

# Redis集群部署文档

---

## Redis集群部署文档(centos6系统)

(要让集群正常工作至少需要3个主节点，在这里我们要创建6个redis节点，其中三个为主节点，三个为从节点，对应的redis节点的ip和端口对应关系如下)

127.0.0.1:7000  
127.0.0.1:7001  
  
127.0.0.1:7002  
  
127.0.0.1:7003  
  
127.0.0.1:7004  
  
127.0.0.1:7005

1: 下载redis。官网下载3.0.0版本，之前2.几的版本不支持集群模式

下载地址: <https://github.com/antirez/redis/archive/3.0.0-rc2.tar.gz>

2: 上传服务器，解压，编译

```
tar -zxvf redis-3.0.0-rc2.tar.gz
mv redis-3.0.0-rc2.tar.gz redis3.0
cd /usr/local/redis3.0
make
make install
```

3: 创建集群需要的目录

```
mkdir -p /usr.local/cluster
cd /usr.local/cluster
mkdir 7000
mkdir 7001
mkdir 7002
mkdir 7003
mkdir 7004
mkdir 7005
```

4: 修改配置文件redis.conf

```
cp /usr/local/redis3.0/redis.conf /usr/local/cluster
vi redis.conf
##修改配置文件中的下面选项
port 7000
daemonize yes
cluster-enabled yes
cluster-config-file nodes.conf
cluster-node-timeout 5000
appendonly yes
##修改完redis.conf配置文件中的这些配置项之后把这个配置文件分别拷贝到7000/7001/7002/7003/7004/7005目录下面
cp /usr/local/cluster/redis.conf /usr/local/cluster/7000
cp /usr/local/cluster/redis.conf /usr/local/cluster/7001
cp /usr/local/cluster/redis.conf /usr/local/cluster/7002
cp /usr/local/cluster/redis.conf /usr/local/cluster/7003
cp /usr/local/cluster/redis.conf /usr/local/cluster/7004
cp /usr/local/cluster/redis.conf /usr/local/cluster/7005

##注意：拷贝完成之后要修改7001/7002/7003/7004/7005目录下面redis.conf文件中的port参数，分别改为对应的文件夹的名称
```

#### 5: 分别启动这6个redis实例

```
cd /usr/local/cluster/7000
redis-server redis.conf
cd /usr/local/cluster/7001
redis-server redis.conf
cd /usr/local/cluster/7002
redis-server redis.conf
cd /usr/local/cluster/7003
redis-server redis.conf
cd /usr/local/cluster/7004
redis-server redis.conf
cd /usr/local/cluster/7005
redis-server redis.conf

##启动之后使用命令查看redis的启动情况ps -ef|grep redis
如下图显示则说明启动成功
```

#### 6: 执行redis的创建集群命令创建集群

```
cd /usr/local/redis3.0/src
./redis-trib.rb create --replicas 1 127.0.0.1:7000 127.0.0.1:7001 127.0.0.1:7002 127.0.0.1:7003 127.0.0.1:7004 127.0.0.1:7005
```

6.1执行上面的命令的时候会报错，因为是执行的ruby的脚本，需要ruby的环境

错误内容: /usr/bin/env: ruby: No such file or directory

所以需要安装ruby的环境，这里推荐使用yum install ruby安装

```
yum install ruby
```

6.2然后再执行第6步的创建集群命令，还会报错，提示缺少rubygems组件，使用yum安装

错误内容：

```
./redis-trib.rb:24:in `require': no such file to load -- rubygems (LoadError)
```

```
from ./redis-trib.rb:24
```

```
yum install rubygems
```

