前言

redis 是我们目前大规模使用的缓存中间件，由于它强大高效而又便捷的功能，得到了广泛的使用。现在的2.x的稳定版本是2.8.19,也是我们项目中普遍用到的版本。
redis在年初发布了3.0.0,官方支持了redis cluster,也就是集群。至此结束了redis 没有官方集群的时代，之前我们用redis cluster用的最多的应该是twitter 发布的Twemproxy(https://github.com/twitter/twemproxy)
还有就是豌豆荚开发的codis(https://github.com/wandoulabs/codis)
这2个我会在后续去使用它们。下面的文字，我尽量用通熟易懂的方式来阐述。
redis cluster 理论知识

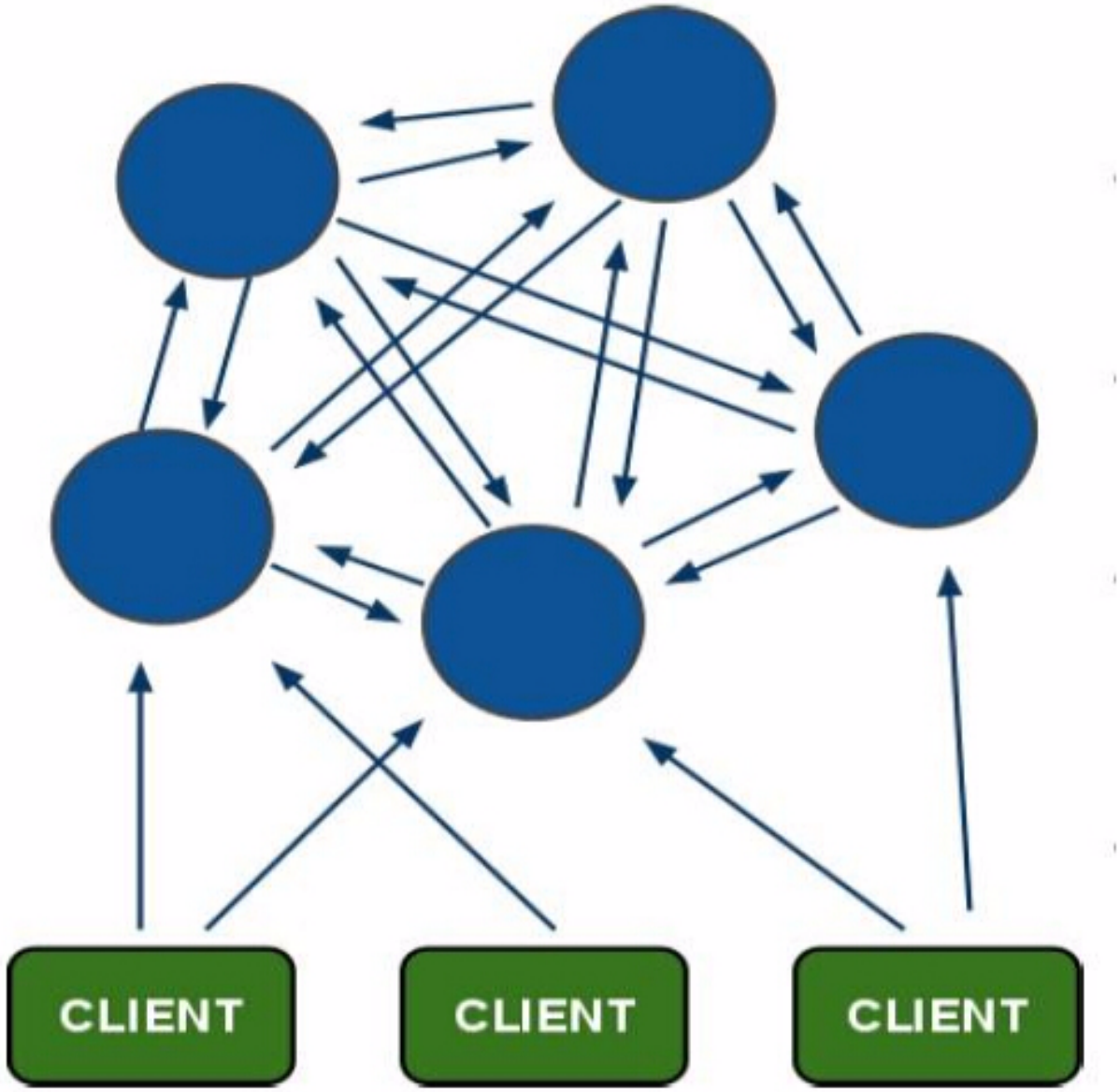
截止写这篇文章前，redis 3.x的最新版本是3.0.5。今天的学习和实践就是用这个版本。
Redis Cluster设计要点

redis cluster在设计的时候，就考虑到了去中心化，去中间件，也就是说，集群中的每个节点都是平等的关系，都是对等的，每个节点都保存各自的数据和整个集群的状态。每个节点都和其他所有节点连接，而且这些连接保持活跃，这样就保证了我们只需要连接集群中的任意一个节点，就可以获取到其他节点的数据。
那么redis 是如何合理分配这些节点和数据的呢?
Redis 集群没有并使用传统的一致性哈希来分配数据，而是采用另外一种叫做哈希槽 (hash slot)的方式来分配的。redis cluster 默认分配了 16384 个slot，当我们set一个key 时，会用CRC16算法来取模得到所属的slot，然后将这个key 分到哈希槽区间的节点上，具体算法就是：CRC16(key) % 16384。
注意的是：必须要3个以后的主节点，否则在创建集群时会失败，我们在后续会实践到。
所以，我们假设现在有3个节点已经组成了集群，分别是：A, B, C 三个节点，它们可以是一台机器上的三个端口，也可以是三台不同的服务器。那么，采用哈希槽 (hash slot)的方式来分配16384个slot 的话，它们三个节点分别承担的slot 区间是：
节点A覆盖0 - 5460;节点B覆盖5461 - 10922;节点C覆盖10923 - 16383.
如下图所示：

那么，现在我想设置一个key ,比如叫my_name:

