第16章 Redis集群-最新版5.x

1.哨兵的不足

1.主库读写压力太大，从库不能分担

2.资源利用率不高　　内存取决于主库，跟从库无关

3.连接过程繁琐，效率低

4.扩容不方便

2.集群的重要概念

1.Redis集群，无论有几个节点，一共只有16384个槽位

2.所有的槽都必须被正确分配，哪怕有1个槽不正常，整个集群都不可用

3.每个节点的槽的顺序不重要，重要的是槽的数量

4.HASH算法足够平均，足够随机

5.每个槽被分配到数据的概率是大致相当的

6.集群的高可用依赖于主从复制

7.集群节点之间槽位的数量允许在2%的误差范围内

8.集群通讯会使用基础端口号+10000的端口，自动创建的，不是配置文件配置的，生产要注意的是防火墙注意要放开此端口

3.目录规划

主节点　6380

从节点　6381

集群手动安装部署

4.db-51操作

#1.发送SSH认证，方便后面传输

```
ssh-copy-id 10.0.0.52
ssh-copy-id 10.0.0.53
```

#2.创建目录

```
pkill redis
mkdir -p /opt/redis_{6380,6381}/{conf,logs,pid}
mkdir -p /data/redis_{6380,6381}
```

#3.生成主节点配置文件

```
cat >/opt/redis_6380/conf/redis_6380.conf<<EOF
bind 10.0.0.51
port 6380
daemonize yes
pidfile "/opt/redis_6380/pid/redis_6380.pid"
logfile "/opt/redis_6380/logs/redis_6380.log"
dbfilename "redis_6380.rdb"
dir "/data/redis_6380/"
appendonly yes
 appendfilename "redis.aof"
 appendfsync everysec
 cluster-enabled yes
 cluster-config-file nodes_6380.conf
 cluster-node-timeout 15000
 EOF
```

#4.复制主节点的配置文件到从节点并更改端口号

```
cd /opt/
cp redis_6380/conf/redis_6380.conf redis_6381/conf/redis_6381.co
nf
sed -i 's#6380#6381#g' redis_6381/conf/redis_6381.conf
```

#5.更改授权为redis

```
chown -R redis:redis /opt/redis_*
chown -R redis:redis /data/redis_*
```

#6.生成主节点的systemd启动文件

```
cat >/usr/lib/systemd/system/redis-master.service<<EOF
[Unit]
Description=Redis persistent key-value database
After=network.target
After=network-online.target
Wants=network-online.target

[Service]
```

```
ExecStart=/usr/local/bin/redis-server /opt/redis_6380/conf/redis_6380.conf --supervised systemd
ExecStop=/usr/local/bin/redis-cli -h $(ifconfig eth0|awk 'NR==2{print $2}') -p 6380 shutdown
Type=notify
User=redis
Group=redis
RuntimeDirectory=redis
RuntimeDirectoryMode=0755

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

#7.复制master节点的启动文件给slave节点并修改端口号

```
cd /usr/lib/systemd/system/
cp redis-master.service redis-slave.service
sed -i 's#6380#6381#g' redis-slave.service
```

#8.重载并启动集群节点

```
systemctl daemon-reload
systemctl start redis-master
systemctl start redis-slave
[root@db-51 /usr/lib/systemd/system]# ps -ef|grep redis
redis 1895 1 0 10:04 ? 00:00:00 /usr/local/bin/redis-server 10.0.0.51:6380 [cluster]
redis 1905 1 0 10:04 ? 00:00:00 /usr/local/bin/redis-server 10.0.0.51:6381 [cluster]
root 1950 1444 0 10:11 pts/0 00:00:00 grep --color=auto redis
```

#9.把创建好的目录和启动文件发送给db02和db03

```
scp -r /opt/redis_638* 10.0.0.52:/opt/
scp -r /opt/redis_638* 10.0.0.53:/opt/
scp -r /usr/lib/systemd/system/redis-*.service 10.0.0.52:/usr/lib/systemd/system/
```

```
scp -r /usr/lib/systemd/system/redis-*.service 10.0.0.53:/usr/lib/systemd/system/
```

5.db-52操作

```
pkill redis
find /opt/redis_638* -type f -name "*.conf"|xargs sed -i "/bind/s#51#52#g"
cd /usr/lib/systemd/system/
sed -i 's#51#52#g' redis-*.service
mkdir -p /data/redis_{6380,6381}
chown -R redis:redis /opt/redis_*
chown -R redis:redis /data/redis_*
systemctl daemon-reload
systemctl start redis-master
 systemctl start redis-slave
[root@db-52 /usr/lib/systemd/system]# ps -ef|grep redis
redis 1827 1 0 10:05 ? 00:00:00 /usr/local/bin/redis-server 10.0.0.52:6380 [cluster]
redis 1837 1 0 10:05 ? 00:00:00 /usr/local/bin/redis-server 10.0.0.52:6381 [cluster]
root 1867 1450 0 10:10 pts/0 00:00:00 grep --color=auto redis
```

6.db-53操作

```
pkill redis
find /opt/redis_638* -type f -name "*.conf"|xargs sed -i "/bind/s#51#53#g"
cd /usr/lib/systemd/system/
sed -i 's#51#53#g' redis-*.service
mkdir -p /data/redis_{6380,6381}
chown -R redis:redis /opt/redis_*
chown -R redis:redis /data/redis_*
systemctl daemon-reload
systemctl start redis-master
 systemctl start redis-slave
[root@db-53 /usr/lib/systemd/system]# ps -ef|grep redis
```

```
redis 1827 1 0 10:05 ? 00:00:00 /usr/local/bin/redis-server 10.0.0.53:6380 [cluster]
redis 1837 1 0 10:05 ? 00:00:00 /usr/local/bin/redis-server 10.0.0.53:6381 [cluster]
root 1867 1450 0 10:10 pts/0 00:00:00 grep --color=auto redis
```

7.集群手动发现节点

