MongoDB Sharding Cluster And ycsb Testing on CentOS Machine

Veeravathni, Anugnu

anugnuv@cloud-evolutions.com

Abstract

In the technology facing world with increasing Automation and Machine Learning techniques, which stores millions of data to analysis, there is a need for new databases to store those large amounts of data. Due to this increasing size of databases and requirement for storing those large amounts of unstructured data, most of the companies are adopting NoSQL databases. Where MongoDB is one of the best among them which stores data as JSON like Documents. It’s common use cases are Internet of Things, Mobile, Catalogs and Content Management. The important thing is it supports Horizontal scaling through Sharding.



Contents

[I. Overview on MongoDB Cluster 2](#_Toc506486245)

[A. Components of Cluster: 2](#_Toc506486246)

[B. What is sharding and why? 2](#_Toc506486247)

[a) Sharding: 3](#_Toc506486248)

[b) Types of Sharding: 3](#_Toc506486249)

[II. Before You Install 3](#_Toc506486250)

[A. Instance Requirement: 3](#_Toc506486251)

[B. Preparing Instances: 3](#_Toc506486252)

[a) Adding Hosts in every Instance: 3](#_Toc506486253)

[b) Security: 4](#_Toc506486254)

[III. Installing MongoDB 4](#_Toc506486255)

[IV. Config Servers Replication 5](#_Toc506486256)

[Procedure: 5](#_Toc506486257)

[V. Sharding Servers 7](#_Toc506486258)

[Procedure 7](#_Toc506486259)

[VI. Query Router setup 8](#_Toc506486260)

[A. Creating User: 8](#_Toc506486261)

[B. Installation MongoDB from TarBall: 8](#_Toc506486262)

[C. Adding Config Servers and Shards to Mongos: 9](#_Toc506486263)

[D. Sharding Database and Collections Using Shard Key: 10](#_Toc506486264)

[VII. Testing 10](#_Toc506486265)

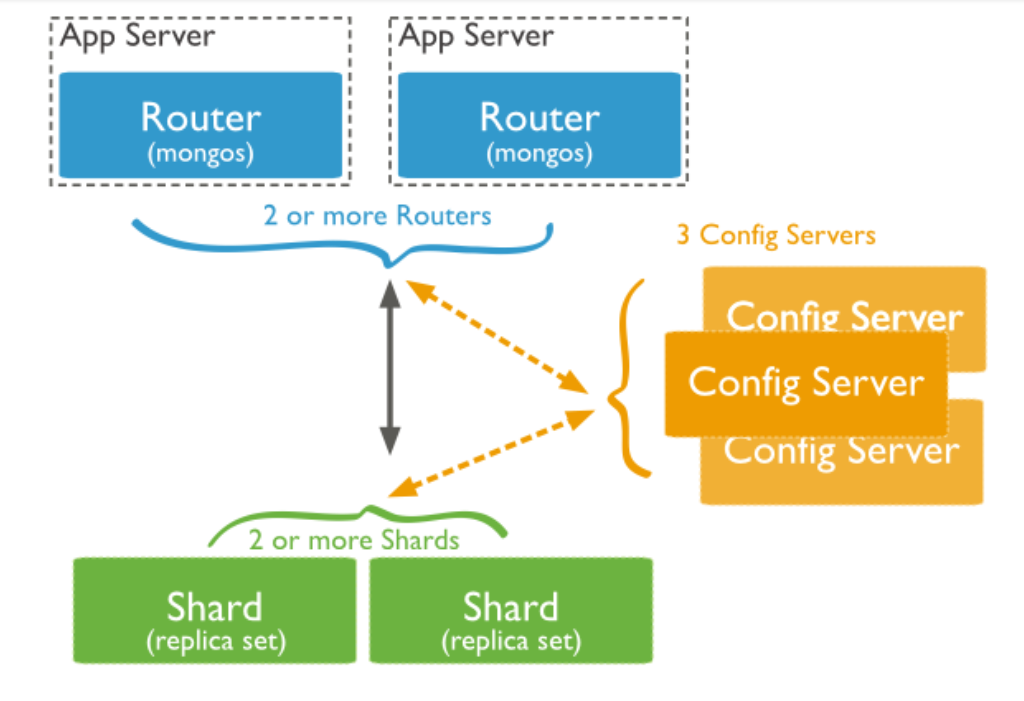
[References: 11](#_Toc506486266)

# Overview on MongoDB Cluster

Let’s talk a bit about Mongo Cluster before we start working on it.

