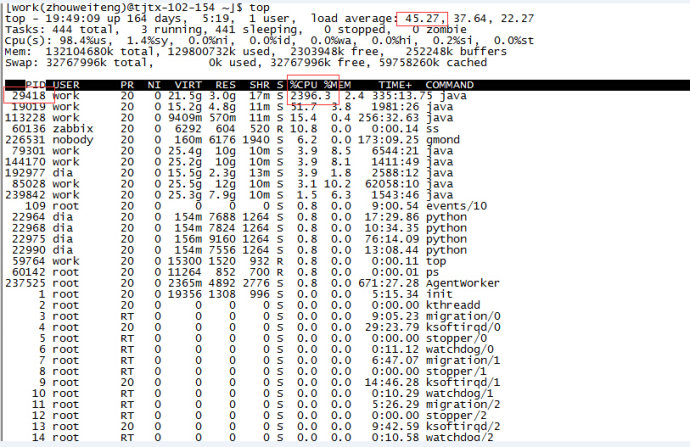
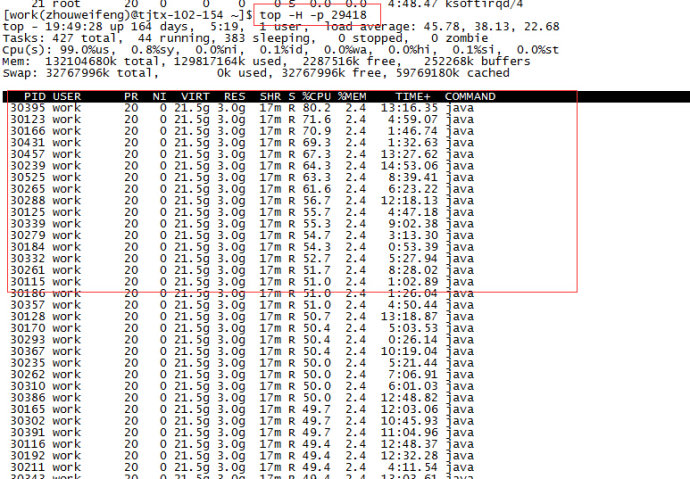
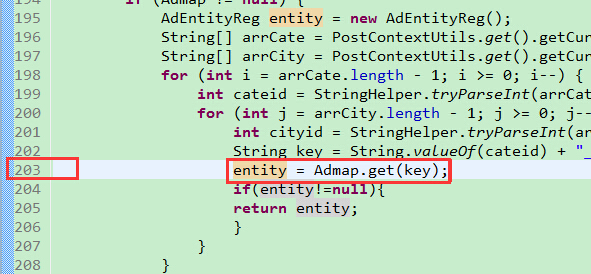
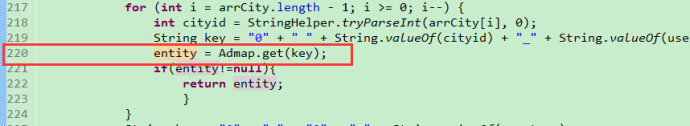
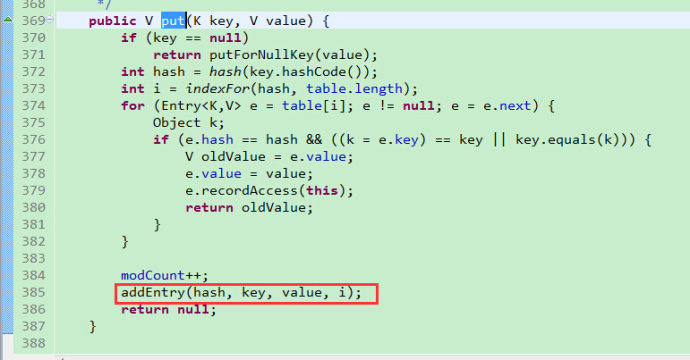
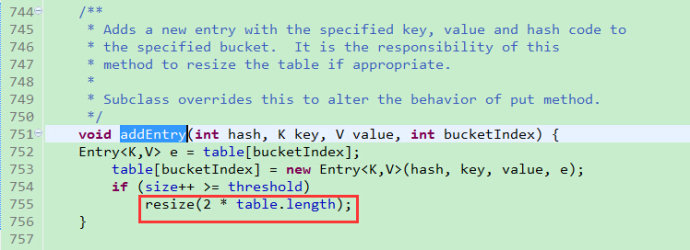
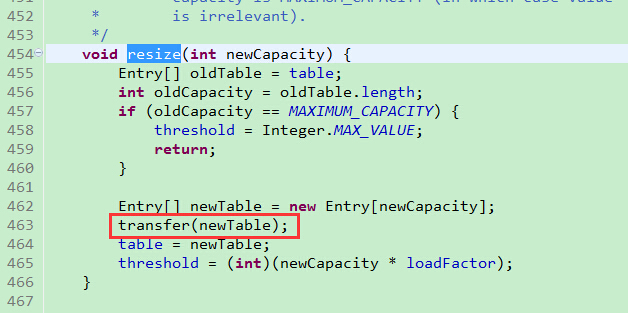
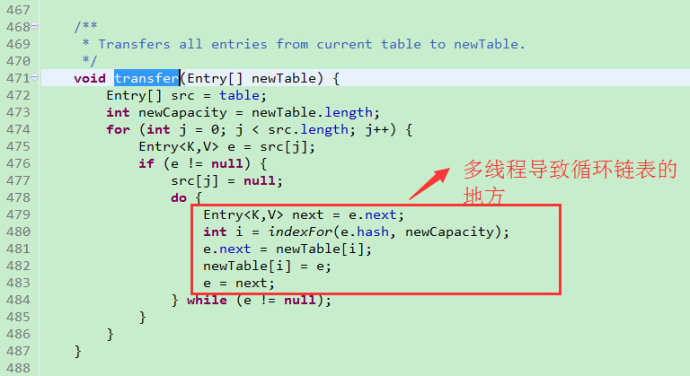
