Redis Cluster集群安装说明

目录

[Redis Cluster集群环境搭建 1](#_Toc447699052)

[1. Redis Cluster集群介绍 1](#_Toc447699053)

[2. 环境准备 2](#_Toc447699054)

[1) 安装redis 2](#_Toc447699055)

[2) 安装ruby 2](#_Toc447699056)

[3) 系统参数 2](#_Toc447699057)

[3. 集群安装 3](#_Toc447699058)

[1) 配置说明 3](#_Toc447699059)

[2) 配置目录 3](#_Toc447699060)

[3) 配置文件 3](#_Toc447699061)

[4) 启动节点 5](#_Toc447699062)

[5) 创建集群 5](#_Toc447699063)

[4. 集群管理 6](#_Toc447699064)

[1) 检查集群状态 6](#_Toc447699065)

[2) 添加新主节点 7](#_Toc447699066)

[3) 添加新从节点 8](#_Toc447699067)

[4) 移除从节点 9](#_Toc447699068)

[5) 移除主节点 9](#_Toc447699069)

[6) 重新分配slot 9](#_Toc447699070)

[7) 查看统计信息 12](#_Toc447699071)

[8) 停止节点 13](#_Toc447699072)

[5. 客户端使用 13](#_Toc447699073)

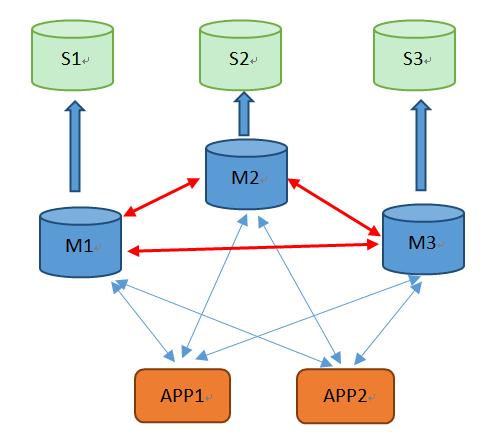
[1) Java客户端 (Jedis) 13](#_Toc447699074)

[6. 注意事项 13](#_Toc447699075)

[1) 密码设置 13](#_Toc447699076)

1. Redis Cluster集群介绍

Redis Cluster是官方提供的方案，是一种服务器Sharding（分片）技术，3.0版本开始正式提供。集群环境至少需要3个节点。推荐使用6个节点配置，即3个主节点，3个从节点。拓扑结构如下图所示：



1. 环境准备
2. 软件环境

Centos 2.6.32-279.el6.x86\_6

gcc version 4.4.6

redis-3.0.7

ruby 1.8.7

1. 安装redis

wget http://download.redis.io/releases/redis-3.0.7.tar.gz

tar xzf redis-3.0.7.tar.gz

cd redis-3.0.7

make

1. 安装ruby

后面配置集群需要用到ruby命令redis-trib.rb。所以先安装ruby环境。依次执行以下命令。

yum install -y ruby

yum install -y rubygems

gem install redis

1. 系统参数

* /proc/sys/net/core/somaxconn参数

定义了系统中每一个端口最大的监听队列的长度,这是个全局的参数,默认值为128，redis默认为511，需要如下设置：echo 511 >/proc/sys/net/core/somaxconn

* /proc/sys/vm/overcommit\_memory参数

定义了内存分配策略。可选值：0、1、2。

0表示将检查是否有足够的可用内存供应用进程使用；如果有足够的可用内存，内存申请允许；否则，内存申请失败，并把错误返回给应用进程。

1 表示内核允许分配所有的物理内存，而不管当前的内存状态如何。

2 表示内核允许分配超过所有物理内存和交换空间总和的内存，需要如下设置：echo 1 > /proc/sys/vm/overcommit\_memory

1. 集群安装
2. 配置说明

* 服务器分配

该集群环境为3主3从分配在2台服务器上，分配如下：

主172.16.6.55：7000 ,从172.16.6.29：7000

主172.16.6.55：7001 ,从172.16.6.29：7001

主172.16.6.29：7002 ,从172.16.6.55：7002

* 缓存无持久化。
* 内存设置，各个实例设置内存1G，使用allkeys-lru（任何key近期最少使用）算法来移除key。
* 配置参数解释，参考同目录下文档《Redis参数配置说明》。

