# Redis 5 Cluster集群

Redis Cluster集群是一个由多个主从节点组成的分布式服务器群，它具有复制、高可用和分片特性。Redis集群将所有数据存储区域划分为16384个槽位(slots)，每个节点负责一部分槽位，槽位的信息存储于每个节点中。Redis集群要将每个节点设置成集群模式，它没有中心节点，可水平扩展，它的性能和高可用性均优于主从模式和哨兵模式，而且集群配置非常简单。Redis 5集群架构图如下图所示。

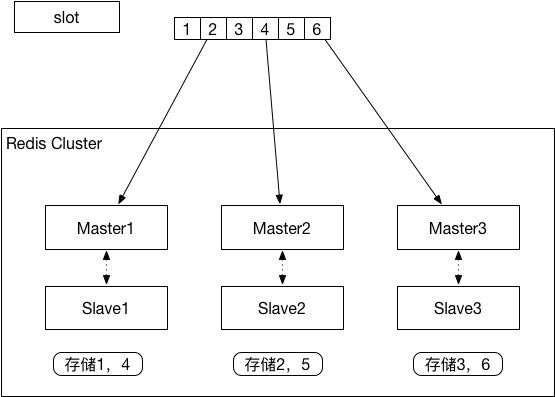


图 6‑9 Redis5集群环境

从Redis Cluster的架构图中可以很容易的看出首先将数据根据hash规则分配到6个slot(槽)中，然后根据CRC算法和取模算法将6个slot分别存储到3个不同的Master节点中，每个master节点又配套部署了一个slave节点，当一个master出现问题后，slave节点可以顶上。这种Redis Cluster方案对比哨兵模式方案的优点在于提高了读写的并发，分散了I/O，在保障高可用的前提下提高了性能。



## 集群环境

Redis集群在物理结构上是由集群上多个节点构成的，这些节点分为两类，一类叫“主节点”(Master),另一类叫”从节点”(Slave)。

Redis集群节点要求：

**1）主节点（Master）不能少于总结点的一半。**

**2）主节点（Master）至少要有3个。**

一个Redis集群正常工作至少需要要三个Master（主节点）且不能少于总节点的一半，本集群环境使用一台节点服务器，在这台服务器开启6个Redis实例，每个Redis实例占用1个端口，模拟3个Master(主节点)和3个Slave(从节点）环境，组成一个Redis Cluster(集群），这里用一台机器部署6个Redis实例，也可以多台机器部署Redis集群，修改Redis配置文件的ip地址就可以了。本集群实验采用三主三从模式，每个Master(主节点)处理各自的数据，提供读写能力，每个Slave(从节点)异步复制主节点的数据。

表格 6‑3 Redis 5集群的实验环境

|  |  |  |  |
| --- | --- | --- | --- |
| IP地址 | 端口 | 版本 | 操作系统 |
| 192.168.11.11 | 8001 | Redis 5.0.5 | CentOS7 64位 |
| 8002 |
| 8003 |
| 8004 |
| 8005 |
| 8006 |

以root用户登录Linux服务器。

## 开始Redis 5 集群搭建

第一步：安装Redis 5

1，首先，安装gcc基础依赖包，使用如下命令。

$ yum install gcc-c++

2, 下载Redis 5并解压缩

$ wget <http://download.redis.io/releases/redis-5.0.5.tar.gz>

解压缩Redis 5压缩包

$ tar -xzvf redis-5.0.5.tar.gz

进入解压后的目录

$ cd redis-5.0.5/

使用make命令编译Redis源文件。

$ make MALLOC=libc

编译成功后，安装Redis 。

$ make install PREFIX=/usr/local/redis

安装成功后，需要对Redis 进行部署，把Redis 的配置文件 redis.conf拷贝到 “/usr/local/redis/conf”目录下。

$ mkdir /usr/local/redis/conf

$ cp redis.conf /usr/local/redis/conf

3，配置Redis的命令

最后需要将Redis的命令所在目录添加到系统参数PATH中,修改/etc/profile文件。

$ vi /etc/profile

在 /etc/profile文件最后一行添加以下内容。

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

然后使用source 命令使/etc/profile这个文件立即生效。

$ source /etc/profile

至此，Redis 5在Linux上的安装和配置就结束了。

第二步：在 /usr/local下创建文件夹redis-cluster，然后在其下面创建6个文件夹如下。

$ mkdir -p /usr/local/redis-cluster

$ cd /usr/local/redis-cluster/

$ mkdir 8001 8002 8003 8004 8005 8006

第三步：把之前的redis.conf配置文件copy到8001目录下。

$ cp /usr/local/redis/conf/redis.conf /usr/local/redis-cluster/8001

修改redis.conf为如下内容。

1. daemonize yes
2. port 8001 分别对每个节点机器的端口号进行设置
3. dir /usr/local/redis-cluster/8001/ 设定数据文件存放位置，必须要指定不同的目录位置，不然会丢失数据
4. cluster-enabled yes 启动集群模式
5. cluster-config-file nodes-8001.conf 集群节点信息文件，这里nodes-8001.conf最好和port对应上
6. cluster-node-timeout 5000
7. bind 192.168.11.11 修改为主机的IP，默认ip为127.0.0.1 ，需要改为其它节点机器可访问的ip，否则创建集群时无法访问对应机器的端口，无法创建集群。
8. protected-mode no 关闭保护模式
9. appendonly yes

如果要设置密码需要增加如下配置：

1. requirepass xxx 设置Redis的访问密码
2. masterauth xxx 设置集群节点间的访问密码，跟第10项里一致

第四步：把修改后的redis.conf配置文件，复制到8002，8003,8004,8005和8006，修改第2、3、5项里的端口号。

第三步完成了一个节点的配置了，其实接下来就是机械化的再完成另外五个节点，其实可以这么做：把 8001 实例 复制到另外五个文件夹中，唯一要修改的就是 redis.conf 中的所有和端口的相关的信息即可，其实就那么三个位置。

