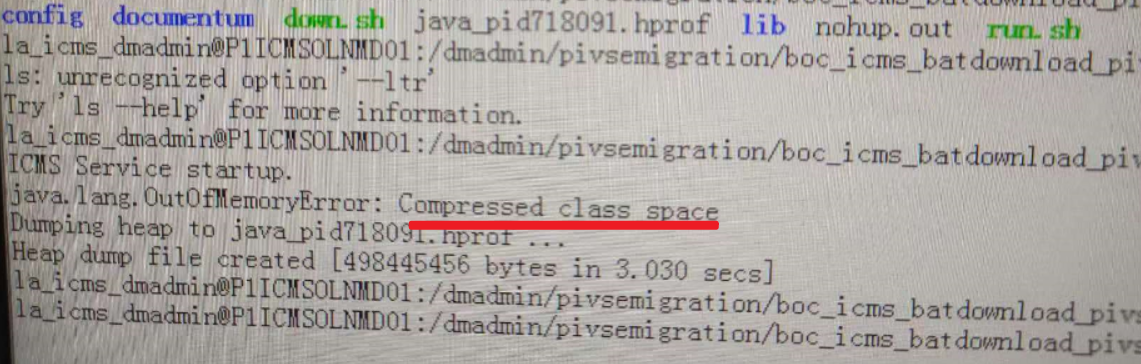
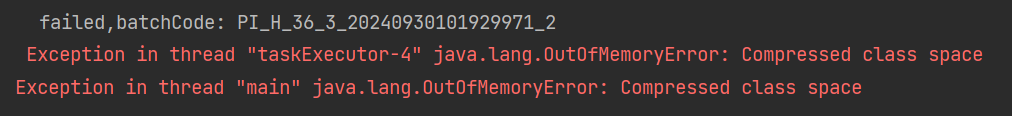
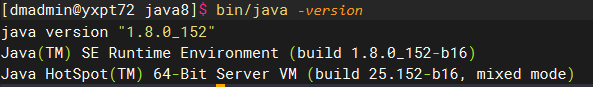
# 1内存异常