## Components of Cluster:

1. Query Router: It is a query router which acts as interface for client and shard, we can have more than one router, but at least one per application. We can scale them as required.
2. Config Server: Stores metadata and config settings for cluster. Replica set of 3 is preferred for large applications or just one for testing purpose. We can also replicate them.
3. Shard Server: This is our actual database where all the data get stored. We can both scale and replicate them as per our requirement.



## What is sharding and why?

Sharding is a method for distributing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations. MongoDB supports horizontal scaling through sharding. Horizontal Scaling involves dividing the system dataset and load over multiple servers, adding additional servers to increase capacity as required.

### Sharding:

MongoDB shards at collection level. To distribute the documents in a collection, MongoDB partitions the collection using the shard key. A database can have a mixture of sharded and unsharded collections. Unsharded collections are stored on a primary shard.

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. MongoDB partitions sharded data into chunks. Each chunk has an inclusive lower and exclusive upper range based on the shard key.

### Types of Sharding:

Sharding are based on Range or Hash.

1. Range: evenly dividing data by their shard key ranges. A range of shard keys whose values are close are more likely to reside on the same chunk.
2. Hash: evenly dividing data by their shard key hash. hashing is done by itself. And chances are database loads are evenly distributed. While a range of shard keys may be close, their hashed values are unlikely to be on the same chunk.

# Before You Install

## Instance Requirement:

Here we are going to use 3 config server, 2 shard servers without replication and 1 query router, so we need 6 instances here. You can also go with minimum instances where 1 config server, 2 shards and 1 query router can be used, but make sure to modify things as per that.

## Preparing Instances:

* After you launch CentOS machine, add a user with admin rights.

Note: Run this commands from root or from user but then after the selinux and firewall settings, every command must be done from user with admin rights only, not from the root.

* Update the required system files.

yum update -y

### Adding Hosts in every Instance:

Now add all the hostnames to all the server you want to include in this cluster.

vi /etc/hosts

Press ESC button and then "i" to insert data into that file. It should look like this

192.168.50.xxx mongoconfig1

192.168.50.xxx mongoconfig2

192.168.50.xxx mongoconfig3

192.168.50.xxx mongoqueryrouter

192.168.50.xxx mongoshard1

192.168.50.xxx mongoshard2

### Security:

* Now we must disable SELINUX, to exempt the security issues while connecting.

vi /etc/selinux/config

You can change “selinux=enforcing” to “selinux=disabled” to open all communication ports, which is not a good practice though.

Or you can open the specific port, if you know which you are going to use.

Installing semanage package

yum install policycoreutils-python

opening tcp & udp ports on 27017, 27018 and 27019.

semanage port -a -t mongod\_port\_t -p tcp 27017

semanage port -a -t mongod\_port\_t -p tcp 27018

semanage port -a -t mongod\_port\_t -p tcp 27019

semanage port -a -t mongod\_port\_t -p udp 27017

semanage port -a -t mongod\_port\_t -p udp 27018

semanage port -a -t mongod\_port\_t -p udop 27019

* Then you must change firewall settings. Here you can open the required ports to let the communication between instances happen or can disable firewall if it’s just for testing.

systemctl stop firewalld

or open required port permenantly

firewall-cmd --zone=public --permanent --add-port=portnumber/udp

firewall-cmd --reload

To check the opened ports in firewall, type this

firewall-cmd --list-all

* Now restart the machine to apply the changes made, use reboot command or restart from GUI of your source.

# Installing MongoDB

Note: Below steps are for all servers except query routers, installation for query routers are quite different.

* As the instance restarted, login with admin user, not with root and follow the below procedure.
* Create a file in /etc with VI editor to copy required links for download and install it.

Get the updated links from official MongoDB document, as of now am using mongodb 3.6.

sudo vi /etc/yum.repos.d/mongodb-org-3.6.repo

* Press ESC button and then "i" to insert data into that file. Below are the details to copy, just right click on that file to paste it. Make sure it copied correctly. Then press ESC button followed by ":", then "wq" to write and quit from editor.

