Redis集群详解

Redis有三种集群模式，分别是：

\* 主从模式

\* Sentinel模式

\* Cluster模式

1

2

3

4

5

三种集群模式各有特点，关于Redis介绍可以参考这里：NoSQL（二)——Redis

Redis官网：https://redis.io/ ，最新版本5.0.4

主从模式

主从模式介绍

主从模式是三种模式中最简单的，在主从复制中，数据库分为两类：主数据库(master)和从数据库(slave)。

其中主从复制有如下特点：

\* 主数据库可以进行读写操作，当读写操作导致数据变化时会自动将数据同步给从数据库

\* 从数据库一般都是只读的，并且接收主数据库同步过来的数据

\* 一个master可以拥有多个slave，但是一个slave只能对应一个master

\* slave挂了不影响其他slave的读和master的读和写，重新启动后会将数据从master同步过来

\* master挂了以后，不影响slave的读，但redis不再提供写服务，master重启后redis将重新对外提供写服务

\* master挂了以后，不会在slave节点中重新选一个master

1

2

3

4

5

6

7

8

9

10

11

工作机制：

当slave启动后，主动向master发送SYNC命令。master接收到SYNC命令后在后台保存快照（RDB持久化）和缓存保存快照这段时间的命令，然后将保存的快照文件和缓存的命令发送给slave。slave接收到快照文件和命令后加载快照文件和缓存的执行命令。

复制初始化后，master每次接收到的写命令都会同步发送给slave，保证主从数据一致性。

安全设置：

当master节点设置密码后，

客户端访问master需要密码

启动slave需要密码，在配置文件中配置即可

客户端访问slave不需要密码

1

2

3

4

5

缺点：

从上面可以看出，master节点在主从模式中唯一，若master挂掉，则redis无法对外提供写服务。

主从模式搭建

环境准备：

master节点 192.168.30.128

slave节点 192.168.30.129

slave节点 192.168.30.130

1

2

3

4

5

全部下载安装：

# cd /software

# wget http://download.redis.io/releases/redis-5.0.4.tar.gz

# tar zxf redis-5.0.4.tar.gz && mv redis-5.0.4/ /usr/local/redis

# cd /usr/local/redis && make && make install

# echo $?

0

1

2

3

4

5

6

7

8

9

10

全部配置成服务：

服务文件

# vim /usr/lib/systemd/system/redis.service

[Unit]

Description=Redis persistent key-value database

After=network.target

After=network-online.target

Wants=network-online.target

[Service]

ExecStart=/usr/local/bin/redis-server /usr/local/redis/redis.conf --supervised systemd

ExecStop=/usr/libexec/redis-shutdown

Type=notify

User=redis

Group=redis

RuntimeDirectory=redis

RuntimeDirectoryMode=0755

[Install]

WantedBy=multi-user.target

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

shutdown脚本

# vim /usr/libexec/redis-shutdown

#!/bin/bash

#

# Wrapper to close properly redis and sentinel

test x"$REDIS\_DEBUG" != x && set -x

REDIS\_CLI=/usr/local/bin/redis-cli

# Retrieve service name

SERVICE\_NAME="$1"

if [ -z "$SERVICE\_NAME" ]; then

SERVICE\_NAME=redis

fi

# Get the proper config file based on service name

CONFIG\_FILE="/usr/local/redis/$SERVICE\_NAME.conf"

# Use awk to retrieve host, port from config file

HOST=`awk '/^[[:blank:]]\*bind/ { print $2 }' $CONFIG\_FILE | tail -n1`

PORT=`awk '/^[[:blank:]]\*port/ { print $2 }' $CONFIG\_FILE | tail -n1`

PASS=`awk '/^[[:blank:]]\*requirepass/ { print $2 }' $CONFIG\_FILE | tail -n1`

SOCK=`awk '/^[[:blank:]]\*unixsocket\s/ { print $2 }' $CONFIG\_FILE | tail -n1`

# Just in case, use default host, port

HOST=${HOST:-127.0.0.1}

if [ "$SERVICE\_NAME" = redis ]; then

PORT=${PORT:-6379}

else

PORT=${PORT:-26739}

fi

# Setup additional parameters

# e.g password-protected redis instances

[ -z "$PASS" ] || ADDITIONAL\_PARAMS="-a $PASS"

# shutdown the service properly

if [ -e "$SOCK" ] ; then

$REDIS\_CLI -s $SOCK $ADDITIONAL\_PARAMS shutdown

else

$REDIS\_CLI -h $HOST -p $PORT $ADDITIONAL\_PARAMS shutdown

fi

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

# chmod +x /usr/libexec/redis-shutdown

# useradd -s /sbin/nologin redis

# chown -R redis:redis /usr/local/redis

# chown -R reids:redis /data/redis

# yum install -y bash-completion && source /etc/profile #命令补全

# systemctl daemon-reload

# systemctl enable redis

1

2

3

4

5

6

7

8

9

10

11

12

13

修改配置：

192.168.30.128

# mkdir -p /data/redis

# vim /usr/local/redis/redis.conf

bind 192.168.30.128 #监听ip，多个ip用空格分隔

daemonize yes #允许后台启动

logfile "/usr/local/redis/redis.log" #日志路径

dir /data/redis #数据库备份文件存放目录

masterauth 123456 #slave连接master密码，master可省略

requirepass 123456 #设置master连接密码，slave可省略

appendonly yes #在/data/redis/目录生成appendonly.aof文件，将每一次写操作请求都追加到appendonly.aof 文件中

# echo 'vm.overcommit\_memory=1' >> /etc/sysctl.conf

# sysctl -p

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

192.168.30.129

# mkdir -p /data/redis

# vim /usr/local/redis/redis.conf

bind 192.168.30.129

daemonize yes

logfile "/usr/local/redis/redis.log"

dir /data/redis

replicaof 192.168.30.128 6379

masterauth 123456

requirepass 123456

appendonly yes

# echo 'vm.overcommit\_memory=1' >> /etc/sysctl.conf

# sysctl -p

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

192.168.30.130

# mkdir -p /data/redis

# vim /usr/local/redis/redis.conf

bind 192.168.30.130

daemonize yes

logfile "/usr/local/redis/redis.log"

dir /data/redis

replicaof 192.168.30.128 6379

masterauth 123456

requirepass 123456

appendonly yes

# echo 'vm.overcommit\_memory=1' >> /etc/sysctl.conf

# sysctl -p

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

全部启动redis：

# systemctl start redis

1

查看集群状态：

# redis-cli -h 192.168.30.128 -a 123456

Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.

