Redis课程3

1. Redis的集群

解决单点问题

主从复制方案。

192.168.3.37-Master

192.168.3.38-Slave

192.168.3.39-Slave

在Slave节点的redis.conf中配置maser信息：

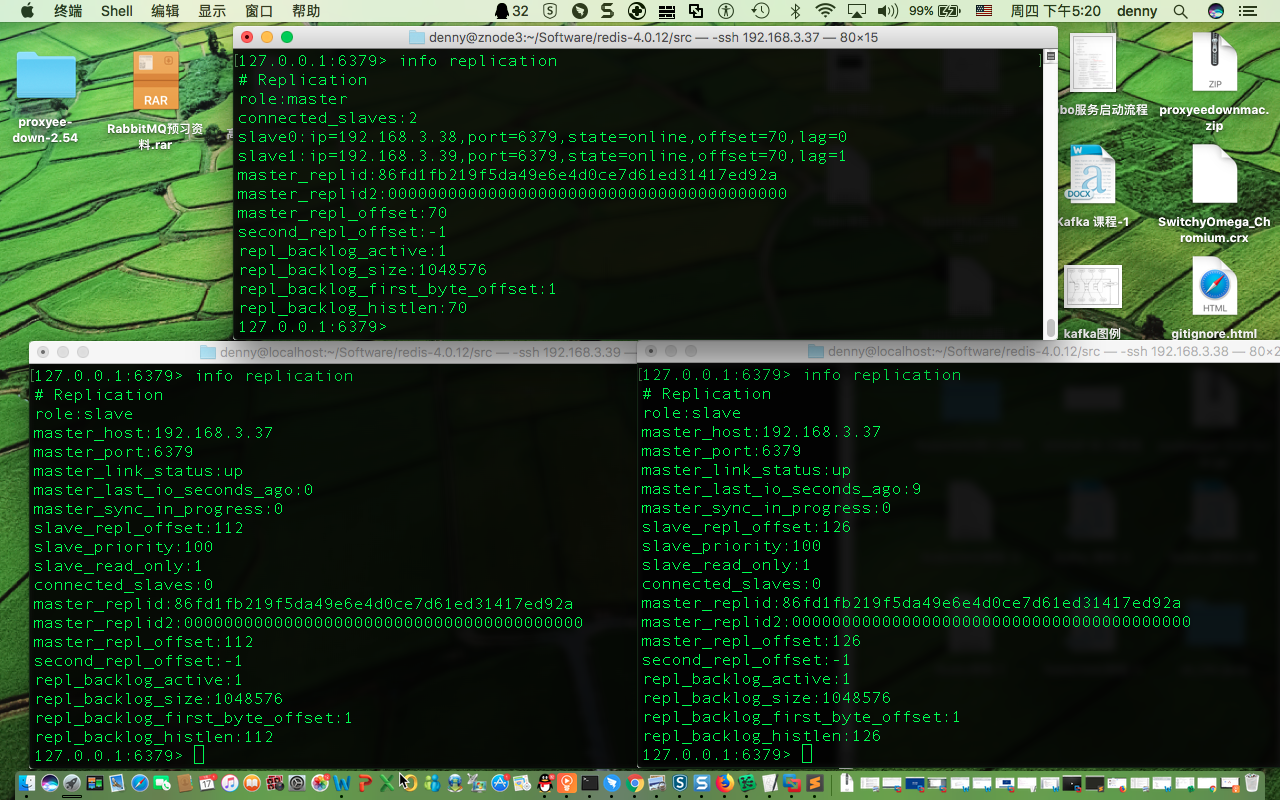
|  |
| --- |
| slaveof 192.168.3.37 6379 |

在Master和Slave节点的redis.conf中配置取消本机绑定和保护模式：

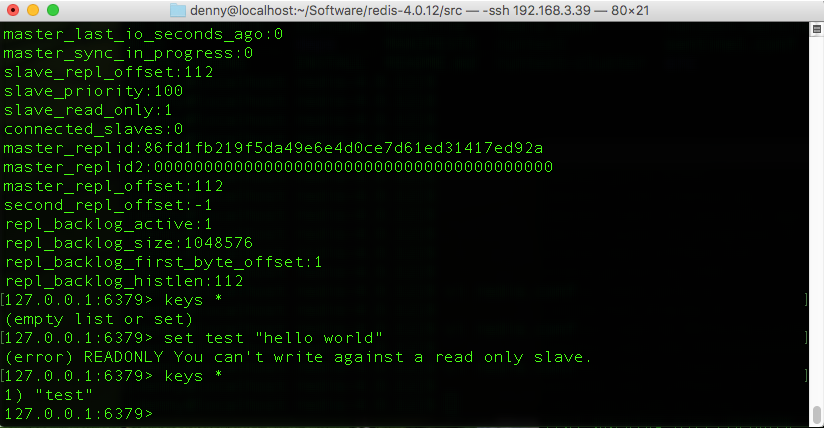
|  |
| --- |
| protected-mode no  #bind 127.0.0.1 |

