MongoDB 安装文档

MongoDB副本集搭建

环境准备

```
操作系统：CentOS7
Mongodb：mongodb3.2
机器三台：
    192.168.247.130/131/132
```

调整内核参数

更改文件数限制及程序数限制

```
# vim /etc/security/limits.conf
*                soft    nofile         655350
*                hard    nofile         655350
*                soft    noproc         655350
*                hard    noproc         655350

# CentOS7 修改
vim /etc/systemd/system.conf
    [Manager]
    DefaultLimitNOFILE=655350
    DefaultLimitNPROC=655350
```

禁用内核大内存页使用

```
vim /etc/rc.local
  echo never >  /sys/kernel/mm/transparent_hugepage/enabled
  echo never >  /sys/kernel/mm/transparent_hugepage/defrag
chmod +x /etc/rc.d/rc.local
```

安装mongodb

```
tar -xf mongodb-linux-x86_64-3.2.4.tgz
mv mongodb-linux-x86_64-3.2.4/bin /usr/local/mongodb3.2/
cd /usr/local/mongodb3.2/
mkdir data conf logs pid
ls /usr/local/mongodb3.2/
bin  conf  data  logs  pid
```

配置副本集配置文件

```
vim conf/mongo.conf
systemLog:
    quiet: false
    path: /usr/local/mongodb3.2/logs/mongod.log
    logAppend: false
    destination: file
processManagement:
    fork: true
    pidFilePath: /usr/local/mongodb3.2/pid/mongod.pid
net:
    bindIp: 192.168.247.130     # 三台依次配置，更改bindIP
    port: 27017
    maxIncomingConnections: 65536
    wireObjectCheck: true
    ipv6: false
storage:
    dbPath: /usr/local/mongodb3.2/data
    indexBuildRetry: true
    journal:
        enabled: true
```

```
    directoryPerDB: false
    engine: wiredTiger
    syncPeriodSecs: 60
#    mmapv1:
#        quota:
#            enforced: false
#            maxFilesPerDB: 8
#        smallFiles: true
#        journal:
#            commitIntervalMs: 100
    wiredTiger:
        engineConfig:
            cacheSizeGB: 8
            journalCompressor: snappy
            directoryForIndexes: false
        collectionConfig:
            blockCompressor: snappy
        indexConfig:
            prefixCompression: true
operationProfiling:
    slowOpThresholdMs: 100
    mode: off
replication:
    oplogSizeMB: 1024
    replSetName: rs1
```

启动mongod 服务

三台机器依次进行启动
/usr/local/mongodb3.2/bin/mongod -f /usr/local/mongodb3.2/conf/mongo.conf

初始化副本集

登录任意一台mongo shell:

```
mongo 192.168.247.130:27017
> use admin             # 使用admin数据库
> # 定义副本集的配置
> config = {
    "_id" : "rs1",
    "members" : [
        {
            "_id" : 0,
            "host" : "192.168.247.130:27017"
        },
        {
            "_id" : 1,
            "host" : "192.168.247.131:27017"
        },
        {
            "_id" : 2,
            "host" : "192.168.247.132:27017"
        }
```

```
    ]
}
# {_id:0,host:'192.168.247.131:27017',arbiterOnly:true}  定义了一个仲裁节点
# {_id:0,host:'192.168.247.131:27017',priority:1} 可以设置节点的权重，在选举的时候优先选取权重高的
> #初始化副本集配置
> rs.initiate(config)
```

