应用系统架构优化方法与实战案例第9讲

顺藤摸瓜
JVM原理剖析与调优性能收集

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

关于老师

福富软件应用性能调优团队核心成员梁敬彬、黄锏、黄维新、张荣志

当前课程进度

DATAGURU 炼数成金

应用系统架构
优化方法与案例实战

- 第01讲 混沌初开，携手走进性能优化的神秘世界
- 第02讲 乾坤始奠，基线理论与耗时分布信息获取
- 第03讲 锐意进取，平台性能收集手段与研究思路
- 第04讲 有备而战，Nmon工具的收集与诊断原理
- 第05讲 大道至简，数据库整体调优之收集的法宝
- 第06讲 心中无剑，Dt工具介绍与诊断优化的探讨
- 第07讲 抽丝剥茧，Dt工具的使用与实战落地案例
- 第08讲 电光火石，数据库整体调优之火箭般定位
- 第09讲 顺藤摸瓜，Jvm原理剖析与调优性能收集
- 第10讲 顺水推舟，数据库调优之SQL不改写优化
- 第11讲 峰回路转，数据库调优之SQL巧思妙改写
- 第12讲 平流缓进，从SQL优化迈进存储过程调优
- 第13讲 千里之行，数据库设计之建模缺陷与规避
- 第14讲 厚积薄发，系统框架优化之缓存机制应用
- 第15讲 众妙之门，系统框架之分布式的林林种种
- 第16讲 万事俱备，优化实施的相关实战宝典手册
- 第17讲 只欠东风，系列综合实战案例与全课总结

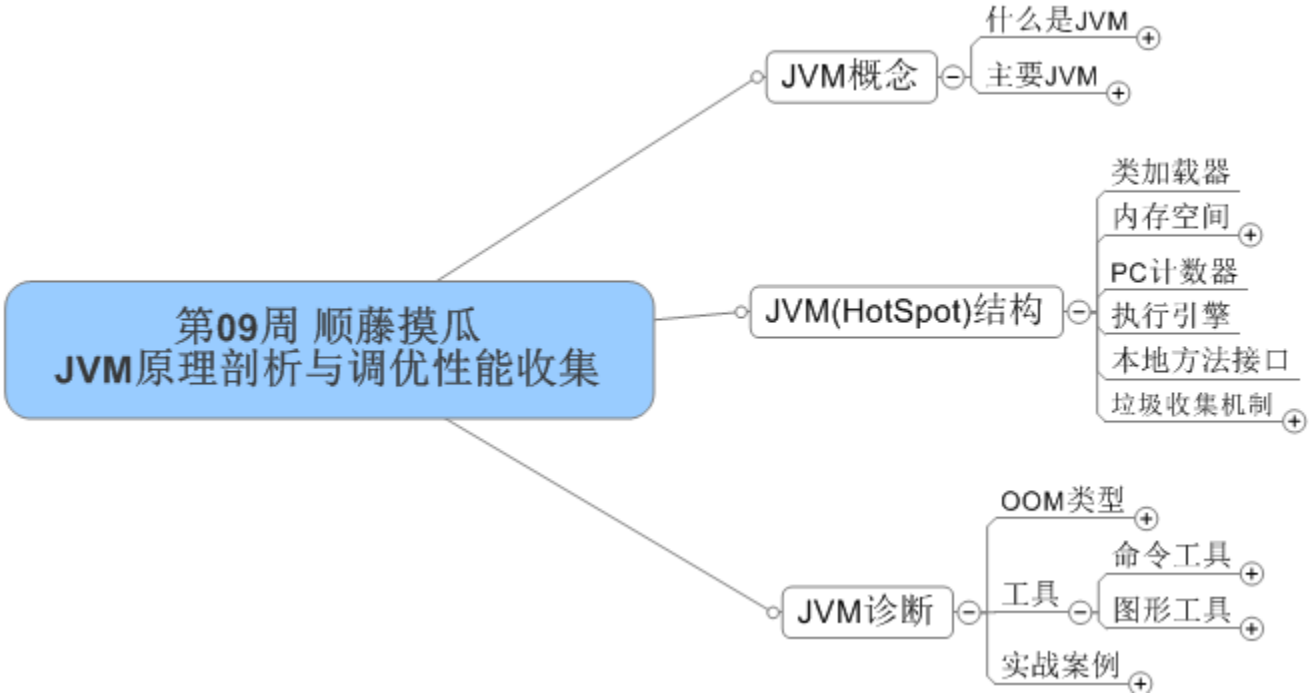
上周回顾

来来来，我们先一起回顾一下……

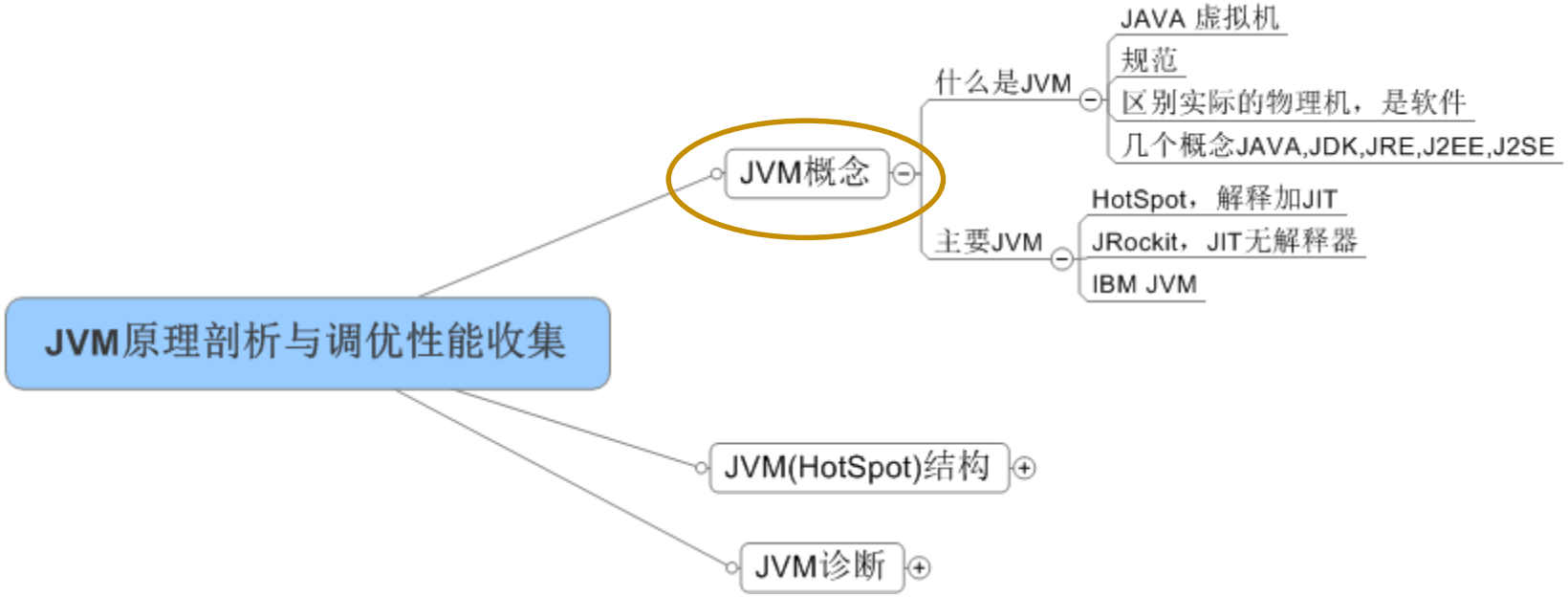
上周课程综述

上周作业回顾

本周课程的主要内容


第09周 顺藤摸瓜
JVM原理剖析与调优性能收集
JVM概念
什么是JVM
主要JVM
JVM(HotSpot)结构
类加载器
内存空间
PC计数器
执行引擎
本地方法接口
垃圾收集机制
JVM诊断
OOM类型
工具
命令工具
图形工具
实战案例

JVM概念


JVM原理剖析与调优性能收集
JVM概念
什么是JVM
JAVA 虚拟机
规范
区别实际的物理机，是软件
几个概念JAVA,JDK,JRE,J2EE,J2SE
主要JVM
HotSpot，解释加JIT
JRockit，JIT无解释器
IBM JVM
JVM(HotSpot)结构
JVM诊断

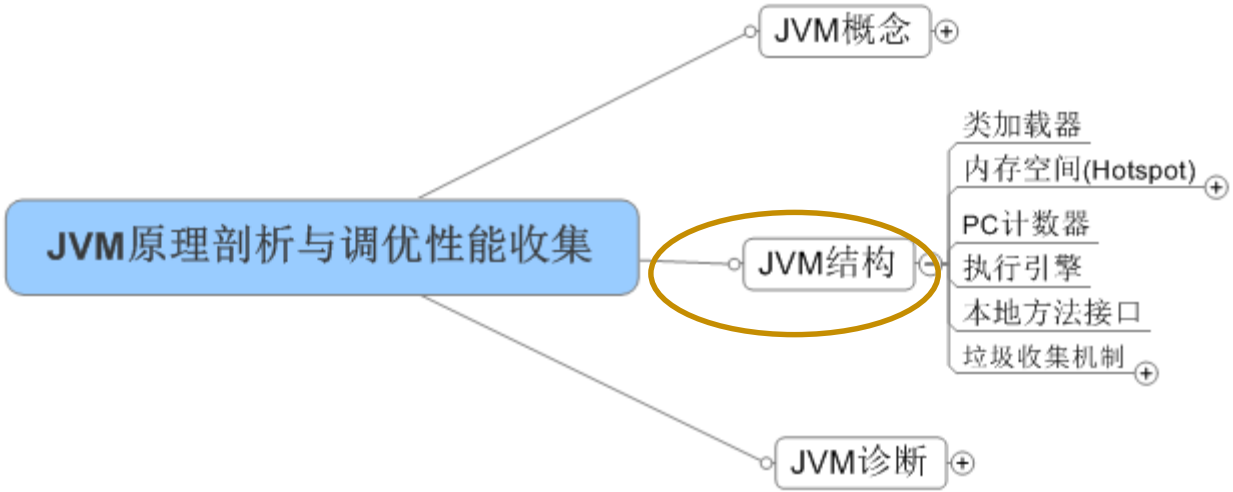
什么是JVM

- JVM是一套规范，用来定义JAVA虚拟机
- JVM即JAVA 虚拟机。java语言的平台无关性，是通过在不同的平台加一个中间层，即安装JVM虚拟机来实现的,而JVM虚拟机不是平台无关性的。
- java编译器只要面向JVM，生成JVM能理解的代码或字节码文件。JVM在执行字节码时，把字节码转化成具体平台上的机器指令执行。这就是JAVA能够“Write Once,Run Anywhere(一次编写，到处运行)”的原因。
- 虚拟机（Virtual Machine）指通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统。
- 什么是JVM,JRE,JDK,J2SE,J2EE?

