**[redis 3.2.5的集群部署](http://blog.csdn.net/myrainblues/article/details/25881535)**

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

架构细节:

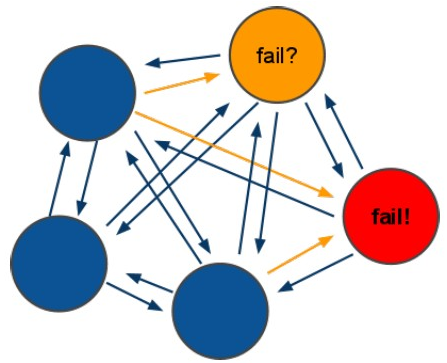
(1)所有的redis节点彼此互联(PING-PONG机制)，内部使用二进制协议优化传输速度和带宽；

(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。



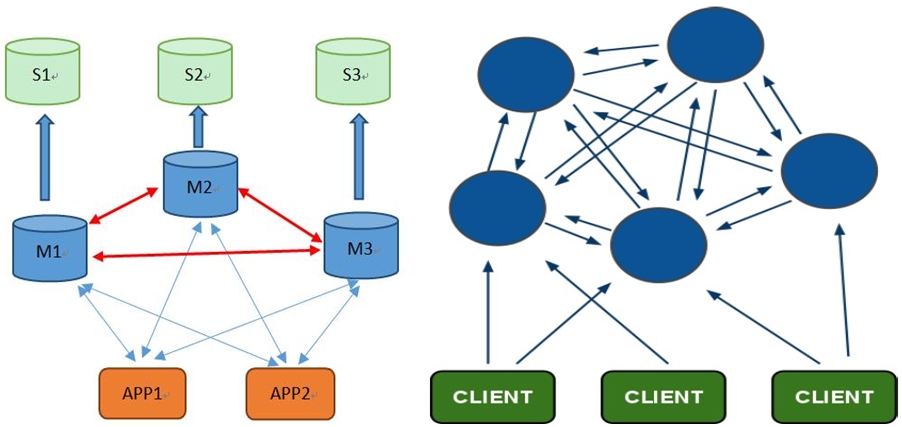
(1)选举过程是集群中所有master参与，如果半数以上master节点与master节点通信超过(cluster-node-timeout)时长，认为当前master节点挂掉。

(2)当集群不可用时，所有对集群的操作都不可用，收到((error： CLUSTERDOWN The cluster is down)。

a:如果集群任意master挂掉，且当前master没有slave，集群进入fail状态，也可以理解成集群的slot映射[0-16383]不完整，进入fail状态。

b:如果集群超过半数以上master挂掉，无论是否有slave集群都进入fail状态。

## redis cluster 架构



**下载和解包**

安装环境

#yum -y install gcc rubygems ruby zib

修改最大可打开文件数

修改文件/etc/security/limits.conf，加入以下两行：

\* soft nofile 102400

\* hard nofile 102400

# End of file

其中 102400 为一个进程最大可以打开的文件个数，当与 RedisServer 的连接数多时，需要设定为合适的值。修改后，需要重新登录才会生效，如果是 crontab则需要重启 crontab，如service crond restart有些平台可能是 service cron restart。

添加vm.overcommit\_memory=1

#vi /etc/sysctl.conf

vm.overcommit\_memory=1

关闭hugepage

默认值为“[always] madvise never ”， 建议设置为 never ， 以开启内核的“Transparent Huge Pages (THP)”特性，设置后 redis 进程需要重启。为了永久生效，请将以下命令参数加入到文件/etc/rc.local 中。

# echo never > /sys/kernel/mm/transparent\_hugepage/enabled

TCP 监听队列大小

即TCP listen的backlog大小，“/proc/sys/net/core/somaxconn”的默认值一般较小如128，需要修改大一点，比如改成 32767。立即生效还可以使用命令：sysctl -w net.core.somaxconn=32767。要想永久生效，在文件/etc/sysctl.conf 中增加一行：net.core.somaxconn = 32767，然后执行命令“sysctl -p”。

关闭防火墙

# service iptables stop

关闭selinux

# vi /etc/sysconfig/selinux

SELINUX=disabled

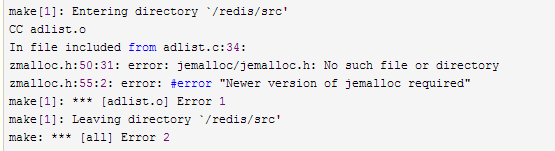
# 安装Cluster

 安装软件

# cd /usr/local/src/redis-3.2.5

# make

# make install



原因是没有安装jemalloc内存分配器，可以安装jemalloc或直接输入make MALLOC=libc && make install。

拷贝bin文件

#cd /usr/local/src/redis-3.2.5/src/

# cp redis-trib.rb redis-cli redis-server redis-benchmark redis-check-aof redis-check-dump /usr/local/bin/

#cd /usr/local/

# mkdir 7000 7001 7002   //分别代表三个节点，其对应端口 7000，7001，7002

创建7000节点为例

# cd ./7000

# cp /usr/local/src/redis-3.2.5/redis.conf   ./     //拷贝到当前7000目录

# vi redis.conf     //编辑配置，主要修改以下几个参数

daemonize yes //redis后台运行

pidfile /var/run/redis\_7000.pid //pidfile文件对应7000

port 7000 //端口7000

cluster-enabled yes //开启集群，把注释#去掉

cluster-config-file nodes.conf //集群的配置，配置文件首次启动自动生成

cluster-node-timeout 5000 //请求超时设置5秒够了

appendonly on //aof日志开启，它会每次写操作都记录一条日志（全持久化）   
#save 900 1  
#save 300 10 //注释掉这几个，这是写磁盘的配置（半持久化）  
#save 60 10000 //开启的话，在启动节点后，会生成相应的dump.rdb

