# Mongodb

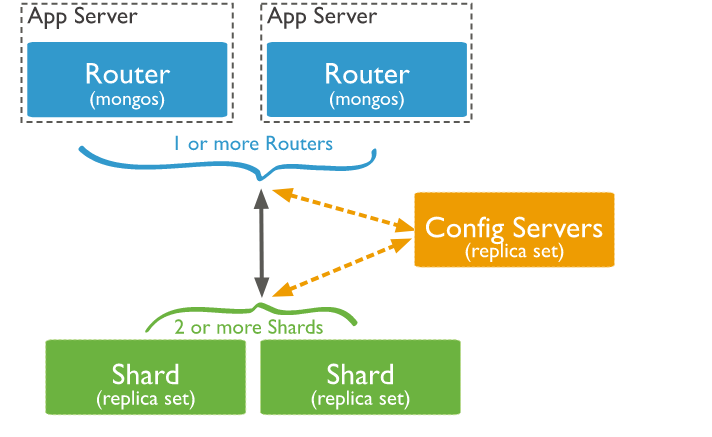
## 版本配置副本集+分片

### 一：概念

在搭建集群之前，需要首先了解几个概念：路由，分片，副本集，配置服务器等

相关概念

先看一张图



从图中可以看到有四个组件：mongos，config server，shard，replica set。

**Mongos：**数据库集群请求的入口，所有的请求都通过mongos进行协调，不需要在应用程勋添加一个路由选择器，mongos自己就是一个请求分发中心，它负责把对应的数据请求请求转发到对应的shard服务器上，在生产环境通常有多mongos作为请求的入口，防止其中一个挂掉所有的mongodb请求都没有办法操作。

**Config server：**顾名思义为配置服务器，存储所有数据库元信息（路由，分片）的配置。Mongos本身没有物理存储分片服务器和数据路由信息，只是缓存在内存里，配置服务器则实际存储这些数据。Mongos第一次启动或者关掉重启就会从config server 加载配置信息，以后如果配置服务器信息变化会通知到所有的mongos更新自己的状态，这样mongos就能继续准备路由，在生产环境通常有多个config server 配置服务器，因为它存储了分片路由的元数据，防止数据丢失！

**Shard：** 分片（sharding）是指将数据库拆分，将其分散在不同的机器上的过程。将数据分散到不同的机器上，不需要功能强大的服务器就可以存储更多的数据和处理更大的负载。基本思想就是将集合切成小块，这些块分散到诺干片里，每个偏只负责总数据的一部分，最后通过一个均衡器来对各个分片进行均衡（数据迁移）。

**Replica set:** 中文翻译副本集，其实就是shard的备份，防止shard挂掉之后数据丢失。复制提供了数据的冗余备份，并在多个服务器上存储数据副本，提高了数据的可用性，并可以保证数据的安全性。

**仲裁者（Arbiter）,是复制集中的一个MongoDB实例，它并不保存数据，仲裁节点使用最小的资源并且不要求硬件设备，不能将Arbiter部署在同一个数据集节点中，可以部署在其他应用服务器或者监视服务器中，也可以部署在单独的虚拟机中，为了确保复制集中有技术的投票成员（包括primary），需要添加仲裁节点作为投票，否则primary不能运行时不会自动切换primary。**

**简单了解之后，我们可以这样总结一下，应用请求mongos来操作mongodb的增删改查，配置服务器存储数据库元信息，并且和mongos做同步，数据最终存入在shard（分片）上，为了防止数据丢失同步在副本集中存储了一份，仲裁在数据存储到分片的时候决定存储到哪个节点**

### 二：准备环境

### Os：Red Hat Enterprise Linux Server release 6.5 (Santiago)

**Software**：mongodb-linux-x86\_64-rhel62-3.4.4.tgz

vim /etc/profile

export MONGODB\_HOME=/usr/local/mongodb

export PATH=$MONGODB\_HOME/bin:$PATH#

source /etc/profile

### 端口分配

**mongos： 27017**

**config： 21000**

**shard1： 27001**

**shard2： 27002**

**shard3： 27003**

### 三：服务器规划

| **192.168.51.36** | **192.168.51.37** | **192.168.51.38** |
| --- | --- | --- |
| mongos | mongos | mongos |
| config server | config server | config server |
| shard server1 主节点 | shard server1 副节点 | shard server1 仲裁 |
| shard server2 仲裁 | shard server2 主节点 | shard server2 副节点 |
| shard server3 副节点 | shard server3 仲裁 | shard server3 主节点 |