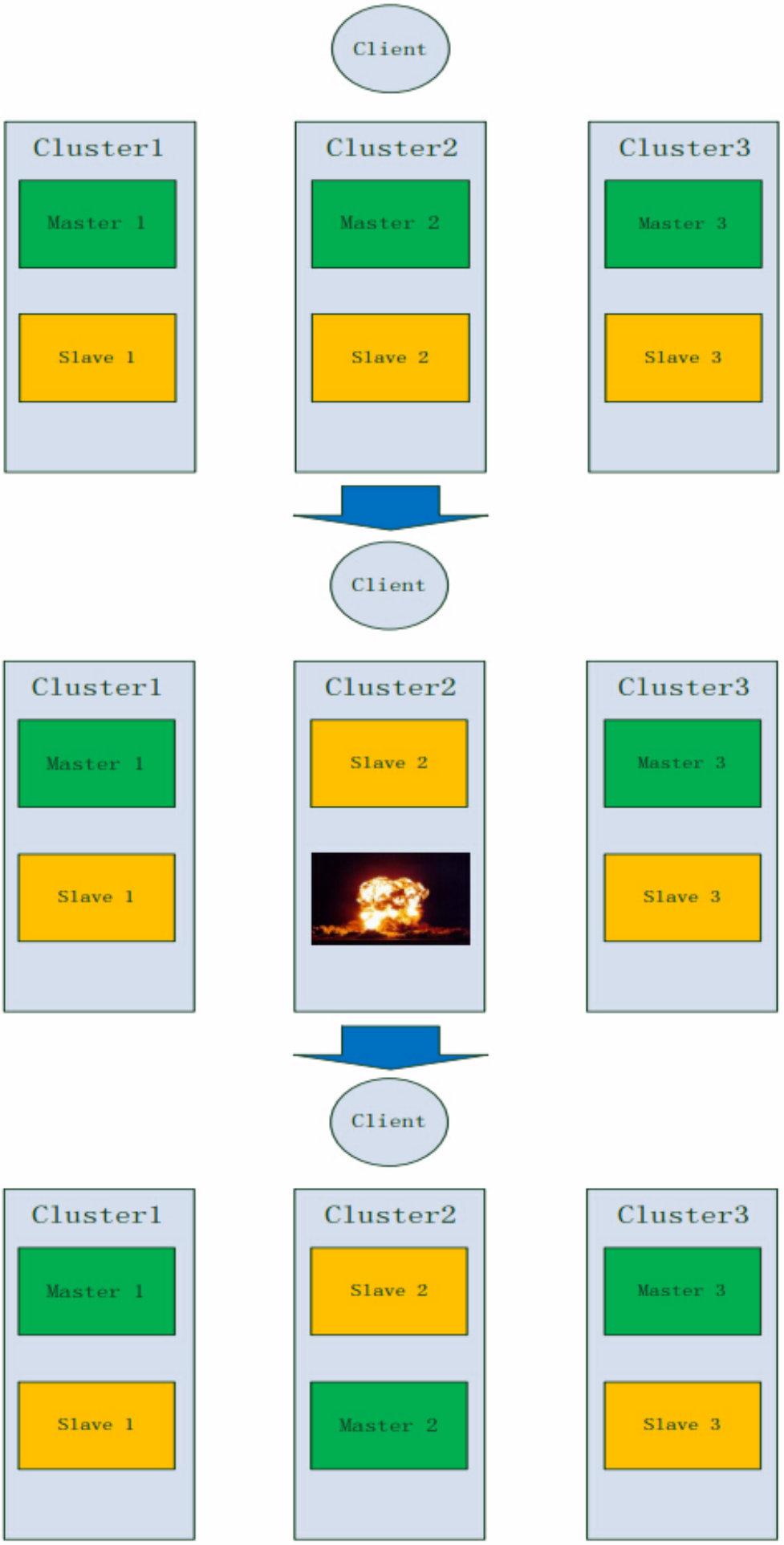
```
set my_name yangyi
```

按照redis cluster的哈希槽算法：CRC16('my_name')%16384 = 2412。 那么就会把这个key 的存储分配到 A 上了。
同样，当我连接(A,B,C)任何一个节点想获取my_name这个key时，也会这样的算法，然后内部跳转到B节点上获取数据。
这种哈希槽的分配方式有好也有坏，好处就是很清晰，比如我想新增一个节点D，redis cluster的这种做法是从各个节点的前面各拿取一部分slot到D上，我会在接下来的实践中实验。大致就会变成这样：
节点A覆盖1365-5460节点B覆盖6827-10922节点C覆盖12288-16383节点D覆盖0-1364,5461-6826,10923-12287
同样删除一个节点也是类似，移动完成后就可以删除这个节点了。
所以redis cluster 就是这样的一个形状：



Redis Cluster主从模式

redis cluster 为了保证数据的高可用性，加入了主从模式，一个主节点对应一个或多个从节点，主节点提供数据存取，从节点则是从主节点拉取数据备份，当这个主节点挂掉后，就会有这个从节点选取一个来充当主节点，从而保证集群不会挂掉。
上面那个例子里, 集群有ABC三个主节点, 如果这3个节点都没有加入从节点，如果B挂掉了，我们就无法访问整个集群了。A和C的slot也无法访问。
所以我们在集群建立的时候，一定要为每个主节点都添加了从节点, 比如像这样, 集群包含主节点A、B、C, 以及从节点A1、B1、C1, 那么即使B挂掉系统也可以继续正确工作。
B1节点替代了B节点，所以Redis集群将会选择B1节点作为新的主节点，集群将会继续正确地提供服务。 当B重新开启后，它就会变成B1的从节点。
不过需要注意，如果节点B和B1同时挂了，Redis集群就无法继续正确地提供服务了。
流程下图所示：



redis cluster 动手实践

按照官方的教程，全程再演示一次，官方教程是在一台机器上启动6个节点，3个当主，3个当从(http://redis.io/topics/cluster-tutorial)：
先下载官方的redis 版本(3.0.5) :http://download.redis.io/releases/redis-3.0.5.tar.gz

下载不了，请自行翻墙。我这次是在centos6.5上演示，用的是root 账户。
如果之前已经下载了redis的 2.x版本，只需要将/usr/local/bin/redis-*这几个命令先删除即可。
1.解压

```
[root@web3 ~]# tar zxvf redis-3.0.5.tar.gz
```

2.安装

```
[root@web3 ~]# cd redis-3.0.5
[root@web3 ~]# make && make install
```

3.将redis-trib.rb 复制到/usr/local/bin

```
[root@web3 redis-3.0.5]# cd src/
[root@web3 src]# cp redis-trib.rb /usr/local/bin
```

4.开始集群搭建

```
[root@web3 redis-3.0.5]# vi redis.conf
#修改以下地方
port 7000
cluster-enabled yes
cluster-config-file nodes.conf
cluster-node-timeout 5000
appendonly yes
```

新建6个节点：

```
[root@web3 redis-3.0.5]# mkdir -p /usr/local/cluster-test
[root@web3 redis-3.0.5]# cd /usr/local/cluster-test/
[root@web3 cluster-test]# mkdir 7000
[root@web3 cluster-test]# mkdir 7001
[root@web3 cluster-test]# mkdir 7002
[root@web3 cluster-test]# mkdir 7003
[root@web3 cluster-test]# mkdir 7004
[root@web3 cluster-test]# mkdir 7005
```

