部

署

方

案

**深圳市金未来信息技术有限公司**

**2018年11月**

目录

**[第1章 关于本方案](#_Toc8025_WPSOffice_Level1)** **[3](#_Toc8025_WPSOffice_Level1)**

**[第2章 环境说明](#_Toc16883_WPSOffice_Level1)** **[3](#_Toc16883_WPSOffice_Level1)**

[2.1 配置环境](#_Toc16883_WPSOffice_Level2) [3](#_Toc16883_WPSOffice_Level2)

[2.2 服务器规划](#_Toc12662_WPSOffice_Level2) [3](#_Toc12662_WPSOffice_Level2)

[2.3 前期准备](#_Toc16983_WPSOffice_Level2) [3](#_Toc16983_WPSOffice_Level2)

**[第3章 MONGODB分片介绍](#_Toc12662_WPSOffice_Level1)** **[4](#_Toc12662_WPSOffice_Level1)**

[3.1 分片的目的](#_Toc25895_WPSOffice_Level2) [4](#_Toc25895_WPSOffice_Level2)

[3.2 分片设计思路](#_Toc24906_WPSOffice_Level2) [4](#_Toc24906_WPSOffice_Level2)

[3.3 分片机制提供三种优势](#_Toc20571_WPSOffice_Level2) [4](#_Toc20571_WPSOffice_Level2)

[3.4 分片集群架构](#_Toc25902_WPSOffice_Level2) [4](#_Toc25902_WPSOffice_Level2)

**[第4章 集群中数据分布](#_Toc16983_WPSOffice_Level1)** **[5](#_Toc16983_WPSOffice_Level1)**

[4.1 Chunk是什么](#_Toc6991_WPSOffice_Level2) [5](#_Toc6991_WPSOffice_Level2)

[4.2 Chunksize的选择](#_Toc2828_WPSOffice_Level2) [5](#_Toc2828_WPSOffice_Level2)

[4.3 Chunk分裂及迁移](#_Toc30122_WPSOffice_Level2) [5](#_Toc30122_WPSOffice_Level2)

**[第5章 数据分区](#_Toc25895_WPSOffice_Level1)** **[5](#_Toc25895_WPSOffice_Level1)**

[5.1 分片键shard ked](#_Toc30655_WPSOffice_Level2) [5](#_Toc30655_WPSOffice_Level2)

[5.2 两种分片](#_Toc7518_WPSOffice_Level2) [5](#_Toc7518_WPSOffice_Level2)

[5.3 分片键选择建议](#_Toc5080_WPSOffice_Level2) [5](#_Toc5080_WPSOffice_Level2)

**[第6章 部署分片集群](#_Toc24906_WPSOffice_Level1)** **[6](#_Toc24906_WPSOffice_Level1)**

[6.1 环境准备](#_Toc31129_WPSOffice_Level2) [6](#_Toc31129_WPSOffice_Level2)

[6.2 Config server配置](#_Toc12090_WPSOffice_Level2) [6](#_Toc12090_WPSOffice_Level2)

[6.3 配置分片副本集](#_Toc21076_WPSOffice_Level2) [7](#_Toc21076_WPSOffice_Level2)

[6.4 配置路由服务器mongos](#_Toc29441_WPSOffice_Level2) [9](#_Toc29441_WPSOffice_Level2)

[6.5 启用分片](#_Toc12606_WPSOffice_Level2) [9](#_Toc12606_WPSOffice_Level2)

[6.6 测试](#_Toc6959_WPSOffice_Level2) [10](#_Toc6959_WPSOffice_Level2)

**[第7章 添加用户权限](#_Toc20571_WPSOffice_Level1)** **[11](#_Toc20571_WPSOffice_Level1)**

[7.1 添加权限](#_Toc7954_WPSOffice_Level2) [11](#_Toc7954_WPSOffice_Level2)

[7.2 KeyFile满足的条件](#_Toc32564_WPSOffice_Level2) [11](#_Toc32564_WPSOffice_Level2)