配置好了就相应地把这个修改后的配置文件拷贝到7001，7002目录，注意要修改监听端口port7001，7002。

接下来进入节点目录启动服务，依次执行redis-server   redis.conf，可以看到生成了appendonly.aof ，nodes.conf。

#ps -ef | grep redis

 root 885 0.8 0.2 129452 2620 ? Ssl 20:10 0:21 redis-server \*:7000 [cluster]  
 root 887 0.8 0.2 129452 2616 ? Ssl 20:10 0:21 redis-server \*:7001 [cluster]  
 root 893 0.8 0.2 128356 2612 ? Ssl 20:10 0:21 redis-server \*:7002 [cluster]

 #netstat -tnlp | grep redis 可以看到redis监听端口

 tcp 0 0 0.0.0.0:7000 0.0.0.0:\* LISTEN 885/redis-server \*  
 tcp 0 0 0.0.0.0:7001 0.0.0.0:\* LISTEN 887/redis-server \*  
 tcp 0 0 0.0.0.0:7002 0.0.0.0:\* LISTEN 893/redis-server \*  
 tcp 0 0 0.0.0.0:17000 0.0.0.0:\* LISTEN 885/redis-server \*  
 tcp 0 0 0.0.0.0:17001 0.0.0.0:\* LISTEN 887/redis-server \*  
 tcp 0 0 0.0.0.0:17002 0.0.0.0:\* LISTEN 893/redis-server \*

除了看到配置文件中设置的端口700\*，还有700\*+10000（1700\*）, 前者是客户端访问的，后者是集群内部节点之间访问的。在多台Server间搭建集群，如果开了防火墙，需要设置iptables开放上面所有端口。Server2步骤和Server1一样，设置端口7003，7004，7005。

**创建集群**

前面已经准备好了搭建集群的redis节点，接下来要把这些节点都串连起来搭建集群。官方提供了一个工具：redis-trib.rb（/usr/local/src/redis-3.2.5/src/redis-trib.rb），它是用ruby写的一个程序，所以还得安装ruby。

#yum -y install ruby ruby-devel rubygems rpm-build

再用gem这个命令来安装redis接口，gem貌似是ruby的一个工具包。

#gem install redis     //等一会儿就好了,当然两台Server都要安装

确认所有的redis节点都已经启动，接下来使用参数create 创建集群（在Server1中来创建）。

#/usr/local/src/redis-3.2.5/src/redis-trib.rb  create \

--replicas 1 192.168.0.7:7000 192.168.0.7:7001 192.168.0.7:7002 192.168.0.21:7003 192.168.0.21:7004 192.168.0.21:7005

**注意：这里如果在同一台 机子上面配置六个节点的话，可以写127.0.0.1的ip加端口，但是如果在两台机子上各有三个节点的话，就必须写192.168.开头的ip地址，不然会出现“Waiting for the cluster to join…..一直等待下去的情况。**

如果提示如下错误：

>>> Creating cluster

[ERR] Sorry, can't connect to node 192.168.0.21:7003

**protected-mode no**  这是3.2.0以后redis.conf中新增的配置项，默认值为yes，限制从其它机器登录Redis server， 而只能从127.0.0.1登录。为保证redis-trib.rb工具的正常运行，需要设置为no，完成后可以改回yes。要想从非127.0.0.1访问也需要改为no，或将各个节点的配置文件中的“bind 127.0.0.1”这一项中的回环ip改成本机ip即可。

解释下:--replicas  1 表示自动为每一个master节点分配一个slave节点，上面有6个节点，程序会按照一定规则生成 3个master（主）3个slave(从)，前面已经提醒过，防火墙一定要开放监听的端口，否则会创建失败。

运行中，提示Can I set the above configuration? (type 'yes' to accept): yes    //输入yes

[OK] All nodes agree about slots configuration

>>> Check for open slots...

>>> Check slots coverage...

[OK] All 16384 slots covered. 说明创建成功！

**注意：当集群或各个redis实例非正常关闭（如使用kill命令），重新启动集群，就需要使用如下命令启动各个节点（在nodes-7000.conf所在目录下执行）：**

**#../bin/redis-server redis.conf --cluster-config-file nodes-7000.conf**

**还可以对master和slave进行锁定**

连接到要添加的slave数据库中，执行replicate操作，后面的ID为Master 192.168.0.7:7003的ID。

#./ redis-cli -c -h 192.168.0.21 -p 7002

192.168.0.21:7002> cluster replicate 7e26effeeec1015545eddeccf0ad6d90ecd71bcf

OK

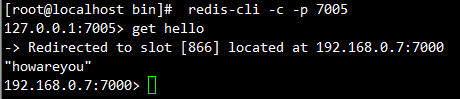
**测试**

get 和 set数据

在192.168.0.7上终端输入：

#./ redis-cli -c -p 7000

进入命令窗口，直接 set  hello   howareyou，直接根据hash匹配切换到相应的slot节点上。然后找到这个节点所对应的从节点，在从节点上执行redis-cli -c -p 7004进入命令行后，输入get hello出现如下界面：

