小象科技 ChinaHadoop.cn

Redis 从入门到精通

黄健宏（huangz）

Redis 的集群

在前面的课程中，我们陆续学习了 Redis 的复制特性，以及 Redis Sentinel 和 twemproxy 这两个程序，其中：

- 复制特性可以创建指定服务器的复制品，这些复制品可以用于扩展系统处理读请求的能力。

- Redis Sentinel 可以在复制特性的基础上，通过监视主从服务器并在主服务器故障时执行自动故障转移来保证系统的可用性。

- twemproxy 使用分片策略来将数据库划分到多个不同的服务器，以此来扩展系统储存的数据量，并通过将命令请求分散给不同的服务器来处理，以此来扩展系统处理命令请求的能力。

以上这些特性或程序都是独立的，如果我们需要一个完整地包含复制、高可用和分片特性的 Redis 服务器群，那么就需要用到 Redis 的集群（cluster）特性。

小象科技
ChinaHadoop.cn

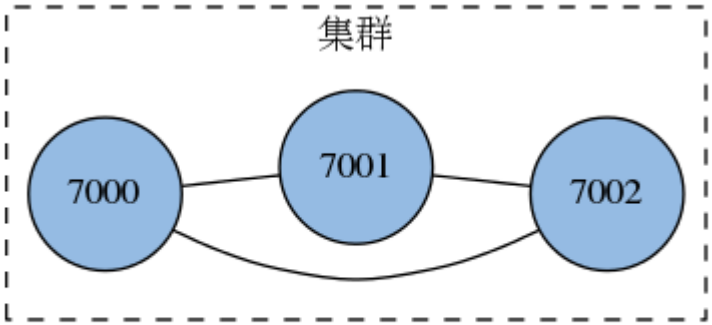
集群

Redis 的分布式数据库实现

分布式

Redis 集群是一个由多个 Redis 服务器组成的分布式网络服务器群，集群中的各个服务器被称为节点（node），这些节点会相互连接并进行通信。

分布式的 Redis 集群没有中心节点，所以用户不必担心某个节点会成为整个集群的性能瓶颈。

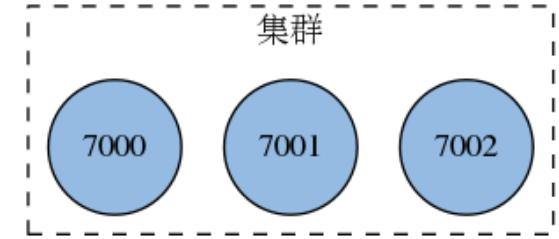


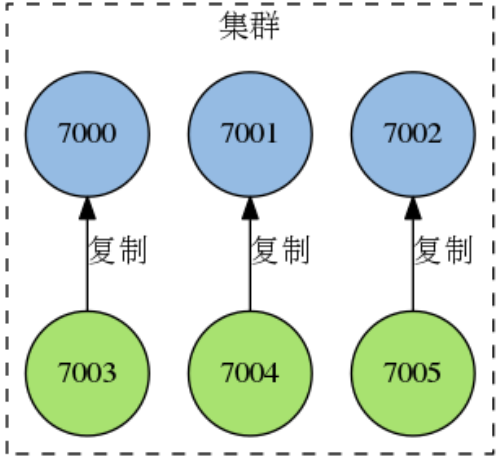
集群中的三个节点互相连接。

复制

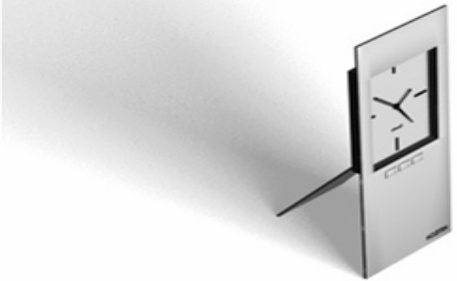
Redis 集群的每个节点都有两种角色可选，一个是主节点(master node)，另一个是从节点(slave node)：其中主节点用于储存数据，而从节点则是某个主节点的复制品。