```
redis-cli -h 10.0.0.51 -p 6380 CLUSTER MEET 10.0.0.52 6380
redis-cli -h 10.0.0.51 -p 6380 CLUSTER MEET 10.0.0.53 6380
redis-cli -h 10.0.0.51 -p 6380 CLUSTER MEET 10.0.0.51 6381
redis-cli -h 10.0.0.51 -p 6380 CLUSTER MEET 10.0.0.52 6381
redis-cli -h 10.0.0.51 -p 6380 CLUSTER MEET 10.0.0.53 6381
[root@db-51 /usr/lib/systemd/system]# redis-cli -h 10.0.0.51 -p 6380 CLUSTER NODESf94ba5f443351335939c47d38ba52bd7a61810c6 10.0.0.51:6381@16381 master - 0 1625538093000 0 connected
9ba96389fad931c700f1d4ce51d73c317ef7f1fa 10.0.0.52:6380@16380 master - 0 1625538095733 2 connected
998c6c46e278f6abfb379b1a3e7a641aa151a270 10.0.0.53:6380@16380 master - 0 1625538096761 4 connected
b8eaf9bcae99f0d14e6bf77ae8e6fd520e2028e0 10.0.0.53:6381@16381 master - 0 1625538095000 5 connected
5ed7cf99f8862d7383cfeedd7d8048d4f61c6e38 10.0.0.51:6380@16380 myself,master - 0 1625538092000 1 connected
e280c1b0ef811f67aa9c8675c0f3b0a29de084a9 10.0.0.52:6381@16381 master - 0 1625538094000 3 connected
```

8.集群手动分配槽位

#1.槽位规划

```
db01:6380  5461      0-5460
db02:6380  5461   5461-10921
db03:6380  5462   10922-16383
```

#2.分配槽位

```
[root@db-51 /usr/lib/systemd/system]# redis-cli -h 10.0.0.51 -p 6380 CLUSTER ADDSLOTS {0..5460}
OK
```

```
[root@db-51 /usr/lib/systemd/system]# redis-cli -h 10.0.0.52 -p 6380 CLUSTER ADDSLOTS {5461..10921}
OK
[root@db-51 /usr/lib/systemd/system]# redis-cli -h 10.0.0.53 -p 6380 CLUSTER ADDSLOTS {10922..16383}
OK
```

#3.查看集群状态

```
[root@db-51 /usr/lib/systemd/system]# redis-cli -h 10.0.0.51 -p 6380 CLUSTER NODESf94ba5f443351335939c47d38ba52bd7a61810c6 10.0.0.51:6381@16381 master - 0 1625538194000 0 connected
9ba96389fad931c700f1d4ce51d73c317ef7f1fa 10.0.0.52:6380@16380 master - 0 1625538197967 2 connected 5461-10921
998c6c46e278f6abfb379b1a3e7a641aa151a270 10.0.0.53:6380@16380 master - 0 1625538195922 4 connected 10922-16383
b8eaf9bcae99f0d14e6bf77ae8e6fd520e2028e0 10.0.0.53:6381@16381 master - 0 1625538197000 5 connected
5ed7cf99f8862d7383cfeedd7d8048d4f61c6e38 10.0.0.51:6380@16380 myself,master - 0 1625538195000 1 connected 0-5460
e280c1b0ef811f67aa9c8675c0f3b0a29de084a9 10.0.0.52:6381@16381 master - 0 1625538196946 3 connected

[root@db-51 /usr/lib/systemd/system]# redis-cli -h 10.0.0.51 -p 6380 CLUSTER INFO
cluster_state:ok
cluster_slots_assigned:16384
cluster_slots_ok:16384
cluster_slots_pfail:0
cluster_slots_fail:0
cluster_known_nodes:6
cluster_size:3
cluster_current_epoch:5
cluster_my_epoch:1
cluster_stats_messages_ping_sent:219
cluster_stats_messages_pong_sent:236
cluster_stats_messages_meet_sent:6
cluster_stats_messages_sent:461
cluster_stats_messages_ping_received:235
```

```
cluster_stats_messages_pong_received:225
cluster_stats_messages_meet_received:1
cluster_stats_messages_received:461
```

9.手动分配复制关系

1.先获取集群节点信息

```
redis-cli -h 10.0.0.52 -p 6381 CLUSTER nodes
```

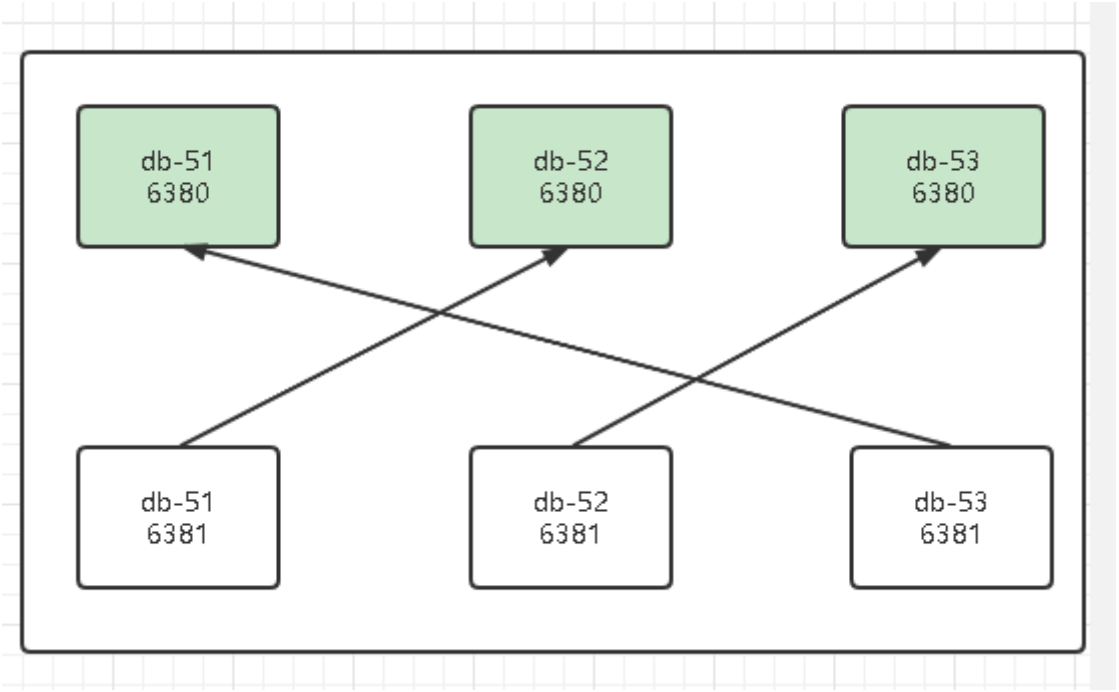
2.过滤删除不必要的信息

6380的ID 10.0.0.51

6380的ID 10.0.0.53

6380的ID 10.0.0.52

3.画图



4.配置复制关系

