Java生产环境下性能监控与调优详解

1、基于JDK命令行工具的监控

（1）、JVM参数类型

<1>、标准参数

-help

-server -client

-version -showversion

-cp -classpath

<2>、X参数

非标准化参数

-Xint:解释执行

-Xcomp:第一次使用就编译成本地代码

-Xmixed:混合模式，JVM自己来决定是否编译成本地代码

[pcc@localhost pcc-api]$ java -version

openjdk version "1.8.0\_171"

OpenJDK Runtime Environment (build 1.8.0\_171-b10)

OpenJDK 64-Bit Server VM (build 25.171-b10, mixed mode)

[pcc@localhost pcc-api]$ java -Xint -version

openjdk version "1.8.0\_171"

OpenJDK Runtime Environment (build 1.8.0\_171-b10)

OpenJDK 64-Bit Server VM (build 25.171-b10, interpreted mode)

[pcc@localhost pcc-api]$ java -Xcomp -version

openjdk version "1.8.0\_171"

OpenJDK Runtime Environment (build 1.8.0\_171-b10)

OpenJDK 64-Bit Server VM (build 25.171-b10, compiled mode)

<3>、XX参数

特点：

非标准化参数，相对不稳定，主要用于JVM调优和Debug

类型：

Boolean类型

格式：-XX：[+-]<name>表示启用或者禁用name属性

比如：-XX:+UseConcMarkSweepGC //启用cms垃圾收集器

-XX:+UseG1GC //开启G1垃圾收集器

非Boolean类型

格式：-XX：<name>=<value>表示name属性的值是value

比如：-XX：MaxGCPauseMillis=500 //GC的最大停顿时间为500

XX:GCTimeRatio=19 //

-xss(设置堆栈)