重新登录查看副本集的状态

```
rs1:PRIMARY> rs.status()
{
    "set" : "rs1",
    "date" : ISODate("2018-03-02T15:56:41.885Z"),
    "myState" : 1,
    "term" : NumberLong(1),
    "heartbeatIntervalMillis" : NumberLong(2000),
    "members" : [
        {
            "_id" : 0,
            "name" : "192.168.247.130:27017",
            "health" : 1,
            "state" : 1,
            "stateStr" : "PRIMARY",         ## 130 主机为主，其他的两台为从
            "uptime" : 1452,
            "optime" : {
                "ts" : Timestamp(1520005992, 1),
                "t" : NumberLong(1)
            },
            "optimeDate" : ISODate("2018-03-02T15:53:12Z"),
            "electionTime" : Timestamp(1520005991, 1),
            "electionDate" : ISODate("2018-03-02T15:53:11Z"),
            "configVersion" : 1,
            "self" : true
        },
        {
            "_id" : 1,
            "name" : "192.168.247.131:27017",
            "health" : 1,
            "state" : 2,
            "stateStr" : "SECONDARY",
            "uptime" : 220,
            "optime" : {
                "ts" : Timestamp(1520005992, 1),
                "t" : NumberLong(1)
            },
            "optimeDate" : ISODate("2018-03-02T15:53:12Z"),
            "lastHeartbeat" : ISODate("2018-03-02T15:56:40.303Z"),
            "lastHeartbeatRecv" : ISODate("2018-03-02T15:56:41.800Z"),
            "pingMs" : NumberLong(2),
            "syncingTo" : "192.168.247.130:27017",
            "configVersion" : 1

        },
```

```
        {
            "_id" : 2,
            "name" : "192.168.247.132:27017",
            "health" : 1,
            "state" : 2,
            "stateStr" : "SECONDARY",
            "uptime" : 220,
            "optime" : {
                "ts" : Timestamp(1520005992, 1),
                "t" : NumberLong(1)
            },
            "optimeDate" : ISODate("2018-03-02T15:53:12Z"),
            "lastHeartbeat" : ISODate("2018-03-02T15:56:40.303Z"),
            "lastHeartbeatRecv" : ISODate("2018-03-02T15:56:41.743Z"),
            "pingMs" : NumberLong(1),
            "syncingTo" : "192.168.247.130:27017",
            "configVersion" : 1
        }
    ],
    "ok" : 1
}
```

主库插入测试数据

```
rs1:PRIMARY> use testdb;    ## 创建测试数据库
rs1:PRIMARY> for (var i = 1; i <= 100000; i++) {
... db.table1.save({id:i,"test1":"testval1"});      ## 写入测试数据
... }
rs1:PRIMARY> db.table1.find()       ## 查看写入的数据
{ "_id" : ObjectId("5a99793023de7a931e5c7c90"), "id" : 1, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c91"), "id" : 2, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c92"), "id" : 3, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c93"), "id" : 4, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c94"), "id" : 5, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c95"), "id" : 6, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c96"), "id" : 7, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c97"), "id" : 8, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c98"), "id" : 9, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c99"), "id" : 10, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c9a"), "id" : 11, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c9b"), "id" : 12, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c9c"), "id" : 13, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c9d"), "id" : 14, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c9e"), "id" : 15, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7c9f"), "id" : 16, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7ca0"), "id" : 17, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7ca1"), "id" : 18, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7ca2"), "id" : 19, "test1" : "testval1" }
{ "_id" : ObjectId("5a99793023de7a931e5c7ca3"), "id" : 20, "test1" : "testval1" }
rs1:PRIMARY> db.table1.count()      ## 查看插入的数据量
100000
```

默认的情况下，mongo 副本集的从节点是不可以进行数据的写入以及查询的。

从节点开启数据的查询的方法：

```
mongo 192.168.247.131:27017
rs1:SECONDARY> rs.slaveOk()     ## 开启数据查询
rs1:SECONDARY> db.table1.count()    ## 查看查询的数据量
100000
```

