Redis安装及集群搭建手册

1、 环境信息

```
centos7
redis5
```

单机安装

```
wget http://download.redis.io/releases/redis-5.0.3.tar.gz
tar xzf redis-5.0.3.tar.gz -C /usr/local/
cd /usr/local/redis-5.0.3
make
# 启动Redis
src/redis-server
# 使用客户端命令窗口
src/redis-cli
redis> set foo bar
OK
redis> get foo
"bar"
```

2、整体集群信息

```
# 以直接在一台机器上实现上述的伪集群，因为端口号特意设置为不同的。
# 重点：不论机器多少，对于部署过程都是一样的，只不过是在不同机器启动redis-server而已
192.168.100.242 (6381- 6386共6个端口)
# 注意事项：如果你的服务器有多个IP，那你操作下面步骤时，尽量使用你的客户端能够访问的IP
```

3、安装Redis

每台服务器上面都要下载安装

```
wget http://download.redis.io/releases/redis-5.0.3.tar.gz
tar -zxvf redis-5.0.3.tar.gz
cd redis-5.0.3
make
# 安装到 /usr/local/redis 目录中 安装的文件只有一个bin目录
make install PREFIX=/usr/local/redis/

# 创建配置文件和data存放目录
mkdir /usr/local/redis/conf /usr/local/redis/data
```

4、准备配置文件

准备6个redis.conf配置文件（为了方便学习，redis.conf根据不同端口来命名，方便一台机器上构建伪集群）

```
# 配置文件进行了精简，完整配置可自行和官方提供的完整conf文件进行对照。端口号自行对应修改
#后台启动的意思
daemonize yes
 #端口号
port 6381
# IP绑定，redis不建议对公网开放，直接绑定0.0.0.0没毛病
bind 0.0.0.0
# redis数据文件存放的目录
dir /usr/local/redis/data
# 开启AOF
appendonly yes
 # 开启集群
cluster-enabled yes
# 会自动生成在上面配置的dir目录下
cluster-config-file nodes-6381.conf
cluster-node-timeout 5000
# 这个文件会自动生成
pidfile /var/run/redis_6381.pid
```

5、启动6个Redis实例

```
# 一定要注意每个配置文件中的端口号哦
/usr/local/redis/bin/redis-server /usr/local/redis/conf/6381.conf
/usr/local/redis/bin/redis-server /usr/local/redis/conf/6382.conf
/usr/local/redis/bin/redis-server /usr/local/redis/conf/6383.conf
/usr/local/redis/bin/redis-server /usr/local/redis/conf/6384.conf
/usr/local/redis/bin/redis-server /usr/local/redis/conf/6385.conf
/usr/local/redis/bin/redis-server /usr/local/redis/conf/6386.conf
```

6、 创建cluster

```
# 5.0版本的方式
/usr/local/redis/bin/redis-cli --cluster create 192.168.100.242:6381
192.168.100.242:6382 \
192.168.100.242:6383 192.168.100.242:6384 192.168.100.242:6385
192.168.100.242:6386 \
--cluster-replicas 1

# 自动设置主从，而且会提示你，是否运行使用自动的配置
Can I set the above configuration? (type 'yes' to accept): yes
# 执行后的信息
>>> Performing hash slots allocation on 6 nodes...
Master[0] -> Slots 0 - 5460
Master[1] -> Slots 5461 - 10922
Master[2] -> Slots 10923 - 16383
Adding replica 192.168.100.242:6384 to 192.168.100.242:6381
Adding replica 192.168.100.242:6385 to 192.168.100.242:6382
Adding replica 192.168.100.242:6386 to 192.168.100.242:6383
>>> Trying to optimize slaves allocation for anti-affinity
[WARNING] Some slaves are in the same host as their master
M: 68326caa0238cb877afc3e6df23eb92558fcbc3c 192.168.100.242:6381
```

```
   slots:[0-5460] (5461 slots) master
M: 764500b86fadebd535ac2b5b778a73486fe7d2b7 192.168.100.242:6382
   slots:[5461-10922] (5462 slots) master
M: 93293699b8966ccc202bb29c659a9f60e26e4c86 192.168.100.242:6383
   slots:[10923-16383] (5461 slots) master
S: a842c5188c441453fd303520424132d45914fe5b 192.168.100.242:6384
   replicates 764500b86fadebd535ac2b5b778a73486fe7d2b7
S: 0a7061773b2512c91b6173bc27451b19fe02f269 192.168.100.242:6385
   replicates 93293699b8966ccc202bb29c659a9f60e26e4c86
S: 42806ad3f740f639ec321b78ea492c42a4040176 192.168.100.242:6386
   replicates 68326caa0238cb877afc3e6df23eb92558fcbc3c
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 192.168.100.242:6381)
M: 68326caa0238cb877afc3e6df23eb92558fcbc3c 192.168.100.242:6381
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
S: 0a7061773b2512c91b6173bc27451b19fe02f269 192.168.100.242:6385
   slots: (0 slots) slave
   replicates 93293699b8966ccc202bb29c659a9f60e26e4c86
S: a842c5188c441453fd303520424132d45914fe5b 192.168.100.242:6384
   slots: (0 slots) slave
   replicates 764500b86fadebd535ac2b5b778a73486fe7d2b7
M: 93293699b8966ccc202bb29c659a9f60e26e4c86 192.168.100.242:6383
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
M: 764500b86fadebd535ac2b5b778a73486fe7d2b7 192.168.100.242:6382
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
S: 42806ad3f740f639ec321b78ea492c42a4040176 192.168.100.242:6386
   slots: (0 slots) slave
   replicates 68326caa0238cb877afc3e6df23eb92558fcbc3c
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

7、 集群检验和测试