$ cd /usr/local/redis-cluster/8001

$ cp redis.conf /usr/local/redis-cluster/8002

$ cp redis.conf /usr/local/redis-cluster/8003

$ cp redis.conf /usr/local/redis-cluster/8004

$ cp redis.conf /usr/local/redis-cluster/8005

$ cp redis.conf /usr/local/redis-cluster/8006

第五步：分别启动6个Redis实例，然后检查是否启动成功。

$ redis-server /usr/local/redis-cluster/8001/redis.conf

$ redis-server /usr/local/redis-cluster/8002/redis.conf

$ redis-server /usr/local/redis-cluster/8003/redis.conf

$ redis-server /usr/local/redis-cluster/8004/redis.conf

$ redis-server /usr/local/redis-cluster/8005/redis.conf

$ redis-server /usr/local/redis-cluster/8006/redis.conf

执行启动Redis实例命令的返回结果如下所示。

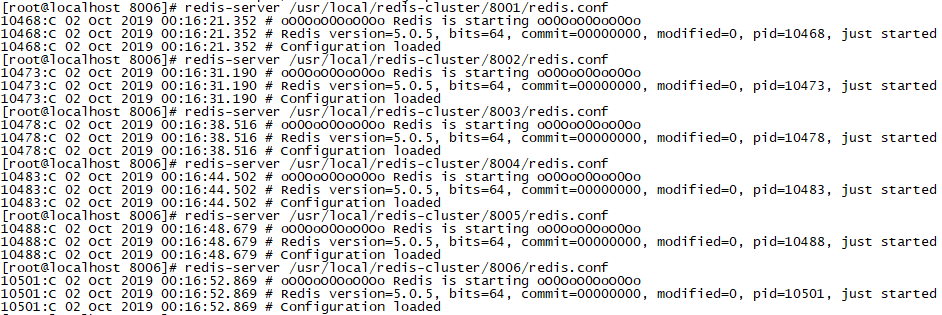


图 6‑10 启动6个Redis实例

使用命令ps -ef | grep redis，查看Redis是否启动成功，可以看出6个节点已经全部启动成功了。

[root@localhost 8006]# ps -ef | grep redis

root 10469 1 0 00:16 ? 00:00:00 redis-server 127.0.0.1:8001 [cluster]

root 10474 1 0 00:16 ? 00:00:00 redis-server 127.0.0.1:8002 [cluster]

root 10479 1 0 00:16 ? 00:00:00 redis-server 127.0.0.1:8003 [cluster]

root 10484 1 0 00:16 ? 00:00:00 redis-server 127.0.0.1:8004 [cluster]

root 10489 1 0 00:16 ? 00:00:00 redis-server 127.0.0.1:8005 [cluster]

root 10502 1 0 00:16 ? 00:00:00 redis-server 127.0.0.1:8006 [cluster]

root 10526 2993 0 00:19 pts/1 00:00:00 grep --color=auto redis

一个个命令启动Redis实例比较麻烦，可以使用脚本的方式启动Redis实例，使用以下命令创建启动Redis集群脚本。

$ touch startRedisCluster.sh

$ chmod +x startRedisCluster.sh

使用vi startRedisCluster.sh命令修改startRedisCluster.sh脚本，添加以下内容。

redis-server /usr/local/redis-cluster/8001/redis.conf

redis-server /usr/local/redis-cluster/8002/redis.conf

redis-server /usr/local/redis-cluster/8003/redis.conf

redis-server /usr/local/redis-cluster/8004/redis.conf

redis-server /usr/local/redis-cluster/8005/redis.conf

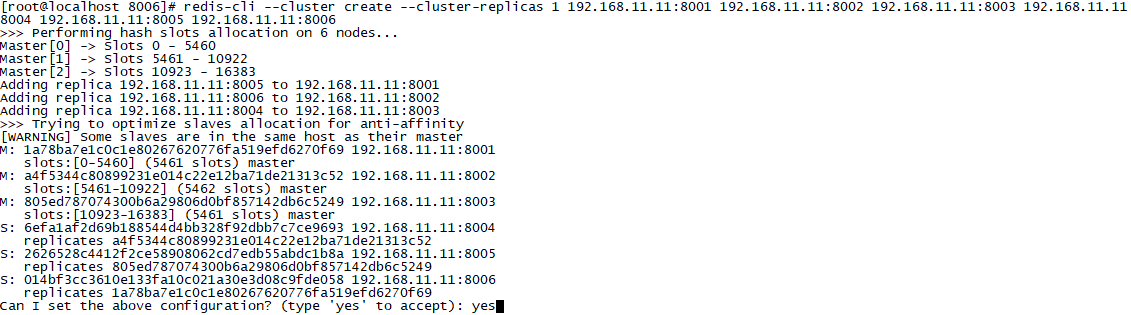
redis-server /usr/local/redis-cluster/8006/redis.conf

使用以下命令启动脚本。

$ ./startRedisCluster.sh

第六步：用redis-cli创建整个Redis集群(redis5以前版本的集群是依靠ruby脚本redis-trib.rb实现的)

$ redis-cli --cluster create --cluster-replicas 1 192.168.11.11:8001 192.168.11.11:8002 192.168.11.11:8003 192.168.11.11:8004 192.168.11.11:8005 192.168.11.11:8006



按照提示输入yes。

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 192.168.11.11:8001)

M: 1a78ba7e1c0c1e80267620776fa519efd6270f69 192.168.11.11:8001

slots:[0-5460] (5461 slots) master

1 additional replica(s)

S: 2626528c4412f2ce58908062cd7edb55abdc1b8a 127.0.0.1:8005

slots: (0 slots) slave

replicates 805ed787074300b6a29806d0bf857142db6c5249

S: 6efa1af2d69b188544d4bb328f92dbb7c7ce9693 127.0.0.1:8004

slots: (0 slots) slave

replicates a4f5344c80899231e014c22e12ba71de21313c52

M: 805ed787074300b6a29806d0bf857142db6c5249 127.0.0.1:8003

slots:[10923-16383] (5461 slots) master

1 additional replica(s)

S: 014bf3cc3610e133fa10c021a30e3d08c9fde058 127.0.0.1:8006

slots: (0 slots) slave

replicates 1a78ba7e1c0c1e80267620776fa519efd6270f69

M: a4f5344c80899231e014c22e12ba71de21313c52 127.0.0.1:8002

slots:[5461-10922] (5462 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.

