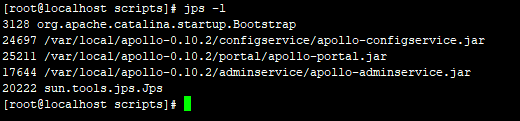
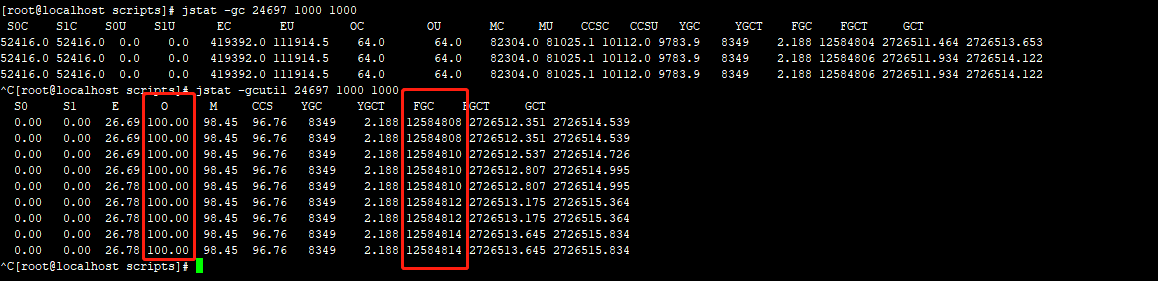
**记一次内网环境Java进程JVM参数优化——2021.8.5**

选择组件为部署在内网的第三方的配置中心，最初由前同事部署，平时使用略卡，借此课程机会做下试验

1、jps获取进程

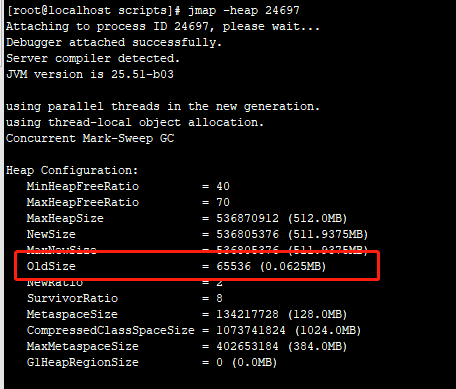


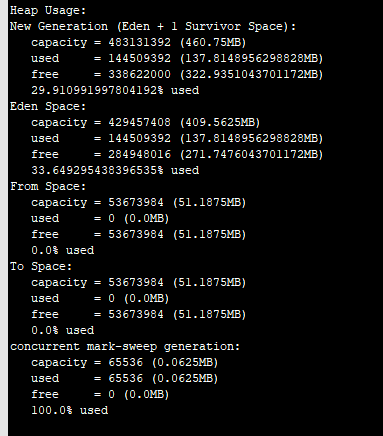
2、选择其中一个进程，如24697，执行jstat –gc（-gcutil）



由此可以看出，当前JVM old区已经爆满，此时将会频繁触发fullgc，由倒数第三列也可以看出，每隔几秒就进行一次fullgc

3、使用jmap查看jvm内存各个区域使用情况，其中一部分如下



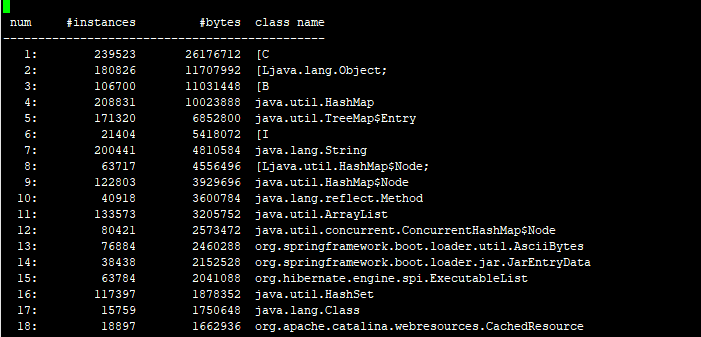


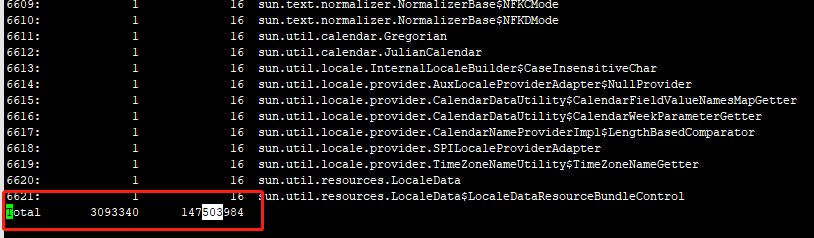
可以看出oldsize仅为64K，这显然是JVM参数配置不合理。另外也能得到各个区所占空间大小

