Redis 6.0 集群搭建实践

mp.weixin.qq.com/s/fZZjGb5vYY_4bn4jO8U7Zg

点击上方“民工哥技术之路”，选择“设为星标”

回复“1024”获取独家整理的学习资料！

本文是Redis集群学习的实践总结（基于Redis 6.0+），详细介绍逐步搭建Redis集群环境的过程，并完成集群伸缩的实践。

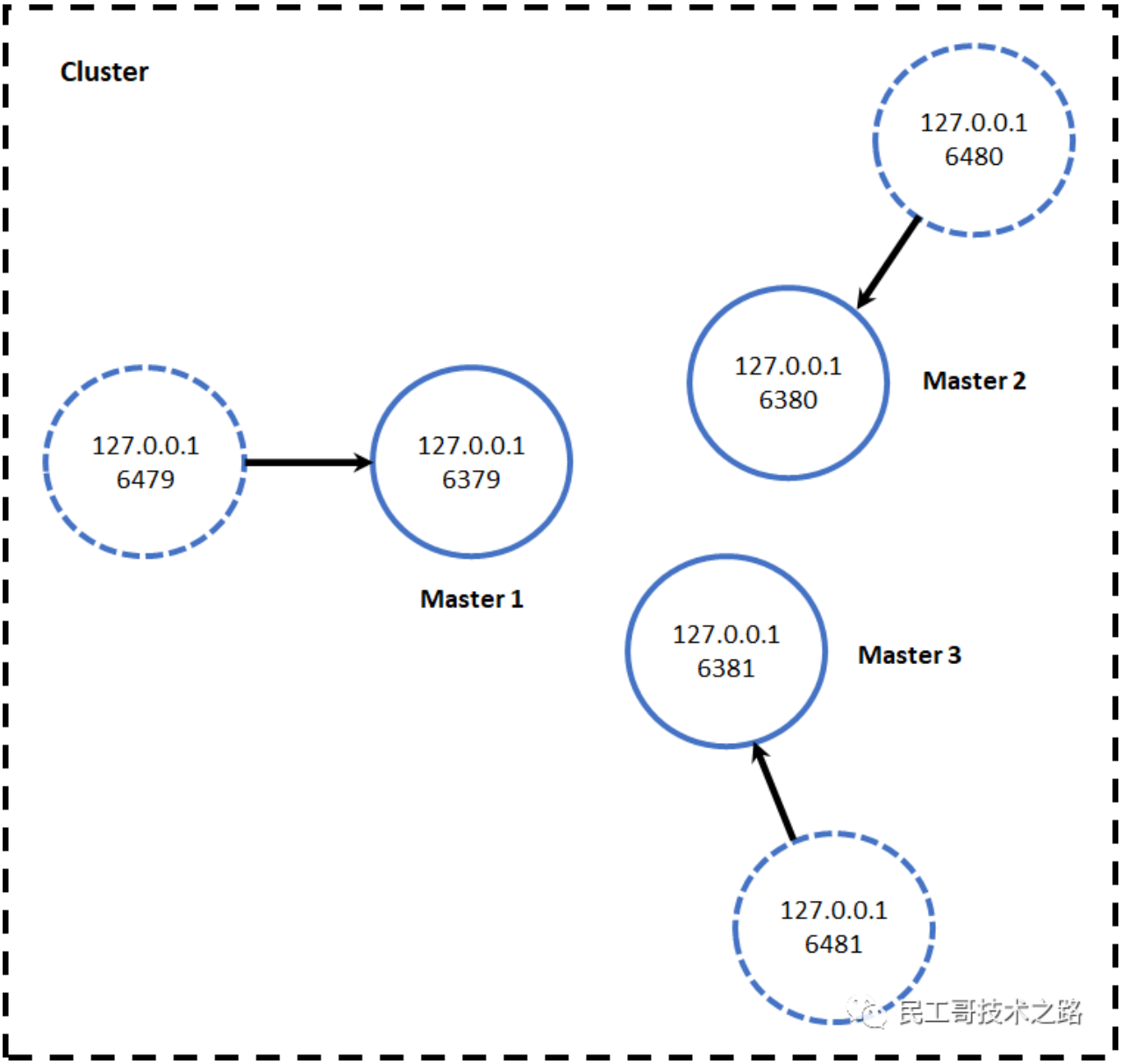
Redis集群简介

Redis集群(Redis Cluster) 是Redis提供的分布式数据库方案，通过 分片(sharding) 来进行数据共享，并提供复制和故障转移功能。相比于主从复制、哨兵模式，Redis集群实现了较为完善的高可用方案，解决了存储能力受到单机限制，写操作无法负载均衡的问题。

本文是Redis集群学习的实践总结，详细介绍逐步搭建Redis集群环境的过程，并完成集群伸缩的实践。

Redis集群环境搭建

方便起见，这里集群环境的所有节点全部位于同一个服务器上，共6个节点以端口号区分，3个主节点+3个从节点。集群的简单架构如图：



本文基于最新的Redis 6.0+，直接从github下载最新的源码编译获得常用工具 redis-server，redis-cli 。值得注意的是，从Redis 5.0以后的版本，集群管理软件 redis-trib.rb 被集成到 redis-cli 客户端工具中（详细可参考cluster-tutorial）。

本节介绍集群环境搭建时，并未借助 redis-trib.rb 快速管理，而是按照标准步骤一步步搭建，这也是为了熟悉集群管理的基本步骤。在集群伸缩实践一节将借助 redis-trib.rb 完成集群重新分片工作。

集群的搭建可以分为四步：

- 启动节点：将节点以集群方式启动，此时节点是独立的。
- 节点握手：将独立的节点连成网络。
- 槽指派：将16384个槽位分配给主节点，以达到分片保存数据库键值对的效果。
- 主从复制：为从节点指定主节点。

启动节点

每个节点初始状态仍为 Master服务器，唯一不同的是：使用 Cluster 模式启动。需要对配置文件进行修改，以端口号为6379的节点为例，主要修改如下几项：

```
# redis_6379_cluster.conf
port 6379
cluster-enabled yes
cluster-config-file "node-6379.conf"