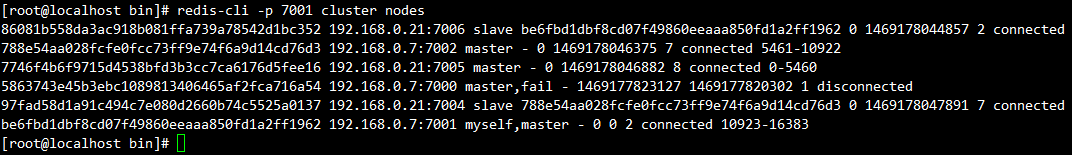


# ./redis-trib.rb  check  192.168.0.7:7002 //检查集群状态（ip和端口可以是任何一个节点）

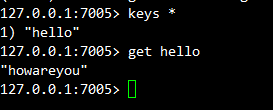
还是要说明一下，redis集群由16383个slot组成，通过分片分布到多个节点上，读写都发生在master节点。假设先把Server2服务Down掉（Server2有1个Master, 2个Slave） ,  查看一下发生了什么事，Server1的3个节点全部都是Master，其他几个Server2的不见了，测试一下依然没有问题，集群依然能继续工作。

原因：redis集群通过选举方式进行容错，保证一台Server挂了还能跑，这个选举是全部集群超过半数以上的Master发现其他Master挂了后，会将其对应的Slave节点升级成Master。

疑问：要是挂的是Server1怎么办？cluster is down!!没办法，超过半数的Master挂了那救不了了，整个集群就无法工作了。 要是有三台Server，每台两个Master，切记对应的主从节点不要放在一台Server，互相交叉配置主从，挂哪台也没事。可以通过执行redis-cli -p 7001 cluster nodes命令进行查看，下图中的7000的机子已经挂了，然后对应的7005从slave升为了master。



并且这时故障已经转移，获取到了key，value。

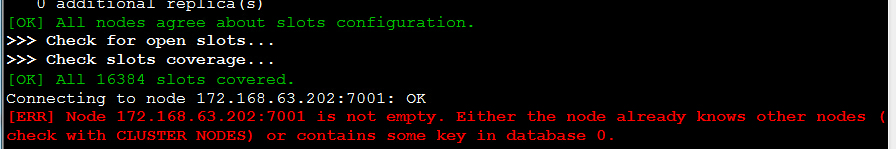


**关于一致性**

**从 slaves 读数据**

默认不能从 slaves 读取数据，但建立连接后，执行一次命令 [READONLY](http://redis.io/commands/readonly)，即可从 slaves读取数据。如果想再次恢复不能从 slaves 读取数据，可以执行命令 READWRITE。

我还没有这么大胆拿redis来做数据库持久化网站数据，只是拿来做cache，官网说的很清楚，Redis Cluster is not able to guarantee **strong consistency**。



解决方法：

1)、将需要新增的节点下aof、rdb等本地备份文件删除；

2)、同时将新Node的集群配置文件删除，即删除redis.conf里面cluster-config-file所在的文件；

3)、再次添加新节点如果还是报错，则登录新Node, redis-cli -c -p 端口号对数据库进行清除：

172.168.63.201:7001>  flushdb      #清空当前数据库

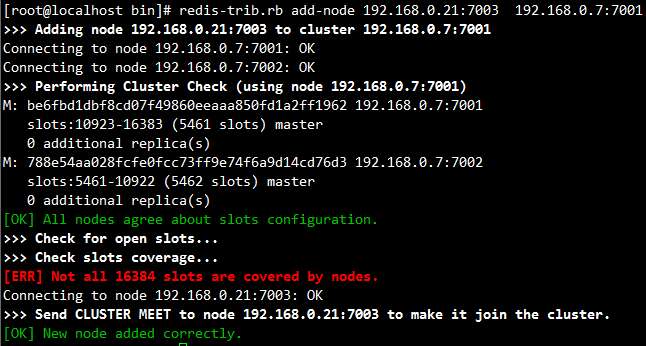
**添加master节点**

照上面的操作，新加一个配置文件和端口，例如7006/redis.conf，端口是7006，并且更改防火墙的设置后，进行如下操作，使用redis-trib.rb添加节点。在已有集群服务器（192.168.0.7）上执行。

注意：add-node的使用方法为new\_host:new\_port existing\_host:existing\_port，前面是新添加的节点信息，后面是已存在的节点信息。

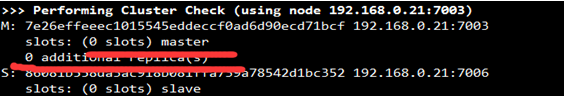
# ./redis-trib.rb add-node 192.168.0.21:7006 192.168.0.7:7001

在线添加时，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（添加成功时出现的界面并没有下面的红字）。



node新节点没有包含任何数据， 因为它没有包含任何slot，会自动成为一个主节点， 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，同时因为没有slots而不参加选举和failover。选择其中一个节点，检查集群状态，发现刚添加的节点已经在集群中了，角色是master，后面要通过shard命令分配slot。

# ./redis-trib.rb check 192.168.0.21:7006



加入新的节点后，需要将其他的节点中的hash slot移动到新的节点中，以达到负载均衡的效果，其中要指出的是当主节点过多，那么就不能按照下面的基数进行分了，应按照主节点的基数进行平均分配，可以在线reshard slot来解决，只是需要reshard的master节点是老节点的ip和端口。

#./redis-trib.rb reshard 192.168.0.7:7000

根据提示选择要迁移的slot数量(ps:这里选500，最好是16384的约数。如4096，或各个主节点平均分配)

How many slots do you want to move (from 1 to 16384)? 500

选择要接受这些slot的node-id

What is the receiving node ID? f51e26b5d5ff74f85341f06f28f125b7254e61bf （此id为新增节点的id）

选择slot来源all表示从所有的master重新分配，或数据要提取slot的master节点id，最后用done结束

Please enter all the source node IDs.

