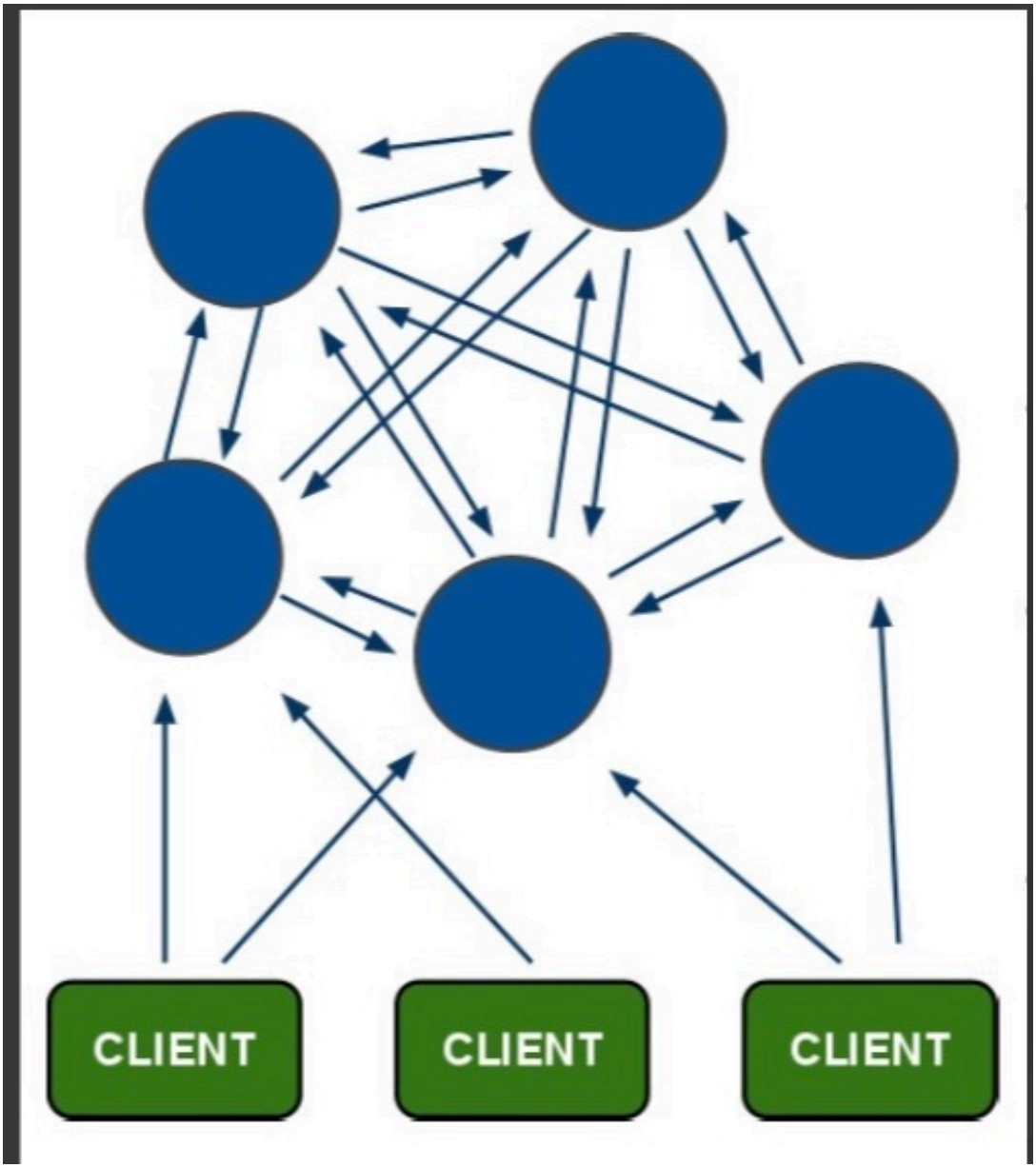