### 安装MongoDB

**#解压**

**tar -xzvf mongodb-linux-x86\_64-3.4.6.tgz -C /usr/local/**

**#改名**

**mv mongodb-linux-x86\_64-3.4.6 mongodb**

### 分别在每台机器建立conf、mongos、config、shard1、shard2、shard3六个目录，因为mongos不存储数据，只需要建立日志文件目录即可。

**mkdir -p /usr/local/mongodb/conf**

**mkdir -p /usr/local/mongodb/mongos/log**

**mkdir -p /usr/local/mongodb/config/data**

**mkdir -p /usr/local/mongodb/config/log**

**mkdir -p /usr/local/mongodb/shard1/data**

**mkdir -p /usr/local/mongodb/shard1/log**

**mkdir -p /usr/local/mongodb/shard2/data**

**mkdir -p /usr/local/mongodb/shard2/log**

**mkdir -p /usr/local/mongodb/shard3/data**

**mkdir -p /usr/local/mongodb/shard3/log**

### 配置环境变量

**vim /etc/profile**

**# 内容**

**export MONGODB\_HOME=/usr/local/mongodb**

**export PATH=$MONGODB\_HOME/bin:$PATH**

**# 使立即生效**

**source /etc/profile**

### Config server配置服务器（三台机器）

mongodb3.4以后要求配置服务器也创建副本集，不然集群搭建不成功，特别注意此处和3.4以前的版本有不一样

添加配置文件（三台服务器都要进行添加）

#### vim /software/mongodb/conf/config.conf

**## 配置文件内容**

**pidfilepath = /usr/local/mongodb/config/log/configsrv.pid**

**dbpath = /usr/local/mongodb/config/data**

**logpath = /usr/local/mongodb/config/log/congigsrv.log**

**logappend = true**

**bind\_ip = 0.0.0.0**

**port = 21000**

**fork = true**

**#declare this is a config db of a cluster;**

**configsvr = true**

**#副本集名称**

**replSet=configs**

**#设置最大连接数**

**maxConns=20000**

启动三台服务器的config server

mongod -f /usr/local/mongodb/conf/config.conf

登录任意一台conf配置中心

mongo --port 20000

**config = {\_id : "configs", members : [{\_id : 0, host : "192.168.51.36:21000" },{\_id : 1, host : "192.168.51.37:21000" },{\_id : 2, host : "192.168.51.38:21000" }]};**

初始化副本集

rs.initiate(config);

### 五：配置分片副本集（三台机器）

#### 设置第一个副本集分片

**vi /usr/local/mongodb/conf/shard1.conf**

**#dfilepath = /usr/local/mongodb/shard1/log/shard1.pid**

**dbpath = /usr/local/mongodb/shard1/data**

**logpath = /usr/local/mongodb/shard1/log/shard1.log**

**logappend = true**

**bind\_ip = 0.0.0.0**

**port = 27001**

**fork = true**

**#打开web监控**

**httpinterface=true**

**rest=true**

**#副本集名称**

**replSet=shard1**

**#declare this is a shard db of a cluster;**

**shardsvr = true**

**#设置最大连接数**

**maxConns=20000**

启动

mongod -f /usr/local/mongodb/conf/shard1.conf

登陆任意一台服务器，初始化副本集，不要登录到arbiterOnly这台机器

mongo --port 20000

use admin

**config = {\_id : "shard1",members : [{\_id : 0, host : "192.168.51.38:27001" },{\_id : 1, host : "192.168.51.37:27001" },{\_id : 2, host : "192.168.51.36:27001", arbiterOnly: true }]}**

初始化副本集

rs.initiate(config);

查看状态

rs.status();

#### 配置第二个副本集分片

**vi /usr/local/mongodb/conf/shard2.conf**

**#配置文件内容**

**#——————————————–**

**pidfilepath = /usr/local/mongodb/shard2/log/shard2.pid**

**dbpath = /usr/local/mongodb/shard2/data**

**logpath = /usr/local/mongodb/shard2/log/shard2.log**

**logappend = true**

**bind\_ip = 0.0.0.0**

**port = 27002**

**fork = true**

**#打开web监控**

**httpinterface=true**

**rest=true**

**#副本集名称**

**replSet=shard2**

**#declare this is a shard db of a cluster;**

**shardsvr = true**

**#设置最大连接数**

**maxConns=20000**

启动

mongod -f /software/mongodb/conf/shard2.conf

登陆任意一台服务器，初始化副本集，不要登录到arbiterOnly这台机器

mongo --port 20000

use admin

**config = {\_id : "shard2",members : [{\_id : 0, host : "192.168.51.38:27002" },{\_id : 1, host : "192.168.51.36:27002" },{\_id : 2, host : "192.168.51.37:27002", arbiterOnly: true }]}**

