一、前言

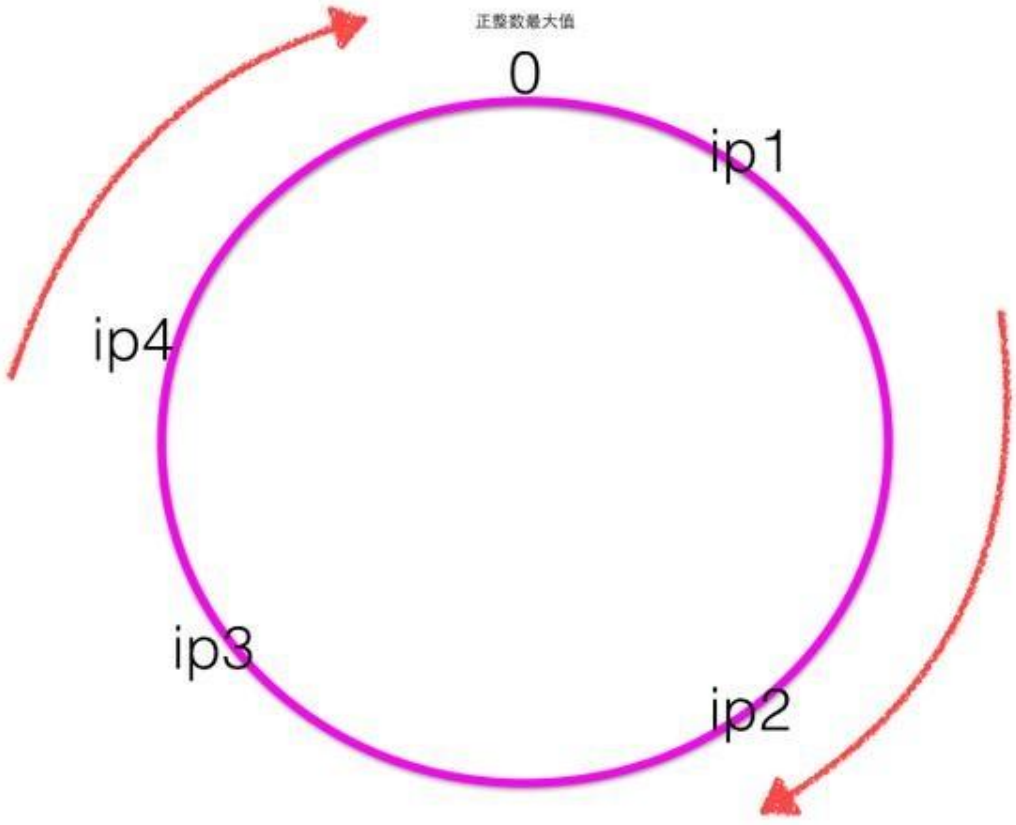
在解决分布式系统中负载均衡的问题时候可以使用Hash算法让固定的一部分请求落到同一台服务器上，这样每台服务器固定处理一部分请求（并维护这些请求的信息），起到负载均衡的作用。

但是普通的余数hash（hash(比如用户id)%服务器机器数）算法伸缩性很差，当新增或者下线服务器机器时候，用户id与服务器的映射关系会大量失效。一致性hash则利用hash环对其进行了改进。