6.3再次执行第6步的命令，还会报错，提示不能加载redis，是因为缺少redis和ruby的接口，使用gem 安装

错误内容：

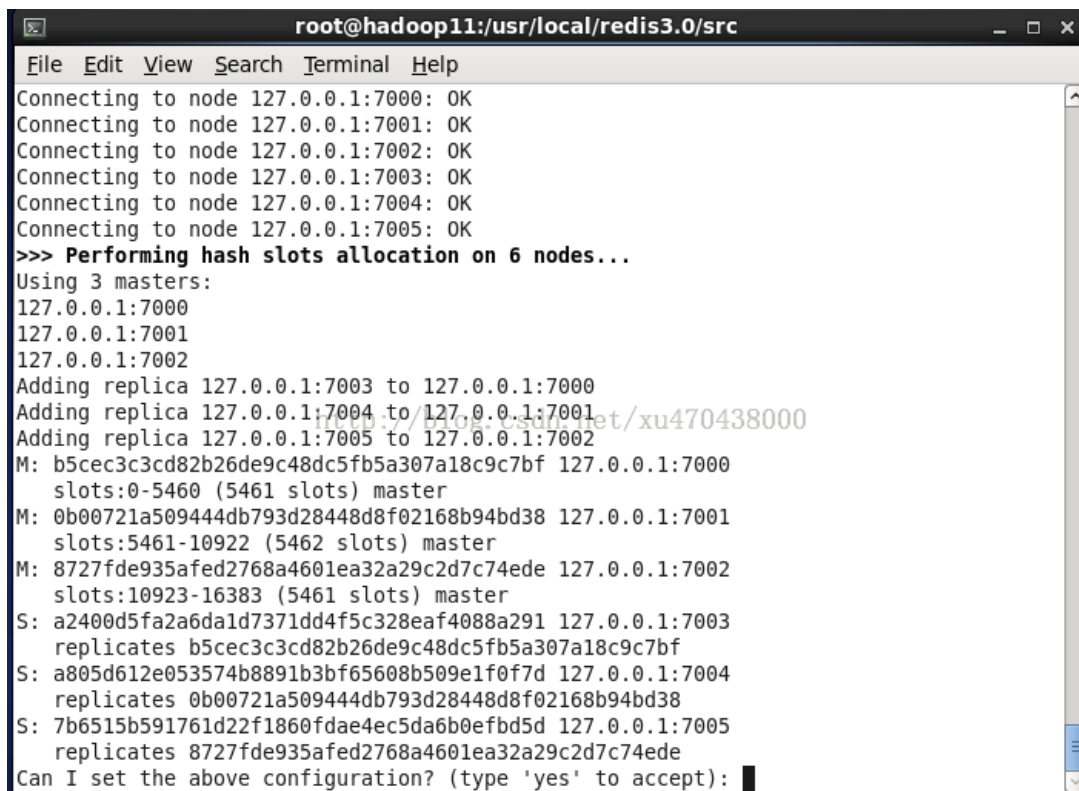
```
/usr/lib/ruby/site_ruby/1.8/rubygems/custom_require.rb:31:in `gem_original_require': no such file to load -- redis (LoadError)
```

```
from /usr/lib/ruby/site_ruby/1.8/rubygems/custom_require.rb:31:in `require'
```

```
from ./redis-trib.rb:25
```

```
gem install redis
```

