hash tag：在分布式中，数据一旦分开就很难被整合使用，数据不被分开就一定能使用事物，那么将事务放到一个节点就可以了。

如果key：{oo}k1，{oo}k2这样就可以落到同一个节点。

三种集群：twemproxy（代理模式）、predixy、Redis Cluster

在linux安装git：

yum -y install git

在光标的位置按“yy”，复制当前行；然后在光标的位置按“p”，粘贴到下一行。

vim快捷键：d+G删除光标之后的所有数据；2+dd删除光标下的两行。

关闭redis方法：redis-cli –p 6380 shutdown

### 集群一：twemproxy

地址：<https://github.com/twitter/twemproxy>

1.在家目录创建文件夹：make twemproxy

2.下载：git clone https://github.com/twitter/twemproxy.git

如果报错：HTTP request failed

使用命令：yum update nss命令，升级一下就行。

3.cd twemproxy

4.yum install automake libtool –y

5.autoreconf –fvi -- 执行完后多了一个configure

6../configure

7.make

8.cd src -- 有个nutcracker可执行程序

9.cp nutcracker /usr/bin -- 将nutcracker设置为环境变量

10.cd ../scripts

cp nutcracker.init /etc/init.d/twemproxy

cd /etc/init.d

chmod +x twemproxy --通过查看此文件知道，需要/etc/nutcracker/nutcracker.yml文件

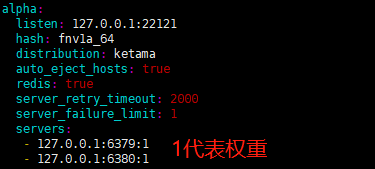
11.mkdir /etc/nutcracker

cd ~/twemproxy/twemproxy/conf

cp ./\* /etc/nutcracker/

cd /etc/nutcracker/

修改nutcracker.yml：vim nutcracker



12.启动两个实例，6379和6380

在家目录新增data/6379和data/6380两个目录

cd 6379

redis-server --port 6379

cd 6380

redis-server --port 6380

redis手工启动，就拿当前目录做持久化目录

13.启动

service twemproxy start

连接redis实例：redis-cli –p 22121

14.22121端口不能使用keys \*命令

支持hash tag

不支持watch、事务命令，因为数据分治。

15.关闭：service twemproxy stop

### 集群二：predixy

地址：<https://github.com/joyieldInc/predixy>

predixy支持Redis Sentinel和Redis Cluster来使用redis，一个配置里这两种形式只能出现一种。

1.在家目录创建predixy目录，并进入

mkdir predixy

cd predixy

2.下载，在github的release有编译完的，直接下载。

wget <https://github.com/joyieldInc/predixy/releases/download/1.0.5/predixy-1.0.5-bin-amd64-linux.tar.gz>

3.解压：tar xf predixy-1.0.5-bin-amd64-linux.tar.gz

4.cd predixy-1.0.5

5.cd conf

6.vim predixy.conf

解开7617的注释：Bind 127.0.0.1:7617

解开哨兵的配置文件：Include sentinel.conf

注释掉默认的：# Include try.conf

7.vim sentinel.conf

方案一：一套哨兵监控两套主从复制

|  |
| --- |
| SentinelServerPool {  Databases 16  Hash crc16  HashTag "{}"  Distribution modula  MasterReadPriority 60  StaticSlaveReadPriority 50  DynamicSlaveReadPriority 50  RefreshInterval 1  ServerTimeout 1  ServerFailureLimit 10  ServerRetryTimeout 1  KeepAlive 120  Sentinels {  + 127.0.0.1:26379  + 127.0.0.1:26380  + 127.0.0.1:26381  }  Group ooxx {  }  Group xxoo {  }  } |

ooxx和xxoo是两套主从复制，这里的ooxx和xxoo就是哨兵配置文件monitor后的名字。

cd ~/predixy/predixy-1.0.5/bin 等待启动

去test目录中，修改哨兵文件并启动 cd ~/test

vim 26379.conf

|  |
| --- |
| port 26379  sentinel monitor ooxx 127.0.0.1 36379 2  sentinel monitor xxoo 127.0.0.1 46379 2 |

配置两套，36379是主，36380是从；46379是主，46380是从。

这是26379的哨兵，它要监控36379和46379的redis的master。

vim 26380.conf

|  |
| --- |
| port 26380  sentinel monitor ooxx 127.0.0.1 36379 2  sentinel monitor xxoo 127.0.0.1 46379 2 |

vim 26381.conf

|  |
| --- |
| port 26381  sentinel monitor ooxx 127.0.0.1 36379 2  sentinel monitor xxoo 127.0.0.1 46379 2 |