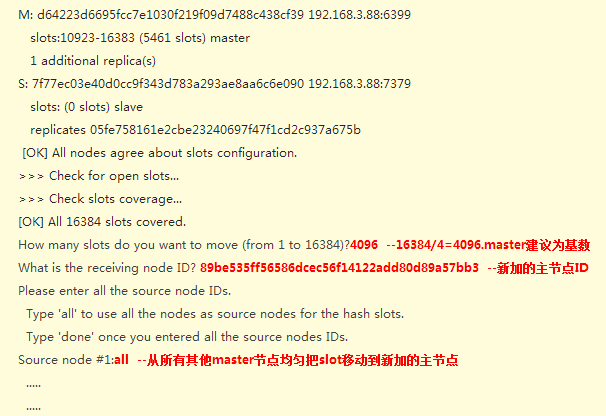
  Type 'all' to use all the nodes as source nodes for the hash slots.

  Type 'done' once you entered all the source nodes IDs.

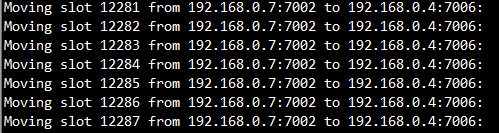
Source node #1:all

打印被移动的slot后，输入yes开始移动slot以及对应的数据

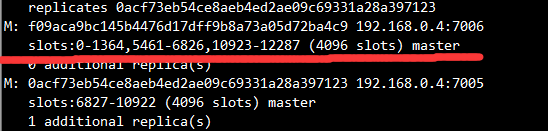
#Do you want to proceed with the proposed reshard plan (yes/no)? yes



然后出现下图滚动：



新被分配slot的节点如下图显示 ：



**数据迁移**

如果迁移的 slots 数量多，应当设置 redis-trib.rb 的超时参数--timeout 值大一点。否则，迁移过程中易遇到超时错误“[ERR] Calling MIGRATE: IOERR error or timeout reading to target instance”，导致只完成部分，可能会造成数据丢失。

如果在迁移过程中遇到下面这样的错误：

>>> Check for open slots...

[WARNING] Node 192.168.0.3:6379 has slots in importing state (5461). [WARNING] Node 192.168.0.5:6380 has slots in migrating state (5461). [WARNING] The following slots are open: 5461

可以考虑使用命令“redis-trib.rb fix 192.168.34.9:7000”尝试修复。需要显示有节点处于migrating或 importing 状态，可以登录到相应的节点，使用命令“cluster setslot 5461 stable” 修改，参数5461为显示问题的 slot 的 ID。

**迁移 slosts**

示例：将 slot8 从节点 A 迁移到节点 B，有如下两种方法：

在节点 B 上执行：CLUSTER SETSLOT 8 IMPORTING A或

在节点 A 上执行：CLUSTER SETSLOT 8 MIGRATING B

主节点添完后，需要给该主节点添加一个slave节点，添加过程和主节点一致，完成后需要进行相应的设置。

# ./redis-trib.rb add-node 192.168.0.21:7007 192.168.0.21:7003

连接到要添加的slave数据库中，执行replicate操作，后面的ID为Master 192.168.0.21:7003的ID，通过redis-trib.rb check可以看到。

#./redis-cli -c -h 192.168.0.21 -p 7007

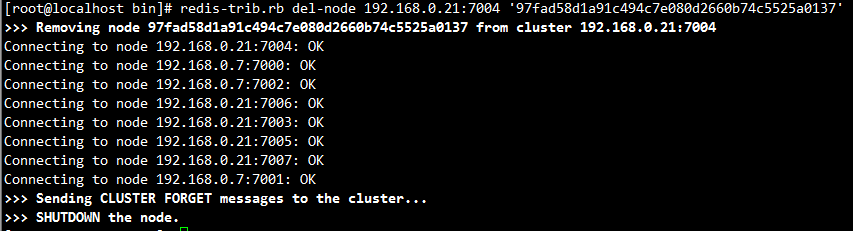
192.168.0.21:7007> cluster replicate 7e26effeeec1015545eddeccf0ad6d90ecd71bcf

根据check结果，可以看到新添加的slave已经成功和Master建立联系。

**删除一个slave节点**

#./redis-trib  del-node  ip:port  '<node-id>'

#./redis-trib.rb  del-node  192.168.0.21:7007  'c7ee2fca17cb79fe3c9822ced1d4f6c5e169e378'



删除一个master节点

删除master节点之前，首先要将此master节点上的数据迁移，然后将slave节点删除，再使用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迁移到一个主节点上)。

把192.168.0.21:7003当前master的slots迁移到192.168.0.21:7006上

#./redis-trib.rb  reshard  192.168.0.21:7006

根据提示选择要迁移的slot数量(ps:这里选择500)

How many slots do you want to move (from 1 to 16384)? 500(被删除master的所有slot数量)

选择要接受这些slot的node-id(192.168.0.21:7006)

What is the receiving node ID? c4a31c852f81686f6ed8bcd6d1b13accdc947fd2 (ps: 192.168.0.21:7006的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:f51e26b5d5ff74f85341f06f28f125b7254e61bf(被删除master的node-id)

