Redis sentinel哨兵集群

1. **Redis主从复制中的问题**

**Redis主从复制模式可以将主节点的数据改变同步给从节点，这样从节点就可以起到两个作用：第一，作为主节点的一个备份，一旦主节点出了故障不能继续对外提供服务，从节点可以作为后备“顶”上来，并且保证数据尽量不丢失。第二，从节点可以扩主节点的读能力，通过实现读写分离结构，可以大大减轻主节点在进行高并发读写操作时的访问压力。**

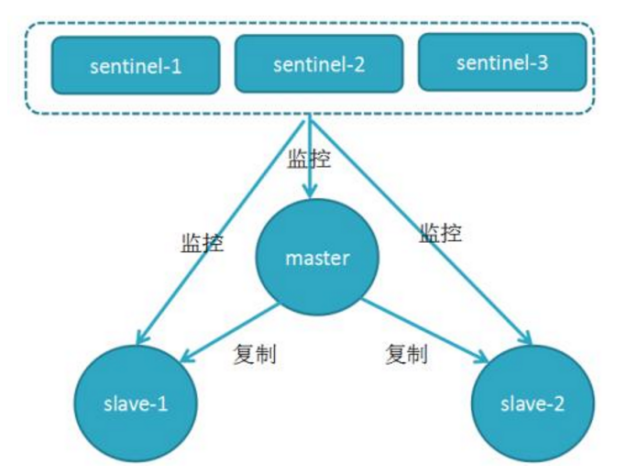
**但是主从同步也带了一些问题：**

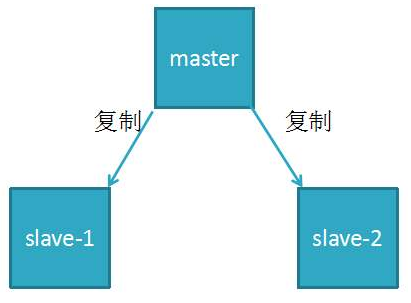
1. **一旦主节点故障，需要手动将一个从节点晋升为主节点**
2. **需要修改客户端的主节点地址**
3. **如果是一主多从结构，还需将其他从节点去复制新的主节点**

**而以上整个过程都需要人工干预。**

1. **Redis sentinel的高可用性**

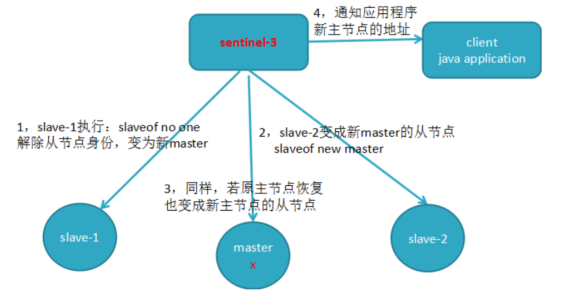
**Redis Sentinel是Redis的高可用实现方案，在实际的生产环境中，对提高整个系统的高可用性是非常有帮助的。哨兵是一个分布式架构，其中包含若干个Sentinel节点和Redis数据节点，每个Sentinel节点都会对数据节点和其他Sentinel节点进行监控，当他发现节点不可达时，会对节点做下线标识。如果被标识的是主节点，他还会和其他Sentinel节点进行“协商”，当大多数Sentinel节点都认为主节点不可达时，它们会选举出一个Sentinel节点来完成自动故障转移的工作，同时会将这个变化实时通知给Redis的应用方。整个过程是完全自动的，不需要人工来介入，所以这套方案很有效的解决了Redis的高可用问题。**