当用户需要处理更多读请求的时候，添加从节点可以扩展系统的读性能。因为 Redis 集群重用了单机 Redis 复制特性的代码，所以集群的复制行为和我们之前介绍的单机复制特性的行为是完全一样的。





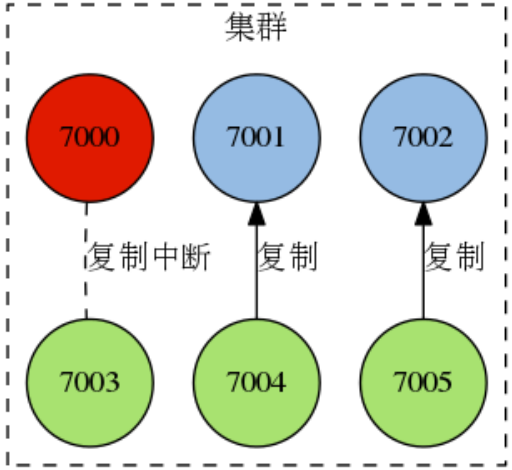
为三个主节点分别添加一个从节点。

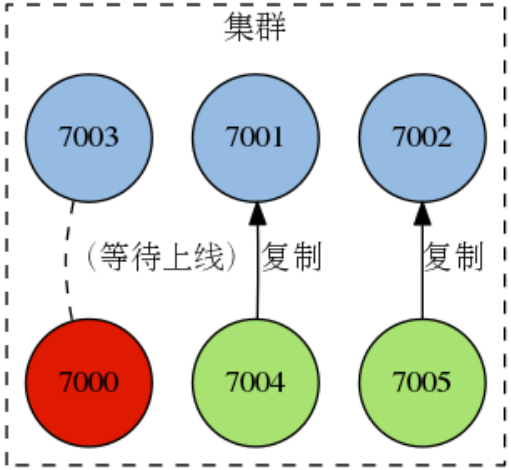


节点故障检测和自动故障转移

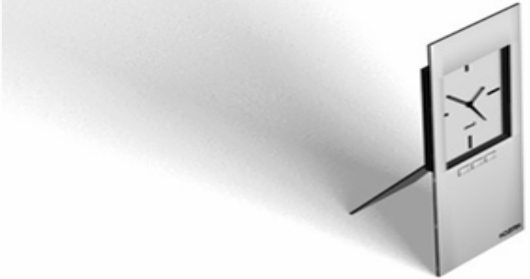
Redis 集群的主节点内置了类似 Redis Sentinel 的节点故障检测和自动故障转移功能，当集群中的某个主节点下线时，集群中的其他在线主节点会注意到这一点，并对已下线的主节点进行故障转移。

集群进行故障转移的方法和 Redis Sentinel 进行故障转移的方法基本一样，不同的是，在集群里面，故障转移是由集群中其他在线的主节点负责进行的，所以集群不必另外使用 Redis Sentinel 。





当 7000 下线时，7001 和 7002 会察觉到这一点，并对 7000 进行故障转移。



分片

集群使用分片来扩展数据库的容量，并将命令请求的负载交给不同的节点来分担。

集群将整个数据库分为 16384 个槽(slot)，所有键都属于这 16384 个槽的其中一个，计算键 key 属于哪个槽的公式为 slot_number = crc16(key) % 16384 ，其中 crc16 为 16 位的循环冗余校验和函数。