[7.3 添加configserver配置](#_Toc18592_WPSOffice_Level2) [12](#_Toc18592_WPSOffice_Level2)

[7.4 添加mongos配置](#_Toc26449_WPSOffice_Level2) [12](#_Toc26449_WPSOffice_Level2)

**[第8章 集群的扩展](#_Toc25902_WPSOffice_Level1)** **[12](#_Toc25902_WPSOffice_Level1)**

[8.1 什么情况需要扩展](#_Toc12558_WPSOffice_Level2) [12](#_Toc12558_WPSOffice_Level2)

[8.2 常用的扩展方案](#_Toc24155_WPSOffice_Level2) [13](#_Toc24155_WPSOffice_Level2)

[8.3 扩展三台机器](#_Toc788_WPSOffice_Level2) [13](#_Toc788_WPSOffice_Level2)

[添加配置文件](#_Toc27257_WPSOffice_Level2) [13](#_Toc27257_WPSOffice_Level2)

**[第9章 常遇到的一些问题](#_Toc6991_WPSOffice_Level1)** **[15](#_Toc6991_WPSOffice_Level1)**

[9.1 分片数据分布不均匀](#_Toc24907_WPSOffice_Level2) [15](#_Toc24907_WPSOffice_Level2)

[9.2 特大块的问题](#_Toc11458_WPSOffice_Level2) [16](#_Toc11458_WPSOffice_Level2)

[9.3 mongodb最大的连接数](#_Toc21959_WPSOffice_Level2) [16](#_Toc21959_WPSOffice_Level2)

[9.4 监控问题](#_Toc9966_WPSOffice_Level2) [17](#_Toc9966_WPSOffice_Level2)

# 关于本方案

### 本文档详细描述了**mongodb集群的搭建与扩容**的方案。例如：shard集群配置，config集群配置，mongos节点配置等。开发人员可通过阅读该文档快速的了解每一个配置文件，便于日后在对集群进一步的进行优化。

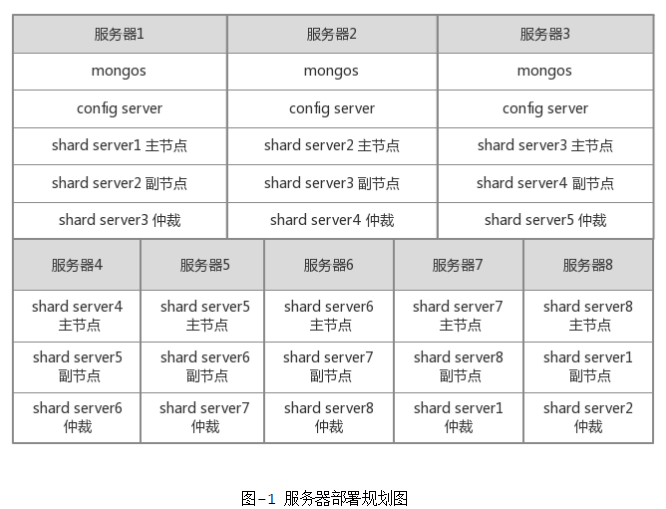
# 环境说明

## 配置环境

系统系统 ubuntu 16.04  
八台服务器：192.168.0.167-174

MongoDB：4.0.3

## 服务器规划



## 前期准备

### 在root用户下操作

# MONGODB分片介绍

## 分片的目的

高数据量和吞吐量的数据库应用会对单机的性能造成较大压力,大的查询量会将单机的CPU耗尽,大的数据量对单机的存储压力较大,最终会耗尽系统的内存而将压力转移到磁盘IO上。

## 分片设计思路

分片为应对高吞吐量与大数据量提供了方法。使用分片减少了每个分片需要处理的请求数，因此，通过水平扩展，集群可以提高自己的存储容量和吞吐量。

## 分片机制提供三种优势

### 对集群进行抽象，让集群“不可见”

MongoDB自带了一个叫做mongos的专有路由进程。mongos就是掌握统一路口的路由器，其会将客户端发来的请求准确无误的路由到集群中的一个或者一组服务器上，同时会把接收到的响应拼装起来发回到客户端。

