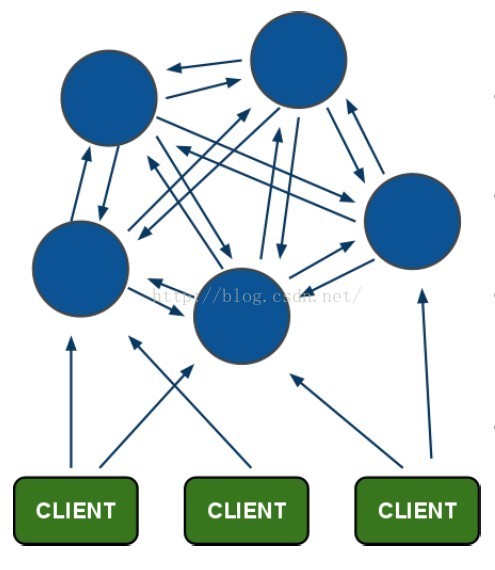
# Redis Cluster集群

## redis-cluster设计

Redis集群搭建的方式有很多种，例如使用zookeeper等，但是redis3.0之后版本支持redis-cluster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。其redis-cluster架构图如下：



其结构特点：

1. 所有的redis节点彼此互联（PING-PONG机制），内部使用二进制协议优化传输速度和带宽。
2. 节点的fail通过集群中超过半数的节点检测失效时才生效。（所以一般节点数为奇数）
3. 客户端与redis节点直连，不需要中间proxy层。客户端不需要连接集群所有节点，连接集群中任何一个**可用节点**即可。
4. Redis-cluster把所有的物理节点映射到[0-16383]slot上（一般是平均分配，也可手动分配）。
5. Redis集群预分好16384个桶，当需要在Redis集群中放置一个key-value时，根据CRC16(key) mod16384的值，决定将key放到哪一个桶中。

### redis cluster节点分配

现在我们三个主节点分别是A,B,C三个节点。他们可以是一台机器上的三个端口，也可以是三台不同的服务器。那么，采用哈希槽的方式来分配16384个slot的话，它们三个节点跟别承担的slot是：

节点A覆盖0 – 5460

节点B覆盖5461 – 10922

节点C覆盖 10923 – 16383

**获取数据：**

如果存入一个值，按照redis cluster哈希槽的算法： CRC16(‘key’)%16384=6782。那么就会把这个key的存储分配到B上了。同样，当我连接A,B,C任何一个节点想获取’key’这个key时，也会使用这样的计算定位，然后内部跳转到B节点上获取数据

**新增一个主节点：**

新增一个节点D，redis cluster的这种做法是从各个节点的前面各自去一部分slot到D上，可以手动指定一个节点对应slot到D,或者正常做法是从每一个节点各取一部分到D上：

节点A覆盖1365-5460  
    节点B覆盖6827-10922  
 节点C覆盖12288-16383  
  节点D覆盖0-1364,5461-6826,10923-12287

同样删除一个节点也是类似，将该节点上的槽转移完成后就可以删除这个节点了。

### Redis 主从模式

redis cluster 为了保证数据的高可用性，加入了主从模式，一个主节点对应一个或多个从节点，主节点提供数据存取，从节点则是从主节点拉取数据备份，当这个主节点挂掉后，就会有这个从节点选取一个来充当主节点，从而保证集群不会挂掉。  
      上面那个例子里, 集群有ABC三个主节点, 如果这3个节点都没有加入从节点，如果B挂掉了，我们就无法访问整个集群了。A和C的slot也无法访问。

（这里是在正常的配置下会出现的情况，原因是16384个 slot没有全部正常分配，集群状态为fail，导致其他节点也不能工作，可以在redis.conf里配置cluster-require-full-coverage no）  
     所以我们在集群建立的时候，一定要为每个主节点都添加了从节点, 比如像这样, 集群包含主节点A、B、C, 以及从节点A1、B1、C1, 那么即使B挂掉系统也可以继续正确工作。  
     B1节点替代了B节点，所以Redis集群将会选择B1节点作为新的主节点，集群将会继续正确地提供服务。 当B重新开启后，它就会变成B1的从节点。  
    不过需要注意，如果节点B和B1同时挂了，Redis集群就无法继续正确地提供服务了。