主库宕机测试

```
# 在192.168.247.130 kill 掉主库
[root@localhost mongodb3.2]# ps -ef |grep mongo
root       1932      1  8 10:32 ?        00:05:17 mongod -f conf/mongo.conf
[root@localhost mongodb3.2]# kill 1932

# 登录131 查看集群的状态
rs1:SECONDARY> rs.status()
{
    "set" : "rs1",
    "date" : ISODate("2018-03-02T16:37:01.154Z"),
    "myState" : 2,
    "term" : NumberLong(2),
    "syncingTo" : "192.168.247.132:27017",
    "heartbeatIntervalMillis" : NumberLong(2000),
    "members" : [
        {
            "_id" : 0,
            "name" : "192.168.247.130:27017",
            "health" : 0,
            "state" : 8,
            "stateStr" : "(not reachable/healthy)",
            "uptime" : 0,
            "optime" : {
                "ts" : Timestamp(0, 0),
                "t" : NumberLong(-1)
            },
            "optimeDate" : ISODate("1970-01-01T00:00:00Z"),
            "lastHeartbeat" : ISODate("2018-03-02T16:36:59.639Z"),
            "lastHeartbeatRecv" : ISODate("2018-03-02T16:34:30.245Z"),
            "pingMs" : NumberLong(1),
            "lastHeartbeatMessage" : "Connection refused",
            "configVersion" : -1
        },
        {
            "_id" : 1,
            "name" : "192.168.247.131:27017",
            "health" : 1,
            "state" : 2,
            "stateStr" : "SECONDARY",
            "uptime" : 3859,

            "optime" : {
```

```
                "ts" : Timestamp(1520008480, 1),
                "t" : NumberLong(2)
            },
            "optimeDate" : ISODate("2018-03-02T16:34:40Z"),
            "syncingTo" : "192.168.247.132:27017",
            "configVersion" : 1,
            "self" : true
        },
        {
            "_id" : 2,
            "name" : "192.168.247.132:27017",
            "health" : 1,
            "state" : 1,
            "stateStr" : "PRIMARY",     ## 此时的192.168.247.132 切换为主
            "uptime" : 2638,
            "optime" : {
                "ts" : Timestamp(1520008480, 1),
                "t" : NumberLong(2)
            },
            "optimeDate" : ISODate("2018-03-02T16:34:40Z"),
            "lastHeartbeat" : ISODate("2018-03-02T16:36:59.472Z"),
            "lastHeartbeatRecv" : ISODate("2018-03-02T16:37:00.200Z"),
            "pingMs" : NumberLong(1),
            "electionTime" : Timestamp(1520008479, 1),
            "electionDate" : ISODate("2018-03-02T16:34:39Z"),
            "configVersion" : 1
        }
    ],
    "ok" : 1
}
```

此时在130 宕机的情况下在132（现有的主上插入数据）

```
mongo 192.168.247.132:27017
rs1:PRIMARY> for (var i = 100001; i <= 100100; i++) { db.table1.save({id:i,"test1":"testval1"});
}
rs1:PRIMARY> db.table1.count()
100100
```

此时启动宕机的130 上的mongo,查看数据

```
mongo 192.168.247.130:27017
rs1:SECONDARY> rs.slaveOk()
rs1:SECONDARY> use testdb
rs1:SECONDARY> db.table1.count()
100100
```

MongDB 单实例迁移至副本集测试

```
1. 验证mongodb 去除auth 的授权后，已授权的用户是否 可以使用 （ OK ）
```

0）建立usp 库的 huohe 用户（实际线上不用）

```
use usp;
db.createUser(
    {
        user: 'huohe',
        pwd: 'huohe',
        roles: [ { role: 'readWrite', db: 'usp'} ]
        }
        )
```

导入测试数据：

```
./bin/mongorestore -u usptest -p usptest --port 7080 --authenticationDatabase usp -d usp
./usp/
```

1）更改mongodb 配置文件

```
auth=false
```

2）重启mongo

```
/usr/local/mongodb3.2/bin/mongod --dbpath=/usr/local/mongodb3.2/data/ --shutdown
/usr/local/mongodb3.2/bin/mongod -f /usr/local/mongodb3.2/conf/mongo.conf
```

3）建立admin 用户

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

2. 验证单节点加入 replset 配置，后 新的复制节点后续添加的 可行性

4）更改mongodb 配置

```
auth=true
replSet=rs1
keyFile=/usr/local/mongodb3.2/conf/pemkey
# pemkey 文件的生成
openssl rand -base64 756 > pemkey
chmod 400 pemkey
```

5）第二次重启mongo

添加replsert 配置

6）初始化副本集：

```
./bin/mongo 127.0.0.1:7080
    use admin;
    db.auth('admin', 'admin')

    config = { "_id" : "rs1", "members" : [{ "_id" : 0, "host" : "192.168.247.133:7080"
}]}

     rs.initiate(config)
```

7）添加新的mongo 节点

启动新的节点， 配置admin 和 huohe 账户

```
use admin;
db.createUser(
    {
        user: 'admin',

        pwd: 'admin',
```

```
            roles: [ { role: 'root', db: 'admin'} ]
            }
            )
    use usp;
    db.createUser(
        {
            user: 'huohe',
            pwd: 'huohe',
            roles: [ { role: 'readWrite', db: 'usp'} ]
            }
            )
```