### 保证集群总是可读写

MongoDB通过多种途径来确保集群的可用性和可靠性。将MongoDB的分片和复制功能结合使用，在确保数据分片到多台服务器的同时，也确保了每分数据都有相应的备份，这样就可以确保有服务器换掉时，其他的从库可以立即接替坏掉的部分继续工作。

### 使集群易于扩展

当系统需要更多的空间和资源的时候，MongoDB使我们可以按需方便的扩充系统容量。

## 分片集群架构



# 集群中数据分布

## Chunk是什么

在一个shard server内部，MongoDB还是会把数据分为chunks，每个chunk代表这个shard server内部一部分数据。

## Chunksize的选择

小的chunksize：数据均衡是迁移速度快，数据分布更均匀。数据分裂频繁，路由节点消耗更多资源。大的chunksize：数据分裂少。数据块移动集中消耗IO资源。通常100-200M

## Chunk分裂及迁移

随着数据的增长，其中的数据大小超过了配置的chunk size，默认是64M，则这个chunk就会分裂成两个。数据的增长会让chunk分裂得越来越多。这时候，各个shard 上的chunk数量就会不平衡。这时候，mongos中的一个组件balancer  就会执行自动平衡。把chunk从chunk数量最多的shard节点挪动到数量最少的节点。

# 数据分区

## 分片键shard ked

分片键是不可变。必须有索引。大小限制512bytes。用于路由查询。

## 两种分片

### 散列分片

散列分片在分片群集中提供更均匀的数据分布

创建库并指定分片键

use admin

#指定testdb分片生效

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} } )

测试

mongo 127.0.0.1:20000

#使用

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();

### 远程分片

基于范围的分片涉及将数据划分为由分片键值确定的连续范围。

创建库并指定分片键

use admin

#指定shop分片

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

#指定数据库里需要分片的集合和片键

sh.shardCollection('shop.user',{userid:1})

#分割数据，每一个分片上面创建40个块，按照1000，2000，3000分割

for (var i=1; i<=40;i++){

sh.splitAt('shop.user',{userid:i\*1000})

}

#插入测试数据

for (var i = 1; i <= 10000; i++){

db.user.save({userid:i,"name":"shop.users"})

};

一定要规划好块的数量，与分割的点。远程分片一般使用插入的数据差异性大，易于区分。

两种分片方式选择一种即可。

## 分片键选择建议

### 递增型

数据文件挪动小。（优势）

因为数据文件递增，所以会把insert的写IO永久放在最后一片上，造成最后一片的写热点。同时，随着最后一片的数据量增大，将不断的发生迁移至之前的片上。

### 随机型

数据分布均匀，insert的写IO均匀分布在多个片上。（优势）

大量的随机IO，磁盘不堪重荷。

### 混合型

大方向随机递增，小范围随机分布。

为了防止出现大量的chunk均衡迁移，可能造成的IO压力。我们需要设置合理分片使用策略（片键的选择、分片算法（range、hash））

# 部署分片集群

## 环境准备

创建目录

#目录和分片编号参照 图-1（下方服务器1为例）

mkdir -p /home/administrator/mongodb/conf

mkdir -p /home/administrator/mongodb/mongos/log

mkdir -p /home/administrator/mongodb/config/data

mkdir -p /home/administrator/mongodb/config/log

mkdir -p /home/administrator/mongodb/shard1/data

mkdir -p /home/administrator/mongodb/shard1/log

mkdir -p /home/administrator/mongodb/shard2/data

mkdir -p /home/administrator/mongodb/shard2/log

mkdir -p /home/administrator/mongodb/shard3/data

mkdir -p /home/administrator/mongodb/shard3/log

## Config server配置

添加配置文件

vim /home/administrator/mongodb/conf/config.conf

## 配置文件内容

pidfilepath = /home/administrator/mongodb/config/log/configsrv.pid

dbpath = /home/administrator/mongodb/config/data

logpath = /home/administrator/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 /home/administrator/mongodb/conf/config.conf