1. 、运行时JVM参数查看

如何查看运行时参数的值：jinfo -flag MaxHeapSize

[root@sww ~]# jinfo -flag MaxHeapSize 8525（进程pid）

-XX:MaxHeapSize=262144000

运行时堆栈的值：jinfo -flag ThreadStackSize 8525

[root@sww ~]# jinfo -flag ThreadStackSize 8525

-XX:ThreadStackSize=1024

-XX:+PrintFlagsInitial //查看初始值

-XX:+PrintFlagsFinal //查看最终的值

JVM中有一些参数并不是都是通过上面的方式直接赋值的需要添加

-XX:+UnlockExperimentalVMOptions解锁实验参数

-XX:+UnlockDiagnosticVMOptions解锁诊断参数

-XX:+PrintCommandLineFlags打印命令行参数



java -XX:+PrintFlagsFinal -version //查看初始化参数值

java -XX:+PrintFlagsFinal -version > flags.txt //将打印的信息转换到指定文件中

sz flags.txt //将某个文件下载到本地

jps讲解

查看运行中的jps

[root@sww ~]# jps

6626 Jps

8413 Bootstrap

8525 Bootstrap

[root@sww ~]# jps -l

6636 sun.tools.jps.Jps

8413 org.apache.catalina.startup.Bootstrap

8525 org.apache.catalina.startup.Bootstrap

与jps相关的网站：

<https://docs.oracle.com/javase/8/docs/technotes/tools/unix/index.html>

https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jps.html#CHDCGECD

查看最大内存

[root@sww ~]# jinfo -flag MaxHeapSize 8525

-XX:MaxHeapSize=262144000

（3）、jstat查看虚拟机统计信息

https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jstatd.html#BABEHFHF



[root@sww ~]# jstat -class 8525 1000 10 //1000代表1000秒，10代表10次

Loaded Bytes Unloaded Bytes Time

6469 12895.6 16 19.1 7.22

6469 12895.6 16 19.1 7.22

6469 12895.6 16 19.1 7.22

6469 12895.6 16 19.1 7.22

6469 12895.6 16 19.1 7.22

6469 12895.6 16 19.1 7.22

6469 12895.6 16 19.1 7.22

6469 12895.6 16 19.1 7.22

6469 12895.6 16 19.1 7.22

6469 12895.6 16 19.1 7.22

[root@sww ~]#

类加载

某一个进程的类加载信息

[root@sww ~]# jps

8413 Bootstrap

8525 Bootstrap

6782 Jps

[root@sww ~]# jstat -class 8525

Loaded Bytes Unloaded Bytes Time

6469 12895.6 16 19.1 7.22

Loaded:加载类个数

Bytes:加载大小

Unloaded：卸载的个数

Time：类加载和卸载的时间

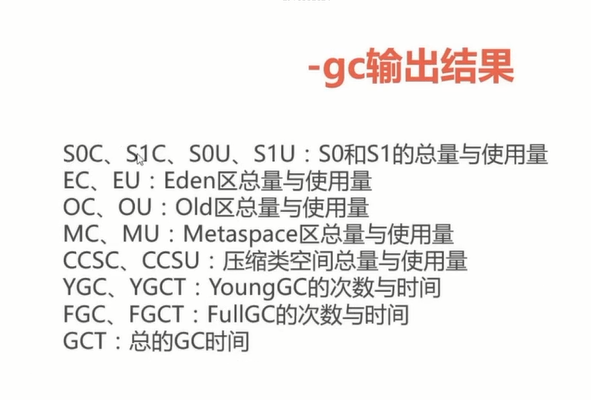
垃圾收集



[root@sww ~]# jstat -gc 8525

S0C S1C S0U S1U EC EU OC OU MC MU CCSC CCSU YGC YGCT FGC FGCT GCT

1664.0 1664.0 0.0 16.6 13696.0 13493.3 34000.0 27627.5 41344.0 40568.4 4736.0 4502.3 709 2.325 4 0.236 2.561



对象分配在堆里面，对象里面有一个指向自己的class对象的指针（例如User对象，有一个指针指向User.class）

JIT编译

[root@sww ~]# jstat -compiler 8525

Compiled Failed Invalid Time FailedType FailedMethod

5898 1 0 19.29 1 org/apache/tomcat/util/IntrospectionUtils setProperty

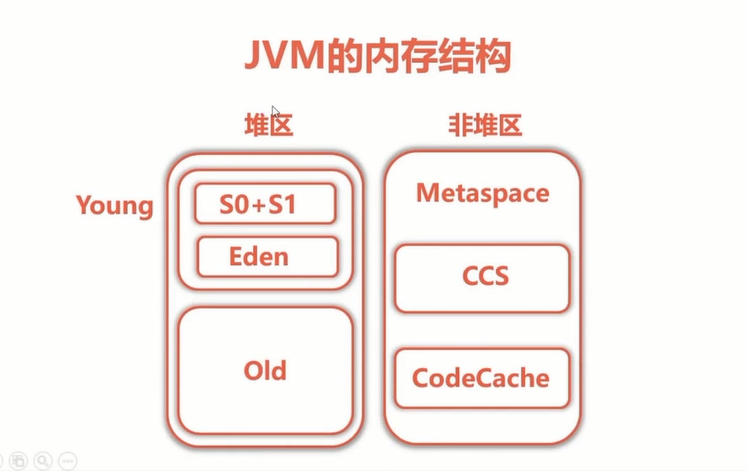
[root@sww ~]# jstat -printcompilation 8525