### 一、主从模式

通过持久化功能，Redis保证了即使在服务器重启的情况下也不会损失（或少量损失）数据，因为持久化会把内存中数据保存到硬盘上，重启会从硬盘上加载数据。

但是由于数据是存储在一台服务器上的，如果这台服务器出现硬盘故障等问题，也会导致数据丢失。为了避免单点故障，通常的做法是将数据库复制多个副本以部署在不同的服务器上，这样即使有一台服务器出现故障，其他服务器依然可以继续提供服务。为此， **Redis 提供了复制（replication）功能，可以实现当一台数据库中的数据更新后，自动将更新的数据同步到其他数据库上**。

在复制的概念中，数据库分为两类，一类是主数据库（master），另一类是从数据库(slave）。主数据库可以进行读写操作，当写操作导致数据变化时会自动将数据同步给从数据库。而**从数据库一般是只读的，并接受主数据库同步过来的数据**。一个主数据库可以拥有多个从数据库，而一个从数据库只能拥有一个主数据库。

#### 主从数据库的配置

主数据库不用配置，从redis的conf文件中可以加载从数据库的信息，也可以在启动时，使用 redis-server --port 6380 --slaveof 127.0.0.1 6379  
从数据库一般是只读，可以改为可写，但写入的数据很容易被主同步没，所以还是只读就可以。

也可以在运行时使用slaveof ip port命令，停止原来的主，切换成刚刚设置的主 slaveof no one会把自己变成主

#### 复制原理

当**从数据库启动时，会向主数据库发送sync命令，主数据库接收到sync后开始在后台保存快照rdb，在**保存快照期间收到的命令缓存起来，当快照完成时，主数据库会将快照和缓存的命令一块发送给从\*\*。复制初始化结束。  
之后，主每收到1个命令就同步发送给从。  
当出现断开重连后，2.8之后的版本会将断线期间的命令传给重数据库。增量复制

主从复制是乐观复制，当客户端发送写执行给主，主执行完立即将结果返回客户端，并异步的把命令发送给从，从而不影响性能。也可以设置至少同步给多少个从主才可写。

无硬盘复制:如果硬盘效率低将会影响复制性能，2.8之后可以设置无硬盘复制，repl-diskless-sync yes

### 二、哨兵

当主数据库遇到异常中断服务后，开发者可以通过手动的方式选择一个从数据库来升格为主数据库，以使得系统能够继续提供服务。**然而整个过程相对麻烦且需要人工介入，难以实现自动化**。 为此，Redis 2.8中提供了哨兵工具来实现自动化的系统监控和故障恢复功能。

**哨兵的作用就是监控redis主、从数据库是否正常运行，主出现故障自动将从数据库转换为主数据库**。

例子：  
1主2从1哨兵

redis-server --port 6379

redis-server --port 6380 --slaveof 192.168.0.167 6379

redis-server --port 6381 --slaveof 192.168.0.167 6379

哨兵配置文件 sentinel.conf

sentinel monitor mymaster 192.168.0.167 6379 1

这里的1代表1个哨兵

注：

配置哨兵监控一个系统时，**只需要配置其监控主数据库即可**，哨兵会自动发现所有复制该主数据库的从数据库

这样哨兵就能监控主6379和从6380、6381，一旦6379挂掉，哨兵就会在2个从中选择一个作为主，**根据优先级选，如果一样就选个id小的**，当6379再起来就作为从存在。

### 三、集群(cluster-enable)

使用集群，只需要将每个数据库节点的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。

即使使用哨兵，redis每个实例也是全量存储，每个redis存储的内容都是完整的数据，浪费内存且有木桶效应。为了最大化利用内存，可以采用集群，就是分布式存储。即每台redis存储不同的内容。  
集群至少需要3主3从，且每个实例使用不同的配置文件，主从不用配置，集群会自己选。  
修改每个实例的配置文件：

cluster-enabled yes --开启集群

cluster-config-file nodes-6382.conf --集群配置文件名，

每个实例配置的要不同，redis会根据文件名自动新建

集群的运行

redis安装目录的src执行./redis-trib.rb create --replicas 1

127.0.0.1:6380 127.0.0.1:6381 127.0.0.1:6382

127.0.0.1:6383 127.0.0.1:6384 127.0.0.1:6385

这里的master选举和zookeeper的相似