初始化配置副本集

mongo --port 21000

#config变量

config = {

\_id : "configs",

members : [

{\_id : 0, host : "192.168.0.130:21000" },

{\_id : 1, host : "192.168.0.131:21000" },

{\_id : 2, host : "192.168.0.132:21000" },

]

}

#初始化副本集

rs.initiate(config)

## 配置分片副本集

配置文件

vim /home/administrator/mongodb/conf/shard1.conf

#配置文件内容

#——————————————–

pidfilepath = /home/administrator/mongodb/shard1/log/shard1.pid

dbpath = /home/administrator/mongodb/shard1/data

logpath = /home/administrator/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

启动服务器的shard1 server

mongod -f /home/administrator/mongodb/conf/shard1.conf

登录任意一台配置服务器，初始化配置副本集

mongo --port 27001

#使用admin数据库

use admin

#定义副本集配置，第三个节点的 "arbiterOnly":true 代表其为仲裁节点。

config = {

\_id : "shard1",

members : [

{\_id : 0, host : "192.168.01.167:27001" },

{\_id : 1, host : "192.168.0.174:27001" },

{\_id : 2, host : "192.168.0.173:27001" , arbiterOnly: true},

]

}

#初始化副本集配置

rs.initiate(config);

设置其他的分片操作和上述一样，注意更改分片名称和端口

## 配置路由服务器mongos

添加配置文件

vim /home/administrator/mongodb/conf/mongos.conf

#内容

pidfilepath = /home/administrator/mongodb/mongos/log/mongos.pid

logpath = /home/administrator/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/192.168.0.167:21000,192.168.0.168:21000,192.168.0.169:21000

#设置最大连接数

maxConns=20000

启动三台服务器的mongos server

mongos -f /home/administrator/mongodb/conf/mongos.conf

## 启用分片

登录任意一台mongos

mongo --port 20000

#使用admin数据库

use admin

#串联路由服务器与分配副本集

sh.addShard("shard1/192.168.0.167:27001,192.168.0.174:27001,192.168.0.173:27001")

sh.addShard("shard2/192.168.0.168:27002,192.168.0.167:27002,192.168.0.174:27002")

sh.addShard("shard3/192.168.0.169:27003,192.168.0.168:27003,192.168.0.167:27003")

sh.addShard("shard4/192.168.0.170:27004,192.168.0.169:27004,192.168.0.168:27004")

sh.addShard("shard5/192.168.0.171:27005,192.168.0.170:27005,192.168.0.169:27005")

sh.addShard("shard6/192.168.0.172:27006,192.168.0.171:27006,192.168.0.170:27006")

sh.addShard("shard7/192.168.0.173:27007,192.168.0.172:27007,192.168.0.171:27007")

sh.addShard("shard8/192.168.0.174:27008,192.168.0.173:27008,192.168.0.172:27008")

#查看集群状态

sh.status()

## 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。

use admin

#指定testdb分片生效

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

mongo 127.0.0.1:20000

#使用

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();

# 添加用户权限

## 添加权限

MongoDB默认是没有用户权限验证的，所以我们需要手动去添加用户，以便可以更加安全的进行访问，在没添加用户验证之前，通过mongos进入数据库时会有如下的警告：

db.createUser({

user:"useradmin",

pwd:"123456",

roles: [ { role: "userAdminAnyDatabase",db:"admin"}]

})

db.auth("useradmin","123456")//认证该用户

这里就添加了一个useradmin这么一个用户，他可以进行所有数据库的用户管理。在添加这个用户后，我们连接mongodb时仍然不需要进行登录，这是因为我们未在配置中开启权限验证。