192.168.30.128:6379> info replication

# Replication

role:master

connected\_slaves:2

slave0:ip=192.168.30.129,port=6379,state=online,offset=168,lag=1

slave1:ip=192.168.30.130,port=6379,state=online,offset=168,lag=1

master\_replid:fb4941e02d5032ad74c6e2383211fc58963dbe90

master\_replid2:0000000000000000000000000000000000000000

master\_repl\_offset:168

second\_repl\_offset:-1

repl\_backlog\_active:1

repl\_backlog\_size:1048576

repl\_backlog\_first\_byte\_offset:1

repl\_backlog\_histlen:168

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

# redis-cli -h 192.168.30.129 -a 123456 info replication

Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.

# Replication

role:slave

master\_host:192.168.30.128

master\_port:6379

master\_link\_status:up

master\_last\_io\_seconds\_ago:1

master\_sync\_in\_progress:0

slave\_repl\_offset:196

slave\_priority:100

slave\_read\_only:1

connected\_slaves:0

master\_replid:fb4941e02d5032ad74c6e2383211fc58963dbe90

master\_replid2:0000000000000000000000000000000000000000

master\_repl\_offset:196

second\_repl\_offset:-1

repl\_backlog\_active:1

repl\_backlog\_size:1048576

repl\_backlog\_first\_byte\_offset:1

repl\_backlog\_histlen:196

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

数据演示：

192.168.30.128:6379> keys \*

(empty list or set)

192.168.30.128:6379> set key1 100

OK

192.168.30.128:6379> set key2 lzx

OK

192.168.30.128:6379> keys \*

1) "key1"

2) "key2"

1

2

3

4

5

6

7

8

9

10

11

12

# redis-cli -h 192.168.30.129 -a 123456

Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.

192.168.30.129:6379> keys \*

1) "key2"

2) "key1"

192.168.30.129:6379> CONFIG GET dir

1) "dir"

2) "/data/redis"

192.168.30.129:6379> CONFIG GET dbfilename

1) "dbfilename"

2) "dump.rdb"

192.168.30.129:6379> get key1

"100"

192.168.30.129:6379> get key2

"lzx"

192.168.30.129:6379> set key3 aaa

(error) READONLY You can't write against a read only replica.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

# redis-cli -h 192.168.30.130 -a 123456

Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.

192.168.30.130:6379> keys \*

1) "key2"

2) "key1"

192.168.30.130:6379> CONFIG GET dir

1) "dir"

2) "/data/redis"

192.168.30.130:6379> CONFIG GET dbfilename

1) "dbfilename"

2) "dump.rdb"

192.168.30.130:6379> get key1

"100"

192.168.30.130:6379> get key2

"lzx"

192.168.30.130:6379> set key3 aaa

(error) READONLY You can't write against a read only replica.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

可以看到，在master节点写入的数据，很快就同步到slave节点上，而且在slave节点上无法写入数据。

Sentinel模式

Sentinel模式介绍

主从模式的弊端就是不具备高可用性，当master挂掉以后，Redis将不能再对外提供写入操作，因此sentinel应运而生。

sentinel中文含义为哨兵，顾名思义，它的作用就是监控redis集群的运行状况，特点如下：

\* sentinel模式是建立在主从模式的基础上，如果只有一个Redis节点，sentinel就没有任何意义

\* 当master挂了以后，sentinel会在slave中选择一个做为master，并修改它们的配置文件，其他slave的配置文件也会被修改，比如slaveof属性会指向新的master

\* 当master重新启动后，它将不再是master而是做为slave接收新的master的同步数据

\* sentinel因为也是一个进程有挂掉的可能，所以sentinel也会启动多个形成一个sentinel集群

\* 多sentinel配置的时候，sentinel之间也会自动监控

\* 当主从模式配置密码时，sentinel也会同步将配置信息修改到配置文件中，不需要担心

\* 一个sentinel或sentinel集群可以管理多个主从Redis，多个sentinel也可以监控同一个redis

\* sentinel最好不要和Redis部署在同一台机器，不然Redis的服务器挂了以后，sentinel也挂了

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

工作机制：

\* 每个sentinel以每秒钟一次的频率向它所知的master，slave以及其他sentinel实例发送一个 PING 命令

\* 如果一个实例距离最后一次有效回复 PING 命令的时间超过 down-after-milliseconds 选项所指定的值， 则这个实例会被sentinel标记为主观下线。

\* 如果一个master被标记为主观下线，则正在监视这个master的所有sentinel要以每秒一次的频率确认master的确进入了主观下线状态

\* 当有足够数量的sentinel（大于等于配置文件指定的值）在指定的时间范围内确认master的确进入了主观下线状态， 则master会被标记为客观下线

\* 在一般情况下， 每个sentinel会以每 10 秒一次的频率向它已知的所有master，slave发送 INFO 命令

\* 当master被sentinel标记为客观下线时，sentinel向下线的master的所有slave发送 INFO 命令的频率会从 10 秒一次改为 1 秒一次

\* 若没有足够数量的sentinel同意master已经下线，master的客观下线状态就会被移除；

若master重新向sentinel的 PING 命令返回有效回复，master的主观下线状态就会被移除

1

2

3

4

5

6

7

8

9

10

11

12

13

14

当使用sentinel模式的时候，客户端就不要直接连接Redis，而是连接sentinel的ip和port，由sentinel来提供具体的可提供服务的Redis实现，这样当master节点挂掉以后，sentinel就会感知并将新的master节点提供给使用者。

Sentinel模式搭建

环境准备：

master节点 192.168.30.128 sentinel端口：26379

slave节点 192.168.30.129 sentinel端口：26379

slave节点 192.168.30.130 sentinel端口：26379

1

2

3

4

5

修改配置：

前面已经下载安装了redis，这里省略，直接修改sentinel配置文件。

192.168.30.128

# vim /usr/local/redis/sentinel.conf

daemonize yes

logfile "/usr/local/redis/sentinel.log"

dir "/usr/local/redis/sentinel" #sentinel工作目录

sentinel monitor mymaster 192.168.30.128 6379 2 #判断master失效至少需要2个sentinel同意，建议设置为n/2+1，n为sentinel个数

sentinel auth-pass mymaster 123456

sentinel down-after-milliseconds mymaster 30000 #判断master主观下线时间，默认30s

1

2

3

4

5

6

7

8

这里需要注意，sentinel auth-pass mymaster 123456需要配置在sentinel monitor mymaster 192.168.30.128 6379 2下面，否则启动报错：

# /usr/local/bin/redis-sentinel /usr/local/redis/sentinel.conf

\*\*\* FATAL CONFIG FILE ERROR \*\*\*

Reading the configuration file, at line 104

>>> 'sentinel auth-pass mymaster 123456'

No such master with specified name.