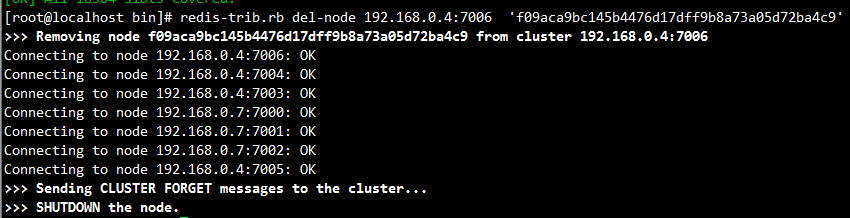
Source node #2:done

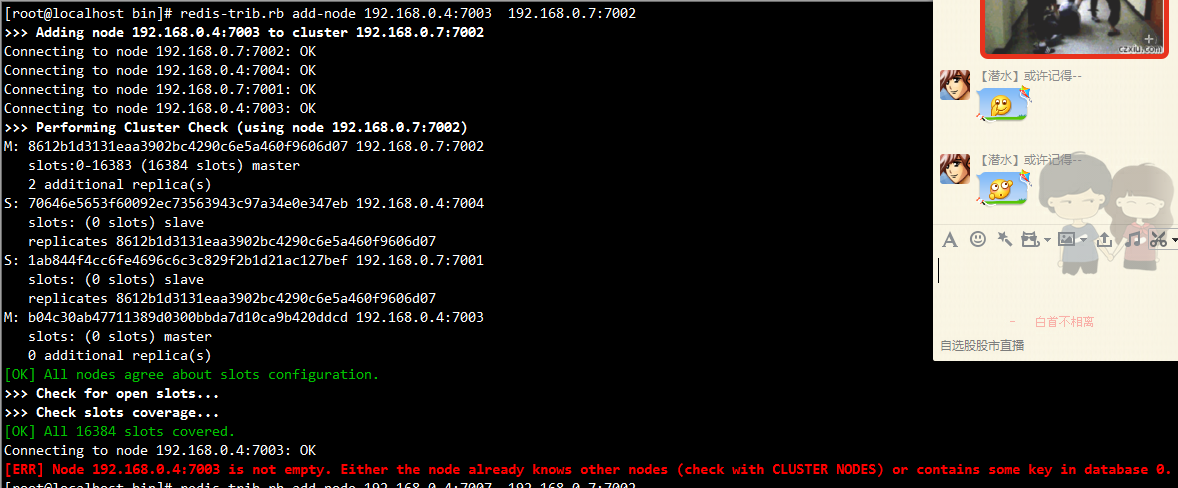
打印被移动的slot后，输入yes开始移动slot以及对应的数据

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

删除空master节点

#./redis-trib.rb  del-node  192.168.0.21:7003  'f51e26b5d5ff74f85341f06f28f125b7254e61bf'





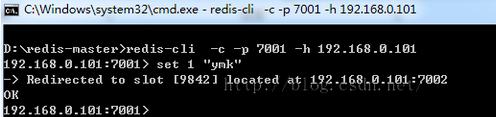
**注意：当删除完主从的一对节点后，再次重新添加这一对主从节点，添加过程虽然有上面的红字提示，但是并不影响添加，当使用check命令后查看到，这对节点已经添加上了。**

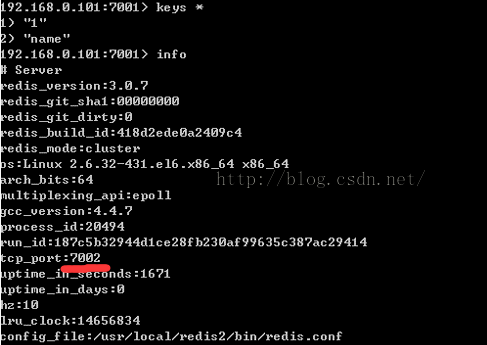
当有增减节点时，可以使用命令：

#./redis-trib.rb rebalance 192.168.0.31:6379 --auto-weights 做一次均衡，简单点可以只指定两个参数，“192.168.0.31:6379”为集群中已知的任何一个节点，参数“--auto-weights”表示自动权重。

## redis-cli客户端使用事项

使用redis-cli连接cluster时要用： redis-cli –c –h –p这样的格式在客户端连上任意一个redis节点，set一个值后，该值不一定存到该节点，而是自动转向至由Redis集群选举出来的一个节点并存入值，此时客户端的物理连接也会被指向该节点。举例：



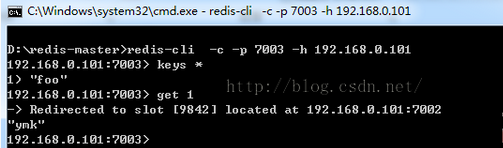


此时redis-cli连接已经被重定向至了7002，而不是在7001下，虽然显示符>前显示的还是7001。

使用redis-cli连接cluster中任意一个节点如：



可以看到7003节点内无 key=1的元素，但是集群内是有key=1的元素的，于是：



reds-cli会动态从集群中查到含有key=1的节点，把客户端连接重定向至该节点并调用该key所绑的value。

**查看集群信息**

对应的 redis 命令为：cluster info，示例：

#127.0.0.1:6381> cluster info

cluster\_state:ok 所有 slots 正常则显示为 OK，否则为 error

cluster\_slots\_assigned:16384 多少 slots 被分配了，即多少被 master 管理了，16384 为全部 slots cluster\_slots\_ok:16384 有多少 slots 是正常的

cluster\_slots\_pfail:0 有多少 slots 可能处于异常状态，处于这个状态并不表示有问题，仍能继续提供服务 cluster\_slots\_fail:0 有多少 slots 处于异常状态，需要修复才能服务

cluster\_known\_nodes:10 集群中的节点数

cluster\_size:3 集群中 master 个数

cluster\_current\_epoch:11 本地的当前时间变量，用于故障切换时生成独一无二的增量版本号

cluster\_stats\_messages\_sent:4049 通过集群消息总线发送的消息总数

cluster\_stats\_messages\_received:4051 通过集群消息总线收到的消息总数

**redis cluster 客户端(Jedis)**

jedis（java cluster client）

官网：https://github.com/xetorthio/jedis

r3c（C++ cluster client）

官网：https://github.com/eyjian/r3c

<span style="color: #333333; font-family: Arial, sans-serif;"><span style="color: #333333; font-family: Arial, sans-serif;"> private static BinaryJedisCluster jc;