将redis.conf 分别拷贝到这6个文件夹中,并修改成对应的端口号

```
#拷贝配置文件
[root@web3 cluster-test]# cp /root/redis-3.0.5/redis.conf /usr/local/cluster-test/7000
[root@web3 cluster-test]# cp /root/redis-3.0.5/redis.conf /usr/local/cluster-test/7001
[root@web3 cluster-test]# cp /root/redis-3.0.5/redis.conf /usr/local/cluster-test/7002
[root@web3 cluster-test]# cp /root/redis-3.0.5/redis.conf /usr/local/cluster-test/7003
[root@web3 cluster-test]# cp /root/redis-3.0.5/redis.conf /usr/local/cluster-test/7004
[root@web3 cluster-test]# cp /root/redis-3.0.5/redis.conf /usr/local/cluster-test/7005
＃修改端口号
root@web3 cluster-test]# sed -i "s/7000/7001/g" /usr/local/cluster-test/7001/redis.conf
[root@web3 cluster-test]# sed -i "s/7000/7002/g" /usr/local/cluster-test/7002/redis.conf
[root@web3 cluster-test]# sed -i "s/7000/7003/g" /usr/local/cluster-test/7003/redis.conf
[root@web3 cluster-test]# sed -i "s/7000/7004/g" /usr/local/cluster-test/7004/redis.conf
[root@web3 cluster-test]# sed -i "s/7000/7005/g" /usr/local/cluster-test/7005/redis.conf
```

分别，启动这6个节点

```
[root@web3 cluster-test]# cd /usr/local/cluster-test/7000/
[root@web3 7000]# redis-server redis.conf
[root@web3 7000]# cd ../7001
[root@web3 7001]# redis-server redis.conf
[root@web3 7001]# cd ../7002
[root@web3 7002]# redis-server redis.conf
[root@web3 7002]# cd ../7003
[root@web3 7003]# redis-server redis.conf
[root@web3 7003]# cd ../7004
[root@web3 7004]# redis-server redis.conf
[root@web3 7004]# cd ../7005
[root@web3 7005]# redis-server redis.conf
[root@web3 7005]#
```

查看6个节点的启动进程情况：

```
[root@web3 7005]# ps -ef|grep redis
root 11380 1 0 07:37 ? 00:00:00 redis-server *:7000 [cluster]
root 11384 1 0 07:37 ? 00:00:00 redis-server *:7001 [cluster]
root 11388 1 0 07:37 ? 00:00:00 redis-server *:7002 [cluster]
root 11392 1 0 07:37 ? 00:00:00 redis-server *:7003 [cluster]
root 11396 1 0 07:37 ? 00:00:00 redis-server *:7004 [cluster]
root 11400 1 0 07:37 ? 00:00:00 redis-server *:7005 [cluster]
root 11404 8259 0 07:38 pts/0 00:00:00 grep redis
```

将6个节点连在一起构招成集群
需要用到的命令就是redis-trib.rb,这是官方的一个用ruby写的一个操作redis cluster的命令，所以，你的机器上需要安装ruby。我们先试一下这个命令：

```
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
```

因为我们要新建集群,所以这里使用create命令.--replicas 1参数表示为每个主节点创建一个从节点.其他参数是实例的地址集合。

由于我机子上没安装ruby，所以，会报错：

```
/usr/bin/env: ruby: No such file or directory
```

那先安装ruby和rubygems：

```
[root@web3 7005]# yum install ruby ruby-devel rubygems rpm-build
```

然后，再执行一次，发现还是报错：

```
[root@web3 7005]# 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
/usr/lib/ruby/site_ruby/1.8/rubygems/custom_require.rb:31:in `gem_original_require&#39;: no such file to load -- redis (LoadError)
from /usr/lib/ruby/site_ruby/1.8/rubygems/custom_require.rb:31:in `require&#39;
from /usr/local/bin/redis-trib.rb:25
[root@web3 7005]#
```

原来是ruby和redis的连接没安装好：

```
[root@web3 7005]# gem install redis

Successfully installed redis-3.2.1
1 gem installed
Installing ri documentation for redis-3.2.1...
Installing RDoc documentation for redis-3.2.1...
```

好了。最重要的时刻来临了，胜败成举在此了，啊啊啊啊啊啊，好激动：

```
[root@web3 7005]# 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
>>> Creating cluster
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: 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4 127.0.0.1:7000
slots:0-5460 (5461 slots) master
M: cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001
slots:5461-10922 (5462 slots) master
M: dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002
slots:10923-16383 (5461 slots) master
S: d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003
replicates 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4
S: 4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004
replicates cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c
S: 30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005
replicates dfa0754c7854a874a6ebd2613b86140ad97701fc
Can I set the above configuration? (type &#39;yes&#39; 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: 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4 127.0.0.1:7000
slots:0-5460 (5461 slots) master
M: cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001
slots:5461-10922 (5462 slots) master
M: dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002
slots:10923-16383 (5461 slots) master
M: d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003
slots: (0 slots) master
replicates 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4
M: 4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004
slots: (0 slots) master
replicates cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c
M: 30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005
slots: (0 slots) master
replicates dfa0754c7854a874a6ebd2613b86140ad97701fc
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
```

```
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈。终于他妈的成功了！！！！

Redis-trib会提示你做了什么配置，输入yes接受。集群就被配置和加入了，意思是，实例会经过互相交流后启动。

测试集群的状态：

```
[root@web3 7000]# redis-trib.rb check 127.0.0.1:7000
Connecting to node 127.0.0.1:7000: OK
Connecting to node 127.0.0.1:7002: OK
Connecting to node 127.0.0.1:7003: OK
Connecting to node 127.0.0.1:7005: OK
Connecting to node 127.0.0.1:7001: OK
Connecting to node 127.0.0.1:7004: OK
>>> Performing Cluster Check (using node 127.0.0.1:7000)
M: 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4 127.0.0.1:7000
slots:0-5460 (5461 slots) master
1 additional replica(s)
M: dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002
slots:10923-16383 (5461 slots) master
1 additional replica(s)
S: d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003
slots: (0 slots) slave
replicates 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4
S: 30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005
slots: (0 slots) slave
replicates dfa0754c7854a874a6ebd2613b86140ad97701fc
M: cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001
slots:5461-10922 (5462 slots) master
1 additional replica(s)
S: 4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004
slots: (0 slots) slave
replicates cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

可以看到有3个主节点，3个从节点。每个节点都是成功的连接状态。

3个主节点［M］是：

7000 (3707debcbe7be66d4a1968eaf3a5ffaf4308efa4)

7001 (cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c)

7002 (dfa0754c7854a874a6ebd2613b86140ad97701fc)

3个从节点［S］是：

7003 (d2237fdcfbba672de766b913d1186cebcb6e1761)->7000

7004 (4b4aef8b48c427a3c903518339d53b6447c58b93)->7001

7005 (30858dbf483b61b9838d5c1f853a60beaa4e7afd) ->7002

5. 测试连接集群

刚才集群搭建成功了。按照redis cluster的特点，它是去中心化，每个节点都是对等的，所以，你连接哪个节点都可以获取和设置数据，我们来试一下。

redis-cli是redis默认的客户端工具，启动时加上｀-c｀参数，就可以连接到集群。

连接任意一个节点端口：

```
[root@web3 7000]# redis-cli -c -p 7000
127.0.0.1:7000>
```

设置一个值：

```
127.0.0.1:7000> set my_name yangyi
-> Redirected to slot [12803] located at 127.0.0.1:7002
OK
127.0.0.1:7002> get my_name
"yangyi"
127.0.0.1:7002>
```