集群中的每个主节点都可以处理 0 个至 16384 个槽，当 16384 个槽都有某个节点在负责处理时，集群进入上线状态，并开始处理客户端发送的数据命令请求。

比如说，如果我们有三个主节点 7000 、 7001 和 7002 ，那么我们可以：

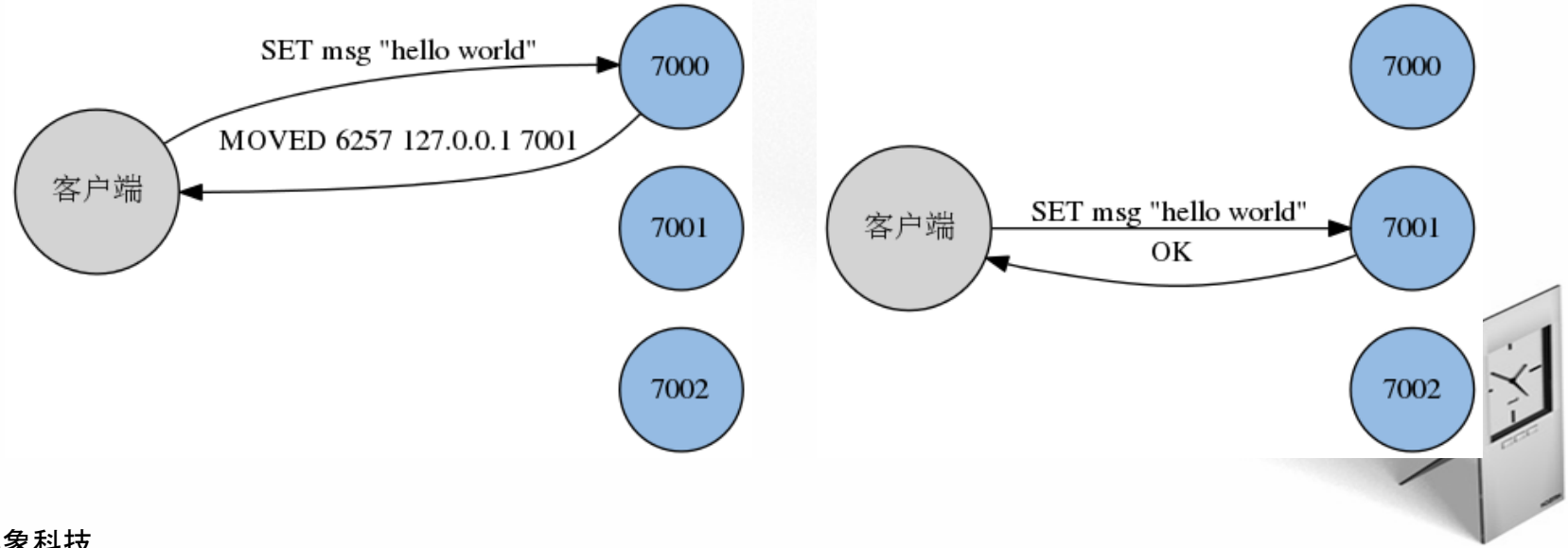
- 将槽 0 至 5460 指派给节点 7000 负责处理；
- 将槽 5461 至 10922 指派给节点 7001 负责处理；
- 将槽 10923 至 16383 指派给节点 7002 负责处理；

这样就可以将 16384 个槽平均地指派给三个节点负责处理。

转向

对于一个被指派了槽的主 节点来说，这个主节点只会处理属于指派给自己的槽的命令 请求。

如果一个 节点接收到了和自己 处理的槽无关的命令 请求，那么 节点会向客户端返回一个 转向错误（redirection error），告诉客户端，哪个 节点才是负责处理这条命令的，之后客 户端需要根据 错误中包含的地址和端口号重新向正确的 节点发送命令请求。



小象科技 ChinaHadoop.cn

集群搭建

设置和建立一个集群的方法

集群搭建步骤

搭建一个 Redis 集群需要执行以下步骤:

1. 创建多个节点。
2. 为每个节点指派槽，并将多个节点连接起来，组成一个集群。
3. 当集群数据库的 16384 个槽都有节点在处理时，集群进入上线状态。

接下来，就让我们来搭建一个包含六个节点的 Redis 集群，其中三个节点为主节点，而另外三个节点为从节点，每个主节点都有一个从节点。

注意，在极端情况下，如果将 16384 个槽都指派给一个主节点，那么只有一个主节点也可以让集群进入上线状态，但是要让集群的故障转移特性生效，最起码要有三个主节点;而要让故障转移真正有意义，最少要为三个主节点分别设置一个从节点，这也是我们使用六个节点作为例子的原因。

