GC参数配置及优化文档

目录

[第一步：jvm参数配置 1](#_Toc505703053)

[第二步：分析gc日志重新配置jvm参数大小 2](#_Toc505703054)

[第三步：对运行系统实时监控 4](#_Toc505703055)

[第四步：dump文件分析 6](#_Toc505703056)

## 第一步：jvm参数配置

-agentlib:ipharmacare\_hook （公司指定文件：解析密钥需用到）

-XX:MetaspaceSize=256m （元空间初始空间大小）

-XX:MaxMetaspaceSize=512m （元空间 最大空间，默认是没有限制的。）

-Xms1024m （设置JVM最大可用内存为1024M。）

-Xmx1024m （设置JVM促使内存1024m。此值可与上值相同）

-Xmn512m （设置年轻代大小为512m）

-XX:+PrintGCDetails （输出GC的详细日志）

-XX:+PrintGCDateStamps （输出GC的时间戳）

-Xloggc:/mnt/yyspace/logs/gc/system\_gc.log （GC输出路径）

-XX:+UseGCLogFileRotation （启用GC日志文件的自动转储）

-XX:NumberOfGCLogFiles=3 （设置滚动日志文件个数为）

-XX:GCLogFileSize=2048K （设置滚动日志文件大小为）

-XX:+PrintStringTableStatistics 在JVM进程退出时，打印出StringTable的统计信息到标准日志输出目录中

-XX:+HeapDumpOnOutOfMemoryError 可以让JVM在出现内存溢出时候Dump出当前的内存转储快照。

-XX:HeapDumpPath=/mnt/yyspace/logs/heap/sys/ （保存快照的地址）

-Dcom.sun.management.jmxremote （以下参数为开启远程监控）

-Djava.rmi.server.hostname=10.1.1.186

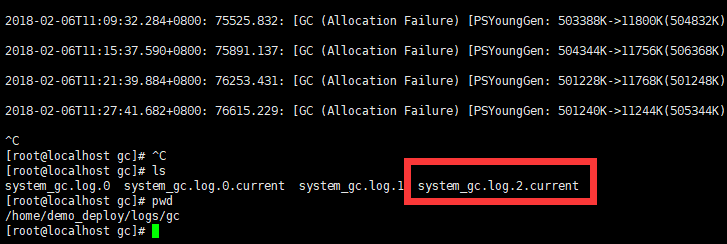
-Dcom.sun.management.jmxremote.port=8999

