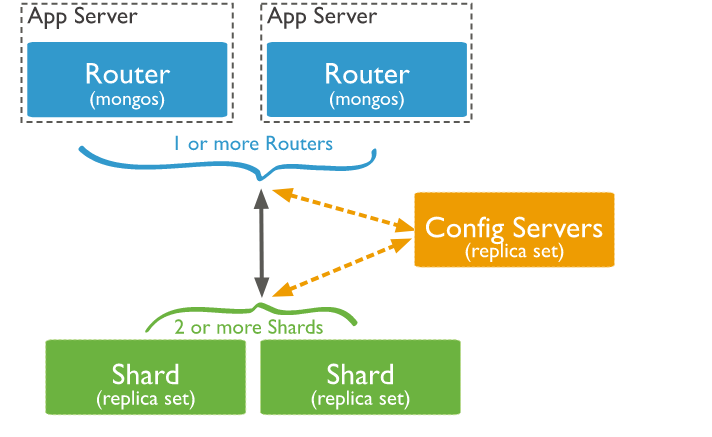
# 3.4.4 版本配置副本集+分片

## 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

## 三：服务器规划

| **服务器110** | **服务器120** | **服务130** |
| --- | --- | --- |
| 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 |

## 四：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

pidfilepath = /software/mongodb/config/log/configsrv.pid

dbpath = /software/mongodb/config/data

logpath = /software/mongodb/config/log/configsrv.log

logappend = true

bind\_ip = 0.0.0.0

port = 20000

fork = true

configsvr = true

replSet=configs

maxConns=20000

启动三台服务器的config server

Mongod -f /software/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 : "92.12.76.13:20000" },{\_id : 1, host : "92.12.76.14:20000" },{\_id : 2, host : "92.12.76.15:20000" }]};

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 /software/mongodb/conf/shard1.conf

pidfilepath = /software/mongodb/shard1/log/shard1.pid

dbpath = /software/mongodb/shard1/data

logpath = /software/mongodb/shard1/log/shard1.log

logappend = true

bind\_ip = 0.0.0.0

port = 27001

fork = true

httpinterface=true

rest=true

replSet=shard1

shardsvr = true

maxConns=20000

启动

mongod -f /software/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 27001

use admin

config = {\_id : "shard1",members : [{\_id : 0, host : "92.12.76.13:27001" },{\_id : 1, host : "92.12.76.14:27001" },{\_id : 2, host : "92.12.76.15:27001", 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 /software/mongodb/conf/shard2.conf

pidfilepath = /software/mongodb/shard2/log/shard2.pid

dbpath = /software/mongodb/shard2/data

logpath = /software/mongodb/shard2/log/shard2.log

logappend = true

bind\_ip = 0.0.0.0

port = 27002

fork = true

httpinterface=true

rest=true

replSet=shard2

shardsvr = true

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 27002

use admin

config = {\_id : "shard2",members : [{\_id : 0, host : "92.12.76.13:27002" , arbiterOnly: true },{\_id : 1, host : "92.12.76.14:27002" },{\_id : 2, host : "92.12.76.15:27002" }]}

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 /software/mongodb/conf/shard3.conf

pidfilepath = /software/mongodb/shard3/log/shard3.pid

dbpath = /software/mongodb/shard3/data

logpath = /software/mongodb/shard3/log/shard3.log

logappend = true

bind\_ip = 0.0.0.0

port = 27003

fork = true

httpinterface=true

rest=true

replSet=shard3

shardsvr = true

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 27003

use admin

config = {\_id : "shard3",members : [{\_id : 0, host : "92.12.76.13:27003" },{\_id : 1, host : "92.12.76.14:27003", arbiterOnly: true},{\_id : 2, host : "92.12.76.15:27003" }]}

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

pidfilepath = /software/mongodb/mongos/log/mongos.pid

logpath = /software/mongodb/mongos/log/mongos.log

logappend = true

bind\_ip = 0.0.0.0

port = 27017

fork = true

configdb = configs/92.12.76.13:20000,92.12.76.14:20000,92.12.76.15:20000

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 27017

use admin

sh.addShard("shard1/92.12.76.13:27001,92.12.76.14:27001,92.12.76.15:27001")

sh.addShard("shard2/92.12.76.13:27002,92.12.76.14:27002,92.12.76.15:27002")

sh.addShard("shard3/92.12.76.13:27003,92.12.76.14:27003,92.12.76.15:27003")

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 testdb;

#插入测试数据

for (var i = 1; i <= 100000; i++)

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

#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

mongod -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>

## 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 /software/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=/software/mongodb/key/mongdb-keyfile

生成密码配置文件

openssl rand -base64 100 > /software/mongodb/key/mongdb-keyfile

chmod 600 /software/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

Mongodb导出sbzsdb库数据

Mongodump –h 10.10.10.20 –port 27017 -u admin -p 123456 -d sbzsdb -o /usr/local/src/ --authenticationDatabase admin

mongodb导入sbzs库数据

mongorestore –h 10.10.10.20 –port 27017 -u admin -p 123456 /usr/local/src/

导入数据

mongorestore /usr/local/src/