集群成功启动：

>info replication



Slave客户端不能执行写入操作：



复制方式：

1. 全量复制

初始化时，如新增slave节点。

RDB快照

1. 增量复制

Slave监听增量复制：

>replconf listener-port 6379

>sync

如下：



Master上的配置：

|  |
| --- |
| min-slaves-to-write 3 //当master连接上3个或以上slave才可写  min-slaves-max-lag 10 //slave最长丢失连接的时间(秒) |

repl\_backlog和slave\_repl\_offset保证slave与master断开重连上后从上次同步的offset开始增量同步。

1. 无磁盘复制

repl-diskless-sync no //指定不生成RBD快照进行全量同步，在内存中生成RBD文件。

1. 哨兵机制

实现集群高可用。

Master选举，哨兵机制。Master 宕机后从Slave中选举Master。

作用：

1. 监控master和slave是否正常运行。
2. 当master出现故障时，从slave中选举新的master。

Raft 实现分布式一致性的协议。

在Master 和 Slave节点上配置哨兵进程，192.168.3.37作为master。

哨兵机制相关配置（sentinel.conf）：

|  |
| --- |
| bind 0.0.0.0  protected-mode no  port 26397  dir "/tmp"  sentinel monitor mymaster 192.168.3.37 6379 2  sentinel auth-pass mymaster 012\_345^678-90  sentinel down-after-milliseconds mymaster 60000  sentinel can-failover mymaster yes  sentinel failover-timeout mymaster 180000  sentinel parallel-syncs mymaster 1 |

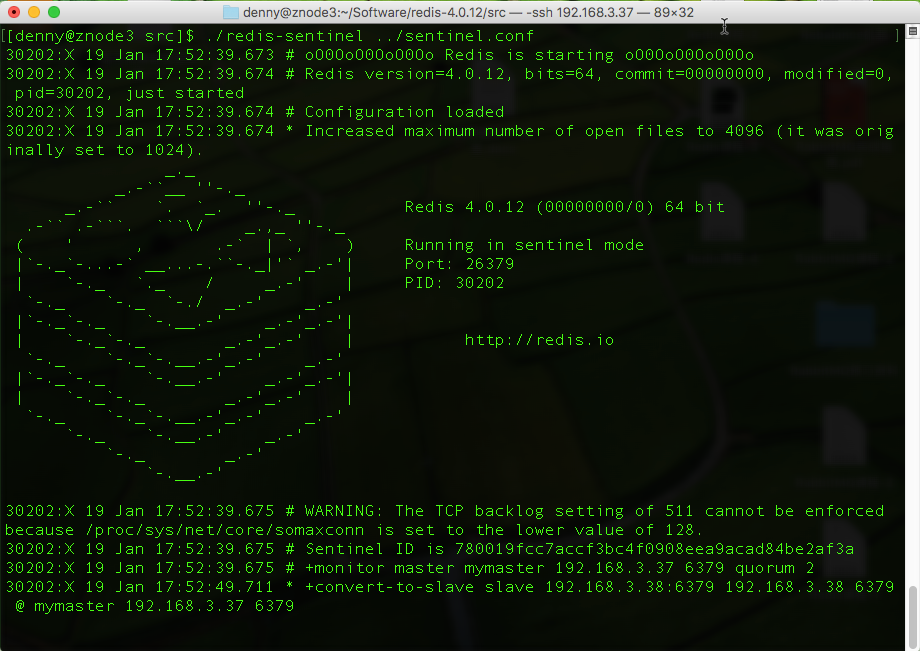
依次启动master和slave节点的redis服务：

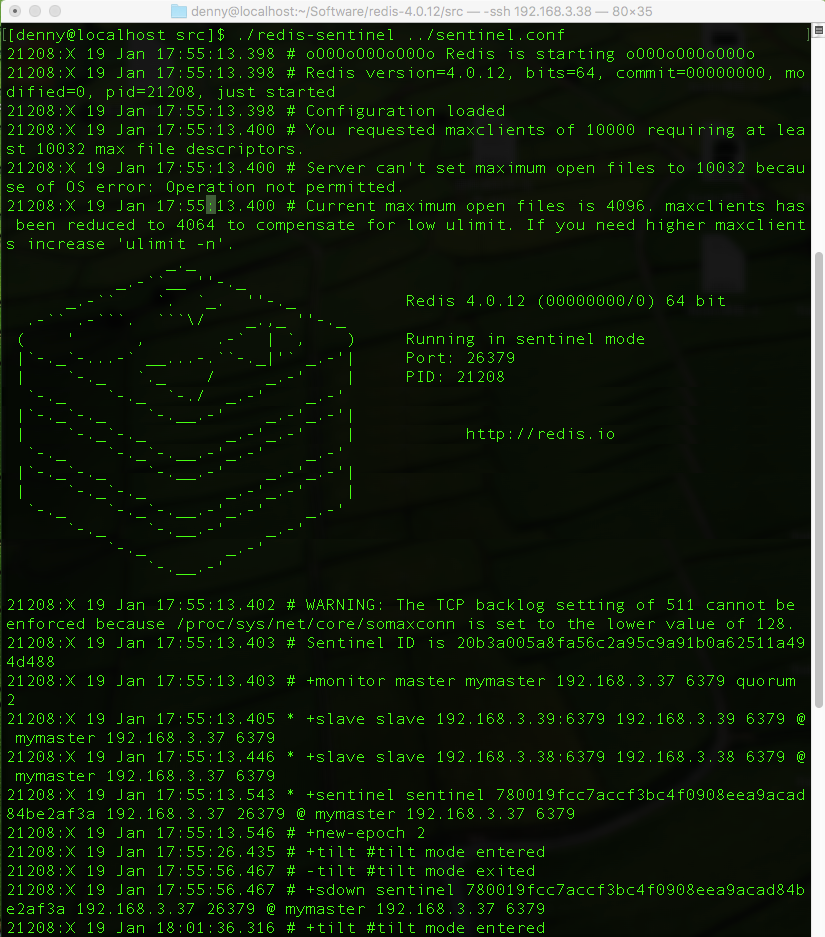
|  |
| --- |
| $ ./redis-server ../redis.conf |

