Redis3.2.6 集群环境搭建

技术文档

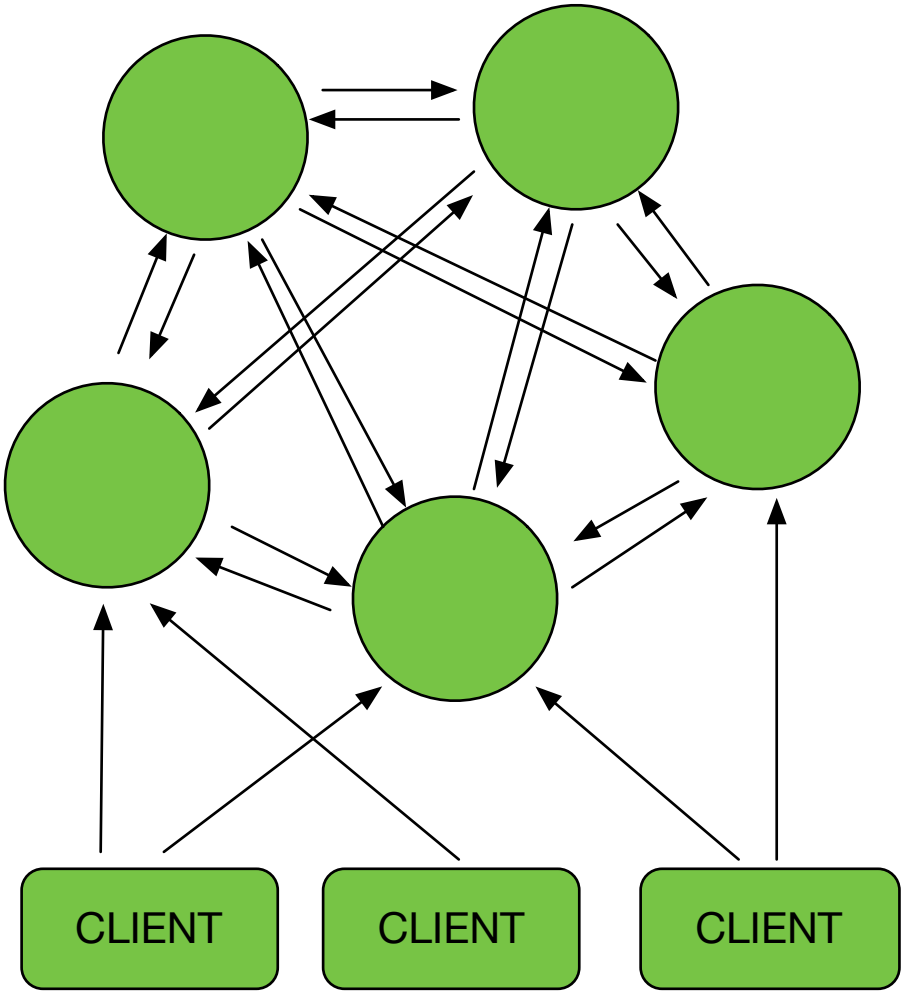
（版本：V1.0.0）

版本	日期	说明	作者
V1.0.0	2016/12/27	创建	Simon Hoo

目　录

一． Redis 集群方案

Redis 集群方案采用 Redis3.0 支持的 redis-cluster 集群架构：

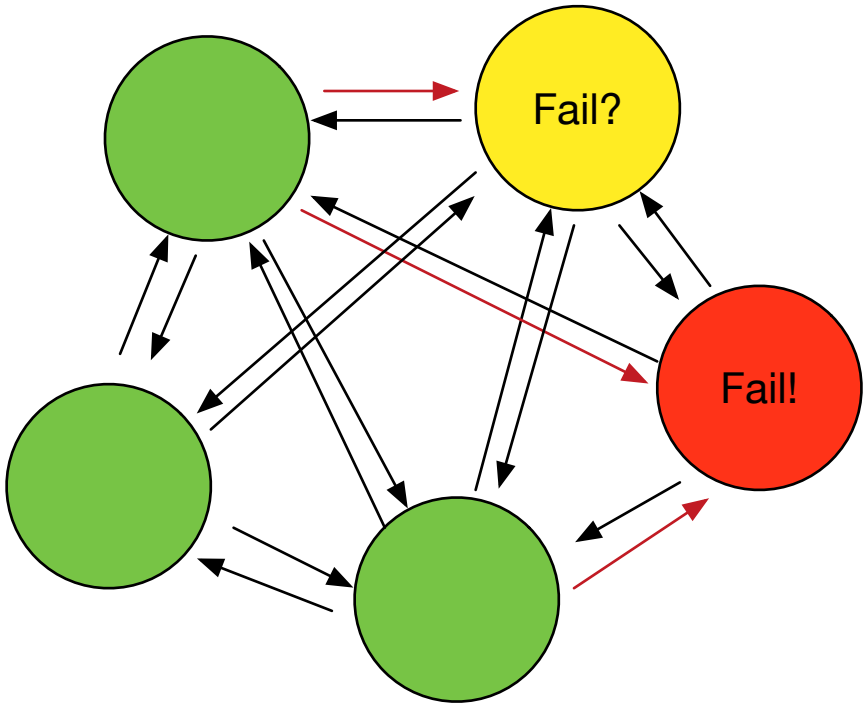


架构方案：

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的节点，连接集群中任何一个可用的节点即可。

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。

容错机制：



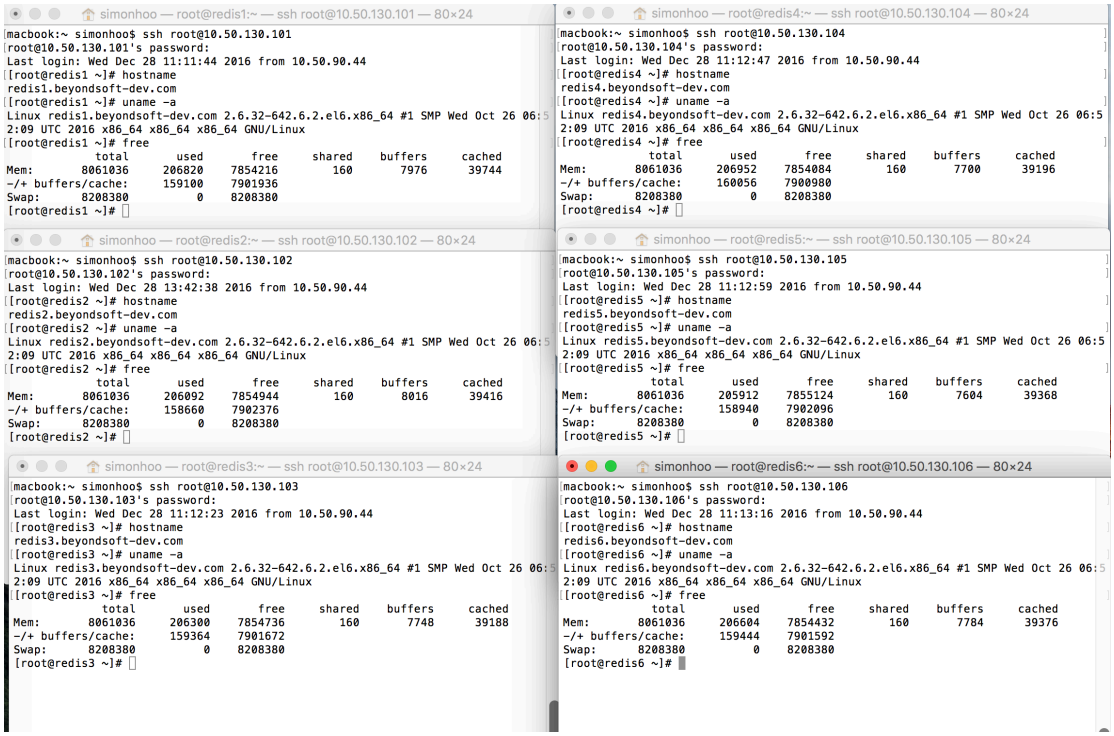
1. redis-cluster 的容错机制是通过选举产生 master,选择过程是集群中所有的 master 参与，如果半数以上的 master 节点与 master 节点的通信超时（cluster-node-timeout）,则认为当前 master 节点挂掉。
2. 什么时候整个集群不可用（cluster_state:fail）,当集群不可用时，所有对集群的操作均不可用，收到 ERRORQ 错误。

二． 环境准备

2.1 服务器清单

序号	IP 地址	HOSTNAME	配置	备注
1	10.50.130.101	redis1.beyondsoft-dev.com	CPU:2*2 内存：8GB	

2	10.50.130.102	redis2.beyondsoft-dev.com	CPU:2*2 内存：8GB	
3	10.50.130.103	redis3.beyondsoft-dev.com	CPU:2*2 内存：8GB	
4	10.50.130.104	redis4.beyondsoft-dev.com	CPU:2*2 内存：8GB	
5	10.50.130.105	redis5.beyondsoft-dev.com	CPU:2*2 内存：8GB	
6	10.50.130.106	redis6.beyondsoft-dev.com	CPU:2*2 内存：8GB	