-Dcom.sun.management.jmxremote.authenticate=false

-Dcom.sun.management.jmxremote.ssl=false

-Dcom.sun.management.jmxremote.local.only=false

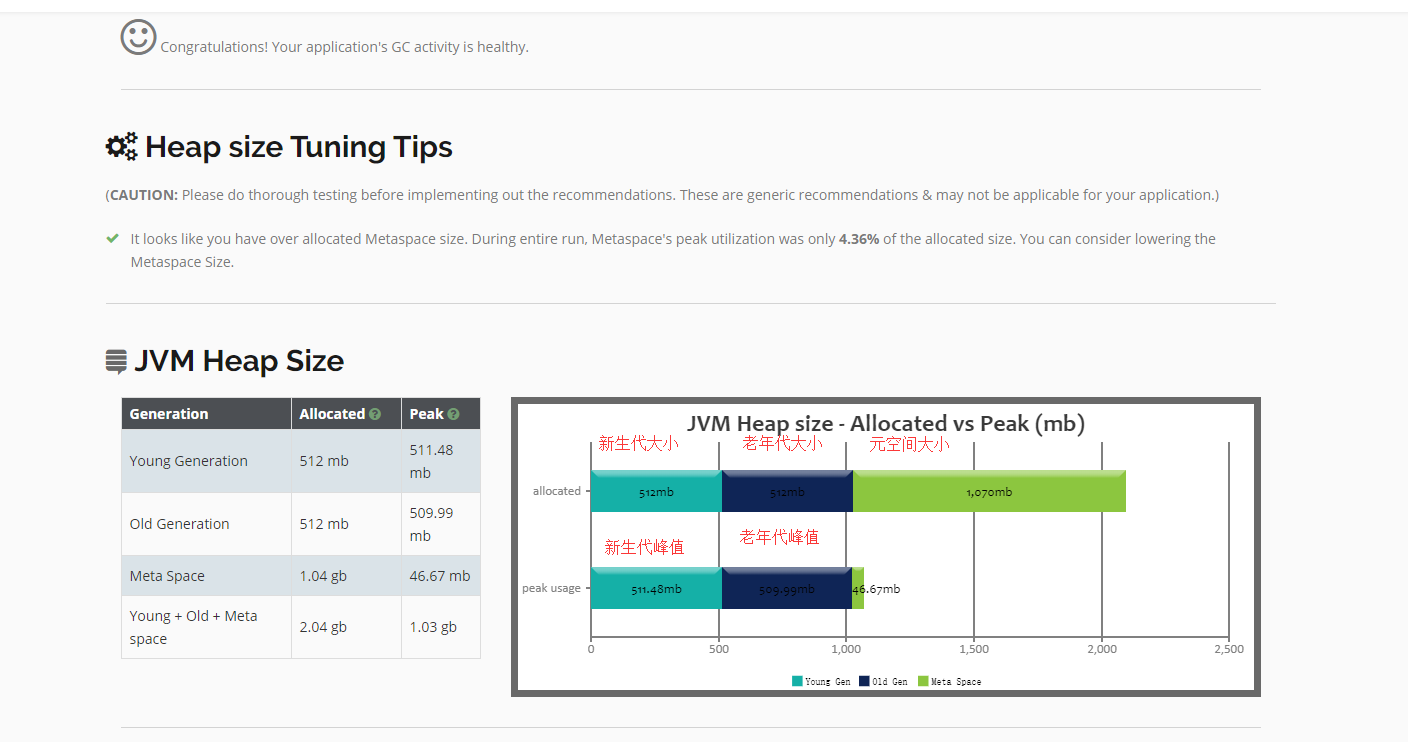
## 第二步：分析gc日志重新配置jvm参数大小



进入项目启动服务器 拉出gc日志到本地，利用在线分析工具进行分析

工具地址：

<http://gceasy.io/diamondgc-report.jsp?oTxnId_value=6a77f7cb-560b-4b87-8e39-4537724b9d9c>