以上是集群信息，表示为每个创建的主服务器节点创建一个从服务器节点，可以转换为以下的表格形式。

|  |  |  |  |
| --- | --- | --- | --- |
| 端口号 | 角色 | 节点ID | 插槽范围(Slots) |
| 8001 | master | 1a78ba7e1c0c1e80267620776fa519efd6270f69 | 0-5460 |
| 8006 | slave | 014bf3cc3610e133fa10c021a30e3d08c9fde058 | -- |
| 8003 | master | 805ed787074300b6a29806d0bf857142db6c5249 | 10923-16383 |
| 8005 | slave | 2626528c4412f2ce58908062cd7edb55abdc1b8a | -- |
| 8002 | master | a4f5344c80899231e014c22e12ba71de21313c52 | 5461-10922 |
| 8004 | slave | 6efa1af2d69b188544d4bb328f92dbb7c7ce9693 | -- |

第七步：验证集群。

1. 连接任意一个客户端即可：

redis-cli -c -a xxx -h 192.168.11.100 -p 8001

redis-cli 的命令参数 -a表示访问服务端密码，-c表示集群模式，然后指定ip地址和端口号。如果Redis集群没有设置密码可以省略-a参数。

例如 使用命令redis-cli -c -h 192.168.11.11 -p 8001访问Redis集群。

[root@localhost redis-5.0.5]# **redis-cli -c -h 192.168.11.11 -p 8001**

192.168.11.11:8001> SET name xinping

-> Redirected to slot [5798] located at 127.0.0.1:8002

OK

127.0.0.1:8002> GET name

"xinping"

注意这里进入到实例为8002的节点了，进行了跳转重定向(redirected)。

1. 进行验证： cluster info（查看集群信息）、cluster nodes（查看节点列表）

使用cluster info命令查看集群信息。

192.168.11.11:8001> cluster info

cluster\_state:ok

cluster\_slots\_assigned:16384

cluster\_slots\_ok:16384

cluster\_slots\_pfail:0

cluster\_slots\_fail:0

cluster\_known\_nodes:6

cluster\_size:3

cluster\_current\_epoch:6

cluster\_my\_epoch:1

cluster\_stats\_messages\_ping\_sent:4178

cluster\_stats\_messages\_pong\_sent:4163

cluster\_stats\_messages\_sent:8341

cluster\_stats\_messages\_ping\_received:4158

cluster\_stats\_messages\_pong\_received:4178

cluster\_stats\_messages\_meet\_received:5

cluster\_stats\_messages\_received:8341

使用cluster nodes查看节点列表

192.168.11.11:8001> cluster nodes

2626528c4412f2ce58908062cd7edb55abdc1b8a 127.0.0.1:8005@18005 slave 805ed787074300b6a29806d0bf857142db6c5249 0 1569949378000 5 connected

6efa1af2d69b188544d4bb328f92dbb7c7ce9693 127.0.0.1:8004@18004 slave a4f5344c80899231e014c22e12ba71de21313c52 0 1569949378000 4 connected

805ed787074300b6a29806d0bf857142db6c5249 127.0.0.1:8003@18003 master - 0 1569949379597 3 connected 10923-16383

014bf3cc3610e133fa10c021a30e3d08c9fde058 127.0.0.1:8006@18006 slave 1a78ba7e1c0c1e80267620776fa519efd6270f69 0 1569949377281 6 connected

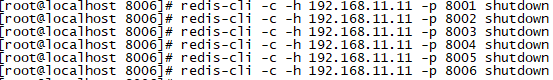
a4f5344c80899231e014c22e12ba71de21313c52 127.0.0.1:8002@18002 master - 0 1569949378791 2 connected 5461-10922

1a78ba7e1c0c1e80267620776fa519efd6270f69 192.168.11.11:8001@18001 myself,master - 0 1569949377000 1 connected 0-5460

1. 关闭集群需要逐个关闭Redis实例，使用命令如下：

$ /usr/local/redis/bin/redis-cli -a xxx -c -h 192.168.11.60 -p 8001 shutdown

在本例中可以使用以下命令停止六个Redis实例。



在此做一个实验，使用以下命令停止端口为8001的Redis实例。

$ redis-cli -c -h 192.168.11.11 -p 8001 shutdown

在查看Redis进程。

[root@localhost ~]# ps -ef | grep redis

root 2508 1 0 14:17 ? 00:00:00 redis-server 127.0.0.1:8002 [cluster]

root 2513 1 0 14:17 ? 00:00:00 redis-server 127.0.0.1:8003 [cluster]

root 2518 1 0 14:17 ? 00:00:00 redis-server 127.0.0.1:8004 [cluster]

root 2525 1 0 14:17 ? 00:00:00 redis-server 127.0.0.1:8005 [cluster]

root 2532 1 0 14:17 ? 00:00:00 redis-server 127.0.0.1:8006 [cluster]

root 2562 2425 0 14:18 pts/0 00:00:00 redis-cli -c -h 192.168.11.11 -p 8001

root 2934 2865 0 14:19 pts/1 00:00:00 grep --color=auto redis

在连接上Redis集群查看集群信息。

[root@localhost ~]# redis-cli -c -h 192.168.11.11 -p 8002

192.168.11.11:8002> cluster nodes

a4f5344c80899231e014c22e12ba71de21313c52 127.0.0.1:8002@18002 myself,master - 0 1569997240000 2 connected 5461-10922

014bf3cc3610e133fa10c021a30e3d08c9fde058 127.0.0.1:8006@18006 master - 0 1569997241556 7 connected 0-5460

2626528c4412f2ce58908062cd7edb55abdc1b8a 127.0.0.1:8005@18005 slave 805ed787074300b6a29806d0bf857142db6c5249 0 1569997241000 5 connected

6efa1af2d69b188544d4bb328f92dbb7c7ce9693 127.0.0.1:8004@18004 slave a4f5344c80899231e014c22e12ba71de21313c52 0 1569997240551 4 connected

1a78ba7e1c0c1e80267620776fa519efd6270f69 127.0.0.1:8001@18001 master,**fail** - 1569997180503 1569997180000 1 disconnected

805ed787074300b6a29806d0bf857142db6c5249 127.0.0.1:8003@18003 master - 0 1569997241000 3 connected 10923-16383