1. 配置目录

// 服务器1

cd /usr/local

mkdir cluster-test

cd cluster-test

mkdir 7000 7001 7002

// 服务器2

cd /usr/local

mkdir cluster-test

cd cluster-test

mkdir 7000 7001 7002

1. 配置文件

* 公共配置文件/usr/local/cluster-test/redis\_common.conf

#GENERAL

daemonize yes

tcp-backlog 511

timeout 0

tcp-keepalive 0

loglevel notice

databases 16

# SNAPSHOTTING

#save 900 1

#save 300 10

#save 60 10000

save ""

dir ./

#REPLICATION

slave-serve-stale-data yes

slave-read-only yes

#not use default

repl-disable-tcp-nodelay yes

slave-priority 100

#SECURITY

#requirepass RedisHello123..

# APPEND ONLY MODE

appendonly no

appendfsync everysec

no-appendfsync-on-rewrite yes

auto-aof-rewrite-min-size 64mb

# LUA SCRIPTING

lua-time-limit 5000

# REDIS CLUSTER

cluster-enabled yes

cluster-node-timeout 5000

cluster-migration-barrier 1

cluster-require-full-coverage no

# SLOW LOG

slowlog-log-slower-than 10000

slowlog-max-len 128

#Event notification

notify-keyspace-events ""

#ADVANCED CONFIG

hash-max-ziplist-entries 512

hash-max-ziplist-value 64

list-max-ziplist-entries 512

list-max-ziplist-value 64

set-max-intset-entries 512

zset-max-ziplist-entries 128

zset-max-ziplist-value 64

activerehashing yes

client-output-buffer-limit normal 0 0 0

client-output-buffer-limit slave 256mb 64mb 60

client-output-buffer-limit pubsub 32mb 8mb 60

hz 10

aof-rewrite-incremental-fsync yes

* 节点1配置文件/usr/local/cluster-test/7000/redis\_7000.conf

include /usr/local/cluster-test/redis\_common.conf

port 7000

pidfile /var/run/redis\_7000.pid

logfile /var/log/redis\_7000.log

maxmemory 1g

maxmemory-policy allkeys-lru

appendfilename "appendonly-7000.aof"

dbfilename dump-7000.rdb

cluster-config-file nodes-7000.conf

auto-aof-rewrite-percentage 80-100

* 其他节点配置

cd /usr/local/cluster-test

cp 7000/redis\_7000.conf 7001/redis\_7001.conf && sed -i 's/7000/7001/g' 7001/redis\_7001.conf

cp 7000/redis\_7000.conf 7002/redis\_7002.conf && sed -i 's/7000/7002/g' 7002/redis\_7002.conf

1. 启动节点

cd 7000

../../redis-3.0.7/src/redis-server ./redis\_7000.conf  
cd 7001

../../redis-3.0.7/src/redis-server ./redis\_7001.conf

依次启动其他节点

1. 创建集群

cd /usr/local/redis-3.0.7/src

./redis-trib.rb create --replicas 1 172.16.6.55:7000 172.16.6.55:7001 172.16.6.29:7002 172.16.6.29:7000 172.16.6.29:7001 172.16.6.55:7002

