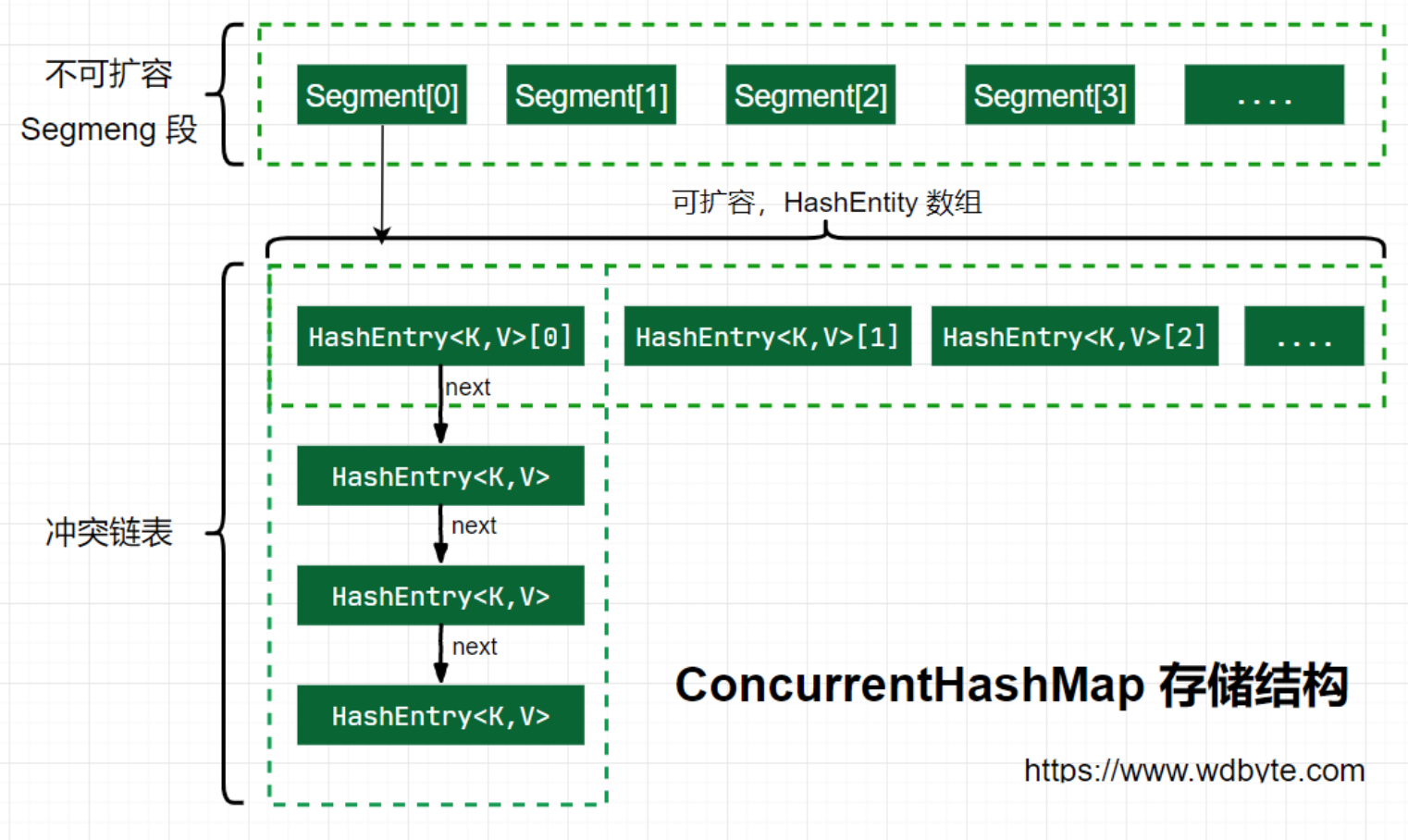
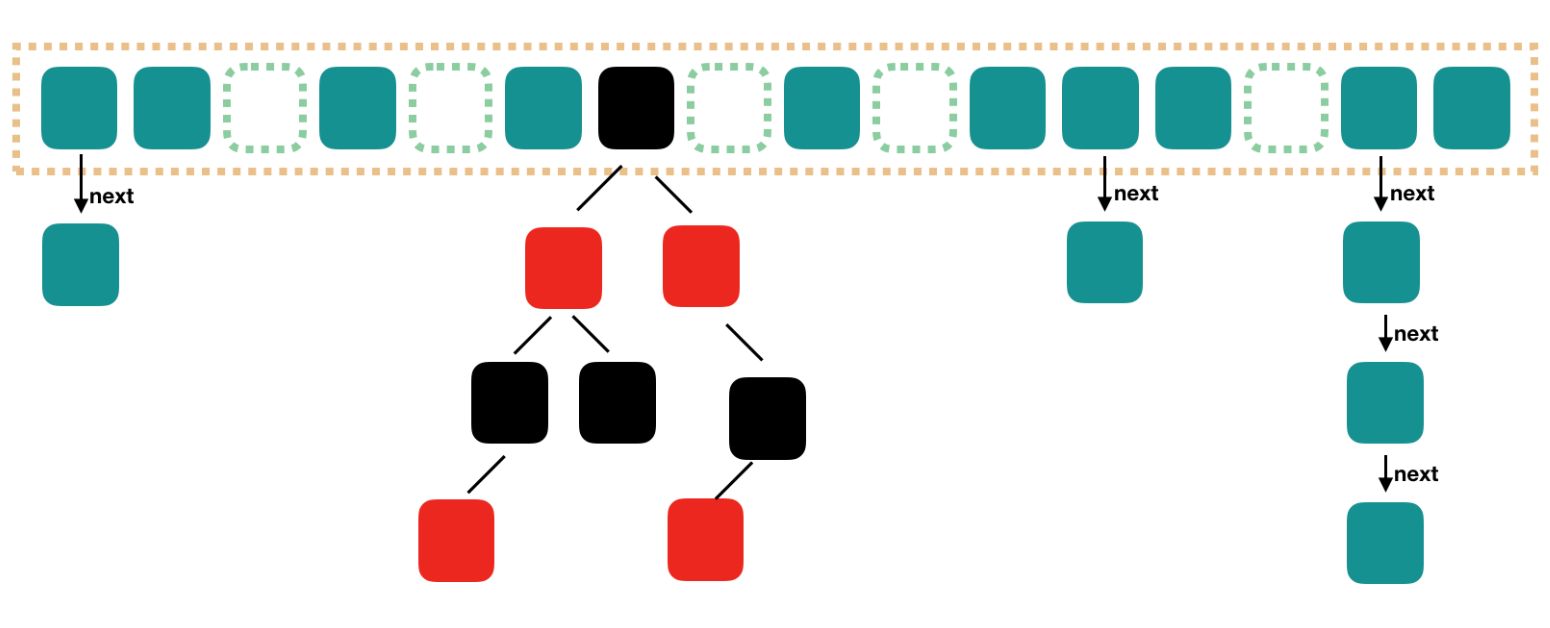
1. 数据结构