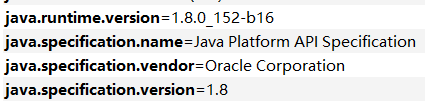




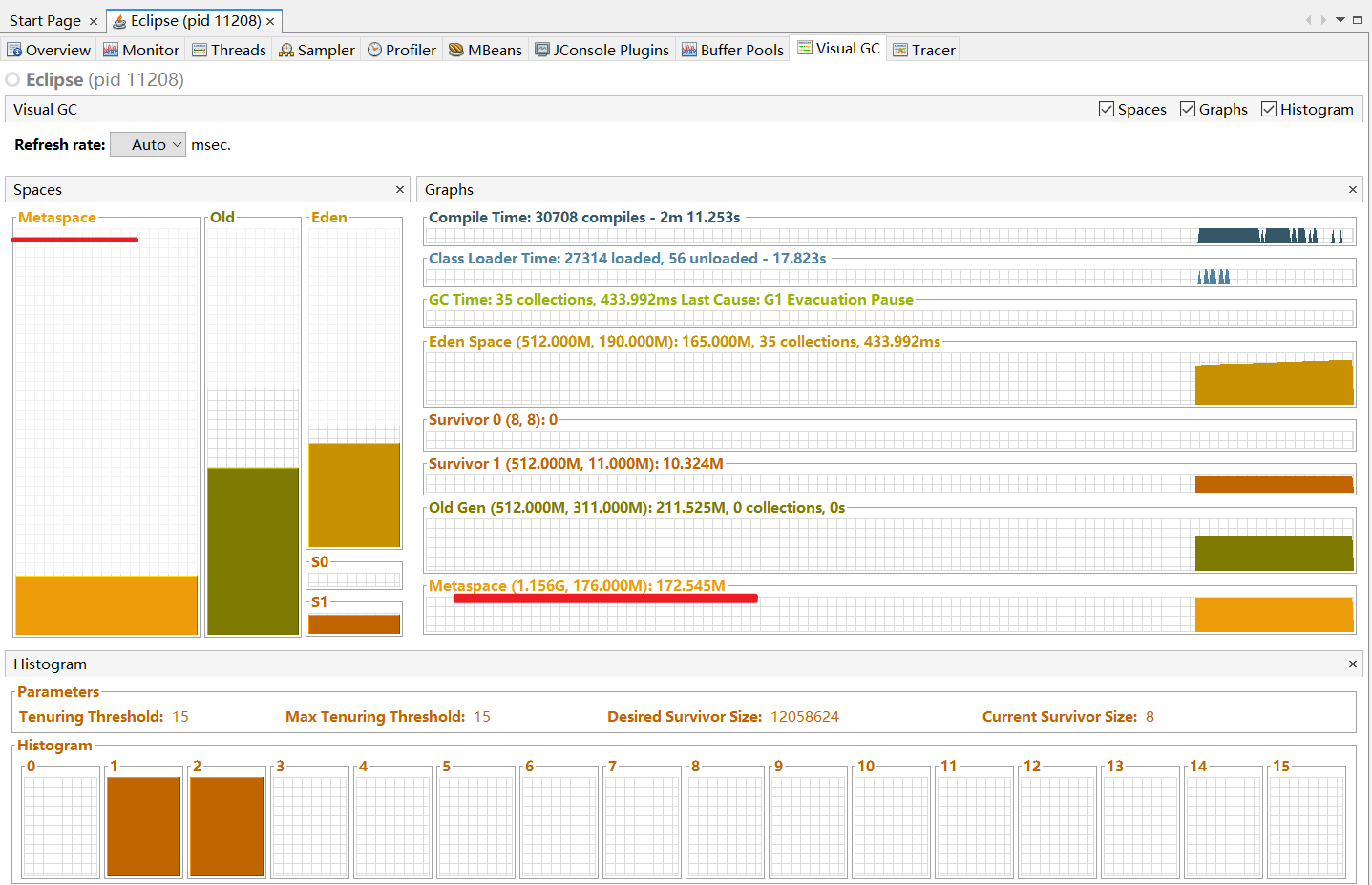
 Compressed Class Space

## Java 版本





# 2.visualvm



# 3.堆文件有啥

-XX:+HeapDumpOnOutOfMemoryError 用于在发生 OutOfMemoryError（内存溢出错误）时生成一个堆转储文件（Heap Dump）。堆转储文件记录了 JVM 在发生内存溢出时的**内存状态，包括所有活动对象及其引用关系**。包含了以下几个方面的信息：