  static {

       //只给集群里一个实例就可以

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

        jedisClusterNodes.add(new HostAndPort("10.10.34.14", 6380));

        jedisClusterNodes.add(new HostAndPort("10.10.34.14", 6381));

        jedisClusterNodes.add(new HostAndPort("10.10.34.14", 6382));

        jedisClusterNodes.add(new HostAndPort("10.10.34.14", 6383));

        jedisClusterNodes.add(new HostAndPort("10.10.34.14", 6384));

        jedisClusterNodes.add(new HostAndPort("10.10.34.14", 7380));

        jedisClusterNodes.add(new HostAndPort("10.10.34.14", 7381));

        jedisClusterNodes.add(new HostAndPort("10.10.34.14", 7382));

        jedisClusterNodes.add(new HostAndPort("10.10.34.14", 7383));

        jedisClusterNodes.add(new HostAndPort("10.10.34.14", 7384));

        jc = new BinaryJedisCluster(jedisClusterNodes);

    }

@Test

    public void testBenchRedisSet() throws Exception {

        final Stopwatch stopwatch = new Stopwatch();

        List list = buildBlogVideos();

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

            String key = "key:" + i;

            stopwatch.start();

            byte[] bytes1 = protostuffSerializer.serialize(list);

            jc.setex(key, 60 \* 60, bytes1);

            stopwatch.stop();

        }

        System.out.println("time=" + stopwatch.toString());

    }</span></span>

**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(mget,mset等), 在slave执行readonly命令后，可执行读操作，但当keys集合对应的slot相同时支持mget操作，不支持多[数据库](http://www.2cto.com/database/" \t "_blank)，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)。

4)JedisCluster 没有针对byte[]的API，需要自己扩展。

**连接redis集群**

接下来就是要解决Tomcat的Session共享问题，使用开源项目：

https://github.com/ran-jit/TomcatRedisClusterEnabledSessionManager

注意，因为使用的是Redis3.2集群，相应的插件一定要支持Redis3.2集群。

下载：https://github.com/ran-jit/TomcatRedisClusterEnabledSessionManager/archive/master.zip

解压找到lib目录中的：

jedis-3.0.0-SNAPSHOT.jar

commons-pool2-2.2.jar

commons-logging-1.1.jar

并将这3个jar包上传到Tomcat7中的lib 目录。

下载：https://github.com/ran-jit/TomcatRedisClusterEnabledSessionManager/releases/download/1.0/TomcatRedisSessionManager-1.0.zip（当前是1.0，如果出最新版，要同步更新到最新版）。

解压后得到：TomcatRedisSessionManager-1.0.jar 和redis.properties。



将TomcatRedisSessionManager-1.0.jar 上传到Tomcat7中的lib 目录，添加Tomcat的环境变量（可选）。

CATALINA\_HOME=/home/wusc/edu/web/boss-tomcat

配置redis.properties，集群中多节点间的IP和端口用逗号隔开，打开Redis集群模式。

# redis hosts ex: 127.0.0.1:6379, 127.0.0.2:6379, 127.0.0.2:6380, ....

redis.hosts=192.168.1.111:7111,192.168.1.112:7112,192.168.1.113:7113,192.168.1.114:7114

# Redis Password

redis.password=

# set true to enable redis cluster mode

redis.cluster.enabled=true

配置好之后把redis.properties 上传到Tomcat7的conf目录中。

注意：此插件支持单节点的Redis，也支持Redis集群，只需要在redis.properties中配置则可。

在Tomcat7中的conf/context.xml 中增加如下两行配置：

<Valve className="com.r.tomcat.session.management.commons.SessionHandlerValve" />

<Manager className="com.r.tomcat.session.management.redis.RedisSessionManager" />

在Tomcat的conf/web.xml中核对确认Tomcat的Session超时时间，默认为30分钟。

<!--==================== Default Session Configuration ================= -->

<!--You can set the default session timeout (in minutes) for all newly -->

<!--created sessions by modifying the value below. -->

<session-config>

<session-timeout>30</session-timeout>

</session-config>

**Redis的监控软件：redis-client，redis-stat，redis desktop manager等**

**redis-stat安装**

redis-stat是一个用ruby写成的监控redis的程序，基于info命令获取信息，不会像基于[MONITOR](http://redis.io/commands/monitor)命令的监控工具一样，对Redis本身有性能上的影响。一方面可以像vmstat一样，把状态信息从终端输出，另一方面也可以通过基于嵌入式的web server使用dashboard页面查看状态信息。

安装依赖

#yum -y install rubygems ruby-devel openssl openssl-devel

安装ruby，redis-stat要求ruby版本为2.2.2以上，所以只能编译安装。

#cd /usr/local/src

#wget https://ruby.taobao.org/mirrors/ruby/2.2/ruby-2.2.4.tar.gz

#tar -xf ruby-2.2.4.tar.gz

#cd ruby-2.2.4

#./configure --prefix=/usr/local/ruby --disable-install-rdoc

#make

#make install

#vim /etc/profile

export PATH=$PATH:/usr/local/ruby/bin

#source /etc/profile

#gem sources --add <https://gems.ruby-china.org/> --remove https://rubygems.org/

如果提示：-bash: /usr/bin/gem: /usr/bin/ruby: bad interpreter: No such file or directory

将/usr/local/ruby/bin/下的gem，ruby两个脚本复制并替换到/usr/bin/下，再执行上面的命令。

#gem sources -l

安装redis-stat

#yum -y install readline-devel

#gem install rb-readline

#gem install redis-stat -V

如果出现如下错误：

ERROR:  Error installing redis-stat:  
redis-stat requires daemons (~> 1.1.9, runtime)