logfile "redis-server-6379.log"
dbfilename "dump-6379.rdb"
daemonize yes
```

其中 cluster-config-file 参数指定了集群配置文件的位置，每个节点在运行过程中，会维护一份集群配置文件；每当集群信息发生变化时（如增减节点），集群内所有节点会将最新信息更新到该配置文件；当节点重启后，会重新读取该配置文件，获取集群信息，可以方便的重新加入到集群中。也就是说，当Redis节点以集群模式启动时，会首先寻找是否有集群配置文件，如果有则使用文件中的配置启动，如果没有，则初始化配置并将配置保存到文件中。集群配置文件由Redis节点维护，不需要人工修改。

为6个节点修改好相应的配置文件后，即可利用 redis-server redis_xxxx_cluster.conf 工具启动6个服务器（xxxx表示端口号，对应相应的配置文件）。利用ps命令查看进程：

```
$ ps -aux | grep redis
... 800  0.1  0.0  49584  2444 ?        Ssl  20:42   0:00 redis-
server 127.0.0.1:6379 [cluster]
... 805  0.1  0.0  49584  2440 ?        Ssl  20:42   0:00 redis-
server 127.0.0.1:6380 [cluster]
... 812  0.3  0.0  49584  2436 ?        Ssl  20:42   0:00 redis-
server 127.0.0.1:6381 [cluster]
... 817  0.1  0.0  49584  2432 ?        Ssl  20:43   0:00 redis-
server 127.0.0.1:6479 [cluster]
... 822  0.0  0.0  49584  2380 ?        Ssl  20:43   0:00 redis-
server 127.0.0.1:6480 [cluster]
... 827  0.5  0.0  49584  2380 ?        Ssl  20:43   0:00 redis-
server 127.0.0.1:6481 [cluster]
```