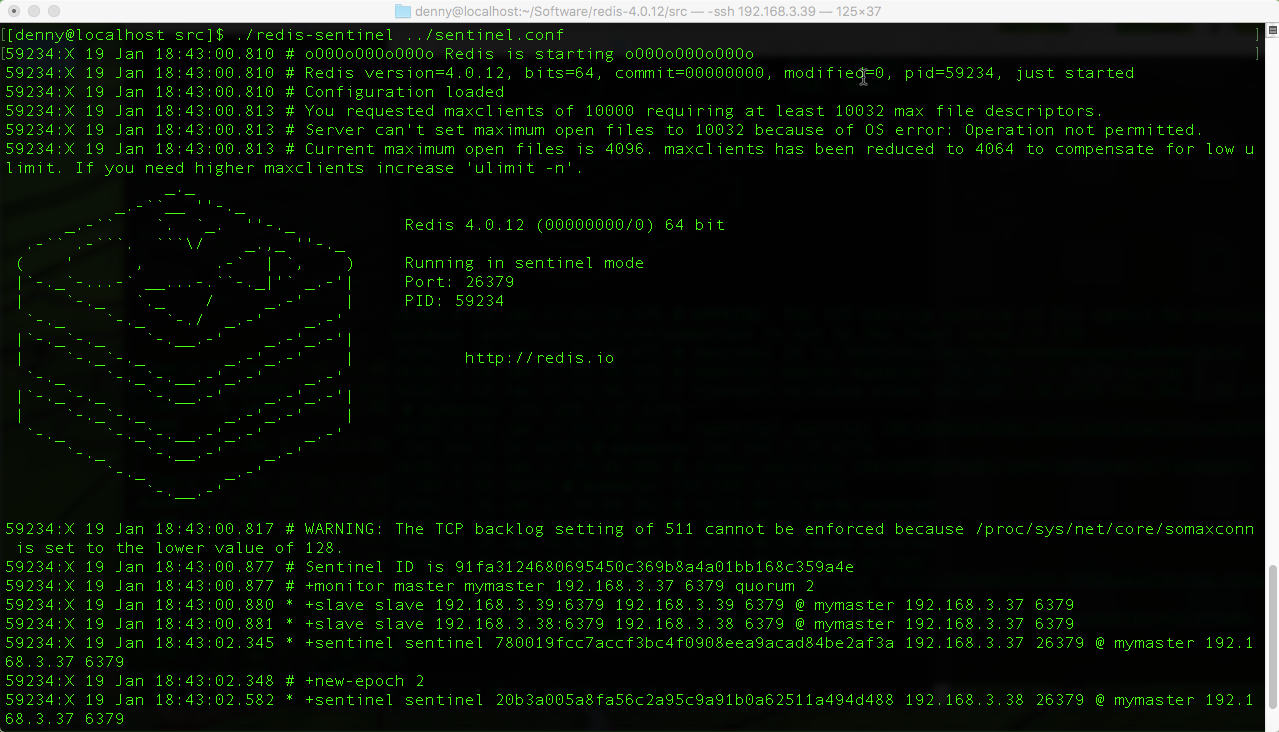
在master和slave节点上依次启动哨兵进程：

|  |
| --- |
| $ ./redis-sentinel ../sentinel.conf |

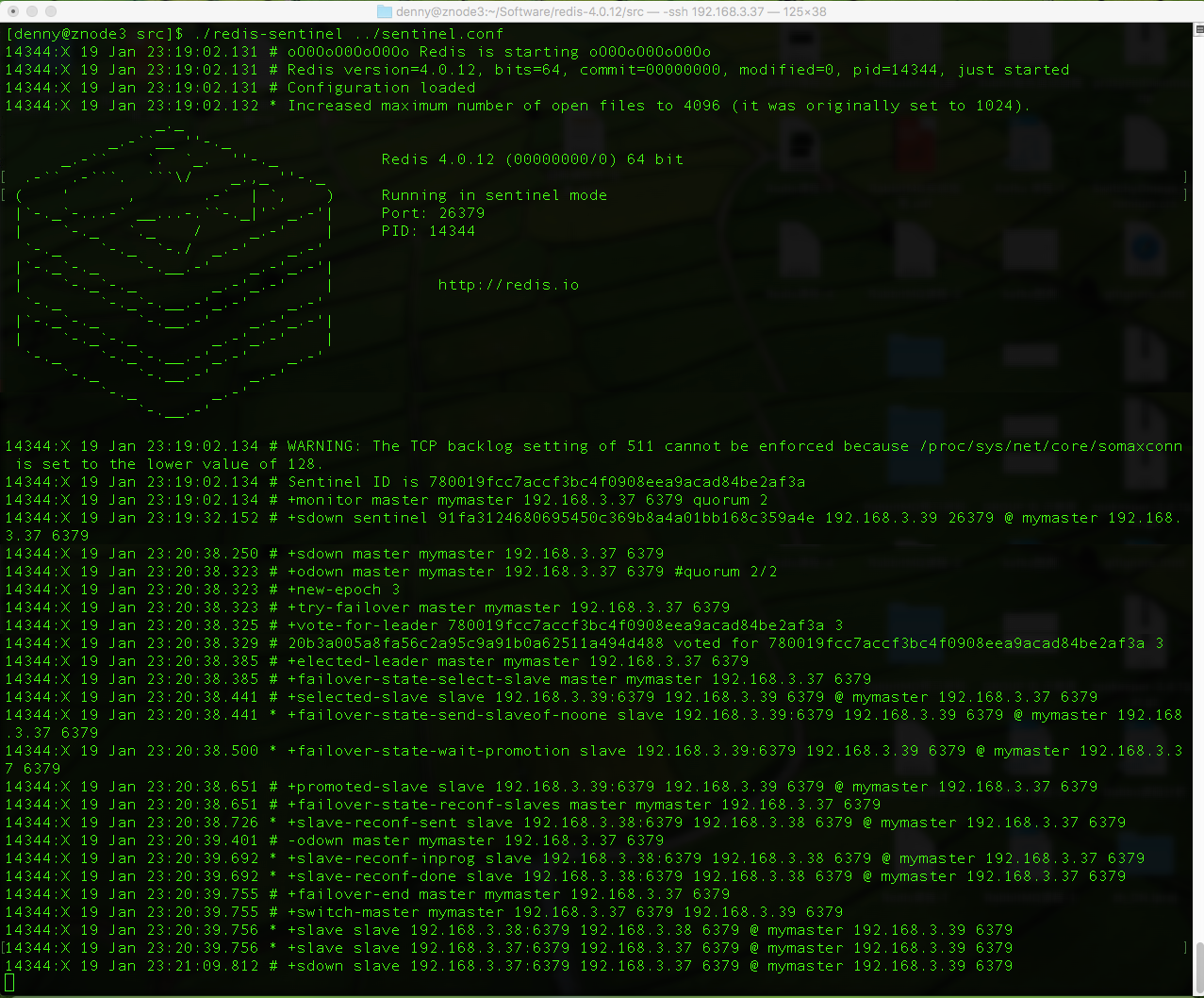
如图：

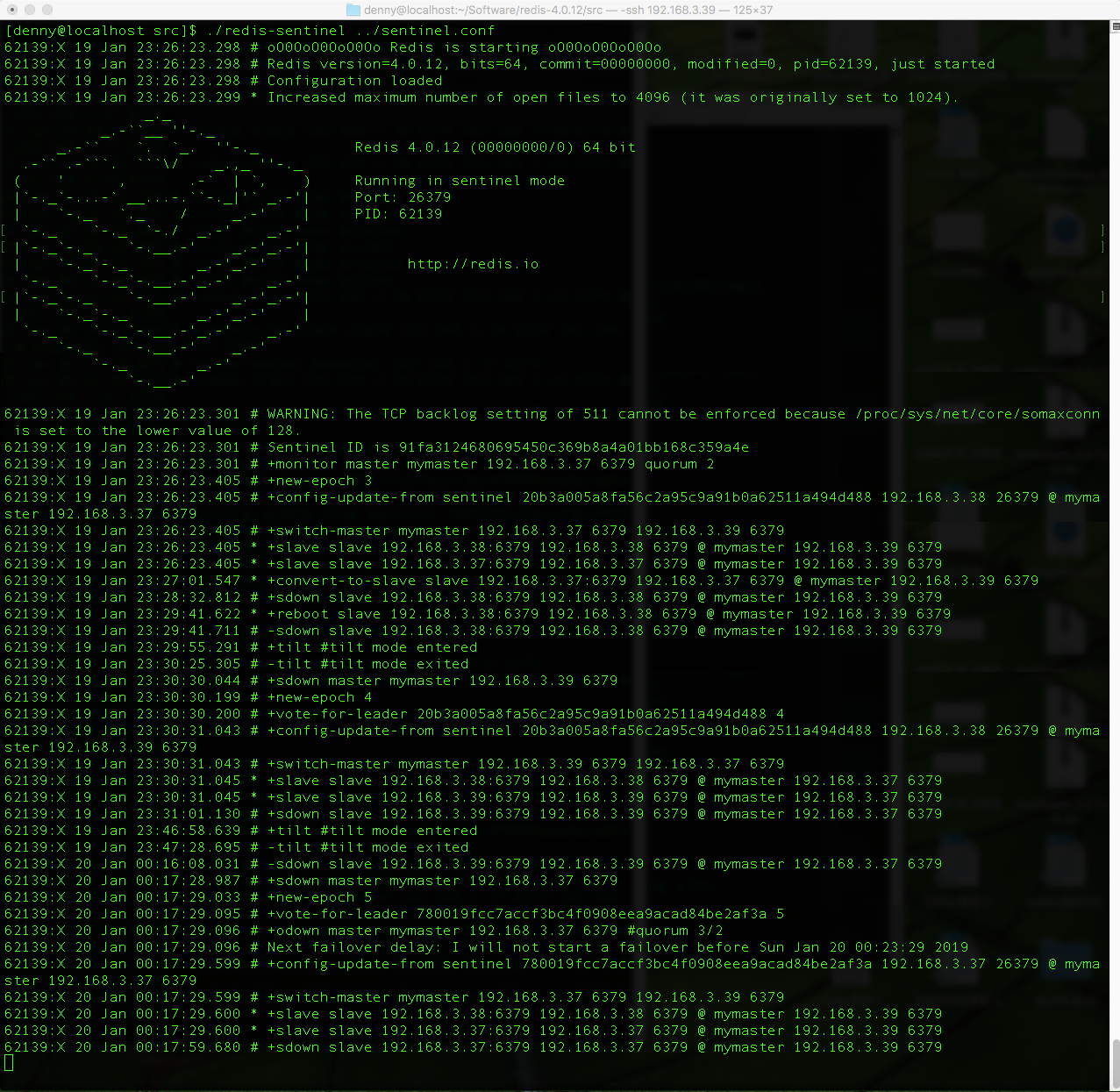






故障转移：





1. Redis-Cluster

数据分片、路由。一致性Hash->codis->redis cluster。

在redis-cluster架构中，redis-master节点一般用于接收读写，而redis-slave节点则一般只用于备份。但如

果不介意读取的是redis-cluster中有可能过期的数据并且对写请求不感兴趣时，则可以通过readonly命令

将slave节点设置为只读模式接受读请求，以实现读写分离。

192.1683.37、192.168.3.38、192.168.3.39搭建Redis Cluster。采用三组一主(Master)二从(Slave)。

端口配置：7000、7001、7002、7003、7004、7005、7006、7007、7008、7001、7001、7001、7002

1. 在192.1683.37、192.1683.38、192.1683.39的 $HOME（即：/home/denny）目录下建立如下目录：192.1683.37：/$HOME/redis-cluster/7000/、/$HOME/redis-cluster/7001/、/$HOME/redis-cluster/7002/