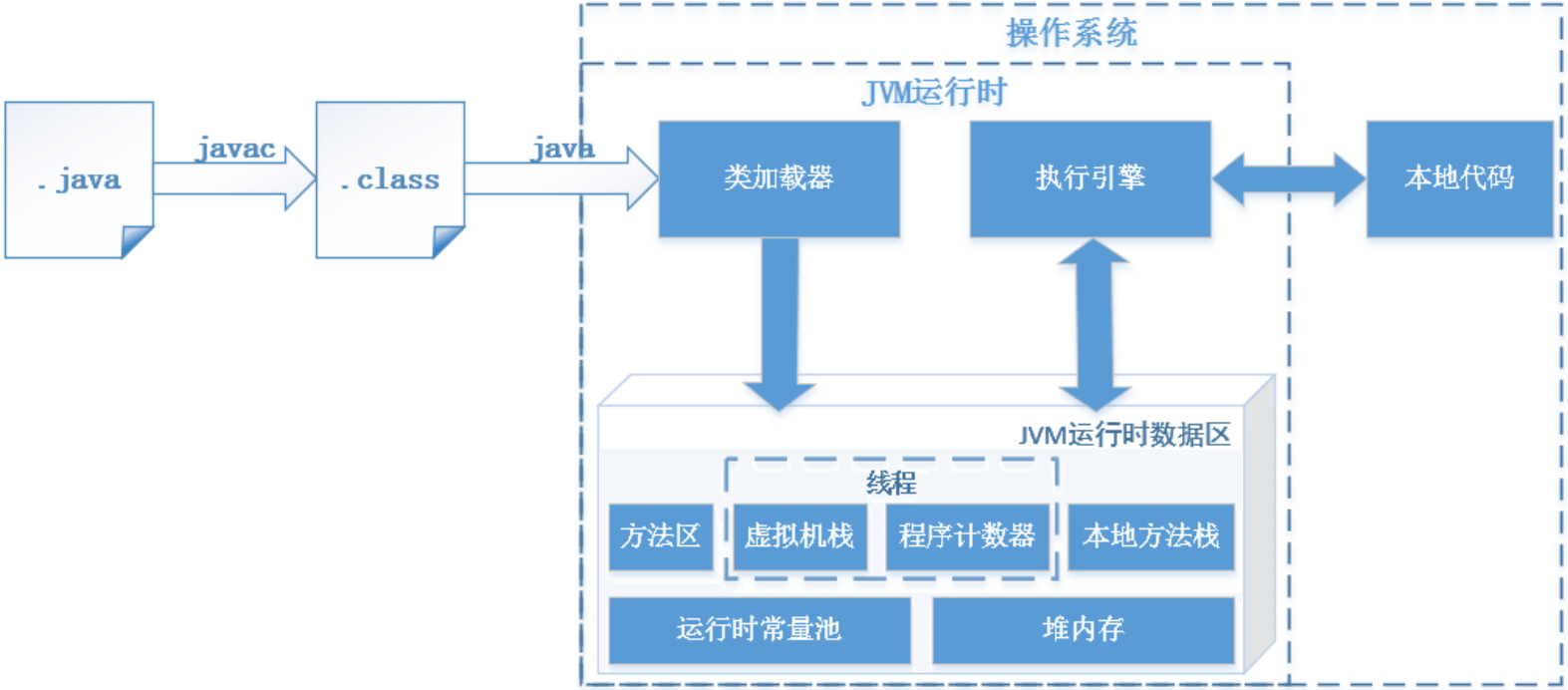
主要JVM及区别

- HotSpot JVM
- Jrockit JVM
- IBM JVM

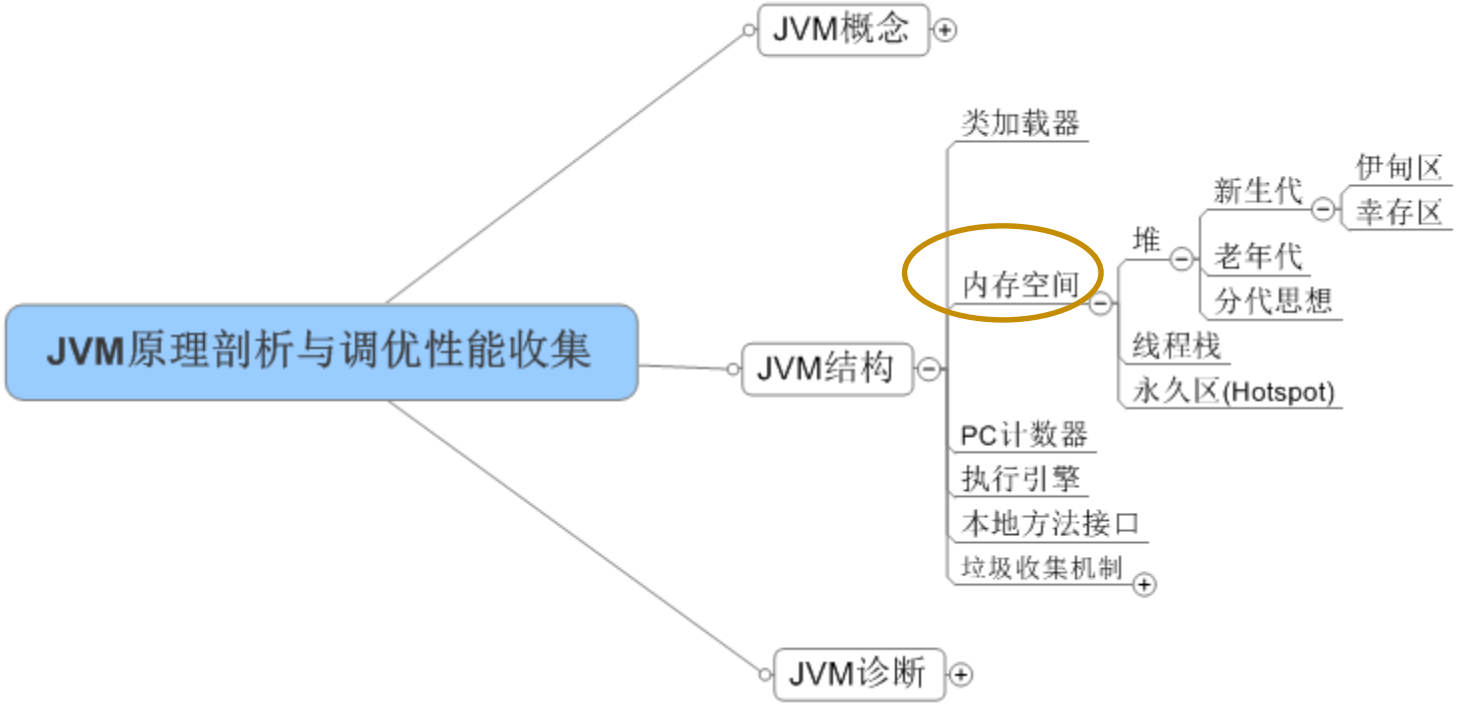
JVM结构


JVM原理剖析与调优性能收集
JVM概念
JVM结构
类加载器
内存空间(Hotspot)
PC计数器
执行引擎
本地方法接口
垃圾收集机制
JVM诊断

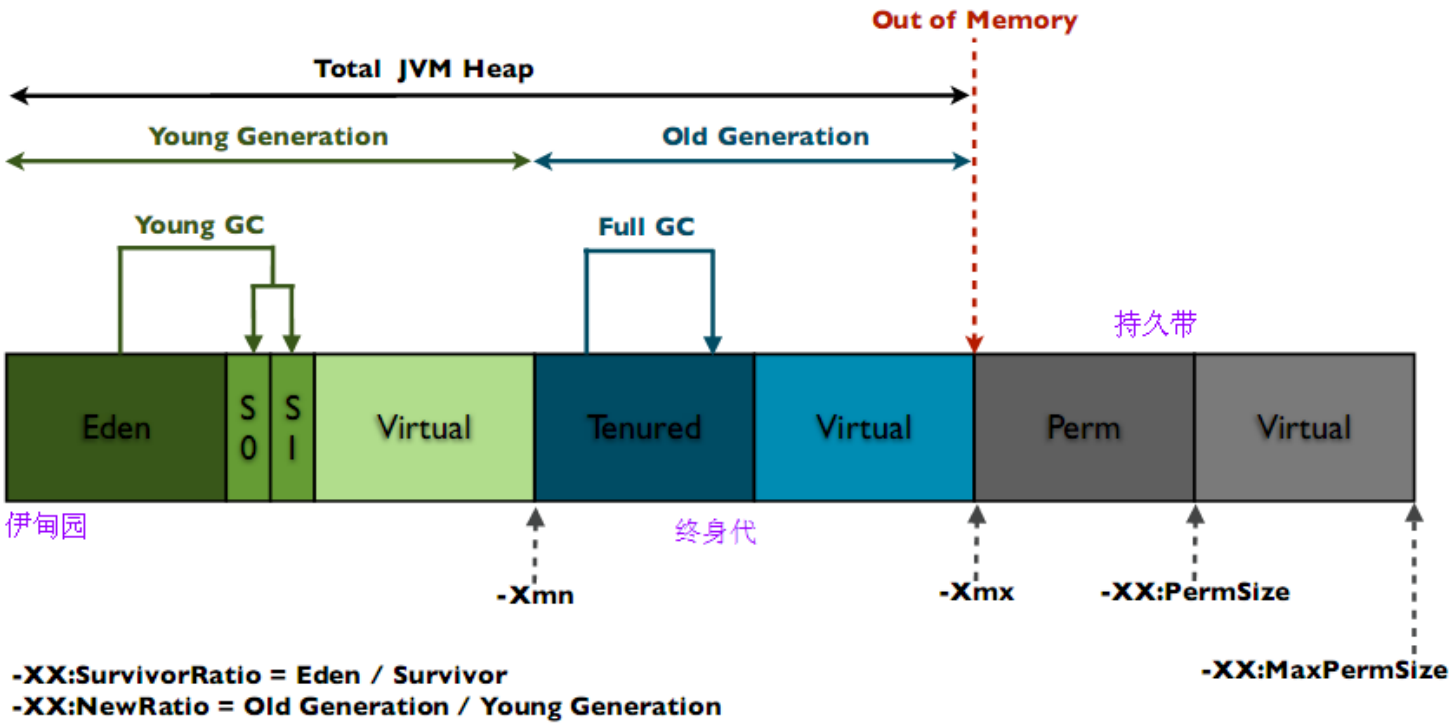
JVM结构图


操作系统
JVM运行时
.java
javac
.class
java
类加载器
执行引擎
本地代码
JVM运行时数据区
线程
方法区
虚拟机栈
程序计数器
本地方法栈
运行时常量池
堆内存

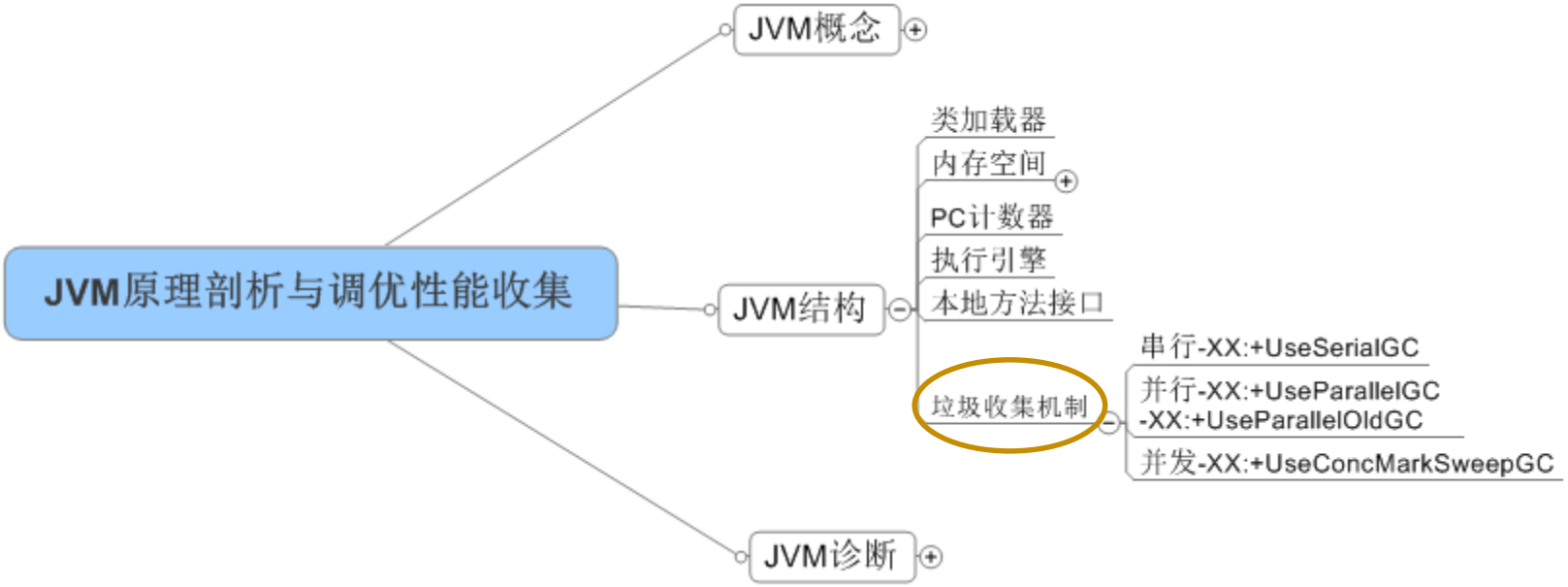
内存结构


JVM原理剖析与调优性能收集
JVM概念
JVM结构
JVM诊断
类加载器
内存空间
堆
新生代
伊甸区
幸存区
老年代
分代思想
线程栈
永久区(Hotspot)
PC计数器
执行引擎
本地方法接口
垃圾收集机制

内存分代管理(Generation)



垃圾收集机制


JVM原理剖析与调优性能收集
JVM概念
JVM结构
类加载器
内存空间
PC计数器
执行引擎
本地方法接口
垃圾收集机制
串行-XX:+UseSerialGC
并行-XX:+UseParallelGC
-XX:+UseParallelOldGC
并发-XX:+UseConcMarkSweepGC
JVM诊断