Compiled Size Type Method

5898 71 1 org/apache/tomcat/util/http/MimeHeaders getUniqueValue



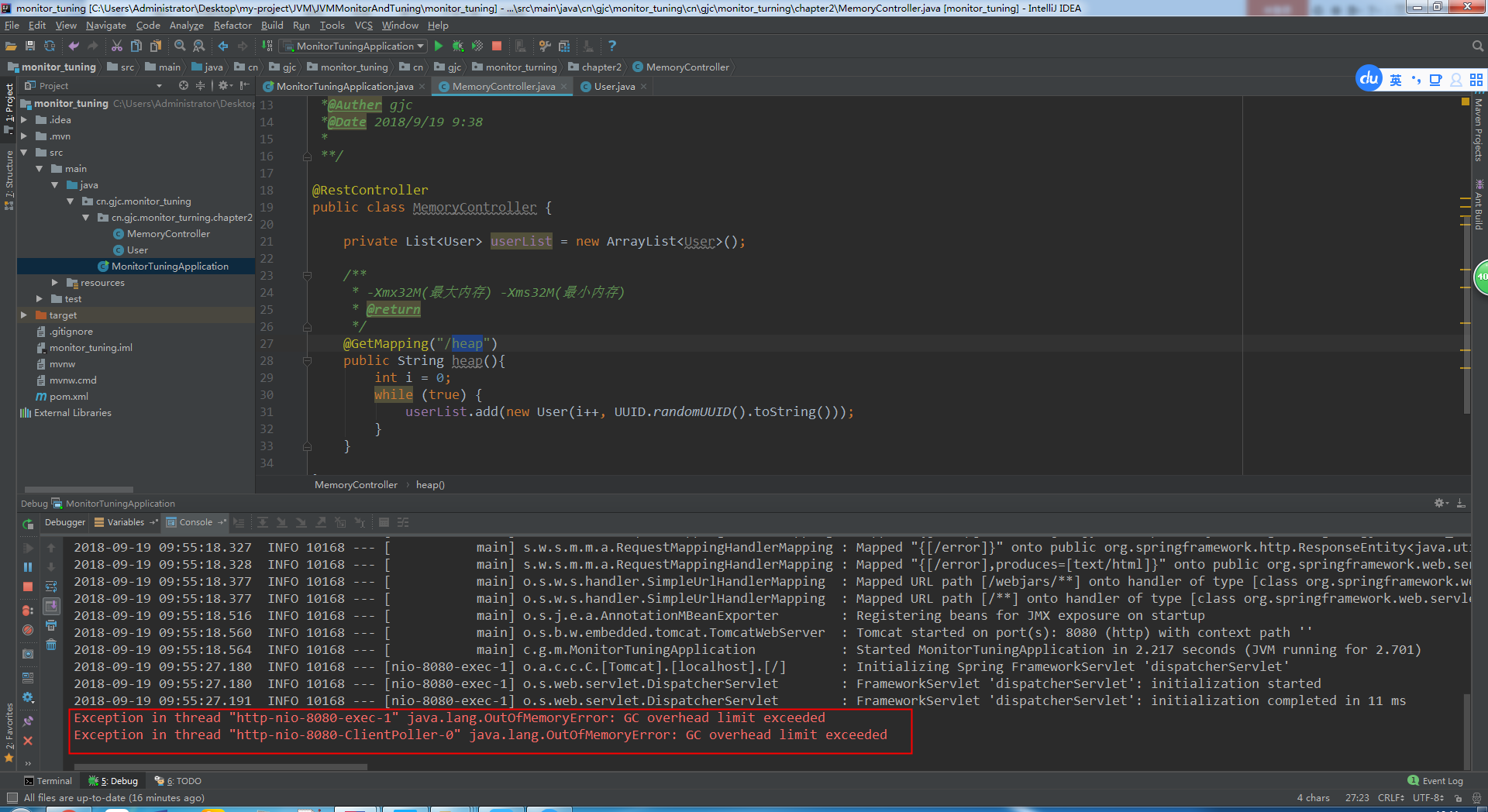
JVM内存结构



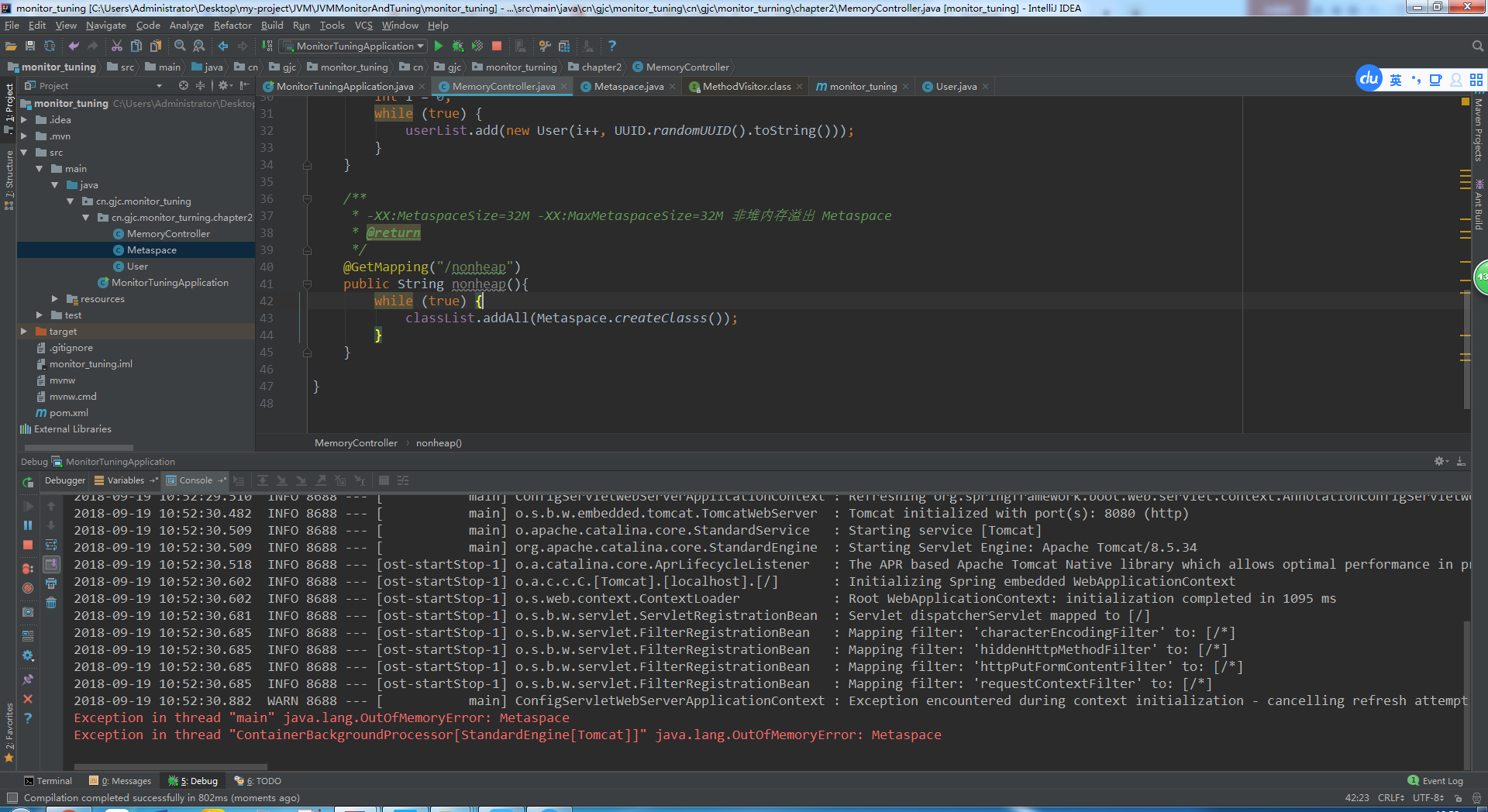
（4）、jmap+MAT实战内存溢出

（5）、jstack实战死循环和死锁

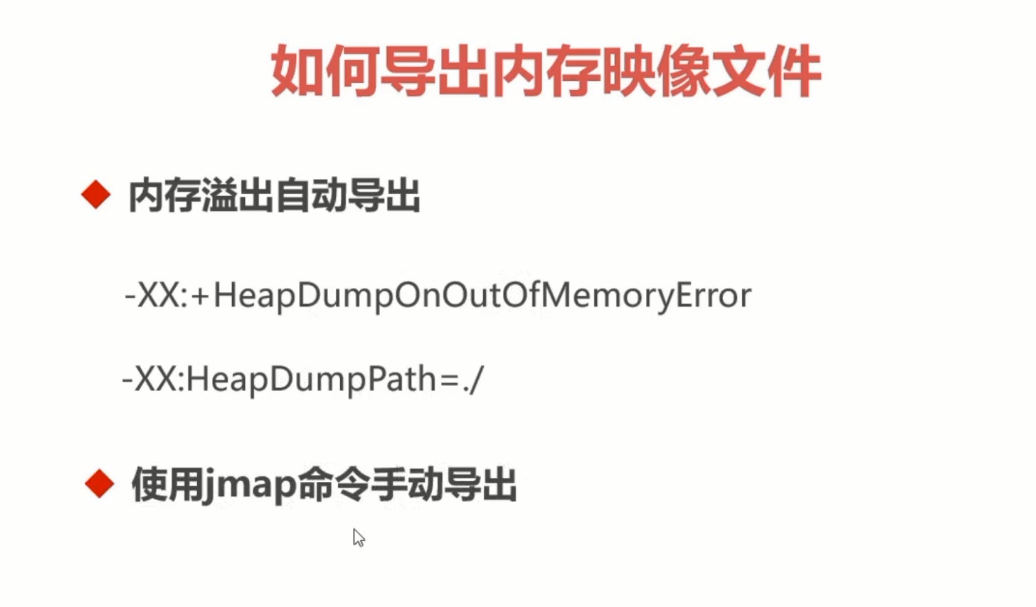
1、堆内存溢出

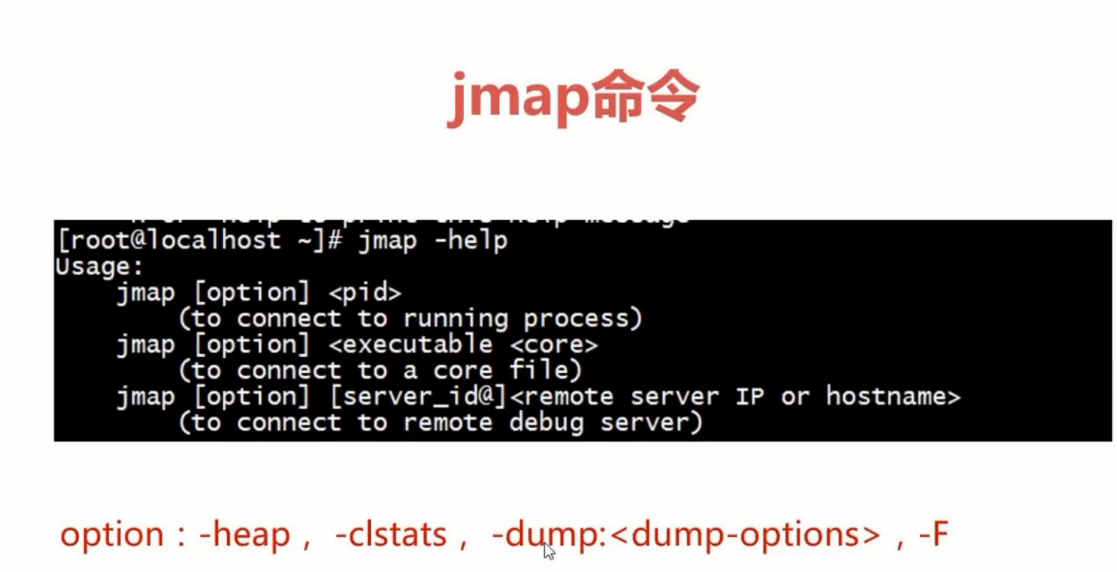


2、非堆内存溢出



生产环境出行内存溢出如何来解决





查找进程号

C:\Users\Administrator> jps -l

4176 cn.gjc.monitor\_tuning.MonitorTuningApplication

6336 org.jetbrains.jps.cmdline.Launcher

6244 sun.tools.jps.Jps

5592

10204 org.jetbrains.jps.cmdline.Launcher