192.1683.38：/$HOME/redis-cluster/7003/、/$HOME/redis-cluster/7004/、/$HOME/redis-cluster/7005/

192.1683.39：/$HOME/redis-cluster/7006/、/$HOME/redis-cluster/7007/、/$HOME/redis-cluster/7008/

1. 在192.1683.37、192.1683.38、192.1683.39的redis安装目录 （即：/home/denny/Software/redis-4.0.12）下建立如下目录：

192.1683.37： /redis-cluster/7000/、/redis-cluster/7001/、/redis-cluster/7002/

192.1683.38： /redis-cluster/7003/、/redis-cluster/7004/、/redis-cluster/7005/

192.1683.39： /redis-cluster/7006/、/redis-cluster/7007/、/redis-cluster/7008/

在192.168.3.37服务器上拷贝一份/home/denny/Software/redis-4.0.12/目录下的redis.conf配置文件d 到：/home/denny/Software/redis-4.0.12/redis-cluster/7000/目录下。修改redis .conf文件中的配置参数如下：

|  |
| --- |
| daemonize yes #开启后台运行  port 7000 #工作端口  bind 192.168.3.37 #绑定机器的内网IP,一定要设置呀老铁，不要用127.0.0.1  dir /home/denny/redis-cluster/7000/ #指定工作目录，rdb,aof持久化文件将会放在该目录下，不同实例一定要配置不同的工作目录  cluster-enabled yes #启用集群模式  cluster-config-file nodes-7000.conf #生成的集群配置文件名称，集群搭建成功后会自动生成，在工作目录下  cluster-node-timeout 5000 #节点宕机发现时间，可以理解为主节点宕机后从节点升级为主节点时间 appendonly yes #开启AOF模式  pidfile /var/run/redis\_7000.pid #pid file所在目录 |

同时将/home/denny/Software/redis-4.0.12/redis-cluster/7000/redis.conf文件拷贝到对应的/7001/和/7002/目录下。执行如下命令修改文件中的端口号：

|  |
| --- |
| $ sed -i 's/7000/7001/g' ./7001/redis.conf  $ sed -i 's/7000/7002/g' ./7002/redis.conf |

将/7000/目录下的redis.conf文件使用scp命令拷贝到192.168.3.38和192.168.3.39服务器上/home/denny/Software/redis-4.0.12/redis-cluster/目录下各个端口命名目录下。参考以上方式在192.168.3.38和192.168.3.39服务器上修改各个不同端口对应的Redis实例的redis.conf，命令如下：

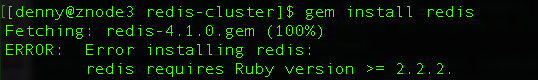
192.168.3.38

|  |
| --- |
| $ sed -i 's/7000/7003/g' ./7003/redis.conf  $ sed -i 's/7000/7004/g' ./7004/redis.conf  $ sed -i 's/7000/7005/g' ./7005/redis.conf  $ sed -i 's/3.37/3.38/g' ./7003/redis.conf  $ sed -i 's/3.37/3.38/g ' ./7004/redis.conf  $ sed -i 's/3.37/3.38/g ' ./7005/redis.conf |

192.168.3.39

|  |
| --- |
| $ sed -i 's/7000/7006/g' ./7006/redis.conf  $ sed -i 's/7000/7007/g' ./7007/redis.conf  $ sed -i 's/7000/7008/g' ./7008/redis.conf  $ sed -i 's/3.37/3.39/g' ./7006/redis.conf  $ sed -i 's/3.37/3.39/g ' ./7007/redis.conf  $ sed -i 's/3.37/3.39/g ' ./7008/redis.conf |

1. 在192.1683.37、192.1683.38、192.1683.39服务器上安装Ruby和RubyGems 。在GentOS-7使用yum安装的Ruby版本是2.0.0版本，但是redis中构建集群的redis-trib.rb工具需要>= 2.2.2版本。



使用如下方式安装ruby-2.2/2.3/2.4以上版本：

安装ruby-2.3版本

|  |
| --- |
| # yum install centos-release-scl-rh  # yum install rh-ruby23 -y  $scl enable rh-ruby23 bash |

|  |
| --- |
| # yum install ruby  # yum install rubygems  # gem install redis --version 3.3.3 |

1. 在192.1683.37、192.1683.38、192.1683.39服务器上安装redisgem，如下：

|  |
| --- |
| $ gem install redis |

1. 在192.1683.37、192.1683.38、192.1683.39服务器上启动各个redis实例，如下：

192.168.3.37的/home/denny/Software/redis-4.0.12/redis-cluster目录下执行

|  |
| --- |
| $ ../src/redis-server ./7000/redis.conf  $ ../src/redis-server ./7001/redis.conf  $ ../src/redis-server ./7002/redis.conf |