```
redis-cli -h 10.0.0.51 -p 6381 CLUSTER REPLICATE db02的6380的ID
redis-cli -h 10.0.0.52 -p 6381 CLUSTER REPLICATE db03的6380的ID
redis-cli -h 10.0.0.53 -p 6381 CLUSTER REPLICATE db01的6380的ID
```

6381的id

```
f94ba5f443351335939c47d38ba52bd7a61810c6 10.0.0.51:6381@16381 master - 0 1625538188000 0 connected
```

```
b8eaf9bcae99f0d14e6bf77ae8e6fd520e2028e0 10.0.0.53:6381@16381 master - 0 1625538188000 5 connected
e280c1b0ef811f67aa9c8675c0f3b0a29de084a9 10.0.0.52:6381@16381 master - 0 1625538189000 3 connected
```

6380的id

```
5ed7cf99f8862d7383cfeedd7d8048d4f61c6e38 10.0.0.51:6380@16380 myself,master - 0 1625538185000 1 connected 0-5460
9ba96389fad931c700f1d4ce51d73c317ef7f1fa 10.0.0.52:6380@16380 master - 0 1625538189793 2 connected 5461-10921
998c6c46e278f6abfb379b1a3e7a641aa151a270 10.0.0.53:6380@16380 master - 0 1625538187751 4 connected 10922-16383
```

配置复制关系

```
[root@db-51 /data/redis_6380]# redis-cli -h 10.0.0.51 -p 6381 CLUSTER REPLICATE 9ba96389fad931c700f1d4ce51d73c317ef7f1fa
OK
[root@db-51 /data/redis_6380]# redis-cli -h 10.0.0.52 -p 6381 CLUSTER REPLICATE 998c6c46e278f6abfb379b1a3e7a641aa151a270
OK
[root@db-51 /data/redis_6380]# redis-cli -h 10.0.0.53 -p 6381 CLUSTER REPLICATE 5ed7cf99f8862d7383cfeedd7d8048d4f61c6e38
OK
```

5.检查复制关系

```
[root@db-51 /data/redis_6380]# redis-cli -h 10.0.0.51 -p 6380 CLUSTER NODES
f94ba5f443351335939c47d38ba52bd7a61810c6 10.0.0.51:6381@16381 slave 9ba96389fad931c700f1d4ce51d73c317ef7f1fa 0 1625540196924 2 connected
9ba96389fad931c700f1d4ce51d73c317ef7f1fa 10.0.0.52:6380@16380 master - 0 1625540194875 2 connected 5461-10921
998c6c46e278f6abfb379b1a3e7a641aa151a270 10.0.0.53:6380@16380 master - 0 1625540195000 4 connected 10922-16383
b8eaf9bcae99f0d14e6bf77ae8e6fd520e2028e0 10.0.0.53:6381@16381 slave 5ed7cf99f8862d7383cfeedd7d8048d4f61c6e38 0 1625540197948 5 connected
5ed7cf99f8862d7383cfeedd7d8048d4f61c6e38 10.0.0.51:6380@16380 myself,master - 0 1625540197000 1 connected 0-5460
```

```
e280c1b0ef811f67aa9c8675c0f3b0a29de084a9 10.0.0.52:6381@16381 sl
ave 998c6c46e278f6abfb379b1a3e7a641aa151a270 0 1625540196000 4 cor
nected
```

第17章 集群写入数据

1.尝试插入一条数据

```
[root@db-51 /data/redis_6380]# redis-cli -h 10.0.0.51 -p 6380
10.0.0.51:6380> set k5 v5
(error) MOVED 12582 10.0.0.53:6380
10.0.0.51:6380> set k5 v5
(error) MOVED 12582 10.0.0.53:6380
10.0.0.51:6380> set k4 v4
(error) MOVED 8455 10.0.0.52:6380
10.0.0.51:6380>
```

加上-c参数

