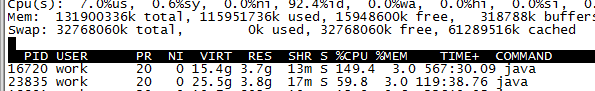
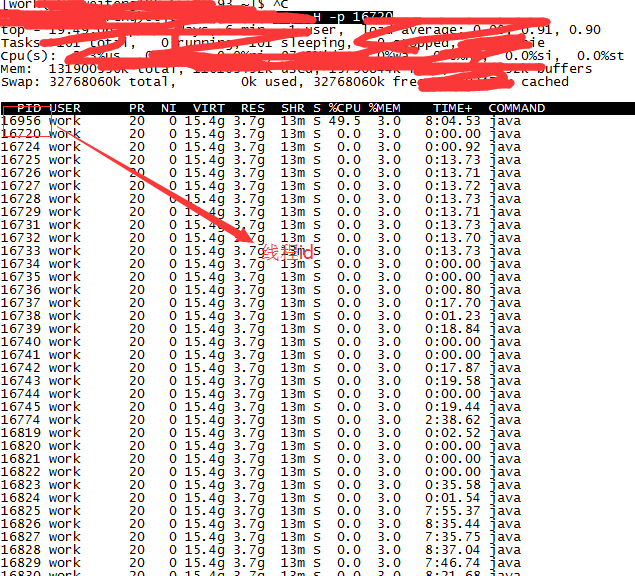
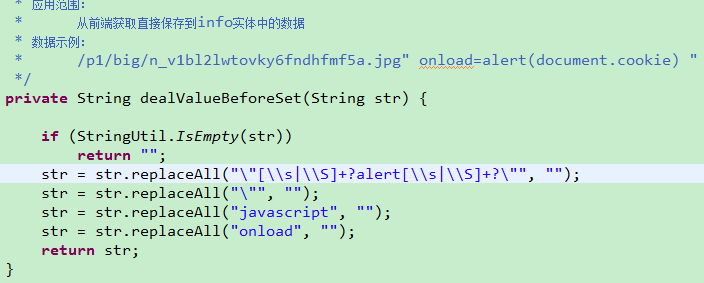
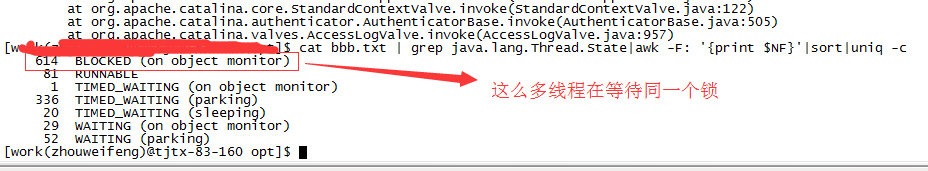
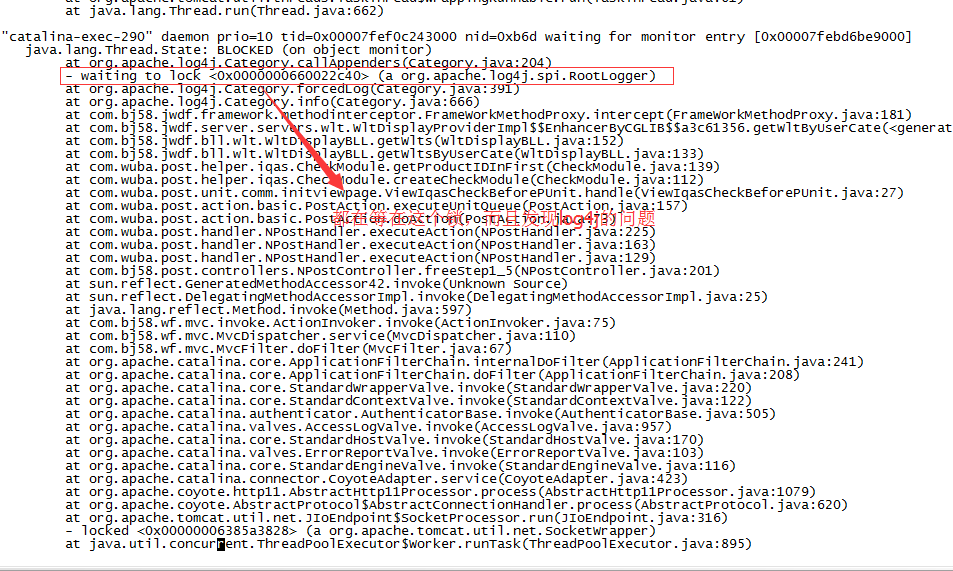
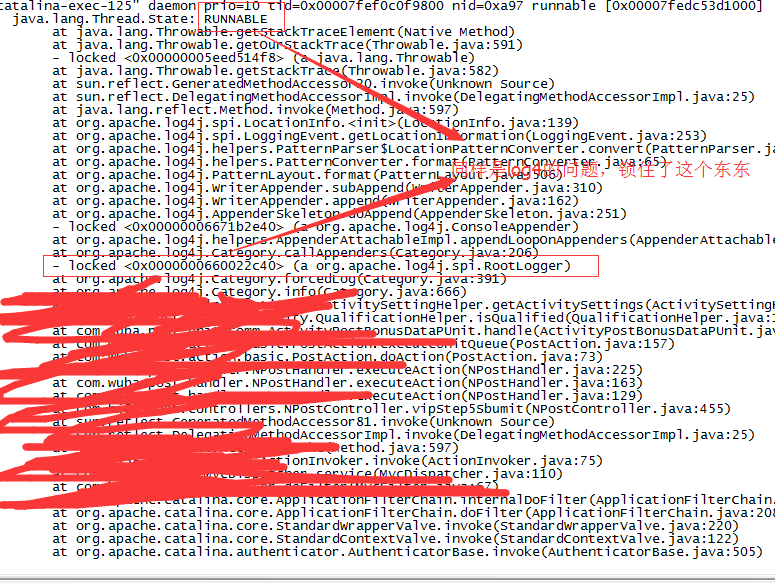
总体思路：保留现场->重启（临时解决）->分析现场->出最终解决方案->上线观察  
  