6.4 再次执行第6步的命令，正常执行

A terminal window titled 'root@hadoop11: /usr/local/redis3.0/src' showing the execution of a Redis cluster creation script. The script connects to six nodes (127.0.0.1:7000-7005) and performs hash slots allocation. It identifies three masters and adds three replicas. The output shows the master-slave replication setup with specific node IDs and slot ranges. The process concludes with a prompt asking if the user wants to set the configuration, with a cursor indicating the user is about to type 'yes'.

```
root@hadoop11: /usr/local/redis3.0/src
File Edit View Search Terminal Help
Connecting to node 127.0.0.1:7000: OK
Connecting to node 127.0.0.1:7001: OK
Connecting to node 127.0.0.1:7002: OK
Connecting to node 127.0.0.1:7003: OK
Connecting to node 127.0.0.1:7004: OK
Connecting to node 127.0.0.1:7005: OK
>>> Performing hash slots allocation on 6 nodes...
Using 3 masters:
127.0.0.1:7000
127.0.0.1:7001
127.0.0.1:7002
Adding replica 127.0.0.1:7003 to 127.0.0.1:7000
Adding replica 127.0.0.1:7004 to 127.0.0.1:7001
Adding replica 127.0.0.1:7005 to 127.0.0.1:7002
M: b5cec3c3cd82b26de9c48dc5fb5a307a18c9c7bf 127.0.0.1:7000
  slots:0-5460 (5461 slots) master
M: 0b00721a509444db793d28448d8f02168b94bd38 127.0.0.1:7001
  slots:5461-10922 (5462 slots) master
M: 8727fde935afed2768a4601ea32a29c2d7c74ede 127.0.0.1:7002
  slots:10923-16383 (5461 slots) master
S: a2400d5fa2a6da1d7371dd4f5c328eaf4088a291 127.0.0.1:7003
  replicates b5cec3c3cd82b26de9c48dc5fb5a307a18c9c7bf
S: a805d612e053574b8891b3bf65608b509e1f0f7d 127.0.0.1:7004
  replicates 0b00721a509444db793d28448d8f02168b94bd38
S: 7b6515b591761d22f1860fdae4ec5da6b0efbd5d 127.0.0.1:7005
  replicates 8727fde935afed2768a4601ea32a29c2d7c74ede
Can I set the above configuration? (type 'yes' to accept):
```

输入yes，然后配置完成。

```
root@hadoop11:usr/local/redis3.0/src
File Edit View Search Terminal Help
replicates 8727fde935afed2768a4601ea32a29c2d7c74ede
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 127.0.0.1:7000)
M: b5cec3c3cd82b26de9c48dc5fb5a307a18c9c7bf 127.0.0.1:7000
  slots:0-5460 (5461 slots) master
M: 0b00721a509444db793d28448d8f02168b94bd38 127.0.0.1:7001
  slots:5461-10922 (5462 slots) master
M: 8727fde935afed2768a4601ea32a29c2d7c74ede 127.0.0.1:7002
  slots:10923-16383 (5461 slots) master
M: a2400d5fa2a6da1d7371dd4f5c328eaf4088a291 127.0.0.1:7003
  slots: (0 slots) master
  replicates b5cec3c3cd82b26de9c48dc5fb5a307a18c9c7bf
M: a805d612e053574b8891b3bf65608b509e1f0f7d 127.0.0.1:7004
  slots: (0 slots) master
  replicates 0b00721a509444db793d28448d8f02168b94bd38
M: 7b6515b591761d22f1860fdae4ec5da6b0efbd5d 127.0.0.1:7005
  slots: (0 slots) master
  replicates 8727fde935afed2768a4601ea32a29c2d7c74ede
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
[root@hadoop11 src]#
```

至此redis集群即搭建成功！

7: 使用redis-cli命令进入集群环境

```
redis-cli -c -p 7000
```

具体集群中的主从关系，可以连接到集群之后执行cluster nodes查看

## 一:关于redis cluster

### 1:redis cluster的现状

reids-cluster计划在redis3.0中推出，可以看作者antirez的声明:<http://antirez.com/news/49> (ps:跳票了好久，今年貌似加快速度了),目前的最新版本是redis3 beta2(2.9.51).

