<https://www.cnblogs.com/liyasong/p/redis_jiqun.html?utm_source=itdadao&utm_medium=referral>

redis是单线程，但是一般的作为缓存使用的话，redis足够了，因为它的读写速度太快了。

　　　   官方的一个简单测试：

　　　　　　测试完成了**50个并发**执行**100000个请求**。

　　　　　　设置和获取的值是一个256字节字符串。

　　　　　　结果:**读的速度是110000次/s,写的速度是81000次/s**

　　　　在这么快的读写速度下，对于一般程序来说足够用了，但是对于访问量特别大的网站来说，还是稍有不足。那么，如何提升redis的性能呢？看标题就知道了，搭建集群。

## 3.0版本之前

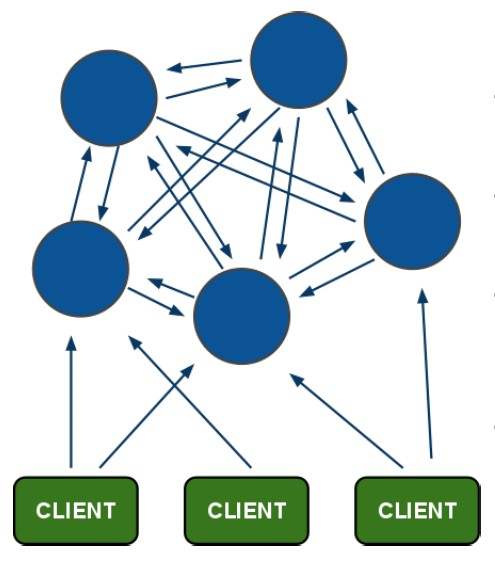
　　　　3.0版本之前的redis是不支持集群的，我们的徐子睿老师说，那个时候，我们的redis如果想要集群的话，就需要一个中间件，然后这个中间件负责将我们需要存入redis中的数据的key通过一套算法计算得出一个值。然后根据这个值找到对应的redis节点，将这些数据存在这个redis的节点中。

　　　　在取值的时候，同样先将key进行计算，得到对应的值，然后就去找对应的redis节点，从对应的节点中取出对应的值。

　　　　这样做有很多不好的地方，比如说我们的这些计算都需要在系统中去进行，所以会增加系统的负担。还有就是这种集群模式下，某个节点挂掉，其他的节点无法知道。而且也不容易对每个节点进行负载均衡。

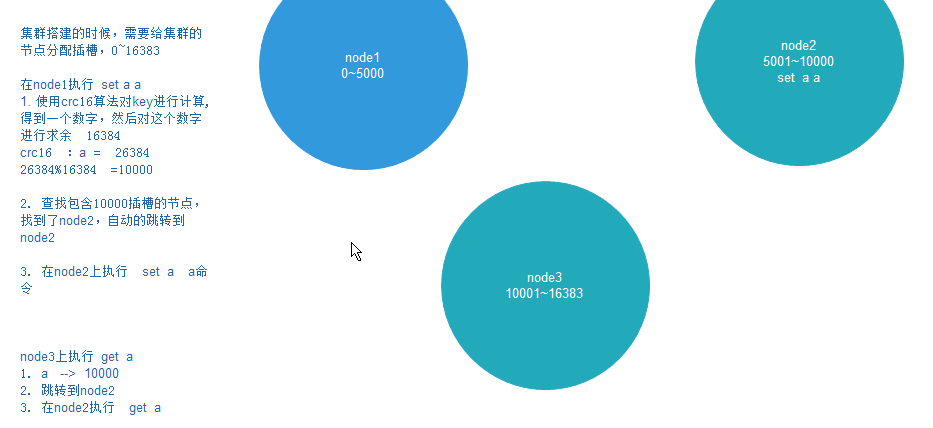
## 3.0版本及以后

　　　　先来一张redis集群的架构图：



　　　　在这个图中，每一个蓝色的圈都代表着一个redis的服务器节点。它们任何两个节点之间都是相互连通的。客户端可以与任何一个节点相连接，然后就可以访问集群中的任何一个节点。对其进行存取和其他操作。

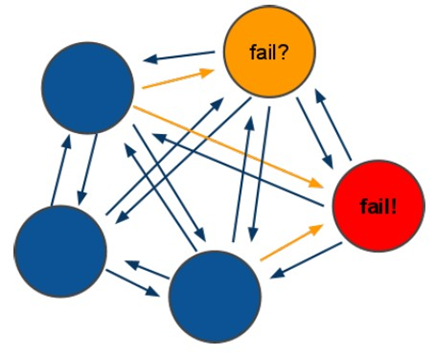
　　　　那么redis是怎么做到的呢？首先，在redis的每一个节点上，都有这么两个东西，一个是插槽（slot）可以理解为是一个可以存储两个数值的一个变量这个变量的取值范围是：0-16383。还有一个就是cluster我个人把这个cluster理解为是一个集群管理的插件。当我们的存取的key到达的时候，redis会根据crc16的算法得出一个结果，然后把结果对 16384 求余数，这样每个 key 都会对应一个编号在 0-16383 之间的哈希槽，通过这个值，去找到对应的插槽所对应的节点，然后直接自动跳转到这个对应的节点上进行存取操作。



　　　　还有就是因为如果集群的话，是有好多个redis一起工作的，那么，就需要这个集群不是那么容易挂掉，所以呢，理论上就应该给集群中的每个节点至少一个备用的redis服务。这个备用的redis称为从节点（slave）。那么这个集群是如何判断是否有某个节点挂掉了呢？

　　　　首先要说的是，每一个节点都存有这个集群所有主节点以及从节点的信息。

　　　　它们之间通过互相的ping-pong判断是否节点可以连接上。如果有一半以上的节点去ping一个节点的时候没有回应，集群就认为这个节点宕机了，然后去连接它的备用节点。如果某个节点和所有从节点全部挂掉，我们集群就进入faill状态。还有就是如果有一半以上的主节点宕机，那么我们集群同样进入发力了状态。这就是我们的redis的投票机制，具体原理如下图所示：



　　　　(1)投票过程是集群中所有master参与,如果半数以上master节点与master节点通信超时(cluster-node-timeout),认为当前master节点挂掉.

　　　　(2):什么时候整个集群不可用(cluster\_state:fail)?

    　　　　a:如果集群任意master挂掉,且当前master没有slave.集群进入fail状态,也可以理解成集群的slot映射[0-16383]不完整时进入fail状态. ps : redis-3.0.0.rc1加入cluster-require-full-coverage参数,默认关闭,打开集群兼容部分失败.

    　　　　b:如果集群超过半数以上master挂掉，无论是否有slave，集群进入fail状态.

分类: [redis](http://www.cnblogs.com/liyasong/category/946379.html" \t "_blank)

标签: [javaee](http://www.cnblogs.com/liyasong/tag/javaee/), [原理](http://www.cnblogs.com/liyasong/tag/%E5%8E%9F%E7%90%86/), [数据库](http://www.cnblogs.com/liyasong/tag/%E6%95%B0%E6%8D%AE%E5%BA%93/), [NoSql](http://www.cnblogs.com/liyasong/tag/NoSql/), [redis](http://www.cnblogs.com/liyasong/tag/redis/)