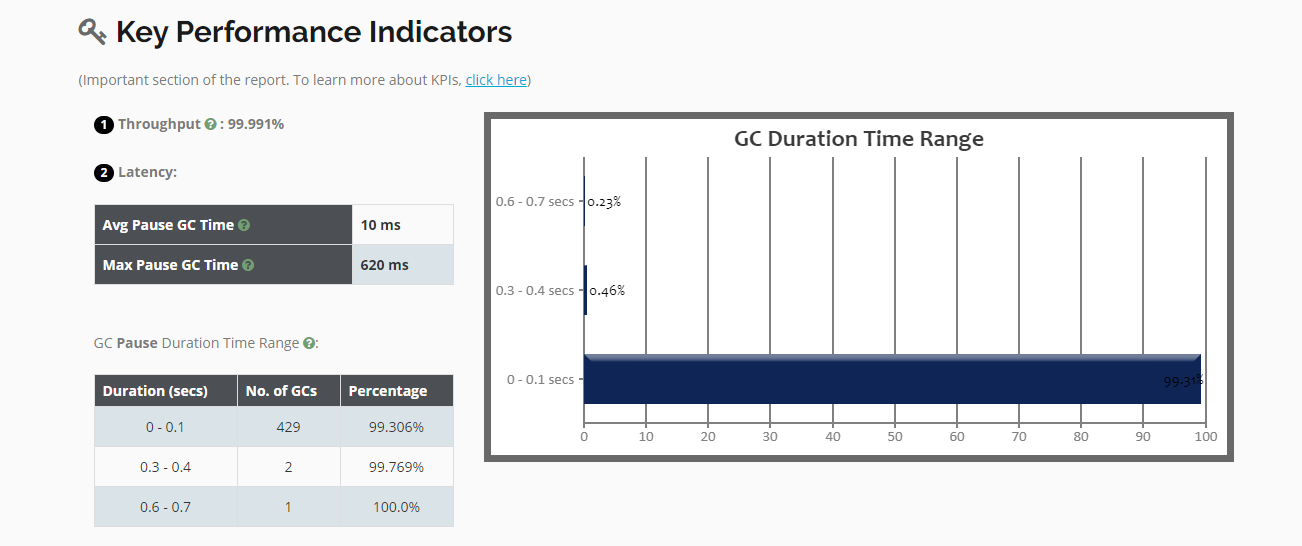


第一行 为整体gc日志的问题情况

第二行 为堆大小优化提供的信息

第三行 图标结构为对应gc大小和使用情况 可根据要求调整jvm配置

（如上图 ：我的元空间使用量远小于分配大小需要对jvm做出调整）



**Throughput  为**吞吐量，**Latency为gc时间（平均时间 ，最大用时）**

吞吐量

吞吐量是您的应用程序在特定时间段内完成的生产性工作量。这带来了什么是生产性工作的问题？什么是非生产性工作？

**生产性工作：**这基本上是您的应用程序花费在处理客户交易上的时间。

**非生产性工作：**这基本上是您的应用程序花在管理工作上的时间，主要是垃圾收集。

假设您的应用程序运行了60分钟。在这60分钟内，假设2分钟用于GC活动。

这意味着应用程序花了3.33％的GC活动（即2/60 \* 100）

这意味着应用程序吞吐量是**96.67％**（即100 - 3.33）。

现在的问题是：什么是可接受的吞吐量％？这取决于应用程序和业务需求。通常应该以超过95％的吞吐量为目标。

此工具还有其他信息可供参考

## 第三步：对运行系统实时监控

1.  创建jstatd服务器授权文件jstatd.all.policy，代码如下然后将改文件放到Java/bin目录下。

grant codebase "file:${java.home}/../lib/tools.jar" {

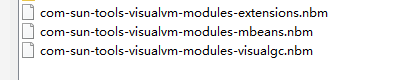
permission java.security.AllPermission;};

2. 切换到java/bin目录下，启动Jstatd服务器

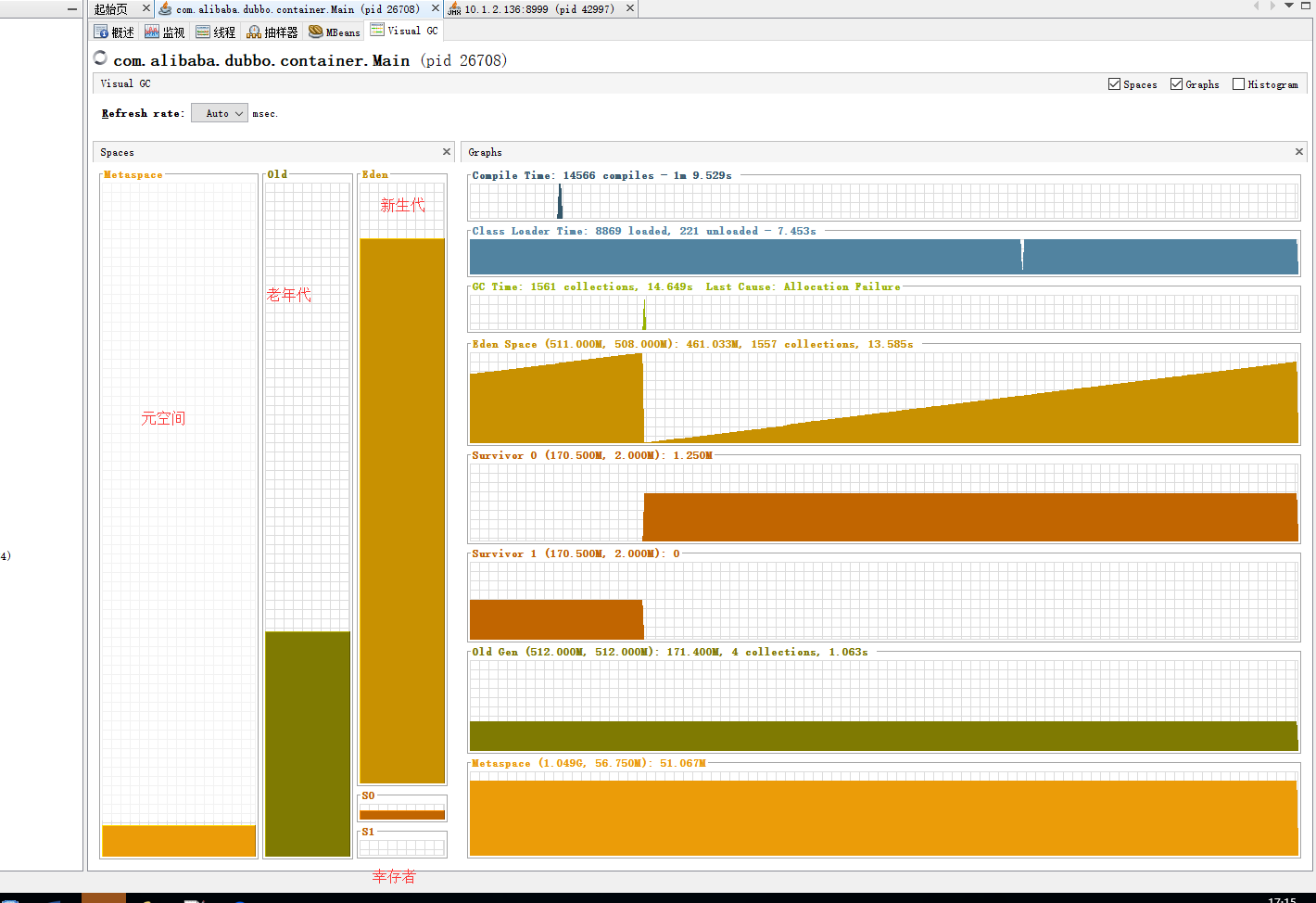
./jstatd -J-Djava.security.policy=jstatd.all.policy