注：本文档中所演示的操作系统为 CentOS 6.8 64 位操作系统。

2.2 服务器设置

Step1. 将 6379 端口在防火墙看开启：

[root@redis1 ~]# vi /etc/sysconfig/iptables

-A INPUT -m state --state NEW -m tcp -p tcp --dport 6379 -j ACCEPT

```
simonhoo — root@redis1:/usr/local/redis-3.2.6/src — ssh root@10.50.130.101...
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 6379 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
~
~
~
~
~
~
~
~
~
-- INSERT --
```

注：必须加在 REJECT 前面。

Step 2. 修改 Selinux 参数：

[root@redis1 ~]# vi /etc/selinux/config

SELINUX=disabled

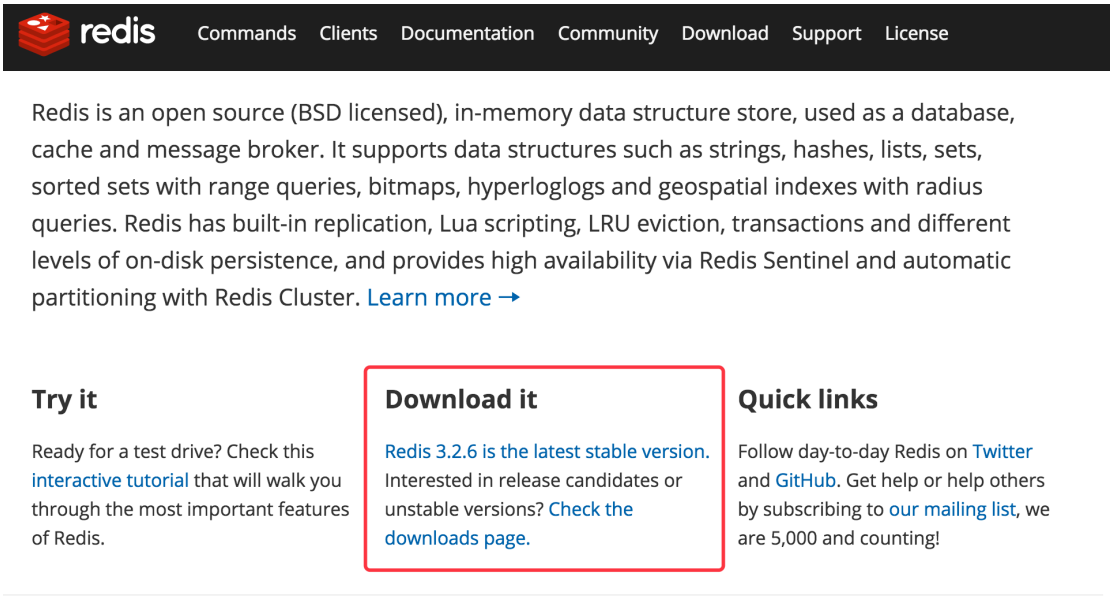
```
simonhoo — root@redis1:/usr/local/redis-3.2.6/src — ssh root@10.50.130.101...
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
#SELINUX=enforcing
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted


~
~
~
~
~
~
~
~
~
"/etc/selinux/config" 14L, 476C
```

注：分别在其它几个节点上同样配置防火墙和 SELINUX。

三． 软件准备

3.1 下载 Redis3.2.6

https://redis.io/



注：本文中演示的 Redis 版本为 Redis 3.2.6，通用于 Redis 3.x 的所有版本，如果 Redis 版本为 2.x 的，则有所不同。

3.2 将软件上传到服务器

Step 1. 分别在 6 个服务器上创建 software 目录，用于存放待安装的软件：

```
[root@redis1 ~]# mkdir /root/software
[root@redis2 ~]# mkdir /root/software
[root@redis3 ~]# mkdir /root/software
[root@redis4 ~]# mkdir /root/software
[root@redis5 ~]# mkdir /root/software
[root@redis6 ~]# mkdir /root/software
```

Step 2. 将本地下载好的软件 SCP 到服务器：

四 . 安装 Redis

4.1 安装系统组件

Step 1. 安装 gcc
[root@redis1 ~]# yum install –y gcc-c++

Step 2. 安装 tcl
[root@redis1 ~]# yum install –y tcl

4.2 安装 Redis

Step 1. 解压 Redis 到/usr/local 目录下
[root@redis1 ~]# tar -zxvf /root/software/redis-3.2.6.tar.gz -C /usr/local/