下面通过一个在工作中碰到的一个问题，进行举例：  
  
1、登录服务器，输入top命令，见下图：  
  
比如图中进程id为“16720”的为问题进程（在我解决这个问题的时候，cpu占到700%了，此图不是这个问题的  
  
真实的图，处理问题的时候忘了截图了）  
  
2、输入命令：top -H -p 16720，见下图（图不是出现问题的图）  
  
  
  
图中显示的是16720进程对应的所有线程（假设此处pid为2928的线程占用cpu很高，超过100%）  
  
3、把2928转为16进制，得到nid为B70（oxb70）  
  
4、输入命令：jstack 16720 > bbb.txt（16720为上面提到的进程id），见下图（此图为真实问题的图）  
  
  
  
5、找到dealValueBeforeSet对应的代码，见下图：  
  
  
  
代码显示，是因为正则替换的时候，导致cpu占用太高。如果有人在攻击你的系统，这时会从浏览器页面上传入  
  
一个非常长的str（代码中的参数），而此处并没有对长度做限制，导致正则计算特别占cpu。但是这个问题不会  
  
直接导致cpu飚到700%，见下面第六步  
  
6、通过分析jstack导出的bbb.txt文件，发现有600多个线程都在等待一个锁，见下图：  
  
  
  
  
此处分析用到的命令为：cat bbb.txt | grep java.lang.Thread.State|awk -F: '{print $NF}'|sort|uniq -c  
  
7、继续分析jstack导出的bbb.txt文件，发现是log4j的问题，见下图：  
  
  
  
  
  
两幅图中锁住与等待的是同一个东西：0x0000000660022c40  
  
8、最终解决方案：  
  
        修改dealValueBeforeSet的代码，增加字符长度限制为1500  
        升级log4j到log4j2  
        梳理日志减少日志的打印，业务异常不打印堆栈信息，只打印一条可以看得的信息  
        检查日志级别是否为info，减少不必要的info级别的日志  
  
9、上线观察，问题得已解决