```
# 检查集群，查看所有节点信息
/usr/local/redis/bin/redis-cli -c -h 192.168.100.242 -p 6381 cluster nodes
# 执行后的信息
[root@node3 redis]# /usr/local/redis/bin/redis-cli -c -h 192.168.100.242 -p 6381
cluster nodes
0a7061773b2512c91b6173bc27451b19fe02f269 192.168.100.242:6385@16385 slave
93293699b8966ccc202bb29c659a9f60e26e4c86 0 1550635081000 5 connected
a842c5188c441453fd303520424132d45914fe5b 192.168.100.242:6384@16384 slave
764500b86fadebd535ac2b5b778a73486fe7d2b7 0 1550635081565 4 connected
93293699b8966ccc202bb29c659a9f60e26e4c86 192.168.100.242:6383@16383 master - 0
1550635081665 3 connected 10923-16383
764500b86fadebd535ac2b5b778a73486fe7d2b7 192.168.100.242:6382@16382 master - 0
1550635081000 2 connected 5461-10922
```

```
68326caa0238cb877afc3e6df23eb92558fcbc3c 192.168.100.242:6381@16381
myself,master - 0 1550635080000 1 connected 0-5460
42806ad3f740f639ec321b78ea492c42a4040176 192.168.100.242:6386@16386 slave
68326caa0238cb877afc3e6df23eb92558fcbc3c 0 1550635080664 6 connected
# 节点id ip+端口 角色 masterid 处理的ping数量 最后一个pong时间 节点配置版本 节点连接状态
slot槽分配情况

# 测试Redis Cluster的一种简单方法是使用redis-cli命令行实用程序
# -c 是支持cluster重定向
[root@node3 redis]# /usr/local/redis/bin/redis-cli -c -h 192.168.100.242 -p 6381
192.168.100.242:6381> set a 1
-> Redirected to slot [15495] located at 192.168.100.242:6383
OK
192.168.100.242:6383> get a
"1"
192.168.100.242:6383> set hello tony
-> Redirected to slot [866] located at 192.168.100.242:6381
OK
192.168.100.242:6381> get hello
"tony"
192.168.100.242:6381> get a
-> Redirected to slot [15495] located at 192.168.100.242:6383
"1"

# 查看一个key属于哪一个节点
CLUSTER KEYSLOT key
```

8、集群slot数量整理 reshard

```
#  /usr/local/redis/bin/redis-cli --cluster help 可以查看所有这个命令和子命令的帮助信
息

# 默认是master平均分了0-16383的所有虚拟slot
# 可以进行调整，部分节点放多一点slot(槽或者位置)。
/usr/local/redis/bin/redis-cli --cluster reshard  <host>:<port> --cluster-from
<node-id> --cluster-to <node-id> --cluster-slots <number of slots> --cluster-yes

# 重新检查集群
[root@node3 redis]# /usr/local/redis/bin/redis-cli --cluster check
192.168.100.242:6382
192.168.100.242:6382 (764500b8...) -> 0 keys | 5462 slots | 1 slaves.
192.168.100.242:6383 (93293699...) -> 1 keys | 5461 slots | 1 slaves.
192.168.100.242:6381 (68326caa...) -> 1 keys | 5461 slots | 1 slaves.
[OK] 2 keys in 3 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 192.168.100.242:6382)
M: 764500b86fadebd535ac2b5b778a73486fe7d2b7 192.168.100.242:6382
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
S: a842c5188c441453fd303520424132d45914fe5b 192.168.100.242:6384
   slots: (0 slots) slave
   replicates 764500b86fadebd535ac2b5b778a73486fe7d2b7
M: 93293699b8966ccc202bb29c659a9f60e26e4c86 192.168.100.242:6383
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
S: 42806ad3f740f639ec321b78ea492c42a4040176 192.168.100.242:6386
```