前面理论知识我们知道了，分配key的时候，它会使用CRC16('my_name')%16384算法，来计算，将这个key 放到哪个节点，这里分配到了12803slot 就分配到了7002(10923-16383)这个节点上。

```
Redirected to slot [12803] located at 127.0.0.1:7002
```

redis cluster 采用的方式很直接，它直接跳转到7002 节点了，而不是还在自身的7000节点。

好，现在我们连接7005这个从节点：

```
[root@web3 7000]# redis-cli -c -p 7005
127.0.0.1:7005> get my_name
-> Redirected to slot [12803] located at 127.0.0.1:7002
"yangyi"
127.0.0.1:7002>
```

我们同样是获取my_name的值，它同样也是跳转到了7002上。

我们再尝试一些其他的可以：

```
127.0.0.1:7002> set age 123
```

```
-> Redirected to slot [741] located at 127.0.0.1:7000
OK
127.0.0.1:7000> set height 565
-> Redirected to slot [8223] located at 127.0.0.1:7001
OK
127.0.0.1:7001> set sex 1
-> Redirected to slot [2584] located at 127.0.0.1:7000
OK
127.0.0.1:7000> set home china
-> Redirected to slot [10814] located at 127.0.0.1:7001
OK
127.0.0.1:7001> set city shanghai
-> Redirected to slot [11479] located at 127.0.0.1:7002
OK
127.0.0.1:7002> set citylocate shanghaipudong
OK
127.0.0.1:7001> set wsdwxzx hhh
-> Redirected to slot [15487] located at 127.0.0.1:7002
OK
127.0.0.1:7002> zadd erew 333 rrr
-> Redirected to slot [10576] located at 127.0.0.1:7001
(integer) 1
127.0.0.1:7000> zrange erew 0 -1
-> Redirected to slot [10576] located at 127.0.0.1:7001
1) "rrr"
127.0.0.1:7001>
```

可以看出，数据都会在7000-7002 这3个主节点来跳转存粗。

6. 测试集群中的节点挂掉

上面我们建立来了一个集群。3个主节点 [7000-7002] 提供数据存粗和读取，3个从节点 [7003-7005] 则是负责把 [7000-7002] 的数据同步到自己的节点上来，我们来看一下 [7003-7005] 的appendonly.aof的内容。看看是不是不这样：

```
[root@web3 7005]# cd /usr/local/cluster-test/7003
[root@web3 7003]# vi appendonly.aof

*2
$6
SELECT
$1
0
*3
$3
set
$3
age
$3
123
*3
$3
set
$3
sex
$1
1
*3
$3
set
$3
job
$3
php
```

我们看下，的确是从7000节点上同步过来的数据，7004，7005也是。

下面，我们先来模拟其中一台Master主服务器挂掉的情况，那就7000挂掉吧：

```
[root@web3 7003]# ps -ef|grep redis
root 11380 1 0 07:37 ? 00:00:03 redis-server *:7000 [cluster]
root 11384 1 0 07:37 ? 00:00:03 redis-server *:7001 [cluster]
root 11388 1 0 07:37 ? 00:00:03 redis-server *:7002 [cluster]
root 11392 1 0 07:37 ? 00:00:03 redis-server *:7003 [cluster]
root 11396 1 0 07:37 ? 00:00:04 redis-server *:7004 [cluster]
root 11400 1 0 07:37 ? 00:00:03 redis-server *:7005 [cluster]
[root@web3 7003]# kill 11380
[root@web3 7003]#
```

好，安装前面的理论，7000主节点挂掉了，那么这个时候，7000的从节点只有7003一个，肯定7003就会被选举称Master节点了：

```
[root@web3 7003]# redis-trib.rb check 127.0.0.1:7000
Connecting to node 127.0.0.1:7000: [ERR] Sorry, can&#39;t connect to node 127.0.0.1:7000
[root@web3 7003]# redis-trib.rb check 127.0.0.1:7001
Connecting to node 127.0.0.1:7001: OK
Connecting to node 127.0.0.1:7004: OK
Connecting to node 127.0.0.1:7005: OK
Connecting to node 127.0.0.1:7003: OK
Connecting to node 127.0.0.1:7002: OK
>>> Performing Cluster Check (using node 127.0.0.1:7001)
M: cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001
slots:5461-10922 (5462 slots) master
1 additional replica(s)
S: 4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004
slots: (0 slots) slave
replicates cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c
S: 30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005
slots: (0 slots) slave
replicates dfa0754c7854a874a6ebd2613b86140ad97701fc
M: d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003
slots:0-5460 (5461 slots) master
0 additional replica(s)
M: dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002
slots:10923-16383 (5461 slots) master
1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
[root@web3 7003]#
```

看了下，上面有了三个M 节点了，果真，7003被选取成了替代7000成为主节点了。那我们来获取原先存在7000节点的数据：

```
[root@web3 7003]# redis-cli -c -p 7001
127.0.0.1:7001> get sex
-> Redirected to slot [2584] located at 127.0.0.1:7003
"1"
127.0.0.1:7003>
```

数据果真没有丢失，而是从7003上面获取了。

OK。我们再来模拟 7000节点重新启动了的情况，那么它还会自动加入到集群中吗？那么，7000这个节点上充当什么角色呢？ 我们试一下：

重新启动 7000 节点：

```
[root@web3 7003]# cd ../7000
[root@web3 7000]# ll
total 56
-rw-r--r-- 1 root root 114 Oct 17 08:16 appendonly.aof
-rw-r--r-- 1 root root 43 Oct 17 08:37 dump.rdb
-rw-r--r-- 1 root root 745 Oct 17 08:00 nodes.conf
-rw-r--r-- 1 root root 41550 Oct 17 07:37 redis.conf
[root@web3 7000]# redis-server redis.conf
```