初始化副本集

rs.initiate(config);

查看状态

rs.status();

#### 设置第三个副本集分片

**vi /usr/local/mongodb/conf/shard3.conf**

**#配置文件内容**

**#=======================**

**pidfilepath = /usr/local/mongodb/shard3/log/shard3.pid**

**dbpath = /usr/local/mongodb/shard3/data**

**logpath = /usr/local/mongodb/shard3/log/shard3.log**

**logappend = true**

**bind\_ip = 0.0.0.0**

**port = 27003**

**fork = true**

**#打开web监控**

**httpinterface=true**

**rest=true**

**#副本集名称**

**replSet=shard3**

**#declare this is a shard db of a cluster;**

**shardsvr = true**

**#设置最大连接数**

**maxConns=20000**

启动

mongod -f /software/mongodb/conf/shard3.conf

登陆任意一台服务器，初始化副本集，不要登录到arbiterOnly这台机器

mongo --port 20000

use admin

**config = {\_id : "shard3",members : [{\_id : 0, host : "192.168.51.36:27003" },{\_id : 1, host : "192.168.51.37:27003" },{\_id : 2, host : "192.168.51.38:27003", arbiterOnly: true }]}**

初始化

rs.initiate(config);

查看状态

rs.status();

### 六：配置路由器服务器mongos

先启动配置服务器和分片服务器，后启动路由实例

启动路由实例

vi /software/mongodb/conf/mongos.conf

**#dfilepath = /usr/local/mongodb/mongos/log/mongos.pid**

**logpath = /usr/local/mongodb/mongos/log/mongos.log**

**logappend = true**

**bind\_ip = 0.0.0.0**

**port = 20000**

**fork = true**

**#监听的配置服务器,只能有1个或者3个 configs为配置服务器的副本集名字**

**configdb = configs/192.168.51.36:21000,192.168.51.37:21000,192.168.51.38:21000**

**#设置最大连接数**

**maxConns=20000**

启动服务

mongos -f /software/mongodb/conf/mongos.conf

启用分片

目前搭建了mongodb配置服务器、路由服务器，各个分片服务器，不过应用程序连接到mongos路由服务器并不能使用分片机制，还需要在程序里设置分片配置，让分片生效。

mongo --port 20000

use admin

**sh.addShard("shard1/192.168.51.36:27001,192.168.51.37:27001,192.168.51.38:27001")**

**sh.addShard("shard2/192.168.51.36:27002,192.168.51.37:27002,192.168.51.38:27002")**

**sh.addShard("shard3/192.168.51.36:27003,192.168.51.37:27003,192.168.51.38:27003")**

查看状态

sh.status();

### 七：测试

目前配置服务、路由服务、分片服务、副本集服务都已经串联起来了，但我们的目的是希望插入数据，数据能够自动分片。连接在mongos上，准备让指定的数据库、指定的集合分片生效。

#指定testdb分片生效

Use admin

db.runCommand( { enablesharding :"testdb"});

#指定数据库里需要分片的集合和片键

db.runCommand( { shardcollection : "testdb.table1",key : {id: 1} } )

我们设置testdb的 table1 表需要分片，根据 id 自动分片到 shard1 ，shard2，shard3 上面去。要这样设置是因为不是所有mongodb 的数据库和表 都需要分片！

测试分片配置结果

我们设置testdb的 table1 表需要分片，根据 id 自动分片到 shard1 ，shard2，shard3 上面去。要这样设置是因为不是所有mongodb 的数据库和表 都需要分片！

测试分片配置结果

mongo 127.0.0.1:27017

#使用testdb

use testdb1;

#插入测试数据

**db.table1.save({id:1111,"test1":"testval1"});**

**db.table1.save({id:2222,"test1":"testval2"});**

**db.table1.save({id:3333,"test1":"testval3"});**

**db.table1.save({id:4444,"test1":"testval4"});**

**db.table1.save({id:5555,"test1":"testval5"});**

#查看分片情况如下，部分无关信息省掉了

db.table1.stats();

{

"sharded" : true,

"ns" : "testdb.table1",

"count" : 100000,

"numExtents" : 13,

"size" : 5600000,

"storageSize" : 22372352,

"totalIndexSize" : 6213760,

"indexSizes" : {

"\_id\_" : 3335808,

"id\_1" : 2877952

},

"avgObjSize" : 56,

"nindexes" : 2,

"nchunks" : 3,

"shards" : {

"shard1" : {

"ns" : "testdb.table1",

"count" : 42183,

"size" : 0,

...