[mongodb-org-3.6]

name=MongoDB Repository

baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.6/x86\_64/

gpgcheck=1

enabled=1

gpgkey=https://www.mongodb.org/static/pgp/server-3.6.asc

* Before we move on, we should verify that the MongoDB repository exists within the yum utility. The repolist command displays a list of enabled repositories:

sudo yum repolist

* Now install MongoDB server in the instance. -y in the command replicate yes, if required in process.

sudo yum install -y mongodb-org

Note: Don't run MongoDB, just install and leave it. If you did start already, stop it.

sudo systemctl stop mongod

# Config Servers Replication

This must be done in all config servers. And make sure you stop the mongod service if you did start it.

## Procedure:

* + Go to mongod.conf and edit configuration details mentioned below.

sudo vi /etc/mongod.conf

Here change connection port and bindIp, to connect through that port and IP.

net:

port: 27019 (port for config replica)

bindIp: 192.168.50.xxx (your IP address)

Now change the replication, where change “myconfigset1” to whatever you want

replication:

replSetName: myconfigset1 (you name it)

Then add the sharding option, nothing to be change here, its default.

sharding:

clusterRole: configsvr

Things to take care of on above changes:

* Uncomment those sections you edited if commented before.
* 2 spaces are must before a new line in this mongod.conf file.
* Follow camelCase style in mongod.conf file.
  + Save it now and start mongod service. (ESC followed by :wq)

sudo systemctl start mongod

sudo systemctl restart mongod (if you already started before)

* + Now connect to mongo shell through your host and your specified port

mongo --host yourhostname --port 27019

* + If the above command executes, you enter mongo shell, you will see a '>' symbol where you can write a query to create, insert, delete or any command on database.
  + Now exit from mongo shell in all the config servers, except the one you consider as primary (it’s your choice) to execute further commands to form a replica set.
  + Now we have to initiate our replica set in the primary config server.

rs.initiate(

{

\_id: "yourconfigsetname",

configsvr: true,

members: [

{ \_id : 0, host : "yourconfig1hostname:27019" },

{ \_id : 1, host : " yourconfig2hostname:27019" },

{ \_id : 2, host : " yourconfig3hostname:27019" }

]

}

)

* + You should probably see "ok : 1", if it worked. You can check the config server status using this command.

rs.status()

* + Any error occured, stop the mongod service, restart server, restart the mongod and run it. Hopefully it works.
  + Now you are all set with Replicating your configuration servers.

# Sharding Servers

This are the server were our database will store, we can have replication set for them. It is recommended to have replica set in real time.

## Procedure

* Stop the mongod service, if you did start it.
* Do this in all primary shards.
* Go to mongod.conf and edit below details.

sudo vi /etc/mongod.conf

Here change connection port and bindIp, to connect through that port and IP.

net:

port: 27018 (port for config replica)

bindIp: 192.168.50.xxx (your IP address)

Now change the replication, where “myshardset” is the name I gave to my set, change it as you want.

replication:

replSetName: myshardset (you name it)

Then add the sharding option, nothing to be change here, its default.

sharding:

clusterRole: shardsvr

* Start the mongod service now

sudo systemctl start mongod (If you haven't started yet).

sudo systemctl restart mongod (If you already started before).

* + Now connect to mongo shell through your host and your specified port

mongo --host yourhostname --port 27018

* + You have entered mongo shell, you will see a '>' where you can query you db. Now we have to initiate our replica set.

rs.initiate(

{

\_id: "yourconfigsetname",

members: [

{ \_id : 0, host : "yourconfig1hostname:27018" },

{ \_id : 1, host : " yourconfig2hostname:27018" },

{ \_id : 2, host : " yourconfig3hostname:27018" }

]

}

)

* + You should probably see "ok : 1", if it worked. You can check the config server status using this command.

rs.status()

# Query Router setup

This is a different story, we can install MongoDB with regular procedure, but it didn’t work with me, it will cause lot of errors while setting up query router, such as default location for database and some other settings. You can try a regular installation, if it doesn’t work go with this.

## Creating User:

Do this commands from root.