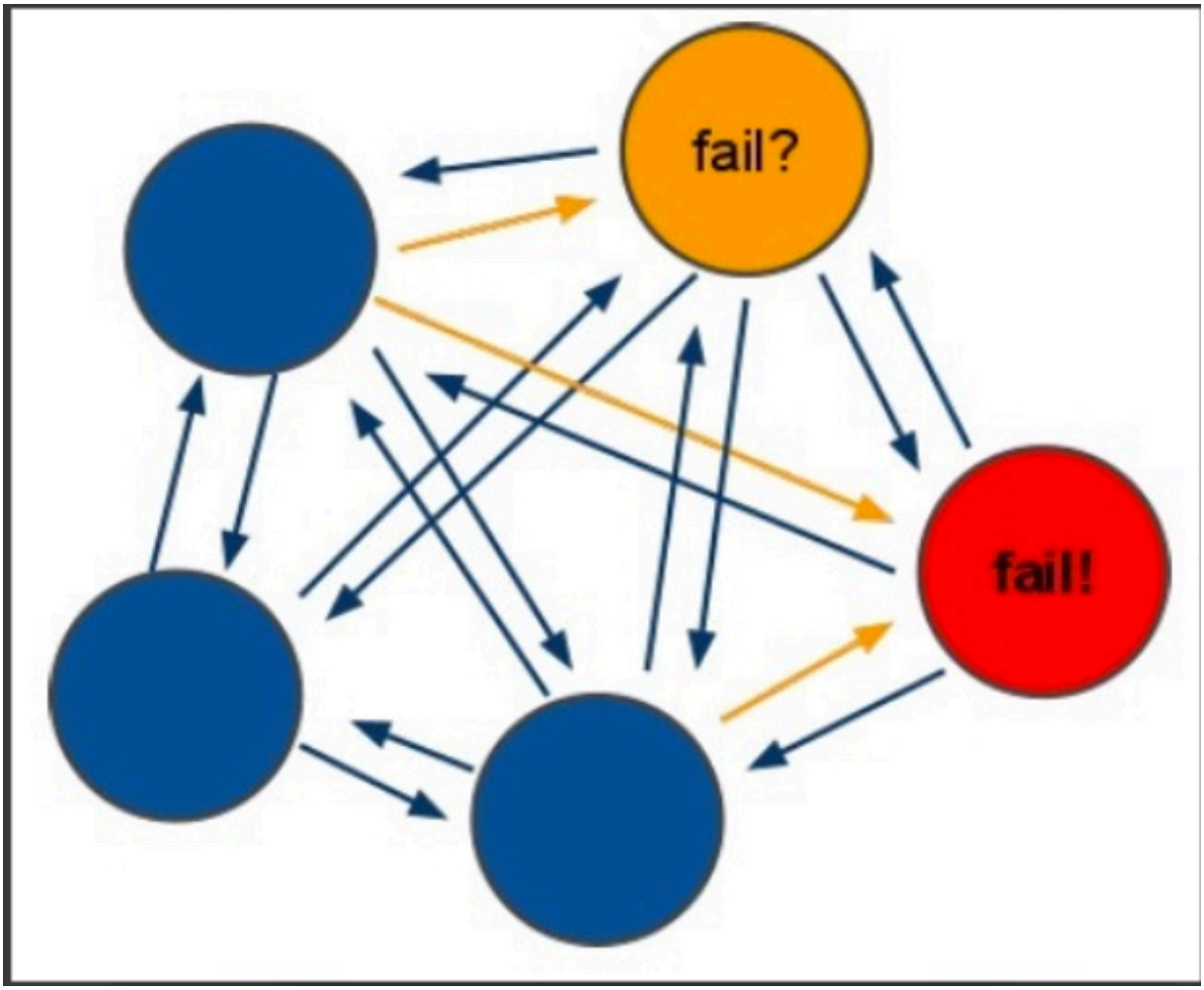
02_Redis集群搭建