可以看出端口为8001的Redis实例已经关闭，如果想打开8001的Redis实例，需要使用如下命令。

$ redis-server /usr/local/redis-cluster/8001/redis.conf

至此，Redis 5集群环境搭建完毕。

## Redis集群特点

Redis Cluster集群方案采用的是虚拟槽分区，槽范围是0~16383,有16384个槽，槽有3个主节点，每个节点大致负责5500个槽的读写，每个节点会维护自身负责的虚拟槽。

键所对应的哈希值通过公式计算： CRC16(key) % 16384 。在集群的实际使用时，set key value 命令会计算hash值，把key-value设置到对应的主节点上。

Redis Cluster集群采用了哈希分区的“虚拟槽分区”方式，所有的键根据哈希函数(CRC16[key] & 16383)映射到0－16383槽内，一共有16384个槽位，每个节点维护部分槽及槽所映射的键值数据，哈希函数: CRC16[key] & 16383 进行按位与操作。槽与节点的关系如下

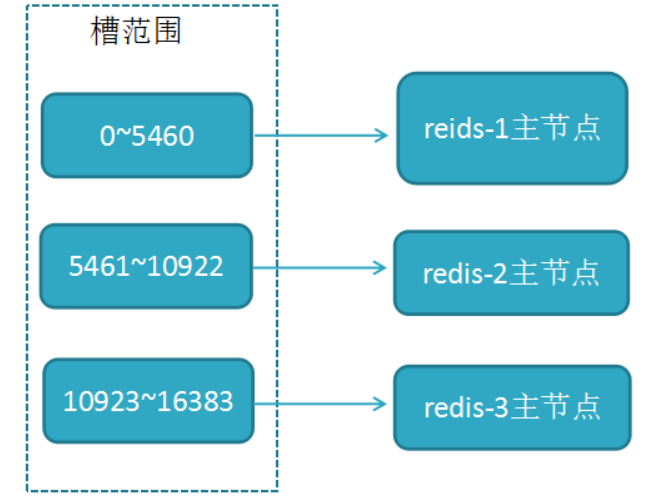


图 6‑11 Redis集群槽与节点的关系

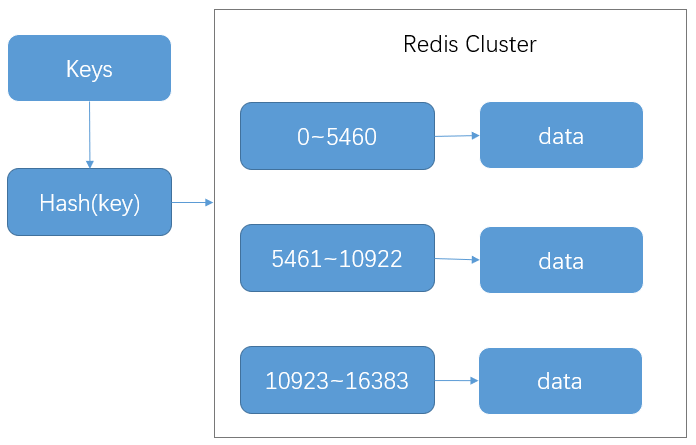


图 6‑12 用hash函数将键映射到槽，再由槽指向数据

通过Redis的cluster nodes命令可以查看当前集群的信息，查看节点列表。

[root@localhost src]# redis-cli -h 192.168.11.11 -p 7000

192.168.11.10:7000>  **cluster nodes**

2113489e118b5ae2ee6ac4ec1b88a703cde8b331 192.168.11.11:7004@17004 slave e4bafc33f3dbc2aa234c482f997428e91159b45a 0 1569302075000 4 connected

e4bafc33f3dbc2aa234c482f997428e91159b45a 192.168.11.10:7001@17001 master - 0 1569302074482 2 connected 10923-16383

5dc15d549306570a6e71f33c1c0f302bebe103a4 192.168.11.10:7002@17002 slave df7b3ff4bc7735bc4ddc0933374d628f75cc250f 0 1569302075490 4 connected

df7b3ff4bc7735bc4ddc0933374d628f75cc250f 192.168.11.11:7003@17003 master - 0 1569302076499 4 connected 5461-10922

9ed6d860785f2e4a0bf7761abe9a0b6f3280a4d6 192.168.11.10:7000@17000 myself,master - 0 1569302074000 8 connected 0-5460

b1cfe7a57700ac29666d825c20db65c52123d8fc 192.168.11.11:7005@17005 slave e4bafc33f3dbc2aa234c482f997428e91159b45a 0 1569302075593 2 connected

该信息反映出了集群中的每个节点的id、身份、连接数和插槽数等。

当我们执行SET age 23命令时，Redis是如何将数据保存到集群中的呢？

192.168.11.11:8002> SET age 23

-> Redirected to slot [741] located at 127.0.0.1:8001

OK

执行步骤如下：

1、 接收命令SET age 23

2、 通过key（age）计算出插槽值，然后根据插槽值找到对应的节点。（age的插槽值为：741）

3、 重定向到该节点执行命令

整个Redis集群提供了**16384**个插槽，也就是说集群中的每个节点分得的插槽数总和为16384。./redis-trib.rb 脚本实现了将16384个插槽平均分配给了N个节点。如下图所示