1

2

3

4

5

6

全部启动sentinel：

# mkdir /usr/local/redis/sentinel && chown -R redis:redis /usr/local/redis

# /usr/local/bin/redis-sentinel /usr/local/redis/sentinel.conf

1

2

3

任一主机查看日志：

# tail -f /usr/local/redis/sentinel.log

21574:X 09 May 2019 15:32:04.298 # Sentinel ID is 30c417116a8edbab09708037366c4a7471beb770

21574:X 09 May 2019 15:32:04.298 # +monitor master mymaster 192.168.30.128 6379 quorum 2

21574:X 09 May 2019 15:32:04.299 \* +slave slave 192.168.30.129:6379 192.168.30.129 6379 @ mymaster 192.168.30.128 6379

21574:X 09 May 2019 15:32:04.300 \* +slave slave 192.168.30.130:6379 192.168.30.130 6379 @ mymaster 192.168.30.128 6379

21574:X 09 May 2019 15:32:16.347 \* +sentinel sentinel 79b8d61626afd4d059fb5a6a63393e9a1374e78f 192.168.30.129 26379 @ mymaster 192.168.30.128 6379

21574:X 09 May 2019 15:32:31.584 \* +sentinel sentinel d7b429dcba792103ef0d80827dd0910bd9284d21 192.168.30.130 26379 @ mymaster 192.168.30.128 6379

1

2

3

4

5

6

7

8

Sentinel模式下的几个事件：

· +reset-master ：主服务器已被重置。

· +slave ：一个新的从服务器已经被 Sentinel 识别并关联。

· +failover-state-reconf-slaves ：故障转移状态切换到了 reconf-slaves 状态。

· +failover-detected ：另一个 Sentinel 开始了一次故障转移操作，或者一个从服务器转换成了主服务器。

· +slave-reconf-sent ：领头（leader）的 Sentinel 向实例发送了 [SLAVEOF](/commands/slaveof.html) 命令，为实例设置新的主服务器。

· +slave-reconf-inprog ：实例正在将自己设置为指定主服务器的从服务器，但相应的同步过程仍未完成。

· +slave-reconf-done ：从服务器已经成功完成对新主服务器的同步。

· -dup-sentinel ：对给定主服务器进行监视的一个或多个 Sentinel 已经因为重复出现而被移除 —— 当 Sentinel 实例重启的时候，就会出现这种情况。

· +sentinel ：一个监视给定主服务器的新 Sentinel 已经被识别并添加。

· +sdown ：给定的实例现在处于主观下线状态。

· -sdown ：给定的实例已经不再处于主观下线状态。

· +odown ：给定的实例现在处于客观下线状态。

· -odown ：给定的实例已经不再处于客观下线状态。

· +new-epoch ：当前的纪元（epoch）已经被更新。

· +try-failover ：一个新的故障迁移操作正在执行中，等待被大多数 Sentinel 选中（waiting to be elected by the majority）。

· +elected-leader ：赢得指定纪元的选举，可以进行故障迁移操作了。

· +failover-state-select-slave ：故障转移操作现在处于 select-slave 状态 —— Sentinel 正在寻找可以升级为主服务器的从服务器。

· no-good-slave ：Sentinel 操作未能找到适合进行升级的从服务器。Sentinel 会在一段时间之后再次尝试寻找合适的从服务器来进行升级，又或者直接放弃执行故障转移操作。

· selected-slave ：Sentinel 顺利找到适合进行升级的从服务器。

· failover-state-send-slaveof-noone ：Sentinel 正在将指定的从服务器升级为主服务器，等待升级功能完成。

· failover-end-for-timeout ：故障转移因为超时而中止，不过最终所有从服务器都会开始复制新的主服务器（slaves will eventually be configured to replicate with the new master anyway）。

· failover-end ：故障转移操作顺利完成。所有从服务器都开始复制新的主服务器了。

· +switch-master ：配置变更，主服务器的 IP 和地址已经改变。 这是绝大多数外部用户都关心的信息。

· +tilt ：进入 tilt 模式。

· -tilt ：退出 tilt 模式。

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

master宕机演示：

192.168.30.128

# systemctl stop redis

# tail -f /usr/local/redis/sentinel.log

22428:X 09 May 2019 15:51:29.287 # +sdown master mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:29.371 # +odown master mymaster 192.168.30.128 6379 #quorum 2/2

22428:X 09 May 2019 15:51:29.371 # +new-epoch 1

22428:X 09 May 2019 15:51:29.371 # +try-failover master mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:29.385 # +vote-for-leader 30c417116a8edbab09708037366c4a7471beb770 1

22428:X 09 May 2019 15:51:29.403 # d7b429dcba792103ef0d80827dd0910bd9284d21 voted for 30c417116a8edbab09708037366c4a7471beb770 1

22428:X 09 May 2019 15:51:29.408 # 79b8d61626afd4d059fb5a6a63393e9a1374e78f voted for 30c417116a8edbab09708037366c4a7471beb770 1