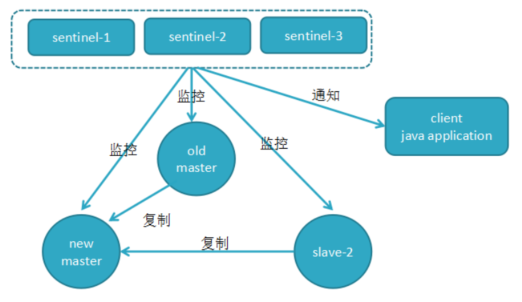


**整个故障转移的处理逻辑基本上可分为4步：**

1. **主节点出现故障，此时两个从节点与主节点失去连接，主从复制失败**
2. **每个Sentinel节点通过定期监控发现主节点出现了故障**
3. **多个Sentinel节点对主节点的故障达成一致，选举出其中一个Sentinel节点作为领导者负责故障转移。**
4. **Sentinel领导者节点执行了故障转移，如下图所示**



1. 故障转移后整个Redis Sentinel的拓扑关系如下所示



1. **实现原理**

**1、三个定时监控任务**

**Redis Sentinel通过三个定时监控任务完成对各个节点发现和监控：**

1. **每隔10秒，每个Sentinel节点会向主节点和从节点发送info命令获取最新的拓扑结构，这个定时任务的作用具体可以表现在三个方面：**

* **通过向主节点执行info命令，获取从节点的信息，这也是为什么Sentinel节点不需要显示配置监控从节点**
* **当有新的从节点加入时都可以立刻感知出来**
* **节点不可达或者故障转移后，可以通过info命令实时更新节点拓扑信息**

1. **每隔2秒，每个Sentinel节点会向Redis数据节点的\_\_sentinel\_\_:hello频道上发送该Sentinel节点对于主节点的判断以及当前Sentinel节点的信息，同时每个Sentinel节点也会订阅该频道，来了解其他Sentinel节点以及它们对主节点的判断，所以这个定时任务可以完成以下两个工作：**

* **发现新的Sentinel节点：通过订阅主节点的\_\_sentinel\_\_:hello了解其他的Sentinel节点信息，如果是新加入的Sentinel节点，将该Sentinel节点信息保存起来，并与该Sentinel节点创建连接。**
* **Sentinel节点之间交换主节点的状态，作为后面客观下线以及领导者选举的依据。**

1. **每隔1秒，每个Sentinel节点会向主节点、从节点、其余Sentinel节点发送一条ping命令做一次心跳检测，来确认这些节点当前是否可达。**
2. **主观下线**

**每个Sentinel节点会每隔1秒对主节点、从节点、其他Sentinel节点发送ping命令做心跳检测，当这些节点超过down-after-milliseconds没有进行有效回复，Sentinel节点就会对该节点做失败判定，这个行为叫做主观下线。从字面意思也可以很容易看出主观下线是当前Sentinel节点的一家之言，存在误判断的可能。**

1. **客观下线**

**当Sentinel主观下线的节点是主节点时，该Sentinel节点会通过Sentinel is-master-down-by-addr命令向其他Sentinel节点询问对主节点的判断，当超过<quorum>个数，Sentinel节点认为主节点确实有问题，这时该Sentinel节点会做出客观下线的决定，这样客观下线的含义是比较明显了，也就是大部分Sentinel节点都对主节点的下线做了同意的判定，那么这个判定就是客观的。**

1. **领导者选举**

**故障转移的工作只需要一个Sentinel节点来完成即可，所以Sentinel节点之间会做一个领导者选举的工作，选出一个Sentinel节点作为领导者进行故障转移的工作。Redis使用了Raft算法实现领导者选举，大致思路如下：**

1. **每个在线的Sentinel节点都有资格成为领导者，当他确认主节点主观下线时，会向其他Sentinel节点发送sentinel is-master-down-by-addr命令，要求将自己设置为领导者。**
2. **收到命令的节点，如果没有同意过其他节点的请求，则会同意该请求，否则拒绝。**
3. **如果某一个节点的票数已经大于等于max（quorum，num（sentinels）/2+1），那么它将成为领导者**
4. **如果此过程没有选举出领导者，将进入下一次选举**
5. **故障转移**