二、一致性Hash概述

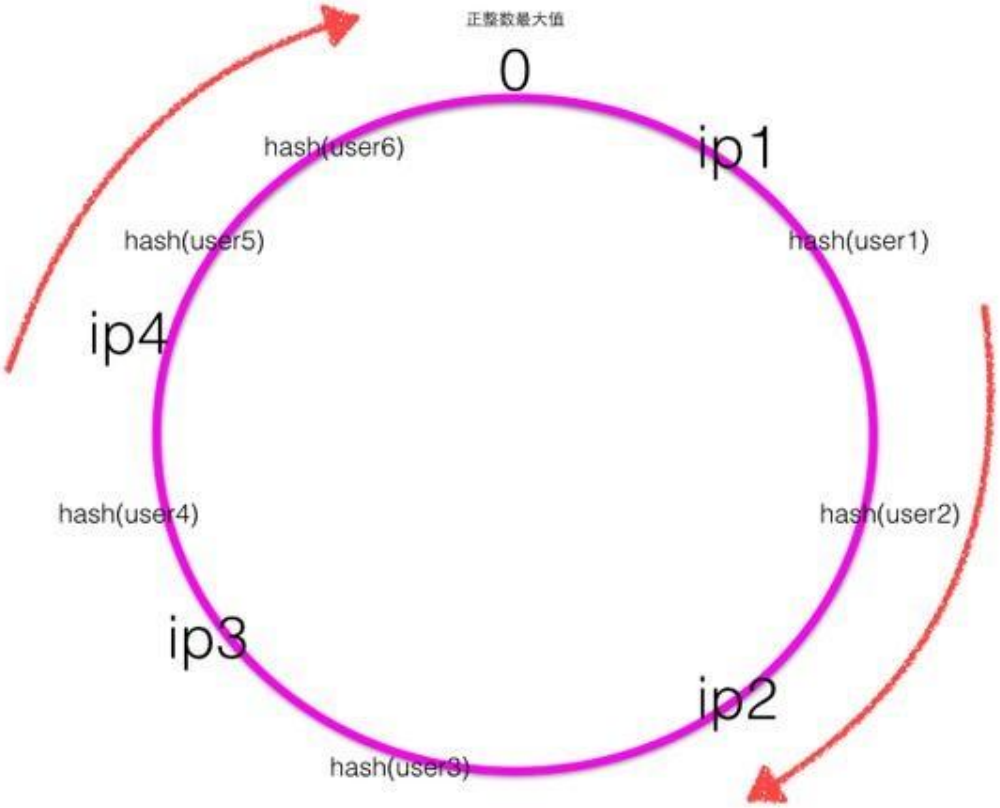
为了能直观的理解一致性hash原理，这里结合一个简单的例子来讲解，假设有4台服务器，地址为ip1,ip2,ip3,ip4。

- 一致性hash是首先计算四个ip地址对应的hash值
- hash(ip1),hash(ip2),hash(ip3),hash(ip3)，计算出来的hash值是0~最大正整数直接的一个值，这四个值在一致性hash环上呈现如下图：



- hash环上顺时针从整数0开始，一直到最大正整数，我们根据四个ip计算的hash值肯定会落到这个hash环上的某一个点，至此我们把服务器的四个ip映射到了一致性hash环

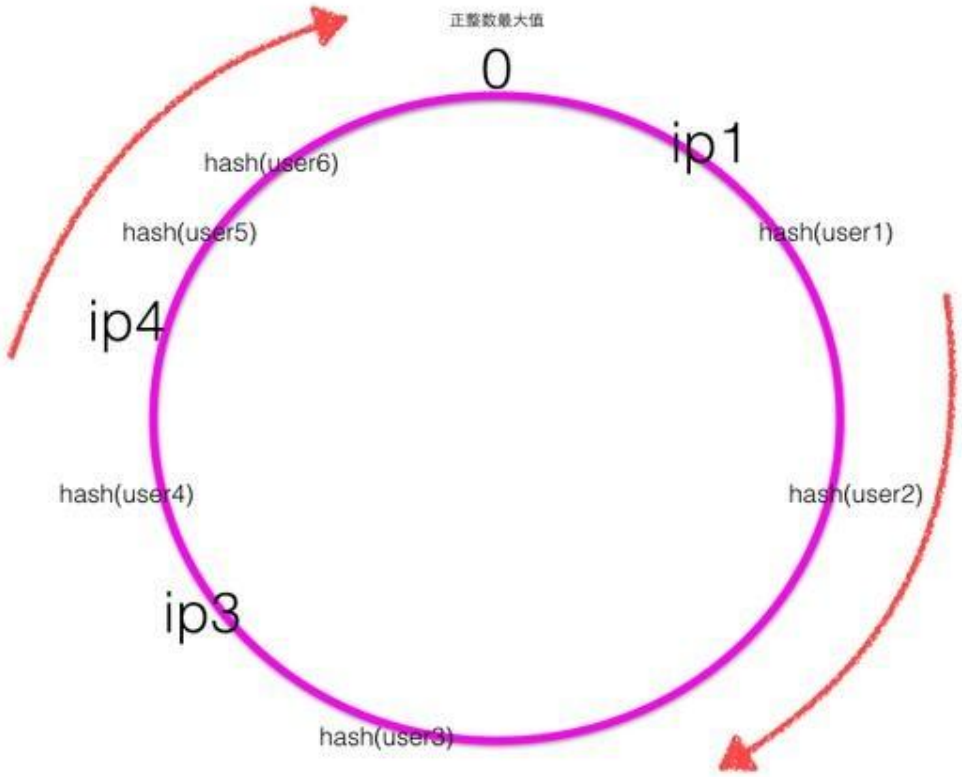
- 当用户在客户端进行请求时候，首先根据hash(用户id)计算路由规则（hash值），然后看hash值落到了hash环的那个地方，根据hash值在hash环上的位置顺时针找距离最近的ip作为路由ip.