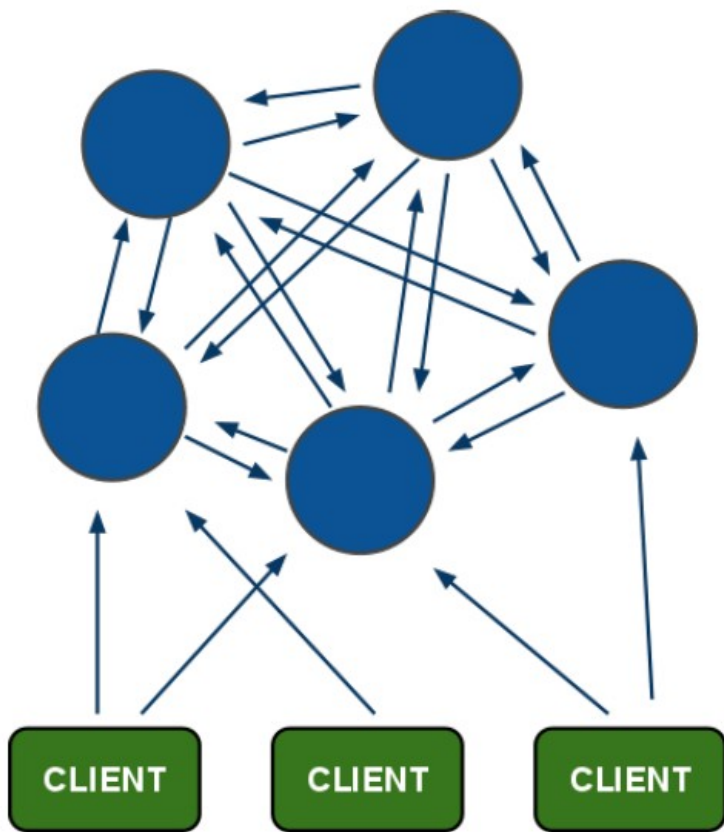
作者的目标:Redis Cluster will support up to ~1000 nodes. 赞...

目前redis支持的cluster特性(已亲测):

- 1):节点自动发现
- 2):slave->master 选举,集群容错
- 3):Hot resharding:在线分片
- 4):进群管理:cluster xxx
- 5):基于配置(nodes-port.conf)的集群管理
- 6):ASK 转向/MOVED 转向机制.

### 2:redis cluster 架构

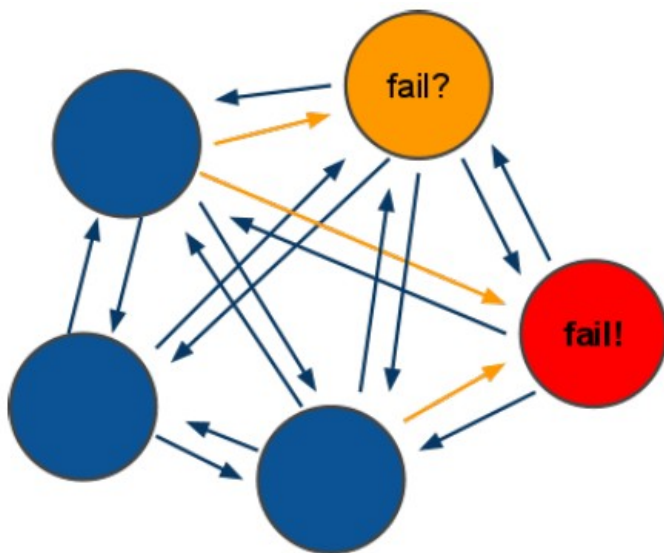
- 1)redis-cluster架构图



架构细节:

- (1)所有的redis节点彼此互联(PING-PONG机制),内部使用二进制协议优化传输速度和带宽.
- (2)节点的fail是通过集群中超过半数的节点检测失效时才生效.
- (3)客户端与redis节点直连,不需要中间proxy层.客户端不需要连接集群所有节点,连接集群中任何一个可用节点即可
- (4)redis-cluster把所有的物理节点映射到[0-16383]slot上,cluster 负责维护node<->slot<->value

2) redis-cluster选举:容错



(1)领着选举过程是集群中所有master参与,如果半数以上master节点与master节点通信超过(cluster-node-timeout),认为当前master节点挂掉.

(2):什么时候整个集群不可用(cluster\_state:fail),当集群不可用时,所有对集群的操作做都不可用,收到((error) CLUSTERDOWN The cluster is down)错误

- a:如果集群任意master挂掉,且当前master没有slave.集群进入fail状态,也可以理解成进群的slot映射[0-16383]不完成时进入fail状态.
- b:如果进群超过半数以上master挂掉,无论是否有slave集群进入fail状态.

## 二:redis cluster的使用

## 1:安装redis cluster

1):安装redis-cluster依赖:redis-cluster的依赖库在使用时有兼容问题,在reshard时会遇到各种错误,请按指定版本安装.