```
[root@db-51 /data/redis_6380]# redis-cli -c -h 10.0.0.51 -p 6380
10.0.0.51:6380> set k5 v5
-> Redirected to slot [12582] located at 10.0.0.53:6380
OK
10.0.0.53:6380> get k5
"v5"
```

2.目前的现象

1.在db01的6380上插入数据提示错误

2.报错提示应该移动到db03的6380上

3.根据提示在db03的6380上执行相同的命令可以写入成功

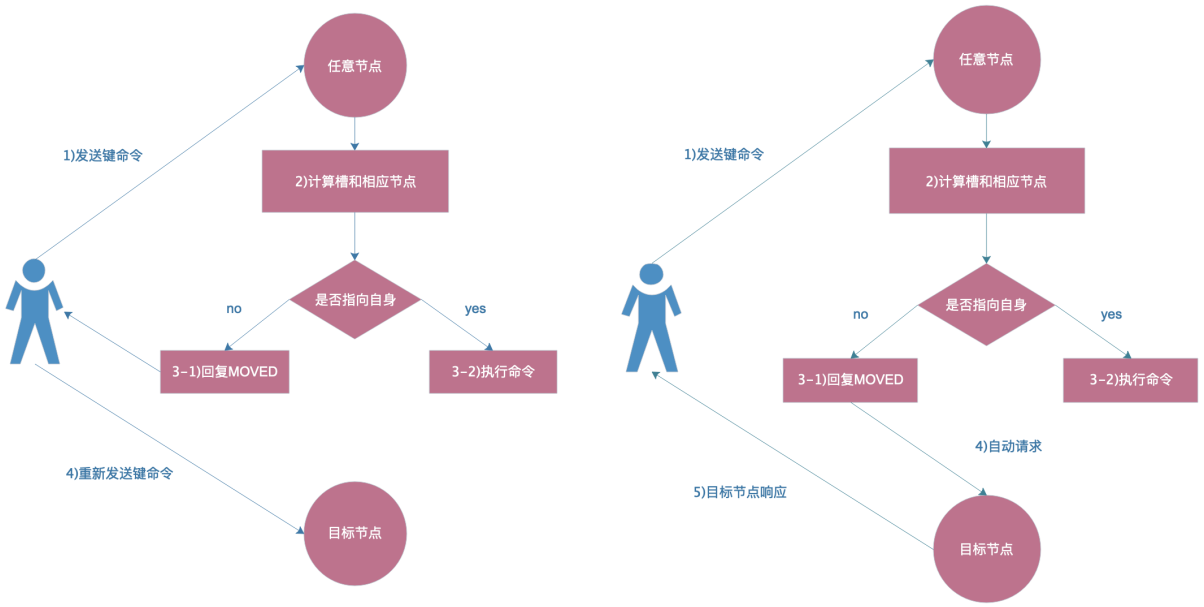
4.db01的6380有的数据可以写入，有的不行

5.使用-c参数后，可以正常写入命令，并且由目标节点返回信息

3.问题原因

因为集群模式有ASK规则，加入-c参数后，会自动跳转到目标节点处理并由目标节点返回信息。

4.ASK路由流程图



第18章 验证集群hash算法是否足够随机足够平均

1.写入测试命令-北京67期小神童-张宇提供命令

```
for i in {1..1000};do redis-cli -c -h 10.0.0.51 -p 6380 set k_${i} v_${i} && echo "${i} is ok";done
```

2.验证足够平均

```
[root@db-51 /data/redis_6380]# redis-cli -c -h 10.0.0.51 -p 6380 dbsize
(integer) 339
[root@db-51 /data/redis_6380]# redis-cli -c -h 10.0.0.52 -p 6380 dbsize
(integer) 326
[root@db-51 /data/redis_6380]# redis-cli -c -h 10.0.0.53 -p 6380 dbsize
(integer) 336
```

3.验证足够随机

```
redis-cli -c -h 10.0.0.51 -p 6380 keys \* > keys.txt
cat keys.txt |awk -F "_" '{print $2}'|sort -rn
```

4.允许节点的槽个数误差在2%以内的依据

```
[root@db01 ~]# redis-cli --cluster rebalance 10.0.0.51:6380
>>> Performing Cluster Check (using node 10.0.0.51:6380)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
*** No rebalancing needed! All nodes are within the 2.00% threshold.
```

5.检查集群健康状态

```
[root@db01 ~]# redis-cli --cluster info 10.0.0.51:6380
10.0.0.51:6380 (f765d849...) -> 3343 keys | 5461 slots | 1 slaves.
10.0.0.52:6380 (5ff2b711...) -> 3314 keys | 5461 slots | 1 slaves.
10.0.0.53:6380 (de167d13...) -> 3343 keys | 5462 slots | 1 slaves.
[OK] 10000 keys in 3 masters.
0.61 keys per slot on average.
```

```
redis-cli -c -h 10.0.0.51 -p 6380 flushall
redis-cli -c -h 10.0.0.52 -p 6380 flushall
redis-cli -c -h 10.0.0.53 -p 6380 flushall
redis-cli -c -h 10.0.0.51 -p 6380 CLUSTER reset
redis-cli -c -h 10.0.0.52 -p 6380 CLUSTER reset
redis-cli -c -h 10.0.0.53 -p 6380 CLUSTER reset
redis-cli -c -h 10.0.0.51 -p 6381 CLUSTER reset
redis-cli -c -h 10.0.0.52 -p 6381 CLUSTER reset
redis-cli -c -h 10.0.0.53 -p 6381 CLUSTER reset
redis-cli -c -h 10.0.0.51 -p 6380 CLUSTER MEET 10.0.0.52 6380
redis-cli -c -h 10.0.0.51 -p 6380 CLUSTER MEET 10.0.0.53 6380
redis-cli -h 10.0.0.51 -p 6380 CLUSTER ADDSLOTS {0..10000}
redis-cli -h 10.0.0.52 -p 6380 CLUSTER ADDSLOTS {10001..12000}
redis-cli -h 10.0.0.53 -p 6380 CLUSTER ADDSLOTS {12001..16383}
```

第19章 使用工具自动部署redis集群-通用ruby法

1.安装依赖-只要在db01上操作-北京67期-小仙女-崔娟提供命令

```
yum install -y rubygems
gem sources -a http://mirrors.aliyun.com/rubygems/
gem sources --remove http://rubygems.org/
gem install redis -v 3.3.3
```

2.还原集群环境