1、ConcurrentHashMap 1.7



Java 7 中 ConcurrentHashMap 的存储结构如上图，ConcurrnetHashMap 由很多个 Segment 组合，而每一个 Segment 是一个类似于 HashMap 的结构，所以每一个 HashMap 的内部可以进行扩容。但是 Segment 的个数一旦初始化就不能改变，默认 Segment 的个数是 16 个，你也可以认为 ConcurrentHashMap 默认支持最多 16 个线程并发。

2、ConcurrentHashMap 1.8



Java8 的 ConcurrentHashMap 相对于 Java7 来说变化比较大，不再是之前的 Segment 数组 + HashEntry 数组 + 链表，而是 Node 数组 + 链表 / 红黑树。当冲突链表达到一定长度时，链表会转换成红黑树。

1. 构造函数

1、ConcurrentHashMap 1.7

无参构造中调用了有参构造，传入了三个参数的默认值，他们的值是：

/\*\*

\* 默认初始化容量

\*/

static final int DEFAULT\_INITIAL\_CAPACITY = 16;

/\*\*

\* 默认负载因子

\*/

static final float DEFAULT\_LOAD\_FACTOR = 0.75f;

/\*\*

\* 默认并发级别

\*/

static final int DEFAULT\_CONCURRENCY\_LEVEL = 16;

1.必要参数校验。

2.校验并发级别 concurrencyLevel 大小，如果大于最大值，重置为最大值。无惨构造默认值是 16.