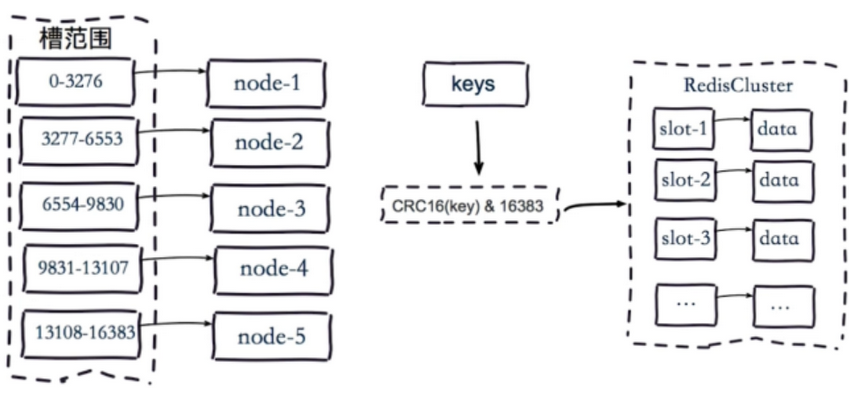


图 6‑13 虚拟槽分配

Apl注意

如果插槽数有部分是没有指定到节点的，那么这部分插槽所对应的key将不能使用。

Redis Cluster的数据分散度高，键值分布与业务无关，键值无法顺序访问，支持批量操作。

使用cluster info命令查看Redis集群信息。

192.168.11.11:7000> cluster info

cluster\_state:ok

cluster\_slots\_assigned:16384

cluster\_slots\_ok:16384

cluster\_slots\_pfail:0

cluster\_slots\_fail:0

cluster\_known\_nodes:6

cluster\_size:3

cluster\_current\_epoch:8

cluster\_my\_epoch:8

cluster\_stats\_messages\_ping\_sent:14321

cluster\_stats\_messages\_pong\_sent:6764

cluster\_stats\_messages\_fail\_sent:3

cluster\_stats\_messages\_sent:21088

cluster\_stats\_messages\_ping\_received:6759

cluster\_stats\_messages\_pong\_received:6678

cluster\_stats\_messages\_meet\_received:5

cluster\_stats\_messages\_fail\_received:3

cluster\_stats\_messages\_received:13445

## 高可用集群的水平扩展

在上一节中我们把六个节点分布在一台机器上，采用三主三从的模式，本节进行集群的水平扩展，新增两个节点，一个主节点一个从节点。类似于下图所示。

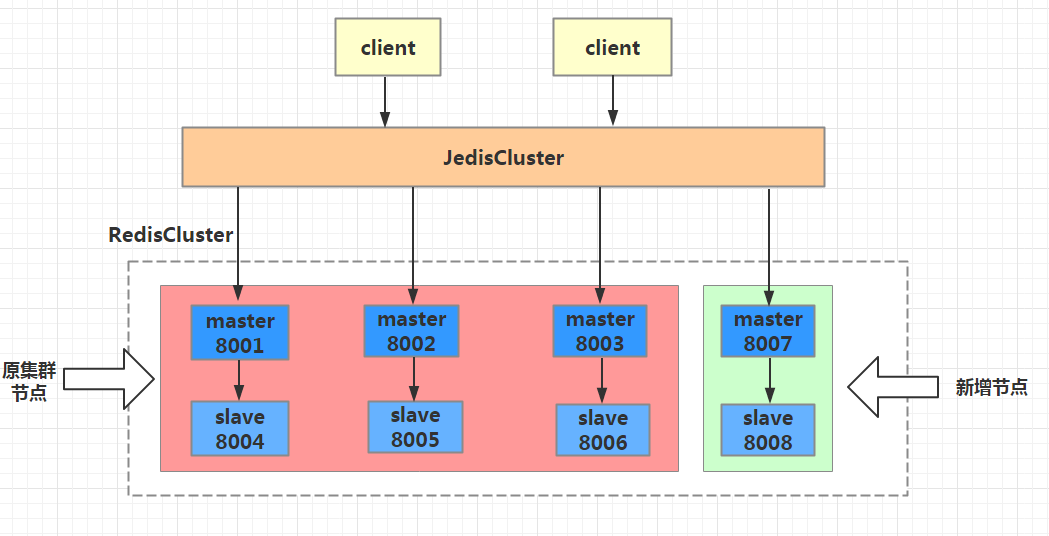


图 6‑14 Redis集群水平扩展

分别启动六个Redis节点

$ redis-server /usr/local/redis-cluster/8001/redis.conf

$ redis-server /usr/local/redis-cluster/8002/redis.conf

$ redis-server /usr/local/redis-cluster/8003/redis.conf

$ redis-server /usr/local/redis-cluster/8004/redis.conf

$ redis-server /usr/local/redis-cluster/8005/redis.conf

$ redis-server /usr/local/redis-cluster/8006/redis.conf

或者调用启动脚本startRedisCluster.sh启动Redis集群。

$ ./startRedisCluster.sh

然后查看集群启动情况，如下图所示。

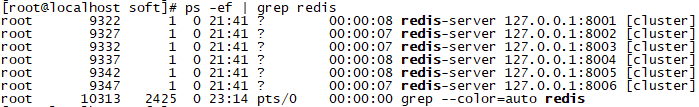


图 6‑15 查看Redis集群的启动情况

客户端连接8001端口的Redis实例

[root@localhost soft]# redis-cli -c -h 192.168.11.11 -p 8001

192.168.11.11:8001>

查看就群状态

192.168.11.11:8001> cluster info

cluster\_state:ok

cluster\_slots\_assigned:16384

cluster\_slots\_ok:16384

cluster\_slots\_pfail:0

cluster\_slots\_fail:0

cluster\_known\_nodes:6

cluster\_size:3

cluster\_current\_epoch:7

cluster\_my\_epoch:7

cluster\_stats\_messages\_ping\_sent:10680

cluster\_stats\_messages\_pong\_sent:10523

cluster\_stats\_messages\_sent:21203

cluster\_stats\_messages\_ping\_received:10523

cluster\_stats\_messages\_pong\_received:10675

cluster\_stats\_messages\_received:21198

查看节点信息

192.168.11.11:8001> cluster nodes

014bf3cc3610e133fa10c021a30e3d08c9fde058 127.0.0.1:8006@18006 master - 0 1570029551121 7 connected 0-5460

805ed787074300b6a29806d0bf857142db6c5249 127.0.0.1:8003@18003 master - 0 1570029551322 3 connected 10923-16383

1a78ba7e1c0c1e80267620776fa519efd6270f69 192.168.11.11:8001@18001 myself,slave 014bf3cc3610e133fa10c021a30e3d08c9fde058 0 1570029549000 1 connected

6efa1af2d69b188544d4bb328f92dbb7c7ce9693 127.0.0.1:8004@18004 slave a4f5344c80899231e014c22e12ba71de21313c52 0 1570029551523 4 connected

a4f5344c80899231e014c22e12ba71de21313c52 127.0.0.1:8002@18002 master - 0 1570029552328 2 connected 5461-10922