22428:X 09 May 2019 15:51:29.451 # +elected-leader master mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:29.451 # +failover-state-select-slave master mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:29.528 # +selected-slave slave 192.168.30.129:6379 192.168.30.129 6379 @ mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:29.528 \* +failover-state-send-slaveof-noone slave 192.168.30.129:6379 192.168.30.129 6379 @ mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:29.594 \* +failover-state-wait-promotion slave 192.168.30.129:6379 192.168.30.129 6379 @ mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:30.190 # +promoted-slave slave 192.168.30.129:6379 192.168.30.129 6379 @ mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:30.190 # +failover-state-reconf-slaves master mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:30.258 \* +slave-reconf-sent slave 192.168.30.130:6379 192.168.30.130 6379 @ mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:30.511 # -odown master mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:31.233 \* +slave-reconf-inprog slave 192.168.30.130:6379 192.168.30.130 6379 @ mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:31.233 \* +slave-reconf-done slave 192.168.30.130:6379 192.168.30.130 6379 @ mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:31.297 # +failover-end master mymaster 192.168.30.128 6379

22428:X 09 May 2019 15:51:31.297 # +switch-master mymaster 192.168.30.128 6379 192.168.30.129 6379

22428:X 09 May 2019 15:51:31.298 \* +slave slave 192.168.30.130:6379 192.168.30.130 6379 @ mymaster 192.168.30.129 6379

22428:X 09 May 2019 15:51:31.298 \* +slave slave 192.168.30.128:6379 192.168.30.128 6379 @ mymaster 192.168.30.129 6379

22428:X 09 May 2019 15:52:31.307 # +sdown slave 192.168.30.128:6379 192.168.30.128 6379 @ mymaster 192.168.30.129 6379

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

从日志中可以看到，master已经从192.168.30.128转移到192.168.30.129上

192.168.30.129上查看集群信息

# /usr/local/bin/redis-cli -h 192.168.30.129 -p 6379 -a 123456

Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.

192.168.30.129:6379> info replication

# Replication

role:master

connected\_slaves:1

slave0:ip=192.168.30.130,port=6379,state=online,offset=291039,lag=1

master\_replid:757aff269236ed2707ba584a86a40716c1c76d74

master\_replid2:47a862fc0ff20362be29096ecdcca6d432070ee9

master\_repl\_offset:291182

second\_repl\_offset:248123

repl\_backlog\_active:1

repl\_backlog\_size:1048576

repl\_backlog\_first\_byte\_offset:1

repl\_backlog\_histlen:291182

192.168.30.129:6379> set key4 linux

OK

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

当前集群中只有一个slave——192.168.30.130，master是192.168.30.129，且192.168.30.129具有写权限。

192.168.30.130上查看redis的配置文件也可以看到replicaof 192.168.30.129 6379，这是sentinel在选举master是做的修改。

重新把192.168.30.128上进程启动

# systemctl start redis

# tail -f /usr/local/redis/sentinel.log

22428:X 09 May 2019 15:51:31.297 # +switch-master mymaster 192.168.30.128 6379 192.168.30.129 6379

22428:X 09 May 2019 15:51:31.298 \* +slave slave 192.168.30.130:6379 192.168.30.130 6379 @ mymaster 192.168.30.129 6379

22428:X 09 May 2019 15:51:31.298 \* +slave slave 192.168.30.128:6379 192.168.30.128 6379 @ mymaster 192.168.30.129 6379

22428:X 09 May 2019 15:52:31.307 # +sdown slave 192.168.30.128:6379 192.168.30.128 6379 @ mymaster 192.168.30.129 6379

22428:X 09 May 2019 16:01:24.872 # -sdown slave 192.168.30.128:6379 192.168.30.128 6379 @ mymaster 192.168.30.129 6379

1

2

3

4

5

6

7

8

9

查看集群信息

# /usr/local/bin/redis-cli -h 192.168.30.128 -p 6379 -a 123456

Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.

192.168.30.128:6379> info replication

# Replication

role:slave

master\_host:192.168.30.129

master\_port:6379

master\_link\_status:up

master\_last\_io\_seconds\_ago:0

master\_sync\_in\_progress:0

slave\_repl\_offset:514774

slave\_priority:100

slave\_read\_only:1

connected\_slaves:0

master\_replid:757aff269236ed2707ba584a86a40716c1c76d74

master\_replid2:0000000000000000000000000000000000000000

master\_repl\_offset:514774

second\_repl\_offset:-1

repl\_backlog\_active:1

repl\_backlog\_size:1048576

repl\_backlog\_first\_byte\_offset:376528

repl\_backlog\_histlen:138247

192.168.30.128:6379> get key4

"linux"

192.168.30.128:6379> set key5

(error) ERR wrong number of arguments for 'set' command

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

即使192.168.30.128重新启动redis服务，也是作为slave加入redis集群，192.168.30.129仍然是master。

Cluster模式

Cluster模式介绍

sentinel模式基本可以满足一般生产的需求，具备高可用性。但是当数据量过大到一台服务器存放不下的情况时，主从模式或sentinel模式就不能满足需求了，这个时候需要对存储的数据进行分片，将数据存储到多个Redis实例中。cluster模式的出现就是为了解决单机Redis容量有限的问题，将Redis的数据根据一定的规则分配到多台机器。

cluster可以说是sentinel和主从模式的结合体，通过cluster可以实现主从和master重选功能，所以如果配置两个副本三个分片的话，就需要六个Redis实例。因为Redis的数据是根据一定规则分配到cluster的不同机器的，当数据量过大时，可以新增机器进行扩容。

使用集群，只需要将redis配置文件中的cluster-enable配置打开即可。每个集群中至少需要三个主数据库才能正常运行，新增节点非常方便。

cluster集群特点：

\* 多个redis节点网络互联，数据共享

\* 所有的节点都是一主一从（也可以是一主多从），其中从不提供服务，仅作为备用

\* 不支持同时处理多个key（如MSET/MGET），因为redis需要把key均匀分布在各个节点上，

并发量很高的情况下同时创建key-value会降低性能并导致不可预测的行为

\* 支持在线增加、删除节点

\* 客户端可以连接任何一个主节点进行读写

1

2

3

4

5

6

7

8

9

10

Cluster模式搭建

环境准备：

三台机器，分别开启两个redis服务（端口）

192.168.30.128 端口：7001,7002

192.168.30.129 端口：7003,7004

192.168.30.130 端口：7005,7006

1

2

3

4

5

6

7

修改配置文件：

192.168.30.128

# mkdir /usr/local/redis/cluster

# cp /usr/local/redis/redis.conf /usr/local/redis/cluster/redis\_7001.conf

# cp /usr/local/redis/redis.conf /usr/local/redis/cluster/redis\_7002.conf

# chown -R redis:redis /usr/local/redis

# mkdir -p /data/redis/cluster/{redis\_7001,redis\_7002} && chown -R redis:redis /data/redis

1

2

3

4

5

6

7

8

9

# vim /usr/local/redis/cluster/redis\_7001.conf

bind 192.168.30.128

port 7001

daemonize yes

pidfile "/var/run/redis\_7001.pid"

logfile "/usr/local/redis/cluster/redis\_7001.log"

dir "/data/redis/cluster/redis\_7001"