Step 2. 在解压后的目录中进行 make 和 make test
[root@redis1 ~]# cd /usr/local/redis-3.2.6
[root@redis1 redis-3.2.6]# make
[root@redis1 redis-3.2.6]# make test

注：要检查 make 和 make test 的结果是否都正确，如果报错，针对性检查并安装系统缺少的组件。

```
simonhoo — root@redis1:/usr/local/redis-3.2.6 — ssh root@10.50.130.101 — 8...
  104 seconds - integration/replication-psync
  3 seconds - integration/aof
  1 seconds - integration/rdb
  1 seconds - integration/convert-zipmap-hash-on-load
  0 seconds - integration/logging
  1 seconds - unit/pubsub
  1 seconds - unit/slowlog
  6 seconds - unit/scripting
  12 seconds - unit/maxmemory
  0 seconds - unit/introspection
  7 seconds - unit/introspection-2
  1 seconds - unit/limits
  56 seconds - unit/obuf-limits
  5 seconds - unit/bitops
  2 seconds - unit/bitfield
  36 seconds - unit/geo
  6 seconds - unit/memefficiency
  30 seconds - unit/hyperloglog

\o/ All tests passed without errors!

Cleanup: may take some time... OK
make[1]: Leaving directory `/usr/local/redis-3.2.6/src'
[root@redis1 redis-3.2.6]#
```

Step 3. 复制 redis-server 和 redis-cli 到/usr/local/bin 目录下：

[root@redis1 redis-3.2.6]# cd src

```
simonhoo — root@redis1:/usr/local/redis-3.2.6/src — ssh root@10.50.130.101...
-rw-rw-r--. 1 root root    5003 12月  6 16:38 rdb.h
-rw-r--r--. 1 root root  143744 12月 28 14:09 rdb.o
-rw-rw-r--. 1 root root    2276 12月  6 16:38 redisassert.h
-rwxr-xr-x. 1 root root 5580303 12月 28 14:09 redis-benchmark
-rw-rw-r--. 1 root root   29329 12月  6 16:38 redis-benchmark.c
-rw-r--r--. 1 root root   69320 12月 28 14:09 redis-benchmark.o
-rwxr-xr-x. 1 root root   22177 12月 28 14:09 redis-check-aof
-rw-rw-r--. 1 root root    6328 12月  6 16:38 redis-check-aof.c
-rw-r--r--. 1 root root   20680 12月 28 14:09 redis-check-aof.o
-rwxr-xr-x. 1 root root 7826320 12月 28 14:09 redis-check-rdb
-rw-rw-r--. 1 root root   12789 12月  6 16:38 redis-check-rdb.c
-rw-r--r--. 1 root root   50776 12月 28 14:09 redis-check-rdb.o
-rwxr-xr-x. 1 root root 5709012 12月 28 14:09 redis-cli
-rw-rw-r--. 1 root root   90077 12月  6 16:38 redis-cli.c
-rw-r--r--. 1 root root  222616 12月 28 14:09 redis-cli.o
lrwxrwxrwx. 1 root root 7826320 12月 28 14:09 redis-sentinel
-rwxr-xr-x. 1 root root 7826320 12月 28 14:09 redis-server
-rwxrwxr-x. 1 root root   60852 12月  6 16:38 redis-trib.rb
-rw-rw-r--. 1 root root    2163 12月 28 14:08 release.c
-rw-r--r--. 1 root root     124 12月 28 14:08 release.h
-rw-r--r--. 1 root root    4560 12月 28 14:09 release.o
-rw-rw-r--. 1 root root   96726 12月  6 16:38 replication.c
-rw-r--r--. 1 root root  141336 12月 28 14:09 replication.o
-rw-rw-r--. 1 root root   12104 12月  6 16:38 rio.c
```

[root@redis1 src]# cp redis-server /usr/local/bin/

[root@redis1 src]# cp redis-cli /usr/local/bin/

```
[root@redis1 src]# cd /usr/local/bin/
[root@redis1 bin]# ll
总用量 13220
-rwxr-xr-x. 1 root root 5709012 12月  28 14:59 redis-cli
-rwxr-xr-x. 1 root root 7826320 12月  28 14:59 redis-server
[root@redis1 bin]#
```