注：选项–replicas 1 表示我们希望为集群中的每个主节点创建一个从节点。

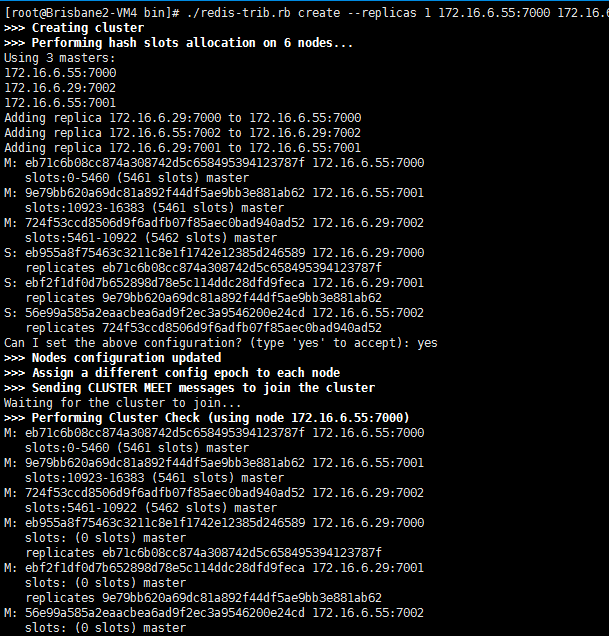
以上命令会依顺序创建前三个节点172.16.6.55:7000 172.16.6.55:7001 172.16.6.29:7002为主节点，并创建以下三个从节点：

172.16.6.29:7000为 172.16.6.55:7000从节点

172.16.6.29:7001为172.16.6.55:7001从节点

172.16.6.55:7002为172.16.6.29:7002从节点

类似结果如下：



注：创建好集群后，可以通过redis-trib.rb check查看集群状态，参考“检查集群状态”小节。

1. 集群管理
2. 检查集群状态

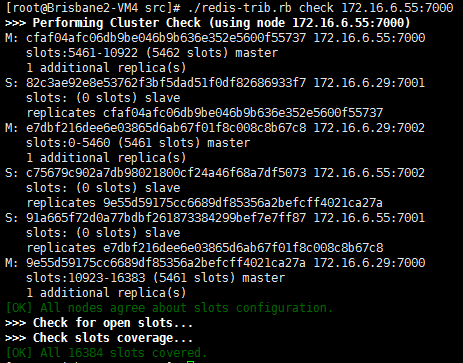
./redis-trib.rb check <集群中任意一个节点>

例如：

cd /usr/local/redis-3.0.7/src

./redis-trib.rb check 127.0.0.1:7000

结果如下：



1. 添加新主节点

* 创建目录

// 服务器2

cd /usr/local

mkdir cluster-test

cd cluster-test

mkdir 7003

* 创建配置

cp 7000/redis\_7000.conf 7003/redis\_7003.conf && sed -i 's/7000/7003/g' 7003/redis\_7003.conf

* 启动新节点

cd 7003

../../redis-3.0.7/src/redis-server ./redis\_7003.conf

* 执行命令

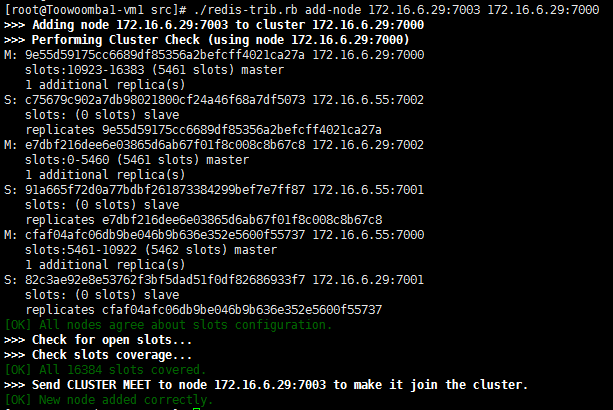
redis-trib.rb add-node <新主节点> <集群中任意一个节点>

例如：

cd /usr/local/redis-3.0.7/src

./redis-trib.rb add-node 172.16.6.29:7003 172.16.6.29:7000

结果如下：



注：该命令之添加了一个空的节点 ，可以通过redis-trib.rb reshard <ip:port>命令从现有的主节点上分配slot到新的节点上。参考本文档“重新分配slot”小节。