#replicaof 192.168.30.129 6379

masterauth 123456

requirepass 123456

appendonly yes

cluster-enabled yes

cluster-config-file nodes\_7001.conf

cluster-node-timeout 15000

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

# vim /usr/local/redis/cluster/redis\_7002.conf

bind 192.168.30.128

port 7002

daemonize yes

pidfile "/var/run/redis\_7002.pid"

logfile "/usr/local/redis/cluster/redis\_7002.log"

dir "/data/redis/cluster/redis\_7002"

#replicaof 192.168.30.129 6379

masterauth "123456"

requirepass "123456"

appendonly yes

cluster-enabled yes

cluster-config-file nodes\_7002.conf

cluster-node-timeout 15000

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

其它两台机器配置与192.168.30.128一致，此处省略

启动redis服务：

# redis-server /usr/local/redis/cluster/redis\_7001.conf

# tail -f /usr/local/redis/cluster/redis\_7001.log

# redis-server /usr/local/redis/cluster/redis\_7002.conf

# tail -f /usr/local/redis/cluster/redis\_7002.log

1

2

3

4

5

6

7

其它两台机器启动与192.168.30.128一致，此处省略

安装ruby并创建集群（低版本）：

如果redis版本比较低，则需要安装ruby。任选一台机器安装ruby即可

# yum -y groupinstall "Development Tools"

# yum install -y gdbm-devel libdb4-devel libffi-devel libyaml libyaml-devel ncurses-devel openssl-devel readline-devel tcl-devel

# mkdir -p ~/rpmbuild/{BUILD,BUILDROOT,RPMS,SOURCES,SPECS,SRPMS}

# wget http://cache.ruby-lang.org/pub/ruby/2.2/ruby-2.2.3.tar.gz -P ~/rpmbuild/SOURCES

# wget http://raw.githubusercontent.com/tjinjin/automate-ruby-rpm/master/ruby22x.spec -P ~/rpmbuild/SPECS

# rpmbuild -bb ~/rpmbuild/SPECS/ruby22x.spec

# rpm -ivh ~/rpmbuild/RPMS/x86\_64/ruby-2.2.3-1.el7.x86\_64.rpm

# gem install redis #目的是安装这个，用于配置集群

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

# cp /usr/local/redis/src/redis-trib.rb /usr/bin/

# redis-trib.rb create --replicas 1 192.168.30.128:7001 192.168.30.128:7002 192.168.30.129:7003 192.168.30.129:7004 192.168.30.130:7005 192.168.30.130:7006

1

2

3

创建集群：

我这里是redis5.0.4，所以不需要安装ruby，直接创建集群即可

# redis-cli -a 123456 --cluster create 192.168.30.128:7001 192.168.30.128:7002 192.168.30.129:7003 192.168.30.129:7004 192.168.30.130:7005 192.168.30.130:7006 --cluster-replicas 1

Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.

>>> Performing hash slots allocation on 6 nodes...

Master[0] -> Slots 0 - 5460

Master[1] -> Slots 5461 - 10922

Master[2] -> Slots 10923 - 16383

Adding replica 192.168.30.129:7004 to 192.168.30.128:7001

Adding replica 192.168.30.130:7006 to 192.168.30.129:7003

Adding replica 192.168.30.128:7002 to 192.168.30.130:7005

M: 80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001

slots:[0-5460] (5461 slots) master

S: b4d3eb411a7355d4767c6c23b4df69fa183ef8bc 192.168.30.128:7002

replicates 6788453ee9a8d7f72b1d45a9093838efd0e501f1

M: 4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003

slots:[5461-10922] (5462 slots) master

S: b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004

replicates 80c80a3f3e33872c047a8328ad579b9bea001ad8

M: 6788453ee9a8d7f72b1d45a9093838efd0e501f1 192.168.30.130:7005

slots:[10923-16383] (5461 slots) master

S: 277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006

replicates 4d74ec66e898bf09006dac86d4928f9fad81f373

Can I set the above configuration? (type 'yes' to accept): yes #输入yes，接受上面配置

>>> Nodes configuration updated

>>> Assign a different config epoch to each node

>>> Sending CLUSTER MEET messages to join the cluster

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

可以看到，

192.168.30.128:7001是master，它的slave是192.168.30.129:7004；

192.168.30.129:7003是master，它的slave是192.168.30.130:7006；

192.168.30.130:7005是master，它的slave是192.168.30.128:7002

1

2

3

4

5

自动生成nodes.conf文件：

# ls /data/redis/cluster/redis\_7001/

appendonly.aof dump.rdb nodes-7001.conf

# vim /data/redis/cluster/redis\_7001/nodes-7001.conf

6788453ee9a8d7f72b1d45a9093838efd0e501f1 192.168.30.130:7005@17005 master - 0 1557454406312 5 connected 10923-16383

277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006@17006 slave 4d74ec66e898bf09006dac86d4928f9fad81f373 0 1557454407000 6 connected

b4d3eb411a7355d4767c6c23b4df69fa183ef8bc 192.168.30.128:7002@17002 slave 6788453ee9a8d7f72b1d45a9093838efd0e501f1 0 1557454408371 5 connected

80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001@17001 myself,master - 0 1557454406000 1 connected 0-5460

b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004@17004 slave 80c80a3f3e33872c047a8328ad579b9bea001ad8 0 1557454407366 4 connected