**领导者选举出的Sentinel节点负责故障转移，具体步骤如下：**

1. **在从节点列表中选出一个节点作为新的主节点，方法如下：**

* **过滤：“不健康”（主观下线、断线）、5秒内没有回复过Sentinel节点ping响应，与主节点失联超过down-after-milliseconds\*10秒。**
* **选择slave-priority（从优先级）最高的从节点列表，如果存在则返回，不存在则继续。**
* **选择复制偏移量最大的从节点**
* **选择runid最小的从节点**

1. **领导者节点会对第一步选出来的从节点执行slaveof no one命令，使其成为主**
2. **领导者节点会向剩余的从节点发送命令，让他们成为新主的从节点**
3. **Sentinel节点集合会将原来的主节点更新为从节点并保持这对其关注，当其恢复后命令它去复制新的主节点。**
4. **安装与部署**

**（一）：配置redis主从复制**

**按照主从复制的配置完成，在主从复制基础上继续完成以下步骤**

* **配置master角色的文件**

因为master有可能down机导致redis服务中断，重新加入群集后将作为从服务器来工作，所以master上也需要配置materauth密码，这样才能与新master验证通过建立新的连接

[root@master ~]# sed -i '288s/^\.\*/masterauth 123456\n/g' /etc/redis/6379.conf

* **配置slave2角色的文件**

**添加一台从服务器，因为群集最少两台从服务器，master down后,一台从服务器提升为主，slave只剩一台,群集不能工作。可以把slave1的6379配置复制一份到slave2后更改下ip为192.168.200.113即可**

**slave2上安装redis，端口号默认6379**

[root@slave2 ~]# scp 192.168.200.112:/etc/redis/6379.conf /etc/redis

[root@slave2 ~]#sed -i '/^bind/s/192.168.200.112/192.168.200.113/g' /etc/redis/6379.conf

[root@slave2 ~]#/etc/init.d/redis\_6379 restart

[root@slave2 ~]# redis-cli -p 6379 -a 123456 -h 192.168.200.111 info replication

# Replication

role:master

connected\_slaves:3

min\_slaves\_good\_slaves:3

slave0:ip=192.168.200.113,port=6379,state=online,offset=1562135,lag=1

slave1:ip=192.168.200.112,port=6380,state=online,offset=1562135,lag=1

slave2:ip=192.168.200.112,port=6379,state=online,offset=1562135,lag=1

master\_replid:e4970e9361f4ba3c6d7e3d723f67b230471ecf99

master\_replid2:ac1ada5cc0f3a83052f637ad1080f12c0c25b8b2

master\_repl\_offset:1562282

second\_repl\_offset:1534069

repl\_backlog\_active:1

repl\_backlog\_size:1048576

repl\_backlog\_first\_byte\_offset:1512833

repl\_backlog\_histlen:49450

主从复制完成，一主三从成功

**（二）：配置Redis sentinel （哨兵在master，可以在多台上装，本实验只装一台）**

[root@master ~]# cp /usr/src/redis-4.0.9/src/redis-sentinel /etc/redis/

####复制sentinel启动服务文件

[root@master ~]# cp /usr/src/redis-4.0.9/sentinel.conf /etc/redis/

######复制sentinel主配置文件

[root@master ~]# cd /etc/redis/

[root@master ~]# mkdir -p /var/redis/data

#####创建日志文件存放位置

**（三）更改配置**

[root@master redis]# vi /etc/redis/sentinel.conf

#########修改配置文件 原配置文件的内容可以全部删除只输入以下6行也可以直接在原配置上更改。.

port 26379

############sentinel监听端口，默认是26379，可以修改

dir "/var/redis/data"