5884 org.jetbrains.idea.maven.server.RemoteMavenServer

使用jmap导出

C:\Users\Administrator\Desktop>jmap -dump:format=b,file=heap.hprof 4176

Dumping heap to C:\Users\Administrator\Desktop\heap.hprof ...

Heap dump file created

查询运行内存中每一个区块大小

C:\Users\Administrator\Desktop>jmap -heap 4176

Attaching to process ID 4176, please wait...

Debugger attached successfully.

Server compiler detected.

JVM version is 25.161-b12

using thread-local object allocation.

Parallel GC with 8 thread(s)

Heap Configuration:

MinHeapFreeRatio = 0

MaxHeapFreeRatio = 100

MaxHeapSize = 33554432 (32.0MB)

NewSize = 11010048 (10.5MB)

MaxNewSize = 11010048 (10.5MB)

OldSize = 22544384 (21.5MB)

NewRatio = 2

SurvivorRatio = 8

MetaspaceSize = 21807104 (20.796875MB)

CompressedClassSpaceSize = 1073741824 (1024.0MB)

MaxMetaspaceSize = 17592186044415 MB

G1HeapRegionSize = 0 (0.0MB)

Heap Usage:

PS Young Generation

Eden Space:

capacity = 6815744 (6.5MB)

used = 6713504 (6.402496337890625MB)

free = 102240 (0.097503662109375MB)

98.49994365985577% used

From Space:

capacity = 2097152 (2.0MB)

used = 0 (0.0MB)

free = 2097152 (2.0MB)

