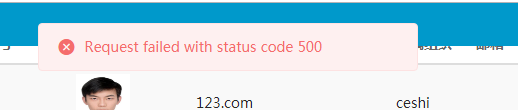
1. 问题:

上传7.5W人的excel之后 ，会出现下面的情况：

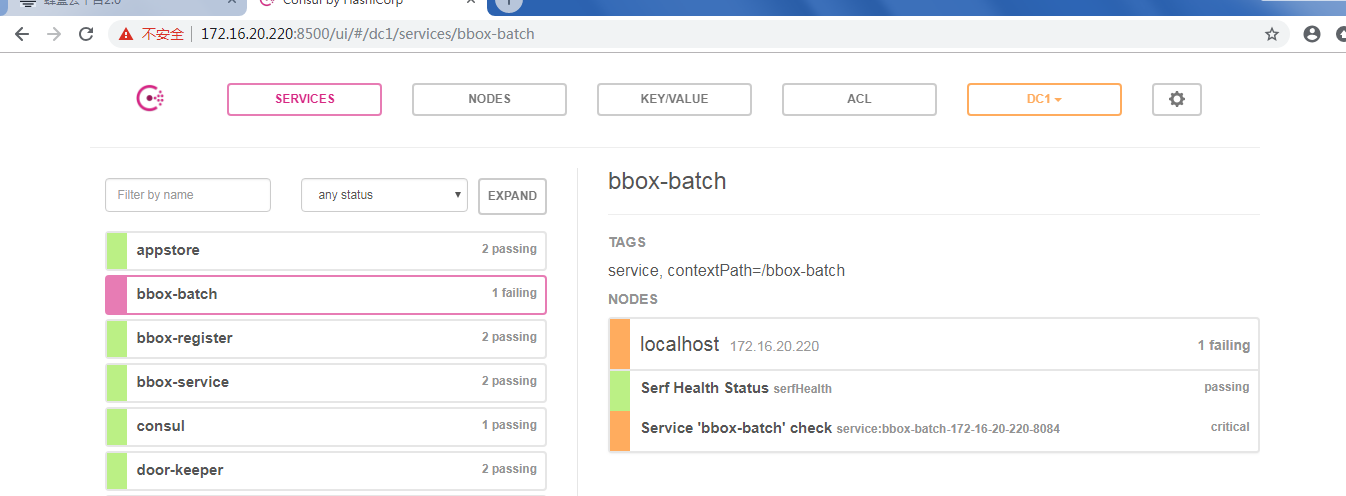




二、排查过程

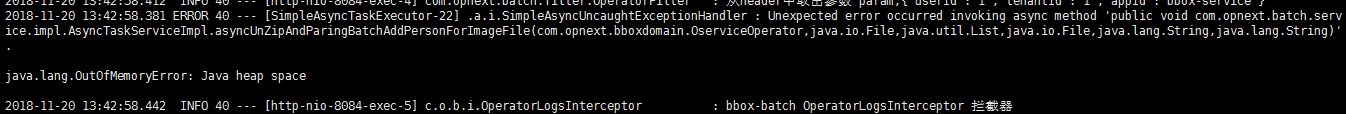
1 查看consul:

发现服务下线了，失去心跳：

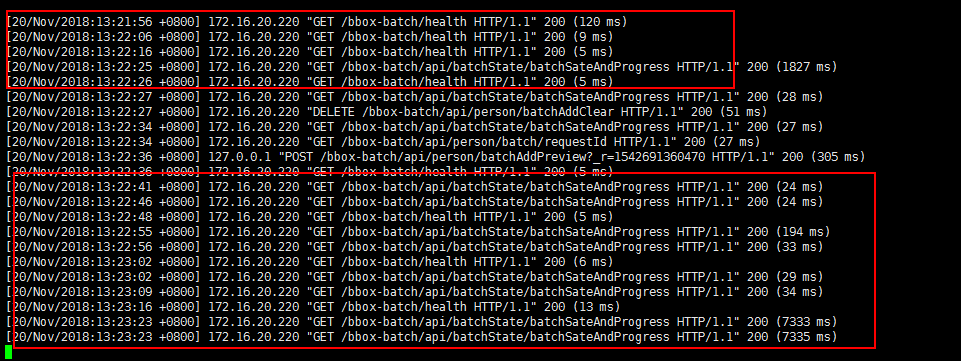


2 查看日志：

Info日志显示服务还正常活着，但是有报错：



Access日志显示，consul 的health check被积压，不能正常返回



结论：

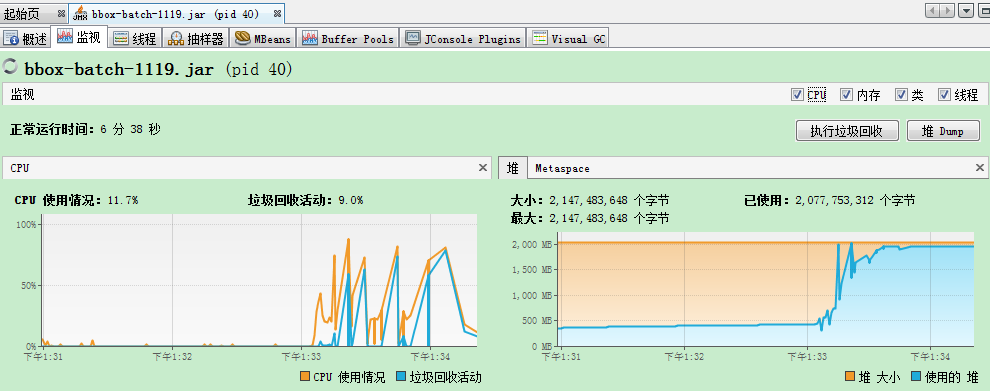
通过此处确定：因为服务负载过高，无法及时响应health接口，被consul认为失去心跳，服务下线。

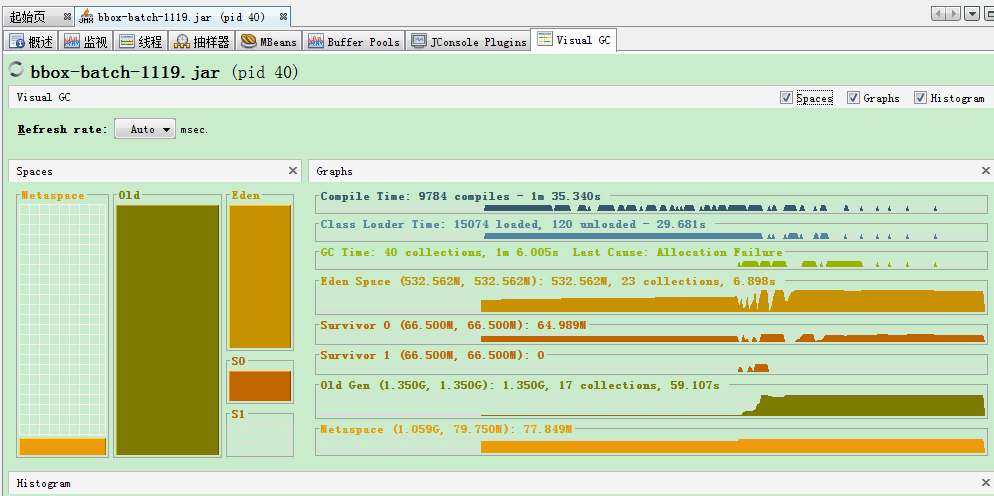
同时由于前端 ->nginx->网关->batch ，网关从consul找不到可用的batch服务，从而给返回500。