装redis-stat同时会安装daemons-1.1.9，如果系统已安装daemons其他版本，需再次卸载。

#gem uninstall daemons

接下来启动redis-stat，如果启动过程中报如下错误：

Faraday: you may want to install system\_timer for reliable timeouts

安装SystemTimer

# gem install SystemTimer

redis-stat命令行模式：

usage: redis-stat [HOST[:PORT] ...] [INTERVAL [COUNT]]

-a, --auth=PASSWORD 设置密码

-v, --verbose 显示更多信息

--style=STYLE 输出编码类型: unicode|ascii

--no-color 取消ANSI颜色编码

--csv=OUTPUT\_CSV\_FILE\_PATH 以CSV格式存储结果

--es=ELASTICSEARCH\_URL 把结果发送到 ElasticSearch: [http://]HOST[:PORT][/INDEX]

--server[=PORT] 运行redis-stat的web server (默认端口号: 63790)

--daemon 使得redis-stat成为进程，必须使用 --server 选项

--version 显示版本号

--help 显示帮助信息

如果是127.0.0.1则直接./redis-stat 1 10

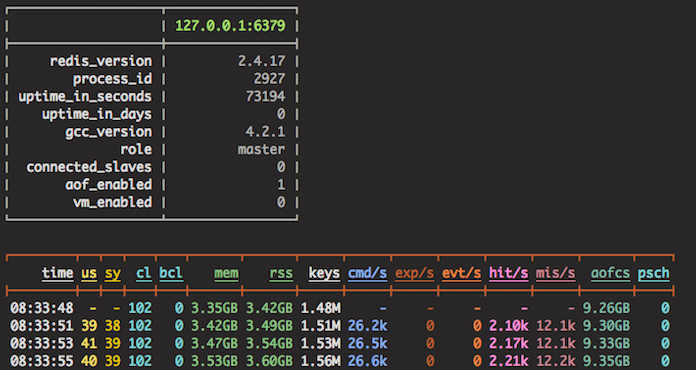
redis-stat 1 10

redis-stat --verbose

如果是多个redis节点或者集群要加上ip和端口号：

redis-stat 192.168.34.6:7003 192.168.34.6:7004 192.168.34.6:7005 192.168.34.9:7000 192.168.34.9:7001 192.168.34.9:7002 30 --verbose

redis-stat 192.168.34.6:7003 10 --csv=/tmp/output.csv --verbose



redis-stat web模式：

当设置--server选项之后，redis-stat会在后台启动一个嵌入式的web server(默认端口号：63790)，可以在浏览器中监控Redis。因为redis-stat通过[Server-sent events](http://www.w3.org/TR/eventsource/" \t "_blank)间隔性的推送更新，可用浏览器查看此页面。