4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003@17003 master - 0 1557454407000 3 connected 5461-10922

vars currentEpoch 6 lastVoteEpoch 0

1

2

3

4

5

6

7

8

9

10

11

12

集群操作

登录集群：

# redis-cli -c -h 192.168.30.128 -p 7001 -a 123456 # -c，使用集群方式登录

1

查看集群信息：

192.168.30.128:7001> CLUSTER INFO #集群状态

cluster\_state:ok

cluster\_slots\_assigned:16384

cluster\_slots\_ok:16384

cluster\_slots\_pfail:0

cluster\_slots\_fail:0

cluster\_known\_nodes:6

cluster\_size:3

cluster\_current\_epoch:6

cluster\_my\_epoch:1

cluster\_stats\_messages\_ping\_sent:580

cluster\_stats\_messages\_pong\_sent:551

cluster\_stats\_messages\_sent:1131

cluster\_stats\_messages\_ping\_received:546

cluster\_stats\_messages\_pong\_received:580

cluster\_stats\_messages\_meet\_received:5

cluster\_stats\_messages\_received:1131

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

列出节点信息：

192.168.30.128:7001> CLUSTER NODES #列出节点信息

6788453ee9a8d7f72b1d45a9093838efd0e501f1 192.168.30.130:7005@17005 master - 0 1557455176000 5 connected 10923-16383

277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006@17006 slave 4d74ec66e898bf09006dac86d4928f9fad81f373 0 1557455174000 6 connected

b4d3eb411a7355d4767c6c23b4df69fa183ef8bc 192.168.30.128:7002@17002 slave 6788453ee9a8d7f72b1d45a9093838efd0e501f1 0 1557455175000 5 connected

80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001@17001 myself,master - 0 1557455175000 1 connected 0-5460

b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004@17004 slave 80c80a3f3e33872c047a8328ad579b9bea001ad8 0 1557455174989 4 connected

4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003@17003 master - 0 1557455175995 3 connected 5461-10922

1

2

3

4

5

6

7

8

这里与nodes.conf文件内容相同

写入数据：

192.168.30.128:7001> set key111 aaa

-> Redirected to slot [13680] located at 192.168.30.130:7005 #说明数据到了192.168.30.130:7005上

OK

192.168.30.130:7005> set key222 bbb

-> Redirected to slot [2320] located at 192.168.30.128:7001 #说明数据到了192.168.30.128:7001上

OK

192.168.30.128:7001> set key333 ccc

-> Redirected to slot [7472] located at 192.168.30.129:7003 #说明数据到了192.168.30.129:7003上

OK

192.168.30.129:7003> get key111

-> Redirected to slot [13680] located at 192.168.30.130:7005

"aaa"

192.168.30.130:7005> get key333

-> Redirected to slot [7472] located at 192.168.30.129:7003

"ccc"

192.168.30.129:7003>

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

可以看出redis cluster集群是去中心化的，每个节点都是平等的，连接哪个节点都可以获取和设置数据。

当然，平等指的是master节点，因为slave节点根本不提供服务，只是作为对应master节点的一个备份。

增加节点：

192.168.30.129上增加一节点：

# cp /usr/local/redis/cluster/redis\_7003.conf /usr/local/redis/cluster/redis\_7007.conf

# vim /usr/local/redis/cluster/redis\_7007.conf

bind 192.168.30.129

port 7007

daemonize yes

pidfile "/var/run/redis\_7007.pid"

logfile "/usr/local/redis/cluster/redis\_7007.log"

dir "/data/redis/cluster/redis\_7007"

#replicaof 192.168.30.129 6379

masterauth "123456"

requirepass "123456"

appendonly yes

cluster-enabled yes

cluster-config-file nodes\_7007.conf

cluster-node-timeout 15000

# mkdir /data/redis/cluster/redis\_7007

# chown -R redis:redis /usr/local/redis && chown -R redis:redis /data/redis

# redis-server /usr/local/redis/cluster/redis\_7007.conf

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

192.168.30.130上增加一节点：

# cp /usr/local/redis/cluster/redis\_7005.conf /usr/local/redis/cluster/redis\_7008.conf

# vim /usr/local/redis/cluster/redis\_7007.conf

bind 192.168.30.130

port 7008

daemonize yes

pidfile "/var/run/redis\_7008.pid"

logfile "/usr/local/redis/cluster/redis\_7008.log"

dir "/data/redis/cluster/redis\_7008"

#replicaof 192.168.30.130 6379

masterauth "123456"

requirepass "123456"

appendonly yes

cluster-enabled yes

cluster-config-file nodes\_7008.conf

cluster-node-timeout 15000

# mkdir /data/redis/cluster/redis\_7008

# chown -R redis:redis /usr/local/redis && chown -R redis:redis /data/redis

# redis-server /usr/local/redis/cluster/redis\_7008.conf

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

集群中增加节点：

192.168.30.129:7003> CLUSTER MEET 192.168.30.129 7007

OK

192.168.30.129:7003> CLUSTER NODES

4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003@17003 myself,master - 0 1557457361000 3 connected 5461-10922

80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001@17001 master - 0 1557457364746 1 connected 0-5460

277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006@17006 slave 4d74ec66e898bf09006dac86d4928f9fad81f373 0 1557457362000 6 connected

b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004@17004 slave 80c80a3f3e33872c047a8328ad579b9bea001ad8 0 1557457363000 4 connected

b4d3eb411a7355d4767c6c23b4df69fa183ef8bc 192.168.30.128:7002@17002 slave 6788453ee9a8d7f72b1d45a9093838efd0e501f1 0 1557457362000 5 connected

e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 192.168.30.129:7007@17007 master - 0 1557457362729 0 connected

6788453ee9a8d7f72b1d45a9093838efd0e501f1 192.168.30.130:7005@17005 master - 0 1557457363739 5 connected 10923-16383

1

2

3

4

5

6

7

8

9

10

11

12

192.168.30.129:7003> CLUSTER MEET 192.168.30.130 7008

OK

192.168.30.129:7003> CLUSTER NODES

4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003@17003 myself,master - 0 1557457489000 3 connected 5461-10922

80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001@17001 master - 0 1557457489000 1 connected 0-5460

