咕泡学院 JavaVIP 高级课程教案

MongoDB 数据库(第 2 版)

第二章

MongoDB 项目实战

及高级应用

关于本文档

主题	咕泡学院 Java VIP 高级课程教案--MongoDB 数据库（第二版）
主讲	Tom 老师
适用对象	咕泡学院 Java 高级 VIP 学员及 VIP 授课老师
数据库版本	MongoDB Community Server 4.0.1
客户端版本	RoboMongo 0.9.0

1、用户管理

1.1、添加用户

为 testdb 添加 tom 用户

```
use testdb

db.createUser({user:"tom",pwd:"123",roles:[{  role:"dbAdmin",db:"testd

b"}]})
```

具体角色有

read：允许用户读取指定数据库

readWrite：允许用户读写指定数据库

dbAdmin：允许用户在指定数据库中执行管理函数，如索引创建、删除，查看统计或访问 system.profile

userAdmin：允许用户向 system.users 集合写入，可以找指定数据库里创建、删除和管理用户

clusterAdmin：只在 admin 数据库中可用，赋予用户所有分片和复制集相关函数的管理权限。

readAnyDatabase：只在 admin 数据库中可用，赋予用户所有数据库的读权限

readWriteAnyDatabase：只在 admin 数据库中可用，赋予用户所有数据库的读写权限

userAdminAnyDatabase：只在 admin 数据库中可用，赋予用户所有数据库的 userAdmin 权限

dbAdminAnyDatabase：只在 admin 数据库中可用，赋予用户所有数据库的 dbAdmin 权限。

root：只在 admin 数据库中可用。超级账号，超级权限

1.2 查看所有用户

```
db.system.users.find()
```

和用户管理相关的操作基本都要在 admin 数据库下运行，要先 use admin;

如果在某个单一的数据库下，那只能对当前数据库的权限进行操作;

1.3、用户删除操作

```
db.system.users.remove({user:"tom"});
```

1.4 查看当前用户权限

```
db.runCommand({usersInfo:"tom",showPrivileges:true})
```

1.5 修改密码

```
use testdb

db.changeUserPassword("tom", "123456")
```

1.6、启用用户

```
db.auth("tom","123")
```

1.7、安全检查 --auth

非 testdb 是不能操作数据库的,启用自己的用户才能访问

非 admin 数据库的用户不能使用数据库命令，admin 数据库中的数据经过认证为管理员用户

5.2、MongoDB 高可用方案实战演

详解 RouteServer（路由服务器）、

ConfigServer（配置服务器）、

Replica Set（副本集）、

Shard（切片）、

Chunk（分块）MongDB 启动与关闭

1、命令行启动

```
$ ./mongod --fork --dbpath=/opt/mongodb/data
```

2、配置文件启动

```
$ ./mongod -f mongodb.cfg
```

mongoDB 基本配置/opt/mongodb/mongodb.cfg

```
dbpath=/opt/mongodb/data

logpath=/opt/mongodb/logs/mongodb.log

logappend=true

fork=true

bind_ip=192.168.209.128

port=27017
```

环境变量配置

```
export PATH=/opt/mongodb/bin:$PATH
```

Mongodb 的三种集群方式的搭建：Master-Slaver/Replica Set / Sharding。

5.3、MongoDB 主从搭建

这个是最简答的集群搭建，不过准确说也不能算是集群，只能说是主备。并且官方已经不推荐这种方式，所以在这里只是简单的介绍下吧，搭建方式也相对简单。

主机配置 /opt/mongodb/master-slave/master/mongodb.cfg

```
dbpath=/opt/mongodb/master-slave/master/data

logpath=/opt/mongodb/master-slave/master/logs/mongodb.log

logappend=true

fork=true

bind_ip=192.168.209.128

port=27001

master=true

source=192.168.209.128:27002
```

从机配置 /opt/mongodb/master-slave/slave/mongodb.cfg

```
dbpath=/opt/mongodb/master-slave/slave/data

logpath=/opt/mongodb/master-slave/slave/logs/mongodb.log

logappend=true

fork=true

bind_ip=192.168.209.128

port=27002

slave=true

source=192.168.209.128:27001
```

启动服务

```
cd /opt/mongodb/master-slave/master/

mongod --config mongodb.cfg #主节点

cd /opt/mongodb/master-slave/slave/

mongod --config mongodb.cfg #从节点
```

连接测试

```
#客户端连接主节点

mongo  --host 192.168.209.128 --port 27001

#客户端从节点

mongo  --host 192.168.209.128 --port 27002
```