经过上述配置开启运行服务器上的RMI支持 ，在本地java/bin下打开jvisualvm，

点击工具---》插件-----》已下载-----》添加插件



等待安装完成，程序输入连接ip即可



如上图 左侧是增长情况，右侧是占用/分配内存大小信息，根据使用情况分配相应的jvm参数配置。

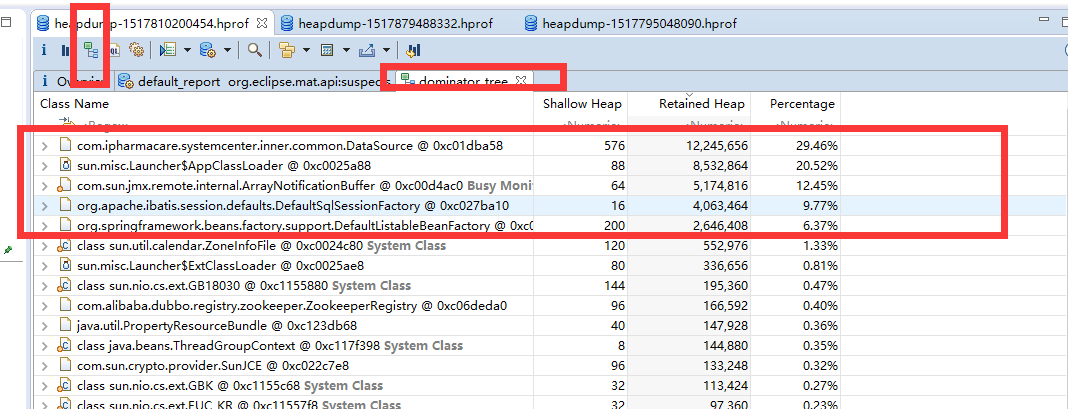
查看监视标签查看 cpu，和堆的使用情况。如果上图的老年代占用比较多的内存 或者 监控标签中 堆大小做gc回收较小时，需要导出右上角 堆Dumo文件，拉取到本地进行分析。



## 第四步：dump文件分析

安装 AMT 分析工具（有独立压缩包解压即可用，无需下载Eclipse）

点击左上角 file 导入dump文件（文件来源可以是实时监控时导出的，也可以是是根据jvm配置内存溢出时系统保存的）



可以分析前几个占用内存比较大的类 首先分析自己代码包路基的类（如com.ipharmacare.systemcenter.inner.common.DataSource这个类），其次才是系统包配置调整

逐层展开需要分析的类 观察其value值 优化代码

