<https://zhuanlan.zhihu.com/p/161822264>

<https://www.jianshu.com/p/5589731a0ff8>

<https://www.zsythink.net/archives/1182>

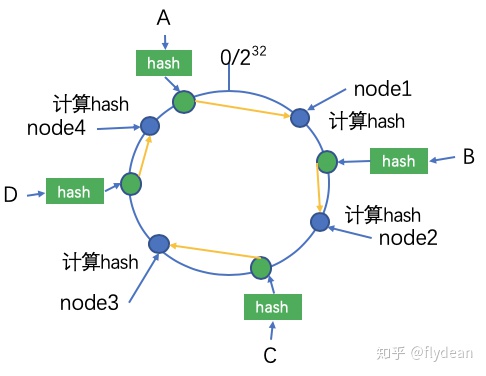
Memcached的客户端需要对key进行一次哈希计算(阶段一哈希)，来判断这个key存放在哪个结点上，通常是hash(key) % N，这样就得到一个在0-N之间的值，也就是结点的编号，其中N是结点个数。

这样的哈希在实际使用时会存在很大的问题，原因是结点有可能会出现宕机情况，也可能需要新增结点到系统中，这样就会导致几乎所有之前的缓存都失效的情况，毕竟N发生了变化。

为了解决这个问题，就出现了一致性哈希算法，它采用一个包含个点的hash环，然后对结点的IP地址和存储对象的key进行hash，hash(IP地址) % 

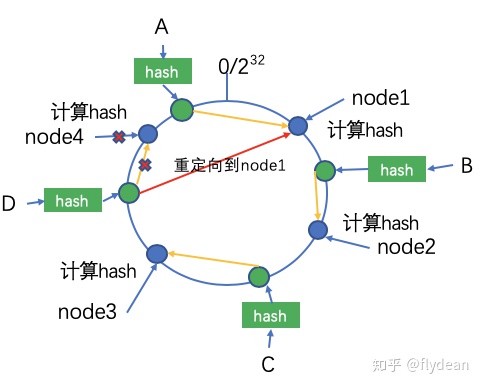
hash(key) % 

这样就会出现如下图所示的情况：



node1，node2，node3和node4总共4个memcached结点，A，B，C，D总共4个数据对象都落在了这个hash环上，按照顺时针方向，碰到的第一个结点，就是该数据对象要存放的位置，A存在node1，B存放在node2，C存放在node3，D存放在node4。

一致性哈希算法可以有效的结点宕机和增加结点的问题，如下图所示：

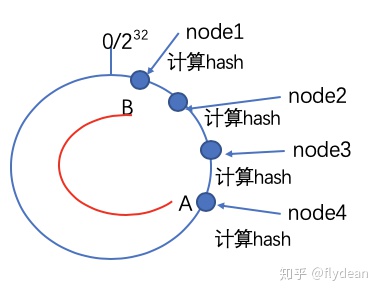


当node4挂掉时，A，B，C都不会受到影响，只有D会重新存储在node1上，这样整个系统中只有少部分key(毕竟一般情况下，挂掉的结点都是少数)会受到影响。

同理，如果增加了结点，也只有少部分key会受到影响。

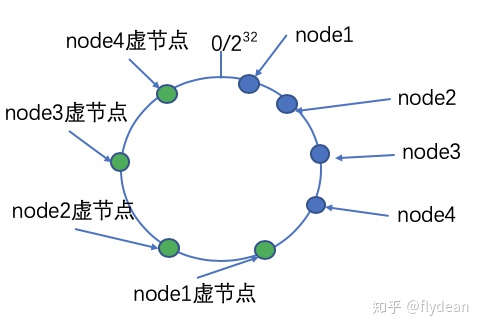
通过一致性哈希算法后，系统具有较强的扩展性和容错性。

一致性哈希算法会出现一种不均衡的问题，即哈希倾斜，尤其是结点较少时，



几个node的哈希值都很接近，导致大量key都存放在node1上，其它node几乎没有值。

为了解决这种问题，可以增加虚拟结点，即对每个结点都计算多个哈希值(一台机器可能有多个唯一的标识符，可以分别用这些标识符来计算哈希值)，每个值都放置在hash环的一个位置上，这样就使结点尽可能的均匀分布在hash环上：



判断key的存放位置时，虚拟结点也视为服务结点，key就存放在它对应的实际结点上。