"ok" : 1

},

"shard2" : {

"ns" : "testdb.table1",

"count" : 38937,

"size" : 2180472,

...

"ok" : 1

},

"shard3" : {

"ns" : "testdb.table1",

"count" :18880,

"size" : 3419528,

...

"ok" : 1

}

},

"ok" : 1

}

### 八：启动关闭

mongodb的启动顺序是，先启动配置服务器，在启动分片，最后启动mongos.

**mongod -f /software/mongodb/conf/config.conf**

**mongod -f /software/mongodb/conf/shard1.conf**

**mongod -f /software/mongodb/conf/shard2.conf**

**mongod -f /software/mongodb/conf/shard3.conf**

**mongos -f /software/mongodb/conf/mongos.conf**

关闭时，直接killall杀掉所有进程

**killall mongod**

**killall mongos**

注意分片的时候由于chunksize值默认是64MB，所以需要插入大量的数据才能够进行分片，此处将chunksize值设置小点，能够看到明显的分片效果

修改chunksize进行测试分片

use config

db.settings.save( { \_id:"chunksize", value: 10 } )

### 九：添加用户

mongos> use admin

switched to db admin

mongos> db.createUser({user:'admin',pwd:'123456', roles:[{role:'root', db:'admin'}]})

Successfully added user: {

"user" : "admin",

"roles" : [

{

"role" : "root",

"db" : "admin"

}

]

}

mongos> use dzswjdb

switched to db dzswjdb

mongos> db.createUser({user:'dzswjdb',pwd:'123456', roles:[{role:'readWrite', db:'dzswjdb'}]})

Successfully added user: {

"user" : "dzswjdb",

"roles" : [

{

"role" : "readWrite",

"db" : "dzswjdb"

}

]

}

mongos> use sbzsdb

switched to db sbzsdb

mongos> db.createUser({user:sbzsdb,pwd:'123456', roles:[{role:'readWrite', db:sbzsdb}]})

Successfully added user: {

"user" : "sbzsdb",

"roles" : [

{

"role" : "readWrite",

"db" : "sbzsdb"

}

]

}

mongos>

### 九：启动认证

将所有的服务先进行关闭

创建key存放的路径

mkdir /usr/local/mongodb/key

然后在所有的shard，config，mongos的配置文件中添加如下配置

clusterAuthMode=keyFile

keyFile=/usr/local/mongodb/key/mongdb-keyfile

生成密码配置文件

openssl rand -base64 100 > /usr/local/mongodb/key/mongdb-keyfile

chmod 600 /usr/local/mongodb/key/mongdb-keyfile

将密码文件复制到另外的两台机器上

启动mongo集群，先启动配置服务器---->分片服务器----> mongos服务器

验证认证是否生效

[root@mongodb1 mongos]# mongo --port 20000

MongoDB shell version v3.4.4

connecting to: mongodb://127.0.0.1:20000/

MongoDB server version: 3.4.4

mongos> use admin

switched to db admin

mongos> show users;

2017-11-14T11:04:48.072+0800 E QUERY [thread1] Error: not authorized on admin to execute command { usersInfo: 1.0 } :

\_getErrorWithCode@src/mongo/shell/utils.js:25:13

DB.prototype.getUsers@src/mongo/shell/db.js:1537:1

shellHelper.show@src/mongo/shell/utils.js:752:9

shellHelper@src/mongo/shell/utils.js:659:15

@(shellhelp2):1:1

mongos> db.auth('admin','123456')

1

mongos> show users;

{

"\_id" : "admin.admin",

"user" : "admin",

"db" : "admin",

"roles" : [

{

"role" : "root",

"db" : "admin"

}

]