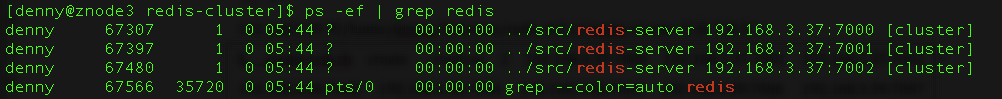
192.168.3.38的/home/denny/Software/redis-4.0.12/redis-cluster目录下执行

|  |
| --- |
| $ ../src/redis-server ./7003/redis.conf  $ ../src/redis-server ./7004/redis.conf  $ ../src/redis-server ./7005/redis.conf |

192.168.3.39的/home/denny/Software/redis-4.0.12/redis-cluster目录下执行

|  |
| --- |
| $ ../src/redis-server ./7006/redis.conf  $ ../src/redis-server ./7007/redis.conf  $ ../src/redis-server ./7008/redis.conf |

查看redis启动，如图：

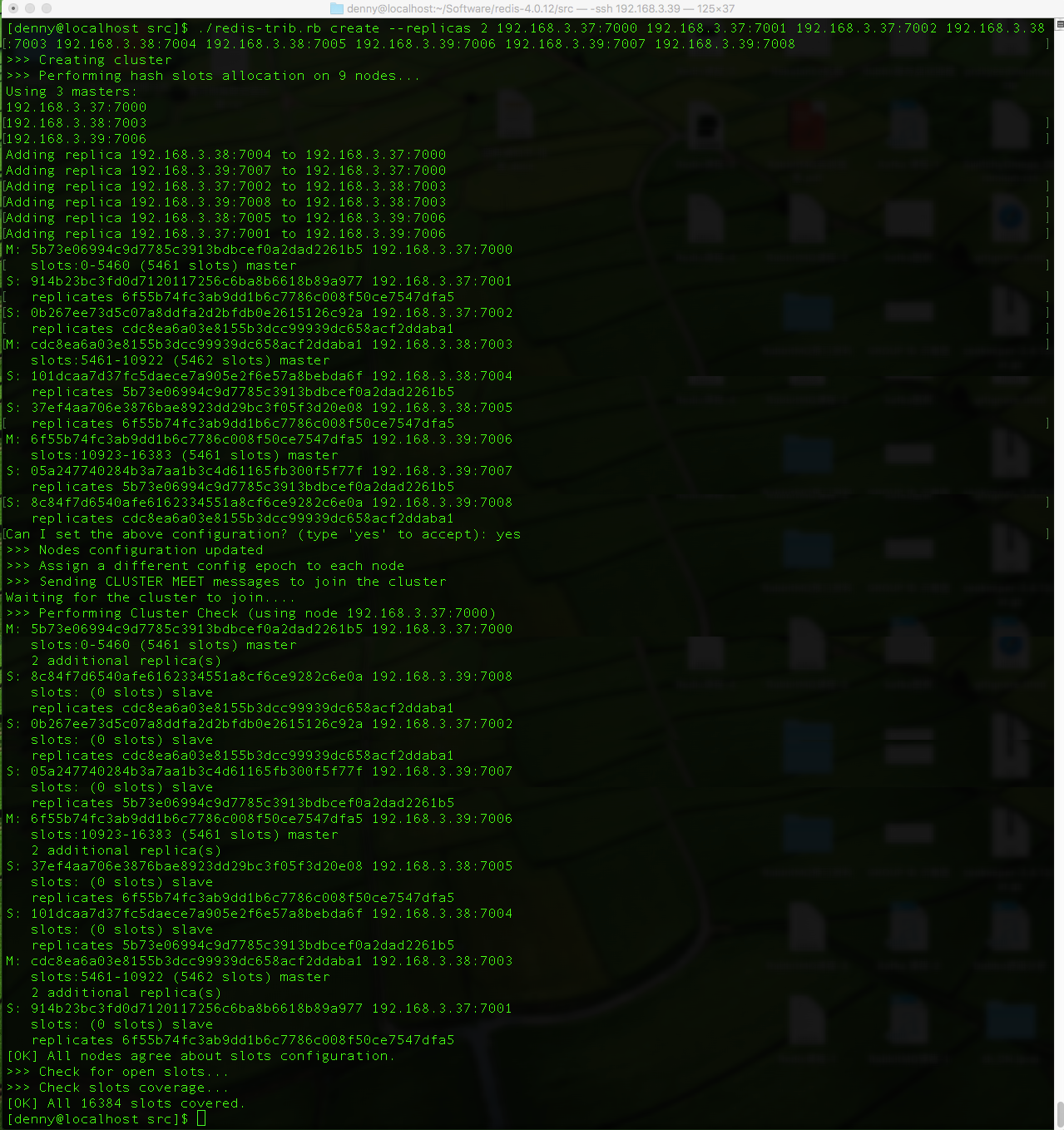


1. 在192.1683.37、192.1683.38、192.1683.39服务器中的一台上使用redis-trib.rb工具创建集群，命令如下：

192.168.3.39的/home/denny/Software/redis-4.0.12/redis-cluster目录下执行

|  |
| --- |
| $ ./redis-trib.rb create --replicas 2 192.168.3.37:7000 192.168.3.37:7001 192.168.3.37:7002 192.168.3.38:7003 192.168.3.38:7004 192.168.3.38:7005 192.168.3.39:7006 192.168.3.39:7007 192.168.3.39:7008 |