## KeyFile满足的条件

1. 在分片集群环境中，副本集内成员之间需要用keyFile认证，mongos与配置服务器，副本集之间也要keyFile认证，集群所有mongod和mongos实例使用内容相同的keyFile文件。  
   2.进行初始化，修改副本集时，都从本地例外登录进行操作  
   3.由于启用了认证，需要建立一个管理员帐号，才能从远程登录。建立管理员帐户，利用管理员账户从远程登录后，需要建立一个可以操作某个数据库的用户，客户端就用这个用户访问数据库。  
   4.分片集群中的管理员帐号需要具备配置服务器中admin和config数据库的读写权限，才能进行分片相关操作  
   5.集群中每个分片有自己的admin数据库，存储了集群的各自的证书和访问权限。

openssl rand -base64 753

利用上述命令就可以生成一个满足条件的keyFile(如下图),然后将生成的code拷入到keyFile.key中：

vim /usr/local/mongodb/key/keyFile.key

然后要给该文件限定权限，必须是600

chmod 600 /usr/local/mongodb/key/keyFile.key

## 添加configserver配置

#开启权限验证

auth=true

keyFile=/usr/local/mongodb/key/keyFile.key

## 添加mongos配置

#指向keyFile

keyFile=/usr/local/mongodb/key/keyFile.key

这样就完成了集群用户的添加

利用新组件的账号登录集群

mongo 10.203.40.114:20000/testdb -u username -p password

具体角色

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

# 集群的扩展

## 什么情况需要扩展

数据容量接近磁盘容量80%。

内存使用量接近内存容量80%。

访问量 iostat -xdk 1 ，如果 %util 接近 100%，说明产生的I/O请求太多，I/O系统已经满负荷，该磁盘可能存在瓶颈。

## 常用的扩展方案

Scale Out，横向扩展，比如：我们向原有的web、邮件系统添加一个新机器。  
Scale UP，纵向扩展， 比如：我们向原有的机器添加CPU、内存。

## 扩展三台机器

### 前期准备

要和以系统系统 ubuntu 16.04  
三台台服务器：192.168.0.135-137

MongoDB：4.0.3部署的环境保持一致

|  |  |  |
| --- | --- | --- |
| 服务器9 | 服务器10 | 服务器11 |
| Shard server9  主节点 | Shard server10  主节点 | Shard server11  主节点 |
| Shard server10  副节点 | Shard server11  副节点 | Shard server9  副节点 |
| Shard server11  仲裁 | Shard server9  仲裁 | Shard server10  仲裁 |

### 创建目录

mkdir -p /home/administrator/mongodb/conf

mkdir -p /home/administrator/mongodb/shard9/data

mkdir -p /home/administrator/mongodb/shard9/log

mkdir -p /home/administrator/mongodb/shard10/data

mkdir -p /home/administrator/mongodb/shard10/log

mkdir -p /home/administrator/mongodb/shard11/data

mkdir -p /home/administrator/mongodb/shard11/log

### 配置分片副本集

添加配置文件

vim /home/administrator/mongodb/conf/shard9.conf

#配置文件内容

#——————————————–

pidfilepath = /home/administrator/mongodb/shard9/log/shard9.pid

dbpath = /home/administrator/mongodb/shard9/data

logpath = /home/administrator/mongodb/shard9/log/shard9.log

logappend = true

bind\_ip = 0.0.0.0

port = 27009

fork = true

#打开web监控

#httpinterface=true

#rest=true

#副本集名称

replSet=shard9

#declare this is a shard db of a cluster;

shardsvr = true

#设置最大连接数

maxConns=20000

### 启动分片服务

mongod -f /home/administrator/mongodb/conf/shard9.conf

登录任意一台配置服务器，初始化配置副本集

mongo --port 27009

#使用admin数据库

use admin

#定义副本集配置，第三个节点的 "arbiterOnly":true 代表其为仲裁节点。

config = {

\_id : "shard9",

members : [

{\_id : 0, host : "192.168.0.135:27009" },

{\_id : 1, host : "192.168.0.136:27009" },

{\_id : 2, host : "192.168.0.137:27009" , arbiterOnly: true},

]

