|  |
| --- |
| Neusoft |
| CentOS7 & Redis3.0集群环境 |
| Ridis 3.0集群环境搭建 |

|  |
| --- |
| 成阳阳  2015/5/15 |
|  |

## Redis集群部署文档(centos7系统)

## 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  [藏代码](javascript:void())

1. #download:zlib-1.2.6.tar
2. ./configure
3. make
4. make install
5. #或 yum安装
6. yum install zlib

##### (2)安装ruby:version(1.9.2)

Java代码  [藏代码](javascript:void())

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
7. #或yum安装
8. yum install ruby

##### (3)安装rubygems:version(1.8.16)

Java代码  [藏代码](javascript:void())

1. # rubygems-1.8.16.tgz
2. cd /path/gem
3. sudo ruby setup.rb
4. sudo cp bin/gem /usr/local/bin
5. #或yum 安装
6. yum install rubygems

##### (4)安装gem-redis:version(3.0.0)

Java代码  [藏代码](javascript:void())

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/3.0.1
4. #从CSDN上下载的资源 redis-3.0.6.gem
5. gem install -l /data/soft/redis-3.0.6.gem

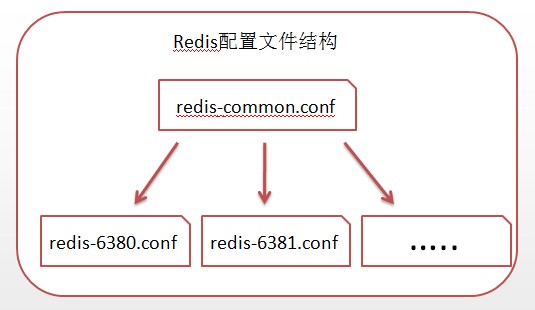
##### (5)安装redis-cluster

Java代码  [藏代码](javascript:void())

1. #安装ActiveTcl-8.6,解压后运行install.sh
2. ./install.sh
3. tar xvf redis-3.0.1.tar.gz
4. cp -rf redis-3.0.1/ /usr/local/
5. ln -s /usr/local/redis-3.0.1 /usr/local/redis
6. make && make install
7. #创建 data logs bin cluster目录
8. mkdir data
9. mkdir logs
10. mkdir bin
11. mkdir conf

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  藏代码**

1. #GENERAL
2. daemonize yes
3. tcp-backlog 511
4. timeout 0
5. tcp-keepalive 0
6. loglevel notice
7. databases 16
8. dir /usr/local/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. #节点互连超时的阀值
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  藏代码**

1. #包含通用配置
2. include /usr/local/redis/conf/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
23. logfile “/usr/local/redis/logs/cluster-nodes-6379.out”

 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/](http://redis.readthedocs.org/en/latest/" \t "_blank)

**Java代码  藏代码**

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  藏代码**

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  藏代码**

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1)
4. ../src/redis-trib.rb create --replicas 1 192.168.24.100:6379 192.168.24.100:7379 192.168.24.100:8379 192.168.24.100:6380 192.168.24.100:7380 192.168.24.100:8380

(3):检查集群状态,

**Java代码  藏代码**

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. ../src/redis-trib.rb check 192.168.24.100:6379

 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  藏代码**

1. [OK] All nodes agree about slots configuration.
2. >>> Check **for** open slots...
3. >>> Check slots coverage...
4. [OK] All 16384 slots covered.

访问集群环境

../src/redis-cli -c -p 6379

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  藏代码**

1. sh establish\_config.sh 6386 > conf/redis-6386.conf

b):启动节点

**Java代码  藏代码**

1. ..src/redis-server /usr/local/redis/conf/redis-9379.conf

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  藏代码**

1. redis-trib.rb add-node 192.168.24.100:9379 192.168.24.100:6379

node:新节点没有包含任何数据， 因为它没有包含任何slot。新加入的加点是一个主节点， 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  藏代码**

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  藏代码**

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  藏代码**

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  藏代码**

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  藏代码**

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  藏代码**

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  藏代码**

1. **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)

参考文档:

[http://redis.io/topics/cluster-spec](http://redis.io/topics/cluster-spec" \t "_blank)

[http://redis.io/topics/cluster-tutorial](http://redis.io/topics/cluster-tutorial" \t "_blank)