节点握手

将上面的每个节点启动后，节点间是相互独立的，他们都处于一个只包含自己的集群当中，以端口号6379的服务器为例，利用 CLUSTER NODES 查看当前集群包含的节点。

```
127.0.0.1:6379> CLUSTER NODES
37784b3605ad216fa93e976979c43def42bf763d :6379@16379 myself,master - 0 0 0 connecte
```

我们需要将各个独立的节点连接起来，构成一个包含多个节点的集群，使用 CLUSTER MEET 命令。

```
$ redis-cli -p 6379 -c          # -c 选项指定以Cluster模式运行redis-
cli 127.0.0.1:6379> CLUSTER MEET 127.0.0.1 6380 OK 127.0.0.1:6379> CLUSTER MEET 127
```

再次查看此时集群中包含的节点情况：

```
127.0.0.1:6379> CLUSTER NODES
c47598b25205cc88abe2e5094d5bfd9ea202335f 127.0.0.1:6380@16380 master - 0 1603632309
87b7dfacde34b3cf57d5f46ab44fd6fffb2e4f52 127.0.0.1:6379@16379 myself,master - 0 160
51081a64ddb3ccf5432c435a8cf20d45ab795dd8 127.0.0.1:6381@16381 master - 0 1603632310
9d587b75bdaed26ca582036ed706df8b2282b0aa 127.0.0.1:6481@16481 master - 0 1603632309
4c23b25bd4bcef7f4b77d8287e330ae72e738883 127.0.0.1:6479@16479 master - 0 1603632308
32ed645a9c9d13ca68dba5a147937fb1d05922ee 127.0.0.1:6480@16480 master - 0 1603632311
```

可以发现此时6个节点均作为主节点加入到集群中， CLUSTER NODES 返回的结果各项含义如下：

```
<id> <ip:port@cport> <flags> <master> <ping-sent> <pong-recv> <config-
epoch> <link-state> <slot> <slot> ... <slot>
```

- 节点id: 由40个16进制字符串组成，节点id只在集群初始化时创建一次，然后保存到集群配置文件（即前文提到的cluster-config-file）中，以后节点重新启动时会直接在集群配置文件中读取。
- port@cport: 前者为普通端口，用于为客户端提供服务；后者为集群端口，分配方法为：普通端口+10000，只用于节点间的通讯。

其余各项的详细解释可以参考官方文档cluster nodes。

槽指派

Redis集群通过分片(sharding)的方式保存数据库的键值对，整个数据库被分为16384个槽(slot)，数据库每个键都属于这16384个槽的一个，集群中的每个节点都可以处理0个或者最多16384个slot。

槽是数据管理和迁移的基本单位。当数据库中的16384个槽都分配了节点时，集群处于上线状态（ok）；如果有任意一个槽没有分配节点，则集群处于下线状态（fail）。

注意，只有主节点有处理槽的能力，如果将槽指派步骤放在主从复制之后，并且将槽位分配给从节点，那么集群将无法正常工作（处于下线状态）。

利用 CLUSTER ADDSLOTS

```
redis-cli  -p 6379 cluster addslots {0..5000}
redis-cli  -p 6380 cluster addslots {5001..10000}
redis-cli  -p 6381 cluster addslots {10001..16383}
```

槽指派后集群中节点情况如下：

```
127.0.0.1:6379> CLUSTER NODES
c47598b25205cc88abe2e5094d5bfd9ea202335f 127.0.0.1:6380@16380 master - 0 1603632880
10000 87b7dfacde34b3cf57d5f46ab44fd6fffb2e4f52 127.0.0.1:6379@16379 myself,master -
5000 51081a64ddb3ccf5432c435a8cf20d45ab795dd8 127.0.0.1:6381@16381 master - 0 16036
16383 9d587b75bdaed26ca582036ed706df8b2282b0aa 127.0.0.1:6481@16481 master - 0 1603
4c23b25bd4bcef7f4b77d8287e330ae72e738883 127.0.0.1:6479@16479 master - 0 1603632880
32ed645a9c9d13ca68dba5a147937fb1d05922ee 127.0.0.1:6480@16480 master - 0 1603632881

127.0.0.1:6379> CLUSTER INFO
cluster_state:ok                          # 集群处于上线状态
cluster_slots_assigned:16384 cluster_slots_ok:16384 cluster_slots_pfail:0 cluster_s
```

