http://blog.csdn.net/donggang1992/article/details/50981341

http://blog.csdn.net/kity9420/article/details/53571718

先简单介绍下Redis集群的方式有哪些

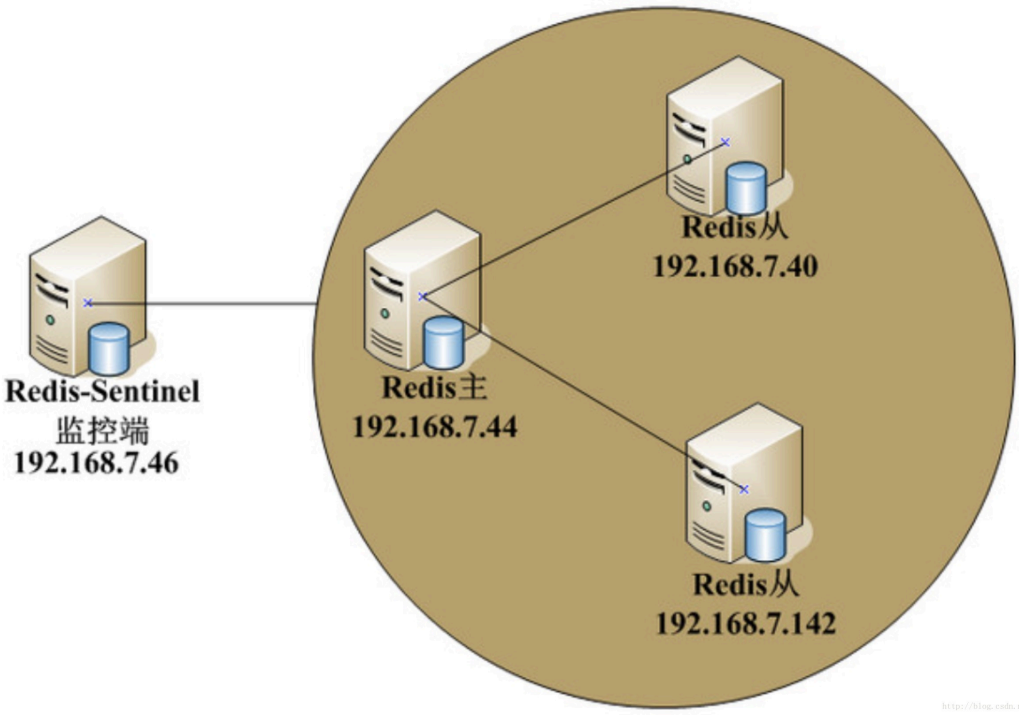
1、单机版 不解释

2、Sentinel 哨兵模式

3、Redis Cluster Redis官方集群方案

4、Redis Sharding集群

正确的说，哨兵模式是一主多从的结构，并不属于真正的集群，真正的集群应该是多主多从的结构，需要有多台不同物理地址的主机，无赖家里只有一台PC，只能使用Sentinel模式来做集群。先来看下Sentinel模式的集群**架构**图



一、sentinel介绍

Redis Sentinel

Sentinel(哨兵)是用于监控redis集群中Master状态的工具，其已经被集成在redis2.4+的版本中

Sentinel作用：

1)：Master状态检测

2)：如果Master异常，则会进行Master-Slave切换，将其中一个Slave作为Master，将之前的Master作为Slave

3)：Master-Slave切换后，master_redis.conf、slave_redis.conf和sentinel.conf的

内容都会发生改变，即master_redis.conf中会多一行slaveof的配置，sentinel.conf的监控目标会随之调换

Sentinel工作方式：

1)：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

2)：如果一个实例（instance）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。

3)：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。

4)：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

5)：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

6)：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 秒一次改为每秒一次

7)：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。

主观下线和客观下线

主观下线：Subjectively Down，简称 SDOWN，指的是当前 Sentinel 实例对某个redis服务器做出的下线判断。

客观下线：Objectively Down， 简称 ODOWN，指的是多个 Sentinel 实例在对 Master Server做出 SDOWN 判断，并且通过 SENTINEL is-master-down-by-addr 命令互相交流之后，得出的Master Server下线判断，然后开启failover.