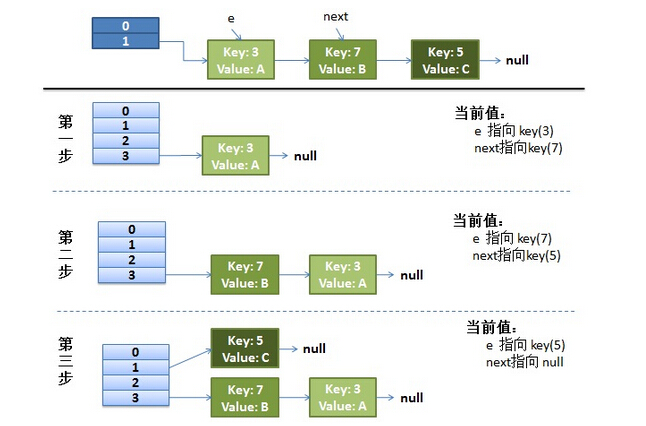
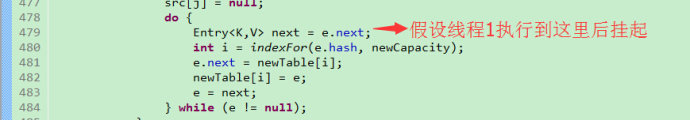
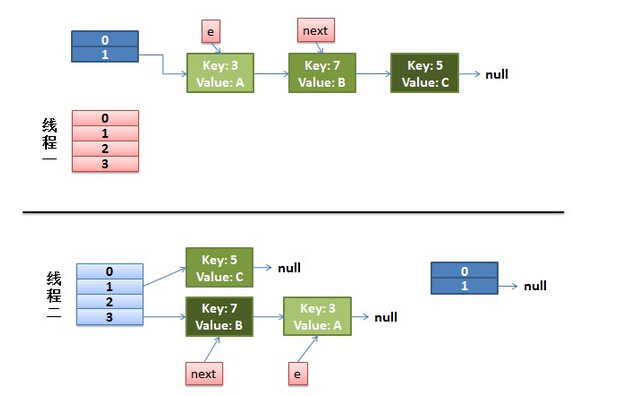
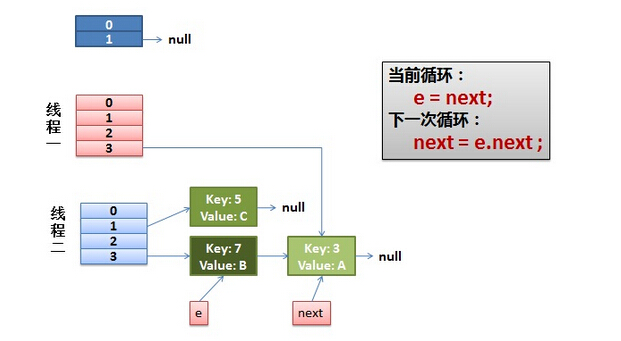
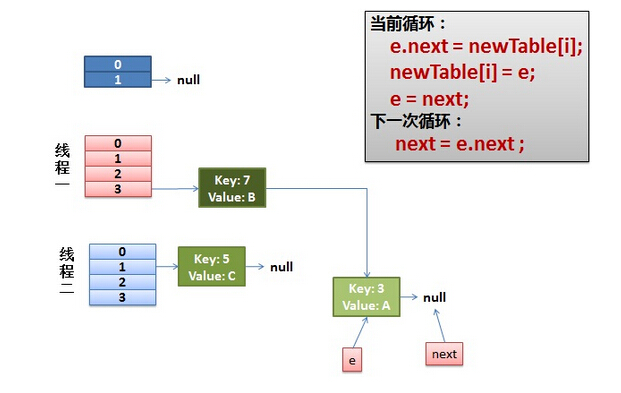
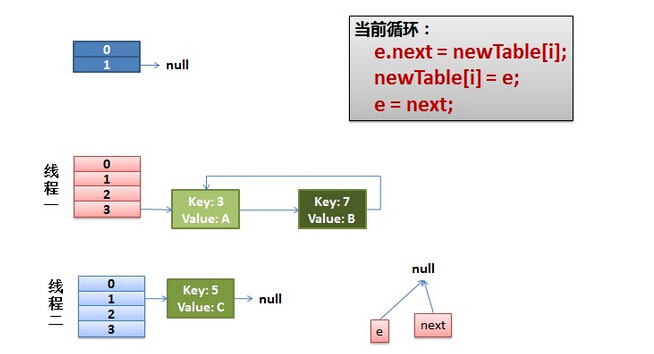
前面文章：<http://blog.sina.com.cn/s/blog_818367a30102wgal.html>也介绍过此类的问题，这次某互联网公  
  