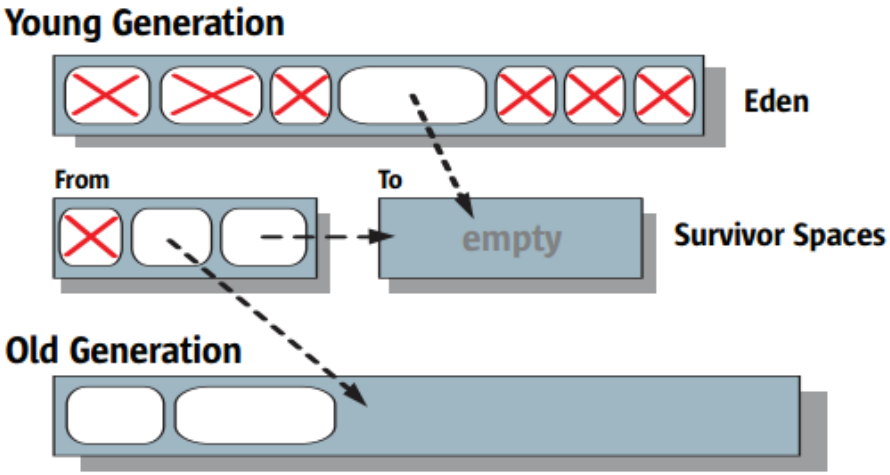
为什么要垃圾回收

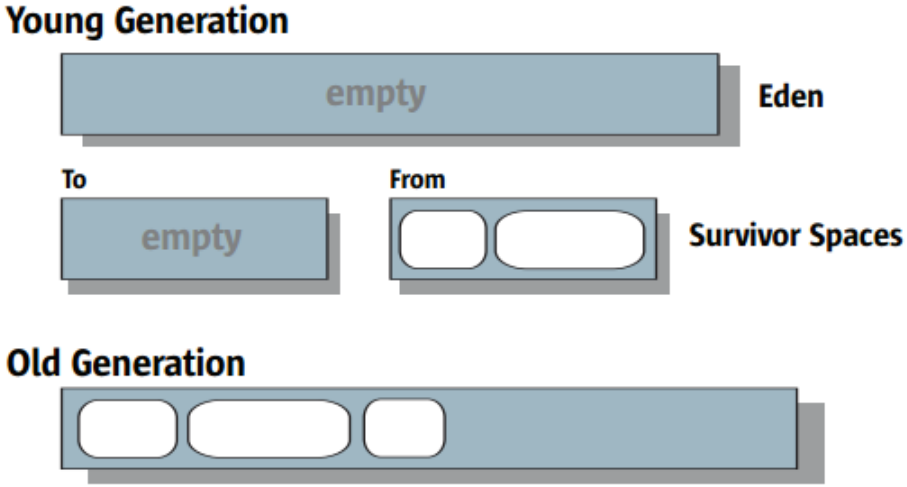
GC算法与StopTheWorld

- 复制算法-将原有的内存空间分为两块，两块空间完全相同，每次只使用其中一块，在垃圾回收时，将正在使用的内存中的存活对象复制到未使用的内存块中，之后，清除正在使用的内存块中的所有对象，交换两个内存的角色，完成垃圾回收
- 标记清除-清除算法将垃圾回收分为两个阶段：标记阶段和清除阶段。一种可行的实现是，在标记阶段，首先通过根节点，标记所有从根节点开始的可达对象。因此，未被标记的对象就是未被引用的垃圾对象。然后，在清除阶段，清除所有未被标记的对象。
- 标记清除压缩-适合用于存活对象较多的场合，如老年代。它在标记-清除算法的基础上做了一些优化。和标记-清除算法一样，标记-压缩算法也首先需要从根节点开始，对所有可达对象做一次标记。但之后，它并不简单的清理未标记的对象，而是将所有的存活对象压缩到内存的一端。之后，清理边界外所有的空间。
- Stop-The-World，JAVA中一种全局暂停的现象