如上图可知user1,user2的请求会落到服务器ip2进行处理，User3的请求会落到服务器ip3进行处理，user4的请求会落到服务器ip4进行处理，user5,user6的请求会落到服务器ip1进行处理。

下面考虑当ip2的服务器挂了的时候会出现什么情况?

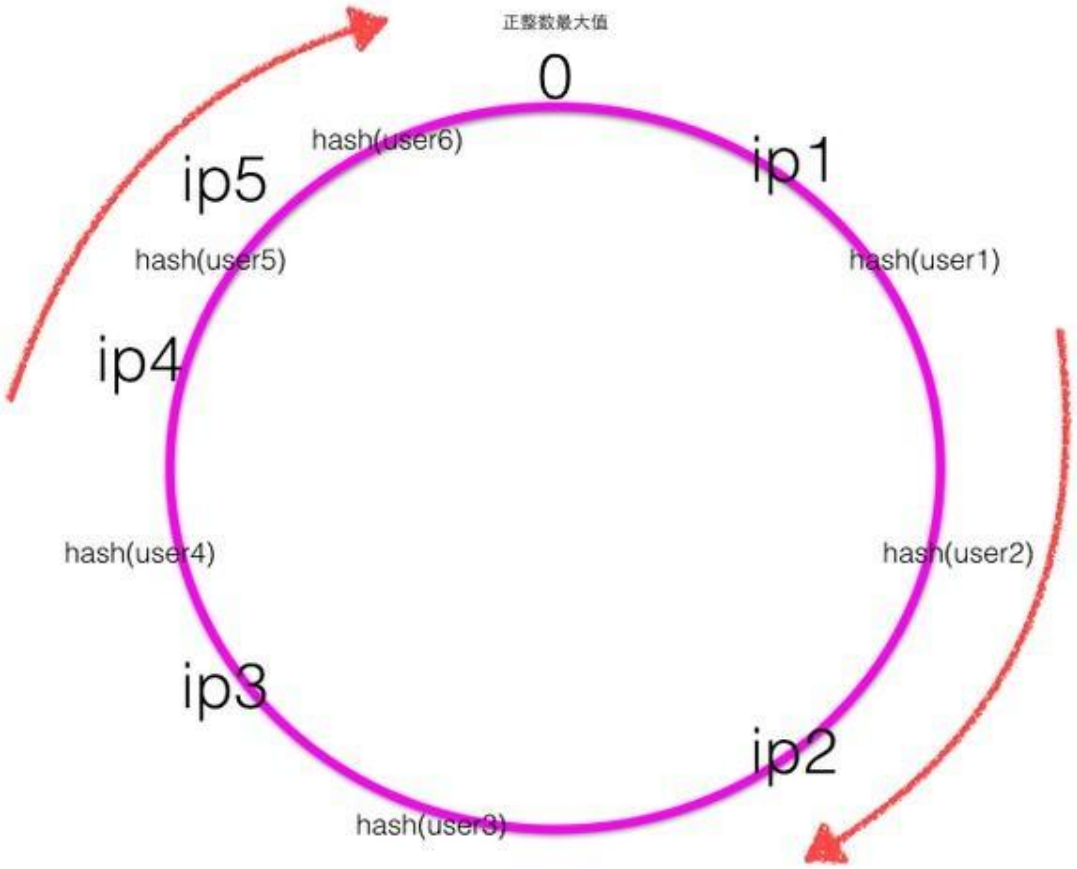
当ip2的服务器挂了的时候，一致性hash环大致如下图:



根据顺时针规则可知user1,user2的请求会被服务器ip3进行处理，而其它用户的请求对应的处理服务器不变，也就是只有之前被ip2处理的一部分用户的映射关系被破坏了，并且其负责处理的请求被顺时针下一个节点委托处理。

下面考虑当新增机器的时候会出现什么情况?

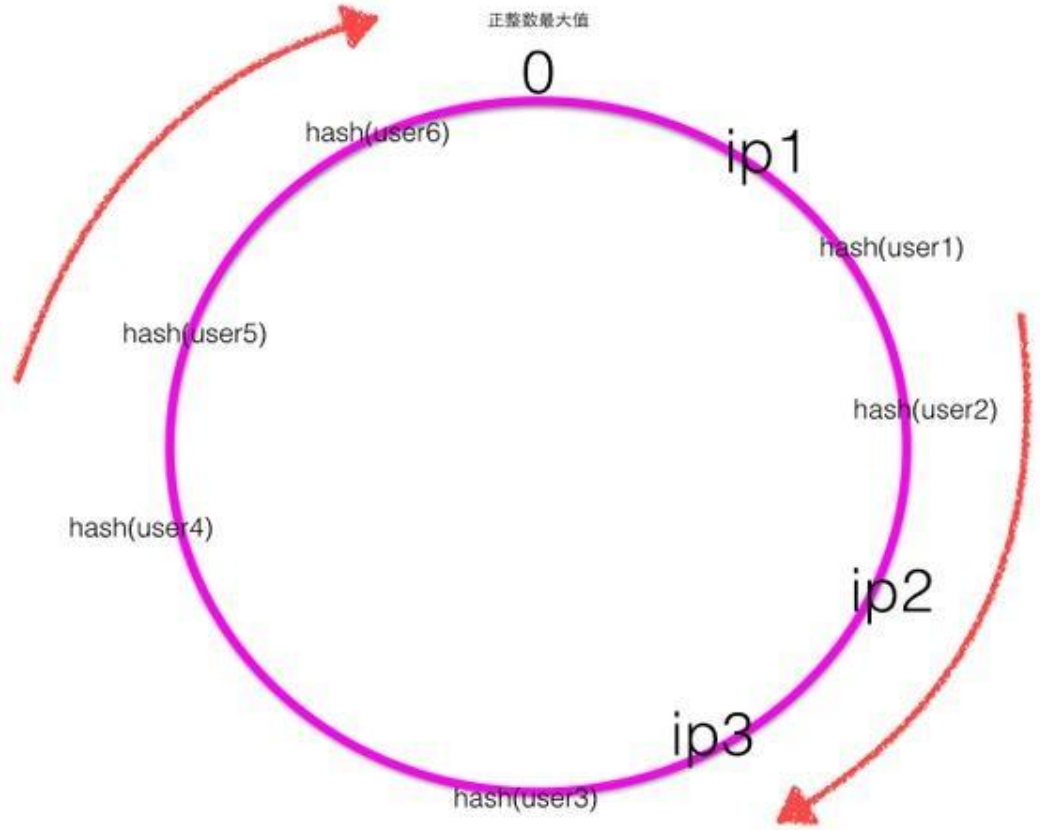
当新增一个ip5的服务器后，一致性hash环大致如下图:



根据顺时针规则可知之前user1的请求应该被ip1服务器处理，现在被新增的ip5服务器处理，其他用户的请求处理服务器不变，也就是新增的服务器顺时针最近的服务器的一部分请求会被新增的服务器所替代。

三、一致性hash的特性

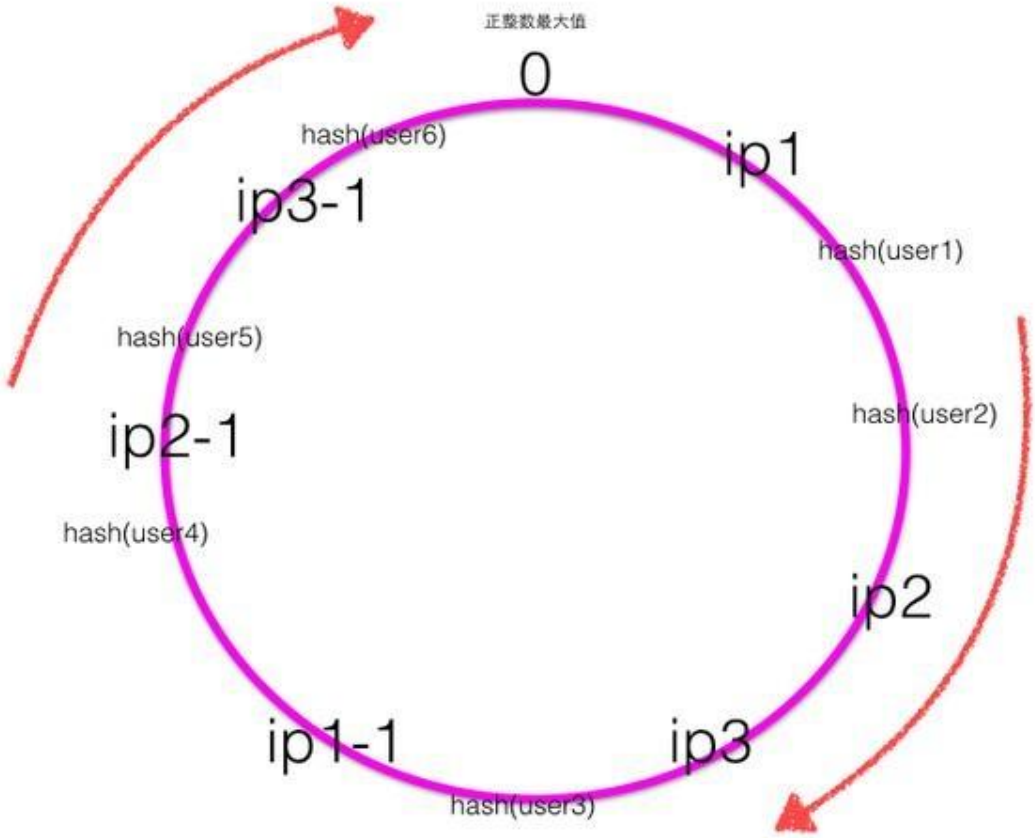
- 单调性(Monotonicity)，单调性是指如果已经有一些请求通过哈希分派到了相应的服务器进行处理，又有新的服务器加入到系统中时候，应保证原有的请求可以被映射到原有的或者新的服务器中去，而不会被映射到原来的其它服务器上去。 这个通过上面新增服务器ip5可以证明，新增ip5后，原来被ip1处理的user6现在还是被ip1处理，原来被ip1处理的user5现在被新增的ip5处理。
- 分散性(Spread)：分布式环境中，客户端请求时候可能不知道所有服务器的存在，可能只知道其中一部分服务器，在客户端看来他看到的部分服务器会形成一个完整的hash环。如果多个客户端都把部分服务器作为一个完整hash环，那么可能会导致，同一个用户的请求被路由到不同的服务器进行处理。这种情况显然是应该避免的，因为它不能保证同一个用户的请求落到同一个服务器。所谓分散性是指上述情况发生的严重程度。
- 平衡性(Balance)：平衡性也就是说负载均衡，是指客户端hash后的请求应该能够分散到不同的服务器上去。一致性hash可以做到每个服务器都进行处理请求，但是不能保证每个服务器处理的请求的数量大致相同，如下图