2626528c4412f2ce58908062cd7edb55abdc1b8a 127.0.0.1:8005@18005 slave 805ed787074300b6a29806d0bf857142db6c5249 0 1570029551000 5 connected

上图可以看出，整个集群运行正常，整个集群包含三个Master节点和三个Slave节点

* 8006端口的实例节点存储0-5460这些hash槽
* 8002端口的实例节点存储5461-10922这些hash槽
* 8003端口的实例节点存储10923-16383这些hash槽

这三个Master节点存储的所有hash槽组成Redis集群的存储槽位，Slave点是每个主节点的备份从节点，不显示存储槽位。

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。

使用哈希槽的可以方便的添加或移除节点。当需要增加节点时，只需要把其他节点的某些哈希槽挪到新节点就可以了。当需要移除节点时，只需要把移除节点上的哈希槽挪到其他节点就行了。新增或移除节点的时候不需要先停掉所有的 Redis 服务。

Redis Cluster使用的是自己做的crc16的简单hash算法，没有用一致性hash算法。Redis的作者认为它的crc16(key) mod 16384的效果已经不错了，虽然没有一致性hash灵活，但实现很简单，节点增删时处理起来也很方便。

Redis集群总共有2的14次方，16384个哈希槽。当往Redis Cluster中加入一个Key时，会根据crc16(key) mod 16384计算这个key应该分布到哪个hash slot中，一个hash slot中会有很多key和value。可以理解成表的分区，使用单节点时的redis时只有一个表，所有的key都放在这个表里；改用Redis Cluster以后会自动生成16384个分区表，当insert数据时会根据上面的简单算法来决定key应该存在哪个分区，每个分区里有很多key。

在原始集群基础上再增加一个主节点(8007)和一个从节点(8008)，结构如下图

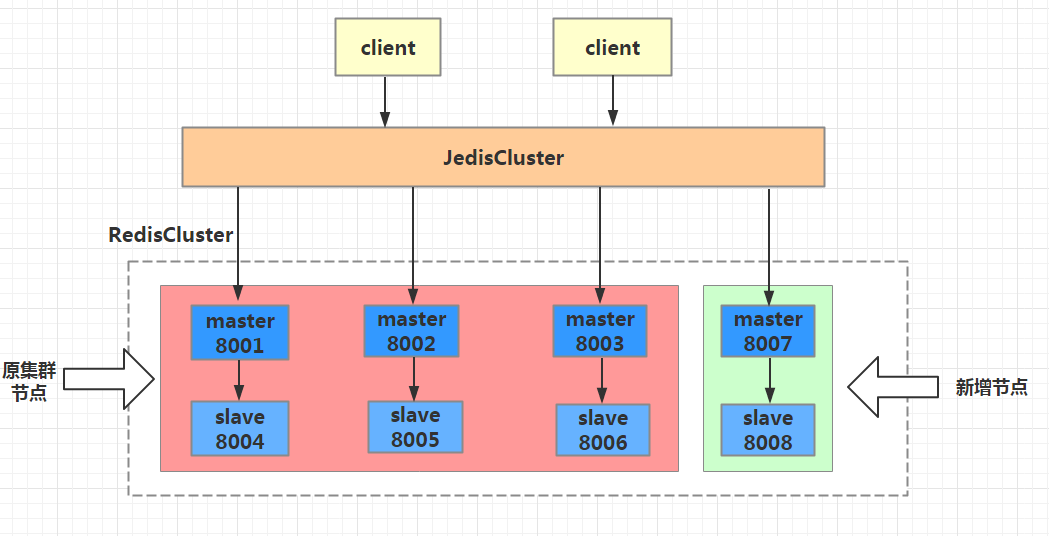


图 6‑16 Redis集群添加节点

1，增加Redis实例

在/usr/local/redis-cluster下创建8007和8008文件夹，并拷贝8001文件夹下的redis.conf文件到8007和8008这两个文件夹下。

$ cd /usr/local/redis-cluster/

$ mkdir 8007 8008

$ cp /usr/local/redis/conf/redis.conf /usr/local/redis-cluster/8007

$ cp /usr/local/redis/conf/redis.conf /usr/local/redis-cluster/8008

按之前的方法修改8007、8008文件夹中redis.conf配置文件的参数，修改完成后进行启动。

[root@localhost 8008]# redis-server /usr/local/redis-cluster/8007/redis.conf

10764:C 02 Oct 2019 23:51:23.582 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo

10764:C 02 Oct 2019 23:51:23.582 # Redis version=5.0.5, bits=64, commit=00000000, modified=0, pid=10764, just started

10764:C 02 Oct 2019 23:51:23.582 # Configuration loaded

[root@localhost 8008]# redis-server /usr/local/redis-cluster/8008/redis.conf

10769:C 02 Oct 2019 23:51:27.998 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo

10769:C 02 Oct 2019 23:51:27.998 # Redis version=5.0.5, bits=64, commit=00000000, modified=0, pid=10769, just started

10769:C 02 Oct 2019 23:51:27.998 # Configuration loaded

查询Redis集群启动情况

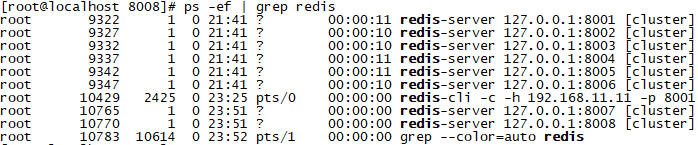


图 6‑17 查看Redis进程

这时候客户端连接8001端口的Redis实例，查看节点信息，会发现并无8007、8008节点信息。

[root@localhost 8008]# redis-cli -c -h 192.168.11.11 -p 8001

192.168.11.11:8001> cluster nodes

014bf3cc3610e133fa10c021a30e3d08c9fde058 127.0.0.1:8006@18006 master - 0 1570032821000 7 connected 0-5460

805ed787074300b6a29806d0bf857142db6c5249 127.0.0.1:8003@18003 master - 0 1570032820000 3 connected 10923-16383

1a78ba7e1c0c1e80267620776fa519efd6270f69 192.168.11.11:8001@18001 myself,slave 014bf3cc3610e133fa10c021a30e3d08c9fde058 0 1570032819000 1 connected