垃圾收集机制-串行

-XX:+UseSerialGC
新生代的垃圾回收机制：使用复制算法

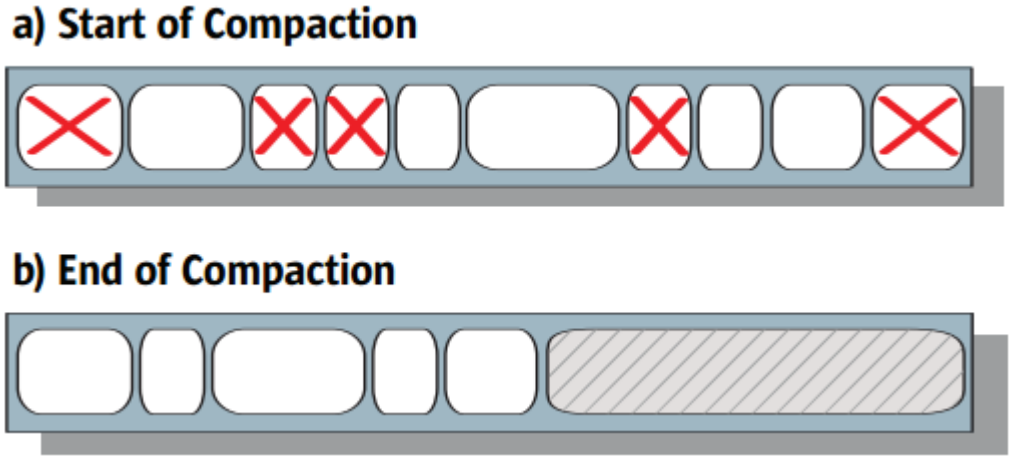




垃圾收集机制-串行

-XX:+UseSerialGC

老年代的垃圾回收机制：标记清除压缩算法

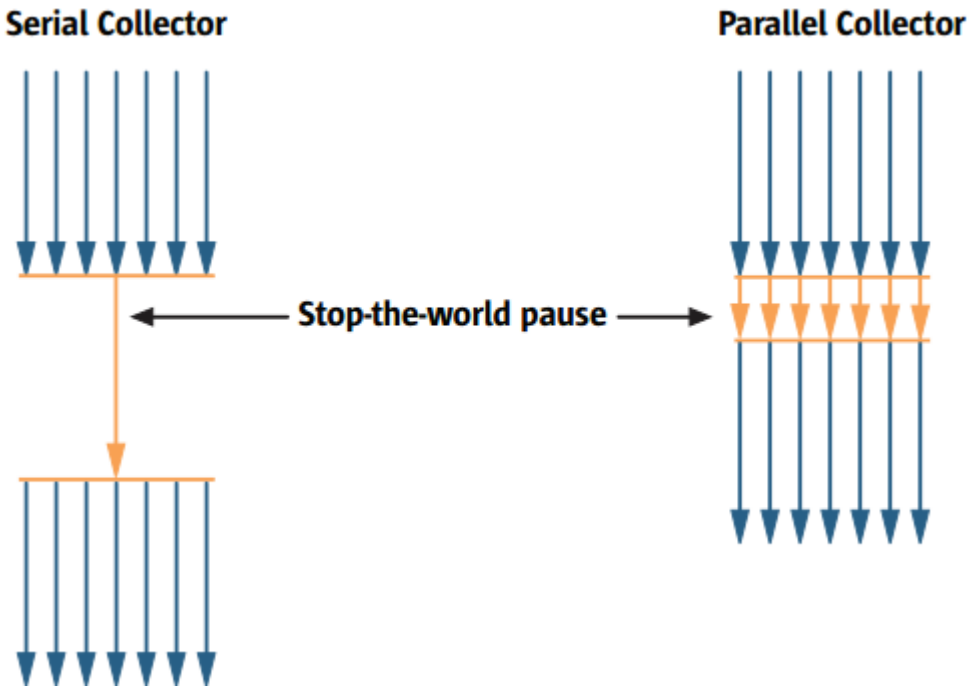


垃圾收集机制-并行

-XX:+UseParallelGC,**-XX:+UseParallelOldGC**

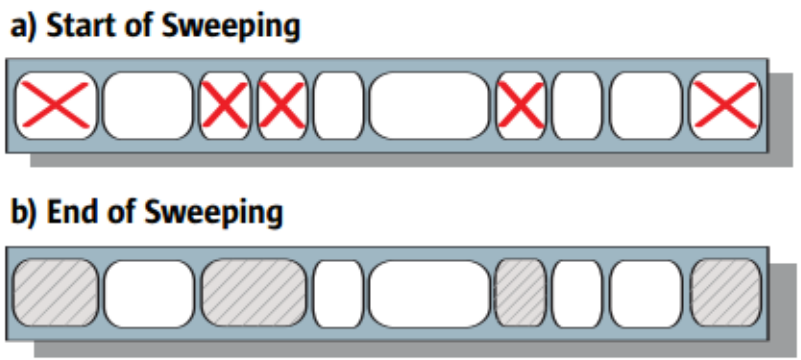
老年代的垃圾回收机制：

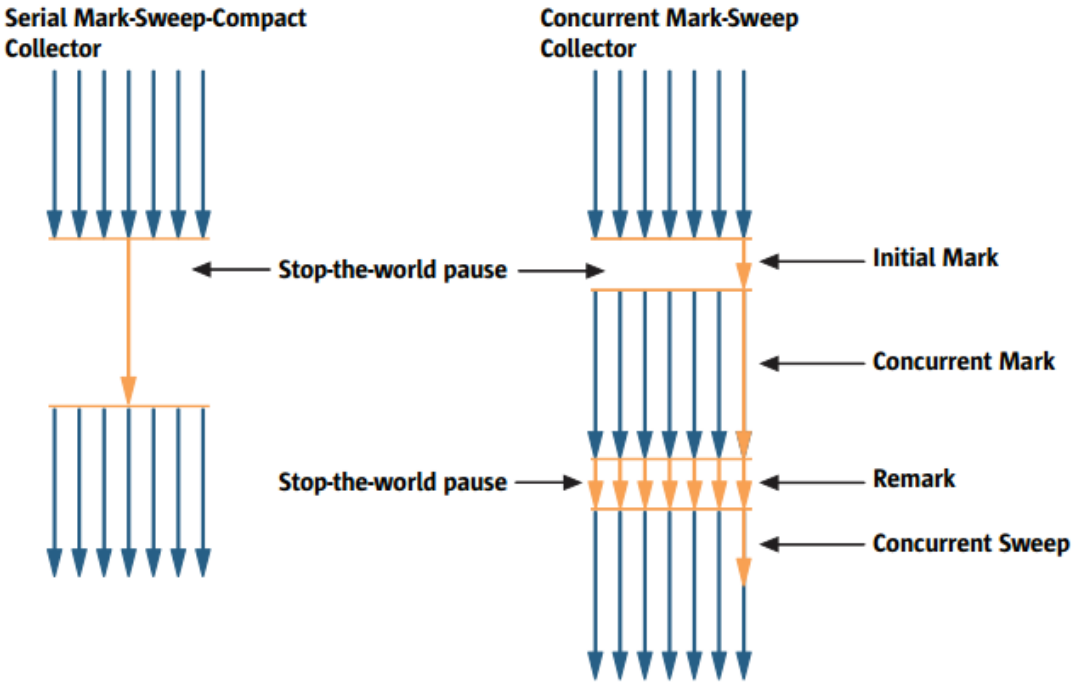
标记清除压缩算法



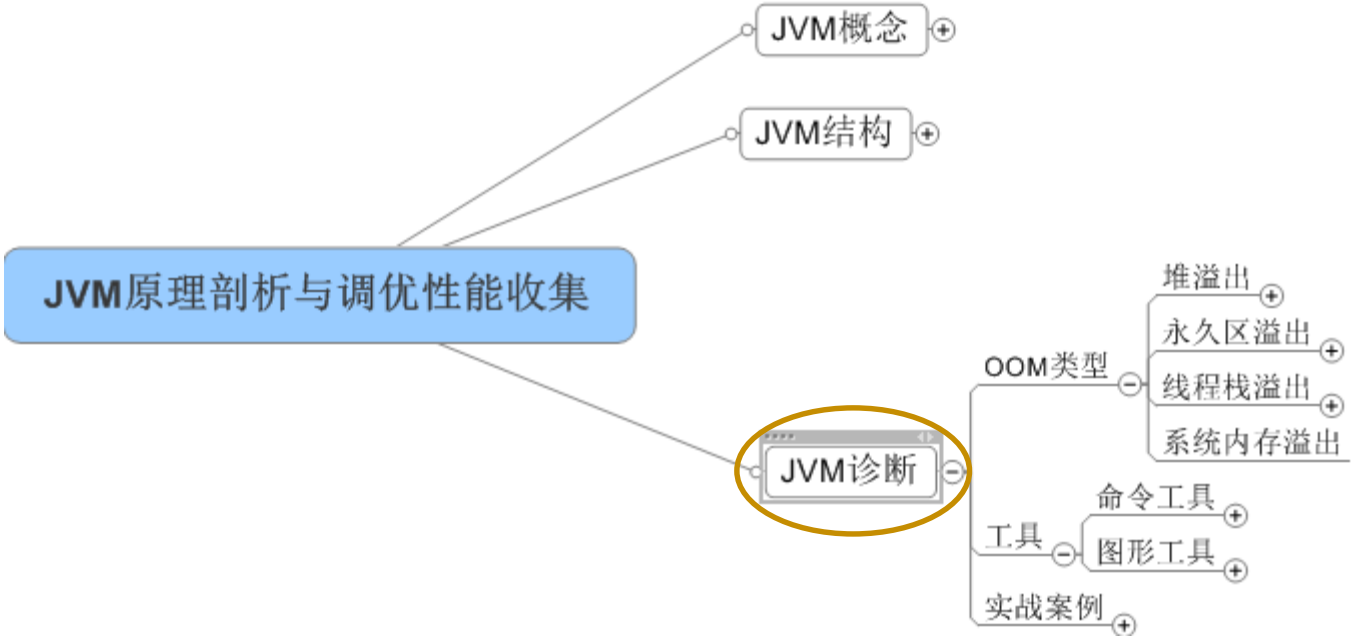
垃圾收集机制-并发

-XX:+UseConcMarkSweepGC
老年代的垃圾回收机制：
标记清除算法





JVM诊断


JVM原理剖析与调优性能收集
JVM概念
JVM结构
JVM诊断
OOM类型
堆溢出
永久区溢出
线程栈溢出
系统内存溢出
工具
命令工具
图形工具
实战案例

OutOfMemoryError： PermGen space

```
        at sun.reflect.GeneratedMethodAccessor13.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
        Truncated. see log file for complete stacktrace
>
<2014-12-19 下午03时35分52秒 CST> <Error> <HTTP> <BEA-101362> <[ServletContext@1677168094[app:jtitsm_web module:WebContent path: spe
c-version:null]] could not deserialize the servlet-context scoped attribute with name: "AxisEngine"
java.io.NotSerializableException: org.apache.axis.configuration.FileProvider
        at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1164)
        at java.io.ObjectOutputStream.defaultWriteFields(ObjectOutputStream.java:1518)
        at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1483)
        at java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1400)
        at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1158)
        Truncated. see log file for complete stacktrace
>
<2014-12-19 下午03时37分06秒 CST> <Warning> <HTTP> <BEA-101162> <User defined listener org.springframework.web.context.ContextLoader
Listener failed: java.lang.OutOfMemoryError: PermGen space.
java.lang.OutOfMemoryError: PermGen space
>
<2014-12-19 下午03时37分06秒 CST> <Error> <Socket> <BEA-000405> <Uncaught Throwable in processSockets
 java.lang.OutOfMemoryError: PermGen space.
java.lang.OutOfMemoryError: PermGen space
        at sun.misc.Unsafe.defineClass(Native Method)
        at sun.reflect.ClassDefiner.defineClass(ClassDefiner.java:45)
        at sun.reflect.MethodAccessorGenerator$1.run(MethodAccessorGenerator.java:381)
        at java.security.AccessController.doPrivileged(Native Method)
        at sun.reflect.MethodAccessorGenerator.generate(MethodAccessorGenerator.java:377)
        Truncated. see log file for complete stacktrace
>
<2014-12-19 下午03时37分06秒 CST> <Warning> <Socket> <BEA-000449> <Closing socket as no data read from it on 192.168.17.29:58,542 du
```

OutOfMemoryError：unable to create new native thread

```
Thread1240 created
Thread1241 created
Thread1242 created
Thread1243 created
Thread1244 created
Thread1245 created
Exception in thread "main" java.lang.OutOfMemoryError: unable to create new native thread
        at java.lang.Thread.start0(Native Method)
        at java.lang.Thread.start(Thread.java:640)
        at Main.main(Main.java:16)
```

OutOfMemoryError：Java heap space

```
Exception in thread "Session.26" java.lang.OutOfMemoryError: Java heap space
        at TestThread.useHeapMem(TestThread.java:29)
        at TestThread.run(TestThread.java:14)
        at java.lang.Thread.run(Thread.java:662)
Session.27
Exception in thread "Session.27" java.lang.OutOfMemoryError: Java heap space
        at TestThread.useHeapMem(TestThread.java:29)
        at TestThread.run(TestThread.java:14)
        at java.lang.Thread.run(Thread.java:662)
Session.28
Exception in thread "Session.28" java.lang.OutOfMemoryError: Java heap space
        at TestThread.useHeapMem(TestThread.java:29)
        at TestThread.run(TestThread.java:14)
        at java.lang.Thread.run(Thread.java:662)
Session.29
Exception in thread "Session.29" java.lang.OutOfMemoryError: Java heap space
        at TestThread.useHeapMem(TestThread.java:29)
        at TestThread.run(TestThread.java:14)
        at java.lang.Thread.run(Thread.java:662)
```

调整参数的方法

- 静态调整：在jvm启动的时候，带入参数。

```
C:\Users\roosejay>jinfo -flags 6788
Attaching to process ID 6788, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 24.0-b56

-Xms24m -Xmx256m -XX:MaxPermSize=96m -Dsun.jvmstat.perdata.syncWaitMs=10000 -Dsun.java2d.noddraw=tr
e -Dsun.java2d.d3d=false -Dnetbeans.keyring.no.master=true -Djdk.home=D:\程序\JDK -Dnetbeans.home=D
\程序\JDK\lib\visualvm\platform -Dnetbeans.user=C:\Users\roosejay\AppData\Roaming\VisualVM\7u14 -Dn
tbeans.default_userdir_root=C:\Users\roosejay\AppData\Roaming\VisualVM -XX:+HeapDumpOnOutOfMemoryEr
or -XX:HeapDumpPath=C:\Users\roosejay\AppData\Roaming\VisualVM\7u14\var\log\heapdump.hprof -Dnetbea
s.system_http_proxy=DIRECT -Dsun.awt.keepWorkingSetOnMinimize=true -Dnetbeans.dirs=D:\程序\JDK\lib\
isualvm\visualvm;D:\程序\JDK\lib\visualvm\profiler
```

- 动态调整：使用工具在jvm运行的时候，进行调整。使用jinfo等工具进行调整。(只能调整部分参数)

```
C:\Users\roosejay>jinfo -flag +HeapDumpBeforeFullGC  6788
```

```
C:\Users\roosejay>jinfo -flag +HeapDumpBeforeFullGC  6964
Exception in thread "main" java.io.IOException: Command failed in target VM
        at sun.tools.attach.WindowsVirtualMachine.execute(WindowsVirtualMachine.java:112)
        at sun.tools.attach.HotSpotVirtualMachine.executeCommand(HotSpotVirtualMachine.java:217)
        at sun.tools.attach.HotSpotVirtualMachine.setFlag(HotSpotVirtualMachine.java:190)
        at sun.tools.jinfo.JInfo.flag(JInfo.java:129)
        at sun.tools.jinfo.JInfo.main(JInfo.java:76)
```

堆内存

- -Xms*n*
  - 默认值：运行时自动控制；JDK 5为2M；启用并行GC时大于1/64物理内存或1G
  - 堆空间初始大小，1M的整数倍，大于1M，，
- -Xmx*n*
  - 默认值：运行时自动控制；JDK 5为64M；启用并行GC时小于1/4物理内存或1G
  - 堆空间最大大小，1M的整数倍，大于2M；
- –XX:MinHeapFreeRatio=*minimum*
  - 默认：40
  - 堆空间的空闲比率小于下限时，将自动扩展
- –XX:MaxHeapFreeRatio=maximum
  - 默认：70
  - 堆空间的空闲比率大于上限时，将自动收缩

分代

- –XX:NewSize=*n*或-Xm*size*
  - 默认值：根据平台自动设定
  - 年轻代的大小
- –XX:NewRatio=n
  - 默认值：client：2；server：8
  - 年老除以年轻代的倍数。
- –XX:SurvivorRatio=*n*
  - 默认值：32
  - Eden区除以一个Survivor区的倍数。
- –XX:MaxPermSize=*n*
  - 默认值：根据平台自动设定
  - 持久代的最大空间大小

GC类型

- –XX:+UseSerialGC
  - 串行GC
- –XX:+UseParallelGC
  - 并行GC
- –XX:+UseParallelOldGC
  - 并行 + 压缩，
- –XX:+UseConcMarkSweepGC
  - Concurrent mark–sweep (CMS)，并发GC
- 默认GC的选择：
  - Client模式启用：-XX:+UseSerialGC
  - Server模式启用：-XX:+UseParallelGC
  - -XX:+AggressiveOpts
    - 开启推荐的参数项

GC统计数据

- -verbos:gc
  - 将GC信息显示到标准输出
- -Xloggc:file
  - 将GC信息输出到文件
- –XX:+PrintGC
  - 输出每次GC的基本信息（默认）
- -XX:+PrintGCDetails
  - 输出每次GC的详细信息
- -XX:+PrintGCTimeStamps
  - 在GC开始时，输出时间

用于并行(压缩)GC

- –XX:ParallelGCThreads=*n*
  - 默认值：CPU数
  - 垃圾收集的线程数
- –XX:MaxGCPauseMillis=*n*
  - 默认值：无
  - 最大GC停顿时间（毫秒即1/1000秒）
- –XX:GCTimeRatio=*n*
  - 默认值：99
  - 垃圾收集时间比率，占程序运行总时间的1/(1+n)

用于CMS GC

- –XX:+CMSIncrementalMode或-Xincgc
  - 默认值：禁用
  - 启用后，并发GC将周期性停止，以便程序更好的运行
- –XX:+CMSIncrementalPacing
  - 默认值：禁用
  - 启用后，基于应用程序行为，自动控制CMS GC在停止前允许进行的工作总量。
- –XX:ParallelGCThreads=*n*
  - 与并行GC相同

其他

- JIT相关参数
  - -Xint
    - 在运行时，仅执行解析，不编译
  - -XX:+PrintCompilation
    - 显示运行时编译情况
  - -Xprof
    - 将程序Profile信息(解析时间、编译时间)打印到标准输出
- 其他
  - -Xshare:*auto|on|off*
    - 是否使用共享的类信息（通常用于Client端加快JVM启动）
  - -verbose:class （默认）
    - Display information about each class loaded.