0.0% used

To Space:

capacity = 1572864 (1.5MB)

used = 0 (0.0MB)

free = 1572864 (1.5MB)

0.0% used

PS Old Generation

capacity = 22544384 (21.5MB)

used = 22195336 (21.16712188720703MB)

free = 349048 (0.33287811279296875MB)

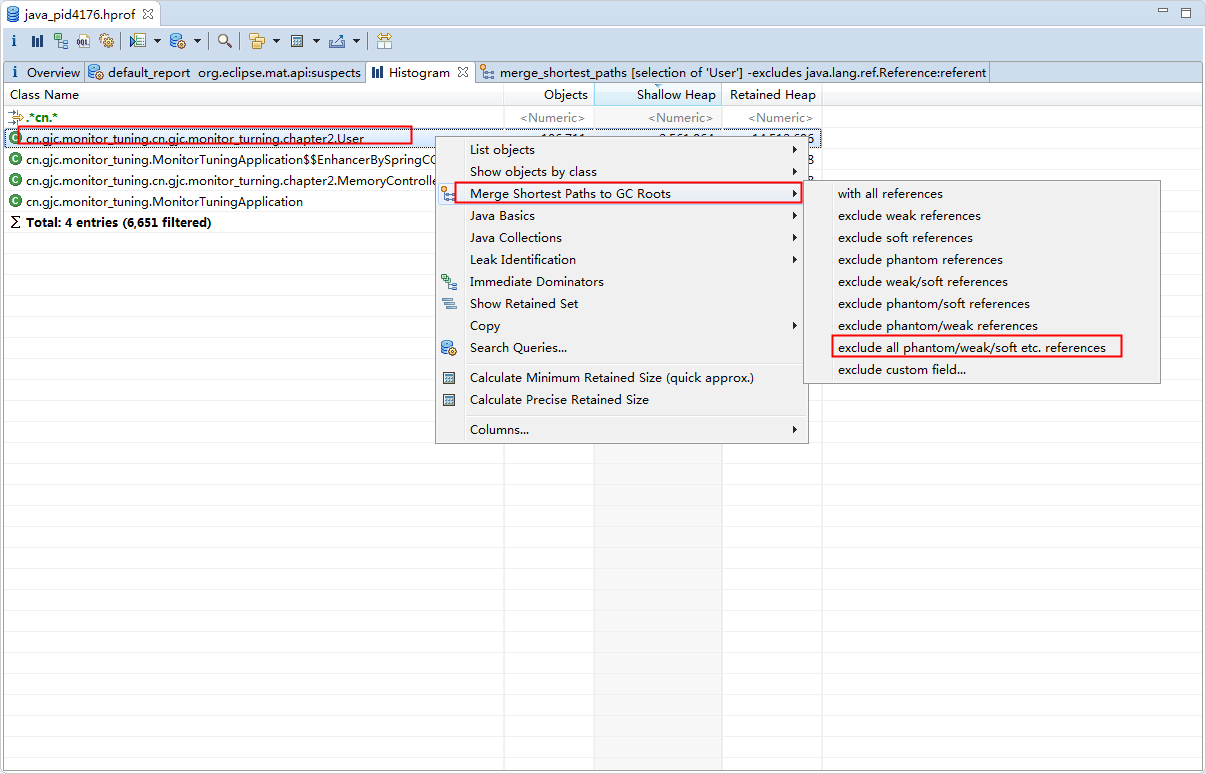
98.45172970793968% used

找到内存溢出，如何定位问题？

Mat分析内存溢出

MAT工具网站：www.eclipse.org/mat/downloads.php

查询内存溢出中的强引用



Jstack实战与死锁

查看jstack信息

C:\Users\Administrator\Desktop>jps -l

4176 cn.gjc.monitor\_tuning.MonitorTuningApplication

6336 org.jetbrains.jps.cmdline.Launcher

6868 sun.tools.jps.Jps

5592

10204 org.jetbrains.jps.cmdline.Launcher

5884 org.jetbrains.idea.maven.server.RemoteMavenServer

//打印信息

C:\Users\Administrator\Desktop>jstack 4176 > 4176.txt

//Linux环境还需要使用 sz 4176.txt进行下载相应的文件

//查看所有进程

Top -p 4176 -H