3.寻找并发级别 concurrencyLevel 之上最近的 2 的幂次方值，作为初始化容量大小，默认是 16。

4.记录 segmentShift 偏移量，这个值为【容量 = 2 的 N 次方】中的 N，在后面 Put 时计算位置时会用到。默认是 32 - sshift = 28.

5.记录 segmentMask，默认是 ssize - 1 = 16 -1 = 15.

6.初始化 segments [0]，默认大小为 2，负载因子 0.75，扩容阀值是 2\*0.75=1.5，插入第二个值时才会进行扩容。

2、ConcurrentHashMap 1.8

ConcurrentHashMap 的初始化是通过自旋和 CAS 操作完成的。里面需要注意的是变量 sizeCtl ，它的值决定着当前的初始化状态。

1.-1 说明正在初始化

2.-N 说明有 N-1 个线程正在进行扩容

3.表示 table 初始化大小，如果 table 没有初始化

4.表示 table 容量，如果 table　已经初始化。

二、Put方法

ConcurrentHashMap 1.7：

1、计算要 put 的 key 的位置，获取指定位置的 Segment。

2、如果指定位置的 Segment 为空，则初始化这个 Segment.

初始化 Segment 流程：

1.检查计算得到的位置的 Segment 是否为 null.

2.为 null 继续初始化，使用 Segment [0] 的容量和负载因子创建一个 HashEntry 数组。

3.再次检查计算得到的指定位置的 Segment 是否为 null.

4.使用创建的 HashEntry 数组初始化这个 Segment.

5.自旋判断计算得到的指定位置的 Segment 是否为 null，使用 CAS 在这个位置赋值为 Segment.

3、Segment.put 插入 key,value 值。

0.由于 Segment 继承了 ReentrantLock，所以 Segment 内部可以很方便的获取锁，put 流程就用到了这个功能。

1.tryLock () 获取锁，获取不到使用 scanAndLockForPut 方法继续获取。

2.计算 put 的数据要放入的 index 位置，然后获取这个位置上的 HashEntry 。

3.遍历 put 新元素，为什么要遍历？因为这里获取的 HashEntry 可能是一个空元素，也可能是链表已存在，所以要区别对待。

1.如果这个位置上的 HashEntry 不存在：

1.如果当前容量大于扩容阀值，小于最大容量，进行扩容。

2.直接头插法插入。

2.如果这个位置上的 HashEntry 存在：

1.判断链表当前元素 Key 和 hash 值是否和要 put 的 key 和 hash 值一致。一致则替换值

2.不一致，获取链表下一个节点，直到发现相同进行值替换，或者链表表里完毕没有相同的。

1.如果当前容量大于扩容阀值，小于最大容量，进行扩容。

2.直接链表头插法插入。

4.如果要插入的位置之前已经存在，替换后返回旧值，否则返回 null.

这里面的第一步中的 scanAndLockForPut 操作这里没有介绍，这个方法做的操作就是不断的自旋 tryLock() 获取锁。当自旋次数大于指定次数时，使用 lock() 阻塞获取锁。在自旋时顺表获取下 hash 位置的 HashEntry。

ConcurrentHashMap 1.8

1、根据 key 计算出 hashcode 。

2、判断是否需要进行初始化。

3、即为当前 key 定位出的 Node，如果为空表示当前位置可以写入数据，利用 CAS 尝试写入，失败则自旋保证成功。

4、如果当前位置的 hashcode == MOVED == -1, 则需要进行扩容。

5、如果都不满足，则利用 synchronized 锁写入数据。

6、如果数量大于 TREEIFY\_THRESHOLD 则要转换为红黑树。

1. 扩容

ConcurrentHashMap 的扩容只会扩容到原来的两倍。老数组里的数据移动到新的数组时，位置要么不变，要么变为 index+ oldSize，参数里的 node 会在扩容之后使用链表头插法插入到指定位置。

1. Get方法

1、计算得到 key 的存放位置。

2、遍历指定位置查找相同 key 的 value 值。