一.Redis集群简介

1.redis-cluster架构图

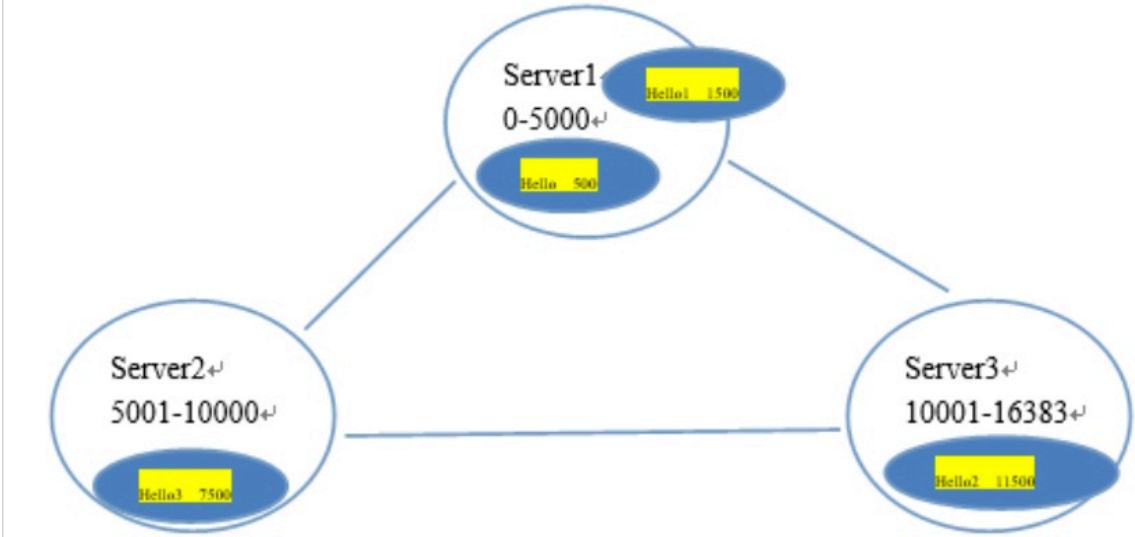


2.redis-cluster集群:容错



3.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
Redis 集群中内置了 16384 个哈希槽，当需要在 Redis 集群中放置一个 key-value 时，redis 先对 key 使用 crc16 算法算出一个结果，
然后把结果对 16384 求余数，这样每个 key 都会对应一个编号在 0-16383 之间的哈希槽，redis 会根据节点数量大致均等的将哈希槽映射到不同的节点



二.Redis集群的搭建

Redis集群中至少应该有三个节点。要保证集群的高可用，需要每个节点有一个备份机。
Redis集群至少需要6台服务器。

搭建伪分布式。可以使用一台虚拟机运行6个redis实例。然后修改redis的端口号7001-7006(自定义端口,只要不和其他程序重复)

1.集群搭建环境

1.1.使用ruby脚本搭建集群,需要ruby的运行环境.

```
yum install ruby
yum install rubygems
```

1.2.上传并安装ruby脚本运行使用的包。

```
[root@yyg ~]# gem install redis-3.2.2.gem
Successfully installed redis-3.2.2
1 gem installed
Installing ri documentation for redis-3.2.2...
```

```
    Installing RDoc documentation for redis-3.2.2..
    [root@localhost ~]#

    [root@localhost ~]# cd /usr/local/redis-3.2.2/src
    [root@localhost src]# ll *.rb
    -rwxrwxr-x. 1 root root 48141 Apr  1  2015 redis-trib.rb
```

2.搭建步骤

需要6台redis服务器,搭建伪分布式集群,用一台虚拟机模拟出6个节点，创建出3个master节点、3个salve节点。这几个服务器分别运行在不同的端口7001-7006.

2.1.具体搭建步骤:

1️⃣.下载并解压:

```
cd /usr/local/redis-3.2.2
wget http://download.redis.io/releases/redis-3.2.2.tar.gz
tar -zxvf redis-3.2.2.tar.gz
```

2️⃣.编译安装:

```
cd redis-3.2.2
make && make install
```

3️⃣.将/usr/local/redis-3.2.2/src/redis-trib.rb复制到 /usr/local/bin目录下.

```
    cd /usr/local/redis-3.2.2/src
    cp redis-trib.rb /usr/local/bin/
```

4️⃣.创建 redis的6个节点

在/url/local/redis-3.2.2目录下创建 redis-cluster目录

`mkdir redis-cluster`

在redis-cluster目录下,创建名为7001,7002,7003,7004,7005,7006的目录,并将/usr/local/redis-3.2.2/conf/redis.conf 拷贝到这三个目录中.

```
cd /usr/local/redis-3.2.2/redis-cluster
```

```
mkdir 7001 7002 7003 7004 7005 7006

cd /usr/local/redis-3.2.2/conf
cp redis.conf redis_cluster/7001
cp redis.conf redis_cluster/7002
cp redis.conf redis_cluster/7003
cp redis.conf redis_cluster/7004
cp redis.conf redis_cluster/7005
cp redis.conf redis_cluster/7006
```