redis-stat 192.168.34.6:7003(实例ip和端口号) --server

redis-stat 192.168.34.6:7003(实例ip和端口号) --verbose --server=8080 5

redis-stat 192.168.34.6:7003(实例ip和端口号) --server=9000 --daemon

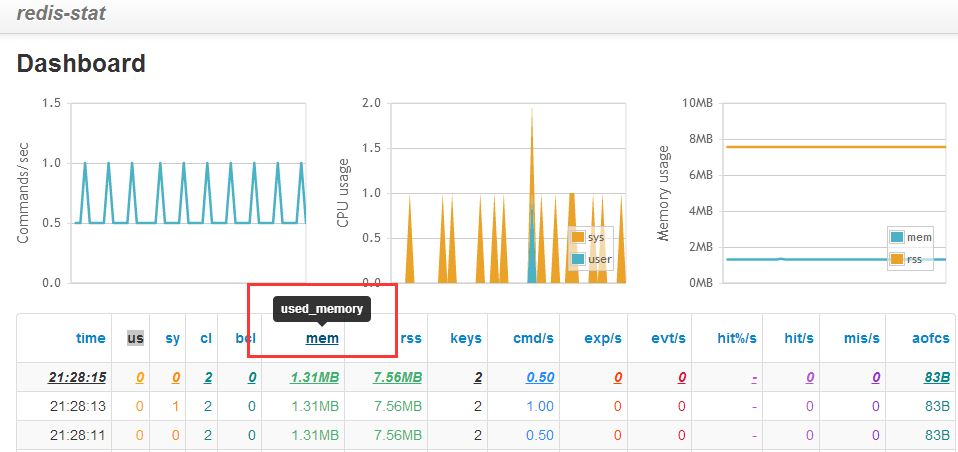
killall -9 redis-stat-daemon

运行web模式

#./redis-stat 192.168.34.6:7003(实例ip和端口号) --server=8080 5 --daemon

如：<http://192.168.34.6:8080/>

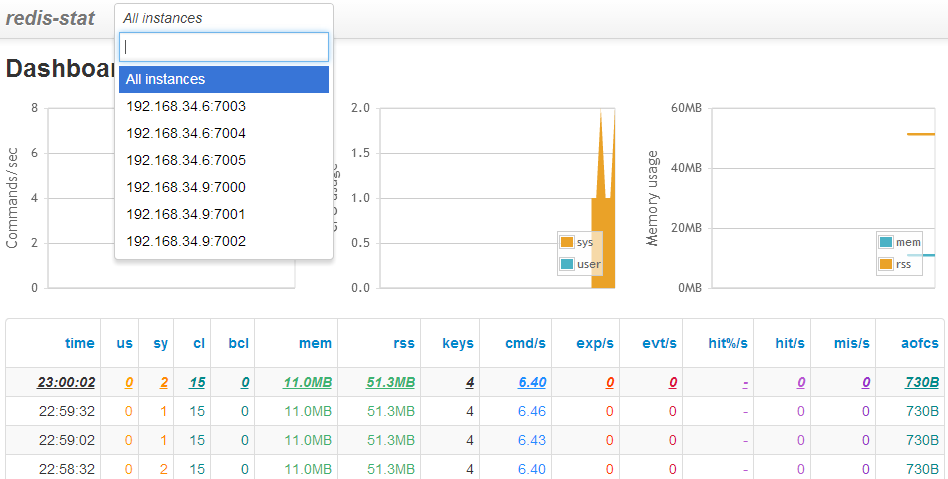
鼠标指针放在字段上面就会显示字段含义，如下图：



如果是多个节点或者集群可以输入多个ip和端口号，并指定web访问端口：

#./redis-stat 192.168.34.6:7003 192.168.34.6:7004 192.168.34.6:7005 192.168.34.9:7000 192.168.34.9:7001 192.168.34.9:7002 --server=9000 30

然后在浏览器中输入执行这个命令的物理机的ip和端口号，如上面的命令是在192.168.34.9上执行的，那么在浏览器中输入 <http://192.168.34.9:9000>即可出现下图：



将光标点入上面的框中，即可查询各个节点的信息。

安装redis-dump

#gem install redis-dump -V

redis-dump导出数据

#redis-dump -u 127.0.0.1:6379 >test.json //导出数据

redis-load还原数据

#< test.json redis-load //导入数据

附录

卸载gem

#gem uninstall (gem-name)

一般先用gem list查看一下，找到要卸载的gem的名字。

查看远程可用软件

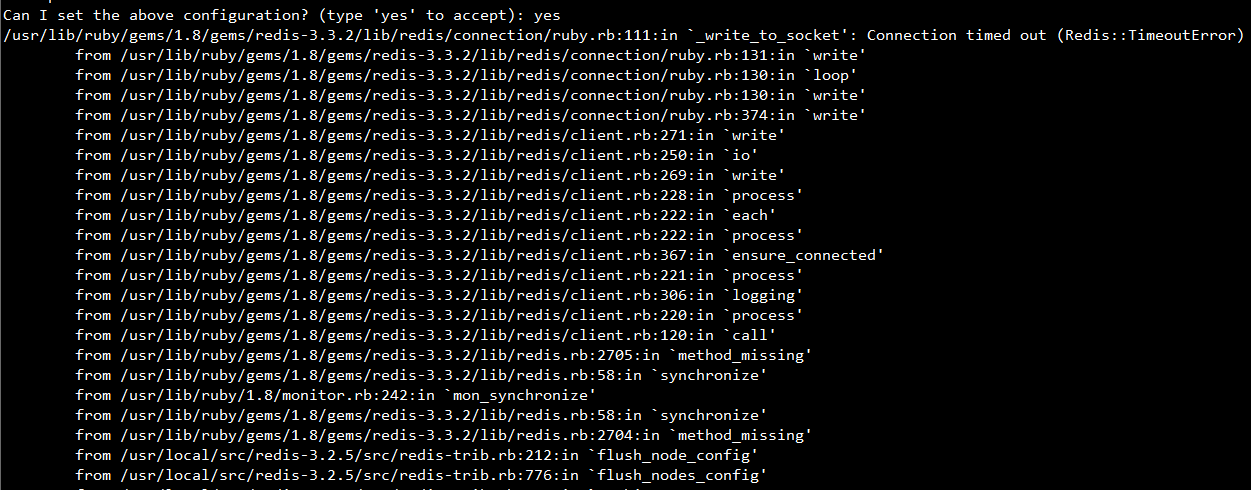
gem list -h

gem list --remote 软件名

如：查找所有redis开头的软件

#gem list --remote "^redis-.\*"

创建redis-3.2.5集群出现以下报错：



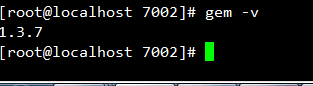
这是由于gem的版本低或者gem版本本身的问题造成的。

解决办法：

在哪台机子上创建集群，就在哪台机子上执行下面的步骤。

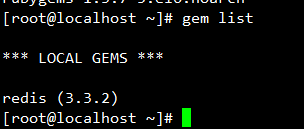
查看gem版本

#gem -v



然后使用gem list查看redis接口软件版本

#gem list

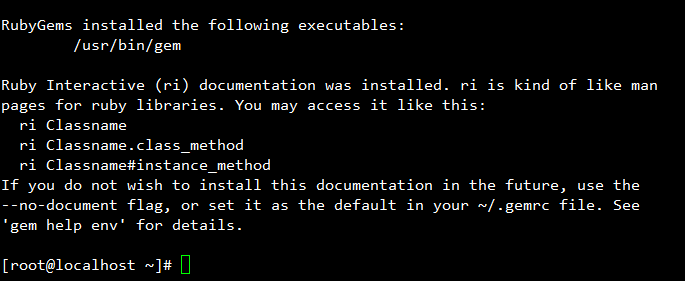


卸载gem

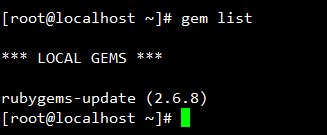
#gem  uninstall  redis  --version  3.3.2

然后更新gem

#gem update --system



#gem list



然后执行安装redis接口软件

#gem install redis --version 3.0.0

然后删除各个节点目录下面的node-\*.conf，appendonly.aof，dump.rdb等文件，重新启动实例，重新创建集群即可。如果出现下面的报错，是因为这些文件没有删除导致的，删除即可：