主从复制

上述步骤后，集群节点均作为主节点存在，仍不能实现Redis的高可用，配置主从复制之后，才算真正实现了集群的高可用功能。

CLUSTER REPLICATE <node_id> 用来让集群中接收命令的节点成为 node_id 所指定节点的从节点，并开始对主节点进行复制。

```
redis-cli  -p 6479 cluster replicate 87b7dfacde34b3cf57d5f46ab44fd6fffb2e4f52
redis-cli  -p 6480 cluster replicate c47598b25205cc88abe2e5094d5bfd9ea202335f
redis-cli  -p 6481 cluster replicate 51081a64ddb3ccf5432c435a8cf20d45ab795dd8

127.0.0.1:6379> CLUSTER NODES
c47598b25205cc88abe2e5094d5bfd9ea202335f 127.0.0.1:6380@16380 master - 0 1603633105
10000 87b7dfacde34b3cf57d5f46ab44fd6fffb2e4f52 127.0.0.1:6379@16379 myself,master -
5000 51081a64ddb3ccf5432c435a8cf20d45ab795dd8 127.0.0.1:6381@16381 master - 0 16036
16383 9d587b75bdaed26ca582036ed706df8b2282b0aa 127.0.0.1:6481@16481 slave 51081a64d
4c23b25bd4bcef7f4b77d8287e330ae72e738883 127.0.0.1:6479@16479 slave 87b7dfacde34b3c
32ed645a9c9d13ca68dba5a147937fb1d05922ee 127.0.0.1:6480@16480 slave c47598b25205cc8
```

顺带补充，上述步骤1.2，1.3，1.4可以利用 redis-trib.rb 工具整体实现，在Redis 5.0之后直接利用 redis-cli 完成，参考命令如下：

```
redis-cli --
cluster create  127.0.0.1:6379 127.0.0.1:6479  127.0.0.1:6380 127.0.0.1:6480  127.0
-cluster-replicas 1
--cluster-replicas 1 指示给定的创建节点列表是以主节点+从节点对组成的。
```

在集群中执行命令

集群此时处于上线状态，可以通过客户端向集群中的节点发送命令。接收命令的节点会计算出命令要处理的键属于哪个槽，并检查这个槽是否指派给自己。

- 如果键所在的slot刚好指派给了当前节点，会直接执行这个命令。
- 否则，节点向客户端返回 MOVED 错误，指引客户端转向 redirect 至正确的节点，并再次发送此前的命令。

此处，我们利用 CLUSTER KEYSLOT 查看到键 name 所在槽号为5798（被分配在6380节点），当对此键操作时，会被重定向到相应的节点。对键 fruits 的操作与此类似。

```
127.0.0.1:6379> CLUSTER KEYSLOT name
(integer) 5798
127.0.0.1:6379> set name huey -
> Redirected to slot [5798] located at 127.0.0.1:6380 OK 127.0.0.1:6380>

127.0.0.1:6379> get fruits -> Redirected to slot [14943] located at 127.0.0.1:6381
"apple"
127.0.0.1:6381>
```

值得注意的是，当我们将命令通过客户端发送给一个从节点时，命令会被重定向至对应的主节点。

```
127.0.0.1:6480> KEYS *
1) "name"
127.0.0.1:6480> get name -> Redirected to slot [5798] located at 127.0.0.1:6380
"huey"
```

