Redis部署

拓扑

主机名	ip	角色
redis-vip	192.168.140	
redis-cluster1	192.168.1.41	主6379 从6380
redis-cluster2	192.168.1.42	主6379 从6380
redis-cluster3	192.168.1.43	主6379 从6380

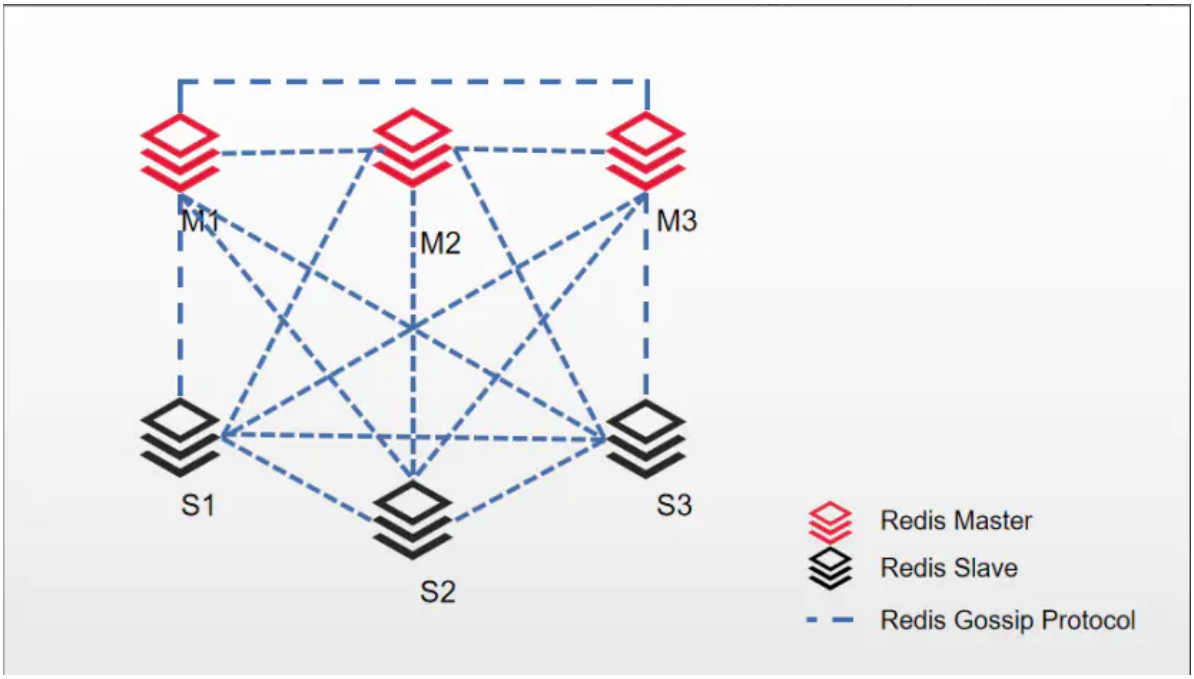
Redis三主三从集群

Redis-Cluster采用无中心结构，每个节点保存数据和整个集群状态,每个节点都和其他所有节点连接。

一组Redis Cluster是由多个Redis实例组成，官方推荐使用6实例，其中3个为主节点，3个为从节点。一旦有主节点发生故障的时候，Redis Cluster可以选举出对应的从节点成为新的主节点，继续对外服务，从而保证服务的高可用性。

注意：当集群内一个***master***以及其对应的***slave***同时宕机，集群将无法提供服务

注意：当存活的主节点数小于总节点数的一半时，整个集群就无法提供服务了。



部署环境：

IP地址	端口	角色	Redis版本
192.168.1.41	7000	redis-master	5.0.8
192.168.1.41	7001	redis-slave	5.0.8
192.168.1.42	7000	redis-master	5.0.8
192.168.1.42	7001	redis-slave	5.0.8
192.168.1.43	7000	redis-master	5.0.8
192.168.1.43	7001	redis-slave	5.0.8

1.安装Redis

1、安装C/C++环境

Redis编译时需要使用C/C++环境：

```
yum install -y gcc gcc-c++ make
```

2、下载Redis安装包

下载地址：http://download.redis.io/releases/

```
mkdir /data
cd /data
wget [http://download.redis.io/releases/redis-5.0.8.tar.gz]
(https://links.jianshu.com/go?
to=http%3A%2F%2Fdownload.redis.io%2Freleases%2Fredis-
5.0.8.tar.gz)
```

3、解压、编译

```
tar -zxvf redis-5.0.8.tar.gz
cd redis-5.0.8
make
```

4、创建Redis相关工作目录(目录可自定义)

```
mkdir /data/redis-cluster/{data/{redis_7000,redis_7001},conf,log}
-p
```

5、复制redis配置文件

```
cp redis.conf /data/redis-cluster/conf/redis_7000.conf
cp redis.conf /data/redis-cluster/conf/redis_7001.conf
```