JVM诊断工具-文本工具

jps

- jps：显示进程号和简短名称

```
D:\>jps
4480 Jps
2196 Main
```

- jps -l：显示进程号和完整路径

```
D:\>jps -l
2196 com.sun.tools.hat.Main
928 sun.tools.jps.Jps
```

- jps -m：显示传递给Main方法的参数

```
D:\>jps -m
4896 Jps -m
2196 Main d:\java_pid18448.hprof
```

- jps -q：仅显示进程号

```
D:\>jps -q
3192
2196
```

- jps -v：显示传递给JVM的参数

```
D:\>jps -v
2196 Main -Denv.class.path=.;D:\程序\JDK\lib\dt.jar;D:\程序\JDK\lib\tools.jar; -Dapplication.home=D:
\jdk1.6.0_27 -Xms8m
3108 Jps -Denv.class.path=.;D:\程序\JDK\lib\dt.jar;D:\程序\JDK\lib\tools.jar; -Dapplication.home=D:\
程序\JDK -Xms8m
```

jstat

Jstat的使用

```
D:\>jstat -options
-class
-compiler
-gc
-gccapacity
-gccause
-gcnew
-gcnewcapacity
-gcold
-gcoldcapacity
-gcpermcapacity
-gcutil
-printcompilation
```

查看载入的类

```
D:\>jstat -class 2196 1000
Loaded  Bytes  Unloaded  Bytes     Time
   714   841.5        0     0.0     1.00
   714   841.5        0     0.0     1.00
   714   841.5        0     0.0     1.00
   714   841.5        0     0.0     1.00
```

查看新生代GC情况

```
D:\>jstat -gcnew 2196 1000
 S0C    S1C    S0U    S1U   TT MTT  DSS      EC       EU     YGC     YGCT
 256.0  256.0    0.0   36.7 15  15  128.0   2176.0   650.9     25    0.050
 256.0  256.0    0.0   36.7 15  15  128.0   2176.0   650.9     25    0.050
```

查看所有GC情况

```
D:\>jstat -gcutil 2196 3000
  S0     S1     E      O      P     YGC     YGCT    FGC    FGCT     GCT
  0.00  14.32  29.91  95.81  33.95     25    0.050     0    0.000    0.050
  0.00  14.32  29.91  95.81  33.95     25    0.050     0    0.000    0.050
  0.00  14.32  29.91  95.81  33.95     25    0.050     0    0.000    0.050
  0.00  14.32  29.91  95.81  33.95     25    0.050     0    0.000    0.050
  0.00  14.32  29.91  95.81  33.95     25    0.050     0    0.000    0.050
```

jinfo

- jinfo 48608

```
Attaching to process ID 48608, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 23.7-b01
Java System Properties:
java.runtime.name = Java(TM) SE Runtime Environment
```

- jinfo –flags 48608：查看设置

```
Attaching to process ID 48608, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 23.7-b01
-Xmx25m -Xms25m
```

- jinfo -flag MaxPermSize 48608：查看某个的配置

```
-XX:MaxPermSize=67108864
```

jmap

- jmap –histo <pid> 对象实例的统计

```
D:\jdk1.6.0_27\bin>jmap -histo 2196

 num     #instances         #bytes  class name
----------------------------------------------
   1:         20929        3888416  [C
   2:          9434        1035800  <constMethodKlass>
   3:         16278         761808  <symbolKlass>
   4:          9434         757952  <methodKlass>
   5:           714         430544  <constantPoolKlass>
   6:         16925         406200  java.lang.String
   7:         16562         397488  java.util.Hashtable$Entry
   8:           714         300304  <instanceKlassKlass>
   9:           659         289000  <constantPoolCacheKlass>
  10:           958         264872  [B
```

- jmap –dump:format=b,file=heap.bin <pid> 获取堆的信息，二进制文件

```
D:\jdk1.6.0_27\bin>jmap -dump:format=b,file=beap.bin 2196
Dumping heap to D:\jdk1.6.0_27\bin\beap.bin ...
Heap dump file created
```

jstack

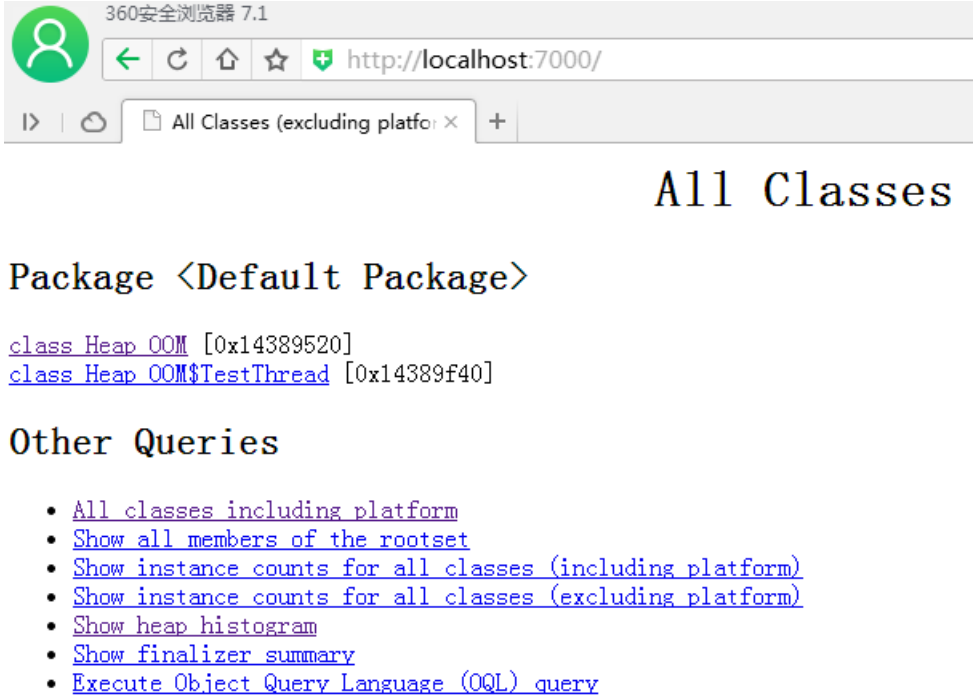
- jstack <pid>获取栈信息，查看是否有锁
- 使用系统信号通知进行，打印CoreDump到标准输出
  - Linux下：kill -3 <pid>
  - Windows下：Ctrl-Beak

```
D:\jdk1.6.0_27\bin>jstack  2196
2014-12-28 14:20:16
Full thread dump Java HotSpot(TM) Client VM (20.2-b06 mixed mode):

"DestroyJavaVM" prio=6 tid=0x006fa000 nid=0xa18 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"Query Listener" prio=6 tid=0x187ccc00 nid=0x1660 runnable [0x188af000]
   java.lang.Thread.State: RUNNABLE
        at java.net.PlainSocketImpl.socketAccept(Native Method)
        at java.net.PlainSocketImpl.accept(PlainSocketImpl.java:408)
        - locked <0x09916a10> (a java.net.SocksSocketImpl)
        at java.net.ServerSocket.implAccept(ServerSocket.java:462)
        at java.net.ServerSocket.accept(ServerSocket.java:430)
        at com.sun.tools.hat.internal.server.QueryListener.waitForRequests(QueryListener.java:76)
        at com.sun.tools.hat.internal.server.QueryListener.run(QueryListener.java:65)
        at java.lang.Thread.run(Thread.java:662)
```

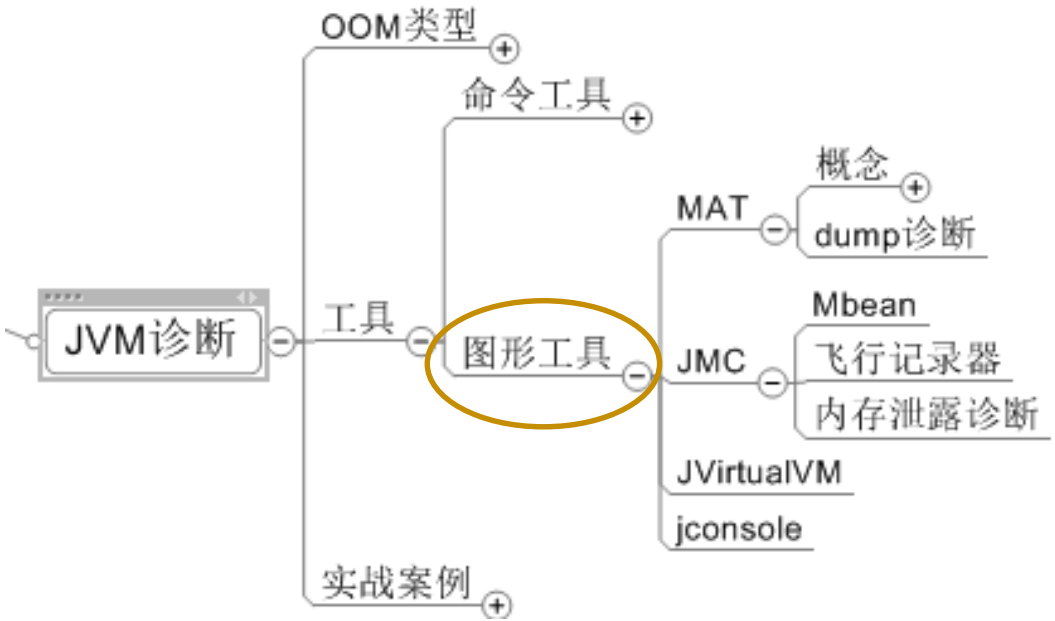
jhat

使用jhat xxx.hprof（堆分析工具）启动，通过web浏览器进行访问

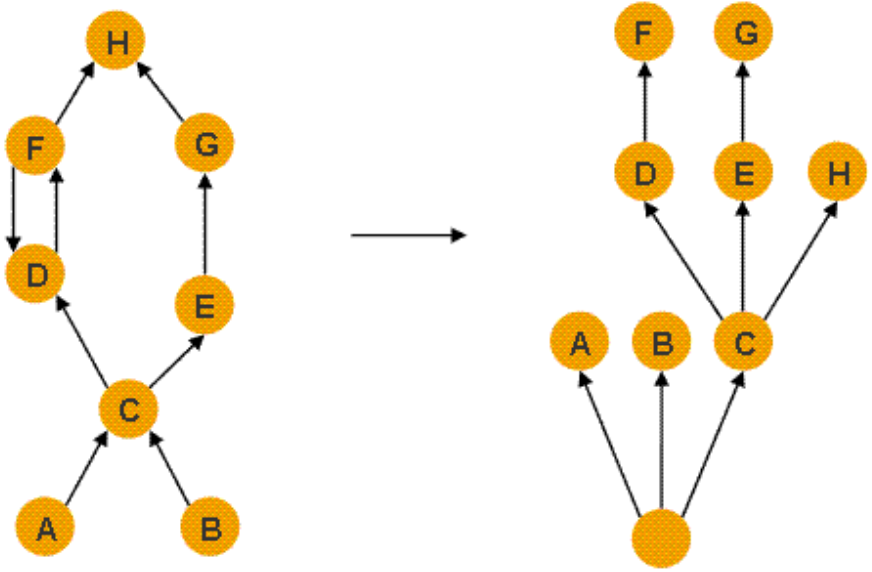
```
D:\jdk1.6.0_27\bin>jhat d:\java_pid18448.hprof
Reading from d:\java_pid18448.hprof...
Dump file created Sat Dec 27 15:18:55 CST 2014
Snapshot read, resolving...
Resolving 7405 objects...
Chasing references, expect 1 dots.
Eliminating duplicate references.
Snapshot resolved.
Started HTTP server on port 7000
Server is ready.
```



JVM诊断工具-图形工具



工具涉及的几个概念-支配树



对象引用图

支配树

在对象引用图中，所有指向对象B的路径都经过对象A，
则认为对象A支配对象B
如果对象A是离对象B最近的一个支配对象，
则认为对象A为对象B的直接支配者

支配者被回收，被支配对象也被回收

工具涉及的几个概念-浅堆与深堆

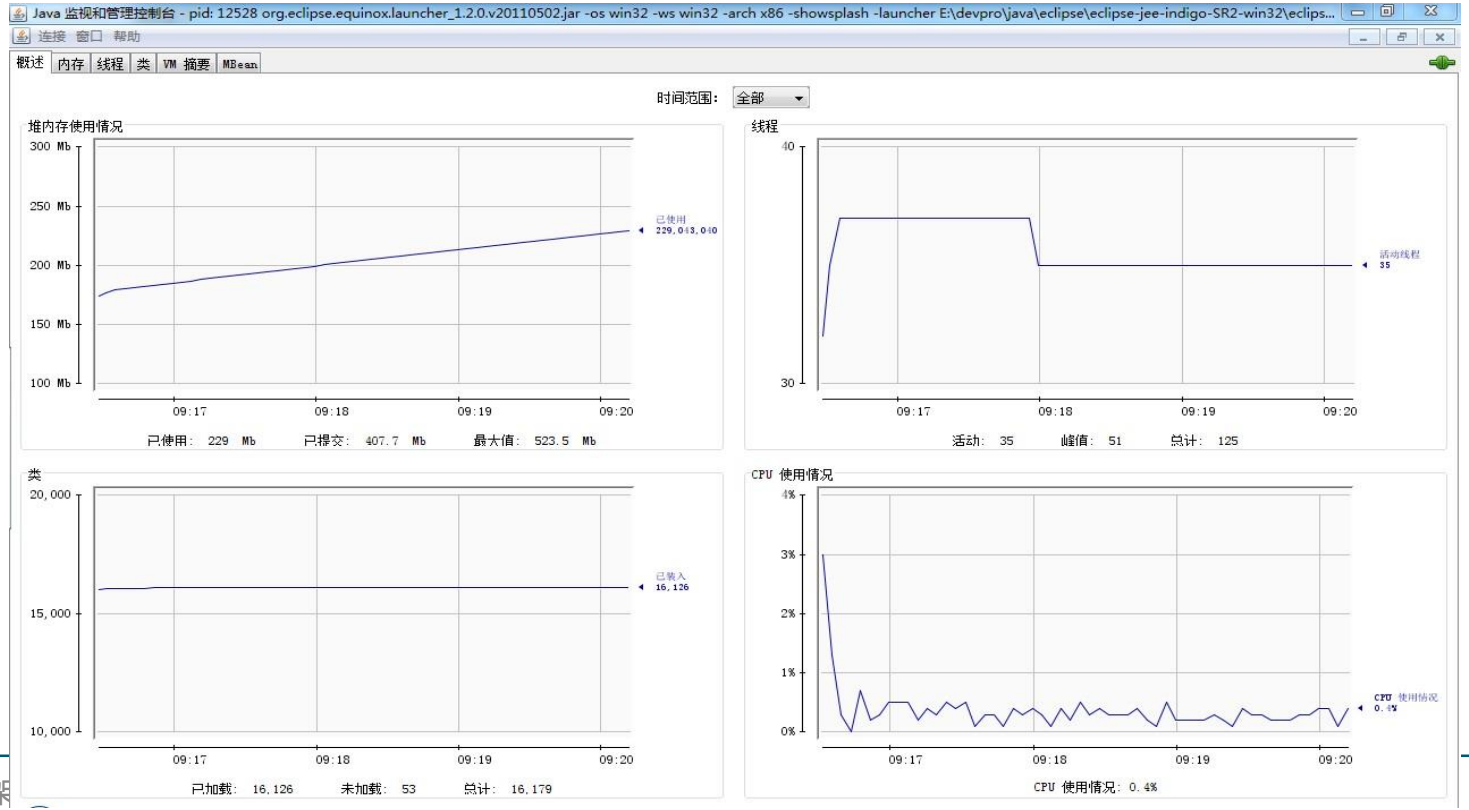
- 浅堆
  - 一个对象结构所占用的内存大小
  - 对象大小按照8字节对齐
  - 浅堆大小和对象的内容无关，只和对象的结构有关
- 深堆
  - 一个对象被GC回收后，可以真实释放的内存大小
  - 只能通过对象访问到的（直接或者间接）所有对象的浅堆之和 （支配树）