创建节点

集群中的节点就是运行在集群模式下的 Redis 服务器，为了构建一个集群，我们需要一一创建集群中的每个节点。

为了让 Redis 服务器以集群模式运行，我们需要在启动服务器时，打开服务器的集群模式选项：

```
cluster-enabled yes
```

另外，如果有多个节点运行在同一台机器里面，那么我们还需要为每个节点指定不同的端口号：

```
port 7000
```

我们可以将这两个配置值写入到 redis.conf 文件里面，然后执行以下命令来启动一个节点：

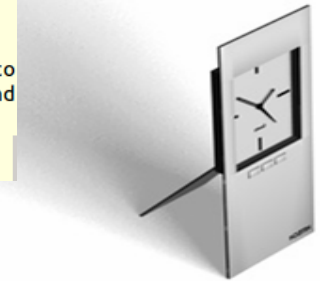
```
$ redis-server redis.conf
```

节点运行示例

小象科技 ChinaHadoop.cn

```
huangz@pc:~/rediscluster/7000$ redis-server redis.conf
[4935] 10 Oct 18:45:36.943 # You requested maxclients of 10000 requiring at least 10032 max file descriptors.
[4935] 10 Oct 18:45:36.943 # Redis can't set maximum open files to 10032 because of OS error: Operation not perm
itted.
[4935] 10 Oct 18:45:36.943 # Current maximum open files is 1024. maxclients has been reduced to 4064 to compensa
te for low ulimit. If you need higher maxclients increase 'ulimit -n'.
[4935] 10 Oct 18:45:36.945 * Node configuration loaded, I'm 902ad043f5e12dcaad2b5075d5070bad692f2c2e
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 2.9.11 (937384d0/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in cluster mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7000
      |    `-._   `._    /     _.-'    |     PID: 4935
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

[4935] 10 Oct 18:45:36.947 # Server started, Redis version 2.9.11
[4935] 10 Oct 18:45:36.947 # WARNING overcommit_memory is set to 0! Background save may fail under low memory co
ndition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command
 'sysctl vm.overcommit_memory=1' for this to take effect.
[4935] 10 Oct 18:45:36.948 * The server is now ready to accept connections on port 7000
```



节点创建示例

为了在同一台机器上构建一个包含六个 节点的集群，其中三个主 节点分别运行在机器的 7000 、7001 和 7002 端口，而三个从节点则分别运行在机器的 7003 、7004 和 7005 端口，我们可以先创建一个 rediscluster 文件夹，然后分别创建 7000 至 7005 这六个文件夹，每个文件夹都包含一个 redis.conf 文件，它的内容为：

```
port <number>
cluster-enabled yes
```

然后只要分别执行：

```
~/rediscluster/7000$ redis-server redis.conf
…
~/rediscluster/7005$ redis-server redis.conf
```

就可以创建出六个节点了（节点启动时默认为主节点，之后要将其中三个 节点转为从节点）。

创建集群

在创建出六个节点之后，我们需要让这六个节点互相连接以构成一个集群，然后为三个主节点指派槽，并为这三个主节点分别设置一个从节点。

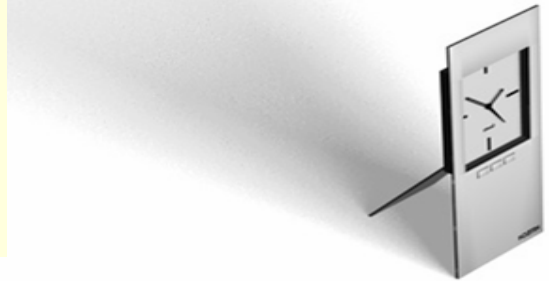
创建集群的操作可以通过使用位于 Redis 安装文件夹内的 redis-trib.rb 程序来完成，这是一个使用 Ruby 编写的 Redis 集群管理程序，它具有创建集群、检查集群的上线情况和槽指派情况、对集群进行重新分片、向集群添加新节点或者从集群中移除节点等功能。