8）重启mongo 加入 auth 和 replSet

```
    systemLog:
        quiet: false
        path: /usr/local/mongodb3.2/logs/mongod.log
        logAppend: false
        destination: file
    processManagement:
        fork: true
        pidFilePath: /usr/local/mongodb3.2/pid/mongod.pid
    net:
        bindIp: 192.168.247.131,127.0.0.1
        port: 7080
        maxIncomingConnections: 65536
        wireObjectCheck: true
        ipv6: false
    storage:
        dbPath: /usr/local/mongodb3.2/data
        indexBuildRetry: true
        journal:
            enabled: true
        directoryPerDB: false
        engine: wiredTiger
        syncPeriodSecs: 60
    #    mmapv1:
    #        quota:
    #            enforced: false
    #            maxFilesPerDB: 8
    #        smallFiles: true
    #        journal:
    #            commitIntervalMs: 100
        wiredTiger:
            engineConfig:
                cacheSizeGB: 8
                journalCompressor: snappy
                directoryForIndexes: false
            collectionConfig:
                blockCompressor: snappy
            indexConfig:
                prefixCompression: true
    operationProfiling:
        slowOpThresholdMs: 100

        mode: off
```

```
        replication:
            oplogSizeMB: 10240
            replSetName: rs1    ## 开启 名字与第一个节点一样
        security:
            keyFile: /usr/local/mongodb3.2/conf/pemkey  # 与主mongo 的pemkey 一致
            clusterAuthMode: keyFile
            authorization: enabled          ## 开启auth 认证
            javascriptEnabled:  true
    9) 老的节点添加 副本集的节点
        use admin;
        db.auth('admin', 'admin')
        rs.status()
        rs.add('192.168.247.131:7080')  # 添加新的mongo 节点

        ./bin/mongo 127.0.0.1:7080   # 重新登录老的主节点
        db.getMongo().setSlaveOk();     # 设置从库可以读取数据
    10) 新的加入的节点，查看数据是否同步
        ./bin/mongo 127.0.0.1:7080
        use usp;
        db.getMongo().setSlaveOk();
        show tables;
        db.ziru_house_install_op_his.findOne()
    11) 移除旧的节点
        新的主节点执行: rs.remove('ip:port')
        添加仲裁节点: rs.addArb('ip:port')



3. 副本集 java 连接验证，宕机 连接验证 ???
```

MongoDB 分片集群的搭建

MongoDB 分片集群的几个概念

mongos: 数据库集群请求的入口，所有的请求都通过mongos进行协调，不需要在应用程序添加一个路由选择器。

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。

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。

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。

环境准备

```
操作系统: CentOS7
mongodb: 3.6

目录规划:

    /usr/local/mongodb3.6/
```

```
                data
                    configsvr
                    shard1
                    shard2
                    shard3
                config
                    mongos.conf
                    configsvr.conf
                    shard1.conf
                    shard2.conf
                    shard3.conf
                logs
                pid
端口规划：
    mongos：20000
    configServer：21000
    shard1：27000
    shard2：27001
    shard3：27002
机器划分：
    192.168.247.130：  mongos/configServer/shard1主节点/shard2仲裁/shard3副节点
    192.168.247.131：  mongos/configServer/shard1副节点/shard2主节点/shard3仲裁
    192.168.247.132：  mongos/configServer/shard1仲裁/shard2副节点/shard3主节点
```

安装mongodb3.6

```
tar -xf mongodb-linux-x86_64-rhel70-3.6.3.tgz
mkdir /usr/local/mongodb3.6
mv mongodb-linux-x86_64-rhel70-3.6.3/bin /usr/local/mongodb3.6

vim /etc/profile
    export MONGO_HOME=/usr/local/mongodb3.6
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$ZK_HOME/bin:$CODIS_HOME/bin:$MONGO_HOME/bin:$PATH
```

建立mongodb 的数据以及配置目录

```
mkdir -p
/usr/local/mongodb3.6/{data/{configsvr,shard1,shard2,shard3},logs/{shard1,shard2,shard3},config}
touch
/usr/local/mongodb3.6/config/{mongos.conf,configsvr.conf,shard1.conf,shard2.conf,shard3.conf}
```

