一致性Hash算法

- 产生背景
  - 一致性Hash算法，是对Hash算法的一种优化
  - Hash算法
    - 对数据进行哈希函数计算后，% 服务器数量，就知道数据该放在哪个服务器上了
    - 缺陷是，当某台服务器发生故障时，所有数据需要都需要重新进行取模运算，因为 服务器数量发生了变化
    - 如果是一个缓存服务的话，此时所有缓存失效，容易引起缓存雪崩
- 是什么
  - 分布式系统中，进行数据分片的一种算法
  - 一致性Hash算法
    - 首先将服务器通过IP地址或主机名进行哈希计算，然后 % 2^32，会得到一个 [0, 2^32-1] 范围内的整数
    - 将该整数映射到一个环形序列中，我们叫 Hash环
    - 然后将数据进行哈希计算，同样 % 2^32，映射到Hash环上
    - 从数据在环上的位置，顺时针方向， 遇到的第一个服务器节点位置，那么该数据就放在该服务器节点上
    - 这样，当某服务器故障时，受影响的只是部分数据， 不会造成全部数据缓存失效
    - 新增节点时，也不需要移动原有数据
  - 带虚拟节点的一致性Hash算法
    - 在服务器映射到Hash环的时候，可能会出现Hash倾斜，就是多个服务器节点映射到环上的位置比较接近，导致各个节点负载不均衡，仍然会产生普通Hash算法的问题
    - 将真实的服务器节点通过虚拟节点的方式复制多个，这时映射到环上的节点会均衡很多
    - 虚拟节点增加了Hash环的分散性
- 应用
  - Memcache 分布式缓存
  - 负载均衡
  - 数据库分库分表
- 扩展
  - redis 的HashSlot 方案
    - 类似于Hash环，Redis Cluster采用HashSlot来实现Key值的均匀分布和实例的增删管理。
    - 首先默认分配了16384个Slot（这个大小正好可以使用2kb的空间保存），每个Slot相当于一致性Hash环上的一个节点。接入集群的所有实例将均匀地占有这些Slot，而最终当我们Set一个Key时，使用CRC16(Key) % 16384来计算出这个Key属于哪个Slot，并最终映射到对应的实例上去。
    - 当增加节点时，会把每台机器上的一部分slot移动到新节点上，同理，删除也是归还slot到 各个机器上
    - 每个节点都保存有完整的HashSlot - 节点映射表，也就是说，每个节点都知道自己拥有哪些Slot，以及某个确定的Slot究竟对应着哪个节点
    - 无论向哪个节点发出寻找Key的请求，该节点都会通过CRC(Key) % 16384计算该Key究竟存在于哪个Slot，并将请求转发至该Slot所在的节点。
    - 总结一下就是两个要点：映射表和内部转发，这是通过著名的**Gossip协议**来实现的