开三个页面，启动三个哨兵：

redis-server 26379.conf --sentinel

redis-server 26380.conf --sentinel

redis-server 26381.conf --sentinel

开四个页面，启动两套主从复制集群：

在test目录下，创建36379、36380、46379、46380四个目录

分别进入四个目录启动：

redis-server --port 36379

redis-server --port 36380 --replicaof 127.0.0.1 36379

redis-server --port 46379

redis-server --port 46380 --replicaof 127.0.0.1 46379

启动predixy：./predixy ../conf/predixy.conf

启动后predixy可以发现三台哨兵，哨兵可以发现四台主从关系。

通过7617端口连接redis：redis-cli –p 7617

支持hash tag

不支持key \*

不支持watch，因为代理后面是两套主从，是分治的。

方案二：

|  |
| --- |
| SentinelServerPool {  Databases 16  Hash crc16  HashTag "{}"  Distribution modula  MasterReadPriority 60  StaticSlaveReadPriority 50  DynamicSlaveReadPriority 50  RefreshInterval 1  ServerTimeout 1  ServerFailureLimit 10  ServerRetryTimeout 1  KeepAlive 120  Sentinels {  + 127.0.0.1:26379  + 127.0.0.1:26380  + 127.0.0.1:26381  }  Group ooxx {  }  } |

此时数据只会向ooxx的主从复制中存放。

重新启动predixy：./predixy ../conf/sentinel.conf

此时只能发现36379和36380这一套主从复制。

不支持key \*

支持watch

此时如果36379挂了，连接7617是没影响的，过一段时间投票重新选举后可以继续使用。

### 集群三：Redis Cluster

Redis集群有16384个哈希槽

Redis自己组建一套无主模型的cluster且每台主机每个实例得认领一些槽位。

启动几台？每台认领什么槽位？如果其中一个挂了数据是不是丢失了？redis自身是不是有哨兵在三个主里面挂一个从？

#### 方案一：

1. cd ~/redis-5.0.5/utils/create-cluster

./create-cluster start --启动

./create-cluster create --分槽位

./create-cluster stop --停止

./create-cluster clean --清理

2. vim create-cluster

|  |
| --- |
| NODES=6  REPLICAS=1 |

6个实例，三主三从。

3. ./create-cluster start --启动6个实例

4. ./create-cluster create --分槽位，会显示具体的分赃信息、主从信息，输入yes使用默认分赃信息。

创建了30001-30006六个实例，都可以连接。

连接方式：redis-cli -c -p 30001

5. 支持hash tag、支持key \*

6.假设k1在节点1，k2在节点2，此时watch k1，然后在节点1开启事务，如果此时修改k2，会跳转到节点2，事务就没用了。但是可以使用hash tag的方式，在同一个节点就可以使用watch和事务了。

7.关闭并清理

./create-cluster stop

./create-cluster clean

#### 方案二：

1.启动

./create-cluster start --启动6个实例

2.分赃和启动自己操作

redis-cli --cluster help --查看帮助文档

|  |
| --- |
| redis-cli --cluster create 127.0.0.1:30001 127.0.0.1:30002 127.0.0.1:30003 127.0.0.1:30004 127.0.0.1:30005 127.0.0.1:30006 --cluster-replicas 1  输入yes，使用默认的分赃 |

--cluster-replicas 1 副本的数量

可以用这种方式启动六台物理机，搭建一套分布式的集群。

3. 连接方式：redis-cli -c -p 30001

4. reshard移动槽位，解决倾斜问题

redis-cli --cluster reshard 127.0.0.1:30001 随便连接一个存活的节点

How many slots do you want to move (from 1 to 16384)? 2000 选择要移动的槽位数

What is the receiving node ID? 1fb249f1e200e7fcf6348a893c2c003b893739b9 移动给谁，此处是30002的id

Source node #1: b963c10f44b8ae6aeb1186f633911e2000da8c16 选择从谁那里抽取槽位，all是都抽取部分，或者输入某一个的id，此处从30001抽取。回车后还可以接着输入其它节点继续抽取，或者输入done停止。

Do you want to proceed with the proposed reshard plan (yes/no)? yes 确认要移动的槽位

5.查看槽位信息

redis-cli --cluster check 127.0.0.1:30001