```
[root@db-51 /]# redis-cli -c -h 10.0.0.51 -p 6380 flushall
OK
[root@db-51 /]# redis-cli -c -h 10.0.0.52 -p 6380 flushall
OK
[root@db-51 /]# redis-cli -c -h 10.0.0.53 -p 6380 flushall
OK
[root@db-51 /]# redis-cli -h 10.0.0.51 -p 6380 CLUSTER RESET
OK
[root@db-51 /]# redis-cli -h 10.0.0.52 -p 6380 CLUSTER RESET
OK
[root@db-51 /]# redis-cli -h 10.0.0.53 -p 6380 CLUSTER RESET
OK
[root@db-51 /]# redis-cli -h 10.0.0.51 -p 6381 CLUSTER RESET
OK
[root@db-51 /]# redis-cli -h 10.0.0.52 -p 6381 CLUSTER RESET
OK
[root@db-51 /]# redis-cli -h 10.0.0.53 -p 6381 CLUSTER RESET
OK

```

3.快速部署Redis集群

```
cd /opt/redis/src/
./redis-trib.rb create --replicas 1 10.0.0.51:6380 10.0.0.52:6380 10.0.0.53:6380
```

10.0.0.51:6381 10.0.0.52:6381 10.0.0.53:6381

第20章 使用工具自动部署redis集群-高科技版

1.还原集群状态

```
redis-cli -c -h 10.0.0.51 -p 6380 flushall
redis-cli -c -h 10.0.0.52 -p 6380 flushall
redis-cli -c -h 10.0.0.53 -p 6380 flushall
redis-cli -h 10.0.0.51 -p 6380 CLUSTER RESET
redis-cli -h 10.0.0.52 -p 6380 CLUSTER RESET
redis-cli -h 10.0.0.53 -p 6380 CLUSTER RESET
redis-cli -h 10.0.0.51 -p 6381 CLUSTER RESET
redis-cli -h 10.0.0.52 -p 6381 CLUSTER RESET
redis-cli -h 10.0.0.53 -p 6381 CLUSTER RESET
```

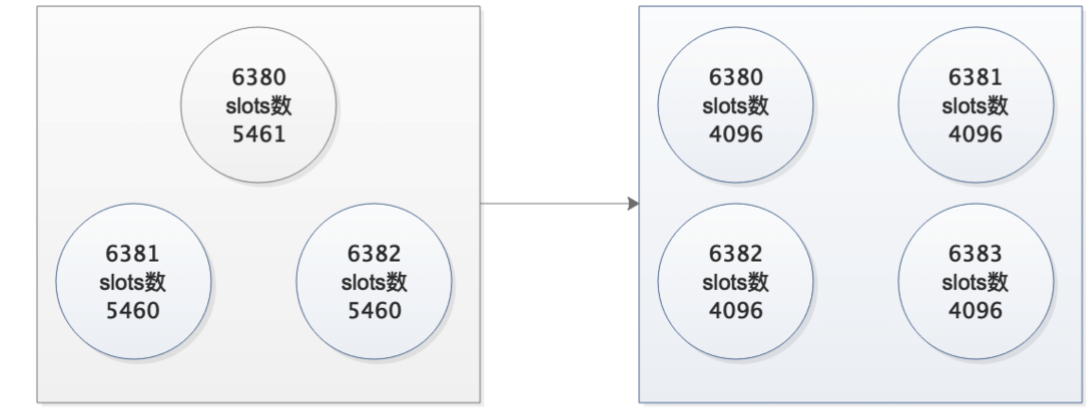
2.快速部署Redis集群

```
redis-cli --cluster create 10.0.0.51:6380 10.0.0.52:6380 10.0.0.53:6380 10.0.0.51:6381 10.0.0.52:6381 10.0.0.53:6381 --cluster-replicas 1
```

3.检查集群

```
redis-cli --cluster info 10.0.0.51:6380
redis-cli -c -h 10.0.0.51 -p 6380 cluster nodes
redis-cli -c -h 10.0.0.51 -p 6380 cluster info
```

第21章 使用工具扩容



1.需要考虑的问题

1.迁移时槽的数据会不会迁过去

2.迁移过程集群读写受影响吗

3.需要限速吗

4.如何确保迁移后的完整性

2.如何设计实验验证迁移过程是否受影响

1.迁移过程中，一个窗口读数据，一个窗口写数据

2.观察是否会中断

3.创建新节点

#1.创建节点

```
mkdir -p /opt/redis_{6390,6391}/{conf,logs,pid}
mkdir -p /data/redis_{6390,6391}
cd /opt/
cp redis_6380/conf/redis_6380.conf redis_6390/conf/redis_6390.co
nf
cp redis_6380/conf/redis_6380.conf redis_6391/conf/redis_6391.co
nf
```

```
sed -i 's#6380#6390#g' redis_6390/conf/redis_6390.conf
sed -i 's#6380#6391#g' redis_6391/conf/redis_6391.conf
redis-server /opt/redis_6390/conf/redis_6390.conf
redis-server /opt/redis_6391/conf/redis_6391.conf
ps -ef|grep redis
redis-cli -c -h 10.0.0.51 -p 6380 cluster meet 10.0.0.51 6390
redis-cli -c -h 10.0.0.51 -p 6380 cluster meet 10.0.0.51 6391
redis-cli -c -h 10.0.0.51 -p 6380 cluster nodes
```

#2.扩容步骤

#重新分配槽位

```
redis-cli --cluster reshard 10.0.0.51:6380
```

#第一次交互：每个节点最终分配多少个槽

```
How many slots do you want to move (from 1 to 16384)? 4096
```

#第二次交互：接受节点的ID