集群故障转移

集群中主节点下线时，复制此主节点的所有的从节点将会选出一个节点作为新的主节点，并完成故障转移。和主从复制的配置相似，当原先的从节点再次上线，它会被作为新主节点的的从节点存在于集群中。

下面模拟6379节点宕机的情况（将其SHUTDOWN），可以观察到其从节点6479将作为新的主节点继续工作。

```
462:S 26 Oct 14:08:12.750 * FAIL message received from c47598b25205cc88abe2e5094d5b
462:S 26 Oct 14:08:13.447 # configEpoch set to 6 after successful failover 462:M 26
```

6379节点从宕机状态恢复后，将作为6380节点的从节点存在。

```
127.0.0.1:6379> CLUSTER NODES
51081a64ddb3ccf5432c435a8cf20d45ab795dd8 127.0.0.1:6381@16381 master - 0 1603692968
16383 c47598b25205cc88abe2e5094d5bfd9ea202335f 127.0.0.1:6380@16380 master - 0 1603
10000 4c23b25bd4bcef7f4b77d8287e330ae72e738883 127.0.0.1:6479@16479 master - 0 1603
5000 87b7dfacde34b3cf57d5f46ab44fd6fffb2e4f52 127.0.0.1:6379@16379 myself,slave 4c2
9d587b75bdaed26ca582036ed706df8b2282b0aa 127.0.0.1:6481@16481 slave 51081a64ddb3ccf
32ed645a9c9d13ca68dba5a147937fb1d05922ee 127.0.0.1:6480@16480 slave c47598b25205cc8
```

前文提到 cluster-config-file 会记录下集群节点的状态，打开节点6379的配置文件 nodes-6379.conf ，可以看到 CLUSTER NODES 所示信息均被保存在配置文件中：

```
51081a64ddb3ccf5432c435a8cf20d45ab795dd8 127.0.0.1:6381@16381 master - 0 1603694920
16383 c47598b25205cc88abe2e5094d5bfd9ea202335f 127.0.0.1:6380@16380 master - 0 1603
10000 4c23b25bd4bcef7f4b77d8287e330ae72e738883 127.0.0.1:6479@16479 master - 0 1603
5000 87b7dfacde34b3cf57d5f46ab44fd6fffb2e4f52 127.0.0.1:6379@16379 myself,slave 4c2
9d587b75bdaed26ca582036ed706df8b2282b0aa 127.0.0.1:6481@16481 slave 51081a64ddb3ccf
32ed645a9c9d13ca68dba5a147937fb1d05922ee 127.0.0.1:6480@16480 slave c47598b25205cc8
vars currentEpoch 6 lastVoteEpoch 0
```

集群伸缩实践

集群伸缩的关键在于对集群的进行重新分片，实现槽位在节点间的迁移。本节将以在集群中添加节点和删除节点为例，对槽迁移进行实践。

借助于 redis-cli 中集成的 redis-trib.rb 工具进行槽位的管理，工具的帮助菜单如下：

```
$redis-cli--clusterhelp
Cluster Manager Commands:
createhost1:port1...hostN:portN--cluster-replicas<arg>checkhost:port--cluster-
search-multiple-owners
infohost:port
fixhost:port--cluster-search-multiple-owners--cluster-fix-with-unreachable-masters
reshardhost:port--cluster-from<arg>
--cluster-to<arg>
--cluster-slots<arg>
--cluster-yes--cluster-timeout<arg>
--cluster-pipeline<arg>
--cluster-replace
rebalancehost:port--cluster-weight<node1=w1...nodeN=wN>
--cluster-use-empty-masters--cluster-timeout<arg>
--cluster-simulate--cluster-pipeline<arg>
--cluster-threshold<arg>
--cluster-replace
add-nodenew_host:new_portexisting_host:existing_port--cluster-slave--cluster-
master-id<arg>del-nodehost:portnode_id
callhost:portcommandargarg..argset-timeouthost:portmilliseconds
importhost:port--cluster-from<arg>
--cluster-copy--cluster-replace
backuphost:portbackup_directory
help


node, set-
timeout you can specify the host and port of any working node in the cluster.
```