(1)确保系统安装zlib,否则gem install会报(no such file to load -- zlib)

Java代码 ☆

```
1. #download:zlib-1.2.6.tar
2. ./configure
3. make
4. make install
```

(1)安装ruby:version(1.9.2)

Java代码 ☆

```
1. # ruby1.9.2
2. cd /path/ruby
3. ./configure -prefix=/usr/local/ruby
4. make
5. make install
6. sudo cp ruby /usr/local/bin
```

(2)安装rubygem:version(1.8.16)

Java代码 ☆

```
1. # rubygems-1.8.16.tgz
2. cd /path/gem
3. sudo ruby setup.rb
4. sudo cp bin/gem /usr/local/bin
```

(3)安装gem-redis:version(3.0.0)

Java代码 ☆

```
1. gem install redis --version 3.0.0
2. #由于源的原因,可能下载失败,就手动下载下来安装
3. #download地址:http://rubygems.org/gems/redis/versions/3.0.0
4. gem install -l /data/soft/redis-3.0.0.gem
```

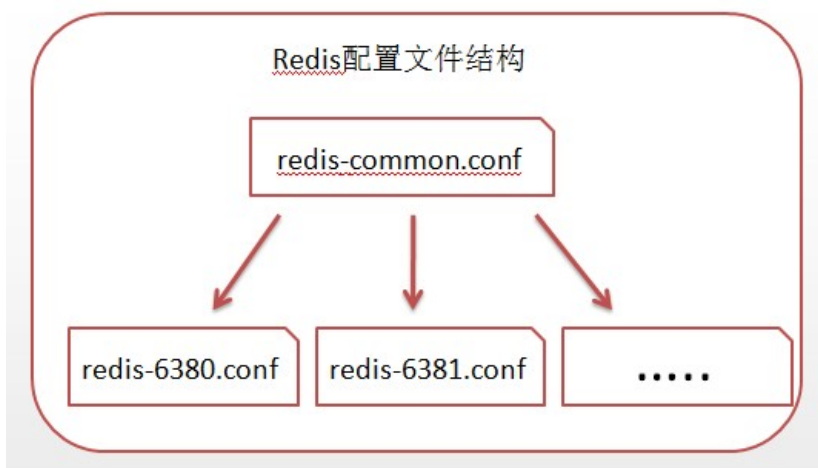
2)安装redis-cluster

Java代码 ☆

```
1. cd /path/redis
2. make
3. sudo cp /opt/redis/src/redis-server /usr/local/bin
4. sudo cp /opt/redis/src/redis-cli /usr/local/bin
5. sudo cp /opt/redis/src/redis-trib.rb /usr/local/bin
```

2:配置redis cluster

1)redis配置文件结构:



使用包含(include)把通用配置和特殊配置分离,方便维护.

## 2)redis通用配置.

Java代码 ☆

```

1. #GENERAL
2. daemonize no
3. tcp-backlog 511
4. timeout 0
5. tcp-keepalive 0
6. loglevel notice
7. databases 16
8. dir /opt/redis/data
9. slave-serve-stale-data yes
10. #slave只读
11. slave-read-only yes
12. #not use default
13. repl-disable-tcp-nodelay yes
14. slave-priority 100
15. #打开aof持久化
16. appendonly yes
17. #每秒一次aof写
18. appendfsync everysec
19. #关闭在aof rewrite的时候对新的写操作进行fsync
20. no-appendfsync-on-rewrite yes
21. auto-aof-rewrite-min-size 64mb
22. lua-time-limit 5000
23. #打开redis集群
24. cluster-enabled yes
25. #节点互连超时的阈值
26. cluster-node-timeout 15000
27. cluster-migration-barrier 1
28. slowlog-log-slower-than 10000
29. slowlog-max-len 128
30. notify-keyspace-events ""
31. hash-max-ziplist-entries 512
32. hash-max-ziplist-value 64
33. list-max-ziplist-entries 512
34. list-max-ziplist-value 64
35. set-max-intset-entries 512
36. zset-max-ziplist-entries 128
37. zset-max-ziplist-value 64
38. activerehashing yes
39. client-output-buffer-limit normal 0 0 0
40. client-output-buffer-limit slave 256mb 64mb 60
41. client-output-buffer-limit pubsub 32mb 8mb 60
42. hz 10
43. aof-rewrite-incremental-fsync yes
  