Create a 'mongod' user and add that user to 'mongod' group which is already exist. Then add that user to wheel group, which gives him sudo permissions.

su root (it prompts for password)

useradd username (it should be ‘mongod’)

passwd username (it will prompt for password, enter it)

usermod -aG mongod mongod (adding mongod user to mongod group)

usermod -aG wheel mongod (adding mongod user to wheel group)

su mongod (changing user to ‘mongod’)

## Installation of MongoDB from Tar Ball:

* Update the YUM repository before downloading mongodb.

sudo yum -y update

* Download mongodb zip file. Get updated link from official mongodb site.

sudo curl -O https://fastdl.mongodb.org/linux/mongodb-linux-x86\_64-3.6.2.tgz

* Extract that file to /opt directory.

sudo tar -zxvf mongodb-linux-x86\_64-3.6.2.tgz -C /opt

* rename that file or move it to /opt/mongod

sudo mv givepathofextractdonemineis/opt/mongodb-linux-x86\_64-3.6.2/ /opt/mongod

* Change mode and get permissions for that folder to our 'mongod' user

sudo chmod -R 777 /opt/mongod (-R is recursive permission)

sudo chown -R mongod:mongod /opt/mongod (owning for user:group 'mongod')

* Create a directory data/db to store mongodb database in it, then own it to mongod

sudo mkdir -p /data/db (-p will create a parent folder)

sudo chmod -R 777 /data/db (777 will give all rwdx auth)

sudo chown -R mongod:mongod /data/db (-R is for recursive)

* Now go to bashrc and add the environment variable for mongod, to inform path for execution file.

sudo vi ~/.bashrc (to edit in bashrc)

Here, we must add the path of mongod. Where mine is export PATH=/opt/mongod/bin:$PATH

export PATH=/yourpath/bin:$PATH

To check list of environmental variables

printenv

## Adding Config Servers and Shards to Mongos:

* Now let’s connect config servers to mongos,

mongos --configdb yourconfigsetname/yourconfig1hostname:27019, yourconfig2hostname:27019, yourconfig3hostname:27019 --bind\_ip 192.168.50.xxx

The above IP address is of your query router.

* Now open another session of your query router without closing it and connect to mongo shell

mongo --host hostname --port 27017

* Now in the mongo shell add the shard servers using shardsvr group or individually

sh.addShard("yourshardsetname/yourshardhostname:27017")

(or)

sh.addShard("shardhostname:27017")

* Now let’s check the logs on every other server to make sure it worked.

Go to every server and check logs.

sudo tail -f /var/log/mongodb/mongod.log

You will see connected messages in the logs.

* To remove a sharded server from mongos.

db.adminCommand({removeShard:shardID}) (you can find it in sh.status())

## Sharding Database and Collections Using Shard Key:

Now we have to add the shards database and collections, to shard the incoming data. Remember, you should always connect config severs to run any command in mongos, so the metadata related to that need to be stored. Enabling sharding for a database does not redistribute data but make it possible to shard the collections in that database. To distribute the documents in a collection, MongoDB partitions the collection using the shard key. For more information on shard key <https://docs.mongodb.com/manual/core/sharding-shard-key/>

* Now let’s shard database.

sh.enableSharding("yourdatabasename")

* Now let’s shard collection.

sh.shardCollection("yourdatabasename.yourcollectionname", { yourkeyname : preferredvalue } )

# Testing Sharded Cluster

* To test the cluster, first you need to connect to mongos, connect to config servers, then open mongo shell in duplicate window.

In query router.

mongos –configdb yourconfigsetname/yourconfig1hostname:27019, yourconfig2hostname:27019, yourconfig3hostname:27019 --bind\_ip 192.168.50.xxx

In duplicate query router or another query router session

mongo --host hostname --port 27017

* Now you entered mongos, you have to create a database and shard it.

use yourdatabasename

db.createCollection("nameit", {capped: false})

sh.enableSharding(“yourdatabasename”)

To check the status of that database.

sh.status()

You will see the database has been added to any of the shard.

* Now let’s shard collection with shard key. Be careful while doing this, because once the shard key is assigned, we can’t change it.

sh.shardCollection("yourdatabasename.yourcollectionname", { name : 1 } )

Now check status of your collection

sh.status()

You will see the collection has been added with shard key name in the list.