配置configServer配置服务器

三台都进行配置

```
vim /usr/local/mongodb3.6/config/configsvr.conf
systemLog:
    quiet: false
    path: /usr/local/mongodb3.6/logs/configsvr.log
    logAppend: false
    destination: file
```

```
processManagement:
    fork: true
    pidFilePath: /usr/local/mongodb3.6/pid/configsvr.pid
net:
    bindIp: 192.168.247.130
    port: 21000
    maxIncomingConnections: 65536
    wireObjectCheck: true
    ipv6: false
storage:
    dbPath: /usr/local/mongodb3.6/data/configsvr        ## configserver 需要存储数据（路由分片配置
信息）
    indexBuildRetry: true
    journal:
        enabled: true
    directoryPerDB: false
    engine: wiredTiger
    syncPeriodSecs: 60
    wiredTiger:
        engineConfig:
            cacheSizeGB: 8
            journalCompressor: snappy
            directoryForIndexes: false
        collectionConfig:
            blockCompressor: snappy
        indexConfig:
            prefixCompression: true
operationProfiling:
    slowOpThresholdMs: 100
    mode: off
replication:
    oplogSizeMB: 1024
    replSetName: configserver   ## 3.4 之后需要configserver 也要配置副本集
sharding:
    clusterRole: configsvr  ## 指定为configsvr 则为configserver, 指定为shardsvr 则为shard 节点
    archiveMovedChunks: true
```

启动configServer

```
三台依次启动
/usr/local/mongodb3.6/bin/mongod -f /usr/local/mongodb3.6/config/configsvr.conf
```

初始化conifgServer 的副本集

```
# mongo 192.168.247.130:21000
> use admin
> config = {
    "_id" : "configserver",
    "members" : [
        {

            "_id" : 0,
```

```
            "host" : "192.168.247.130:21000"
        },
        {
            "_id" : 1,
            "host" : "192.168.247.131:21000"
        },
        {
            "_id" : 2,
            "host" : "192.168.247.132:21000"
        }
    ]
  }
 > rs.initiate(config) # 初始化副本集
```

查看副本集的状态

```
> rs.status()
```

配置shard 节点

三台配置三个shard 分片，启动9个mongod 进程

```
vim /usr/local/mongodb3.6/config/shard1.conf
systemLog:
    quiet: false
    path: /usr/local/mongodb3.6/logs/shard1.log
    logAppend: false
    destination: file
processManagement:
    fork: true
    pidFilePath: /usr/local/mongodb3.6/pid/shard1.pid
net:
    bindIp: 192.168.247.130     ## 三台的shard1 副本集 更改bindip和端口即可，不同的分片更改分片名即可
    port: 27000
    maxIncomingConnections: 65536
    wireObjectCheck: true
    ipv6: false
storage:
    dbPath: /usr/local/mongodb3.6/data/shard1
    indexBuildRetry: true
    journal:
        enabled: true
    directoryPerDB: false
    engine: wiredTiger
    syncPeriodSecs: 60
    wiredTiger:
        engineConfig:
            cacheSizeGB: 8
            journalCompressor: snappy
            directoryForIndexes: false
        collectionConfig:
            blockCompressor: snappy
```

```
        indexConfig:
            prefixCompression: true
operationProfiling:
    slowOpThresholdMs: 100
    mode: off
replication:
    oplogSizeMB: 1024
    replSetName: shard1
sharding:
    clusterRole: shardsvr
    archiveMovedChunks: true
```

启动shard1 副本集即可

```
/usr/local/mongodb3.6/bin/mongod -f /usr/local/mongodb3.6/config/shard1.conf
/usr/local/mongodb3.6/bin/mongod -f /usr/local/mongodb3.6/config/shard2.conf
/usr/local/mongodb3.6/bin/mongod -f /usr/local/mongodb3.6/config/shard3.conf
```

初始化副本集

```
# mongo 192.168.247.130:27000
> use admin
> config = {
    "_id" : "shard1",
    "members" : [
        {
            "_id" : 0,
            "host" : "192.168.247.130:27000"
        },
        {
            "_id" : 1,
            "host" : "192.168.247.131:27000"
        },
        {
            "_id" : 2,
            "host" : "192.168.247.132:27000"
        }
    ]
  }
> rs.initiate(config)
> rs.status()
```

```
# mongo 192.168.247.131:27001
> use admin
> config = {
    "_id" : "shard2",
    "members" : [
        {
            "_id" : 0,

            "host" : "192.168.247.130:27001"
```

```
        },
        {
            "_id" : 1,
            "host" : "192.168.247.131:27001"
        },
        {
            "_id" : 2,
            "host" : "192.168.247.132:27001"
        }
    ]
  }
> rs.initiate(config)
> rs.status()
```

```
# mongo 192.168.247.132:27002
> use admin
> config = {
    "_id" : "shard3",
    "members" : [
        {
            "_id" : 0,
            "host" : "192.168.247.130:27002"
        },
        {
            "_id" : 1,
            "host" : "192.168.247.131:27002"
        },
        {
            "_id" : 2,
            "host" : "192.168.247.132:27002"
        }
    ]
  }
> rs.initiate(config)
> rs.status()
```