服务器ip1,ip2,ip3经过hash后落到了一致性hash环上，从图中hash值分布可知ip1会负责处理大概80%的请求，而ip2和ip3则只会负责处理大概20%的请求，虽然三个机器都在处理请求，但是明显每个机器的负载不均衡，这样称为一致性hash的倾斜，虚拟节点的出现就是为了解决这个问题。

五、虚拟节点

当服务器节点比较少的时候会出现上节所说的一致性hash倾斜的问题，一个解决方法是多加机器，但是加机器是有成本的，那么就加虚拟节点，比如上面三个机器，每个机器引入1个虚拟节点后的一致性hash环的图如下：

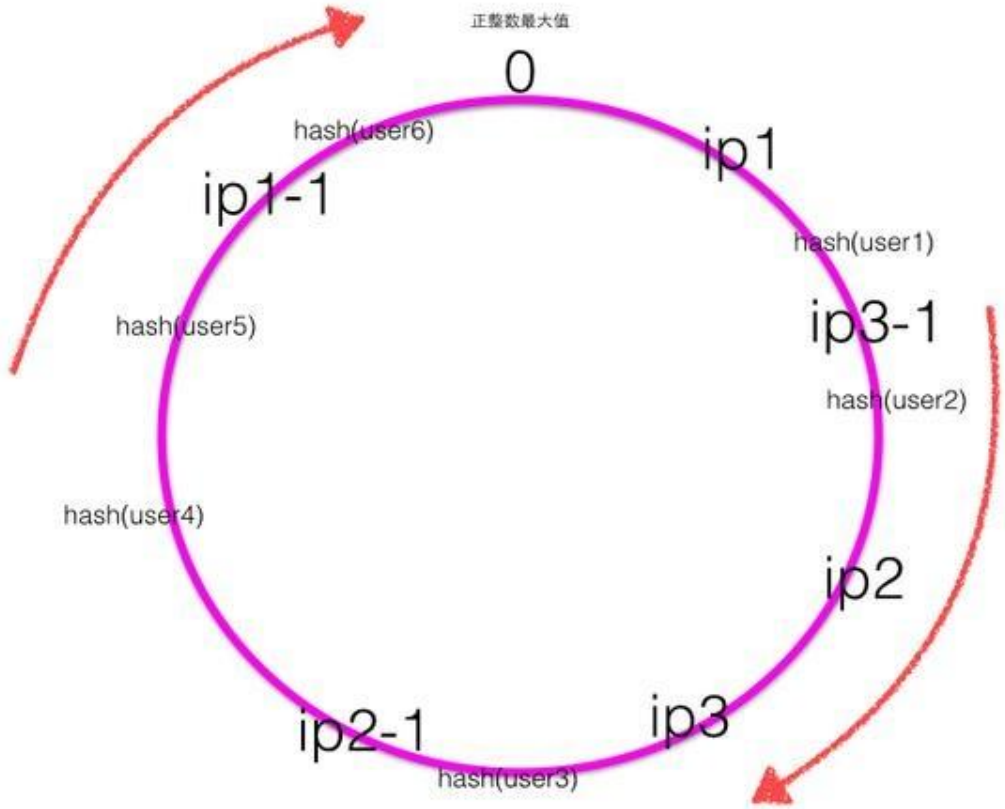


其中ip1-1是ip1的虚拟节点，ip2-1是ip2的虚拟节点，ip3-1是ip3的虚拟节点。

可知当物理机器数目为M，虚拟节点为N的时候，实际hash环上节点个数为M*（N+1）。比如当客户端计算的hash值处于ip2和ip3或者处于ip2-1和ip3-1之间时候使用ip3服务器进行处理。