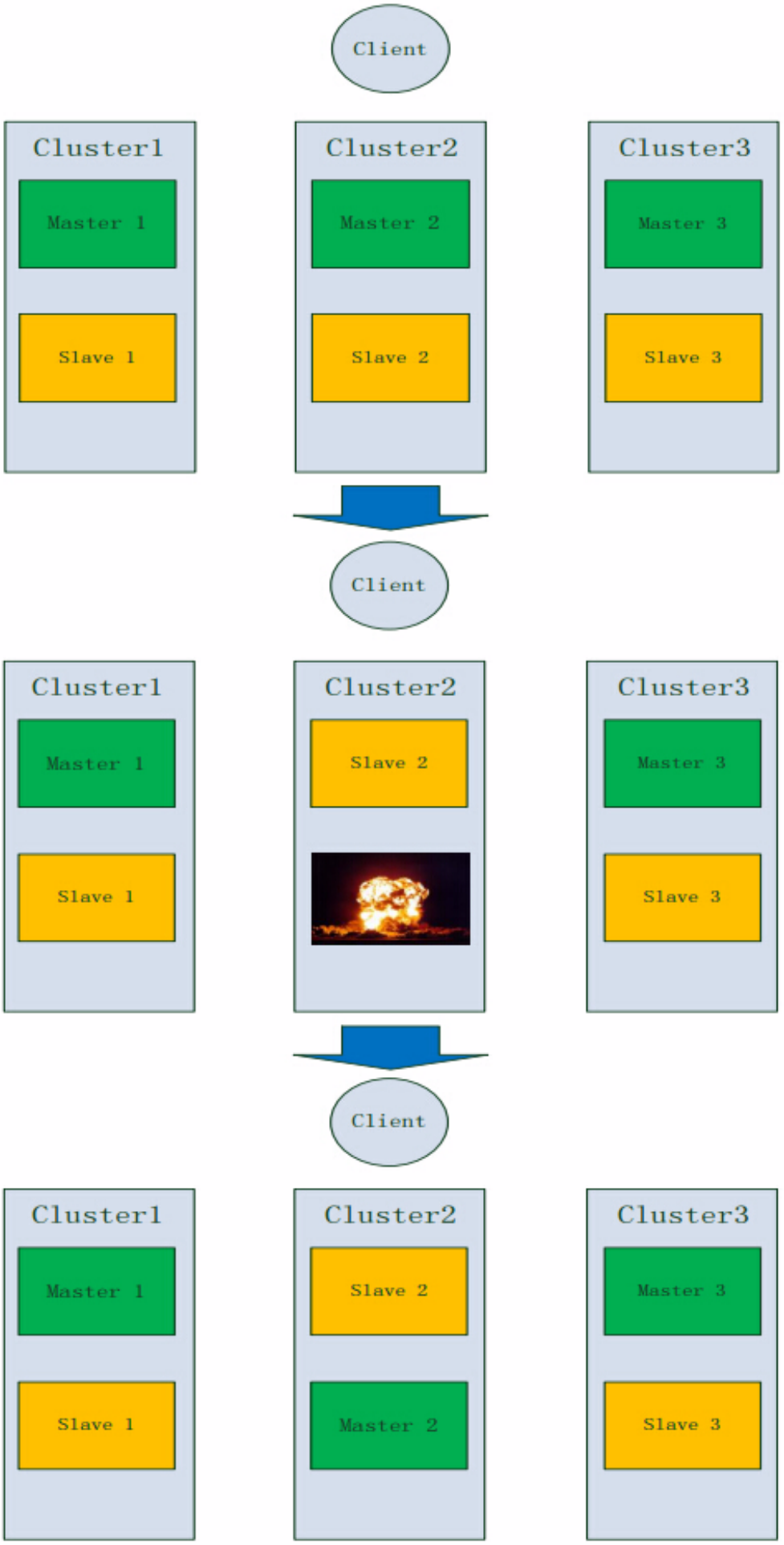
启动好了，现在，再来检查一下集群：

```
redis-trib.rb check 127.0.0.1:700`
```

```
[root@web3 7000]# redis-trib.rb check 127.0.0.1:7001
Connecting to node 127.0.0.1:7001: OK
Connecting to node 127.0.0.1:7004: OK
Connecting to node 127.0.0.1:7005: OK
Connecting to node 127.0.0.1:7003: OK
Connecting to node 127.0.0.1:7000: OK
Connecting to node 127.0.0.1:7002: OK
>>> Performing Cluster Check (using node 127.0.0.1:7001)
M: cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001
slots:5461-10922 (5462 slots) master
1 additional replica(s)
S: 4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004
slots: (0 slots) slave
replicates cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c
S: 30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005
slots: (0 slots) slave
replicates dfa0754c7854a874a6ebd2613b86140ad97701fc
M: d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003
slots:0-5460 (5461 slots) master
1 additional replica(s)
S: 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4 127.0.0.1:7000
slots: (0 slots) slave
```

```
replicates d2237fdcfbba672de766b913d1186cebcb6e1761
M: dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002
slots:10923-16383 (5461 slots) master
1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

你看，7000节点启动起来了，它却作为了 70003 的从节点了。验证了之前的这张图：



一定要保证有3个master 节点，不然，集群就挂掉了。

7. 集群中新加入节点

我们再来测试一下，新加入一个节点，分2种情况，1是作为主节点，2是作为一个节点的从节点。我们分别来试一下：

1. 新建一个7006节点 作为一个新的主节点加入：

首先就是新建一个 7006的文件夹和redis.conf:

```
[root@web3 cluster-test]# cd /usr/local/cluster-test/
[root@web3 cluster-test]# mkdir 7006
[root@web3 cluster-test]# cp 7005/redis.conf 7006/redis.conf
#修改端口
[root@web3 cluster-test]sed -i "s/7005/7006/g" /usr/local/cluster-test/7006/redis.conf
```

启动 7006

```
[root@web3 7006]# redis-server redis.conf
[root@web3 7006]# ps -ef|grep redis
root 11384 1 0 07:37 ? 00:00:05 redis-server *:7001 [cluster]
root 11388 1 0 07:37 ? 00:00:05 redis-server *:7002 [cluster]
root 11392 1 0 07:37 ? 00:00:05 redis-server *:7003 [cluster]
root 11396 1 0 07:37 ? 00:00:06 redis-server *:7004 [cluster]
root 11400 1 0 07:37 ? 00:00:05 redis-server *:7005 [cluster]
root 12100 1 0 08:42 ? 00:00:01 redis-server *:7000 [cluster]
root 12132 1 0 09:09 ? 00:00:00 redis-server *:7006 [cluster]
root 12136 8259 0 09:10 pts/0 00:00:00 grep redis
```

ok,7006 端口已经启动好了，并且进程也存在了，下面就是加入到集群中，也是得用到redis-trib.rb命令：

```
redis-trib.rb add-node 127.0.0.1:7006 127.0.0.1:7000
```

add-node是加入指令，127.0.0.1:7006表示新加入的节点，127.0.0.1:7000表示加入的集群的一个节点，用来辨识是哪个集群，理论上哪个都可以。

```
[root@web3 7006]# redis-trib.rb add-node 127.0.0.1:7006 127.0.0.1:7000
>>> Adding node 127.0.0.1:7006 to cluster 127.0.0.1:7000
Connecting to node 127.0.0.1:7000: OK
Connecting to node 127.0.0.1:7004: OK
Connecting to node 127.0.0.1:7005: OK
Connecting to node 127.0.0.1:7003: OK
Connecting to node 127.0.0.1:7001: OK
Connecting to node 127.0.0.1:7002: OK
>>> Performing Cluster Check (using node 127.0.0.1:7000)
S: 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4 127.0.0.1:7000
slots: (0 slots) slave
replicates d2237fdcfbba672de766b913d1186cebcb6e1761
S: 4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004
slots: (0 slots) slave
replicates cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c
S: 30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005
slots: (0 slots) slave
replicates dfa0754c7854a874a6ebd2613b86140ad97701fc
M: d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003
slots:0-5460 (5461 slots) master
1 additional replica(s)
M: cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001
slots:5461-10922 (5462 slots) master
1 additional replica(s)
M: dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002
slots:10923-16383 (5461 slots) master
1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
Connecting to node 127.0.0.1:7006: OK
>>> Send CLUSTER MEET to node 127.0.0.1:7006 to make it join the cluster.
[OK] New node added correctly.
```

这个命令执行完成之后，它顺便检查来其他的6个节点都是成功的，最后几句话：