Jconsole

- JConsole：JDK自带基于JMX客户端的监控工具
  - 标准JMX管理客户端
  - VM整体摘要信息
  - 内存区监控
  - 线程监控
  - 类加载概况
- 程序命令：%JAVA_HOME%/bin/jconsole.exe

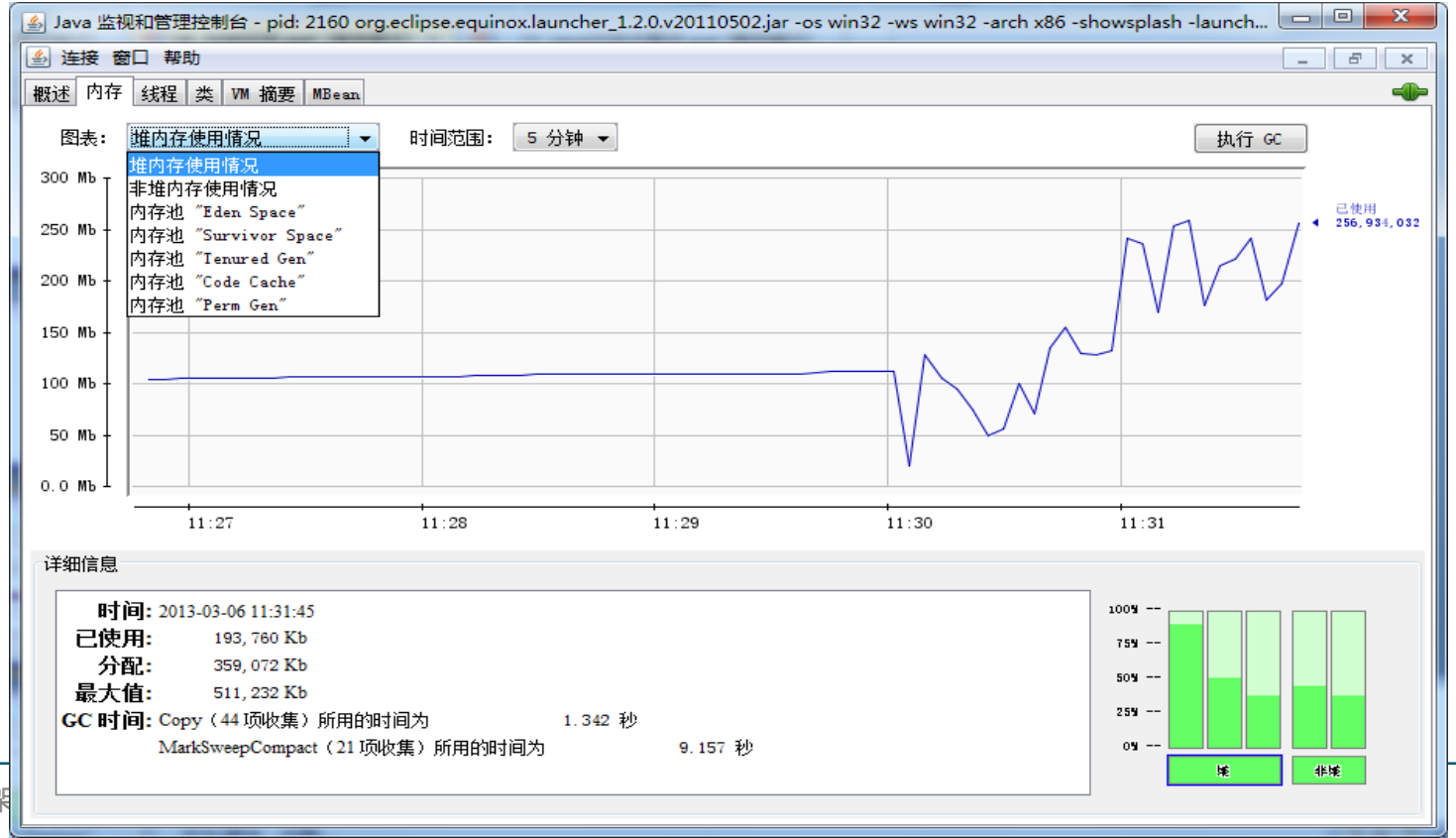
Jconsole示例-1/5

- 综合监控：内存量、线程数、类数量、CPU使用率



Jconsole示例-2/5

- 分代内存使用情况监控



Jconsole示例-3/5

- 线程监控和堆栈追踪

Java 监视和管理控制台 - pid: 2160 org.eclipse.equinox.launcher_1.2.0.v20110502.jar -os win32 -ws win32 -arch x86 -showsplash -launch...

连接 窗口 帮助

概述 | 内存 | 线程 | 类 | VM 摘要 | MBean

时间范围: 10 分钟

线程的数目

80 70 60 50 40 30 20 10

11:25 11:26 11:27 11:28 11:29 11:30 11:31 11:32 11:33 11:34

峰 71

活动线程 53

线程

[Timer] - Main Queue Handler
Worker-JM
RMI TCP Accept-0
RMI Scheduler (0)
JMX server connection timeout
RMI TCP Connection(3)-134.128.
RMI TCP Connection(5)-134.128.
[ThreadPool Manager] - Idle Th
Bundle File Closer
Java indexing

名称: RMI TCP Connection(5)-134.128.63.12
状态: TIMED_WAITING 在 com.sun.jmx.remote.internal.ArrayNotificationBuffer@e222eb 上
阻塞总数: 0 等待总数: 9

堆栈追踪:
java.lang.Object.wait(Native Method)
com.sun.jmx.remote.internal.ArrayNotificationBuffer.fetchNotifications(ArrayNotificationBuffer.
com.sun.jmx.remote.internal.ArrayNotificationBuffer$ShareBuffer.fetchNotifications(ArrayNotifica
com.sun.jmx.remote.internal.ServerNotifForwarder.fetchNotifs(ServerNotifForwarder.java:258)

Filter 检测到死锁

Jconsole示例-4/5

- VM摘要

Java 监视和管理控制台 - pid: 2160 org.eclipse.equinox.launcher_1.2.0.v20110502.jar -os win32 -ws win32 -arch x86 -showsplash -launcher E:\devpro\eclipse\eclipse-jee-indigo-SR2-win32\eclipse.exe -...

连接 窗口 帮助

概述 | 内存 | 线程 | 类 | VM 摘要 | MBean

VM 摘要

2013年3月6日 星期三 上午11时36分29秒 GMT+08:00

连接名称: pid: 2160 org.eclipse.equinox.launcher_1.2.0.v20110502.jar -os win32 -ws win32 -arch x86 -showsplash -launcher E:\devpro\eclipse\eclipse-jee-indigo-SR2-win32\eclipse.exe -name Eclipse --launcher.library E:\devpro\eclipse\eclipse-jee-indigo-SR2-win32\plugins\org.eclipse.equinox.launcher.win32.win32.x86_1.1.100.v20110502\eclipse_1406.dll -startup E:\devpro\eclipse\eclipse-jee-indigo-SR2-win32\plugins\org.eclipse.equinox.launcher_1.2.0.v20110502.jar --launcher.overrideVmargs -exitdata 1f5c_50 -product org.eclipse.epp.package.jee.product -vm E:\devpro\java\jdk\jdk1.6.0_12\jre\bin\javaw.exe -vmargs -Dosgi.requiredJavaVersion=1.5 -Xmn128m -Xms256m -Xmx512m -Dsun.lang.ClassLoader.allowArraySyntax=true -javaagent:lombok.jar -Xbootclasspath/a:lombok.jar -XX:MaxPermSize=256m -jar E:\devpro\eclipse\eclipse-jee-indigo-SR2-win32\plugins\org.eclipse.equinox.launcher_1.2.0.v20110502.jar

虚拟机: Java HotSpot(TM) Client VM 版本 11.2-b01

供应商: Sun Microsystems Inc.

名称: 2160@SHIW

正常运行时间: 11 minutes

处理 CPU 时间: 1 minute

JIT 编译器: HotSpot Client Compiler

编译总时间: 7.322 秒

活动线程: 53

峰: 71

守护线程: 44

已启动的线程总数: 178

当前类已装入: 20, 777

已装入类的总数: 20, 998

已卸载类的总数: 221

当前堆大小: 188, 656 Kb

堆大小的最大值: 511, 232 Kb

垃圾收集器: Name = 'Copy', Collections = 46, Total time spent = 1.472 秒

垃圾收集器: Name = 'MarkSweepCompact', Collections = 21, Total time spent = 9.157 秒

分配的内存: 359, 072 Kb

暂挂结束操作: 0 个对象

操作系统: Windows Vista 6.1

体系结构: x86

处理器的数目: 4

分配的虚拟内存: 543, 740 Kb

物理内存总量: 2, 097, 151 Kb

可用物理内存: 308, 084 Kb

交换空间总量: 4, 194, 303 Kb

可用交换空间: 998, 976 Kb

VM 参数: -Dosgi.requiredJavaVersion=1.5 -Xmn128m -Xms256m -Xmx512m -Dsun.lang.ClassLoader.allowArraySyntax=true -javaagent:lombok.jar -Xbootclasspath/a:lombok.jar -XX:MaxPermSize=256m

类路径: E:\devpro\eclipse\eclipse-jee-indigo-SR2-win32\plugins\org.eclipse.equinox.launcher_1.2.0.v20110502.jar;lombok.jar

库路径: E:\devpro\java\jdk\jdk1.6.0_12\jre\bin;.;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;E:\devpro\Oracle\product\10.2.0\client_1\bin;C:\Program Files\Common Files\NetSarang;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Lenovo\Bluetooth Software\;C:\Program Files\EgisTec BioExcess\;C:\Program Files\Intel\WiFi\bin\;C:\Program Files\Common Files\Intel\WirelessCommon\;C:\Program Files\TortoiseSVN\bin;C:\Program Files\Microsoft Windows Performance Toolkit\;C:\Program Files\Pandoc\bin;C:\Program Files\TortoiseGit\bin

引导类路径: E:\devpro\java\jdk\jdk1.6.0_12\jre\lib\resources.jar;E:\devpro\java\jdk\jdk1.6.0_12\jre\lib\rt.jar;E:\devpro\java\jdk\jdk1.6.0_12\jre\lib\sunrsasign.jar;E:\devpro\java\jdk\jdk1.6.0_12\jre\lib\jsse.jar;E:\devpro\java\jdk\jdk1.6.0_12\jre\lib\jce.jar;E:

Jconsole示例-5/5

- JMX客户端

Java 监视和管理控制台 - pid: 12528 org.eclipse.equinox.launcher_1.2.0.v20110502.jar -os win32 -ws win32 -arch x86 -showsplash -launcher E...