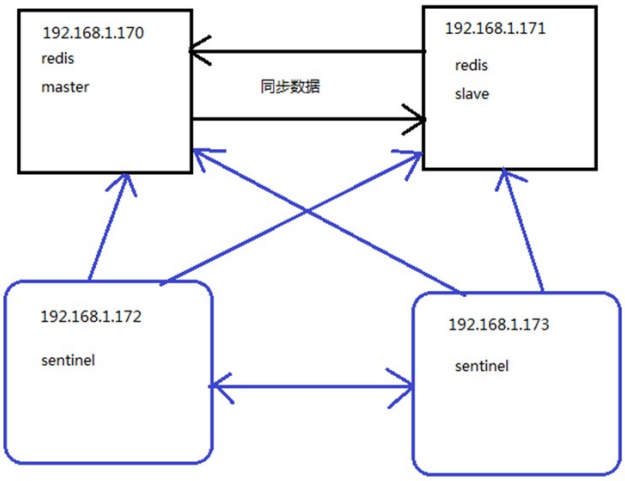
通俗来讲就是：

redis的sentinel系统用来管理多个redis服务器，可以实现一个功能上实现HA的集群。该系统主要执行三个任务：

①监控（ Monitoring ）： Redis Sentinel实时监控主服务器和从服务器运行状态。

②提醒（notification）： 当被监控的某个 Redis 服务器出现问题时， Redis Sentinel 可以向系统管理员发送通知， 也可以通过 API 向其他程序发送通知

一个简单的主从结构加sentinel集群的架构图如下：



主观下线状态：单个sentinel实例对主节点做出的下线判断，
客观下线状态：多个sentinel实例对主节点做出的下线判断

主节点：主观下线状态和客观下线状态
从节点：只有主观下线状态

上图是一主一从节点，加上两个部署了sentinel的集群，sentinel集群之间会互相通信，沟通交流redis节点的状态，做出相应的判断并进行处理，这里的主观下线状态和客观下线状态是比较重要的状态，它们决定了是否进行故障转移
可以 通过订阅指定的频道信息，当服务器出现故障得时候通知管理员
客户端可以将 Sentinel 看作是一个只提供了订阅功能的 Redis 服务器，你不可以使用 PUBLISH 命令向这个服务器发送信息，但你可以用 SUBSCRIBE 命令或者 PSUBSCRIBE 命令， 通过订阅给定的频道来获取相应的事件提醒。
一个频道能够接收和这个频道的名字相同的事件。 比如说， 名为 +sdown 的频道就可以接收所有实例进入主观下线（SDOWN）状态的事件。

二、搭建redis-sentinel 集群环境

1.、在/usr/local/ 下新建一个目录redis-sentinel，然后在此目录下新建7501/ 7502/ 7503/ 7504/ 7505/ 7506/ 六个目录。
2.、将redis安装目录下的reids.conf，拷贝到前4个目录下，分别命名为：
Redis-7501.conf redis-7502.conf redis-7503.conf redis-7504.conf
修改配置文件内容（以redis-7501.conf为例）：

```
daemonize yes
Port 7501
Bind 192.168.12.90
logfile "./redis-7501.log"
```

3、 将redis安装目录下的sentinel.conf拷贝到7505/和7506/目录下分别命名：
Sentinel-7505.conf sentinel-7506.conf
修改配置文件(以sentinel-7505.conf为例)：

```
port 7505
sentinel monitor mymaster 192.168.12.90 7501 2
```

注：我们稍后要启动四个redis实例，其中端口为7501的redis设为master，其他三个设为slave 。所以my mymaster 后跟的是master的ip和端口，最后一个’ 2’ 代表我要启动只要有2个sentinel认为master下线，就认为该master客观下线，启动failover并选举产生新的master。通常最后一个参数不能多于启动的sentinel实例数。

4、启动redis和sentinel

分别启动4个redis实例：

```
redis-server redis-7501.conf
...
```

然后分别登陆7502 7503 7504三个实例，动态改变主从关系，成为7501的slave:

```
redis-cli -h 192.168.12.90 -p 7502
192.168.12.90:7502> SLAVEOF 192.168.12.90 7501
```

以后台启动模式启动两个sentinel（哨兵）：

```
redis-sentinel sentinel-7505.conf &
```

5、sentinel一些命令介绍

要使用sentinel的命令，我们需要用redis-cli命令进入到sentinel：

```
redis-cli -h 192.168.12.90 -p 7505
```