}

#初始化副本集配置

rs.initiate(config)

设置后面2个分片副本集 和上面一样

登陆任意一台mongos

mongo --port 20000

#使用admin数据库

use admin

#串联路由服务器与分配副本集

sh.addShard("shard9/192.168.0.135:27009,192.168.0.136:27009,192.168.0.137:27009")

sh.addShard("shard10/192.168.0.136:27010,192.168.0.137:27010,192.168.0.135:27010")

sh.addShard("shard11/192.168.0.137:27011,192.168.0.135:27011,192.168.0.136:27011")

#查看集群状态

sh.status()

以上完成集群的扩展，你会发现数据会自动迁移到新的分片

# 常遇到的一些问题

## 分片数据分布不均匀

1，向集群中插入数据要达到一定的大小（块默认64M）才会开始分片，当数据没有达到64M就不会向其他的分片迁移数据，从而导致分片上面的数据分布不均匀。

决解方案：

设置块的大小,官方推荐 块大小64M

db.settings.save( { \_id:"chunksize", value: <sizeInMB> } )

2，当分片中的数据达到分片的要求，就会向其他的分片迁移数据，这样就会产生大量的存储碎片，就会造成分片上面的数据显示分布不均匀，其实是一部分存储碎片暂时占用了空间导致。

解决方案：

设置分片的大小

use config

db.shards.update( { \_id : "shard1" }, { $set : { maxSize : 1024 } } )

3，当各分片服务器时间不一致的时候，会导致数据分片和插入的效率，从而导致数据分布不均匀。

解决方案：

同步服务器的时间

sudo ntpdate stdtime.sinica.edu.tw

4，每个分片中的块的大小也会影响数据分布不均匀。块设置太大会导致数据分片不均匀。

决解方案：

设置块的大小

db.settings.save( { \_id:"chunksize", value: <sizeInMB> } )

## 特大块的问题

当某些块中的数据量特别大, 形成特大块的时候, Balancer就无法对数据块进行拆分, 也就无法对块进行挪动。

解决方案：

1. 首先关闭Balancer

sh.stopBalancer()

2. 查询特大块

use config

db.chunks.find({jumbo:true})

3. 拆分特大块

sh.splitAt("db.collection", {shardkye:"拆分的临界值"})

4. 手动挪动块(可选步骤)

sh.moveChunk("db.collection", {shardkey:"shardkey所在的块"}, "需要移动的目标分片ID");

5. 重启Balancer

sh.startBalancer()

## mongodb最大的连接数

查看

ulimit -n

# vim /etc/sysctl.conf, 加入以下内容，重启生效

fs.file-max=12000

net.nf\_conntrack\_max = 12000

net.netfilter.nf\_conntrack\_max = 12000

2.修改ulimit的open file，系统默认的ulimit对文件打开数量的限制是1024

# vim /etc/security/limits.conf //加入以下配置，重启即可生效

\* hard nofile 12000

\* soft nofile 12000

## 监控问题

决解方案：

1，设置分片的大小，要小于机器内存的大小。使用mongodb自带的监控函数，mongostat , mongotop,查看监控的信息。当监控的数据达到设置分片大小的百分之80，就可以扩增机器，或者布署一个新的集群。

2，也可以使用第三方的监控（zabbix、nagios、MMS）监控集群,利用WEB界面来获取各分片的信息。

## 解决负载均衡问题

当迁移的数据庞大时，会造成服务器压力，IO请求增大。

解决方案：

可以将数据迁移在数据库读写操作少的时间进行减少IO

设置定时迁移时间：00:00~6:00

use config

db.settings.update({ \_id : "balancer" }, { $set : { activeWindow : { start : "00:00", stop: "6:00"}}})