2.修改Redis配置文件

1、修改redis-master配置文件

vi /data/redis-cluster/conf/redis_7000.conf

```
port 7000                  #修改redis监听端口(可以自定义)
bind 0.0.0.0               #表示redis允许所有地址连接。默认127.0.0.1，仅
允许本地连接。
daemonize yes              #允许redis后台运行
pidfile /var/run/redis_7000.pid          #pid存放目录
logfile "/var/log/redis-sentinel.log"    #设置Sentinel日志存放路径
dir /data/redis-cluster/data/redis_7000 #工作目录
cluster-enabled yes        #是否开启集群
cluster-config-file /data/redis-cluster/conf/nodes_7000.conf
```

```
#集群配置文件的名称，每个节点都有一个集群相关的配置文件，持久化保存集群的信息
#这个文件并不需要手动配置，这个配置文件有Redis生成并更新，
cluster-node-timeout 15000
#节点互连超时的阀值。集群节点超时毫秒数，默认15秒
appendonly yes
#Redis会把每次写入的数据在接收后都写入 appendonly.aof 文件，
#每次启动时Redis都会先把这个文件的数据读入内存里，先忽略RDB文件。
requirepass 123456     #设置redis密码
masterauth 123456      #主从同步master的密码(如果没有设置redis密码，则无需配置)
```

2、修改redis-slave配置文件

vi /data/redis-cluster/conf/redis_7001.conf #修改redis-slave配置文件，具体如下：

```
port 7001
bind 0.0.0.0
daemonize yes
pidfile /var/run/redis-cluster/redis_7001.pid
logfile "/data/redis-cluster/log/redis_7001.log"
dir /data/redis-cluster/data/redis_7001
cluster-enabled yes
cluster-config-file /data/redis-cluster/conf/nodes_7001.conf
cluster-node-timeout 15000
appendonly yes
requirepass 123456
masterauth 123456
```

3.分发到集群其他服务器

复制redis以及工作目录到其他服务器

```
scp -r redis-5.0.8 redis-cluster root@10.10.41.112:/data
scp -r redis-5.0.8 redis-cluster root@10.10.41.113:/data
```

设置软链接，方便启动redis服务

```
ln -s /data/redis-5.0.8/src/redis-server /usr/bin/redis-server
ln -s /data/redis-5.0.8/src/redis-cli /usr/bin/redis-cli
```

4.启动Redis

集群内每台服务器分别启动两个redis

```
redis-server /data/redis-cluster/conf/redis_7000.conf
redis-server /data/redis-cluster/conf/redis_7001.conf
```

```
# 验证是否启动成功
ps -ef | grep redis

# 查看版本
redis-cli --version
redis-server --version
```

5.创建Redis Cluster

注意：redis5.0以上集群创建方式改为了C编写的redis-cli创建，不用再安装麻烦的ruby。

创建集群，--cluster-replicas 1指定从库数量1，创建顺序三主-三从。即主-主-主-从-从-从。

如果redis设置了密码，则创建集群时需要添加密码信息 -a 密码：

```
redis-cli -a 123456 --cluster create 192.168.1.41:7000
192.168.1.42:7000 192.168.1.43:7000 192.168.1.41:7001
192.168.1.42:7001 192.168.1.43:7001 --cluster-replicas 1
```

确认集群配置信息，确认无误则输入 yes 并按回车

```
[root@redis-cluster1 data]# redis-cli -a 123456 --cluster create
192.168.1.41:7000 192.168.1.42:7000 192.168.1.43:7000
192.168.1.41:7001 192.168.1.42:7001 192.168.1.43:7001 --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 192.168.1.42:7001 to 192.168.1.41:7000
Adding replica 192.168.1.43:7001 to 192.168.1.42:7000
Adding replica 192.168.1.41:7001 to 192.168.1.43:7000
M: 73931524b237c2f5918cacb9f2688b2654fabe92 192.168.1.41:7000
   slots:[0-5460] (5461 slots) master
M: 6076ab2b7f9390c1e77f01990a5b9b3714c12587 192.168.1.42:7000
   slots:[5461-10922] (5462 slots) master
M: e74f106c5a7f7901f9f061931450101641c07680 192.168.1.43:7000
   slots:[10923-16383] (5461 slots) master
S: ca6d4f79e8b6b2bb372a9f735b62a9c4c21d276e 192.168.1.41:7001
   replicates e74f106c5a7f7901f9f061931450101641c07680
S: 8c3c40fb07d2f1194502d1ad69d3233b8c389847 192.168.1.42:7001
   replicates 73931524b237c2f5918cacb9f2688b2654fabe92
S: ffc56fb9b57b0759e6d5e203b352000588e56c11 192.168.1.43:7001
   replicates 6076ab2b7f9390c1e77f01990a5b9b3714c12587
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 192.168.1.41:7000)
M: 73931524b237c2f5918cacb9f2688b2654fabe92 192.168.1.41:7000
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
M: 6076ab2b7f9390c1e77f01990a5b9b3714c12587 192.168.1.42:7000
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
S: 8c3c40fb07d2f1194502d1ad69d3233b8c389847 192.168.1.42:7001
   slots: (0 slots) slave
   replicates 73931524b237c2f5918cacb9f2688b2654fabe92
S: ca6d4f79e8b6b2bb372a9f735b62a9c4c21d276e 192.168.1.41:7001
   slots: (0 slots) slave
   replicates e74f106c5a7f7901f9f061931450101641c07680
S: ffc56fb9b57b0759e6d5e203b352000588e56c11 192.168.1.43:7001
   slots: (0 slots) slave
   replicates 6076ab2b7f9390c1e77f01990a5b9b3714c12587
M: e74f106c5a7f7901f9f061931450101641c07680 192.168.1.43:7000
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
```

```
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

出现以上信息，表示集群配置成功。

6验证集群Redis-Cluster

1、登录redis集群

-a Redis密码,-c表示集群模式 ,指定IP和端口

注意：可能会redirected进入到其它server

2、验证集群信息