集群伸缩-添加节点

考虑在集群中添加两个节点，端口号为6382和6482，其中节点6482对6382进行复制。

- (1) 启动节点：按照1.1中介绍的步骤，启动6382和6482节点。
- (2) 节点握手：借助 redis-cli --cluster add-node 命令分别添加节点6382和6482。

```
redis-cli --cluster add-node 127.0.0.1:6382 127.0.0.1:6379 redis-cli --
cluster add-node 127.0.0.1:6482 127.0.0.1:6379

$ redis-cli --cluster add-node 127.0.0.1:6382 127.0.0.1:6379
>>> Adding node 127.0.0.1:6382 to cluster 127.0.0.1:6379
>>> Performing Cluster Check (using node 127.0.0.1:6379)
S: 87b7dfacde34b3cf57d5f46ab44fd6fffb2e4f52 127.0.0.1:6379 slots: (0 slots) slave
    replicates 4c23b25bd4bcef7f4b77d8287e330ae72e738883
M: 51081a64ddb3ccf5432c435a8cf20d45ab795dd8 127.0.0.1:6381 slots:[10001-
16383] (6383 slots) master 1 additional replica(s)
M: c47598b25205cc88abe2e5094d5bfd9ea202335f 127.0.0.1:6380 slots:[5001-
10000] (5000 slots) master 1 additional replica(s)
M: 4c23b25bd4bcef7f4b77d8287e330ae72e738883 127.0.0.1:6479 slots:[0-
5000] (5001 slots) master 1 additional replica(s)
S: 9d587b75bdaed26ca582036ed706df8b2282b0aa 127.0.0.1:6481 slots: (0 slots) slave
    replicates 51081a64ddb3ccf5432c435a8cf20d45ab795dd8
S: 32ed645a9c9d13ca68dba5a147937fb1d05922ee 127.0.0.1:6480 slots: (0 slots) slave
    replicates c47598b25205cc88abe2e5094d5bfd9ea202335f
[OK] All nodes agree about slots configuration. >>> Check for open slots... >>> Che
[OK] All 16384 slots covered. >>> Send CLUSTER MEET to node 127.0.0.1:6382 to make
[OK] New node added correctly.
```

- 移动的槽位数：最终平均每个主节点有4096个slot，因此总共移动4096 slots
- 接收槽位的目标节点ID：节点6382的ID
- 移出槽位的源节点ID：节点6379/6380/6381的ID
  重新分片：借助 redis-cli --cluster reshard 命令对集群重新分片，使得各节点槽位均衡（分别从节点6379/6380/6381中迁移一些slot到节点6382中）。需要指定：

```
$ redis-cli --cluster reshard 127.0.0.1 6479
>>> Performing Cluster Check (using node 127.0.0.1:6479)
M: 4c23b25bd4bcef7f4b77d8287e330ae72e738883 127.0.0.1:6479 slots:[0-
5000] (5001 slots) master 1 additional replica(s)
S: 32ed645a9c9d13ca68dba5a147937fb1d05922ee 127.0.0.1:6480 slots: (0 slots) slave
  replicates c47598b25205cc88abe2e5094d5bfd9ea202335f
M: 706f399b248ed3a080cf1d4e43047a79331b714f 127.0.0.1:6482 slots: (0 slots) master
M: af81109fc29f69f9184ce9512c46df476fe693a3 127.0.0.1:6382 slots: (0 slots) master
M: 51081a64ddb3ccf5432c435a8cf20d45ab795dd8 127.0.0.1:6381 slots:[10001-
16383] (6383 slots) master 1 additional replica(s)
S: 9d587b75bdaed26ca582036ed706df8b2282b0aa 127.0.0.1:6481 slots: (0 slots) slave
  replicates 51081a64ddb3ccf5432c435a8cf20d45ab795dd8
S: 87b7dfacde34b3cf57d5f46ab44fd6fffb2e4f52 127.0.0.1:6379 slots: (0 slots) slave
  replicates 4c23b25bd4bcef7f4b77d8287e330ae72e738883
M: c47598b25205cc88abe2e5094d5bfd9ea202335f 127.0.0.1:6380 slots:[5001-
10000] (5000 slots) master 1 additional replica(s)
[OK] All nodes agree about slots configuration. >>> Check for open slots... >>> Che
[OK] All 16384 slots covered.
How many slots do you want to move (from 1 to 16384)? 4096 What is the receiving no
```