接下来就是需要 找出batch 服务为什么会负载过高，不能响应心跳。

3 通过jvirsualVM查看jvm情况

堆已经被全部使用，老年代产生大数据



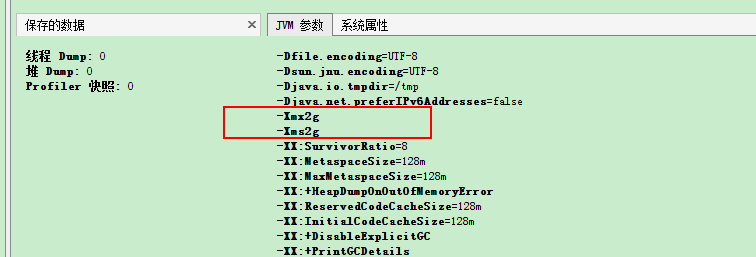


结论

堆空间不够，而内存溢出，导致进程僵死，而不能正常响应heatbeat中的consul 的healthcheck检查失败。

此时猜测是否是堆大小设置不合理：

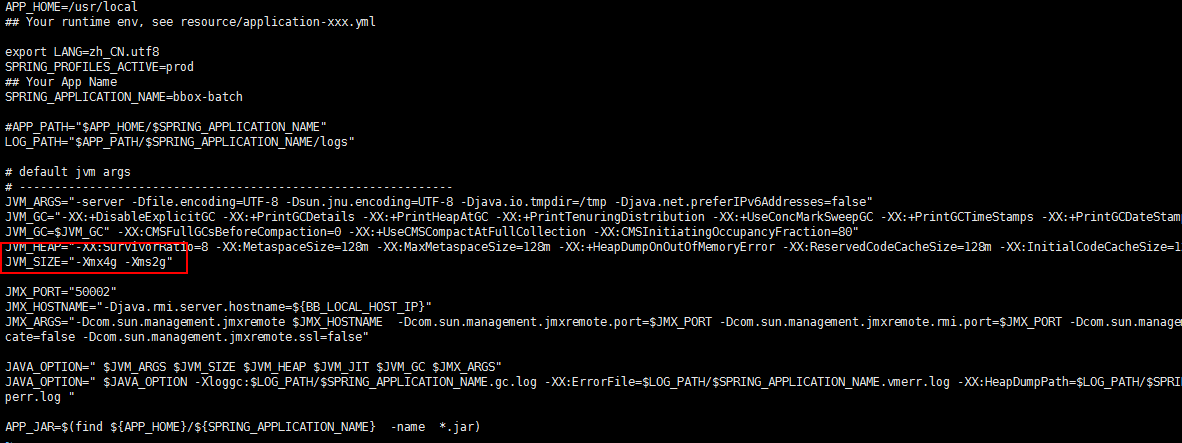
查看jvm参数：



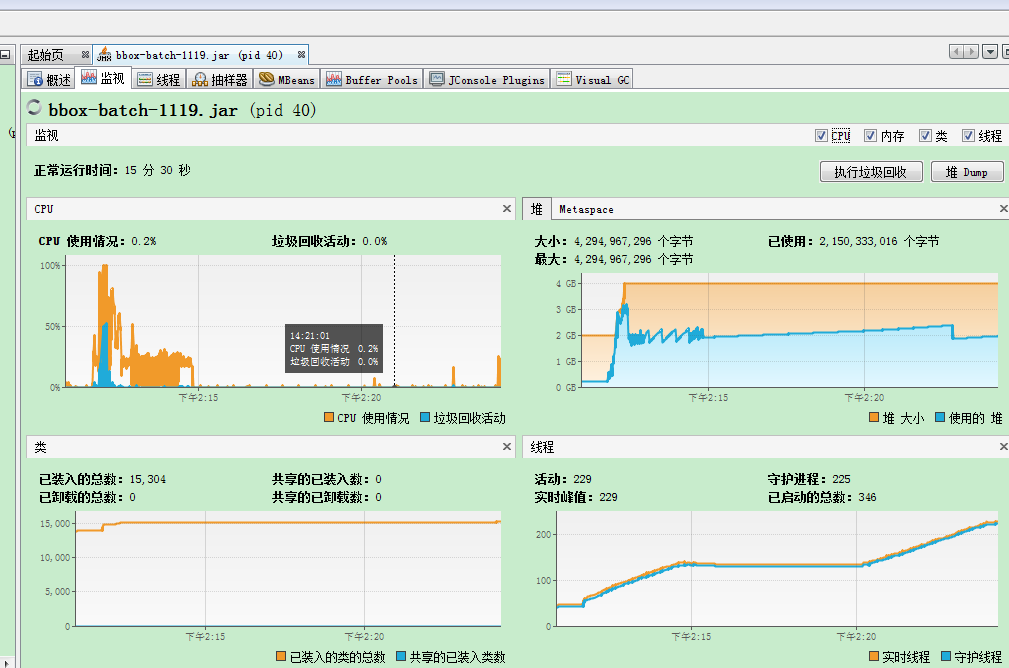
设置的是2G， 此时情况显示，2G已经被使用完，明显不够，是否考虑增加堆设置