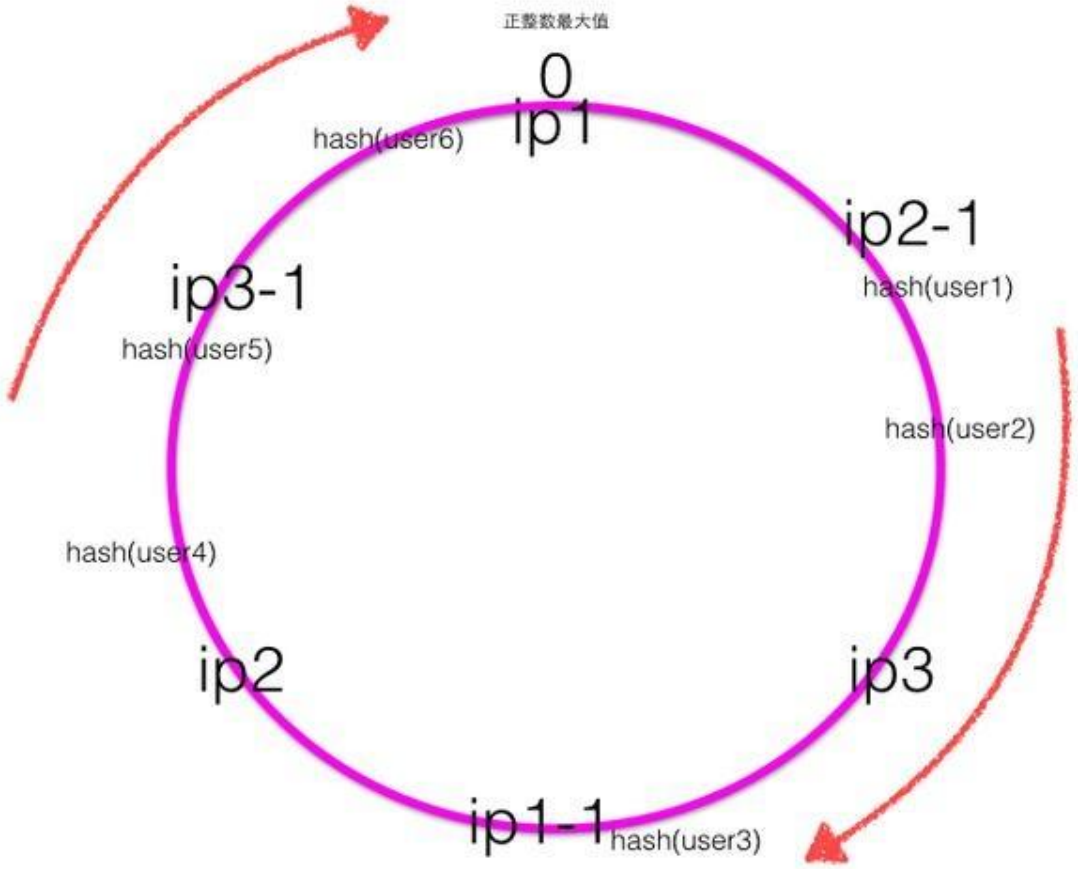
六、均匀一致性hash

上节我们使用虚拟节点后的图看起来比较均衡，但是如果生成虚拟节点的算法不够好很可能会得到下面的环：



可知每个服务节点引入1个虚拟节点后，情况相比没有引入前均衡性有所改善，但是并不均衡。

均衡的一致性hash应该是如下图：



均匀一致性hash的目标是如果服务器有N台，客户端的hash值有M个，那么每个服务器应该处理大概M/N个用户的。也就是每台服务器负载尽量均衡。dubbo提供的一致性hash负载均衡算法就是不均匀的，我们自己实现了dubbo的spi扩展实现了均匀一致性hash.

七、总结

在分布式系统中一致性hash起着不可忽略的地位，无论是分布式缓存，还是分布式Rpc框架的负载均衡策略都有所使用。