一致性哈希将整个哈希值空间组织成一个虚拟的圆环，如假设某哈希函数H的值空间为0 – 2^32-1（即哈希值是一个32位无符号整形).

整个空间按顺时针方向组织。0和2^32-1在零点中方向重合。

下一步将各个服务器使用H进行一个哈希，具体可以选择服务器的ip或主机名作为关键字进行哈希，这样每台机器就能确定其在哈希环上的位置

同时数据定位算法不变，只是多了一步虚拟节点到实际节点的映射，例如定位到“Redis-1#1”、“Redis-1#2”、“Redis-1#3”三个虚拟节点的数据均定位到Redis-1上。这样就解决了服务节点少时数据倾斜的问题。在实际应用中，通常将虚拟节点数设置为32甚至更大，因此即使很少的服务节点也能做到相对均匀的数据分布。

应用场景：

缓存集群常用该算法如memcache：

memcached在实现分布式群集部署时，memcached服务之间是不能进行通讯的，分布式也是通过客户端的算法吧数据保存在不同的memcached中，所以当我们做完群集客户端往里面写入数据时，会出现下面的情况。客户端往一个memcached节点写入数据后，另外两个节点是查询不到的，此时就需要用一致性hash算法。

首先求出memcached 服务器（节点）的哈希值，并将其配置到0～232的圆（continuum）上。然后用同样的方法求出存储数据的键的哈希值，并映射到圆上。然后从数据映射到的位置开始顺时针查找，将数据保存到找到的第一个服务器上。如果超过232 仍然找不到服务器，就会保存到第一台memcached 。

但是memcache还是有问题如：

memcached对容错不做处理的，也就是说当Memcached 服务端挂掉或者 Memcached 重启复活，由于Memcached重启后无数据，对高并发的大型系统中，是没办法接受的，我们在集群的情况下，要做到负载均衡和容错 ，这种情况下， 服务器端挂掉，负载就不那么均衡，缓存数据也会丢失 ，重新对数据库进行操作，加大的系统的性能， 数据丢失容错也没做得那么好。 Memcache自身并没有实现集群功能，如果想用Memcahce实现集群需要借助第三方软件或者自己设计编程实现，如repcached、memagent、 memcached-ha等。

Redis集群模式也使用了一致性hash算法：

(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是通过集群中超过半数的节点检测失效时才生效.

(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（插槽）上,cluster 负责维护node<->slot<->value