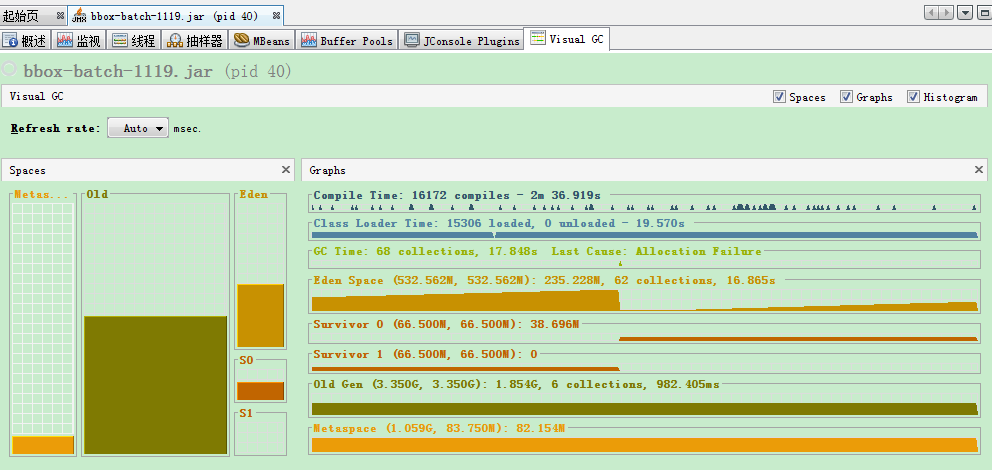
测试：设置堆为：2G~4G， 将最大值设置为4G



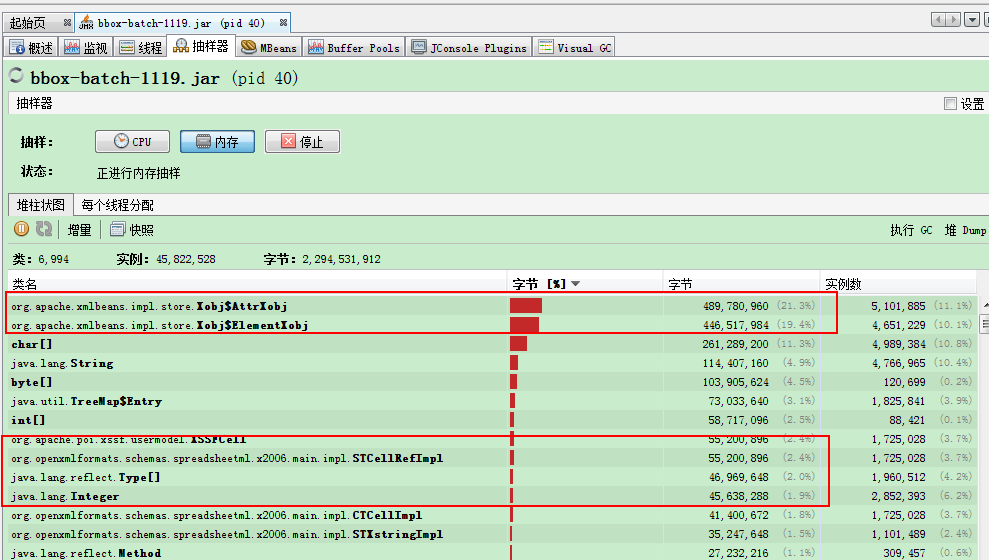


结果：再次功能正常，查看jvm情况如下 ：





4、接下来查看：此时内存中，是哪些对象占用太多的空间：



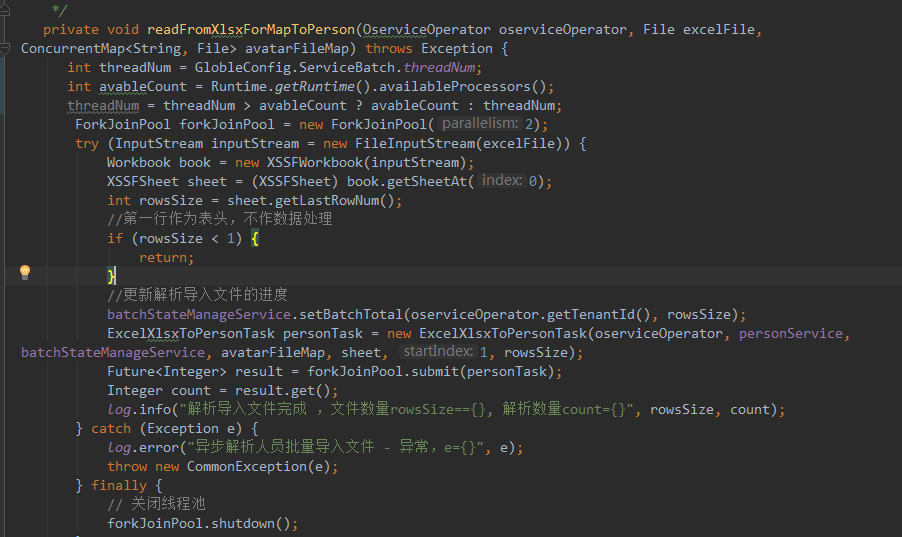
发现是excel解析相关的类，占用了太多的内存，而没有被释放。

此时定位到是解析相关的问题。

此时去检测代码 ，

1. 是否是有对象没有及时释放
2. 或者是其他什么原因

查看代码：