```

## 3)redis特殊配置.

Java代码 ☆

```

1. #包含通用配置
2. include /opt/redis/redis-common.conf
3. #监听tcp端口
4. port 6379
5. #最大可用内存
6. maxmemory 100m
7. #内存耗尽时采用的淘汰策略:
8. # volatile-lru -> remove the key with an expire set using an LRU algorithm
9. # allkeys-lru -> remove any key accordingly to the LRU algorithm
10. # volatile-random -> remove a random key with an expire set
11. # allkeys-random -> remove a random key, any key
12. # volatile-ttl -> remove the key with the nearest expire time (minor TTL)
13. # noeviction -> don't expire at all, just return an error on write operations
14. maxmemory-policy allkeys-lru
  
```

```
15. #aof存储文件
16. appendfilename "appendonly-6379.aof"
17. #rdb文件,只用于动态添加slave过程
18. dbfilename dump-6379.rdb
19. #cluster配置文件(启动自动生成)
20. cluster-config-file nodes-6379.conf
21. #部署在同一机器的redis实例,把<span style="font-size: 1em; line-height: 1.5;">auto-aof-rewrite搓开,防止瞬间fork所有redis
    进程做rewrite,占用大量内存</span>
22. auto-aof-rewrite-percentage 80-100
```

### 3:cluster 操作

cluster集群相关命令,更多redis相关命令见文档:<http://redis.readthedocs.org/en/latest/>

#### Java代码 ☆

```
1. 集群
2. CLUSTER INFO 打印集群的信息
3. 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。
4. 节点
5. 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。
6. CLUSTER FORGET <node_id> 从集群中移除 node_id 指定的节点。
7. CLUSTER REPLICATE <node_id> 将当前节点设置为 node_id 指定的节点的从节点。
8. CLUSTER SAVECONFIG 将节点的配置文件保存到硬盘里面。
9. 槽(slot)
10. CLUSTER ADDSLOTS <slot> [slot ...] 将一个或多个槽(slot)指派(assign)给当前节点。
11. CLUSTER DELSLOTS <slot> [slot ...] 移除一个或多个槽对当前节点的指派。
12. 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。
13. 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。
14. CLUSTER SETSLOT <slot> MIGRATING <node_id> 将本节点的槽 slot 迁移到 node_id 指定的节点中。
15. CLUSTER SETSLOT <slot> IMPORTING <node_id> 从 node_id 指定的节点中导入槽 slot 到本节点。
16. CLUSTER SETSLOT <slot> STABLE 取消对槽 slot 的导入(import)或者迁移(migrate)。
17. 键
18. CLUSTER KEYSLOT <key> 计算键 key 应该被放置在哪个槽上。
19. CLUSTER COUNTKEYSINSLOT <slot> 返回槽 slot 目前包含的键值对数量。
20. CLUSTER GETKEYSINSLOT <slot> <count> 返回 count 个 slot 槽中的键。
```

### 4:redis cluster 运维操作

#### 1)初始化并构建集群

(1)#启动集群相关节点(必须是空节点),指定配置文件和输出日志

#### Java代码 ☆

```
1. redis-server /opt/redis/conf/redis-6380.conf > /opt/redis/logs/redis-6380.log 2>&1 &
2. redis-server /opt/redis/conf/redis-6381.conf > /opt/redis/logs/redis-6381.log 2>&1 &
3. redis-server /opt/redis/conf/redis-6382.conf > /opt/redis/logs/redis-6382.log 2>&1 &
4. redis-server /opt/redis/conf/redis-7380.conf > /opt/redis/logs/redis-7380.log 2>&1 &
5. redis-server /opt/redis/conf/redis-7381.conf > /opt/redis/logs/redis-7381.log 2>&1 &
6. redis-server /opt/redis/conf/redis-7382.conf > /opt/redis/logs/redis-7382.log 2>&1 &
```