不带任何参数地执行 redis-trib.rb 可以看到它的各项用法：

```
huangz@pc:~/download/redis/src$ ./redis-trib.rb
Usage: redis-trib <command> <options> <arguments ...>

  create          host1:port1 ... hostN:portN
                  --replicas <arg>
  check           host:port
  fix             host:port
  reshard         host:port
                  --from <arg>
                  --to <arg>
                  --slots <arg>
                  --yes
  add-node        new_host:new_port existing_host:existing_port
                  --slave
                  --master-id <arg>
  del-node        host:port node_id
  set-timeout     host:port milliseconds
  call            host:port command arg arg .. arg
  import          host:port
                  --from <arg>
  help            (show this help)

For check, fix, reshard, del-node, set-timeout you can specify the host and port of any working node in the cluster.
```



redis-trib.rb 的 create 方法

为了创建一个包含三个主节点和三个从节点的集群，我们需要执行以下命令：

```
$ ./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
```

其中，create 方法表示我们要创建一个集群，而之后的 --replicas 1 则表示让 redis-trib.rb 为集群中的每个主节点设置一个从节点，再之后输入的是各个节点的 IP 地址和端口号。

在输入该命令之后， redis-trib 会为各个节点指派槽以及角色，并询问用户是否接受这种节点配置（见下页）。

节点配置


```
>>> 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: d92c9d6b06d14cdc5281fa70212656929f709162 127.0.0.1:7000
   slots:0-5460 (5461 slots) master
M: 03c9d2b38827633bc7238027f9ca3ffb316eefd9 127.0.0.1:7001
   slots:5461-10922 (5462 slots) master
M: ed4c6724daa4cfcdfdb1521717faf7b50eebbdd7 127.0.0.1:7002
   slots:10923-16383 (5461 slots) master
S: b00aaf836f615f547048dfa310abee57dbaa1798 127.0.0.1:7003
   replicates d92c9d6b06d14cdc5281fa70212656929f709162
S: f5df06d76af8047267f72fcf4b9c2b57df124ece 127.0.0.1:7004
   replicates 03c9d2b38827633bc7238027f9ca3ffb316eefd9
S: 31e940304bec2f5c0f1cea9fec8e62be1a226375 127.0.0.1:7005
   replicates ed4c6724daa4cfcdfdb1521717faf7b50eebbdd7
Can I set the above configuration? (type 'yes' to accept):
```

redis-trib.rb 首先尝试连接给定的六个节点，检查它们是否存在。

在确定这些节点都是可连接之后，redis-trib.rb 再将 7000 、7001 和 7002 设置为主节点，而 7003 、7004 和 7005 则分别被设置为三个主节点的从节点。