分别修改这6个配置文件，修改如下内容:

```
port  7001                        //端口7001,7002,7003,7004,7005,7006

bind 本机ip  //默认ip为127.0.0.1,需要将其改为其他节点机器可访问的真实ip,
               //否则创建集群时无法访问对应的端口无法创建集群.
daemonize    yes                          //redis后台运行
pidfile  /var/run/redis_7001.pid     //pidfile文件对应7001,7002,7003..
..
cluster-enabled  yes                      //开启集群  把注释#去掉
cluster-config-file  nodes_7001.conf   //集群的配置,
                           //配置文件首次启动自动生成nodes_7001,7002,7
003文件.
cluster-node-timeout  15000         //请求超时时间,默认15秒, 可自行设置
appendonly  yes    //aof日志开启,有需要就开启,它会每次写操作都记录一条日
志
```

5️⃣.启动各个节点

一个节点一个节点的启动命令:

```
cd /usr/local/redis-3.2.2
redis-server redis-cluster/7001/redis.conf
redis-server redis-cluster/7002/redis.conf
redis-server redis-cluster/7003/redis.conf

redis-server redis-cluster/7004/redis.conf
redis-server redis-cluster/7005/redis.conf
redis-server redis-cluster/7006/redis.conf
```

为了方便每个redis的启动,可以创建启动redis服务的shell脚本:

```
cd /usr/local/redis-3.2.2/redis-cluster/
touch start-all-redis.sh
```

注意:以下内容是shell脚本的内容.

```
!/bin/sh
redis-server /usr/local/redis-3.2.2/redis-cluster/7001/redis.conf
redis-server /usr/local/redis-3.2.2/redis-cluster/7002/redis.conf
redis-server /usr/local/redis-3.2.2/redis-cluster/7003/redis.conf
redis-server /usr/local/redis-3.2.2/redis-cluster/7004/redis.conf
redis-server /usr/local/redis-3.2.2/redis-cluster/7005/redis.conf
redis-server /usr/local/redis-3.2.2/redis-cluster/7006/redis.conf
```

更改shell脚本的权限

```
chmod +x start-all-redis.sh
```

创建关闭每个redis服务的shell脚本:

```
cd /usr/local/redis-3.2.2/redis-cluster/
touch stop-all-redis.sh

#!/bin/sh
redis-cli -h 10.11.51.24 -p 7001 shutdown
redis-cli -h 10.11.51.24 -p 7002 shutdown
redis-cli -h 10.11.51.24 -p 7003 shutdown
redis-cli -h 10.11.51.24 -p 7004 shutdown
redis-cli -h 10.11.51.24 -p 7005 shutdown
redis-cli -h 10.11.51.24 -p 7006 shutdown

#更改shell脚本的权限
chmod +x start-all-redis.sh
```

6️⃣.检查 redis启动情况

```
ps -ef | grep redis
```

```
root      61020      1  0 02:14 ?        00:00:01 redis-server 127.0.
0.1:7001 [cluster]
root      61024      1  0 02:14 ?        00:00:01 redis-server 127.0.
0.1:7002 [cluster]
root      61029      1  0 02:14 ?        00:00:01 redis-server 127.0.
0.1:7003 [cluster]

netstat -tnlp | grep redis
tcp        0      0 127.0.0.1:17001             0.0.0.0:*
       LISTEN      61020/redis-server
tcp        0      0 127.0.0.1:17002             0.0.0.0:*
       LISTEN      61024/redis-server
tcp        0      0 127.0.0.1:17003             0.0.0.0:*
       LISTEN      61029/redis-server
tcp        0      0 127.0.0.1:7001             0.0.0.0:*
      LISTEN      61020/redis-server
tcp        0      0 127.0.0.1:7002             0.0.0.0:*
       LISTEN      61024/redis-server
tcp        0      0 127.0.0.1:7003             0.0.0.0:*
       LISTEN      61029/redis-server
```

7️⃣.创建集群

Redis 官方提供了redis-trib.rb 这个工具，就在解压目录的 src 目录中，第三步中已将它复制到 /usr/local/bin 目录中，可以直接在命令行中使用了。使用下面这个命令即可完成安装。

```
redis-trib.rb create --replicas 1 192.168.31.245:7001 192.168.31.245:7002
192.168.31.245:7003 192.168.31.210:7004 192.168.31.210:7005
192.168.31.210:7006
```

