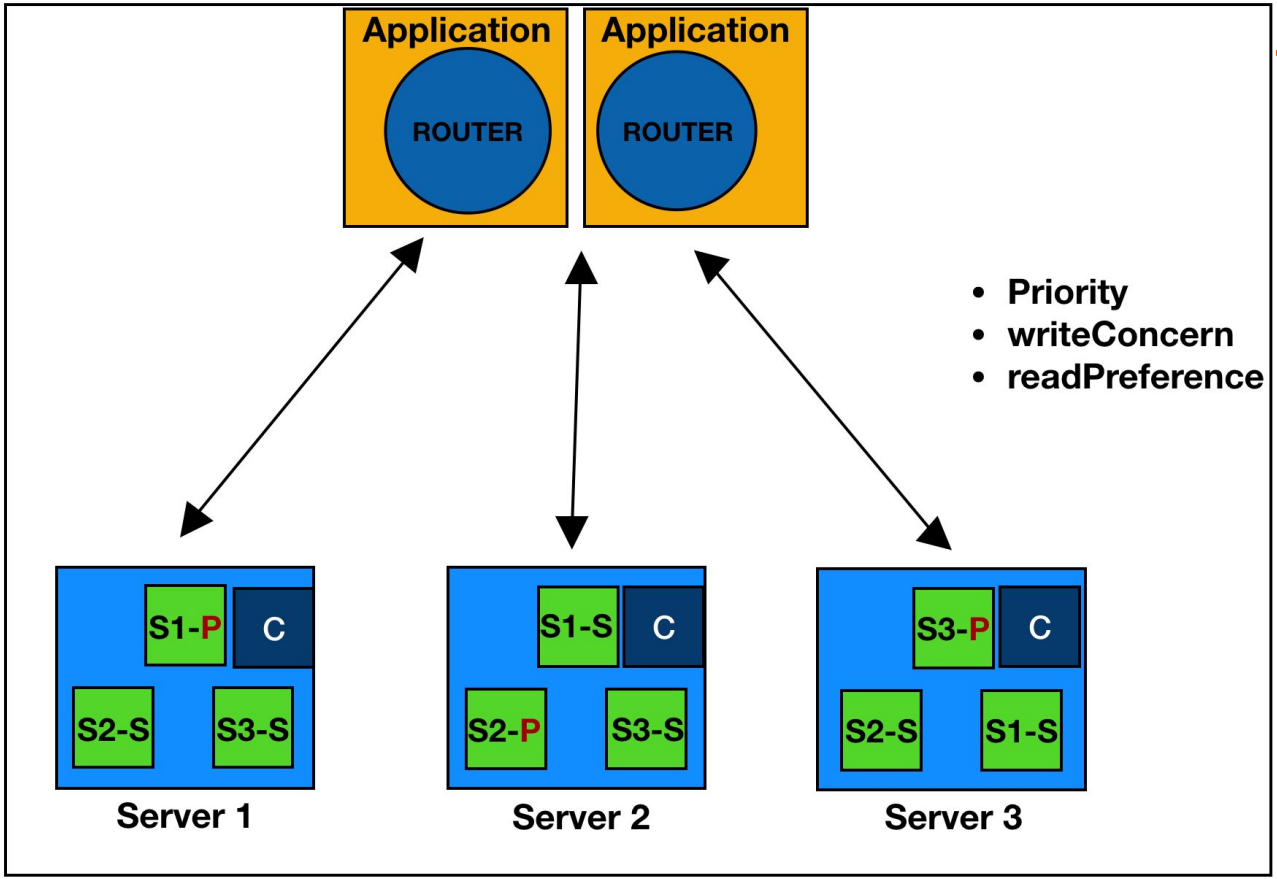
Topology- Shards

Shard instances deployed in dedicated servers design



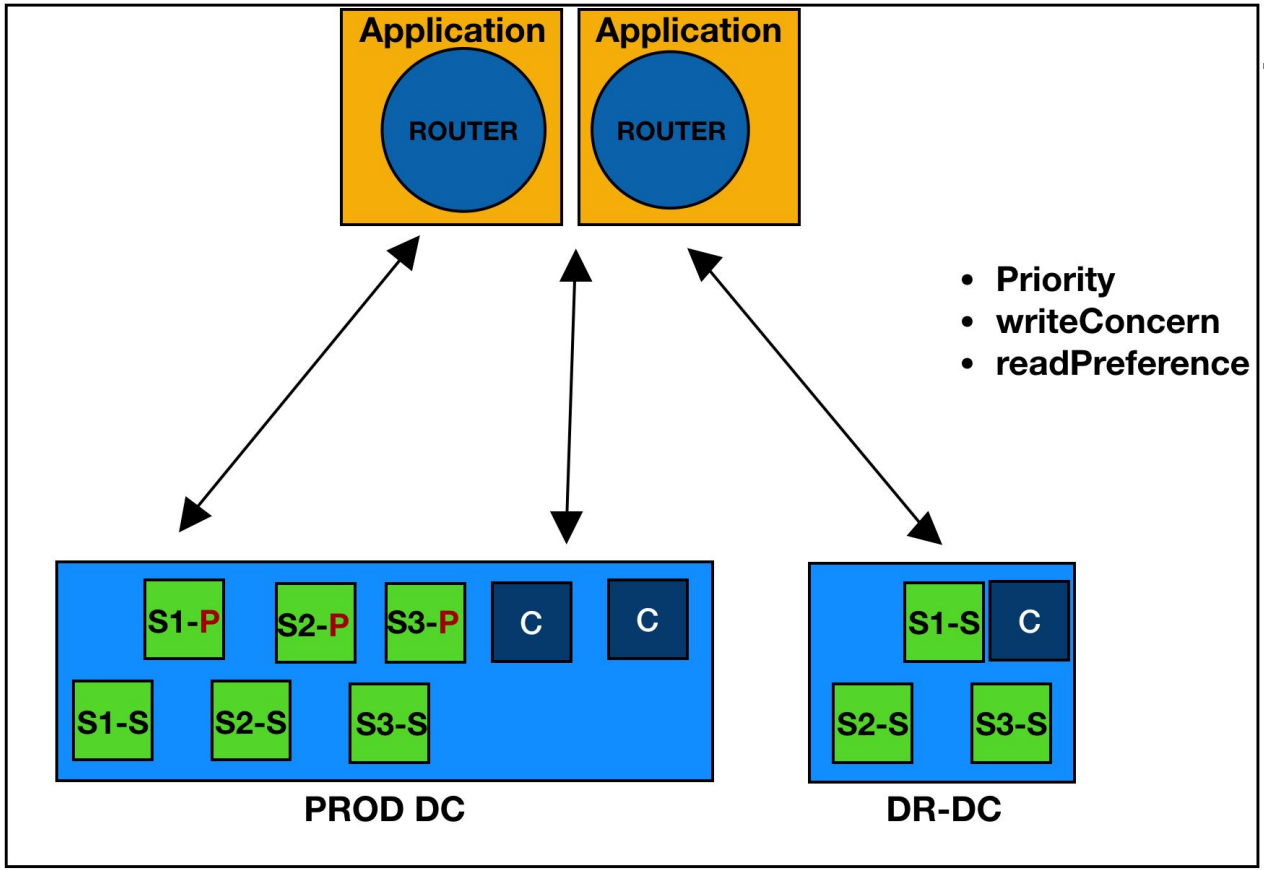
PERCONA

Multiple Shard members in each Server + HA design



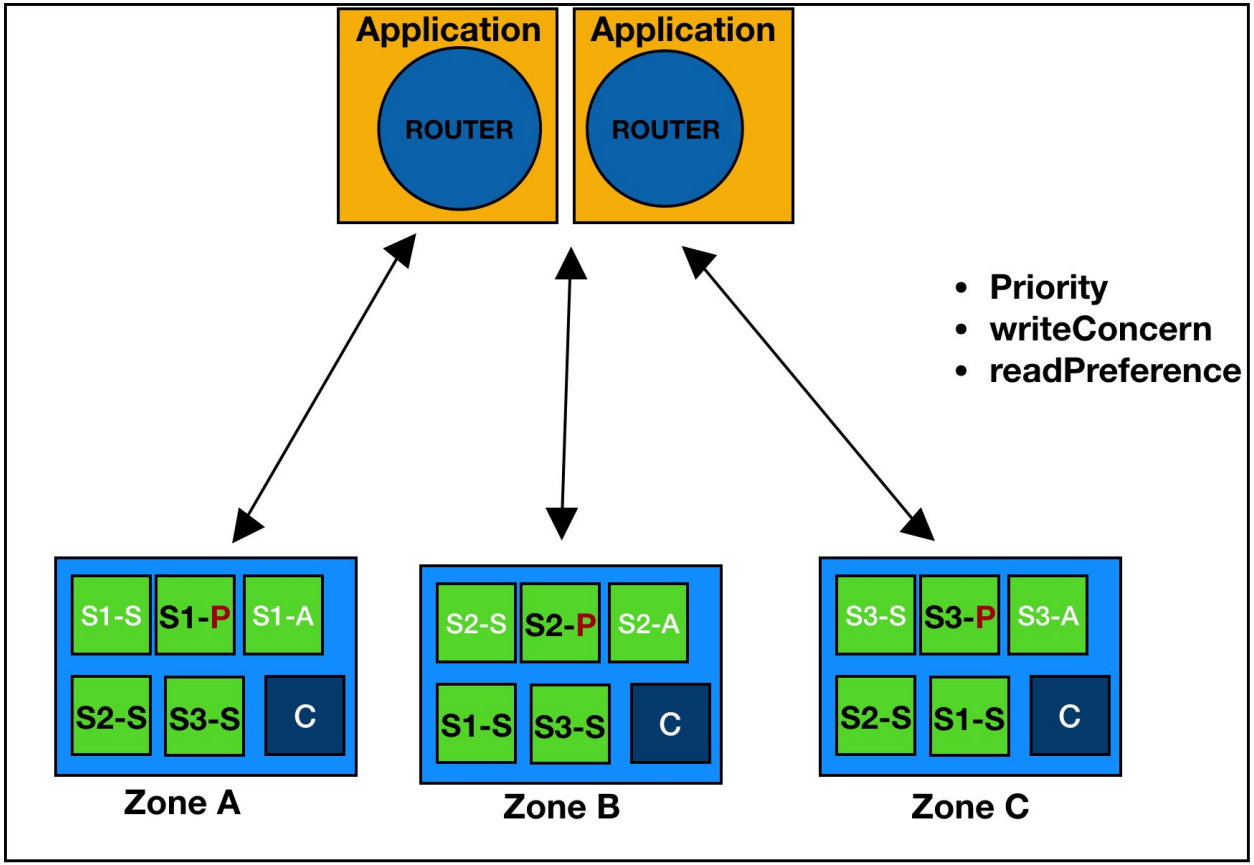
PERCONA

Production DC and DR DC + HA design



PERCONA

Using Zones / Tags + HA design



PERCONA

Points to Note

POINTS TO NOTE

- SHARD KEY - Pain Point!
  - *Monotonically increased key*
  - *Do your tests!*
- SHARD EARLY!
- ZONES

POINTS TO NOTE

BACKUP + RESTORE

https://www.percona.com/blog/2018/12/13/mongodb-backup-how-and-when-to-use-psmdb-hotbackup-and-mongodb_consistent_backup/

Question?

vinodh.krishnaswamy@percona.com
https://in.linkedin.com/in/vinodhkrish

THANK YOU