6efa1af2d69b188544d4bb328f92dbb7c7ce9693 127.0.0.1:8004@18004 slave a4f5344c80899231e014c22e12ba71de21313c52 0 1570032821590 4 connected

a4f5344c80899231e014c22e12ba71de21313c52 127.0.0.1:8002@18002 master - 0 1570032821388 2 connected 5461-10922

2626528c4412f2ce58908062cd7edb55abdc1b8a 127.0.0.1:8005@18005 slave 805ed787074300b6a29806d0bf857142db6c5249 0 1570032821085 5 connected

2，添加主节点

配置8007为集群主节点需要使用add-node命令新增一个主节点8007，8007为新增节点，8001为已知存在节点，看到日志最后有"[OK] New node added correctly"提示代表新节点加入成功。

[root@localhost 8008]# **redis-cli --cluster add-node 192.168.11.11:8007** 192.168.11.11:8001

>>> Adding node 192.168.11.11:8007 to cluster 192.168.11.11:8001

>>> Performing Cluster Check (using node 192.168.11.11:8001)

S: 1a78ba7e1c0c1e80267620776fa519efd6270f69 192.168.11.11:8001

slots: (0 slots) slave

replicates 014bf3cc3610e133fa10c021a30e3d08c9fde058

M: 014bf3cc3610e133fa10c021a30e3d08c9fde058 127.0.0.1:8006

slots:[0-5460] (5461 slots) master

1 additional replica(s)

M: 805ed787074300b6a29806d0bf857142db6c5249 127.0.0.1:8003

slots:[10923-16383] (5461 slots) master

1 additional replica(s)

S: 6efa1af2d69b188544d4bb328f92dbb7c7ce9693 127.0.0.1:8004

slots: (0 slots) slave

replicates a4f5344c80899231e014c22e12ba71de21313c52

M: a4f5344c80899231e014c22e12ba71de21313c52 127.0.0.1:8002

slots:[5461-10922] (5462 slots) master

1 additional replica(s)

S: 2626528c4412f2ce58908062cd7edb55abdc1b8a 127.0.0.1:8005

slots: (0 slots) slave

replicates 805ed787074300b6a29806d0bf857142db6c5249

[OK] All nodes agree about slots configuration.

>>> Check for open slots...

>>> Check slots coverage...

[OK] All 16384 slots covered.

>>> Send CLUSTER MEET to node 192.168.11.11:8007 to make it join the cluster.

[OK] New node added correctly.

查看集群状态,可以看到已知节点为7个，nodes中也显示了8007节点，但是现在没有hash槽分配到8007。

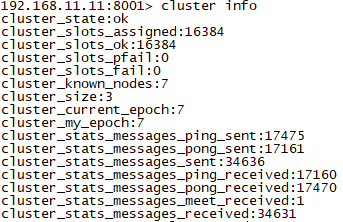


图 6‑18 查看Redis Cluster集群信息

查看Redis Cluster集群信息。

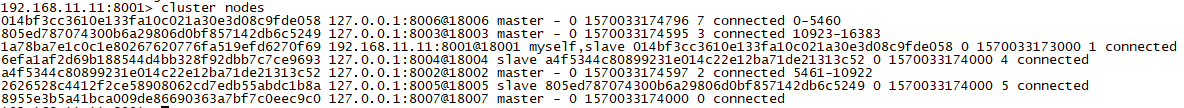


图 6‑19 查看Redis Cluster的节点信息

为新节点手工分配hash槽，使用redis-cli命令为8007分配hash槽，找到集群中的任意一个主节点(8001)，对其进行重新分片工作。

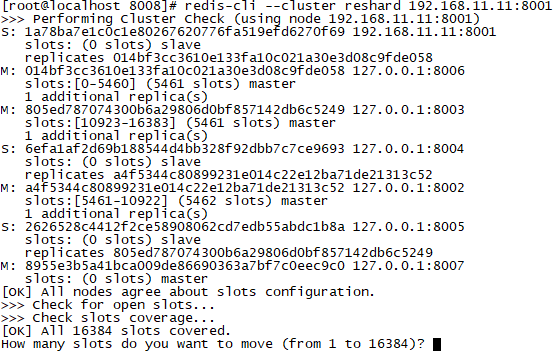
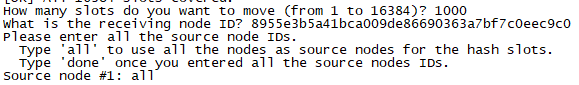


图 6‑20 Redis Cluster集群分配槽

会询问要分多少个槽出来（1000）？分给哪个节点（8007）



然后有两种方式，一种是all，可以将所有节点用作散列槽的源节点，一种是done，这种是选择从哪个节点上拿出来节点分给8007节点。all是随机的，比如说要分出1000个节点，则从3个主节点分别拿出333个，333个和334个节点分给8007节点，本案例我们选择all，将所有节点用作散列槽的源节点。

这时候我们再看一下节点信息

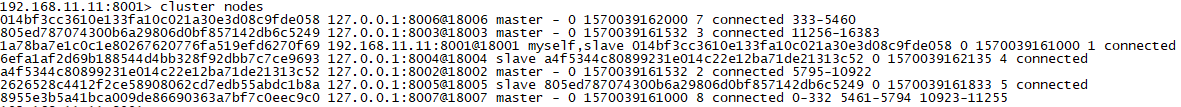


图 6‑21 查看Redis Cluster集群的节点信息

可以看到0-332 ，5461-5794已经分给8007了。

3，添加从节点

本节配置从节点8008为主节点8007的从节点，添加从节点8008到集群中去并查看集群状态。

$ **redis-cli --cluster add-node 192.168.11.11:8008 192.168.11.11:8001**

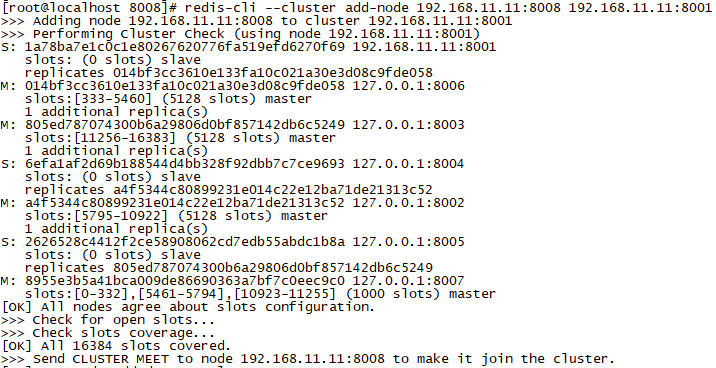


图 6‑22 Redis集群添加从节点

查看Redis节点状态

[root@localhost soft]# redis-cli -c -h 192.168.11.11 -p 8001

192.168.11.11:8001> cluster nodes

0c94890f97011ac1a479cc60991b913d38cf26ff 127.0.0.1:8008@18008 master - 0 1570039591000 0 connected