如图：



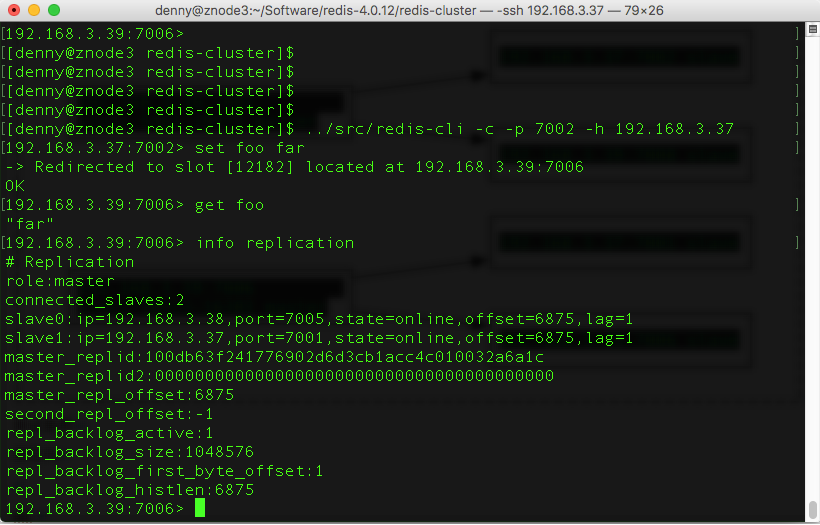
修改进程mit -n 65535 >>/etc/profile

集群示意图：

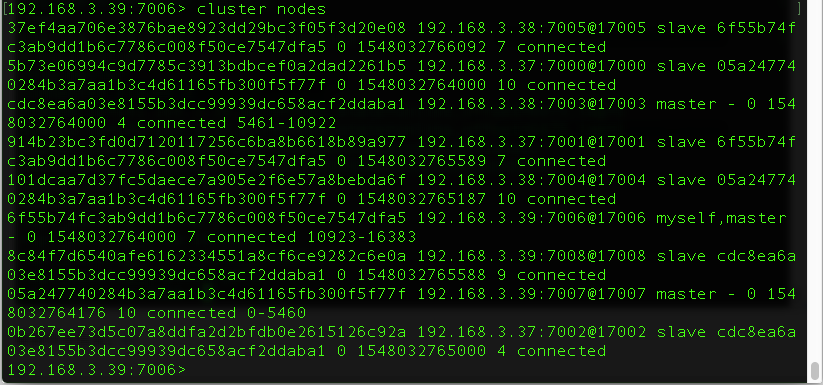


1. 测试集群

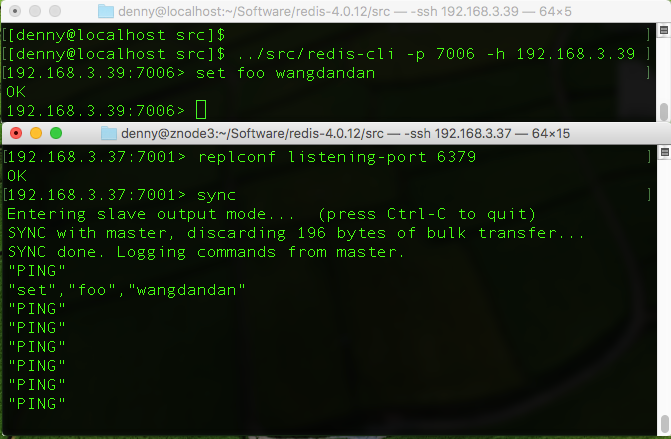
重定向



集群节点信息：

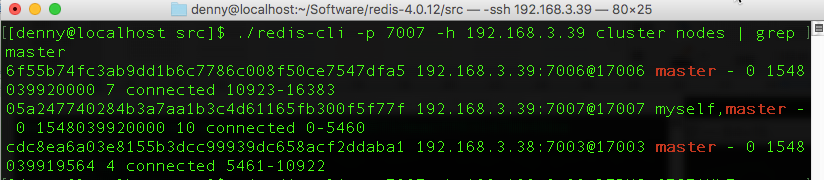


从节点数据同步：

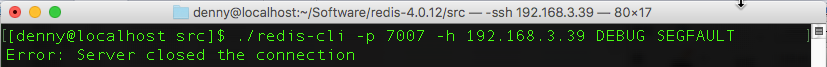


故障转移测试：

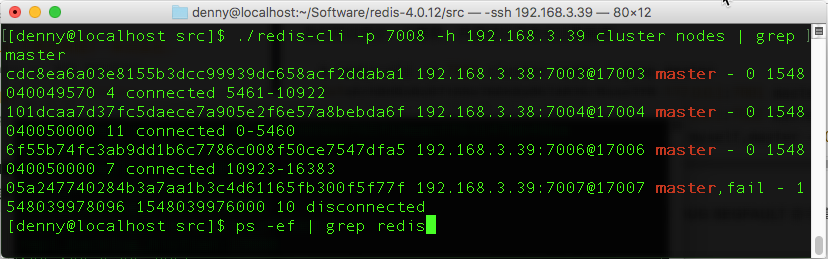
先查看Master信息：



使用DEBUG SEGFAULT 命令搞挂7007 节点：



再次查看master信息：



7007Master节点转移到了 7004节点。

1. 常用监控命令

info server

info replication

info memory

info persistence

info stats

info cup

info cluster

cluster nodes

cluster info

cluster meet

cluster forget

1. Redis的实践应用

Redis Java客户端：jedis、redisson、lettuce。

Jedis 通过socket建立会话，提供了线程池。

Redisson实现了基本命令，同时提供了分布式锁、队列、原子递增等应用功能的封装。

Lettuce基于Netty构建可伸缩的客户端。支持同步、异步和响应式。

分布式锁：

实现方式：

1. 数据库
2. Zookeeper

临时有序节点

1. Redis

redis setnx

SETNX：当且仅当 key 不存在，设置给定的value。若给定的 key 已经存在，则 [SETNX](http://redisdoc.com/string/setnx.html#setnx) 不做任何