基本上只要在主节点和备节点上分别执行这两条命令，Master-Slaver 就算搭建完成了。我没有试过主节点挂掉后备节点是否能变成主节点，不过既然已经不推荐了，大家就没必要去使用了。

5.4、MongoDB 副本集

中文翻译叫做副本集，不过我并不喜欢把英文翻译成中文，总是感觉怪怪的。其实简单来说就是集群当中包含了多份数据，保证主节点挂掉了，备节点能继续提供数据服务，提供的前提就是数据需要和主节点一致。如下图：

Mongodb(M)表示主节点，Mongodb(S)表示备节点，Mongodb(A)表示仲裁节点。主备节点存储数据，仲裁节点不存储数据。客户端同时连接主节点与备节点，不连接仲裁节点。

默认设置下，主节点提供所有增删查改服务，备节点不提供任何服务。但是可以通过设置使备节点提供查询服务，这样就可以减少主节点的压力，当客户端进行数据查询时，请求自动转到备节点上。这个设置叫做 Read Preference Modes，同时 Java 客户端提供了简单的配置方式，可以不必直接对数据库进行操作。

仲裁节点是一种特殊的节点，它本身并不存储数据，主要的作用是决定哪一个备节点在主节点挂掉之后提升为主节点，所以客户端不需要连接此节点。这里虽然只有一个备节点，但是仍然需要一个仲裁节点来提升备节点级别。我开始也不相信必须要有仲裁节点，但是自己也试过没仲裁节点的话，主节点挂了备节点还是备节点，所以咱们还是需要它的。

介绍完了集群方案，那么现在就开始搭建了。

1.建立数据文件夹

一般情况下不会把数据目录建立在 mongodb 的解压目录下，不过这里方便起见，就建在 mongodb 解压目录下吧。

```
#三个目录分别对应主，备，仲裁节点
mkdir -p /opt/mongodb/replset/master
mkdir -p /opt/mongodb/replset/slaver
mkdir -p /opt/mongodb/replset/arbiter
```

2.建立配置文件

由于配置比较多，所以我们将配置写到文件里。

vi /opt/mongodb/replset/master/mongodb.cfg

```
dbpath=/opt/mongodb/replset/master/data
logpath=/opt/mongodb/replset/master/logs/mongodb.log
logappend=true
replSet=shard002
bind_ip=192.168.209.128
port=27017
fork=true
```

vi /opt/mongodb/replset/slave/mongodb.cfg

```
dbpath=/opt/mongodb/replset/slave/data
logpath=/opt/mongodb/replset/slave/logs/mongodb.log
logappend=true
replSet=shard002
bind_ip=192.168.209.129
port=27017
fork=true
```

vi /opt/mongodb/replset/arbiter/mongodb.cfg

```
dbpath=/opt/mongodb/replset/arbiter/data

logpath=/opt/mongodb/replset/arbiter/logs/mongodb.log

logappend=true

replSet=shard002

bind_ip=192.168.209.130

port=27017

fork=true
```

参数解释：
dbpath：数据存放目录
logpath：日志存放路径
logappend：以追加的方式记录日志
replSet：replica set 的名字
bind_ip：mongodb 所绑定的 ip 地址
port：mongodb 进程所使用的端口号，默认为 27017
fork：以后台方式运行进程
3、分发到集群下的其他机器

```
#将从节点配置发送到 192.168.209.129

scp -r /opt/mongodb/replset/slave

root@192.168.209.129:/opt/mongodb/replset

#将仲裁节点配置发送到 192.168.209.130

scp -r /opt/mongodb/replset/arbiter

root@192.168.209.130:/opt/mongodb/replset
```

4.启动 mongodb
进入每个 mongodb 节点的 bin 目录下

```
#登录 192.168.209.128 启动主节点
monood -f /opt/mongodb/replset/master/mongodb.cfg
#登录 192.168.209.129 启动从节点
mongod -f /opt/mongodb/replset/slave/mongodb.cfg
#登录 192.168.209.130 启动仲裁节点
mongod -f /opt/mongodb/replset/arbiter/mongodb.cfg
```

注意配置文件的路径一定要保证正确，可以是相对路径也可以是绝对路径。

5.配置主，备，仲裁节点

可以通过客户端连接 mongodb，也可以直接在三个节点中选择一个连接 mongodb。

```
#ip 和 port 是某个节点的地址
mongo 192.168.209.128:27017
use admin
cfg={_id:"shard002",members:[{_id:0,host:'192.168.209.128:27017',pri
ority:9},{_id:1,host:'192.168.209.129:27017',priority:1},{_id:2,host
:'192.168.209.130:27017',arbiterOnly:true}]};
#使配置生效
rs.initiate(cfg)
```