* Now let’s add some data to test sharding.

db.yourcollectionname.save({

“name”: “anyname”,

“shop”: [“amazon”, ”bestbuy”, ”walmart”],

})

* This will be added to any of the shard, now go to your shard servers and check databases list by typing

db or show dbs (to list databases)

use yourdatabasename (to enter your database)

db.yourcollectionname.find() (to list data)

* You will see the data added in one of the servers.
* Keep adding some more data to know the data has being sharding to other servers too.

# YCSB (Yahoo! Cloud Serving Benchmark)

It is difficult to decide which system is right for your application, partially because the features differ between systems, and partially because there is not an easy way to compare the performance of one system versus another. So, we have YCSB tool to compare common set of workloads for evaluating the performance of different “key-value” and “cloud” serving stores. It is an open-source specification and program suite for evaluating retrieval and maintenance capabilities of computer programs. It is often used to compare relative performance of NoSQL database management systems.

The original benchmark was developed by Brian Cooper in the research division of Yahoo! who released it in 2010 with the stated goal of "facilitating performance comparisons of the new generation of cloud data serving systems", particularly for transaction-processing workloads which differed from ones measured by benchmarks designed for more traditional database management systems.

YCSB comes with 6 out of the box workloads, each testing a different common use case.

Workload A: Update heavy workload This workload has a mix of 50/50 reads and writes. An application example is a session store recording recent actions.

Workload B: Read mostly workload This workload has a 95/5 reads/write mix. Application example: photo tagging; add a tag is an update, but most operations are to read tags.

Workload C: Read only This workload is 100% read. Application example: user profile cache, where profiles are constructed elsewhere (e.g., Hadoop).

Workload D: Read latest workload in this workload, new records are inserted, and the most recently inserted records are the most popular. Application example: user status updates; people want to read the latest.

Workload E: Short ranges in this workload, short ranges of records are queried, instead of individual records. Application example: threaded conversations, where each scan is for the posts in a given thread (assumed to be clustered by thread id).

Workload F: Read-modify-write.

The YCSB client can be installed on any machine and can able to connect using the ip address of the host machine or server to be tested, we can perform testing operations remotely. But it is suggested to have client and testing server to be in the same network to get accurate results.

## Installation

The YCSB tool requires latest version of JAVA and Maven to be installed in the machine, so we first install those and then extract YCSB tool from Internet.

* Install Java or OpenJDK, am installing OpenJDK here.

sudo yum install java-1.8.0-openjdk

sudo yum install java-1.8.0-openjdk-devel

java -version

* Installing Maven:

Install Wget, which is a tool used to download using external links.

sudo yum install wget

Change directory to local.

cd /usr/local

Now download Maven files.

wget <http://www-eu.apache.org/dist/maven/maven-3/3.5.2/binaries/apache-maven-3.5.2-bin.tar.gz>

Extract Maven.

sudo tar xzf apache-maven-3.5.2-bin.tar.gz

sudo ln -s apache-maven-3.5.2 maven

Set Environment Variable

sudo vi /etc/profile.d/maven.sh

It opens an editor, add below details in it

export M2\_HOME=/usr/local/maven

export PATH=${M2\_HOME}/bin:${PATH}

Check version

mvn -version

* Download and extract YCSB

curl -O --location https://github.com/brianfrankcooper/YCSB/releases/download/0.5.0/ycsb-0.5.0.tar.gz

Extract YCSB

tar xfvz ycsb-0.5.0.tar.gz

Change directory to ycsb-xversion

cd ycsb-0.5.0

## Performance Testing and Evaluation

After successful setting up MongoDB Sharding Cluster and installing YCSB, now it’s time to test the machines optimal capability and performance.

### Things to Remember, before running the test:

* Operation type (load/run)
* Workload types (a,b,c,d,e,f)

Example: workloads/workloada, workloads/workloadb

* Recordcount (value for no. of records to load)

Example: recordcount=100000

* Operationcount (value for no. of records to run)

Example: operationcount=100000

* Threads (no. of ycsb client threads/parallel operations)

Example: threads 2/4/8/16/32/64

* mongodb.url="" (If you are running from another server)

mongodb.url="mongodb://user:pwd@server1.example.com:9999,server2.example.com:9999/dbname" -p mongodb.auth="true"

* '-s' is for status.
* '-p' is for loading property files.