```
Connecting to node 127.0.0.1:7006: OK
>>> Send CLUSTER MEET to node 127.0.0.1:7006 to make it join the cluster.
[OK] New node added correctly.
```

表示新的节点连接成功了，而且也已经加入到集群了，我们再来检查一下：

```
[root@web3 7006]# redis-trib.rb check 127.0.0.1:7006
Connecting to node 127.0.0.1:7006: OK
Connecting to node 127.0.0.1:7001: OK
Connecting to node 127.0.0.1:7004: OK
Connecting to node 127.0.0.1:7000: OK
Connecting to node 127.0.0.1:7002: OK
Connecting to node 127.0.0.1:7005: OK
Connecting to node 127.0.0.1:7003: OK
>>> Performing Cluster Check (using node 127.0.0.1:7006)
M: efc3131fbdc6cf929720e0e0f7136cae85657481 127.0.0.1:7006
```

```
slots: (0 slots) master
0 additional replica(s)
M: cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001
slots:5461-10922 (5462 slots) master
1 additional replica(s)
S: 4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004
slots: (0 slots) slave
replicates cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c
S: 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4 127.0.0.1:7000
slots: (0 slots) slave
replicates d2237fdcfbba672de766b913d1186cebcb6e1761
M: dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002
slots:10923-16383 (5461 slots) master
1 additional replica(s)
S: 30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005
slots: (0 slots) slave
replicates dfa0754c7854a874a6ebd2613b86140ad97701fc
M: d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003
slots:0-5460 (5461 slots) master
1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

可以看到了，7006 已经成为了主节点。

我们也可以连接到客户端后，来看这个集群节点情况：

```
[root@web3 7006]# redis-cli -c -p 7006
127.0.0.1:7006> cluster nodes
cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001 master - 0 1445073797986 2 connected 5461-10922
4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004 slave cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 0 1445073799497 2 connected
efc3131fbdc6cf929720e0e0f7136cae85657481 127.0.0.1:7006 myself,master - 0 0 0 connected
3707debcbe7be66d4a1968eaf3a5ffaf4308efa4 127.0.0.1:7000 slave d2237fdcfbba672de766b913d1186cebcb6e1761 0 1445073797482 7 connected
dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002 master - 0 1445073798489 3 connected 10923-16383
30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005 slave dfa0754c7854a874a6ebd2613b86140ad97701fc 0 1445073798993 3 connected
d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003 master - 0 1445073799498 7 connected 0-5460
127.0.0.1:7006>
```

可以看到 有7个节点。7006 也作为了master节点，但是，但是，你看看后面的：

```
efc3131fbdc6cf929720e0e0f7136cae85657481 127.0.0.1:7006
slots: (0 slots) master
```

卧槽，什么情况。o slots也就是说，虽然它现在是主节点，但是，缺没有分配任何节点给它，也就是它现在还不负责数据存取。那加上去有毛用啊！！！！

看来，redis cluster 不是在新加节点的时候帮我们做好了迁移工作，需要我们手动对集群进行重新分片迁移，也是这个命令：

```
redis-trib.rb reshard 127.0.0.1:7000
```

这个命令是用来迁移slot节点的，后面的127.0.0.1:7000是表示是哪个集群，端口填 [7000-7006] 都可以，我们运行下：

```
[root@web3 7006]# redis-trib.rb reshard 127.0.0.1:7000
Connecting to node 127.0.0.1:7006: OK
Connecting to node 127.0.0.1:7001: OK
Connecting to node 127.0.0.1:7004: OK
Connecting to node 127.0.0.1:7000: OK
Connecting to node 127.0.0.1:7002: OK
Connecting to node 127.0.0.1:7005: OK
Connecting to node 127.0.0.1:7003: OK
>>> Performing Cluster Check (using node 127.0.0.1:7006)
M: efc3131fbdc6cf929720e0e0f7136cae85657481 127.0.0.1:7006
slots: (0 slots) master
0 additional replica(s)
M: cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001
slots:5461-10922 (5462 slots) master
1 additional replica(s)
S: 4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004
slots: (0 slots) slave
replicates cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c
S: 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4 127.0.0.1:7000
slots: (0 slots) slave
replicates d2237fdcfbba672de766b913d1186cebcb6e1761
M: dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002
slots:10923-16383 (5461 slots) master
1 additional replica(s)
S: 30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005
slots: (0 slots) slave
replicates dfa0754c7854a874a6ebd2613b86140ad97701fc
M: d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003
```

```
slots:0-5460 (5461 slots) master
1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
How many slots do you want to move (from 1 to 16384)?
```

它提示我们需要迁移多少slot到7006上，我们可以算一下：16384/4 = 4096，也就是说，为了平衡分配起见，我们需要移动4096个槽点到7006上。

好，那输入4096:

```
How many slots do you want to move (from 1 to 16384)? 4096
What is the receiving node ID?
```

它又提示我们，接受的node ID是多少，7006的id 我们通过上面就可以看到是efc3131fbdc6cf929720e0e0f7136cae85657481:

```
What is the receiving node ID? efc3131fbdc6cf929720e0e0f7136cae85657481
Please enter all the source node IDs.
Type &#39;all&#39; to use all the nodes as source nodes for the hash slots.
Type &#39;done&#39; once you entered all the source nodes IDs.
Source node #1:
```

接着， redis-trib 会向你询问重新分片的源节点（source node）， 也即是， 要从哪个节点中取出 4096 个哈希槽， 并将这些槽移动到7006节点上面。

如果我们不打算从特定的节点上取出指定数量的哈希槽， 那么可以向 redis-trib 输入 all ， 这样的话， 集群中的所有主节点都会成为源节点， redis-trib 将从各个源节点中各取出一部分哈希槽， 凑够 4096 个， 然后移动到7006节点上:

```
Source node #1:all
```

接下来就开始迁移了，并且会询问你是否确认：

```
Moving slot 1359 from d2237fdcfbba672de766b913d1186cebcb6e1761
Moving slot 1360 from d2237fdcfbba672de766b913d1186cebcb6e1761
Moving slot 1361 from d2237fdcfbba672de766b913d1186cebcb6e1761
Moving slot 1362 from d2237fdcfbba672de766b913d1186cebcb6e1761
Moving slot 1363 from d2237fdcfbba672de766b913d1186cebcb6e1761
Moving slot 1364 from d2237fdcfbba672de766b913d1186cebcb6e1761
Do you want to proceed with the proposed reshard plan (yes/no)?
```

输入 yes 并使用按下回车之后， redis-trib 就会正式开始执行重新分片操作， 将指定的哈希槽从源节点一个个地移动到7006节点上面。

迁移完毕之后，我们来检查下：

```
[root@web3 7006]# redis-trib.rb check 127.0.0.1:7000
Connecting to node 127.0.0.1:7000: OK
Connecting to node 127.0.0.1:7006: OK
Connecting to node 127.0.0.1:7004: OK
Connecting to node 127.0.0.1:7005: OK
Connecting to node 127.0.0.1:7003: OK
Connecting to node 127.0.0.1:7001: OK
Connecting to node 127.0.0.1:7002: OK
>>> Performing Cluster Check (using node 127.0.0.1:7000)
S: 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4 127.0.0.1:7000
slots: (0 slots) slave
replicates d2237fdcfbba672de766b913d1186cebcb6e1761
M: efc3131fbdc6cf929720e0e0f7136cae85657481 127.0.0.1:7006
slots:0-1364,5461-6826,10923-12287 (4096 slots) master
0 additional replica(s)
S: 4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004
slots: (0 slots) slave
replicates cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c
S: 30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005
slots: (0 slots) slave
replicates dfa0754c7854a874a6ebd2613b86140ad97701fc
M: d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003
slots:1365-5460 (4096 slots) master
1 additional replica(s)
M: cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001
slots:6827-10922 (4096 slots) master
1 additional replica(s)
M: dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002
slots:12288-16383 (4096 slots) master
1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

我们着重看7006:

```
0-1364,5461-6826,10923-12287 (4096 slots)
```

这些原来在其他节点上的slot 杯迁移到了7006上。原来，它只是间隔的移动，并不是衔接的整体移动，我们看下有数据了没?