277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006@17006 slave 4d74ec66e898bf09006dac86d4928f9fad81f373 0 1557457489000 6 connected

b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004@17004 slave 80c80a3f3e33872c047a8328ad579b9bea001ad8 0 1557457488000 4 connected

b4d3eb411a7355d4767c6c23b4df69fa183ef8bc 192.168.30.128:7002@17002 slave 6788453ee9a8d7f72b1d45a9093838efd0e501f1 0 1557457489472 5 connected

1a1c7f02fce87530bd5abdfc98df1cffce4f1767 192.168.30.130:7008@17008 master - 0 1557457489259 0 connected

e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 192.168.30.129:7007@17007 master - 0 1557457489000 0 connected

6788453ee9a8d7f72b1d45a9093838efd0e501f1 192.168.30.130:7005@17005 master - 0 1557457490475 5 connected 10923-16383

1

2

3

4

5

6

7

8

9

10

11

12

13

可以看到，新增的节点都是以master身份加入集群的

更换节点身份：

将新增的192.168.30.130:7008节点身份改为192.168.30.129:7007的slave

# redis-cli -c -h 192.168.30.130 -p 7008 -a 123456 cluster replicate e51ab166bc0f33026887bcf8eba0dff3d5b0bf14

1

cluster replicate后面跟node\_id，更改对应节点身份。也可以登入集群更改

# redis-cli -c -h 192.168.30.130 -p 7008 -a 123456

192.168.30.130:7008> CLUSTER REPLICATE e51ab166bc0f33026887bcf8eba0dff3d5b0bf14

OK

192.168.30.130:7008> CLUSTER NODES

277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006@17006 slave 4d74ec66e898bf09006dac86d4928f9fad81f373 0 1557458316881 3 connected

80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001@17001 master - 0 1557458314864 1 connected 0-5460

4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003@17003 master - 0 1557458316000 3 connected 5461-10922

6788453ee9a8d7f72b1d45a9093838efd0e501f1 192.168.30.130:7005@17005 master - 0 1557458315872 5 connected 10923-16383

b4d3eb411a7355d4767c6c23b4df69fa183ef8bc 192.168.30.128:7002@17002 slave 6788453ee9a8d7f72b1d45a9093838efd0e501f1 0 1557458317890 5 connected

1a1c7f02fce87530bd5abdfc98df1cffce4f1767 192.168.30.130:7008@17008 myself,slave e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 0 1557458315000 7 connected

b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004@17004 slave 80c80a3f3e33872c047a8328ad579b9bea001ad8 0 1557458315000 1 connected

e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 192.168.30.129:7007@17007 master - 0 1557458314000 0 connected

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

查看相应的nodes.conf文件，可以发现有更改，它记录当前集群的节点信息

# cat /data/redis/cluster/redis\_7001/nodes-7001.conf

1a1c7f02fce87530bd5abdfc98df1cffce4f1767 192.168.30.130:7008@17008 slave e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 0 1557458236169 7 connected

6788453ee9a8d7f72b1d45a9093838efd0e501f1 192.168.30.130:7005@17005 master - 0 1557458235000 5 connected 10923-16383

277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006@17006 slave 4d74ec66e898bf09006dac86d4928f9fad81f373 0 1557458234103 6 connected

b4d3eb411a7355d4767c6c23b4df69fa183ef8bc 192.168.30.128:7002@17002 slave 6788453ee9a8d7f72b1d45a9093838efd0e501f1 0 1557458235129 5 connected

80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001@17001 myself,master - 0 1557458234000 1 connected 0-5460

b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004@17004 slave 80c80a3f3e33872c047a8328ad579b9bea001ad8 0 1557458236000 4 connected

e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 192.168.30.129:7007@17007 master - 0 1557458236000 0 connected

4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003@17003 master - 0 1557458233089 3 connected 5461-10922

vars currentEpoch 7 lastVoteEpoch 0

1

2

3

4

5

6

7

8

9

10

11

删除节点：

192.168.30.130:7008> CLUSTER FORGET 1a1c7f02fce87530bd5abdfc98df1cffce4f1767

(error) ERR I tried hard but I can't forget myself... #无法删除登录节点

192.168.30.130:7008> CLUSTER FORGET e51ab166bc0f33026887bcf8eba0dff3d5b0bf14

(error) ERR Can't forget my master! #不能删除自己的master节点

192.168.30.130:7008> CLUSTER FORGET 6788453ee9a8d7f72b1d45a9093838efd0e501f1

OK #可以删除其它的master节点

192.168.30.130:7008> CLUSTER NODES

277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006@17006 slave 4d74ec66e898bf09006dac86d4928f9fad81f373 0 1557458887328 3 connected

80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001@17001 master - 0 1557458887000 1 connected 0-5460

4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003@17003 master - 0 1557458886000 3 connected 5461-10922

b4d3eb411a7355d4767c6c23b4df69fa183ef8bc 192.168.30.128:7002@17002 slave - 0 1557458888351 5 connected

1a1c7f02fce87530bd5abdfc98df1cffce4f1767 192.168.30.130:7008@17008 myself,slave e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 0 1557458885000 7 connected

b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004@17004 slave 80c80a3f3e33872c047a8328ad579b9bea001ad8 0 1557458883289 1 connected

e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 192.168.30.129:7007@17007 master - 0 1557458885310 0 connected

192.168.30.130:7008> CLUSTER FORGET b4d3eb411a7355d4767c6c23b4df69fa183ef8bc

OK #可以删除其它的slave节点

192.168.30.130:7008> CLUSTER NODES

277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006@17006 slave 4d74ec66e898bf09006dac86d4928f9fad81f373 0 1557459031397 3 connected

80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001@17001 master - 0 1557459032407 1 connected 0-5460

4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003@17003 master - 0 1557459035434 3 connected 5461-10922

6788453ee9a8d7f72b1d45a9093838efd0e501f1 192.168.30.130:7005@17005 master - 0 1557459034000 5 connected 10923-16383

1a1c7f02fce87530bd5abdfc98df1cffce4f1767 192.168.30.130:7008@17008 myself,slave e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 0 1557459032000 7 connected

b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004@17004 slave 80c80a3f3e33872c047a8328ad579b9bea001ad8 0 1557459034000 1 connected

e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 192.168.30.129:7007@17007 master - 0 1557459034427 0 connected