注意：cfg 是相当于设置一个变量，可以是任意的名字，当然最好不要是 mongodb 的关键字，conf，config 都可以。最外层的_id 表示 replica set 的名字，members 里包含的是所有节点的地址以及优先级。优先级最高的即成为主节点，即这里的 192.168.209.128:27017。特别注意的是，对于仲裁节点，需要有个特别的配置——arbiterOnly:true。这个千万不能少了，不然主备模式就不能生效。

配置的生效时间根据不同的机器配置会有长有短，配置不错的话基本上十几秒内就能生效，有的配置需要一两分钟。如果生效了，执行 rs.status()命令会看到如下信息：

```
{
        "set" : "testrs",
```

```
        "date" : ISODate("2013-01-05T02:44:43Z"),
        "myState" : 1,
        "members" : [
                {
                        "_id" : 0,
                        "name" : "192.168.209.128:27004",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 200,
                        "optime" : Timestamp(1357285565000, 1),
                        "optimeDate"                                    :
ISODate("2017-12-22T07:46:05Z"),
                        "self" : true
                },
                {
                        "_id" : 1,
                        "name" : "192.168.209.128:27003",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 200,
```

```
                        "optime" : Timestamp(1357285565000, 1),
                        "optimeDate"                                    :
ISODate("2017-12-22T07:46:05Z"),
                        "lastHeartbeat"                                 :
ISODate("2017-12-22T02:44:42Z"),
                        "pingMs" : 0
                },
                {
                        "_id" : 2,
                        "name" : "192.168.209.128:27005",
                        "health" : 1,
                        "state" : 7,
                        "stateStr" : "ARBITER",
                        "uptime" : 200,
                        "lastHeartbeat"                                 :
ISODate("2017-12-22T02:44:42Z"),
                        "pingMs" : 0
                }
        ],
        "ok" : 1
}
```

如果配置正在生效，其中会包含如下信息：

```
"stateStr" : "STARTUP"
```

同时可以查看对应节点的日志，发现正在等待别的节点生效或者正在分配数据文件。

现在基本上已经完成了集群的所有搭建工作。至于测试工作，可以留给大家自己试试。一个是往主节点插入数据，能从备节点查到之前插入的数据（查询备节点可能会遇到某个问题，可以自己去网上查查看）。二是停掉主节点，备节点能变成主节点提供服务。三是恢复主节点，备节点也能恢复其备的角色，而不是继续充当主的角色。二和三都可以通过 rs.status()命令实时查看集群的变化。

5.5、MongoDB 数据分片

和 Replica Set 类似，都需要一个仲裁节点，但是 Sharding 还需要配置节点和路由节点。就三种集群搭建方式来说，这种是最复杂的。

配置数据节点

```
mkdir -p /opt/mongodb/shard/replset/replica1/data

mkdir -p /opt/mongodb/shard/replset/replica1/logs

mkdir -p /opt/mongodb/shard/replset/replica2/data

mkdir -p /opt/mongodb/shard/replset/replica2/logs

mkdir -p /opt/mongodb/shard/replset/replica3/data

mkdir -p /opt/mongodb/shard/replset/replica3/logs
```

```
vi /opt/mongodb/shard/replset/replica1/mongodb.cfg
```

```
dbpath=/opt/mongodb/shard/replset/replica1/data

logpath=/opt/mongodb/shard/replset/replica1/logs/mongodb.log

logappend=true

fork=true

bind_ip=192.168.209.128

port=27001

replSet=shard001

shardsvr=true
```

```
vi /opt/mongodb/shard/replset/replica2/mongodb.cfg
```

```
dbpath=/opt/mongodb/shard/replset/replica2/data

logpath=/opt/mongodb/shard/replset/replica2/logs/mongodb.log

logappend=true

fork=true

bind_ip=192.168.209.128

port=27002

replSet=shard001

shardsvr=true
```

vi /opt/mongodb/shard/replset/replica3/mongodb.cfg

```
dbpath=/opt/mongodb/shard/replset/replica3/data

logpath=/opt/mongodb/shard/replset/replica3/logs/mongodb.log

logappend=true

fork=true

bind_ip=192.168.209.128

port=27003

replSet=shard001

shardsvr=true
```

2.启动数据节点

```
mongod -f /opt/mongodb/shard/replset/replica1/mongodb.cfg

#192.168.209.128:27001

mongod -f /opt/mongodb/shard/replset/replica2/mongodb.cfg

#192.168.209.128:27002
```

```
mongod -f /opt/mongodb/shard/replset/replica3/mongodb.cfg
#192.168.209.128:27003
```