从代码发现两个问题：

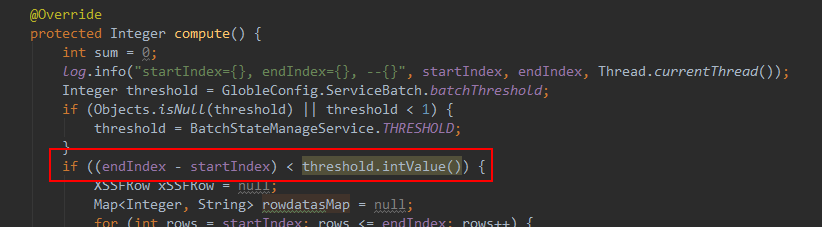
1. workbook没有手动关闭，workbook需要手动关闭。如果不关闭可能发生内存泄漏的风险。
2. 使用了forkjoin框架，任务拆分为单个任务50个，执行任务过多。

继续排查：

1. 发现使用了forkjoin框架 ，同时每个任务执行50行。

此时这7.5W 拆分出50行的单个任务， 7.5W/50 = 1500个任务。

任务数有点多，此时猜测：是否是因为拆分的任务数太多了，然后又有那么多的并发数，又因为使用的是forkjoin框架 ，需要等需要的线程处理完返回结果之后 ，中间解析excel才会被释放。