Step 3. 验证 Redis 安装是否成功：

[root@redis1 ~]# redis-server

```
simonhoo — root@redis1:~ — ssh root@10.50.130.101 — 80×42
[root@redis1 ~]# redis-server &
[1] 6858
[root@redis1 ~]# 6858:C 28 Dec 15:02:13.695 # Warning: no config file specified,
 using the default config. In order to specify a config file use redis-server /p
ath/to/redis.conf
6858:M 28 Dec 15:02:13.697 * Increased maximum number of open files to 10032 (it
 was originally set to 1024).
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 3.2.6 (00000000/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in standalone mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 6379
      |    `-._   `._    /     _.-'    |     PID: 6858
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

6858:M 28 Dec 15:02:13.711 # WARNING: The TCP backlog setting of 511 cannot be e
nforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
6858:M 28 Dec 15:02:13.711 # Server started, Redis version 3.2.6
6858:M 28 Dec 15:02:13.711 # WARNING overcommit_memory is set to 0! Background s
ave may fail under low memory condition. To fix this issue add 'vm.overcommit_me
mory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.over
commit_memory=1' for this to take effect.
6858:M 28 Dec 15:02:13.712 # WARNING you have Transparent Huge Pages (THP) suppo
rt enabled in your kernel. This will create latency and memory usage issues with
 Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transpare
nt_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retai
n the setting after a reboot. Redis must be restarted after THP is disabled.
6858:M 28 Dec 15:02:13.712 * DB loaded from disk: 0.000 seconds
6858:M 28 Dec 15:02:13.712 * The server is now ready to accept connections on po
rt 6379

[root@redis1 ~]#
```

[root@redis1 ~]# redis-cli

```
[root@redis1 ~]# redis-cli
127.0.0.1:6379> set "name" "simon"
OK
127.0.0.1:6379> get "name"
"simon"
127.0.0.1:6379>
```

注：安装其它 5 台服务器，重复本章节的 4.1~4.2。

五 . 配置集群模式

5.1 配置 redis.conf

Step 1. 配置 redis.conf
[root@redis1 ~]# mkdir /etc/redis
[root@redis1 ~]# cd /etc/redis
[root@redis ~]# vi redis.conf

port 6379
daemonize yes
cluster-enabled yes
cluster-config-file /etc/redis/nodes.conf
cluster-node-timeout 5000
appendonly yes
requirepass Ab123456

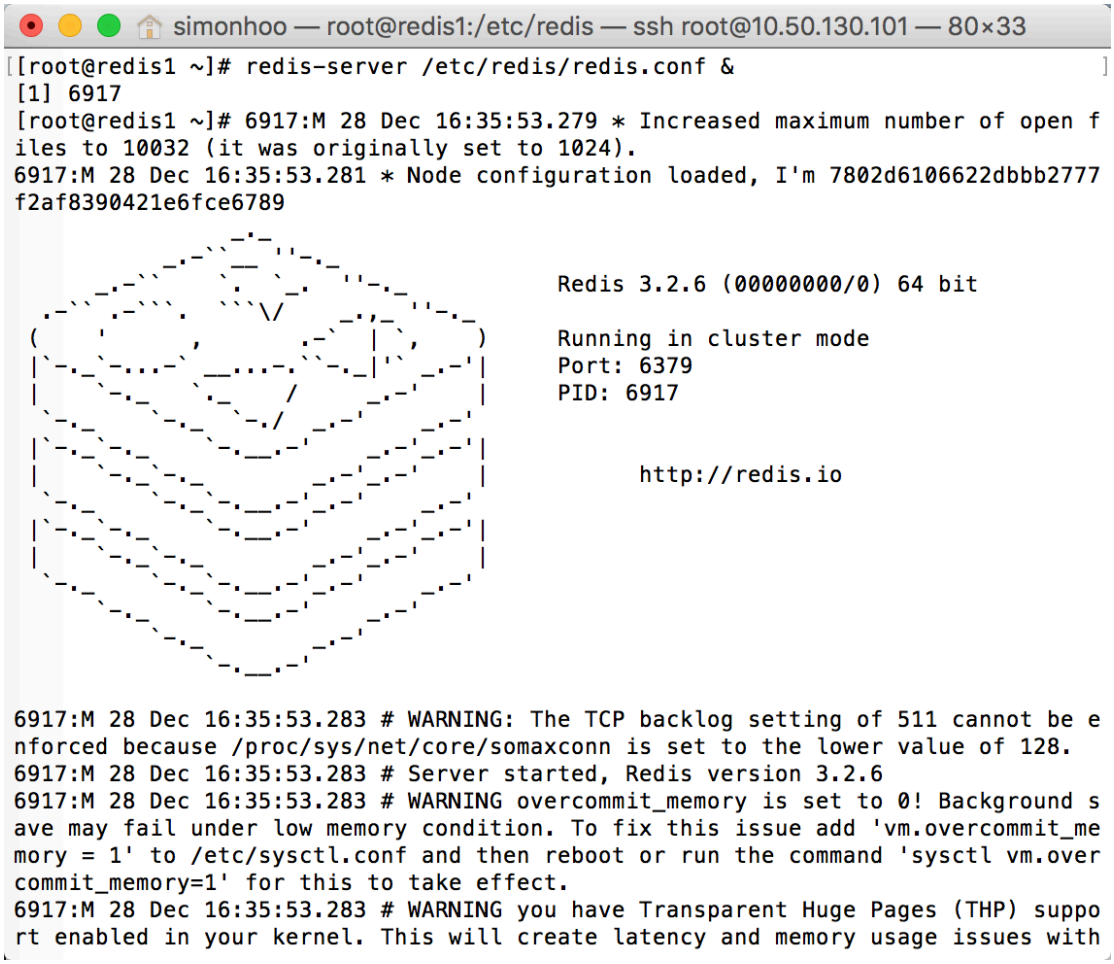
注 1：cluster-node-timeout 是集群中各节点相互通讯时，允许“失联”的最大毫秒数，本演示中配置的为 5 秒，如果超过 5 秒某个节点没有向其它节点汇报成功，认为该节点挂了。