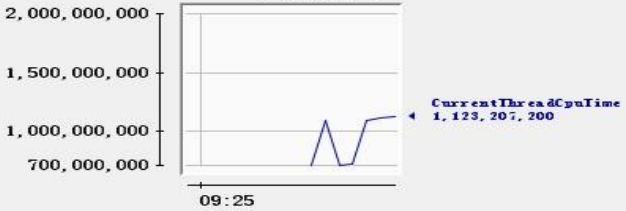
连接 窗口 帮助

概述 内存 线程 类 VM 摘要 MBean

- JMImplementation
- com.sun.management
- java.lang
  - ClassLoading
  - Compilation
  - GarbageCollector
  - Memory
    - 属性
      - Verbose
      - HeapMemoryUsage
      - NonHeapMemoryUsage
      - ObjectPendingFinaliza
    - 操作
    - 通知
  - MemoryManager
  - MemoryPool
    - Code Cache
    - Eden Space
    - Perm Gen
    - Survivor Space
    - Tenured Gen
  - OperatingSystem
  - Runtime
  - Threading
    - 属性
      - ThreadCount
      - CurrentThreadCpuTime
      - CurrentThreadCpuTimeS
      - ThreadContentionMonit
      - DaemonThreadCount
      - PeakThreadCount
      - ObjectMonitorUsageSup
      - SynchronizerUsageSupp

属性值

名称	值
CurrentThreadCpuTime	放弃图表



刷新

MBeanAttributeInfo

名称	值
属性:	
名称	CurrentThreadCpuTime
描述	CurrentThreadCpuTime
可读	true
可写	false

描述符

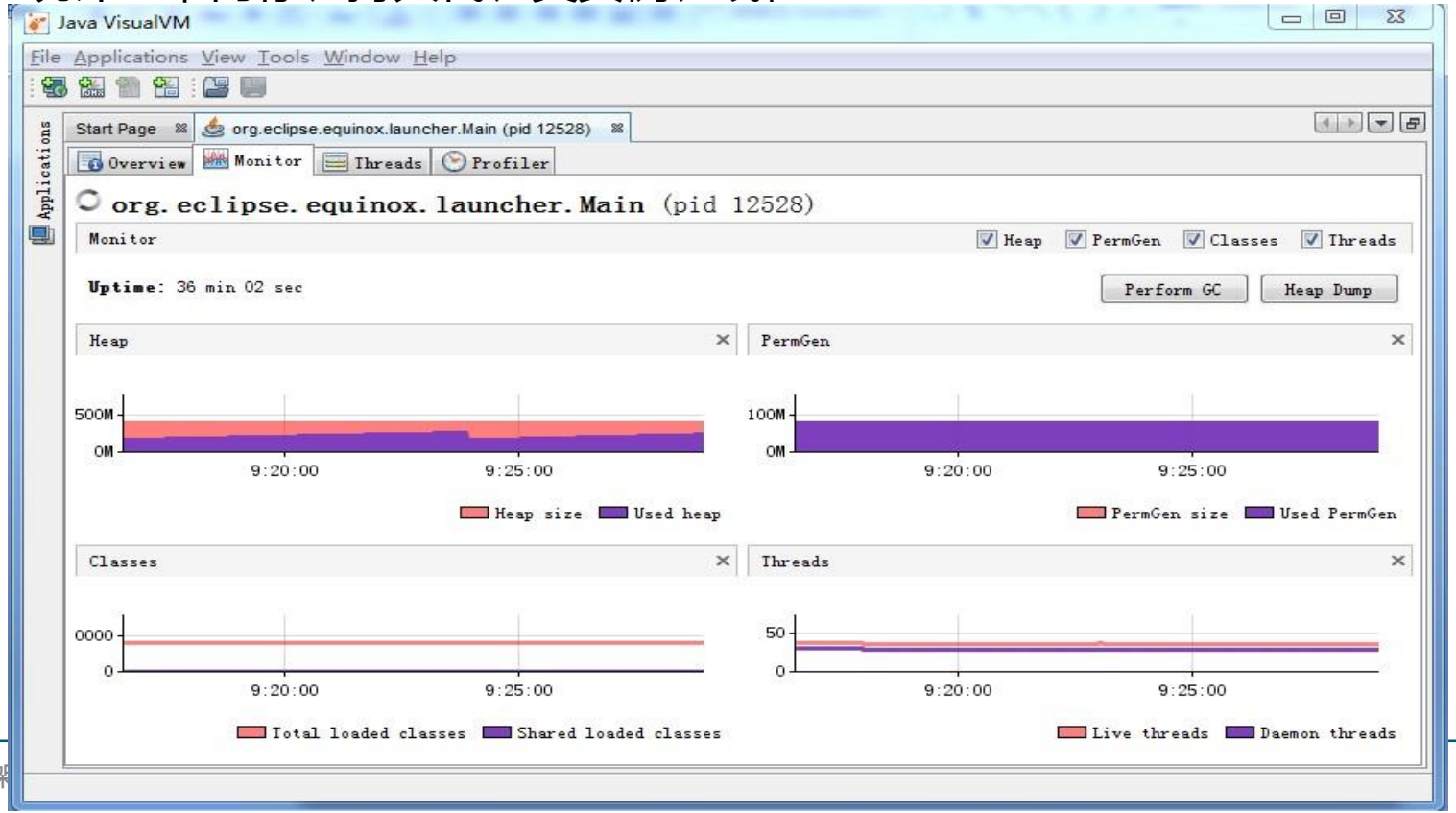
名称	值
属性:	
openType	javax.management.openmbean.SimpleType(name=java.lang.Long)
originalType	long

JVisualVM

- VisualVM：JDK6以后自带的高级监控分析工具
  - 内存使用监控（不分代）
  - 线程监控及产生dump
  - CPU和内存详细信息收集和分析
- 程序命令：%JAVA_HOME%/bin/jvisualvm.exe

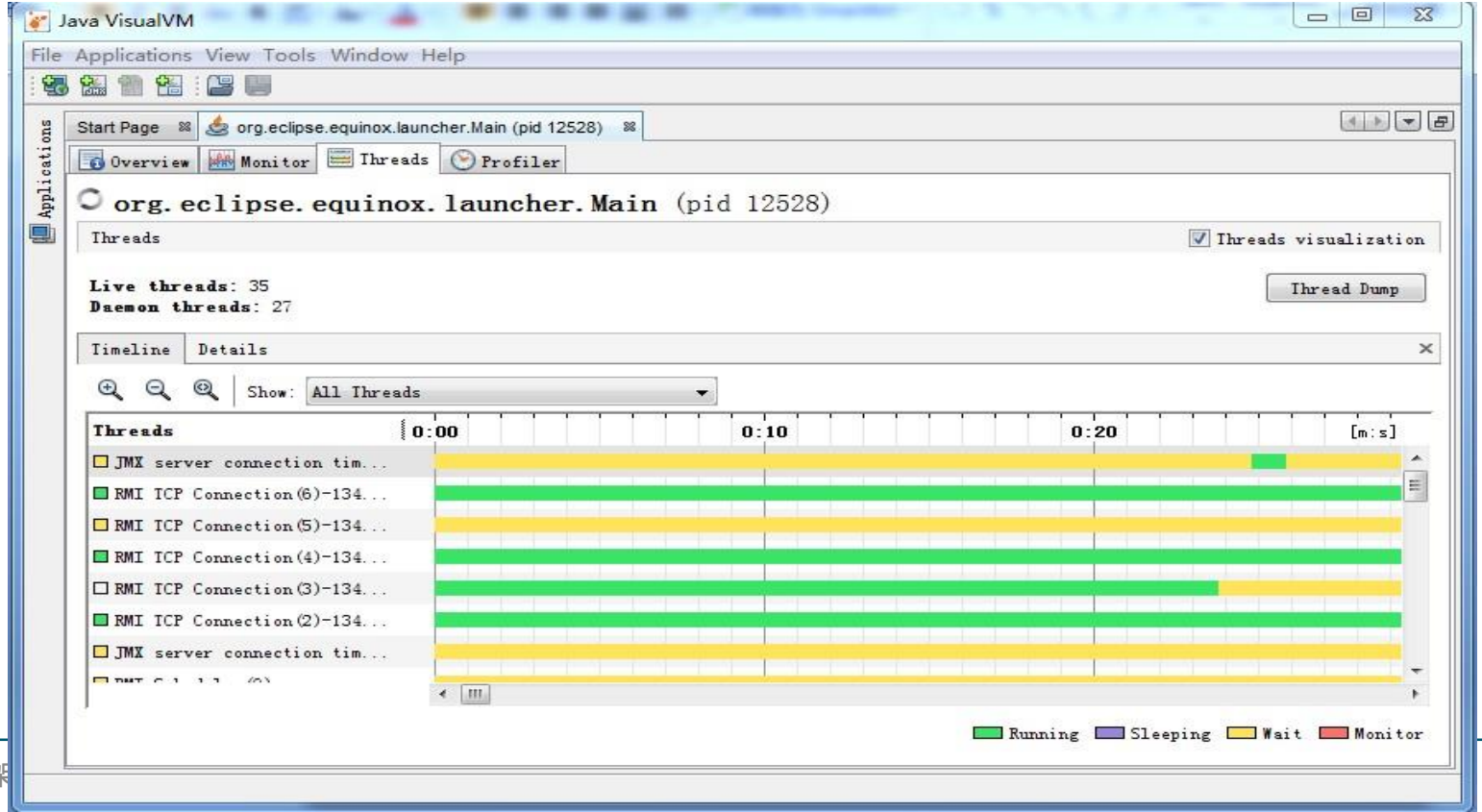
JVisualVM示例-1/5

- 统计：堆内存、持久代、类实例、线程



JVisualVM示例-2/5

- 线程监控



JVisualVM示例-3/5

- 一段时间内CPU和内存信息分析收集

Java VisualVM

File Applications View Tools Window Help

org.eclipse.equinox.launcher.Main (pid 17096) | Start Page

Overview | Monitor | Threads | Profiler

org.eclipse.equinox.launcher.Main (pid 17096)

Profiler | Settings

Profile: CPU | Memory | Stop

Status: application terminated

Profiling results

Hot Spots - Method	Self time [%]	Self time	Invocations
org.eclipse.equinox.internal.util.impl.tpt.thre		29310 ms (81.4%)	1
org.slf4j.impl.StaticLoggerBinder.**init** ()		2423 ms (6.7%)	1
org.eclipse.m2e.logback.configuration.LogPlugin.		1285 ms (3.6%)	1
org.eclipse.swt.widgets.Display.**sleep** ()		1263 ms (3.5%)	18
org.eclipse.swt.widgets.RunnableLock.**run** ()		525 ms (1.5%)	16
org.slf4j.impl.StaticLoggerBinder.**<init>** ()		321 ms (0.9%)	1
org.eclipse.osgi.baseadaptor.bundlefile.ZipBundl		50.9 ms (0.1%)	1793
org.eclipse.ui.internal.Workbench.**runUI** ()		45.7 ms (0.1%)	1
org.eclipse.ui.internal.Workbench.**createSplas**		41.3 ms (0.1%)	1
com.ibm.icu.impl.ICUResourceBundleReader.**readD**		39.5 ms (0.1%)	6
org.eclipse.ui.internal.Workbench.**<init>** (...		37.4 ms (0.1%)	1
org.eclipse.osgi.internal.loader.BundleLoader.l		31.5 ms (0.1%)	52

[Method Name Filter]

JVisualVM示例-4/5

- 线程CPU时间

Java VisualVM

File Applications View Tools Window Help

Applications

org.eclipse.equinox.launcher.Main (pid 17096) | Start Page | CPU: 09:33:05 上午

View: Methods

Call Tree - Method	Time [%]	Time	Invocat...
All threads		0.913 ms (100%)	1
main		0.913 ms (100%)	1
org.eclipse.ui.internal.ide.IDEWorkbenchPlugin$2.**run** ()		0.913 ms (100%)	1
Self time		0.661 ms (72.4%)	1
org.eclipse.swt.widgets.Display.**timerExec** (int, Runnable)		0.142 ms (15.6%)	1
Self time		0.078 ms (8.5%)	1
org.eclipse.swt.widgets.Display.**checkDevice** ()		0.063 ms (6.9%)	1
Self time		0.047 ms (5.1%)	1
org.eclipse.swt.graphics.Device.**isDisposed** ()		0.016 ms (1.8%)	1
org.eclipse.swt.widgets.Display.**getCurrent** ()		0.109 ms (11.9%)	1
Self time		0.075 ms (8.2%)	1
org.eclipse.swt.widgets.Display.**findDisplay** (Thread)		0.033 ms (3.6%)	1