由于猜测是forkjoin的任务拆分的问题，做以下测试：

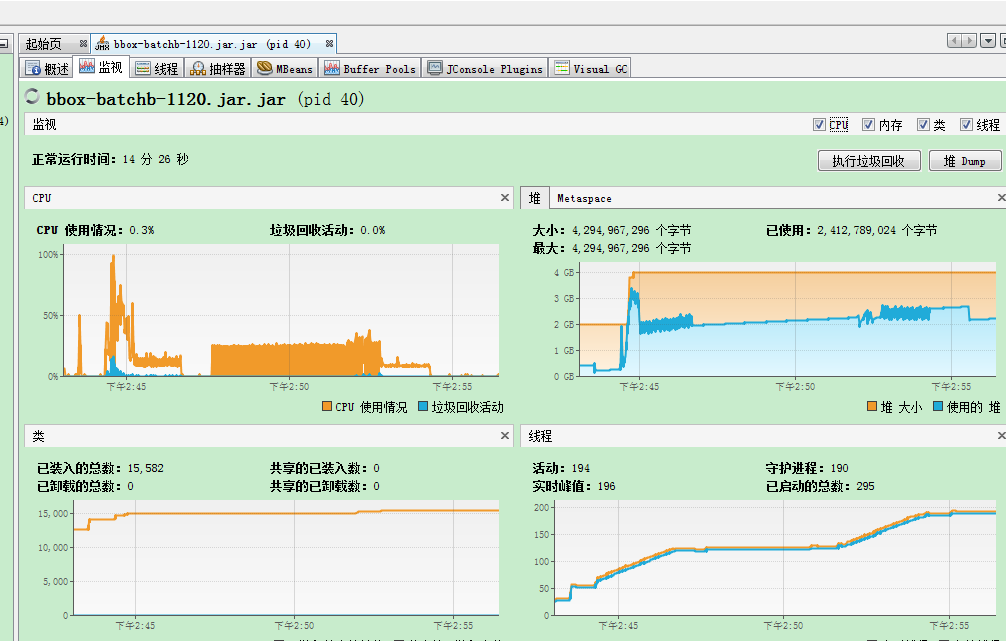
测试1：

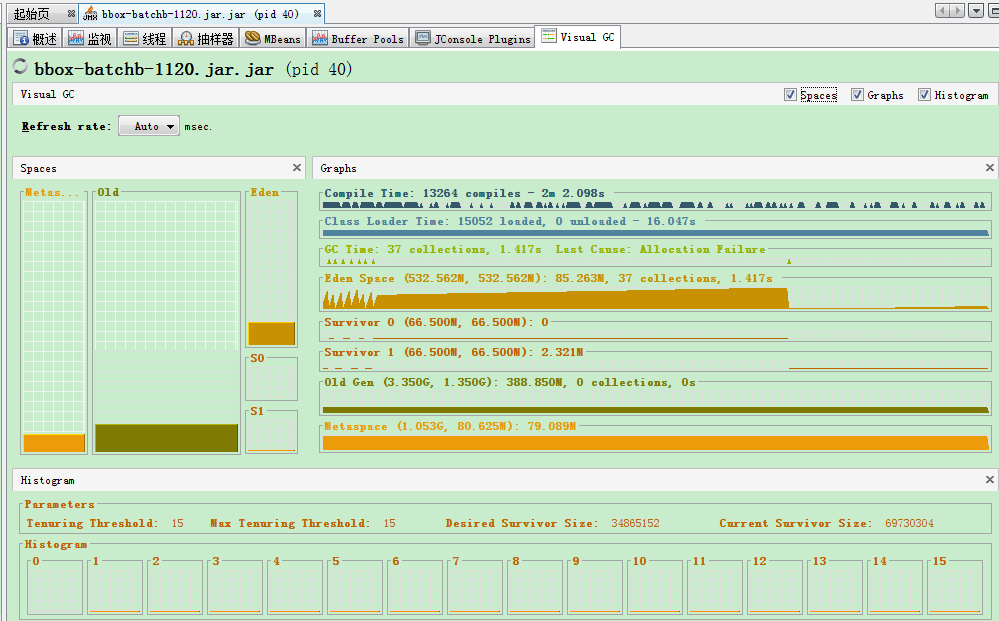
修改单个任务的处理量为10W, 使其只有一个任务，再次执行，

结果：情况与修改之前一样

测试2：去除forkjoin框架， workbook添加关闭：

结果：情况相同，并没有什么变化，结论跟forkjoin关联不大。





再次分析，重点着重分析excel解析的过程

通过

堆内存中解析的元素的element数量，发现还是excel解析过大，查找是否有其他解析方式。

后面发现果然还有。Poi解析 excel 有两种方式，一个是dom解析，一种是sax解析。

dom方式是一次性全部加载到了内存。而sax方式会按需加载，对内存消耗较少。

就目前情况来看使用了dom方式。

所以综合结论：

1. workboot 需要手动关闭，避免内存泄漏风险；
2. 不用forkjoin框架 ，拆分任务过多，使用过多处理器，会占用其他服务正常资源；
3. Excel解析使用poi 提供的sax方式。