1. 添加新从节点

* 创建目录

// 服务器1

cd /usr/local

mkdir cluster-test

cd cluster-test

mkdir 7003

* 创建配置

cp 7000/redis\_7000.conf 7003/redis\_7003.conf && sed -i 's/7000/7003/g' 7003/redis\_7003.conf

* 启动新节点

cd 7003

../../redis-3.0.7/src/redis-server ./redis\_7003.conf

* 没有指定主节点，系统会在其他的复制集中的主节点中随机选取一个作为这个从节点的主节点。

redis-trib.rb add-node –slave <新从节点> <集群中任意一个节点>

例如：

cd /usr/local/redis-3.0.7/src

./redis-trib.rb add-node --slave 172.16.6.55:7003 172.16.6.55:7000

* 指定主节点

redis-trib.rb add-node --slave --master-id <主节点ID> <新从节点> <集群中任意一个节点>

例如：

cd /usr/local/redis-3.0.7/src

./redis-trib.rb add-node --slave --master-id

3c3a0c74aae0b56170ccb03a76b60cfe7dc1912e 172.16.6.55:7003 172.16.6.55:7000

1. 移除从节点

redis-trib del-node ip:port '<节点ID>'

例如：

./redis-trib.rb del-node 172.16.6.55:7003 'c7ee2fca17cb79fe3c9822ced1d4f6c5e169e378'

1. 移除主节点

* 移除空主节点

参考“移除从节点”小节。

* 移除非空节点，先移除该节点全部slot到其他主节点

redis-trib.rb reshard <ip:port>

例如：

cd /usr/local/redis-3.0.7/src

./ redis-trib.rb reshard 172.16.6.29:7003

然后再参考“移除空节点”小节。

1. 重新分配slot

命令格式：

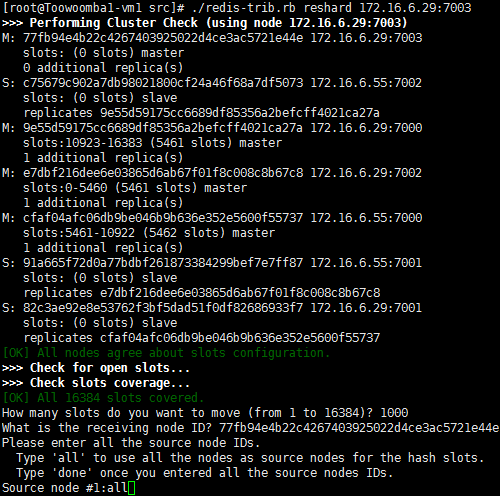
redis-trib.rb reshard <ip:port>

例如：

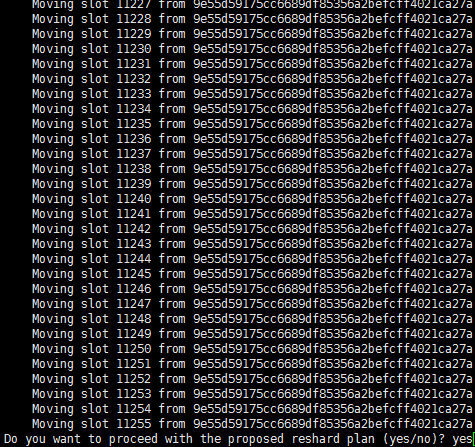
cd /usr/local/redis-3.0.7/src

./redis-trib.rb reshard 172.16.6.29:7003

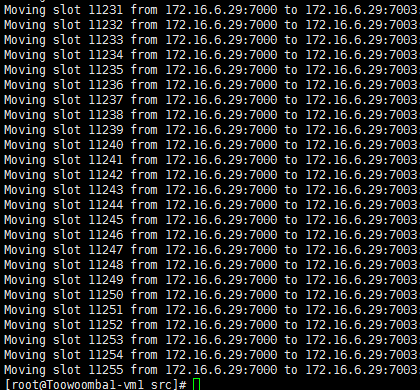
类似结果：



图一



图二



图三

注：

How many slots do you want to move (from 1 to 16384)? 1000 //设置slot数1000

What is the receiving node ID? 03ccad2ba5dd1e062464bc7590400441fafb63f2 //新节点node id