注 2：requirepass 是 Redis 访问密码，为了安全起见，该参数建议必须配置，从但客户端 Jedis 版本必须使用 2.8.x 以上的版本，否则需要通过扩展 JedisCluster 来实现对密码访问的支持。此外几个 Redis 节点的密码应该设置为相同的。

注 3：分别在其它几个节点上创建与上面相同的 redis.conf 文件，内容也相同。

注 4：重启/重建 Redis 集群时，必须删除去/etc/redis/nodes.conf 文件。

Step 2. 以次启动所有节点
[root@redis1 ~]# redis-server /etc/redis/redis.conf
[root@redis2 ~]# redis-server /etc/redis/redis.conf
[root@redis3 ~]# redis-server /etc/redis/redis.conf
[root@redis4 ~]# redis-server /etc/redis/redis.conf
[root@redis5 ~]# redis-server /etc/redis/redis.conf
[root@redis6 ~]# redis-server /etc/redis/redis.conf



5.2 安装 Redis 集群所需的 Ruby 工具

Step 1. 安装 Ruby 工具：

Redis 集群需要借助其它工具将相关节点加入到 Cluster 中，而这个工具是由 Redis 提供一个名为 redis-trib.rb 的 ruby 脚本，否则接下来创建 cluster 会失败。

[root@redis1 ~]# cd /usr/local/redis-3.2.6/src
[root@redis1 src]# yum install –y ruby
[root@redis1 src]# yum install -y rubygems
[root@redis1 src]# gem install redis --version 3.0.0
[root@redis1 src]# gem list

```
simonhoo — root@redis1:/usr/local/redis-3.2.6/src — ssh root@10.50.130.101...
[[root@redis1 src]# gem list

*** LOCAL GEMS ***

redis (3.0.0)
rubygems-update (2.6.8)
[root@redis1 src]#
```

Step 2. 设置 Ruby 连接 Redis 的密码：

[root@redis1 ~]# vi /usr/lib/ruby/gems/1.8/gems/redis-3.0.0/lib/redis/client.rb

:password => "Ab123456"

```
simonhoo — root@redis1:~ — ssh root@10.50.130.101 — 80×24
require "redis/errors"
require "socket"
require "cgi"

class Redis
  class Client

    DEFAULTS = {
      :url => lambda { ENV["REDIS_URL"] },
      :scheme => "redis",
      :host => "127.0.0.1",
      :port => 6379,
      :path => nil,
      :timeout => 5.0,
      :password => "Ab123456",
      :db => 0,
      :driver => nil,
      :id => nil,
      :tcp_keepalive => 0,
      :reconnect_attempts => 1,
      :inherit_socket => false
    }
```