司的发布系统又出现类似的问题。下面再进行一下分析  
  
**一、负载达到45.27，cpu占用超过2000，见图  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://album.sina.com.cn/pic/002n39qrzy70iqfLMuxdd)  
  
二、查看进程29418各线程占用cpu情况  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://s1.sinaimg.cn/orignal/002n39qrcc0fe59e595d0MhUae)  
  
3、jstack** **29418 > bbbb.txt**分析第二步图中，占用cpu最高和第二高的线程，首先把对应的pid转为十六进制（nid）  
  
30395->0x76bb，30123->0x75ab，下面看看这两个线程对应的堆栈信息，见下图：  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://album.sina.com.cn/pic/002n39qrzy70iqPs7ke92)  
  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://album.sina.com.cn/pic/002n39qrzy70iqYkelyd0)  
  
**4、代码位置定位后，下面看代码，见下图：[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://album.sina.com.cn/pic/002n39qrzy70iraCTTld3)  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://album.sina.com.cn/pic/002n39qrzy70iraDeaC52)  
  
注：本身get是没有问题的，多个线程get获取数据不会出现问题，如果Admap中存放的数据结构形成环形链表，就是导致死循环  
  
5、那Admap是在哪初始化的呢，见下图：  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://album.sina.com.cn/pic/002n39qrzy70irHJrEVbd)  
  
6、hashmap.get(key)出现死循环的原因剖析，看一下hashmap的源码，见下的图：  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://s10.sinaimg.cn/orignal/002n39qr2fdb5577841a9YbU52)  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://s12.sinaimg.cn/orignal/002n39qr4762fd3aa71ab3iN11)  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://s14.sinaimg.cn/orignal/002n39qrd0208611a28edN7cba)  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://s16.sinaimg.cn/orignal/002n39qrzy70is3y0471f)**transfer的作用：把旧Entry数组中的数据转移到新扩容的Entry数组里面  
  
**7、多线程访问图解造成循环链表  
  
首先看看单线程的扩容以及rehash过程**  
  
    假设hash算法是：key%length（key对数组长度求余）  
  
    其中entry数组长度为2，key分别为3、5、7  
   
    看下图是扩容2倍（也就是数组长度为4）及rehash的过程  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://album.sina.com.cn/pic/002n39qrzy70isGWldR7f)  
  
**再看看多线程下的扩容以及rehash过程**  
1）假设我们有两个线程，我们再看看transfer中的细节 **[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://s5.sinaimg.cn/orignal/002n39qr3f22b330823f4EQea6)**  
而我们的线程2执行完成了。于是我们有下面的样子[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://album.sina.com.cn/pic/002n39qrzy70itkCc0j33)  
  
  
 注意，因为Thread1的 e 指向了key(3)，而next指向了key(7)，其在线程二rehash后，指向了线程二重组后的链表。我们可以看到链表的顺序被反转后。  
  
2）线程一被调度回来执行。  
  
    先是执行 newTalbe[i] = e;  
    然后是e = next，导致了e指向了key(7)，  
    而下一次循环的next = e.next导致了next指向了key(3)**[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://album.sina.com.cn/pic/002n39qrzy70itsGWO256)**

3）线程一接着工作。把key(7)摘下来，放到newTable[i]的第一个，然后把e和next往下移

**[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://album.sina.com.cn/pic/002n39qrzy70itsOvDe0c)**

4）环形链接出现。

e.next = newTable[i] 导致  key(3).next 指向了 key(7)

注意：此时的key(7).next 已经指向了key(3)， 环形链表就这样出现了 **[](http://photo.blog.sina.com.cn/showpic.html#blogid=818367a30102wgkc&url=http://album.sina.com.cn/pic/002n39qrzy70itsS90Q48)**  
于是，当我们的线程一调用到，HashTable.get(11)时，悲剧就出现了——Infinite Loop。

有人把这个问题报给了Sun，不过Sun不认为这个是一个问题。因为HashMap本来就不支持并发。要并发就用ConcurrentHashmap

<http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6423457>  
  
我在这里把这个事情记录下来，只是为了让大家了解并体会一下并发环境下的危险。  
  
参考：<http://mailinator.blogspot.com/2009/06/beautiful-race-condition.html>

**8、解决方案**

synchronized

  concurrenthashmap

  lock等等均可