(2):使用自带的ruby工具(redis-trib.rb)构建集群

#### Java代码 ☆

```
1. #redis-trib.rb的create子命令构建
2. #-replicas 则指定了为Redis Cluster中的每个Master节点配备几个Slave节点
3. #节点角色由顺序决定,先master之后是slave(为方便辨认,slave的端口比master大1000)
4. redis-trib.rb create --replicas 1 10.10.34.14:6380 10.10.34.14:6381 10.10.34.14:6382 10.10.34.14:7380
    10.10.34.14:7381 10.10.34.14:7382
```

(3):检查集群状态,

#### Java代码 ☆

```
1. #redis-trib.rb的check子命令构建
2. #ip:port可以是集群的任意节点
3. redis-trib.rb check 1 10.10.34.14:6380
```

最后输出如下信息,没有任何警告或错误,表示集群启动成功并处于ok状态

#### Java代码 ☆

```
1. [OK] All nodes agree about slots configuration.
```



```
2. >>> Check for open slots...
3. >>> Check slots coverage...
4. [OK] All 16384 slots covered.
```

## 2):添加新master节点

(1)添加一个master节点:创建一个空节点(empty node),然后将某些slot移动到这个空节点上,这个过程目前需要人工干预

a):根据端口生成配置文件(ps:establish\_config.sh是我自己写的输出配置脚本)

Java代码 ☆

```
1. sh establish_config.sh 6386 > conf/redis-6386.conf
```

b):启动节点

Java代码 ☆

```
1. nohup redis-server /opt/redis/conf/redis-6386.conf > /opt/redis/logs/redis-6386.log 2>&1 &
```

c):加入空节点到集群

add-node 将一个节点添加到集群里面, 第一个是新节点ip:port, 第二个是任意一个已存在节点ip:port

Java代码 ☆

```
1. redis-trib.rb add-node 10.10.34.14:6386 10.10.34.14:6381
```

node:新节点没有包含任何数据, 因为它没有包含任何slot。新加入的节点是一个主节点, 当集群需要将某个从节点升级为新的主节点时, 这个新节点不会被选中

d):为新节点分配slot

Java代码 ☆

```
1. redis-trib.rb reshard 10.10.34.14:6386
2. #根据提示选择要迁移的slot数量(ps:这里选择500)
3. How many slots do you want to move (from 1 to 16384)? 500
4. #选择要接受这些slot的node-id
5. What is the receiving node ID? f51e26b5d5ff74f85341f06f28f125b7254e61bf
6. #选择slot来源:
7. #all表示从所有的master重新分配,
8. #或者数据要提取slot的master节点id,最后用done结束
9. Please enter all the source node IDs.
10. Type 'all' to use all the nodes as source nodes for the hash slots.
11. Type 'done' once you entered all the source nodes IDs.
12. Source node #1:all
13. #打印被移动的slot后, 输入yes开始移动slot以及对应的数据.
14. #Do you want to proceed with the proposed reshard plan (yes/no)? yes
15. #结束
```

## 3):添加新的slave节点

a):前三步操作同添加master一样

b)第四步:redis-cli连接上新节点shell,输入命令:cluster replicate 对应master的node-id

Java代码 ☆

```
1. cluster replicate 2b9ebcbd627ff0fd7a7bbcc5332fb09e72788835
```

note:在线添加slave时, 需要dump整个master进程, 并传递到slave, 再由slave加载rdb文件到内存, rdb传输过程中Master可能无法提供服务,整个过程消耗大量io,小心操作.

例如本次添加slave操作产生的rdb文件

Java代码 ☆

```
1. -rw-r--r-- 1 root root 34946 Apr 17 18:23 dump-6386.rdb
2. -rw-r--r-- 1 root root 34946 Apr 17 18:23 dump-7386.rdb
```

4):在线reshard 数据:

对于负载/数据均匀的情况,可以在线reshard slot来解决,方法与添加新master的reshard一样,只是需要reshard的master节点是老节点.