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 //表示从原来所有节点自动分配部分给新节点

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

1. 查看统计信息

* 登录需要查看的节点

redis-cli -c -p <port> -h <host>

例如：

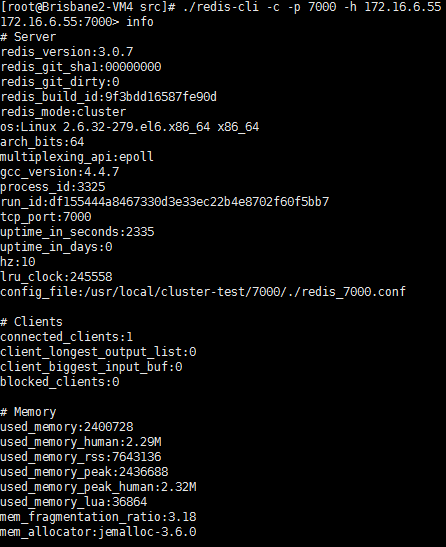
cd /usr/local/redis-3.0.7/src

./redis-cli -c -p 7000 -h 172.16.6.55

* 显示节点信息命令

Info

结果如下：



1. 停止节点

* 登录需要停止的节点

redis-cli -c -p <port> -h <host>

例如：

cd /usr/local/redis-3.0.7/src

./redis-cli -c -p 7003 -h 172.16.6.29

* 停止节点信息命令

shutdown

1. 启动节点

cd 7003

../../redis-3.0.7/src/redis-server ./redis\_7003.conf

1. 客户端使用
2. Java客户端 (Jedis)

static {

// 只给集群里一个实例就可以, APP访问Redis可以通过连接任意一个（或多个）节点实现，

Set<HostAndPort> jedisClusterNodes = new HashSet<HostAndPort>();

jedisClusterNodes.add(new HostAndPort("172.16.6.55", 7000));

jedisClusterNodes.add(new HostAndPort("172.16.6.55", 7001));

jedisClusterNodes.add(new HostAndPort("172.16.6.55", 7002));

jedisClusterNodes.add(new HostAndPort("172.16.6.29", 7000));

jedisClusterNodes.add(new HostAndPort("172.16.6.29", 7001));

jedisClusterNodes.add(new HostAndPort("172.16.6.29", 7002));

jc = new BinaryJedisCluster(jedisClusterNodes);

}

@Test

public void testBenchRedisSet() throws Exception {

final Stopwatch stopwatch = new Stopwatch(); // 执行任务时间监视器

List list = buildBlogVideos(); // 获取资源

for (int i = 0; i < 1000; i++) {

String key = "key:" + i;

stopwatch.start(); // 监视开始

byte[] bytes1 = protostuffSerializer.serialize(list); // 序列化字节

jc.set(key, bytes1);

stopwatch.stop(); // 监视结束

}

System.out.println("time=" + stopwatch.toString()); // 输出执行统计

}

1. 注意事项
2. 密码设置

目前redis cluster集群不支持密码认证。

1. Key集合操作

目前redis cluster集群不支持key集合类的操作。

1. 写入安全

那些与大多数节点相连的客户端所做的写入操作，系统尝试全部都保存下来。不过公认的，还是会有小部分写入会丢失。异步复制，网络分裂（“脑裂”）可能造成异步写入数据丢失，所以Redis 并不能保证数据的强一致性。

1. 可用性

在绝大多数的主节点（master node）是可达的，并且对于每一个不可达的主节点都至少有一个它的从节点（slave）可达的情况下，Redis 集群仍能进行分区（partitions）操作。反之说明大多数不可用，或者其中有主节点及其从节点不可用，则集群进入不可用状态。