**1.所有活动对象**：包括对象的类型、大小、数量以及它们之间的引用关系。

**2.内存分布**：显示不同类型的对象在堆内存中的分布情况，帮助识别哪些对象占用了大

量内存。

**3.线程信息**：可以查看每个线程的状态以及它持有的锁信息，这对于定位死锁等问题非

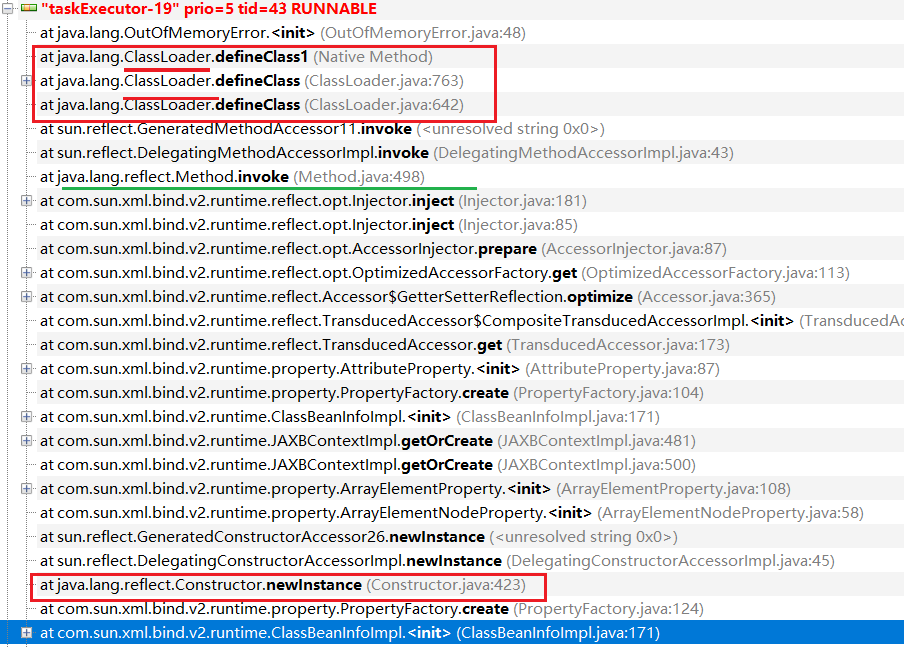
常有用。

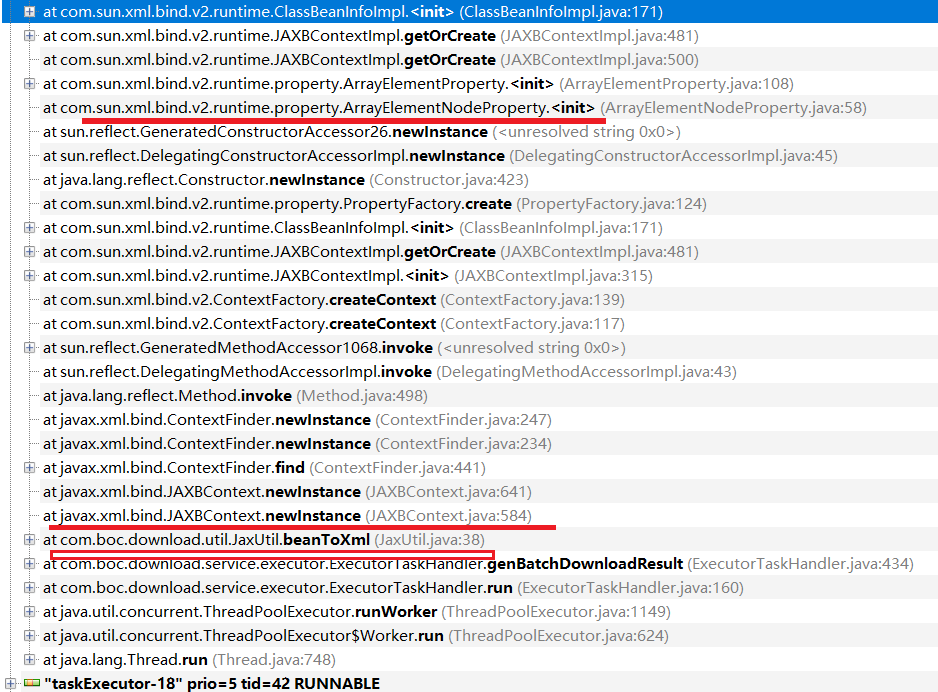
**4.GC日志**：如果启用了GC日志，那么在堆转储文件中也可以看到相关的GC活动记

录。

**5.类加载器信息**：显示了类加载器的状态，有助于发现潜在的类加载器泄露问题。

# 4.问题日志

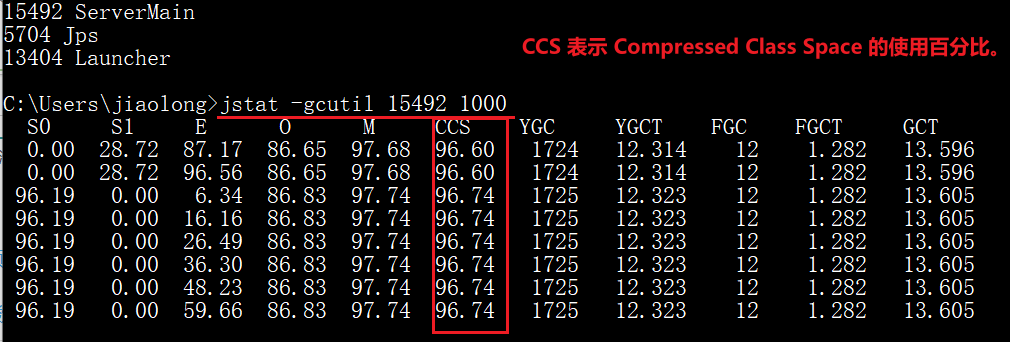


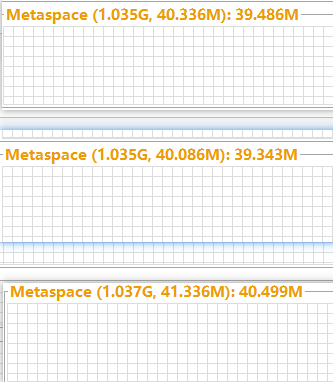


# 5.元空间与压缩空间的关系？

问题1：元空间 与 压缩空间的分配关系？

问题2：如何查看压缩空间的 大小与使用率





# 巨大疑问？

**为什么 dump文件统计的内存 大小 与给 jvm分配的内存大小不一致？**

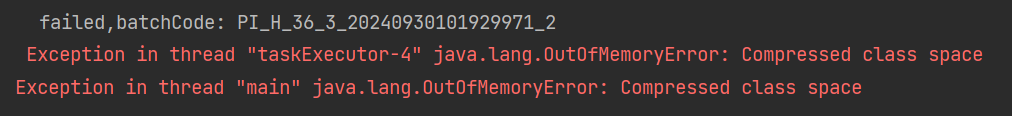
**测试：**

单线程，内存溢出 显示的堆内存使用量 与 分配至jvm的内存是一致的。

**说明不是堆内存溢出**

# 模拟参数

Xmx420m -Xms420m -XX:+HeapDumpOnOutOfMemoryError -XX:CompressedClassSpaceSize=4m



# 启动时添加以下参数，输出内存日志

-XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintHeapAtGC –

#JMC工具监控程序内存使用情况

**-XX:+UnlockCommercialFeatures -XX:+FlightRecorder -XX:StartFlightRecording=duration=60s,filename=recording.jfr**

1. -XX:+UnlockCommercialFeatures
   * 启用JVM中的商业特性。这些特性通常需要特定的许可才能使用，例如某些高级垃圾回收器或诊断工具。
2. -XX:+FlightRecorder
   * 启用Java Flight Recorder（JFR），这是一个高性能的事件采集框架，用于收集JVM和应用程序的运行时数据。JFR可以帮助开发者进行性能调优和故障排查。
3. -XX:StartFlightRecording=duration=60s,filename=recording.jfr
   * 配置Java Flight Recorder的启动参数：
     + duration=60s：指定录制时间为60秒。
     + filename=recording.jfr：指定生成的录制文件名为recording.jfr。