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

保存配置：

192.168.30.130:7008> CLUSTER SAVECONFIG #将节点配置信息保存到硬盘

OK

# cat /data/redis/cluster/redis\_7001/nodes-7001.conf

1a1c7f02fce87530bd5abdfc98df1cffce4f1767 192.168.30.130:7008@17008 slave e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 0 1557458236169 7 connected

6788453ee9a8d7f72b1d45a9093838efd0e501f1 192.168.30.130:7005@17005 master - 0 1557458235000 5 connected 10923-16383

277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006@17006 slave 4d74ec66e898bf09006dac86d4928f9fad81f373 0 1557458234103 6 connected

b4d3eb411a7355d4767c6c23b4df69fa183ef8bc 192.168.30.128:7002@17002 slave 6788453ee9a8d7f72b1d45a9093838efd0e501f1 0 1557458235129 5 connected

80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001@17001 myself,master - 0 1557458234000 1 connected 0-5460

b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004@17004 slave 80c80a3f3e33872c047a8328ad579b9bea001ad8 0 1557458236000 4 connected

e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 192.168.30.129:7007@17007 master - 0 1557458236000 0 connected

4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003@17003 master - 0 1557458233089 3 connected 5461-10922

vars currentEpoch 7 lastVoteEpoch 0

# redis-cli -c -h 192.168.30.130 -p 7008 -a 123456

Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.

192.168.30.130:7008> CLUSTER NODES

277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006@17006 slave 4d74ec66e898bf09006dac86d4928f9fad81f373 0 1557459500741 3 connected

80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001@17001 master - 0 1557459500000 1 connected 0-5460

4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003@17003 master - 0 1557459501000 3 connected 5461-10922

6788453ee9a8d7f72b1d45a9093838efd0e501f1 192.168.30.130:7005@17005 master - 0 1557459500000 5 connected 10923-16383

b4d3eb411a7355d4767c6c23b4df69fa183ef8bc 192.168.30.128:7002@17002 slave 6788453ee9a8d7f72b1d45a9093838efd0e501f1 0 1557459499737 5 connected

1a1c7f02fce87530bd5abdfc98df1cffce4f1767 192.168.30.130:7008@17008 myself,slave e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 0 1557459499000 7 connected

b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004@17004 slave 80c80a3f3e33872c047a8328ad579b9bea001ad8 0 1557459501750 1 connected

e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 192.168.30.129:7007@17007 master - 0 1557459498000 0 connected

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

可以看到，之前删除的节点又恢复了，这是因为对应的配置文件没有删除，执行CLUSTER SAVECONFIG恢复。

模拟master节点挂掉：

192.168.30.128

# netstat -lntp |grep 7001

tcp 0 0 192.168.30.128:17001 0.0.0.0:\* LISTEN 6701/redis-server 1

tcp 0 0 192.168.30.128:7001 0.0.0.0:\* LISTEN 6701/redis-server 1

# kill 6701

1

2

3

4

5

192.168.30.130:7008> CLUSTER NODES

277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006@17006 slave 4d74ec66e898bf09006dac86d4928f9fad81f373 0 1557461178000 3 connected

80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001@17001 master,fail - 1557460950483 1557460947145 1 disconnected

4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003@17003 master - 0 1557461174922 3 connected 5461-10922

6788453ee9a8d7f72b1d45a9093838efd0e501f1 192.168.30.130:7005@17005 master - 0 1557461181003 5 connected 10923-16383

b4d3eb411a7355d4767c6c23b4df69fa183ef8bc 192.168.30.128:7002@17002 slave 6788453ee9a8d7f72b1d45a9093838efd0e501f1 0 1557461179993 5 connected

1a1c7f02fce87530bd5abdfc98df1cffce4f1767 192.168.30.130:7008@17008 myself,slave e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 0 1557461176000 7 connected

b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004@17004 master - 0 1557461178981 8 connected 0-5460

e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 192.168.30.129:7007@17007 master - 0 1557461179000 0 connected

1

2

3

4

5

6

7

8

9

10

对应7001的一行可以看到，master fail，状态为disconnected；而对应7004的一行，slave已经变成master。

重新启动7001节点：

# redis-server /usr/local/redis/cluster/redis\_7001.conf

192.168.30.130:7008> CLUSTER NODES

277daeb8660d5273b7c3e05c263f861ed5f17b92 192.168.30.130:7006@17006 slave 4d74ec66e898bf09006dac86d4928f9fad81f373 0 1557461307000 3 connected

80c80a3f3e33872c047a8328ad579b9bea001ad8 192.168.30.128:7001@17001 slave b6331cbc986794237c83ed2d5c30777c1551546e 0 1557461305441 8 connected

4d74ec66e898bf09006dac86d4928f9fad81f373 192.168.30.129:7003@17003 master - 0 1557461307962 3 connected 5461-10922

6788453ee9a8d7f72b1d45a9093838efd0e501f1 192.168.30.130:7005@17005 master - 0 1557461304935 5 connected 10923-16383

b4d3eb411a7355d4767c6c23b4df69fa183ef8bc 192.168.30.128:7002@17002 slave 6788453ee9a8d7f72b1d45a9093838efd0e501f1 0 1557461306000 5 connected

1a1c7f02fce87530bd5abdfc98df1cffce4f1767 192.168.30.130:7008@17008 myself,slave e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 0 1557461305000 7 connected

b6331cbc986794237c83ed2d5c30777c1551546e 192.168.30.129:7004@17004 master - 0 1557461308972 8 connected 0-5460

e51ab166bc0f33026887bcf8eba0dff3d5b0bf14 192.168.30.129:7007@17007 master - 0 1557461307000 0 connected

1

2

3

4

5

6

7

8

9

10

11

12

可以看到，7001节点启动后为slave节点，并且是7004的slave节点。即master节点如果挂掉，它的slave节点变为新master节点继续对外提供服务，而原来的master节点如果重启，则变为新master节点的slave节点。

另外，如果这里是拿7007节点做测试的话，会发现7008节点并不会切换，这是因为7007节点上根本没数据。集群数据被分为三份，采用哈希槽 (hash slot)的方式来分配16384个slot的话，它们三个节点分别承担的slot 区间是：

————————————————

版权声明：本文为CSDN博主「最爱喝酸奶」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/miss1181248983/article/details/90056960