一致性哈希的原理是：

* 将每个节点(node)映射到数值空间，映射的规则可为 IP、hostname 等。（我用的是IP，把服务器对应IP的哈希值存到哈希环上。）
* 将每个 object 映射到数值空间。
* 对于某个 object，对于所有满足 hash(node) <= hash(object) 的节点，选择 hash(node) 最大的节点存放 object。如果没有满足上述条件的节点，选择 hash(node) 最小的节点存放该 object。

哈希环是有序的，当一个hashcoad进来的时候，需要找大于这个hashcoad的第一个虚拟节点。

Hashcoad存储的时候是有序的，可以往里面任意的放元素但是还可以保证顺序。我用的eclipse使用TreeMap实现。

节点的位置是由自身哈希值决定的，它们的分布并非均匀，特别当节点数目很少时，容易造成 object 的分布不均匀，即平衡性低。为了解决平衡性低的问题，引入了虚拟节点，虚拟节点实际上是物理节点的复制品，一个物理节点包含多个虚拟节点，我们将这些虚拟节点映射到数值空间，对于某个 object，我们计算该 object 存放的虚拟节点，进而得出物理节点。当虚拟节点越多，虚拟节点的位置分布越均匀，相应的，映射到物理节点的 object 数目也越均匀，提高了平衡性。

使用java实现：



