# Redis的集群(Redis Cluster)

## Redis-cluster架构图

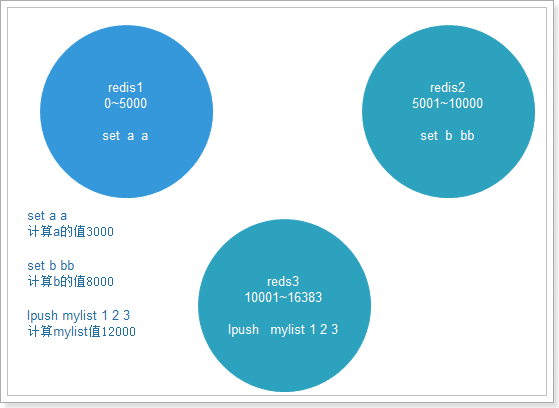


1. 所有的redis节点彼此互联(PING-PONG机制),内部使用二进制协议优化传输速度和带宽。
2. 节点的fail是通过集群中超过半数的节点检测失效时才生效。

(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



#### redis-cluster投票:容错



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

(2):什么时候整个集群不可用(cluster\_state:fail)?

    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]不完成时进入fail状态. ps : **redis-3.0.0.rc1加入cluster-require-full-coverage参数,默认关闭,打开集群兼容部分失败.**

**b:如果集群超过半数以上master挂掉，无论是否有slave，集群进入fail状态**.

## Redis集群搭建

为了保证可以进行投票，需要至少3个主节点。

每个主节点都需要至少一个从节点,所以需要至少3个从节点

一共需要6台redis服务器

可以使用6个redis实例

6个redis实例的端口号,7001~7006

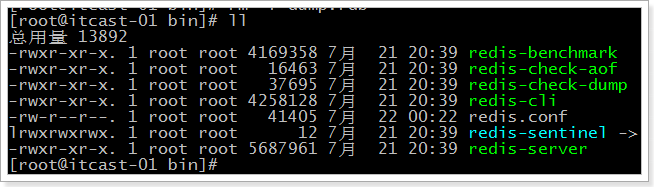
### 准备6个redis实例

1. 把bin目录里面的rdb,和aof文件删除,准备干净的redis

[root@localhost bin]# rm -rf appendonly.aof

[root@localhost bin]# rm -f dump.rdb

效果：



1. 把bin复制6份

[root@localhost bin]# cd ..

[root@localhost redis]# cp -r bin redis-cluster/redis1

[root@localhost redis]# cp -r bin redis-cluster/redis2

[root@localhost redis]# cp -r bin redis-cluster/redis3

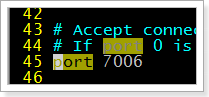
[root@localhost redis]# cp -r bin redis-cluster/redis4

[root@localhost redis]# cp -r bin redis-cluster/redis5

[root@localhost redis]# cp -r bin redis-cluster/redis6

1. 修改端口号

依次修改端口号为7001~7006



1. 启动6个redis实例

编写脚本实现

1. **编写脚本**

[root@localhost redis-cluster]# vim start-all.sh

cd redis1

./redis-server redis.conf

cd ..

cd redis2

./redis-server redis.conf

cd ..

cd redis3

./redis-server redis.conf

cd ..

cd redis4

./redis-server redis.conf

cd ..

cd redis5

./redis-server redis.conf

cd ..

cd redis6

./redis-server redis.conf

cd ..

1. **设置脚本启动权限**

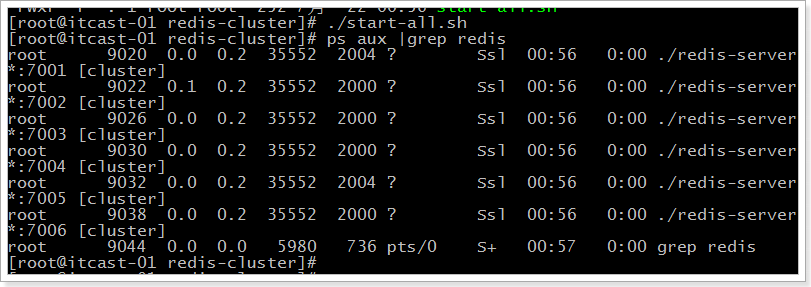
[root@localhost redis-cluster]# **chmod u+x start-all.sh**