(4) 设置主从关系：

```
redis-cli -p 6482 cluster replicate af81109fc29f69f9184ce9512c46df476fe693a3

127.0.0.1:6482> CLUSTER NODES
32ed645a9c9d13ca68dba5a147937fb1d05922ee 127.0.0.1:6480@16480 slave c47598b25205cc8
51081a64ddb3ccf5432c435a8cf20d45ab795dd8 127.0.0.1:6381@16381 master - 0 1603694931
16383 9d587b75bdaed26ca582036ed706df8b2282b0aa 127.0.0.1:6481@16481 slave 51081a64d
706f399b248ed3a080cf1d4e43047a79331b714f 127.0.0.1:6482@16482 myself,slave af81109f
87b7dfacde34b3cf57d5f46ab44fd6fffb2e4f52 127.0.0.1:6379@16379 slave 4c23b25bd4bcef7
c47598b25205cc88abe2e5094d5bfd9ea202335f 127.0.0.1:6380@16380 master - 0 1603694933
10000 4c23b25bd4bcef7f4b77d8287e330ae72e738883 127.0.0.1:6479@16479 master - 0 1603
5000 af81109fc29f69f9184ce9512c46df476fe693a3 127.0.0.1:6382@16382 master - 0 16036
1249 5001-6250 10001-11596
```

集群伸缩-删除节点

这里考虑将新添加的两个节点6382和6482删除，需要将节点6382上分配的槽位迁移到其他节点。

(1) 重新分片：同样借助 redis-cli --cluster reshard 命令，将6382节点上的槽位全部转移到节点6479上。

```
$redis-cli--clusterreshard127.0.0.16382
>>>PerformingClusterCheck(usingnode127.0.0.1:6382)
M:af81109fc29f69f9184ce9512c46df476fe693a3127.0.0.1:6382slots:[0-1249],[5001-
6250],[10001-11596](4096slots)master1additionalreplica(s)
M:51081a64ddb3ccf5432c435a8cf20d45ab795dd8127.0.0.1:6381slots:[11597-16383]
(4787slots)master1additionalreplica(s)
S: 87b7dfacde34b3cf57d5f46ab44fd6fffb2e4f52 127.0.0.1:6379 slots:(0slots)slave
replicates4c23b25bd4bcef7f4b77d8287e330ae72e738883
S: 32ed645a9c9d13ca68dba5a147937fb1d05922ee 127.0.0.1:6480 slots:(0slots)slave
replicatesc47598b25205cc88abe2e5094d5bfd9ea202335f
M:4c23b25bd4bcef7f4b77d8287e330ae72e738883127.0.0.1:6479slots:[1250-5000]
(3751slots)master1additionalreplica(s)
M:c47598b25205cc88abe2e5094d5bfd9ea202335f127.0.0.1:6380slots:[6251-10000]
(3750slots)master1additionalreplica(s)
S: 706f399b248ed3a080cf1d4e43047a79331b714f 127.0.0.1:6482 slots:(0slots)slave
replicatesaf81109fc29f69f9184ce9512c46df476fe693a3
S: 9d587b75bdaed26ca582036ed706df8b2282b0aa 127.0.0.1:6481 slots:(0slots)slave
replicates51081a64ddb3ccf5432c435a8cf20d45ab795dd8
[OK]Allnodesagreeaboutslotsconfiguration.>>>Checkforopenslots...>>>Checkslotscovera
[OK]All16384slotscovered.
Howmanyslotsdoyouwanttomove(from1to16384)?4096WhatisthereceivingnodeID?
4c23b25bd4bcef7f4b77d8287e330ae72e738883
PleaseenterallthesourcenodeIDs.
Type'all'touseallthenodesassourcenodesforthehashslots.
Type'done'onceyouenteredallthesourcenodesIDs.
Sourcenode#1: af81109fc29f69f9184ce9512c46df476fe693a3
Sourcenode#2: done


c47598b25205cc88abe2e5094d5bfd9ea202335f 127.0.0.1:6380@16380 master - 0 1603773540
10000 87b7dfacde34b3cf57d5f46ab44fd6fffb2e4f52 127.0.0.1:6379@16379 myself,slave 4c
4c23b25bd4bcef7f4b77d8287e330ae72e738883 127.0.0.1:6479@16479 master - 0 1603773541
6250 10001-
11596 706f399b248ed3a080cf1d4e43047a79331b714f 127.0.0.1:6482@16482 slave 4c23b25bd
32ed645a9c9d13ca68dba5a147937fb1d05922ee 127.0.0.1:6480@16480 slave c47598b25205cc8
9d587b75bdaed26ca582036ed706df8b2282b0aa 127.0.0.1:6481@16481 slave 51081a64ddb3ccf
af81109fc29f69f9184ce9512c46df476fe693a3 127.0.0.1:6382@16382 master - 0 1603773539
51081a64ddb3ccf5432c435a8cf20d45ab795dd8 127.0.0.1:6381@16381 master - 0 1603773540
16383
```

