Redis Cluster搭建

Redis集群安装

1. 下载Redis源码

   tar zxvf redis-X.X.X.tar.gz

2. cd redis-X.X.X
   make

3. make install
   ls /usr/local/bin/ #确认有以下文件
   redis-check-aof redis-cli redis-server
   redis-benchmark redis-check-rdb redis-sentinel

   cp src/redis-trib.rb /usr/local/bin/

   安装Ruby的依赖
   yum install ruby rubygems

   gem install redis

   确认运行redis-trib.rb 不报错

4. 源码目里utils下的脚本建议在今后的工具去阅读一下

5. 计划搭建一个Cluster结构

   ```
   /data/rcluster/
   ├── tpl.sh
   ├── start.sh
   ├── stop.sh
   ├── 7000
   │   └── 7000.conf
   ├── 7001
   │   └── 7001.conf
   ├── 7002
   │   └── 7002.conf
   ├── 7003
   │   └── 7003.conf
   ├── 7004
   │   └── 7004.conf
   └── 7005
   └── 7005.conf
   ```

   tpl.sh

   > *cat <<EOF*
   > *bind 192.168.11.101*
   > *dir /data/rcluster/$port*
   > *port $port*
   > *cluster-enabled yes*
   > *daemonize yes*
   > *cluster-config-file nodes$port.conf*

```
cluster-node-timeout 5000
appendonly yes
cluster-require-full-coverage no
logfile "./$port.log"
EOF
```

init.sh

```
#!/bin/bash
for ((i=0; i<6; ++i))
do
mkdir 700\$i
export port=700\$i
sh tpl.sh >>700$i/700$i.conf
done
```

stop.sh

```
#!/bin/bash
for ((i=0; i<6; ++i))
do
echo 700$i
/usr/local/bin/redis-cli -p 700$i shutdown
done
```

6. 初始化整个集群
   因为现在有6个节点可以建出来3组一主一从的结构，命令如下：

   redis-trib.rb create --replicas 1 192.168.11.101:7000 192.168.11.101:7001 192.168.11.101:7002 192.168.11.101:7003 192.168.11.101:7004 192.168.11.101:7005

```
>>> Creating cluster
>>> Performing hash slots allocation on 6 nodes...
Using 3 masters:
192.168.11.101:7000
192.168.11.101:7001
192.168.11.101:7002
Adding replica 192.168.11.101:7003 to 192.168.11.101:7000
Adding replica 192.168.11.101:7004 to 192.168.11.101:7001
Adding replica 192.168.11.101:7005 to 192.168.11.101:7002
M: 2ed089f797a407d1e20b9f63b39a09b53154df8c 192.168.11.101:7000
slots:0-5460 (5461 slots) master
M: 2128addbc4eaf4566df699a7e98453c0779f6233 192.168.11.101:7001
slots:5461-10922 (5462 slots) master
M: 5d1b8642594618adf310f3086bf50eb23274a6ec 192.168.11.101:7002
slots:10923-16383 (5461 slots) master
S: 6c4a627fee4194905b7c2ed7bdbad137e1ea67ad 192.168.11.101:7003
replicates 2ed089f797a407d1e20b9f63b39a09b53154df8c
S: acc936d980b59369186f3ba3a16d75c01ba7af4d 192.168.11.101:7004
replicates 2128addbc4eaf4566df699a7e98453c0779f6233
S: 5d4900c21d6d0f874951b3b5a5c4ed57d6399bff 192.168.11.101:7005
```

> *replicates 5d1b8642594618adf310f3086bf50eb23274a6ec*
> *Can I set the above configuration? (type 'yes' to accept): yes*
> *>>> Nodes configuration updated*
> *>>> Assign a different config epoch to each node*
> *>>> Sending CLUSTER MEET messages to join the cluster*
> *Waiting for the cluster to join..*
> *>>> Performing Cluster Check (using node 192.168.11.101:7000)*
> *M: **2ed089f797a407d1e20b9f63b39a09b53154df8c 192.168.11.101:7000***
> ***slots:0-5460 (5461 slots) master***
> *1 additional replica(s)*
> *M: **5d1b8642594618adf310f3086bf50eb23274a6ec 192.168.11.101:7002***
> ***slots:10923-16383 (5461 slots) master***
> *1 additional replica(s)*
> *M: **2128addbc4eaf4566df699a7e98453c0779f6233 192.168.11.101:7001***
> ***slots:5461-10922 (5462 slots) master***
> *1 additional replica(s)*
> *S: 6c4a627fee4194905b7c2ed7bdbad137e1ea67ad 192.168.11.101:7003*
> *slots: (0 slots) slave*
> *replicates 2ed089f797a407d1e20b9f63b39a09b53154df8c*
> *S: acc936d980b59369186f3ba3a16d75c01ba7af4d 192.168.11.101:7004*
> *slots: (0 slots) slave*
> *replicates 2128addbc4eaf4566df699a7e98453c0779f6233*
> *S: 5d4900c21d6d0f874951b3b5a5c4ed57d6399bff 192.168.11.101:7005*
> *slots: (0 slots) slave*
> *replicates 5d1b8642594618adf310f3086bf50eb23274a6ec*
> *[OK] All nodes agree about slots configuration.*
> *>>> Check for open slots...*
> *>>> Check slots coverage...*
> *[OK] All 16384 slots covered.*