###指定日志文件的存放位置

sentinel monitor mymaster 192.168.200.111 6379 1

############告诉sentinel去监听地址为ip:port的一个master，这里的master-name可以自定义，quorum是一个数字，指明当有多少个sentinel认为一个master失效时，master才算真正失效。master-name只能包含英文字母，数字，和“.-\_”这三个字符需要注意的是master-ip 要写真实的ip地址而不要用回环地址（127.0.0.1）。

sentinel down-after-milliseconds mymaster 3000

##########这个配置项指定了需要多少失效时间，一个master才会被这个sentinel主观地认为是不可用的。 单位是毫秒，默认为30秒（30000毫秒），改为3秒（3000）

sentinel auth-pass mymaster 123456

##########设置连接master和slave时的密码，注意的是sentinel不能分别为master和slave设置不同的密码，因此master和slave的密码应该设置相同。

sentinel parallel-syncs mymaster 1

###############这个配置项指定了在发生failover主备切换时最多可以有多少个slave同时对新的master进行 同步，这个数字越小，完成failover所需的时间就越长，但是如果这个数字越大，就意味着越 多的slave因为replication而不可用。可以通过将这个值设为 1 来保证每次只有一个slave 处于不能处理命令请求的状态。

sentinel failover-timeout mymaster 180000

##########故障转移超时时间为180000

protected-mode no

##########保护模式只允许127.0.0.1连接，必须打开。在配置多个sentinel中在每个配置文件加入，该配置必须添加，否则主的redis挂掉，所有的哨兵不会选举出新的master。