```
#查看集群信息
redis-cluster1:7000> cluster info

#查看集群节点列表
redis-cluster1:7000> cluster nodes
```

查看集群内主从关系：

```
redis-cli -a 123456 -h 192.168.1.41 -p 7000 -c cluster slots |
xargs  -n8 | awk '{print $3":"$4"->"$6":"$7}' | sort -nk2 -t ':'
| uniq

# 输出如下
192.168.1.41:7000->192.168.1.42:7001
192.168.1.42:7000->192.168.1.43:7001
192.168.1.43:7000->192.168.1.41:7001
```

3、进行数据验证操作

插入数据：

```
redis-cluster1:7000> set mykey "Hello Redis"
redis-cluster2:7000> get mykey
```

登录其他节点查看数据：

```
redis-cli -a 123456 -h redis-cluster3 -p 7000 -c
redis-cluster3:7000> get mykey
```

4、验证集群故障转移

将其中一个redis-master停止掉后、其对应的slave节点会被选举为master节点，旧master节点重新恢复时，其角色会成为slave。具体可自行验证，验证时可开启日志查看相关信息。

keepalived高可用

安装keepalived

```
# 安装相关 keepalived 依赖
yum -y install kernel-devel openssl-devel popt-devel gcc*
# 解压源码包
tar xf keepalived-2.1.2.tar.gz
cd keepalived-2.1.2
# 编译安装
./configure --prefix=/ && make && make install
# 设置 Keepalived 开机自启
systemctl enable keepalived
```

配置文件

1. redis-cluster1配置文件

[root@redis-cluster1 ~]# vim /etc/keepalived/keepalived.conf

```
! Configuration File for keepalived

global_defs {
   router_id redis-cluster1
}
vrrp_script chk_redis {
        script "/etc/keepalived/redis_check.sh"
        interval 2
        weight -20
```

```
}
vrrp_instance VI_1 {
    state MASTER         # 标识为主服务
    interface ens33      #绑定虚拟机的IP
    virtual_router_id 51# 虚拟路由id，和从机保持一致
    priority 100         #权重，需要高于从机
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    track_script {
        chk_redis ## 执行 redis 监控的服务
    }
    virtual_ipaddress {
        192.168.1.140
    }
}
```

2. redis-cluster2配置文件

[root@redis-cluster2 ~]# vim /etc/keepalived/keepalived.conf

```
! Configuration File for keepalived

global_defs {
   router_id redis-cluster2
}
vrrp_script chk_redis {
        script "/etc/keepalived/redis_check.sh"
        interval 2
        weight -20
}

vrrp_instance VI_1 {
    state BACKUP
    interface ens33
    virtual_router_id 51
    priority 90
    advert_int 1
    authentication {
        auth_type PASS
```

```
        auth_pass 1111
    }
        track_script {
            chk_redis ## 执行 redis 监控的服务
        }
    virtual_ipaddress {
        192.168.1.140
    }
}
```

3. redis-cluster3配置文件

[root@redis-cluster3 ~]# vim /etc/keepalived/keepalived.conf

```
! Configuration File for keepalived

global_defs {
   router_id redis-cluster3
}
vrrp_script chk_redis {
        script "/etc/keepalived/redis_check.sh"
        interval 2
        weight -20
}

vrrp_instance VI_1 {
    state BACKUP
    interface ens33
    virtual_router_id 51
    priority 60
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
        track_script {
            chk_redis ## 执行 redis 监控的服务
        }
    virtual_ipaddress {
        192.168.1.140
    }
}
```

4. 编写监测心跳脚本

```
# 编写检查脚本，两台都需要
vim /etc/keepalived/redis_check.sh

#!/bin/bash
# ps -ef | grep 7000 | grep -v grep | wc -l
counter=$(ps -ef | grep 7000 | grep -v grep | wc -l)
if [ "${counter}" = "0" ]; then
    redis-server /data/redis-cluster/conf/redis_7000.conf
    sleep 2
    counter=$(ps -ef | grep 7000 | grep -v grep | wc -l)
    if [ "${counter}" = "0" ]; then
        /etc/init.d/keepalived stop
    fi
fi

# 添加执行权限
chmod +x /etc/keepalived/redis_check.sh
```

启动keepalived

```
systemctl start keepalived
```