```
   slots: (0 slots) slave
   replicates 68326caa0238cb877afc3e6df23eb92558fcbc3c
S: 0a7061773b2512c91b6173bc27451b19fe02f269 192.168.100.242:6385
   slots: (0 slots) slave
   replicates 93293699b8966ccc202bb29c659a9f60e26e4c86
M: 68326caa0238cb877afc3e6df23eb92558fcbc3c 192.168.100.242:6381
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

9、 测试自动故障转移

```
# cluster集群不保证数据一致，数据也可能丢失
# 首先是运行客户端不断的写入或读取数据，以便能够发现问题
# 然后是模拟节点故障：找一个主节点关闭，主从故障切换的过程中，这个时间端的操作，客户端而言，只能是失败
# 官方描述 https://redis.io/topics/cluster-spec
There is always a window of time when it is possible to lose writes during partitions.
分区的时间窗口内总是有可能丢失写操作。
```

10、手动故障转移

```
# 可能某个节点需要维护（机器下线、硬件升级、系统版本调整等等场景），需要手动的实现转移
# 在slave节点上执行命令
CLUSTER FAILOVER
# 注：CLUSTER  help 可以看到帮助文档和简介。 相对安全的做法
```

11、扩容

```
# 1、 启动新节点
/usr/local/redis/bin/redis-server /usr/local/redis/conf/6387.conf

# 2、 加入到已经存在的集群作为master
/usr/local/redis/bin/redis-cli --cluster add-node 192.168.100.242:6387 192.168.100.242:6382
# 本质就是发送一个新节点通过 CLUSTER MEET命令加入集群
# 新节点没有分配hash槽

# 3、 加入到已经存在的集群作为slave
/usr/local/redis/bin/redis-cli --cluster add-node 192.168.100.242:7006 192.168.100.242:7000 --cluster-slave
# 可以手工指定master，否则就是选择一个slave数量较少的master
/usr/local/redis/bin/redis-cli --cluster add-node 192.168.100.242:7006 192.168.100.242:7000 --cluster-slave --cluster-master-id <node-id>
# 还可以将空master，转换为slave
cluster replicate <master-node-id>

# 4、 检查集群
/usr/local/redis/bin/redis-cli --cluster check 192.168.100.242:6382
```

12、缩容（删除节点）

```
# 注意：删除master的时候要把数据清空或者分配给其他主节点
/usr/local/redis/bin/redis-cli --cluster del-node 192.168.100.242:6381 <node-id>
```

13、关心的问题

```
# 1、 增加了slot槽的计算，是不是比单机性能差？
共16384个槽，slots槽计算方式公开的，java客户端中就使用了：HASH_SLOT = CRC16(key) mod
16384
为了避免每次都需要服务器计算重定向，优秀的java客户端都实现了本地计算，和服务器slots分配进行映
射，有变动时再更新本地内容。

# 2、 redis集群大小
理论是可以做到16384个槽，但是redis官方建议是最大1000个实例

# 3、 批量操作或者

# 4、cluster meet命令中的bus-port是什么？
MEET <ip> <port> [bus-port]
每个Redis群集节点都有一个额外的TCP端口，用于接收来自其他Redis群集节点的传入连接

# 5、集群节点间的通信方式
每个节点使用TCP连接与每个其他节点连接。

# 6、ask和moved重定向的区别
重定向包括两种情况
如果是确定slot不属于当前节点，redis会返回moved
如果当前redis节点正在处理slot迁移，则代表此处请求对应的key暂时不在此节点，返回ask，告诉客户端
本次请求重定向

# 7、数据倾斜和访问倾斜的问题
解决办法 调整key的策略 + slot迁移
迁移过程如下，完整的迁移流程：
在迁移目的节点执行cluster setslot <slot> IMPORTING <node ID>命令，指明需要迁移的slot和
迁移源节点。
在迁移源节点执行cluster setslot <slot> MIGRATING <node ID>命令，指明需要迁移的slot和迁
移目的节点。
在迁移源节点执行cluster getkeysinslot获取该slot的key列表。
在迁移源节点执行对每个key执行migrate命令，该命令会同步把该key迁移到目的节点。
在迁移源节点反复执行cluster getkeysinslot命令，直到该slot的列表为空。
在迁移源节点和目的节点执行cluster setslot <slot> NODE <node ID>，完成迁移操作。
```