动作。

锁的基本操作：获取锁、释放锁、超时时间（避免死锁）、判断是否重入（释放锁的线程是否为已获取锁的线程）。

Redis事务操作：

Multi、Exec、Discard和Watch命令是Redis 事务功能的基础。Redis事务允许在一次单独的步骤中执行一

组命令，保证两个重要事项：

1. Redis会将一个事务中的所有命令序列化，然后按顺序执行。Redis不可能在一个Redis事务的执行过

程中插入执行另一个客户端发出的请求。这样便能保证Redis将这些命令作为一个单独的隔离操作执行。

1. 在一个Redis事务中，Redis要么执行其中的所有命令，要么什么都不执行。因此，Redis事务能够保证原子性。EXEC命令会触发执行事务中的所有命令。

Multi：用于标记事务开始。

Exec：在一个事务中执行所有先前放入队列的命令，然后恢复正常的连接状态。

Discard：清除所有先前在一个事务中放入队列的命令，然后恢复正常的连接状态。

Watch：当某个事务需要按条件执行时，就要使用这个命令将给定的键设置为可控的。

Unwatch：清除所有先前为一个事务监控的键。

使用方法：

使用MULTI命令便可以进入一个Redis事务。这个命令的返回值总是OK。此时，用户可以发出多个Redis命令。Redis会将这些命令放入队列，而不是执行这些命令。一旦调用EXEC命令，那么Redis就会执行事务中的所有命令。

相反，调用DISCARD命令将会清除事务队列，然后退出事务。

Redis使用WATCH命令实现事务的“检查再设置”（CAS）行为，实现乐观锁。

作为WATCH命令的参数的键会受到Redis的监控，Redis能够检测到它们的变化。在执行EXEC命令之前，

如果Redis检测到至少有一个键被修改了，那么整个事务便会中止运行，然后EXEC命令会返回一个Null

值，提醒用户事务运行失败。

**WATCH命令详解**

那么WATCH命令实际做了些什么呢？这个命令会使得EXEC命令在满足某些条件时才会运行事务：我们

要求Redis只有在所有受监控的键都没有被修改时，才会执行事务。（但是，相同的客户端可能会在事务内

部修改这些键，此时这个事务不会中止运行。）否则，Redis根本就不会进入事务。（注意，如果你使用WATCH

命令监控一个易失性的键，然后在你监控这个键之后，Redis再使这个键过期，那么EXEC命令仍然可以

正常工作。）

WATCH命令可以被调用多次。简单说来，所有的WATCH命令都会在被调用之时立刻对相应的键进行监

控，直到EXEC命令被调用之时为止。你可以在单条的WATCH命令之中，使用任意数量的键作为命令参

数。

当调用EXEC命令时，所有的键都会变为未受监控的状态，Redis不会管事务是否被中止。当一个客户单

连接被关闭时，所有的键也都会变为未受监控的状态。

你还可以使用UNWATCH命令（不需要任何参数），这样便能清除所有的受监控键。当我们对某些键施加

乐观锁之后，这个命令有时会非常有用。因为，我们可能需要运行一个用来修改这些键的事务，但是在读

取这些键的当前内容之后，我们可能不打算继续进行操作，此时便可以使用UNWATCH命令，清除所有受

监控的键。在运行UNWATCH命令之后，Redis连接便可以再次自由地用于运行新事务。

Pipeline 管道模式

客户端一次性提交多个操作命令，无需等待服务端返回。大大减少了网络往返时间，提供系统性能。

**Redis应用中的问题：**

1. 数据一致性问题

最终一致性问题

1. 缓存雪崩

如果缓存数据很大，同时大量的KEY存在相同的过期时间。会导致在同一时刻大量的缓存数据过期失效。此时会有大量的请求穿透到数据库层，导致数据库压力过大而故障。

1. 从缓存中取不到值时加锁排队， 通过加锁或排队来控制读读数据库写缓存的线程数量。比如对某一key只允许一个线程查询数据和写缓存，其他线程等待。SETNX实现锁。
2. 缓存过期时间均匀分布
3. 使用多级缓存，如二级缓存，或双缓存策略。

内存缓存、Redis。

1. 缓存穿透

缓存穿透是指查询一个一定不存在的数据，由于缓存是不命中时需要从[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)查询，查不到数据则不写入缓存，这将导致这个不存在的数据每次请求都要到数据库去查询，造成缓存穿透。

1. 对空值进行缓存。缓存空对象，将null变成一个值。

也可以采用一个更为简单粗暴的方法，如果一个查询返回的数据为空（不管是数 据不存在，还是系统故障），我们仍然把这个空结果进行缓存，但它的过期时间会很短，最长不超过五分钟。

 缓存空对象会有两个问题：

 第一，空值做了缓存，意味着缓存层中存了更多的键，需要更多的内存空间 ( 如果是攻击，问题更严重 )，比较有效的方法是针对这类数据设置一个较短的过期时间，让其自动剔除。

 第二，缓存层和存储层的数据会有一段时间窗口的不一致，可能会对业务有一定影响。例如过期时间设置为 5分钟，如果此时存储层添加了这个数据，那此段时间就会出现缓存层和存储层数据的不一致，此时可以利用消息系统或者其他方式清除掉缓存层中的空对象。

1. 设置key 的规则

不满足规则不请求后端。

1. 布隆过滤器

空间效率非常高的概率性算法。

存储40亿数据，一个数据占4个比特位，内存需要16G。

Bitmap （位图）

int 4个字节-32个比特位（可以存储32个十进制的数据）