## 5):删除一个slave节点

Java代码 ☆

```
1. #redis-trib del-node ip:port '<node-id>'
2. redis-trib.rb del-node 10.10.34.14:7386 'c7ee2fca17cb79fe3c9822ced1d4f6c5e169e378'
```

6):删除一个master节点

a):删除master节点之前首先要使用reshard移除master的全部slot,然后再删除当前节点(目前只能把被删除master的slot迁移到一个节点上)

Java代码 ☆

```
1. #把10.10.34.14:6386当前master迁移到10.10.34.14:6380上
2. redis-trib.rb reshard 10.10.34.14:6380
3. #根据提示选择要迁移的slot数量(ps:这里选择500)
4. How many slots do you want to move (from 1 to 16384)? 500(被删除master的所有slot数量)
5. #选择要接受这些slot的node-id(10.10.34.14:6380)
6. What is the receiving node ID? c4a31c852f81686f6ed8bcd6d1b13accdc947fd2 (ps:10.10.34.14:6380的node-id)
7. Please enter all the source node IDs.
8. Type 'all' to use all the nodes as source nodes for the hash slots.
9. Type 'done' once you entered all the source nodes IDs.
10. Source node #1:f51e26b5d5ff74f85341f06f28f125b7254e61bf(被删除master的node-id)
11. Source node #2:done
12. #打印被移动的slot后,输入yes开始移动slot以及对应的数据.
13. #Do you want to proceed with the proposed reshard plan (yes/no)? yes
```

b):删除空master节点

Java代码 ☆

```
1. redis-trib.rb del-node 10.10.34.14:6386 'f51e26b5d5ff74f85341f06f28f125b7254e61bf'
```

## 三:redis cluster 客户端(Jedis)

### 1:客户端基本操作使用

Java代码 ☆

```
1. <span style="color: #333333; font-family: Arial, sans-serif;"><span style="color: #333333; font-family: Arial, sans-serif;"> private static BinaryJedisCluster jc;
2. static {
3.     //只给集群里一个实例就可以
4.     Set<HostAndPort> jedisClusterNodes = new HashSet<HostAndPort>();
5.     jedisClusterNodes.add(new HostAndPort("10.10.34.14", 6380));
6.     jedisClusterNodes.add(new HostAndPort("10.10.34.14", 6381));
7.     jedisClusterNodes.add(new HostAndPort("10.10.34.14", 6382));
8.     jedisClusterNodes.add(new HostAndPort("10.10.34.14", 6383));
9.     jedisClusterNodes.add(new HostAndPort("10.10.34.14", 6384));
10.    jedisClusterNodes.add(new HostAndPort("10.10.34.14", 7380));
11.    jedisClusterNodes.add(new HostAndPort("10.10.34.14", 7381));
12.    jedisClusterNodes.add(new HostAndPort("10.10.34.14", 7382));
13.    jedisClusterNodes.add(new HostAndPort("10.10.34.14", 7383));
14.    jedisClusterNodes.add(new HostAndPort("10.10.34.14", 7384));
15.    jc = new BinaryJedisCluster(jedisClusterNodes);
16. }
17. @Test
18. public void testBenchRedisSet() throws Exception {
19.     final Stopwatch stopwatch = new Stopwatch();
20.     List list = buildBlogVideos();
21.     for (int i = 0; i < 1000; i++) {
22.         String key = "key:" + i;
23.         stopwatch.start();
24.         byte[] bytes1 = protostuffSerializer.serialize(list);
25.         jc.setex(key, 60 * 60, bytes1);
26.         stopwatch.stop();
27.     }
28.     System.out.println("time=" + stopwatch.toString());
29. }</span></span>
```

### 2:jedis客户端的坑.

- 1)cluster环境下redis的slave不接受任何读写操作，
- 2)client端不支持keys批量操作,不支持select dbNum操作， 只有一个db:select 0
- 3)JedisCluster 的info()等单机函数无法调用,返回(No way to dispatch this command to Redis Cluster)错误， .
- 4)JedisCluster 没有针对byte[]的API， 需要自己扩展(附件是我加的基于byte[]的BinaryJedisCluster api)