```
[root@web3 7006]# redis-cli -c -p 7006
127.0.0.1:7006> keys *
1) "city"
```

2) "age"
3) "citylocate"
127.0.0.1:7006> get city
"shanghai"
非常赞，已经有数据了。
2. 新建一个7007从节点，作为7006的从节点

我们再新建一个节点7007，步骤类似，就先省略了。建好后，启动起来，我们看如何把它加入到集群中的从节点中：
```
[root@web3 7007]# redis-trib.rb add-node --slave 127.0.0.1:7007 127.0.0.1:7000
```
add-node的时候加上--slave表示是加入到从节点中，但是这样加，是随机的。这里的命令行完全像我们在添加一个新主服务器时使用的一样，所以我们没有指定要给哪个主服 务器添加副本。这种情况下，redis-trib会将7007作为一个具有较少副本的随机的主服务器的副本。
那么，你猜，它会作为谁的从节点，应该是7006，因为7006还没有从节点。我们运行下。
```
[root@web3 7007]# redis-trib.rb add-node --slave 127.0.0.1:7007 127.0.0.1:7000
...
...
[OK] All 16384 slots covered.
Automatically selected master 127.0.0.1:7006
Connecting to node 127.0.0.1:7007: OK
>>> Send CLUSTER MEET to node 127.0.0.1:7007 to make it join the cluster.
Waiting for the cluster to join.
>>> Configure node as replica of 127.0.0.1:7006.
[OK] New node added correctly.
```
上面提示说，自动选择了7006作为master节点。并且成功了。我们检查下：
```
[root@web3 7007]# redis-trib.rb check 127.0.0.1:7000
Connecting to node 127.0.0.1:7000: OK
Connecting to node 127.0.0.1:7006: OK
Connecting to node 127.0.0.1:7004: OK
Connecting to node 127.0.0.1:7005: OK
Connecting to node 127.0.0.1:7003: OK
Connecting to node 127.0.0.1:7001: OK
Connecting to node 127.0.0.1:7007: OK
Connecting to node 127.0.0.1:7002: OK
>>> Performing Cluster Check (using node 127.0.0.1:7000)
S: 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4 127.0.0.1:7000
slots: (0 slots) slave
replicates d2237fdcfbba672de766b913d1186cebcb6e1761
M: efc3131fbdc6cf929720e0e0f7136cae85657481 127.0.0.1:7006
slots:0-1364,5461-6826,10923-12287 (4096 slots) master
1 additional replica(s)
S: 4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004
slots: (0 slots) slave
replicates cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c
S: 30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005
slots: (0 slots) slave
replicates dfa0754c7854a874a6ebd2613b86140ad97701fc
M: d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003
slots:1365-5460 (4096 slots) master
1 additional replica(s)
M: cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001
slots:6827-10922 (4096 slots) master
1 additional replica(s)
S: 86d05e7c2b197dc182b5e71069e791d033cf899e 127.0.0.1:7007
slots: (0 slots) slave
replicates efc3131fbdc6cf929720e0e0f7136cae85657481
M: dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002
slots:12288-16383 (4096 slots) master
1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```
果然，7007加入到了7006的从节点当中。
你说，我如果想指定一个主节点行不行？当然可以。我们再建一个7008节点。
```
redis-trib.rb add-node --slave --master-id efc3131fbdc6cf929720e0e0f7136cae85657481 127.0.0.1:7008 127.0.0.1:7000
```
--master-id表示指定的主节点node id。这里指定的是7006这个主节点。
```
Waiting for the cluster to join.
>>> Configure node as replica of 127.0.0.1:7006.
[OK] New node added correctly.
[root@web3 7008]#
```
提示我们已经作为7006的附属品，也就是加入到7006的从节点来了，照这么说，7006就有2个从节点了，我们看一下：
```
redis-cli -c -p 7008 cluster nodes |grep efc3131fbdc6cf929720e0e0f7136cae85657481
```

```
2.
3. 86d05e7c2b197dc182b5e71069e791d033cf899e 127.0.0.1:7007 slave efc3131fbdc6cf929720e0e0f7136cae85657481 0 1445089507786 8 connected
4. efc3131fbdc6cf929720e0e0f7136cae85657481 127.0.0.1:7006 master - 0 1445089508289 8 connected 0-1364 5461-6826 10923-12287
5. 44321e7d619410dc4e0a8745366610a0d06d2395 127.0.0.1:7008 myself,slave efc3131fbdc6cf929720e0e0f7136cae85657481 0 0 0 connected
```

我们过滤了下看结果，果真，7007和7008是7006的从节点了。

刚好，我们再做一个实验，我把7006的杀掉，看7007和7008谁会变成主节点：

```
1. [root@web3 7008]# ps -ef|grep redis
2. root 11384 1 0 09:56 ? 00:00:16 redis-server *:7001 [cluster]
3. root 11388 1 0 09:56 ? 00:00:16 redis-server *:7002 [cluster]
4. root 11392 1 0 09:56 ? 00:00:16 redis-server *:7003 [cluster]
5. root 11396 1 0 09:56 ? 00:00:15 redis-server *:7004 [cluster]
6. root 11400 1 0 09:56 ? 00:00:15 redis-server *:7005 [cluster]
7. root 12100 1 0 11:01 ? 00:00:11 redis-server *:7000 [cluster]
8. root 12132 1 0 11:28 ? 00:00:11 redis-server *:7006 [cluster]
9. root 12202 1 0 13:14 ? 00:00:02 redis-server *:7007 [cluster]
0. root 12219 1 0 13:39 ? 00:00:00 redis-server *:7008 [cluster]
1. root 12239 8259 0 13:49 pts/0 00:00:00 grep redis
2. [root@web3 7008]# kill 12132
3. [root@web3 7008]# redis-cli -c -p 7008
4. 127.0.0.1:7008> get ss5rtr
5. -> Redirected to slot [1188] located at 127.0.0.1:7007
6. "66"
7. 127.0.0.1:7007> cluster nodes
8. efc3131fbdc6cf929720e0e0f7136cae85657481 127.0.0.1:7006 master,fail - 1445089780668 1445089779963 8 disconnected
9. d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003 master - 0 1445089812195 7 connected 1365-5460
0. 30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005 slave dfa0754c7854a874a6ebd2613b86140ad97701fc 0 1445089813710 3 connected
1. 86d05e7c2b197dc182b5e71069e791d033cf899e 127.0.0.1:7007 myself,master - 0 0 10 connected 0-1364 5461-6826 10923-12287
2. cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001 master - 0 1445089814214 2 connected 6827-10922
3. 4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004 slave cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 0 1445089812701 2 connected
4. 44321e7d619410dc4e0a8745366610a0d06d2395 127.0.0.1:7008 slave 86d05e7c2b197dc182b5e71069e791d033cf899e 0 1445089814214 10 connected
5. 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4 127.0.0.1:7000 slave d2237fdcfbba672de766b913d1186cebcb6e1761 0 1445089813204 7 connected
6. dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002 master - 0 1445089813204 3 connected 12288-16383
7. 127.0.0.1:7007>
```