以上输出可以看出来三组主从结构初始化，另外16384个Slot在三组机器上的分布。

到本步骤集群搭建完毕，接下来进行一些测试

集群状态查看：
redis-cli -c -h 192.168.11.101 -p 7000 cluster nodes

```
5d1b8642594618adf310f3086bf50eb23274a6ec 192.168.11.101:7002 master - 0 1499839043346 3 connected 10923-16383
2128addbc4eaf4566df699a7e98453c0779f6233 192.168.11.101:7001 master - 0 1499839043851 2 connected 5461-10922
6c4a627fee4194905b7c2ed7bdbad137e1ea67ad 192.168.11.101:7003 slave 2ed089f797a407d1e20b9f63b39a09b53154df8c 0 1499839043346 4 connected
acc936d980b59369186f3ba3a16d75c01ba7af4d 192.168.11.101:7004 slave 2128addbc4eaf4566df699a7e98453c0779f6233 0 1499839042336 5 connected
2ed089f797a407d1e20b9f63b39a09b53154df8c 192.168.11.101:7000 myself,master - 0 0 1 connected 0-5460
5d4900c21d6d0f874951b3b5a5c4ed57d6399bff 192.168.11.101:7005 slave 5d1b8642594618adf310f3086bf50eb23274a6ec 0 1499839044355 6 connected
```

1. 测试
   利用redis-cli命令需要带上 -c 参数（Enable cluster mode (follow -ASK and -MOVED redirections)

#redis-cli -h 192.168.11.101 -p 7000

```
192.168.11.101:7000> set foo bar
(error) MOVED 12182 192.168.11.101:7002
192.168.11.101:7000> get foo
(error) MOVED 12182 192.168.11.101:7002
192.168.11.101:7000> quit
```

#redis-cli -c -h 192.168.11.101 -p 7000

```
192.168.11.101:7000> set foo bar
-> Redirected to slot [12182] located at 192.168.11.101:7002
OK
192.168.11.101:7002> get foo
"bar
```

在这里可以看出来如果没-c参数连接进去，是不支持跳转，直接报错，加上 -c 标识，即可以实现集群间成员的自动跳转。

场景二：

写100个数据观查三个节点上的分布情况

for ((i=0; i<100; ++i)); do redis-cli -c -h 192.168.11.101 -p 7000 set zst$i "The Best MySQL edu at zhishutang.com :)" ; done

redis-cli -c -h 192.168.11.101 -p 7002

```
192.168.11.101:7002> get zst2
"The Best MySQL edu at zhishutang.com :)"
192.168.11.101:7002> get zst1
-> Redirected to slot [2409] located at 192.168.11.101:7000
"The Best MySQL edu at zhishutang.com :)"
192.168.11.101:7000> get zst3
-> Redirected to slot [10539] located at 192.168.11.101:7001
"The Best MySQL edu at zhishutang.com :)"
192.168.11.101:7001> get zst4
"The Best MySQL edu at zhishutang.com :)"
192.168.11.101:7001> get zst5
-> Redirected to slot [2541] located at 192.168.11.101:7000
"The Best MySQL edu at zhishutang.com :)"
```

1. 集群节点重启
   参考stop.sh 如果不行就kill吧。

2. 节点管理
   添加一个节：
   redis-trib.rb add-node ip:port 192.168.11.101:7000
   添加节点分成添加主节点和从节点，节点最好是空的。 第二个参数是集群里成员。