配置路由服务器 mongos

```
vim /usr/local/mongodb3.6/config/mongos.conf
systemLog:
    quiet: false
    path: /usr/local/mongodb3.6/logs/mongos.log
    logAppend: false
    destination: file
processManagement:
    fork: true
    pidFilePath: /usr/local/mongodb3.6/pid/mongos.pid
net:
    bindIp: 192.168.247.130
    port: 20000

    maxIncomingConnections: 65536
```

```
    wireObjectCheck: true
    ipv6: false
replication:
    localPingThresholdMs: 15  # ping时间，单位：毫秒，mongos用来判定将客户端read请求发给哪个
secondary,仅对mongos有效。
sharding:
    configDB: configserver/192.168.247.130:21000,192.168.247.131:21000,192.168.247.132:21000
# configServer 地址
```

启动mongs 路由服务器(三台一次启动)

```
/usr/local/mongodb3.6/bin/mongos -f /usr/local/mongodb3.6/config/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 192.168.247.130:20000   ## 登录任意一台有mongos 路由器的主机
# 使用admin数据库
mongos> user  admin
# 串联路由服务器与分配副本集
mongos> sh.addShard("shard1/192.168.247.130:27000,192.168.247.131:27000,192.168.247.132:27000")
{
    "shardAdded" : "shard1",
    "ok" : 1,
    "$clusterTime" : {
        "clusterTime" : Timestamp(1520051213, 10),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    },
    "operationTime" : Timestamp(1520051213, 10)
}
mongos> sh.addShard("shard2/192.168.247.130:27001,192.168.247.131:27001,192.168.247.132:27001")
mongos> sh.addShard("shard3/192.168.247.130:27002,192.168.247.131:27002,192.168.247.132:27002")
# 查看分片集群的状态
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
    "_id" : 1,
    "minCompatibleVersion" : 5,
    "currentVersion" : 6,
    "clusterId" : ObjectId("5a9a10aae732dbf0fa8ae152")
  }
  shards:
        {  "_id" : "shard1",  "host" :
"shard1/192.168.247.130:27000,192.168.247.131:27000,192.168.247.132:27000",  "state" : 1 }


        {  "_id" : "shard2",  "host" :
```

```
"shard2/192.168.247.130:27001,192.168.247.131:27001,192.168.247.132:27001",  "state" : 1 }
        {  "_id" : "shard3",  "host" :
"shard3/192.168.247.130:27002,192.168.247.131:27002,192.168.247.132:27002",  "state" : 1 }
  active mongoses:
        "3.6.3" : 1
  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:
                No recent migrations
  databases:
        {  "_id" : "config",  "primary" : "config",  "partitioned" : true }
```

创建数据库，启用分片存储

```
# 连接在mongos上，准备让指定的数据库、指定的集合分片生效。我们设置testdb的 table1 表需要分片,不是所有
mongodb 的数据库和表 都需要分片!
# mongo 192.168.247.130:20000   ## 连接任何一台的mongos

# 设置分片chunk大小
use config
db.settings.save({ "_id" : "chunksize", "value" : 1 })
设置1M是为了测试，**否则要插入大量数据才能分片**  ，测试环境需要设置小的chunksize 否则看不到分片效果
# 指定test分片生效
sh.enableSharding("test")
# 指定数据库里需要分片的集合和片键
use test
db.users.createIndex({user_id : 1})
use admin
sh.shardCollection("test.users", {user_id: 1})


# 指定testdb分片生效
mongos> db.runCommand( { enablesharding :"testdb"});
# 指定数据库里需要分片的集合和片键
mongos> db.runCommand( { shardcollection : "testdb.table1",key : {id: 1} } )

# 另一种方法
mongos> sh.enableSharding('testdb1')
mongos> sh.shardCollection('testdb1.users',{uid:1})
mongos> for(i=1;i<=200000;i++) db.users.insert({uid:i,name:'hukey',age:23})
```

插入测试数据

```
mongos> use testdb;
switched to db testdb
mongos> show dbs
admin   0.000GB
config  0.001GB
testdb  0.000GB
mongos> for (var i = 1; i <= 100000; i++) {
... db.table1.save({id:i,"test1":"testval1"});
... }
```