daemonize yes

#########以守护进程模式启动，默认无，添加上，否则占用前台进程

**退出前注意配置文件的最后是没有各个从服务器信息的**

**（四）启动sentinel验证启动成功：**

**[root@master redis]# redis-sentinel /etc/redis/sentinel.conf**

**[root@master~]# netstat -lnpt|grep 6379**

**tcp 0 0 192.168.200.113:6379 0.0.0.0:\* LISTEN 11641/redis-server**

**tcp 0 0 0.0.0.0:26379 0.0.0.0:\* LISTEN 11100/redis-sentine**

**再查看/etc/redis/sentinel.conf文件，sentinel（哨兵）已经发现了所有从服务器**

**# Generated by CONFIG REWRITE**

**sentinel known-slave mymaster 192.168.200.113 6379**

**sentinel known-slave mymaster 192.168.200.112 6379**

**sentinel known-slave mymaster 192.168.200.112 6380**

**sentinel current-epoch 0**

**查看sentinel信息：**

**[root@master redis]#redis-cli -p 26379 -a 123456 -h 192.168.200.111**

**192.168.200.111:26379> info sentinel**

**# Sentinel**

**sentinel\_masters:1**

**sentinel\_tilt:0**

**sentinel\_running\_scripts:0**

**sentinel\_scripts\_queue\_length:0**

**sentinel\_simulate\_failure\_flags:0**

**master0:name=mymaster,status=ok,address=192.168.200.111:6379,slaves=3,sentinels=1**

**查看sentinel配置文件，自动添加了各个从服务器信息**

**[root@master redis]# tail -5 /etc/redis/sentinel.conf**

**sentinel leader-epoch mymaster 10**

**sentinel known-slave mymaster 192.168.200.113 6379**

**sentinel known-slave mymaster 192.168.200.112 6379**

**sentinel known-slave mymaster 192.168.200.112 6380**

**sentinel current-epoch 10**

**（五）测试sentinel的转移功能**

**[root@master redis]#killall -9 redis-server**

**[root@master redis]#rm -f /var/run/redis\_6379.pid**

**##将redis的master关闭,**

**[root@master ~]# redis-cli -p 6379 -a 123456 -h 192.168.200.111 info replication**

**Could not connect to Redis at 192.168.200.111:6379: Connection refused**

**[root@master ~]# redis-cli -p 26379 -a 123456 -h 192.168.200.111 info sentinel**

**# Sentinel**

**sentinel\_masters:1**

**sentinel\_tilt:0**

**sentinel\_running\_scripts:0**

**sentinel\_scripts\_queue\_length:0**

**sentinel\_simulate\_failure\_flags:0**

**master0:name=mymaster,status=ok,address=192.168.200.113:6379,slaves=3,sentinels=1**

**##查看哨兵选择的新的master为192.168.200.113，实现了主服务器的转移**

**再查看sentinel的配置文件，原主服务器已经在从服务器列表中了（虽然可能还不能工作），新主已经不在从服务器列表了**

**# Generated by CONFIG REWRITE**

**sentinel known-slave mymaster 192.168.200.111 6379**

**sentinel known-slave mymaster 192.168.200.112 6379**

**sentinel known-slave mymaster 192.168.200.112 6380**

**sentinel current-epoch 1**

**[root@master ~]# redis-cli -p 6379 -a 123456 -h 192.168.200.113 info replication**

**# Replication**

**role:master**

**connected\_slaves:2**

**slave0:ip=192.168.200.112,port=6379,state=online,offset=1613692,lag=1**

**slave1:ip=192.168.200.112,port=6380,state=online,offset=1613839,lag=0**

**master\_replid:417c338ea53d46eda811fda0cd3dc003a8131407 #新的master id**

**master\_replid2:e4970e9361f4ba3c6d7e3d723f67b230471ecf99 #原master id**

**master\_repl\_offset:1613839**

**second\_repl\_offset:1602288**

**repl\_backlog\_active:1**

**repl\_backlog\_size:1048576**

**repl\_backlog\_first\_byte\_offset:1534386**

**repl\_backlog\_histlen:79454**

**##查看新master上的信息有两个从服务器**

**root@slave1 ~]# grep ^slaveof /etc/redis/6379.conf**

**slaveof 192.168.200.113 6379**

**[root@slave1 ~]# grep ^slaveof /etc/redis/6380.conf**

**slaveof 192.168.200.113 6379**

**##两个从上已经自动更改主服务器为新master主机ip**

**[root@slave2 ~]# grep ^slaveof /etc/redis/6379.conf**

**[root@slave2 ~]#**

**##新主服务器已经删除作为从服务器时指定的原主服务器信息，**

**再启动原master上的redis：**

**[root@master ~]# /etc/init.d/redis\_6379 start**

**Starting Redis server...**

**[root@master ~]# tail -5 /etc/redis/sentinel.conf**

**sentinel leader-epoch mymaster 20**

**sentinel known-slave mymaster 192.168.200.111 6379**

**sentinel known-slave mymaster 192.168.200.112 6380**

**sentinel known-slave mymaster 192.168.200.112 6379**

**sentinel current-epoch 20**

**##哨兵检测到的新的从服务器列表，原主服务器已经在从服务器列表中**

**[root@master ~]# redis-cli -p 6379 -a 123456 -h 192.168.200.113 info replication**

**# Replication**

**role:master**

**connected\_slaves:3**

**slave0:ip=192.168.200.112,port=6379,state=online,offset=1640105,lag=0**

**slave1:ip=192.168.200.112,port=6380,state=online,offset=1640091,lag=1**

**slave2:ip=192.168.200.111,port=6379,state=online,offset=1640091,lag=1**

**master\_replid:417c338ea53d46eda811fda0cd3dc003a8131407**

**master\_replid2:e4970e9361f4ba3c6d7e3d723f67b230471ecf99**

**master\_repl\_offset:1640252**

**second\_repl\_offset:1602288**

**repl\_backlog\_active:1**

**repl\_backlog\_size:1048576**

**repl\_backlog\_first\_byte\_offset:1534386**

**repl\_backlog\_histlen:105867**

**##原主服务器已经是新主服务器的从服务器**

**[root@master ~]# grep ^slaveof /etc/redis/6379.conf**

**slaveof 192.168.200.113 6379**

**##原主服务器配置文件中自动添加了新主服务器的信息**