(2) 删除节点：利用 redis-cli --cluster del-node 命令依次删除从节点6482和主节点6382。

```
$ redis-cli --cluster del-
node 127.0.0.1:6482 706f399b248ed3a080cf1d4e43047a79331b714f >>> Removing node 706f
>>> Sending CLUSTER FORGET messages to the cluster... >>> Sending CLUSTER RESET SOF
$ redis-cli --cluster del-
node 127.0.0.1:6382 af81109fc29f69f9184ce9512c46df476fe693a3 >>> Removing node af81
>>> Sending CLUSTER FORGET messages to the cluster... >>> Sending CLUSTER RESET SOF

127.0.0.1:6379> CLUSTER NODES
c47598b25205cc88abe2e5094d5bfd9ea202335f 127.0.0.1:6380@16380 master - 0 1603773679
10000 87b7dfacde34b3cf57d5f46ab44fd6fffb2e4f52 127.0.0.1:6379@16379 myself,slave 4c
4c23b25bd4bcef7f4b77d8287e330ae72e738883 127.0.0.1:6479@16479 master - 0 1603773678
6250 10001-
11596 32ed645a9c9d13ca68dba5a147937fb1d05922ee 127.0.0.1:6480@16480 slave c47598b25
9d587b75bdaed26ca582036ed706df8b2282b0aa 127.0.0.1:6481@16481 slave 51081a64ddb3ccf
51081a64ddb3ccf5432c435a8cf20d45ab795dd8 127.0.0.1:6381@16381 master - 0 1603773678
16383
```

总结

Redis集群环境的搭建主要包括启动节点、节点握手、槽指派和主从复制等四个步骤，集群伸缩同样涉及这几个方面。借助 redis-cli --cluster 命令来管理集群环境，不仅能增加简便性，还能降低操作失误的风险。

> *原文：https://www.cnblogs.com/hueyxu/p/13884800.html*

涂！又是美团。。。。这次吃相很难看！

全球最大色情网站宣布：封杀特朗普

红旗 Linux 桌面操作系统 11 来了

华为悄悄推出"应用市场"，免费、无广告，贼好用！

有人靠"抢茅台"月入百万，脚本曝光，开源可用！

这款国产SSH客户端很牛逼！网友直呼：666

职场防坑指南（2020 版）

2020 年公众号最受欢迎文章！