014bf3cc3610e133fa10c021a30e3d08c9fde058 127.0.0.1:8006@18006 master - 0 1570039590000 7 connected 333-5460

805ed787074300b6a29806d0bf857142db6c5249 127.0.0.1:8003@18003 master - 0 1570039591000 3 connected 11256-16383

1a78ba7e1c0c1e80267620776fa519efd6270f69 192.168.11.11:8001@18001 myself,slave 014bf3cc3610e133fa10c021a30e3d08c9fde058 0 1570039589000 1 connected

6efa1af2d69b188544d4bb328f92dbb7c7ce9693 127.0.0.1:8004@18004 slave a4f5344c80899231e014c22e12ba71de21313c52 0 1570039590000 4 connected

a4f5344c80899231e014c22e12ba71de21313c52 127.0.0.1:8002@18002 master - 0 1570039590320 2 connected 5795-10922

2626528c4412f2ce58908062cd7edb55abdc1b8a 127.0.0.1:8005@18005 slave 805ed787074300b6a29806d0bf857142db6c5249 0 1570039591832 5 connected

8955e3b5a41bca009de86690363a7bf7c0eec9c0 127.0.0.1:8007@18007 master - 0 1570039590000 8 connected 0-332 5461-5794 10923-11255

可以看到8008是一个master节点，没有被分配任何的hash槽。需要执行replicate命令来指定当前节点(从节点)的主节点id为哪个,首先需要连接新加的8008节点的客户端，然后使用集群命令进行操作，把当前的8008(slave)节点指定到一个主节点下.指定到8007节点，8007的节点id是8955e3b5a41bca009de86690363a7bf7c0eec9c0。

192.168.11.11:8008> CLUSTER REPLICATE 8955e3b5a41bca009de86690363a7bf7c0eec9c0

OK

192.168.11.11:8008> cluster nodes

a4f5344c80899231e014c22e12ba71de21313c52 127.0.0.1:8002@18002 master - 0 1570040634503 2 connected 5795-10922

014bf3cc3610e133fa10c021a30e3d08c9fde058 127.0.0.1:8006@18006 master - 0 1570040634503 7 connected 333-5460

1a78ba7e1c0c1e80267620776fa519efd6270f69 127.0.0.1:8001@18001 slave 8955e3b5a41bca009de86690363a7bf7c0eec9c0 0 1570040634503 8 connected

8955e3b5a41bca009de86690363a7bf7c0eec9c0 127.0.0.1:8007@18007 master - 0 1570040634000 8 connected 0-332 5461-5794 10923-11255

805ed787074300b6a29806d0bf857142db6c5249 127.0.0.1:8003@18003 master - 0 1570040633000 3 connected 11256-16383

6efa1af2d69b188544d4bb328f92dbb7c7ce9693 127.0.0.1:8004@18004 slave a4f5344c80899231e014c22e12ba71de21313c52 0 1570040633596 2 connected

0c94890f97011ac1a479cc60991b913d38cf26ff 127.0.0.1:8008@18008 myself,slave 8955e3b5a41bca009de86690363a7bf7c0eec9c0 0 1570040633000 0 connected

2626528c4412f2ce58908062cd7edb55abdc1b8a 127.0.0.1:8005@18005 slave 805ed787074300b6a29806d0bf857142db6c5249 0 1570040633000 3 connected

扩展Redis集群已经实现，下面进行删除节点。

4，删除从节点

用参数del-node删除从节点8008，指定删除节点ip和端口，以及节点id,本例中节点8008的id是0c94890f97011ac1a479cc60991b913d38cf26ff。

$ redis-cli --cluster del-node 192.168.11.11:8008 0c94890f97011ac1a479cc60991b913d38cf26ff

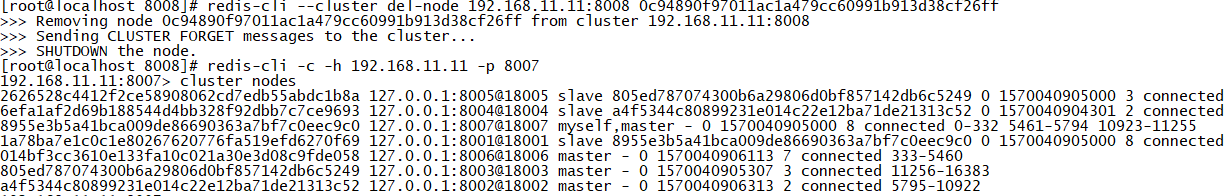


图 6‑23 Redis集群删除从节点

如上图所示，8008这个slave节点已经移除，并且该节点的redis服务也已被停止

5，删除主节点

删除之前加入的主节点8007，这个步骤相对比较麻烦一些，因为主节点的里面是有分配了hash槽的，所以必须先把8007里的hash槽放入到其他的可用主节点中去，然后再进行移除节点操作，不然会出现数据丢失问题。

$ redis-cli --cluster reshard 192.168.11.11:8007



图 6‑24 Redis集群删除主节点

以上信息在询问要将多少个哈希槽从Redis集群的主节点8007上移走，因为主节点8007上刚分配了1000 个哈希槽，所以我们这里输入1000，还有接收节点8006的id，最后输入all然后按回车键，成功的把8007主节点的数据迁移到8006上去了，我们可以看一下现在的集群状态,查看集群节点信息可以看出已经删除主节点8007了。