4、那么是哪些对象(类型、数据)占用了内存呢，使用jmap –histo查看下，由于比较长，输出到一个文件中查看

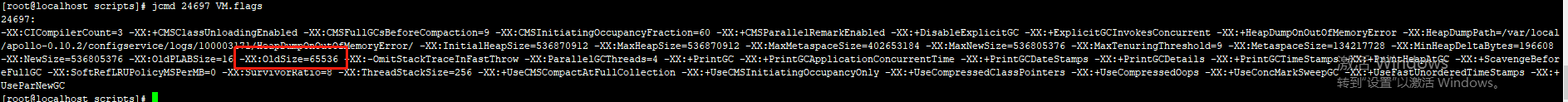


查看输出文件即可看出各类型实例数以及占用空间大小，并拉到最后看总占用，基本与上面输出的占用空间大小相近（140多M）：

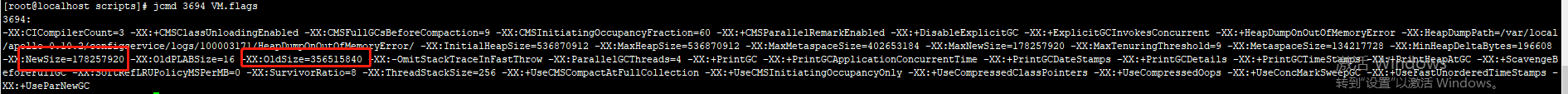




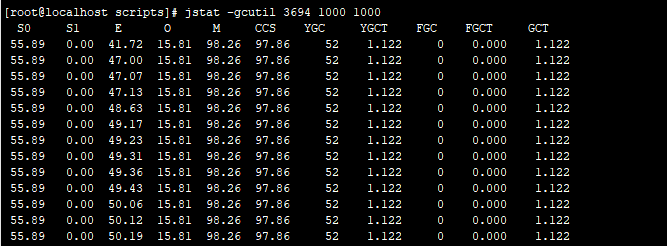
5、由于这是一个第三方组件，从源代码优化的可行性不高，那就从jvm参数上进行优化，通过jcmd VM.flags查看参数：



6、那么重新分配下参数，总堆内存大小为512M，那么新生代设置为170M，老年代大小为340m

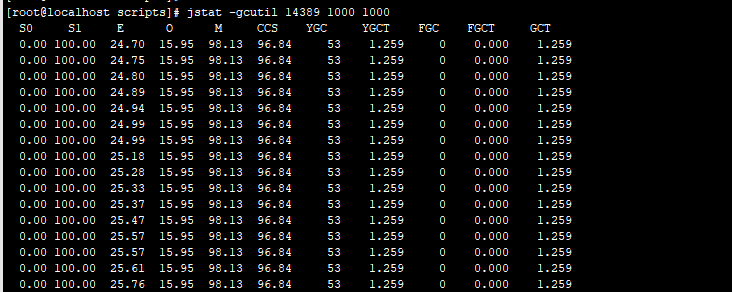


7、再次执行jstat –gcutil



不再频繁进行fullgc

8、经过一段时间后发现，进行young gc后，S1区爆满，这说明年轻代空间太小了，不足以容纳young gc后仍存活的对象

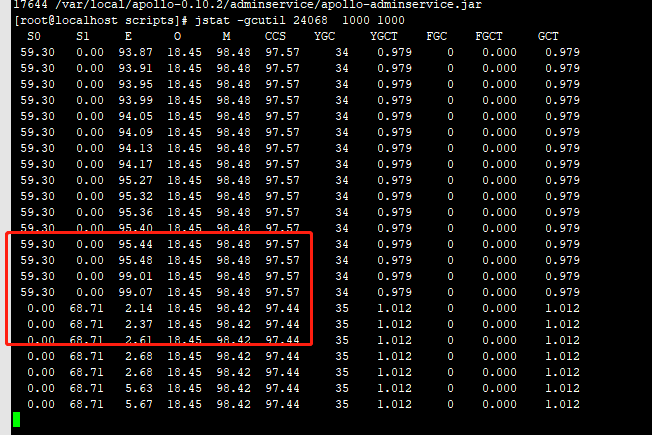


9、那么方法有3个：

一是保持堆大小不变，调整年轻代和老年代的比例；

二是保持年轻代大小不变，调整-XX:SurvivorRatio大小，这个大小默认是8

三是增大堆空间大小。由于内网虚拟机环境配置较低，且需部署多个应用，所以我先尝试调整年轻代比例，设置年轻代老年代大小均为256M，让我们来看一下效果如何：



进行一次young gc后，可以看出年轻代已经足够容纳，而老年代空间也不吃紧，

此次JVM优化于是告一段落，有操作或配置不当请多指教。