对于三个主节点，redis-trib.rb 会分别为它们指派 5461 、5462 和 5461 个槽（默认情况下使用平均分配）。

如果觉得这个配置没问题的话，就可以键入 yes 并按下回车。

连接各个节点


```
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...
```

在键入 yes 之后， redis-trib.rb 就会向各个节点发送指令，首先将它们连接为一个集群，然后再为它们指派槽和设置角色。

进行测试


```
>>> Performing Cluster Check (using node 127.0.0.1:7000)
M: d92c9d6b06d14cdc5281fa70212656929f709162 127.0.0.1:7000
   slots:0-5460 (5461 slots) master
M: 03c9d2b38827633bc7238027f9ca3ffb316eefd9 127.0.0.1:7001
   slots:5461-10922 (5462 slots) master
M: ed4c6724daa4cfcdfdb1521717faf7b50eebbdd7 127.0.0.1:7002
   slots:10923-16383 (5461 slots) master
M: b00aaf836f615f547048dfa310abee57dbaa1798 127.0.0.1:7003
   slots: (0 slots) master
   replicates d92c9d6b06d14cdc5281fa70212656929f709162
M: f5df06d76af8047267f72fcf4b9c2b57df124ece 127.0.0.1:7004
   slots: (0 slots) master
   replicates 03c9d2b38827633bc7238027f9ca3ffb316eefd9
M: 31e940304bec2f5c0f1cea9fec8e62be1a226375 127.0.0.1:7005
   slots: (0 slots) master
   replicates ed4c6724daa4cfcdfdb1521717faf7b50eebbdd7
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

之后，redis-trib.rb 会对集群进行测试，检查是否每个节点都按照原先展示的配置 设置好了。

如果整个集群数据 库的 16384 个槽都有节点在处理，那么集群就会 进入上线状态，之后用户就可以开始向集群 发送命令请求了。

访问集群

使用集群客户端向集群发送命令请求

集群客户端

因为集群功能比起单机功能要复杂得多，所以不同语言的 Redis 客户端通常需要为集群添加特别的支持，或者专门开发一个集群客户端。

目前主要的 Redis 集群客户端（或者说，支持集群功能的 Redis 客户端）有以下这些：

- redis-rb-cluster：antirez 使用 Ruby 编写的 Redis 集群客户端，集群客户端的官方实现。
- predis：Redis 的 PHP 客户端，支持集群功能。
- jedis：Redis 的 JAVA 客户端，支持集群功能。
- StackExchange.Redis：Redis 的 C# 客户端，支持集群功能。
- 内置的 redis-cli ：在启动时给定 -c 参数即可进入集群模式，支持部分集群功能。

为了让示例保持简单，我们这里使用集群模式的 redis-cli 来进行演示，需要更完整功能的话，大家可以选择上面列举的其他客户端。

连接节点并执行命令


```
$ redis-cli -p 7000 -c

127.0.0.1:7000> SET date 2014-10-10  # 键 date 所在的槽位于节点 7000 ，节点直接执行命令
OK

127.0.0.1:7000> SET msg "hello world"  # 键 msg 所在的槽位于节点 7001
-> Redirected to slot [6257] located at 127.0.0.1:7001  # 客户端从 7000 转向至 7001
OK

127.0.0.1:7001> SADD fruits “apple” “banana” “cherry”  # 键 fruits 所在的槽位于节点 7002
-> Redirected to slot [14943] located at 127.0.0.1:7002 # 客户端从 7001 转向至 7002
(integer) 2

127.0.0.1:7002> # 转向是自动完成的，无需任何用户操作
```

复习

回顾本节学习的内容

本节重点

Redis 集群是一个由多个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 的集群没有中心节点，并且带有复制和故障转移特性，这可以避免单个节点成为性能瓶颈，或者因为某个节点下线而导致整个集群下线。

集群中的主节点负责处理槽（储存数据），而从节点则是主节点的复制品。

Redis 集群将整个数据库分为 16384 个槽，数据库中的每个键都属于 16384 个槽中的其中一个。

集群中的每个主节点都可以负责 0 个至 16384 个槽，当 16384 个槽都有节点在负责时，集群进入上线状态，可以执行客户端发送的数据命令。

主节点只会执行和自己负责的槽有关的命令，当节点接收到不属于自己处理的槽的命令时，它会将处理指定槽的节点的地址返回给客户端，而客户端会向正确的节点重新发送命令，这个过程称为“转向”。

集群和 twemproxy 的区别

	twemproxy	集群
运作模式	代理模式，代理本身可能成为性能瓶颈，随着负载的增加需要添加更多 twemproxy 来分担请求负载，但每个 twemproxy 本身也会消耗一定的资源。	分布式，没有中心节点，但是因为每个节点都需要互相进行数据通信，所以在节点的数量较多时，集群用于进行通信所耗费的网络资源会比较多。
分片	基本上是按照池中的服务器数量 N 来分片，每个服务器平均占整个数据库的 1/N。	按照槽来进行分片，通过为每个节点指派不同数量的槽，可以控制不同节点负责的数据量和请求数量。
复制和高可用	需要配合 Redis 的复制特性以及 Redis Sentinel 才能实现复制和高可用。	集群的节点内置了复制和高可用特性。

结论：如果需要完整的分片、复制和高可用特性，并且要避免使用代理带来的性能瓶颈和资源消耗，那么可以选择使用 Redis 集群；如果只需要一部分特性（比如只需要分片，但不需要复制和高可用，诸如此类），那么可以单独选择 twemproxy 、Redis 复制和 Redis Sentinel 中的一个或多个。