3、使数据节点集群生效

```
mongo 192.168.209.128:27001   #ip 和 port 是某个节点的地址
cfg={_id:"shard001",members:[{_id:0,host:'192.168.209.128:27001'},{_i
d:1,host:'192.168.209.128:27002'},{_id:2,host:'192.168.209.128:27003'
}]};
rs.initiate(cfg)  #使配置生效
```

4、配置 configsvr

```
mkdir -p /opt/mongodb/shard/configsvr/config1/data
mkdir -p /opt/mongodb/shard/configsvr/config1/logs
mkdir -p /opt/mongodb/shard/configsvr/config2/data
mkdir -p /opt/mongodb/shard/configsvr/config2/logs
mkdir -p /opt/mongodb/shard/configsvr/config3/data
mkdir -p /opt/mongodb/shard/configsvr/config3/logs
```

/opt/mongodb/shard/configsvr/config1/mongodb.cfg

```
dbpath=/opt/mongodb/shard/configsvr/config1/data
configsvr=true
port=28001
fork=true
logpath=/opt/mongodb/shard/configsvr/config1/logs/mongodb.log
```

```
replSet=configrs

logappend=true

bind_ip=192.168.209.128
```

/opt/mongodb/shard/configsvr/config2/mongodb.cfg

```
dbpath=/opt/mongodb/shard/configsvr/config2/data

configsvr=true

port=28002

fork=true

logpath=/opt/mongodb/shard/configsvr/config2/logs/mongodb.log

replSet=configrs

logappend=true

bind_ip=192.168.209.128
```

/opt/mongodb/shard/configsvr/config3/mongodb.cfg

```
dbpath=/opt/mongodb/shard/configsvr/config3/data

configsvr=true

port=28003

fork=true

logpath=/opt/mongodb/shard/configsvr/config3/logs/mongodb.log

replSet=configrs

logappend=true

bind_ip=192.168.209.128
```

5、启动 configsvr 节点

```
mongod       -f       /opt/mongodb/shard/configsvr/config1/mongodb.cfg
```

```
#192.168.209.128:28001
mongod       -f       /opt/mongodb/shard/configsvr/config2/mongodb.cfg
#192.168.209.128:28002
mongod       -f       /opt/mongodb/shard/configsvr/config3/mongodb.cfg
#192.168.209.128:28003
```

6、使 configsvr 节点集群生效

```
mongo 192.168.209.128:28001   #ip 和 port 是某个节点的地址
use admin  #先切换到 admin
cfg={_id:"configrs",members:[{_id:0,host:'192.168.209.128:28001'},{_i
d:1,host:'192.168.209.128:28002'},{_id:2,host:'192.168.209.128:28003'
}]};
rs.initiate(cfg)  #使配置生效
```

配置路由节点

```
mkdir -p /opt/mongodb/shard/routesvr/logs
#注意:路由节点没有 data 文件夹
```

vi /opt/mongodb/shard/routesvr/mongodb.cfg

```
configdb=configrs/192.168.209.128:28001,192.168.209.128:28002,192.168
.209.128:28003
port=30000
fork=true
logpath=/opt/mongodb/shard/routesvr/logs/mongodb.log
logappend=true
```

```
bind_ip=192.168.209.128
```

7.启动路由节点

```
./mongos          -f          /opt/mongodb/shard/routesvr/mongodb.cfg

#192.168.209.128:30000
```

这里我们没有用配置文件的方式启动，其中的参数意义大家应该都明白。一般来说一个数据节点对应一个配置节点，仲裁节点则不需要对应的配置节点。注意在启动路由节点时，要将配置节点地址写入到启动命令里。

4.配置 Replica Set

这里可能会有点奇怪为什么 Sharding 会需要配置 Replica Set。其实想想也能明白，多个节点的数据肯定是相关联的，如果不配一个 Replica Set，怎么标识是同一个集群的呢。这也是人家 mongodb 的规定，咱们还是遵守吧。配置方式和之前所说的一样，定一个 cfg，然后初始化配置。

8.配置 Sharding

```
mongo 192.168.209.128:30000   #这里必须连接路由节点

sh.addShard("shard001/192.168.209.128:27001");

sh.addShard("shard002/192.168.209.128:27017");

 #shard001、shard002 表示 replica set 的名字 当把主节点添加到 shard 以后，会

自动找到 set 里的主，备，决策节点

use testdb

sh.enableSharding("testdb")    #testdb is database name

sh.shardCollection("testdb.testcon",{"name":”hashed”})

db.collection.status()
```