注：分别在其它几个节点上用同样的方式安装好 Ruby 工具。

5.3 利用 redis-trib.rb 创建 Redis 集群

Step 1. 在 src 目录下运行以下脚本：

[root@redis1 ~]# cd /usr/local/redis-3.2.6/src

[root@redis1 src]# ./redis-trib.rb create --replicas 1 10.50.130.101:6379 10.50.130.102:6379 10.50.130.103:6379 10.50.130.104:6379 10.50.130.105:6379 10.50.130.106:6379

```
simonhoo — root@redis1:/usr/local/redis-3.2.6/src — ssh root@10.50.130.101...
[[root@redis1 src]# ./redis-trib.rb create --replicas 1 10.50.130.101:6379 10.50.]
130.102:6379 10.50.130.103:6379 10.50.130.104:6379 10.50.130.105:6379 10.50.130.
106:6379
>>> Creating cluster
>>> Performing hash slots allocation on 6 nodes...
Using 3 masters:
10.50.130.106:6379
10.50.130.105:6379
10.50.130.104:6379
Adding replica 10.50.130.103:6379 to 10.50.130.106:6379
Adding replica 10.50.130.102:6379 to 10.50.130.105:6379
Adding replica 10.50.130.101:6379 to 10.50.130.104:6379
S: 73d6fa1abf27eb6c9909cce940eb962fd08edb28 10.50.130.101:6379
   replicates dcbfbce127bc8828bc8cc7132f7487034cb1f5f4
S: c6417513ee1092dcb7ea60dbd0d480d4dc8c9495 10.50.130.102:6379
   replicates 135a779c5977c25610b7577f0b6072a93793a12e
S: e712c42a6de73e8c624743b44b1710d09fe18673 10.50.130.103:6379
   replicates 6a2a2a6c7e2b1641013382d7f96e062f29875027
M: dcbfbce127bc8828bc8cc7132f7487034cb1f5f4 10.50.130.104:6379
   slots:10923-16383 (5461 slots) master
M: 135a779c5977c25610b7577f0b6072a93793a12e 10.50.130.105:6379
   slots:5461-10922 (5462 slots) master
M: 6a2a2a6c7e2b1641013382d7f96e062f29875027 10.50.130.106:6379
   slots:0-5460 (5461 slots) master
[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 10.50.130.101:6379)
S: 73d6fa1abf27eb6c9909cce940eb962fd08edb28 10.50.130.101:6379
   slots: (0 slots) slave
   replicates dcbfbce127bc8828bc8cc7132f7487034cb1f5f4
S: c6417513ee1092dcb7ea60dbd0d480d4dc8c9495 10.50.130.102:6379
   slots: (0 slots) slave
   replicates 135a779c5977c25610b7577f0b6072a93793a12e
M: dcbfbce127bc8828bc8cc7132f7487034cb1f5f4 10.50.130.104:6379
   slots:10923-16383 (5461 slots) master
   1 additional replica(s)
M: 6a2a2a6c7e2b1641013382d7f96e062f29875027 10.50.130.106:6379
   slots:0-5460 (5461 slots) master
   1 additional replica(s)
M: 135a779c5977c25610b7577f0b6072a93793a12e 10.50.130.105:6379
   slots:5461-10922 (5462 slots) master
   1 additional replica(s)
S: e712c42a6de73e8c624743b44b1710d09fe18673 10.50.130.103:6379
   slots: (0 slots) slave
   replicates 6a2a2a6c7e2b1641013382d7f96e062f29875027
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
[root@redis1 src]#
```

注 1：只需在其中某个个节点执行以上脚本（本例在第一个节点执行）。

注 2：利用 redis-trib 创建 Cluster，只需要操作一次即可，假设系统关机、重启，把所有的节点全部关闭之后，下次重启后，即自动进入 Cluster 模式，不用现次执行 redis-trib.rb cteate 命令。

Step 2. 查看 Cluster 进程：

```
[root@redis1 ~]# ps -ef|grep redis
[root@redis2 ~]# ps -ef|grep redis
[root@redis3 ~]# ps -ef|grep redis
[root@redis4 ~]# ps -ef|grep redis
[root@redis5 ~]# ps -ef|grep redis
[root@redis6 ~]# ps -ef|grep redis
```

Step 3. 查看节点属性（Master/Slave）

```
[root@redis1 ~]# cd /usr/local/redis-3.2.6/src
[root@redis1 src]# ./redis-trib.rb check 10.50.130.101:6379
[root@redis1 src]# ./redis-trib.rb check 10.50.130.102:6379
[root@redis1 src]# ./redis-trib.rb check 10.50.130.103:6379
[root@redis1 src]# ./redis-trib.rb check 10.50.130.104:6379
[root@redis1 src]# ./redis-trib.rb check 10.50.130.105:6379
[root@redis1 src]# ./redis-trib.rb check 10.50.130.106:6379
```

Step 4. 查看节点/集群信息

redis-cli 客户端登录到任一个节点，查看:

六． Jedis 测试 Redis 集群

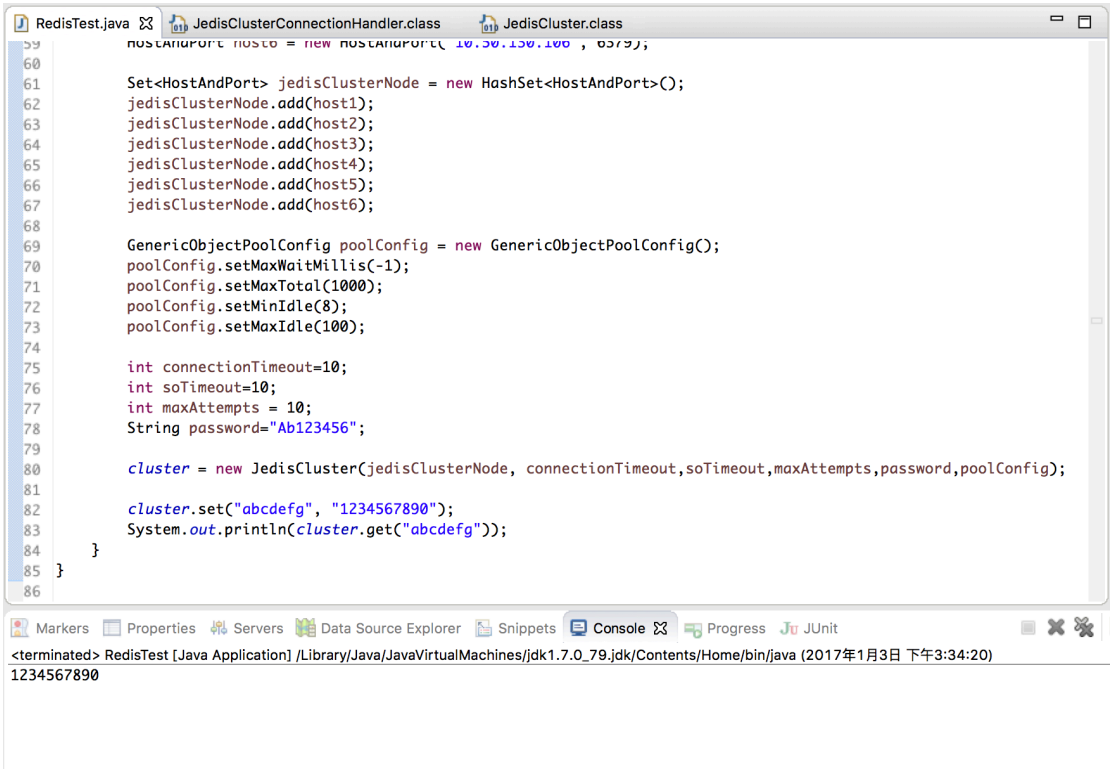
6.1 JAVA 测试代码

```
package com.cottsoft.redis;

import java.util.HashSet;
import java.util.Set;
```

```
import org.apache.commons.pool2.impl.GenericObjectPoolConfig;

import redis.clients.jedis.HostAndPort;
import redis.clients.jedis.JedisCluster;

/**
 * @author Simon.Hoo
 * @version v1.0.0
 * @since JDK1.7
 */
public class RedisTest {

    private static JedisCluster cluster;

    /**
     * @param args
     */
    public static void main(String[] args) {
        HostAndPort host1 = new HostAndPort("10.50.130.101", 6379);
        HostAndPort host2 = new HostAndPort("10.50.130.102", 6379);
        HostAndPort host3 = new HostAndPort("10.50.130.103", 6379);
        HostAndPort host4 = new HostAndPort("10.50.130.104", 6379);
        HostAndPort host5 = new HostAndPort("10.50.130.105", 6379);
        HostAndPort host6 = new HostAndPort("10.50.130.106", 6379);

        Set<HostAndPort> jedisClusterNode = new HashSet<HostAndPort>();
        jedisClusterNode.add(host1);
        jedisClusterNode.add(host2);
        jedisClusterNode.add(host3);
        jedisClusterNode.add(host4);
        jedisClusterNode.add(host5);
        jedisClusterNode.add(host6);

        GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
        poolConfig.setMaxWaitMillis(-1);
        poolConfig.setMaxTotal(1000);
        poolConfig.setMinIdle(8);
        poolConfig.setMaxIdle(100);

        int connectionTimeout=10;
        int soTimeout=10;
```

```
            int maxAttempts = 10;
            String password="Ab123456";

        cluster = new JedisCluster(jedisClusterNode,
connectionTimeout,soTimeout,maxAttempts,password,poolConfig);

        cluster.set("abcdefg", "1234567890");
        System.out.println(cluster.get("abcdefg"));
    }
}
```

6.2 测试结果



PASS.