### Testing with different workloads:

We run workloads depending on our requirement. Let’s Run some default workloads with synchronous and asynchronous process.

Below is workload type ‘a’, with sync and async modes, where test data stores into the file mentioned at the end.

./bin/ycsb load mongodb-async -s -P workloads/workloada > outputLoad.txt

./bin/ycsb run mongodb-async -s -P workloads/workloada > outputRun.txt

./bin/ycsb load mongodb -s -P workloads/workloada > outputLoad.txt

./bin/ycsb run mongodb -s -P workloads/workloada > outputRun.txt

Now Let’s Run some specific workloads to test the machines capabilities.

Here am taking workloads ‘a’ & ‘b’ to test with different properties. We observe ops/sec by increasing no of threads and operations count in the workload, which will let us know the machines capability.

#### Workload A & B with Increasing Threads: -

Workload A:

./bin/ycsb load mongodb -s -P workloads/workloada -p recordcount=1000000 -threads 16 > outputLoadA.txt

For remote host:

./bin/ycsb load mongodb -s -P workloads/workloada -p mongodb.url="mongodb://192.168.50.xxx:27017/dbname" -p mongodb.auth="true" -p recordcount=1000000 -threads 16 > outputLoadA.txt

./bin/ycsb run mongodb -s -P workloads/workloada -p operationcount=1000000 -threads 2 > outputRunA.txt

./bin/ycsb run mongodb -s -P workloads/workloada -p operationcount=1000000 -threads 8 > outputRunA.txt

./bin/ycsb run mongodb -s -P workloads/workloada -p operationcount=1000000 -threads 32 > outputRunA.txt

Workload B:

./bin/ycsb load mongodb -s -P workloads/workloadb -p recordcount=1000000 -threads 16 > outputLoadB.txt

./bin/ycsb run mongodb -s -P workloads/workloadb -p operationcount=1000000 -threads 2 > outputRunB.txt

./bin/ycsb run mongodb -s -P workloads/workloadb -p operationcount=1000000 -threads 8 > outputRunB.txt

./bin/ycsb run mongodb -s -P workloads/workloadb -p operationcount=1000000 -threads 32 > outputRunB.txt

Keep on increasing the threads, until you see a decrease in the ops/sec, then you have to increase/decrease threads in minimum value possible. From those results decide number of thread counts best suitable.

#### Workload A & B with increasing Operation Count: -

Now from the above test, we decide on best thread count, so keeping that value constant, we change number of operations count to see the maximum possible operations by our machine. Let’s say 16 is the maximum thread count where our machine works properly.

Workload A:

./bin/ycsb load mongodb -s -P workloads/workloada -p recordcount=1000000 -threads 16 > outputLoadA.txt

./bin/ycsb run mongodb -s -P workloads/workloada -p operationcount=100000 -threads 16 > outputRunA.txt

./bin/ycsb run mongodb -s -P workloads/workloada -p operationcount=1000000 -threads 16 > outputRunA.txt

./bin/ycsb run mongodb -s -P workloads/workloada -p operationcount=10000000 -threads 16 > outputRunA.txt

Workload B:

./bin/ycsb load mongodb -s -P workloads/workloadb -p recordcount=1000000 -threads 16 > outputLoadB.txt

./bin/ycsb run mongodb -s -P workloads/workloadb -p operationcount=100000 -threads 16 > outputRunB.txt

./bin/ycsb run mongodb -s -P workloads/workloadb -p operationcount=1000000 -threads 16 > outputRunB.txt

./bin/ycsb run mongodb -s -P workloads/workloadb -p operationcount=10000000 -threads 16 > outputRunB.txt

By increasing and decreasing number of operationcount, we can come to a point that at what maximum values our machine is accurate.

After

# References:

MongoDB official site: <https://docs.mongodb.com/manual/tutorial/deploy-shard-cluster/>

Sharding Cluster: <https://www.howtoforge.com/tutorial/deploying-mongodb-sharded-cluster-on-centos-7/>

YCSB Setup: <https://scalegrid.io/blog/how-to-benchmark-mongodb-with-ycsb/>

YCSB Loads: <https://github.com/brianfrankcooper/YCSB/wiki/Running-a-Workload>