```
What is the receiving node ID? 6390的ID
```

#第三次交互：哪些节点需要导出

```
Please enter all the source node IDs.
 Type 'all' to use all the nodes as source nodes for the hash slots.
 Type 'done' once you entered all the source nodes IDs.
Source node #1:all
```

#第四次交互:确认信息

```
Do you want to proceed with the proposed reshard plan (yes/no)? yes
```

4.验证命令

写命令　db-52

```
for i in {1..1000};do redis-cli -c -h 10.0.0.51 -p 6380 set k_${i} v_${i} && echo ${i} is ok;sleep 0.5;done
```

读命令　db-53

```
for i in {1..1000};do redis-cli -c -h 10.0.0.51 -p 6380 get k_${i};sleep 0.5;done
```

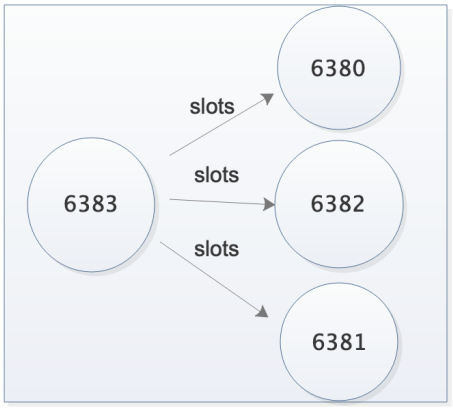
5.调整复制关系

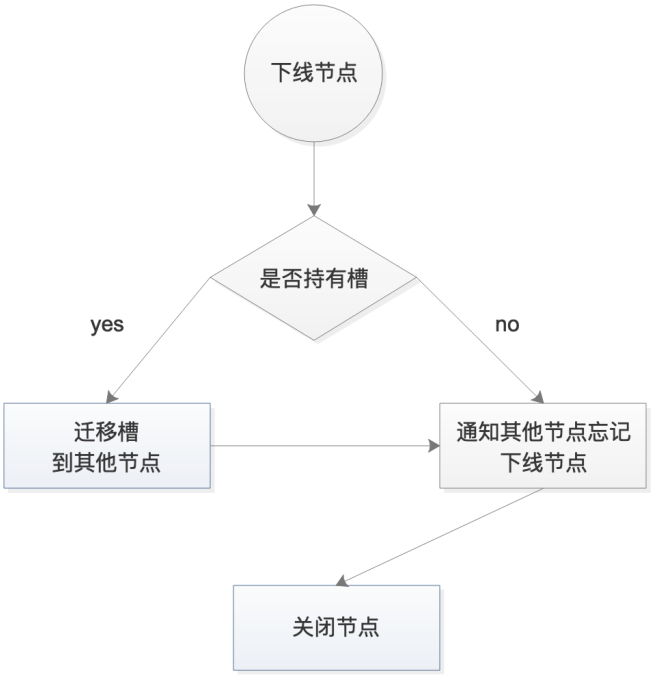
```
redis-cli -h 10.0.0.51 -p 6391 CLUSTER REPLICATE db-51的6380的ID
redis-cli -h 10.0.0.53 -p 6381 CLUSTER REPLICATE db-51的6390的ID
```

6.检查集群

```
redis-cli --cluster info 10.0.0.51:6380
```

第22章 使用工具收缩





1.缩容命令

```
#重新分配槽
redis-cli --cluster reshard 10.0.0.51:6380

#第一次交互：需要迁移多少个槽
How many slots do you want to move (from 1 to 16384)? 1365

#第三次交互：接受节点ID是多少
What is the receiving node ID? db01的6380的ID
```

```
9
10 #第三次交互：哪些节点需要导出
11 Please enter all the source node IDs.
12  Type 'all' to use all the nodes as source nodes for the hash s
lots.
13  Type 'done' once you entered all the source nodes IDs.
14 Source node #1: 6390的ID
15 Source node #2: done
16
17 #第四次交互:确认信息
18 Do you want to proceed with the proposed reshard plan (yes/no)?
yes
19
20 重复上述操作，知道6390所有的槽都被分配完毕
```

2.检查命令

```
1 redis-cli --cluster info 10.0.0.51:6380
```

```
1 redis-cli --cluster check 10.0.0.51:6380
```

3.下线节点

```
1 redis-cli --cluster del-node 10.0.0.51:6391 6391的ID
2 redis-cli --cluster del-node 10.0.0.51:6390 6390的ID
```

4.北京67期海尔兄弟-百事可乐和小神童解决方法-归一再分配法

归一

把要缩容节点的数据都扔到其中一个节点

分配

然后利用集群重新负载均衡命令重新分配

```
redis-cli --cluster rebalance 10.0.0.51:6380
```

第23章 模拟故障转移

1.关闭主节点，测试集群是否依然可用

```
1 10.0.0.51:6381> CLUSTER NODES
```

```
f765d849975ddfda7029d16be717ddffcc4c4bc7 10.0.0.51:6380@16380 sl
ave 2a55b4454e33b3c5a953264c9d69a58a56ab1a85 0 1587000834939 20 co
nnected
5ff2b711ff5b377bf06ce5ef878b3a7aaf881a98 10.0.0.52:6380@16380 sl
ave 7d1328883b4a162d2728f8719fffc53d5fb3d801 0 1587000838082 22 co
nnected
de167d131d45eedcb9b56ef0021ae110d6e55d46 10.0.0.53:6380@16380 sl
ave aef2cbf60bc3109ba76253d52d691e2dba7bd3e5 0 1587000837000 21 co
nnected
aef2cbf60bc3109ba76253d52d691e2dba7bd3e5 10.0.0.51:6381@16381 my
self,master - 0 1587000837000 21 connected 10923-16383
2a55b4454e33b3c5a953264c9d69a58a56ab1a85 10.0.0.52:6381@16381 ma
ster - 0 1587000837000 20 connected 0-5460
7d1328883b4a162d2728f8719fffc53d5fb3d801 10.0.0.53:6381@16381 ma
ster - 0 1587000837070 22 connected 5461-10922
```

2.主动发起故障转移