查看分片数据的分布

```
mongos> sh.status()
…… ……
        {  "_id" : "test",  "primary" : "shard1",  "partitioned" : true }
                test.users
                        shard key: { "user_id" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shard1  5       # 可以看到以user_id 值进行了分片
                                shard2  5
                                shard3  5
                        { "user_id" : { "$minKey" : 1 } } -->> { "user_id" : 2 } on : shard2
Timestamp(8, 1)
                        { "user_id" : 2 } -->> { "user_id" : 16914 } on : shard3 Timestamp(9, 1)
                        { "user_id" : 16914 } -->> { "user_id" : 25370 } on : shard1 Timestamp(7,
1)
                        { "user_id" : 25370 } -->> { "user_id" : 34559 } on : shard1 Timestamp(3,
3)
                        { "user_id" : 34559 } -->> { "user_id" : 43015 } on : shard2 Timestamp(4,
2)
                        { "user_id" : 43015 } -->> { "user_id" : 52535 } on : shard2 Timestamp(4,
3)
                        { "user_id" : 52535 } -->> { "user_id" : 60991 } on : shard3 Timestamp(5,
2)
                        { "user_id" : 60991 } -->> { "user_id" : 70511 } on : shard3 Timestamp(5,
3)
                        { "user_id" : 70511 } -->> { "user_id" : 78833 } on : shard1 Timestamp(6,
2)
                        { "user_id" : 78833 } -->> { "user_id" : 88487 } on : shard1 Timestamp(6,
3)
                        { "user_id" : 88487 } -->> { "user_id" : 96809 } on : shard2 Timestamp(7,
2)
                        { "user_id" : 96809 } -->> { "user_id" : 106364 } on : shard2
Timestamp(7, 3)
                        { "user_id" : 106364 } -->> { "user_id" : 114686 } on : shard3
Timestamp(8, 2)
                        { "user_id" : 114686 } -->> { "user_id" : 124064 } on : shard3
Timestamp(8, 3)
                        { "user_id" : 124064 } -->> { "user_id" : { "$maxKey" : 1 } } on : shard1
Timestamp(9, 0)
```

```
…… ……

或
mongos> db.users.stats()
        "shards" : {
                "shard1" : {
                        "ns" : "test.users",
                        "size" : 3376923,
                        "count" : 53321,
                        "avgObjSize" : 63,
                        "storageSize" : 1085440,
                        …… ……
                "shard2" : {
                        "ns" : "test.users",
                        "size" : 2265226,
                        "count" : 35855,
                        "avgObjSize" : 63,
                        "storageSize" : 741376,
                        "capped" : false,
                        …… ……
                "shard3" : {
                        "ns" : "test.users",
                        "size" : 3319706,
                        "count" : 52589,
                        "avgObjSize" : 63,
                        "storageSize" : 1069056,
                        "capped" : false,
                        …… ……
```

分片集群的启动顺序

先启动配置服务器，在启动分片，最后启动mongos

配置system unit 文件, 使用systemctl 管理

```
创建systemctl unit文件mongod-configsvr.service
[Unit]
Description=Mongodb Config Server
After=network.target
Documentation=https://docs.mongodb.org/manual

[Service]
User=mongod
Group=mongod
Environment="OPTIONS=-f /usr/local/mongodb3.6/config/configsvr.conf"
ExecStart=/usr/local/mongodb3.6/bin/mongod $OPTIONS
# ExecStartPre=/usr/bin/mkdir -p /var/run/mongodb
# ExecStartPre=/usr/bin/chown mongod:mongod /var/run/mongodb
# ExecStartPre=/usr/bin/chmod 0755 /var/run/mongodb
PermissionsStartOnly=true
PIDFile=/usr/local/mongodb3.6/pid/configsvr.pid
Type=forking
```

```
# file size
LimitFSIZE=infinity
# cpu time
LimitCPU=infinity
# virtual memory size
LimitAS=infinity
# open files
LimitNOFILE=64000
# processes/threads
LimitNPROC=64000
# locked memory
LimitMEMLOCK=infinity
# total threads (user+kernel)
TasksMax=infinity
TasksAccounting=false
# Recommended limits for for mongod as specified in
# http://docs.mongodb.org/manual/reference/ulimit/#recommended-settings

[Install]
WantedBy=multi-user.target
```