   如添加一个主节点到集群：
   redis-trib.rb add-node 192.168.11.101:7006 192.168.11.101:7000

```
> \>>> Adding node 192.168.11.101:7006 to cluster 192.168.11.101:7000
> \>>> Performing Cluster Check (using node 192.168.11.101:7000)
```

```
> M: 2ed089f797a407d1e20b9f63b39a09b53154df8c 192.168.11.101:7000
>    slots:0-5460 (5461 slots) master
>    1 additional replica(s)
> M: 5d1b8642594618adf310f3086bf50eb23274a6ec 192.168.11.101:7002
>    slots:10923-16383 (5461 slots) master
>    1 additional replica(s)
> M: 2128addbc4eaf4566df699a7e98453c0779f6233 192.168.11.101:7001
>    slots:5461-10922 (5462 slots) master
>    1 additional replica(s)
> S: 6c4a627fee4194905b7c2ed7bdbad137e1ea67ad 192.168.11.101:7003
>    slots: (0 slots) slave
>    replicates 2ed089f797a407d1e20b9f63b39a09b53154df8c
> S: acc936d980b59369186f3ba3a16d75c01ba7af4d 192.168.11.101:7004
>    slots: (0 slots) slave
>    replicates 2128addbc4eaf4566df699a7e98453c0779f6233
> S: 5d4900c21d6d0f874951b3b5a5c4ed57d6399bff 192.168.11.101:7005
>    slots: (0 slots) slave
>    replicates 5d1b8642594618adf310f3086bf50eb23274a6ec
> [OK] All nodes agree about slots configuration.
> \>>> Check for open slots...
> \>>> Check slots coverage...
> [OK] All 16384 slots covered.
> \>>> Send CLUSTER MEET to node 192.168.11.101:7006 to make it join the cluster.
> [OK] New node added correctly.
```

添加从节节点：
redis-trib.rb add-node --slave 192.168.11.101:7006 192.168.11.101:7007

说明： 这种情况没有指定是给那个主节点添加从，整个集群会随机的选出来一个主。新加入的节点上面没有任何数据（slot）是一个空节点.

如果需要指定给某个主节点添加从节点，可以用：master-id指定

redis-trib.rb add-node --slave --master-id 3c3a0c74aae0b56170ccb03a76b60cfe7dc1912e
192.168.11.101:7006 192.168.11.101:7007

移除节点：
redis-trib.rb del-node 192.168.11.101:700 'node-id'
如： redis-trib.rb del-node 192.168.11.101:700 '3c3a0c74aae0b56170ccb03a76b60cfe7dc1912e'

redis-trib.rb del-node 192.168.11.101:7000 '49bf78b8b0f0cb085fd36cafbcf65cf849a47caa'

```
>     >>> Removing node 49bf78b8b0f0cb085fd36cafbcf65cf849a47caa from cluster 192.168.11.101:7000
>     >>> Sending CLUSTER FORGET messages to the cluster...
>     >>> SHUTDOWN the node.
>     >>>
   需要注意： 第一个连接节点是任何一个Redis都可，后面的node-id是节点的id， 对主节点如果需要移除，需要把上在贩Slot先迁移走。
```

利用 cluster nodes 去检验

1. Slot管理
   数据重新平衡
   新加入节点后，需要做数据平衡

   redis-trib.rb reshard 192.168.11.101:7000
   Step1： 提示数据向那个节点平衡提供相应主节点的node-id
   Step2： 指定移走多少Slot
   Step3： 指定多那个master上移走，提供node-id
   Step4： 输出done

Step5 ： 确认 yes
迁移开始

提示： reshard 提供的可以把指定的节点上的Slot迁到指定节点，如果需要数据平衡，需要考虑使用rebalance参数，如：
** redis-trib.rb rebalance 192.168.11.101:7000**

```
   >>> Performing Cluster Check (using node 192.168.11.101:7000)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
>>> Rebalancing across 4 nodes. Total weight = 4
Moving 1365 slots from 192.168.11.101:7002 to 192.168.11.101:7006
############################################################################################################
############################################################################################################
############################################################################################################
############################################################################################################
############################################################################################################
############################################################################################################
############################################################################################################
############################################################################################################
############################################################################################################
############################################################################################################
############################################################################################################
############################################################################################################
############################################################################################################
#######
Moving 366 slots from 192.168.11.101:7000 to 192.168.11.101:7006
############################################################################################################
############################################################################################################
############################################################################################################
########################################################################
Moving 365 slots from 192.168.11.101:7001 to 192.168.11.101:7006
############################################################################################################
############################################################################################################
############################################################################################################
########################################################################
```

10.故障转移
可以偿试把一组机器里的一个节点关掉试试，或是整个Group里的成员全部关掉。测试一下。