第一个命令很容易理解，第二个命令是对需要进行 Sharding 的数据库进行配置，第三个命令是对需要进行 Sharding 的 Collection 进行配置，这里的 testcon 即为 Collection 的名字。另外还有个 key，这个是比较关键的东西，对于查询效率会有很大的影响。

到这里 Sharding 也已经搭建完成了，以上只是最简单的搭建方式，其中某些配置仍然使用的是默认配置。如果设置不当，会导致效率异常低下，所以建议大家多看看官方文档再进行默认配置的修改。

以上三种集群搭建方式首选 Replica Set，只有真的是大数据，Sharding 才能显现威力，毕竟备节点同步数据是需要时间的。Sharding 可以将多片数据集中到路由节点上进行一些对比，然后将数据返回给客户端，但是效率还是比较低的说。

我自己有测试过，不过具体的机器配置已经不记得了。Replica Set 的 ips 在数据达到 1400W 条时基本能达到 1000 左右，而 Sharding 在 300W 时已经下降到 500 IPS，两者的单位数据大小大概是 10kb。大家在应用的时候还是多多做下性能测试，毕竟不像 Redis 有 benchmark。

5.6、MongoDB 索引

1、索引

1.1、创建索引

```
db.books.ensureIndex{{number:1}}
```

创建索引同时指定索引的名字

```
db.books.ensureIndex({number:1},{name:"book_"})
```

1.2、索引使用需要注意的地方

1)创建索引的时候注意 1 是正序创建索引-1 是倒序创建索引

2)索引的创建在提高查询性能的同事会影响插入的性能 对于经常查询少插入的文档可以考虑用索引

3)符合索引要注意索引的先后顺序

4)每个键全建立索引不一定就能提高性能呢 索引不是万能的

5)在做排序工作的时候如果是超大数据量也可以考虑加上索引 用来提高排序的性能

1.3、唯一索引

解决文档 books 不能插入重复的数值

1.4、剔除重复值

```
#则插入相同的 name 值会报错

db.books.ensureIndex({name:-1},{unique:true})
```

如果建议唯一索引之前已经有重复数值如何处理

```
#剔除重复数值

db.books.ensureIndex({name:1},{name:"book_",unique:true,dropDups:true

})
```

1.5、后台执行创建索引

为了解决创建索引锁表的问题，在不影响查询功能，可以在后台运行

```
db.books.ensureIndex({name:1},{background:true})
```

13.6、强制查询已经建立好的索引

```
#后一个 name 为索引名，正序倒序依据建立索引的规则，否则会报错

db.books.find({name:"323book"}).hint({name:1})
```

1.7、在 shell 查看数据库已经建立的索引

```
db.system.indexes.find()

db.system.namespaces.find()
```

1.8、查询索引信息和查询状态信息

```
db.books.find({name:"123book"}).explain()
```

1.9、批量和精确删除索引

```
db.runCommand({dropIndexes : "books" , index:"name_-1"})

db.runCommand({dropIndexes : "books" , index:"*"})
```

2、二维索引

建立二维索引

```
#默认会建一个[-108,108]的范围

db.map.ensureIndex({gis:"2d"},{min:-1,max:201})
```

5.7、MongoDB 数据转存及恢复

1、导出数据(中断其他操作)

使用 mongoexport 命令行

-d 指明使用的库

-c 指明要导出的表

-o 指明要导出的文件名

-csv 指定导出的 csv 格式

-q 过滤导出

--type<json|csv|tsv>

把数据好 testdb 中的 persons 导出

```
mongoexport -d testdb -c persons -o D:/persons.json
```

导出其他主机数据库的文档

```
mongoexport --host 192.168.0.16 --port 37017
```

2、导入数据(中断其他操作)

```
mongoimport --db testdb --collections persons --file d:/persons.json
```

3、运行时备份 mongodump.exe

API: http://cn.docs.mongodb.org/manual/reference/mongodump

```
mongodump --host 127.0.0.1:27017 -d testdb -o d:/testdb
```

4、运行时恢复 mongorestore.exe

API:http://cn.docs.mongodb.org/manual/reference/mongorestore

恢复数据库

```
db.dropDatabase()

mongorestore --host 127.0.0.1:27017 -d testdb -directoryperdb

d:/testdb/testdb
```

5、mongoDB 是文件数据库这其实就可以用拷贝文件的方式进行备份

6、上锁和解锁

```
db.runCommand({fsync:1,lock:1}) #上锁

db.currentOp() #解锁
```

7、数据修复

当停电等不可逆转灾难来临的时候,由于 mongodb 的存储结构导致会产生垃圾数据,在数据恢复以后这垃圾数据依然存在,这是数据库提供一个自我修复的能力.使用起来很简单

```
db.repairDatabase()
```