```
redis-cli -c -h 10.0.0.51 -p 6380 CLUSTER FAILOVER
redis-cli -c -h 10.0.0.52 -p 6380 CLUSTER FAILOVER
redis-cli -c -h 10.0.0.53 -p 6380 CLUSTER FAILOVER
```

第24章 迁移过程意外中断如何修复

1.模拟场景：迁移时人为中断，导致槽的状态不对

```
[5754->-f765d849975ddfda7029d16be717ddffcc4c4bc7]
```

2.手动修复

```
redis-cli -c -h 10.0.0.52 -p 6380 CLUSTER SETSLOT 5754 STABLE
```

3.使用工具修复-生产建议使用工具修复

```
redis-cli --cluster fix 10.0.0.51:6380
```

第25章 RedisCluster常用命令整理

1.集群(cluster)

CLUSTER INFO 打印集群的信息

CLUSTER NODES 列出集群当前已知的所有节点（node），以及这些节点的相关信息。

节点(node)

CLUSTER MEET <ip> <port> 将 ip 和 port 所指定的节点添加到集群当中，让它成为集群的一份子。

CLUSTER FORGET <node_id> 从集群中移除 node_id 指定的节点。

CLUSTER REPLICATE <node_id> 将当前节点设置为 node_id 指定的节点的从节点。

CLUSTER SAVECONFIG 将节点的配置文件保存到硬盘里面。

2.槽(slot)

CLUSTER ADDSLOTS <slot> [slot ...] 将一个或多个槽（slot）指派（assign）给当前节点。

CLUSTER DELSLOTS <slot> [slot ...] 移除一个或多个槽对当前节点的指派。

CLUSTER FLUSHSLOTS 移除指派给当前节点的所有槽，让当前节点变成一个没有指派任何槽的节点。

CLUSTER SETSLOT <slot> NODE <node_id> 将槽 slot 指派给 node_id 指定的节点，如果槽已经指派给另一个节点，那么先让另一个节点删除该槽>，然后再进行指派。

CLUSTER SETSLOT <slot> MIGRATING <node_id> 将本节点的槽 slot 迁移到 node_id 指定的节点中。

CLUSTER SETSLOT <slot> IMPORTING <node_id> 从 node_id 指定的节点中导入槽 slot 到本节点。

CLUSTER SETSLOT <slot> STABLE 取消对槽 slot 的导入（import）或者迁移（migrate）。

3.键 (key)

CLUSTER KEYSLOT <key> 计算键 key 应该被放置在哪个槽上。

CLUSTER COUNTKEYSINSLOT <slot> 返回槽 slot 目前包含的键值对数量。

CLUSTER GETKEYSINSLOT <slot> <count> 返回 count 个 slot 槽中的键。

第26章 redis自动化-黑客帝国版

1.ansible部署redis集群5.x

目录结构

```
redis_cluster/
├── files
│   ├── redis_6380
│   │   ├── conf
│   │   ├── logs
│   │   └── pid
```

```
│   ├── redis_6381
│   │   ├── conf
│   │   ├── logs
│   │   └── pid
│   └── redis_cmd
│   ├── redis-benchmark
│   ├── redis-check-aof
│   ├── redis-check-rdb
│   ├── redis-cli
│   └── redis-server
├── handlers
│   └── main.yaml
├── tasks
│   └── main.yaml
└── templates
  ├── redis_6380.conf.j2
  ├── redis_6381.conf.j2
  ├── redis-master.service.j2
  └── redis-slave.service.j2
```

调用文件

```
cat >/etc/ansible/redis_cluster.yaml <<EOF
- hosts: redis_cluster
 roles:
 - redis_cluster
EOF
```

tasks内容：

```
cat >>/etc/ansible/roles/redis_cluster/tasks/main.yaml <<EOF
#01.创建用户组
- name: 01_create_group
 group:
 name: redis
 gid: 777

#02.创建用户
- name: 02_create_user
```

```
  user:
  name: redis
  uid: 777
  group: redis
  shell: /sbin/nologin
  create_home: no

#03.拷贝执行文件
- name: 03_copy_cmd
  copy:
  src: redis_cmd/
  dest: /usr/local/bin/
  mode: '0755'

#04.拷贝运行目录
- name: 04_mkdir_conf
  copy:
  src: "{{ item }}"
  dest: /opt/
  owner: redis
  group: redis
  loop:
  - redis_6380
  - redis_6381

#05.创建数据目录
- name: 05_mkdir_data
  file:
  dest: "/data/{{ item }}"
  state: directory
  owner: redis
  group: redis
  loop:
  - redis_6380
  - redis_6381
```

```
45
46 #06.拷贝配置文件模版
47 - name: 06_copy_conf
48  template:
49  src: "{{ item.src}}"
50  dest: "{{ item.dest }}"
51  backup: yes
52  loop:
53  - { src: 'redis_6380.conf.j2', dest: '/opt/redis_6380/conf/red
is_6380.conf' }
54  - { src: 'redis_6381.conf.j2', dest: '/opt/redis_6381/conf/red
is_6381.conf' }
55  - { src: 'redis-master.service.j2', dest: '/usr/lib/systemd/sy
stem/redis-master.service' }
56  - { src: 'redis-slave.service.j2', dest: '/usr/lib/systemd/sys
tem/redis-slave.service' }
57  notify:
58  - restart redis-master
59  - restart redis-slave
60
61 #07.启动服务
62 - name: 07_start_redis
63  systemd:
64  name: "{{ item }}"
65  state: started
66  daemon_reload: yes
67  loop:
68  - redis-master
69  - redis-slave
70 EOF
```

handlers