}

mongos>

### 十：参考文档

<https://www.cnblogs.com/ityouknow/p/7344005.html>

<https://docs.mongodb.com/manual/core/index-creation/>

### 十一：常用命令

导入导出通过最高权限来进行导入导出，就不会报认证错误

导出命令参数

**-h IP**

**-u 用户**

**-p 端口**

**-d 数据库名字**

**-c 集合名字**

**-o 导出的文件名字**

**-q 导出过滤条件**

#### Mongodb导出sbzsdb库数据

**mongodump -h 192.168.51.38 -u admin -p 123456 -d dzswjdb -o /usr/local/src/ --authenticationDatabase admin**

#### mongodb导入sbzs库数据

**mongorestore -h 192.168.51.38 -u admin -p 123456 -d dzswjdb /usr/local/src/dzswjdb --authenticationDatabase admin**

导入数据

mongorestore /usr/local/src/

#### Mongodb导出单张表

**mongodump -h 156.16.42.133 -u admin -p 123456 -d dzswjdb -c CodeDataConfiguration -o /usr/local/src/codedataconfiguration --authenticationDatabase admin**

#### ---创建数据库 然后创建表

**use sbzsdb**

**db.sbzsdb.insert({'name':'sbzstest'})**

**db.sbzs.insert({'name':'sbzstest'})**

#### --删除数据库

**db**

**db.dropDatabase();**

#### 集合操作相当于rdbms中的table

**db.createCollection('CodeDataConfiguration')**

**db.getCollection('CodeDataConfiguration').find({})**

**db.getCollection('CodeDataConfiguration').find({'\_id':ObjectId("595f3159b8555378d7513b32")})**

**db.getCollection('CodeDataConfiguration').drop()**

## 启动

./mongod --dbpath=/data/db/ --rest

## 增加

db.getCollection('CsZlpzSwsxZb').insert({

"\_id" : ObjectId("591eac0b25f81819f8cb5531"),

"swsxmc" : "中国税收居民身份证明申请表",

"swsxDm" : "SXA051005001",

"flzlDm" : "001382",

"\_class" : "com.foresee.dzswj.zlpz.api.vo.CsZlpzSwsxZb"

});

## 删除

db.getCollection('CsZlpzSwsxZb').remove({"\_id" : ObjectId("591eac0b25f81819f8cb5531")})

## 查询语句

查询整表

db.getCollection('CodeDataConfiguration').find({})

根据登记序号查询

db.getCollection('ysqxxZcbwVO').find({"djxh" : "10126101001036920970"});

根据登记序号和业务编码查询

db.getCollection('ysqxxZcbwVO').find({"$and" : [ { "djxh" : "10126101001036920970"},{"ywbm":"SZYSSB"}]});

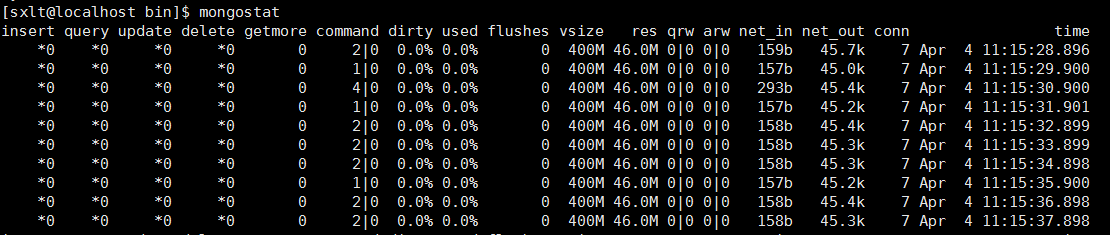
## 创建索引

db.getCollection('ysqxxVO').ensureIndex({"ysqxxSb.yzpzxh":1});

db.getCollection('ysqxxVO').getIndex();

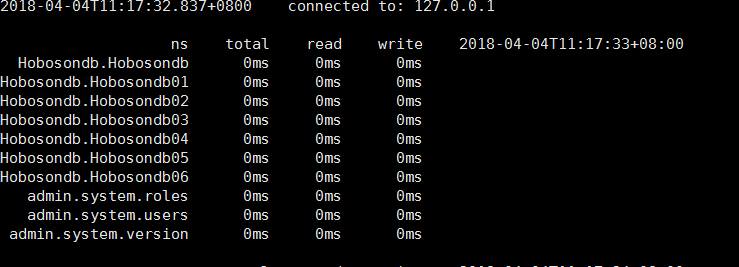
## 查看状态

mongostat



## 读写时间查看

Mongotop



## 数据库锁的使用情况

mongotop --locks

## 排序

db.getCollection('zlpzSxVO').find().sort({lrrq:-1})