① INFO
sentinel的基本状态信息
②SENTINEL masters
列出所有被监视的主服务器，以及这些主服务器的当前状态
③ SENTINEL slaves
列出给定主服务器的所有从服务器，以及这些从服务器的当前状态
④SENTINEL get-master-addr-by-name
返回给定名字的主服务器的 IP 地址和端口号
⑤SENTINEL reset
重置所有名字和给定模式 pattern 相匹配的主服务器。重置操作清除主服务器目前的所有状态， 包括正在执行中的故障转移， 并移除目前已经发现和关联的， 主服务器的所有从服务器和 Sentinel 。
⑥SENTINEL failover
当主服务器失效时， 在不询问其他 Sentinel 意见的情况下， 强制开始一次自动故障迁移，但是它会给其他sentinel发送一个最新的配置，其他sentinel会根据这个配置进行更新

6、**测试**：

（1）登陆到 master：

```
redis-cli -h 192.168.12.90 -p 7501
192.168.12.90:7501> set name "zhangsan"
[root@localhost redis-sentinel]# redis-cli -h 192.168.12.90 -p 7502
192.168.12.90:7502> get name
"zhangsan"
192.168.12.90:7502> set age 24
(error) READONLY You can't write against a read only slave.
```

可以看到：我们的主从模式中，slave默认是只读。

（2）目前7501是master, 我们强制kill掉 7501 的进程以后，可以看到sentinel打出的信息：

```
[root@localhost redis-sentinel]# kill -9 38905
[root@localhost redis-sentinel]# 39212:X 25 Mar 15:25:42.649 # +sdown master mymaster
192.168.12.90 7501
39204:X 25 Mar 15:25:42.652 # +sdown master mymaster 192.168.12.90 7501
39204:X 25 Mar 15:25:42.728 # +odown master mymaster 192.168.12.90 7501 #quorum 2/2
39204:X 25 Mar 15:25:42.728 # +new-epoch 1
39204:X 25 Mar 15:25:42.728 # +try-failover master mymaster 192.168.12.90 7501
39204:X 25 Mar 15:25:42.730 # +vote-for-leader 02779767a170f868be9cd370e4dcc2fbc8c50b5
3 1
39212:X 25 Mar 15:25:42.733 # +new-epoch 1
39212:X 25 Mar 15:25:42.735 # +vote-for-leader 02779767a170f868be9cd370e4dcc2fbc8c50b5
3 1
39204:X 25 Mar 15:25:42.736 # 192.168.12.90:7506 voted for 02779767a170f868be9cd370e4d
cc2fbc8c50b53 1
39204:X 25 Mar 15:25:42.786 # +elected-leader master mymaster 192.168.12.90 7501
39204:X 25 Mar 15:25:42.786 # +failover-state-select-slave master mymaster 192.168.12.
90 7501
39204:X 25 Mar 15:25:42.870 # +selected-slave slave 192.168.12.90:7504 192.168.12.90 7
504 @ mymaster 192.168.12.90 7501
39204:X 25 Mar 15:25:42.871 * +failover-state-send-slaveof-noone slave 192.168.12.90:7
504 192.168.12.90 7504 @ mymaster 192.168.12.90 7501
39204:X 25 Mar 15:25:42.961 * +failover-state-wait-promotion slave 192.168.12.90:7504
192.168.12.90 7504 @ mymaster 192.168.12.90 7501
39212:X 25 Mar 15:25:43.758 # +odown master mymaster 192.168.12.90 7501 #quorum 2/2
39212:X 25 Mar 15:25:43.758 # Next failover delay: I will not start a failover before
Fri Mar 25 15:31:42 2016
39204:X 25 Mar 15:25:43.769 # +promoted-slave slave 192.168.12.90:7504 192.168.12.90 7
504 @ mymaster 192.168.12.90 7501
39204:X 25 Mar 15:25:43.769 # +failover-state-reconf-slaves master mymaster 192.168.12
.90 7501
39204:X 25 Mar 15:25:43.829 * +slave-reconf-sent slave 192.168.12.90:7502 192.168.12.9
0 7502 @ mymaster 192.168.12.90 7501
39212:X 25 Mar 15:25:43.831 # +config-update-from sentinel 192.168.12.90:7505 192.168.
12.90 7505 @ mymaster 192.168.12.90 7501
39212:X 25 Mar 15:25:43.831 # +switch-master mymaster 192.168.12.90 7501 192.168.12.90
 7504
39212:X 25 Mar 15:25:43.831 * +slave slave 192.168.12.90:7503 192.168.12.90 7503 @ mym
aster 192.168.12.90 7504
39212:X 25 Mar 15:25:43.834 * +slave slave 192.168.12.90:7502 192.168.12.90 7502 @ mym
aster 192.168.12.90 7504
```

可以看到，sentinel已经将7504这个redis instance提升为新的master，稍后将7501这个实例启动，动态作为7504的slave，这样就手动恢复了redis 集群。