1. 执行脚本

[root@localhost redis-cluster]# ./start-all.sh

[root@localhost redis-cluster]# ps aux | grep redis

效果



### 准备redis集群的安装环境

**redis集群的管理工具使用的是ruby脚本语言，安装集群需要ruby环境。**

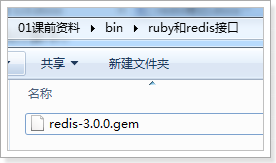
**安装ruby环境**（本课程虚拟机已安装）

[root@itcast-01 redis-cluster]# **yum install ruby**

**安装Ruby的打包系统**（本课程虚拟机已安装）

[root@itcast-01 redis-cluster]# **yum install rubygems**

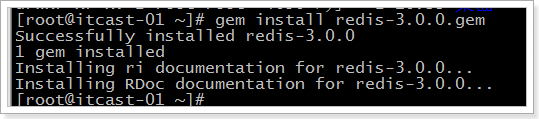
**安装redis的ruby接口库**



**先上传，再执行命令安装**

[root@itcast-01 ~]# **gem install redis-3.0.0.gem**

效果：

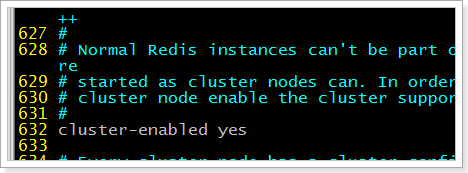


集群管理工具在redis解压文件夹的src的文件夹中

### 安装redis集群

需要修改每个实例的redis.conf配置文件,开启redis-cluster

[root@localhost redis-cluster]# vim redis1/redis.conf

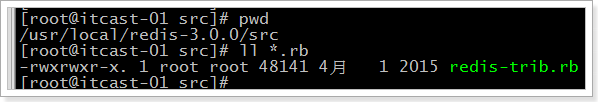


需要修改6个节点

重启redis实例

[root@localhost redis-cluster]# ./start-all.sh

使用redis-cluster的集群管理工具启动集群



**先进入集群管理工具所在的路径**

[root@localhost redis-cluster]# cd /root/redis-3.0.0/src/

再启动命令（注意修改为自己的ip最好不用127.0.0.1和port）

[root@localhost src]# **./redis-trib.rb create --replicas 1 192.168.37.136:7001 192.168.37.136:7002 192.168.37.136:7003 192.168.37.136:7004 192.168.37.136:7005 192.168.37.136:7006**

启动信息

>>> Creating cluster

Connecting to node 192.168.37.131:7001: OK

Connecting to node 192.168.37.131:7002: OK

Connecting to node 192.168.37.131:7003: OK

Connecting to node 192.168.37.131:7004: OK

Connecting to node 192.168.37.131:7005: OK

Connecting to node 192.168.37.131:7006: OK

>>> Performing hash slots allocation on 6 nodes...

Using 3 masters:

192.168.37.131:7001

192.168.37.131:7002

192.168.37.131:7003

Adding replica 192.168.37.131:7004 to 192.168.37.131:7001

Adding replica 192.168.37.131:7005 to 192.168.37.131:7002

Adding replica 192.168.37.131:7006 to 192.168.37.131:7003

M: 955567f988830cdf4328854f815719ea28082ca8 192.168.37.131:7001

slots:0-5460 (5461 slots) master

M: 4f3eeced04b930aa48193699301745a05a70697a 192.168.37.131:7002

slots:5461-10922 (5462 slots) master

M: 871a684dbbc0f43dcc16107710d7bd2f4e6de76a 192.168.37.131:7003

slots:10923-16383 (5461 slots) master

S: baca3ce2223dfcd9c636a7193b12998b1dbb2431 192.168.37.131:7004

replicates 955567f988830cdf4328854f815719ea28082ca8

S: 0b599863ddf2e03b0326c75b874a1af8ae430d2e 192.168.37.131:7005

replicates 4f3eeced04b930aa48193699301745a05a70697a

S: 92f712d954c62f2743e2e572f6582a6ef9a163e0 192.168.37.131:7006

replicates 871a684dbbc0f43dcc16107710d7bd2f4e6de76a

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 192.168.37.131:7001)

M: 955567f988830cdf4328854f815719ea28082ca8 192.168.37.131:7001

slots:0-5460 (5461 slots) master

M: 4f3eeced04b930aa48193699301745a05a70697a 192.168.37.131:7002

slots:5461-10922 (5462 slots) master

M: 871a684dbbc0f43dcc16107710d7bd2f4e6de76a 192.168.37.131:7003

slots:10923-16383 (5461 slots) master

M: baca3ce2223dfcd9c636a7193b12998b1dbb2431 192.168.37.131:7004

slots: (0 slots) master

replicates 955567f988830cdf4328854f815719ea28082ca8

M: 0b599863ddf2e03b0326c75b874a1af8ae430d2e 192.168.37.131:7005

slots: (0 slots) master

replicates 4f3eeced04b930aa48193699301745a05a70697a

M: 92f712d954c62f2743e2e572f6582a6ef9a163e0 192.168.37.131:7006

slots: (0 slots) master

replicates 871a684dbbc0f43dcc16107710d7bd2f4e6de76a

[OK] All nodes agree about slots configuration.

>>> Check for open slots...

>>> Check slots coverage...

[OK] All 16384 slots covered.

# redis集群连接

**redis的单机版，默认是16个数据库，但是redis-Cluster集群版，只有一个数据库**

## 使用redis命令行客户端连接

[root@localhost bin]# **./redis-cli -h 192.168.37.131 -p 7006 -c**

192.168.37.131:7006> set hello money

-> Redirected to slot [866] located at 192.168.37.131:7001

OK

**一定要加-c参数,节点之间就可以互相跳转**

## redis的java客户端连接(Jedis)

使用的是JedisCluster类进行连接

**public** **class** JedisClusterTest {

**public** **static** **void** main(String[] args) {

// 1. 创建JedisCluster对象

// 设置节点信息,放到set中

Set<HostAndPort> nodes = **new** HashSet<>();

nodes.add(**new** HostAndPort("192.168.37.136", 7001));

nodes.add(**new** HostAndPort("192.168.37.136", 7002));

nodes.add(**new** HostAndPort("192.168.37.136", 7003));

nodes.add(**new** HostAndPort("192.168.37.136", 7004));

nodes.add(**new** HostAndPort("192.168.37.136", 7005));

nodes.add(**new** HostAndPort("192.168.37.136", 7006));

// 创建对象

JedisCluster jedisCluster = **new** JedisCluster(nodes);

// 2. 使用jedisCluster操作redis,方法和jedis是一样的

String key = "jedisCluster";

String setResult = jedisCluster.set(key, "Hello redis !");

String getResult = jedisCluster.get(key);

System.***out***.println(setResult + ":" + getResult);

// 3. **关闭jedisCluster连接(在程序执行完之后,才能关闭,他的内部已经封装了连接池)**

jedisCluster.close();

}

}