看，7007获得了成为主节点的机会，7008就变成了7007的从节点。

那么这个时候，重启7006节点，那么他就会变成了一个7007的从节点了。

```
1. 27.0.0.1:7008> cluster nodes
2. 30858dbf483b61b9838d5c1f853a60beaa4e7afd 127.0.0.1:7005 slave dfa0754c7854a874a6ebd2613b86140ad97701fc 0 1445089986148 3 connected
3. 86d05e7c2b197dc182b5e71069e791d033cf899e 127.0.0.1:7007 master - 0 1445089986652 10 connected 0-1364 5461-6826 10923-12287
4. d2237fdcfbba672de766b913d1186cebcb6e1761 127.0.0.1:7003 master - 0 1445089986148 7 connected 1365-5460
5. cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 127.0.0.1:7001 master - 0 1445089987155 2 connected 6827-10922
6. efc3131fbdc6cf929720e0e0f7136cae85657481 127.0.0.1:7006 slave 86d05e7c2b197dc182b5e71069e791d033cf899e 0 1445089985644 10 connected
7. 44321e7d619410dc4e0a8745366610a0d06d2395 127.0.0.1:7008 myself,slave 86d05e7c2b197dc182b5e71069e791d033cf899e 0 0 0 connected
8. dfa0754c7854a874a6ebd2613b86140ad97701fc 127.0.0.1:7002 master - 0 1445089986652 3 connected 12288-16383
9. 4b4aef8b48c427a3c903518339d53b6447c58b93 127.0.0.1:7004 slave cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c 0 1445089987660 2 connected
0. 3707debcbe7be66d4a1968eaf3a5ffaf4308efa4 127.0.0.1:7000 slave d2237fdcfbba672de766b913d1186cebcb6e1761 0 1445089985644 7 connected
1. 127.0.0.1:7008>
```

3. 移除一个主节点

有加肯定有减，redis cluster同样支持移除节点功能，同样也是redis-trib.rb的用法：

```
redis-trib del-node 127.0.0.1:7000 ``
```

和新加节点有点不同的是，移除需要节点的node-id。那我们尝试将7007这个主节点移除：

```
1. [root@web3 7006]# redis-trib.rb del-node 127.0.0.1:7000 86d05e7c2b197dc182b5e71069e791d033cf899e
2. >>> Removing node 86d05e7c2b197dc182b5e71069e791d033cf899e from cluster 127.0.0.1:7000
3. Connecting to node 127.0.0.1:7000: OK
4. Connecting to node 127.0.0.1:7006: OK
5. Connecting to node 127.0.0.1:7004: OK
6. Connecting to node 127.0.0.1:7001: OK
7. Connecting to node 127.0.0.1:7003: OK
8. Connecting to node 127.0.0.1:7007: OK
9. Connecting to node 127.0.0.1:7008: OK
0. Connecting to node 127.0.0.1:7005: OK
1. Connecting to node 127.0.0.1:7002: OK
2. [ERR] Node 127.0.0.1:7007 is not empty! Reshard data away and try again.
```

报错了，它提示我们说，由于7007里面已经有数据了，不能被移除，要先将它的数据转移出去。也就是说得重新分片，用上面增加新节点后的分片方式一样，用我们再来一遍：

```
redis-trib.rb reshard 127.0.0.1:7000
```

由于中间太多内容，就省略不重要的，只简单说明下关键的几步：

```
1. M: 86d05e7c2b197dc182b5e71069e791d033cf899e 127.0.0.1:7007
2. slots:0-1364,5461-6826,10923-12287 (4096 slots) master
3.
4. How many slots do you want to move (from 1 to 16384)?
```

提示，我们要分多少个槽点，由于7007上有4096个槽点，所以这里填写4096

```
How many slots do you want to move (from 1 to 16384)? 4096
What is the receiving node ID?
```

提示我们，需要移动到哪个id上，那就填7001的吧：

```
What is the receiving node ID? cb5c04b6160c3b7e18cad5d49d8e2987b27e0d6c
Please enter all the source node IDs.
Type &#39;all&#39; to use all the nodes as source nodes for the hash slots.
Type &#39;done&#39; once you entered all the source nodes IDs.
Source node #1:
```

这里就是关键了，它要我们从哪个节点去转移数据到7001，因为我们是要删除7007的，所以，我们就得7007的id了

```
Source node #1:86d05e7c2b197dc182b5e71069e791d033cf899e
Source node #2:done
Do you want to proceed with the proposed reshard plan (yes/no)? yes
```

ok,这样就迁移好了。我们看看7007是否为空了：

```
[root@web3 7006]# redis-cli -c -p 7007
127.0.0.1:7007> keys *
(empty list or set)
127.0.0.1:7007> cluster nodes

86d05e7c2b197dc182b5e71069e791d033cf899e 127.0.0.1:7007 myself,master - 0 0 10 connected
```

果然为空了，好。现在再进行移除节点操作：

```
[root@web3 7006]# redis-trib.rb del-node 127.0.0.1:7000 86d05e7c2b197dc182b5e71069e791d033cf899e
>>> Removing node 86d05e7c2b197dc182b5e71069e791d033cf899e from cluster 127.0.0.1:7000
Connecting to node 127.0.0.1:7000: OK
Connecting to node 127.0.0.1:7006: OK
Connecting to node 127.0.0.1:7004: OK
Connecting to node 127.0.0.1:7001: OK
Connecting to node 127.0.0.1:7003: OK
Connecting to node 127.0.0.1:7007: OK
Connecting to node 127.0.0.1:7008: OK
Connecting to node 127.0.0.1:7005: OK
Connecting to node 127.0.0.1:7002: OK
>>> Sending CLUSTER FORGET messages to the cluster...
>>> 127.0.0.1:7006 as replica of 127.0.0.1:7001
>>> 127.0.0.1:7008 as replica of 127.0.0.1:7001
>>> SHUTDOWN the node.
```

哈哈哈哈哈，删除成功，而且还很人性化的将7006和70082个没了爹的孩子送给了7001。好评

来源： http://m.2cto.com/kf/201606/514203.html