运行效果:

```
[root@MiWiFi-R1CM-srv redis-3.2.4]# redis-trib.rb create --replicas 1 192.168.31.245:7000 192.168.31.245:7001 192.168.31.245:7002 192.168.31.210:7003 192.168.31.210:7004 192.168.31.210:7005
>>> Creating cluster
>>> Performing hash slots allocation on 6 nodes...
Using 3 masters:
192.168.31.245:7000
192.168.31.210:7003
192.168.31.245:7001
Adding replica 192.168.31.210:7004 to 192.168.31.245:7000
Adding replica 192.168.31.245:7002 to 192.168.31.210:7003
Adding replica 192.168.31.210:7005 to 192.168.31.245:7001
M: 6499175e9c36526d6feaf5cdd0aaef2eeb4f8aed 192.168.31.245:7000
   slots:0-5460 (5461 slots) master
M: 8178c0e94374079b73ad0bd9f422ba8aea0428b8 192.168.31.245:7001
   slots:10923-16383 (5461 slots) master
S: e24d863eba98c4048250c963005e659f6626d7b6 192.168.31.245:7002
   replicates 3f5f3bef6890185738dee491ae2a0dbf31a57d79
M: 3f5f3bef6890185738dee491ae2a0dbf31a57d79 192.168.31.210:7003
   slots:5461-10922 (5462 slots) master
S: 8b79bb3bb569a60b421b080845c3e422a8365abe 192.168.31.210:7004
   replicates 6499175e9c36526d6feaf5cdd0aaef2eeb4f8aed
S: 97afa2b9761146ea7551905a060ba2b6a83bc888 192.168.31.210:7005
   replicates 8178c0e94374079b73ad0bd9f422ba8aea0428b8
Can I set the above configuration? (type 'yes' to accept):
```

输入 yes 即可，然后出现如下内容，说明安装成功.

```
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 192.168.31.245:7000)
M: 6499175e9c36526d6feaf5cdd0aaef2eeb4f8aed 192.168.31.245:7000
   slots:0-5460 (5461 slots) master
   1 additional replica(s)
S: e24d863eba98c4048250c963005e659f6626d7b6 192.168.31.245:7002
   slots: (0 slots) slave
   replicates 3f5f3bef6890185738dee491ae2a0dbf31a57d79
M: 8178c0e94374079b73ad0bd9f422ba8aea0428b8 192.168.31.245:7001
   slots:10923-16383 (5461 slots) master
   1 additional replica(s)
S: 97afa2b9761146ea7551905a060ba2b6a83bc888 192.168.31.210:7005
   slots: (0 slots) slave
   replicates 8178c0e94374079b73ad0bd9f422ba8aea0428b8
M: 3f5f3bef6890185738dee491ae2a0dbf31a57d79 192.168.31.210:7003
   slots:5461-10922 (5462 slots) master
   1 additional replica(s)
S: 8b79bb3bb569a60b421b080845c3e422a8365abe 192.168.31.210:7004
   slots: (0 slots) slave
   replicates 6499175e9c36526d6feaf5cdd0aaef2eeb4f8aed
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

8.集群验证

比如在第一台机器上连接集群的7003端口的节点，在另外一台连接7005节点，连接方式为 redis-cli -h 192.168.31.245 -c -p 7003 ,加参数 -c 可连接到集群，因为上面 redis.conf 将 bind 改为了ip地址，所以 -h 参数不可以省略.
在7005节点执行命令 set hello world ， 执行结果如下：

```
192.168.31.210:7005> set hello world
-> Redirected to slot [866] located at 192.168.31.245:7000
OK
192.168.31.245:7000> keys *
1) "hello"
```

然后在另外一台7002端口，查看 key 为 hello 的内容， get hello ，执行结果如下：

```
192.168.31.245:7002> get hello
-> Redirected to slot [866] located at 192.168.31.245:7000
"world"
```

说明集群运作正常。

Redis集群运行原理:

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。

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。所以我们在测试的时候看到set 和 get 的时候，直接跳转到了7000端口的节点。

Redis 集群会把数据存在一个 master 节点上，然后在这个 master 和其对应的 salve 之间进行数据同步。当读取数据时，也根据一致性哈希算法到对应的 master 节点获取数据。只有当一个master 挂掉之后，才会启动一个对应的 salve 节点，充当 master 。

注意：

必须要创建3个或以上的主节点，否则在创建集群时会失败，并且当存活的主节点数小于总节点数的一半时，整个集群就无法提供服务了。