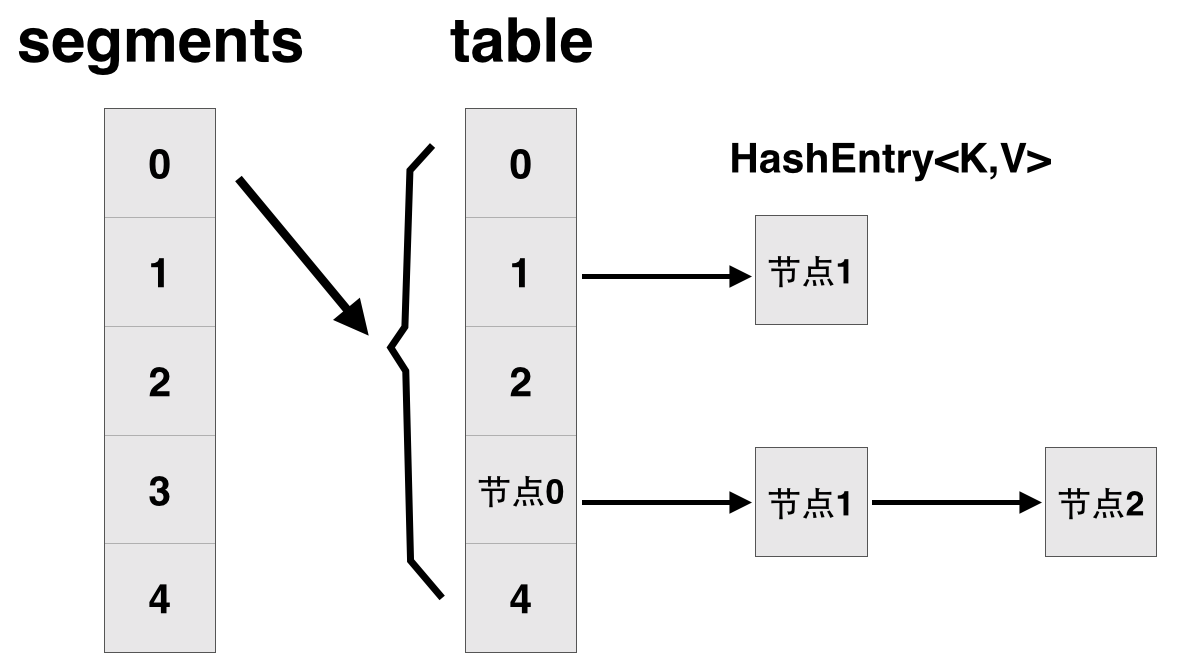
并发就是要解决三个问题：安全性，活跃性，性能。

安全性是根本，没有安全性后面都没有意义。不过在解决安全性的时候要同时考虑活跃性或者性能的问题。

解决安全性的方法主要是加锁，但是加锁有讲究的，有理有据有节。在java中有很多并发容器就是我们学习的榜样。首先就是Map。我们都知道HashTable是线程安全的，但是看源码就是发现，synchronized关键字锁住了整个对象，那么高并发操作性能肯定不好，所有就有了ConcurrentHashMap.

Jdk1.7的设计：16个segments



但是在jdk1.8竟然优化了很多，重点使用CAS。

**、在jdk1.8中主要做了2方面的改进**

**改进一：取消segments字段，直接采用transient volatile HashEntry<K,V>[] table保存数据，采用table数组元素作为锁，从而实现了对每一行数据进行加锁，进一步减少并发冲突的概率。**

**改进二：将原先table数组＋单向链表的数据结构，变更为table数组＋单向链表＋红黑树的结构。对于hash表来说，最核心的能力在于将key hash之后能均匀的分布在数组中。如果hash之后散列的很均匀，那么table数组中的每个队列长度主要为0或者1。但实际情况并非总是如此理想，虽然ConcurrentHashMap类默认的加载因子为0.75，但是在数据量过大或者运气不佳的情况下，还是会存在一些队列长度过长的情况，如果还是采用单向列表方式，那么查询某个节点的时间复杂度为O(n)；因此，对于个数超过8(默认值)的列表，jdk1.8中采用了红黑树的结构，那么查询的时间复杂度可以降低到O(logN)，可以改进性能。**

<https://www.cnblogs.com/chengxiao/p/6842045.html>

<https://my.oschina.net/hosee/blog/618953>

<http://blog.csdn.net/u010723709/article/details/48007881>

http://blog.csdn.net/wangxiaotongfan/article/details/52074160