```
1 [root@m01 ~]# cat
/etc/ansible/roles/redis_cluster/handlers/main.yaml
2 - name: restart redis-master
3  service:
4  name: redis-master
```

```
  state: restarted

- name: restart redis-slave
  service:
  name: redis-slave
   state: restarted
```

templates

```
cat >/etc/ansible/roles/redis_cluster/templates/redis_6380.conf.j2 <<EOF
bind {{ ansible_facts.eth0.ipv4.address }}
port 6380
daemonize yes
pidfile "/opt/redis_6380/pid/redis_6380.pid"
logfile "/opt/redis_6380/logs/redis_6380.log"
dbfilename "redis_6380.rdb"
dir "/data/redis_6380/"
appendonly yes
 appendfilename "redis.aof"
 appendfsync everysec
 cluster-enabled yes
 cluster-config-file nodes_6380.conf
 cluster-node-timeout 15000
 EOF

 cat >/etc/ansible/roles/redis_cluster/templates/redis-master.service.j2 <<EOF
 [Unit]
 Description=Redis persistent key-value database
 After=network.target
 After=network-online.target
 Wants=network-online.target

 [Service]
 ExecStart=/usr/local/bin/redis-server /opt/redis_6380/conf/redis_6380.conf --supervised systemd
```

```
ExecStop=/usr/local/bin/redis-cli -h {{ ansible_facts.eth0.ipv4.address }} -p 6380 shutdown
Type=notify
User=redis
Group=redis
RuntimeDirectory=redis
RuntimeDirectoryMode=0755

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

免交互维护redis集群

1.还原集群状态

```
redis-cli -c -h 10.0.0.51 -p 6380 flushall
redis-cli -c -h 10.0.0.52 -p 6380 flushall
redis-cli -c -h 10.0.0.53 -p 6380 flushall
redis-cli -h 10.0.0.51 -p 6380 CLUSTER RESET
redis-cli -h 10.0.0.52 -p 6380 CLUSTER RESET
redis-cli -h 10.0.0.53 -p 6380 CLUSTER RESET
redis-cli -h 10.0.0.51 -p 6381 CLUSTER RESET
redis-cli -h 10.0.0.52 -p 6381 CLUSTER RESET
redis-cli -h 10.0.0.53 -p 6381 CLUSTER RESET
```

2.免交互初始化集群

```
echo "yes"|redis-cli --cluster create 10.0.0.51:6380 10.0.0.52:6380 10.0.0.53:6380 10.0.0.51:6381 10.0.0.52:6381 10.0.0.53:6381 --cluster-replicas 1
```

3.免交互扩容

添加主节点　　(删除rdb文件和aof文件)

```
redis-cli --cluster add-node 10.0.0.51:6390 10.0.0.51:6380
```

添加从节点

```
redis-cli --cluster add-node 10.0.0.51:6391 10.0.0.51:6380 --cluster-slave --cluster-master-id $(redis-cli -c -h 10.0.0.51 -p 6380 cluster nodes|awk '/51:6390/{print $1}')
```

重新分配槽

```
redis-cli --cluster reshard 10.0.0.51:6380 --cluster-from all --cluster-to $(redis-cli -c -h 10.0.0.51 -p 6380 cluster nodes|awk '/51:6390/{print $1}') --cluster-slots 4096 --cluster-yes
```

4.免交互收缩

迁移槽

```
redis-cli --cluster rebalance 10.0.0.51:6380 --cluster-weight $(redis-cli -c -h 10.0.0.51 -p 6390 cluster nodes|awk '/51:6390/{print $1}')=0
```

下线节点

```
redis-cli --cluster del-node 10.0.0.51:6391 $(redis-cli -c -h 10.0.0.51 -p 6380 cluster nodes|awk '/51:6391/{print $1}')
redis-cli --cluster del-node 10.0.0.51:6390 $(redis-cli -c -h 10.0.0.51 -p 6380 cluster nodes|awk '/51:6390/{print $1}')
```

检查集群

```
redis-cli --cluster info 10.0.0.51:6380
```

1.免交互创建集群

```
echo yes|redis-cli --cluster create 10.0.0.51:6380 10.0.0.52:6380 10.0.0.53:6380 10.0.0.51:6381 10.0.0.52:6381 10.0.0.53:6381 --cluster-replicas 1
```

2.免交互扩容

添加主节点　　(删除rdb文件和aof文件)

redis-cli --cluster add-node 10.0.0.51:6390 10.0.0.51:6380

添加从节点

redis-cli --cluster add-node 10.0.0.51:6391 10.0.0.51:6380 --cluster-slave --cluster-master-id $(redis-cli -c -h 10.0.0.51 -p 6380 cluster nodes|awk '/51:6390/{print $1}')

重新分配槽

```
redis-cli --cluster reshard 10.0.0.51:6380 \
--cluster-from all \
--cluster-to $(redis-cli -h 10.0.0.51 -p 6381 CLUSTER nodes|awk '/6390/{print $1}'
) \
```

```
--cluster-slots 4096 \
--cluster-yes
```

3.免交互收缩

```
redis-cli --cluster rebalance 10.0.0.51:6380 \
--cluster-weight $(redis-cli -h 10.0.0.51 -p 6381 CLUSTER nodes|awk
'/6390/{print $1}'
)=0
```

4.下线节点

```
redis-cli --cluster del-node 10.0.0.51:6391 $(redis-cli -c -h 10.0.0.51 -p 6380
cluster nodes|awk '/51:6391/{print $1}')
redis-cli --cluster del-node 10.0.0.51:6390 $(redis-cli -c -h 10.0.0.51 -p 6380
cluster nodes|awk '/51:6390/{print $1}')
```

5.检查集群

redis-cli --cluster info 10.0.0.51:6380