Call Tree | Hot Spots | Combined | Info

JVisualVM示例-5/5

- 内存对象分配

Java VisualVM

File Applications View Tools Window Help

org.eclipse.equinox.launcher.Main (pid 17336) | Start Page

Overview | Monitor | Threads | Profiler

org.eclipse.equinox.launcher.Main (pid 17336)

Profiler | Settings

Profile: CPU | Memory | Stop

Status: application terminated

Profiling results

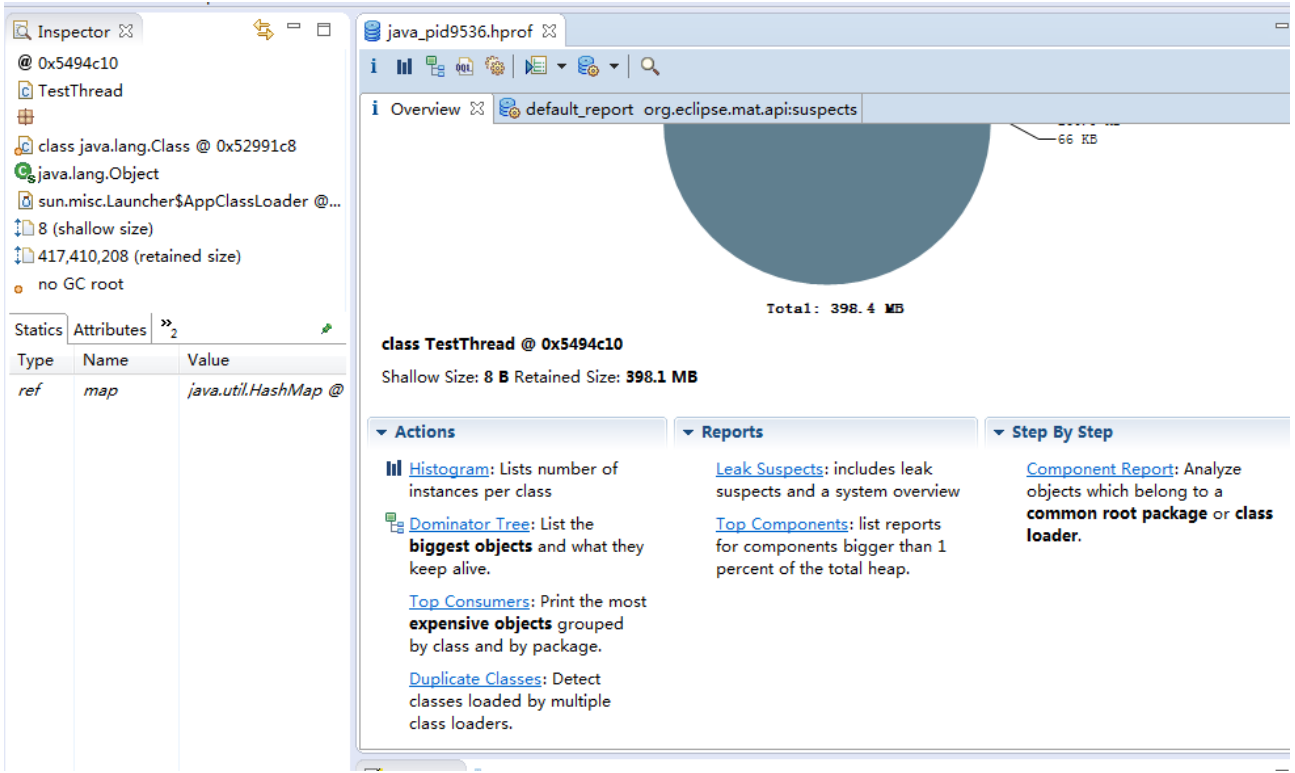
Class Name - Live Allocated Objects	Live Bytes	Live Bytes	Live Objects	Generations
char[]		825,0... (30.1%)	11,855 (21.9%)	4
byte[]		583,9... (21.3%)	732 (1.4%)	4
java.lang.String		274,4... (10%)	11,435 (21.2%)	4
java.lang.Object[]		267,2... (9.8%)	5,378 (10%)	4
org.eclipse.core.internal.resources.ResourceInfo		182,0... (6.6%)	2,844 (5.3%)	1
java.lang.Integer		77,60... (2.8%)	4,850 (9%)	4
java.util.HashMap$Entry		66,48... (2.4%)	2,770 (5.1%)	4
org.eclipse.core.internal.dtree.DataTreeNode		64,89... (2.4%)	2,704 (5%)	2
org.eclipse.core.internal.resources.MarkerInfo		42,43... (1.5%)	1,326 (2.5%)	1
java.lang.Object[][]		40,75... (1.5%)	1 (0%)	1

[Class Name Filter]

Live Results | Class History

MAT

将dump出来的heap文件，使用mat工具打开



MAT-支配树，浅堆和深堆

Overview | default_report org.eclipse.mat.api:suspects | dominator_tree

Class Name	Shallow Heap	Retained Heap
<Regex>	<Numeric>	<Numeric>
class TestThread @ 0x5494c10	8	417,410,208
java.util.HashMap @ 0xae664d8	40	417,410,200
java.util.HashMap$Entry[32768] @ 0x1ee95f88	131,088	417,410,160
java.util.HashMap$Entry @ 0x20d5ce48	24	172,336
java.util.HashMap$Entry @ 0x104b29d8	24	151,888
java.util.HashMap$Entry @ 0x20d5c728	24	144,320
java.util.HashMap$Entry @ 0x165e50f0	24	138,688
java.util.HashMap$Entry @ 0x21341290	24	137,344
java.util.HashMap$Entry @ 0x15338100	24	137,312
java.util.HashMap$Entry @ 0x1f9d4138	24	135,008
java.util.HashMap$Entry @ 0x19100fb0	24	134,840
java.util.HashMap$Entry @ 0x1595b7c8	24	133,664
java.util.HashMap$Entry @ 0xa90c630	24	132,672
java.util.HashMap$Entry @ 0x10871d38	24	131,576
java.util.HashMap$Entry @ 0x1a85ac28	24	130,800

MAT-线程查看

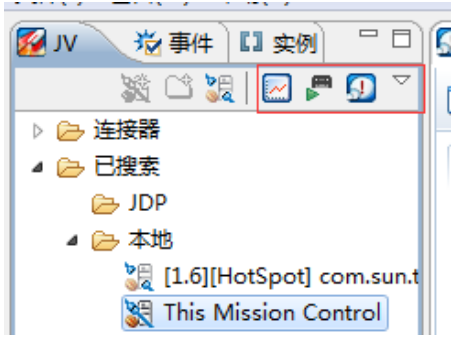
java_pid9536.hprof

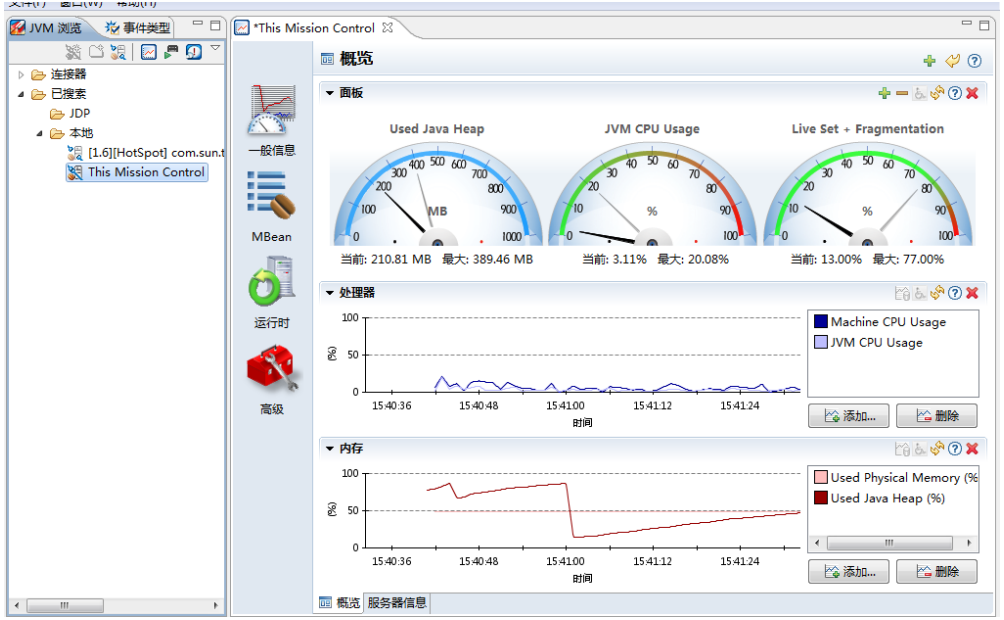
Overview | default_report org.eclipse.mat.api:suspects | dominator_tree | thread_overview

Object / Stack Frame	Name	Shallow Heap	Retained Heap	Context C
<Regex>	<Regex>	<Numeric>	<Numeric>	<Regex>
java.lang.Thread @ 0x217488b0	Session.24	112	18,712	sun.misc.
java.lang.Thread @ 0xae668f0	main	112	496	sun.misc.
java.lang.ref.Finalizer$FinalizerThread @ 0xae66748	Finalizer	112	312	
java.lang.Thread @ 0x1e492e38	Session.23	112	256	sun.misc.
java.lang.Thread @ 0x1e492d00	Session.22	112	256	sun.misc.
java.lang.ref.Reference$ReferenceHandler @ 0xae667d8	Reference Handler	112	192	
java.lang.Thread @ 0xae666d8	Signal Dispatcher	112	192	sun.misc.
java.lang.Thread @ 0xae66668	Attach Listener	112	192	sun.misc.
Σ Total: 8 entries		**896**	**20,608**	

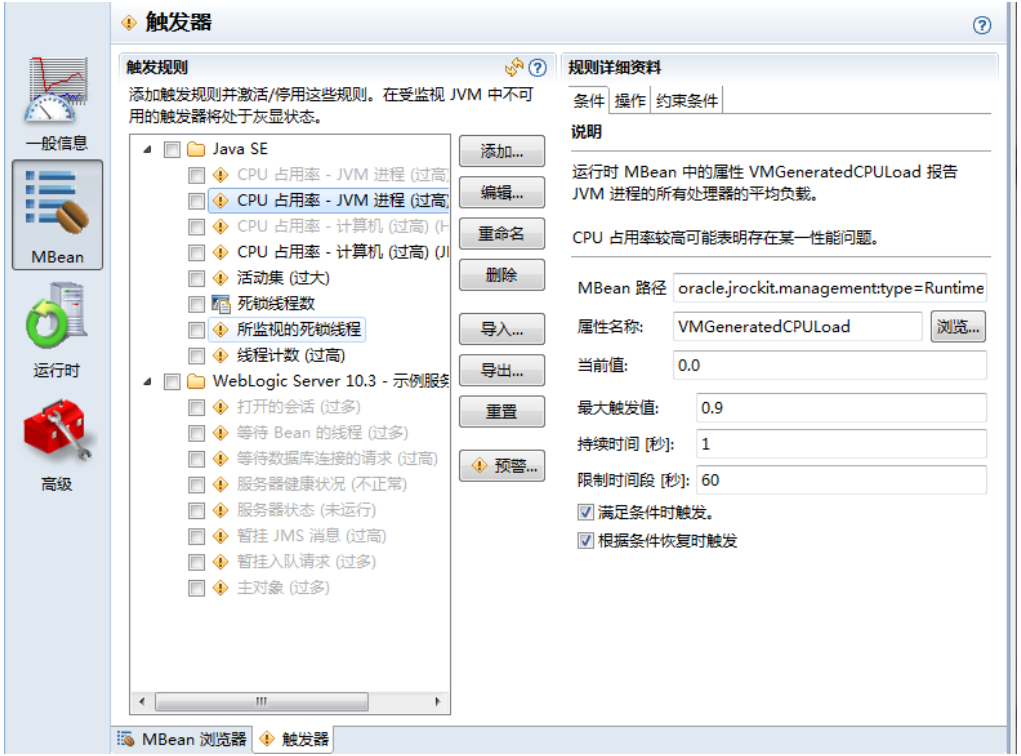
JMC-控制台(1)

-Xmanagement:ssl=false,authenticate=false,autodiscovery=true interface=xx.xx.xx.xx -Djava.rmi.server.hostname=xx.xx.xx.xx

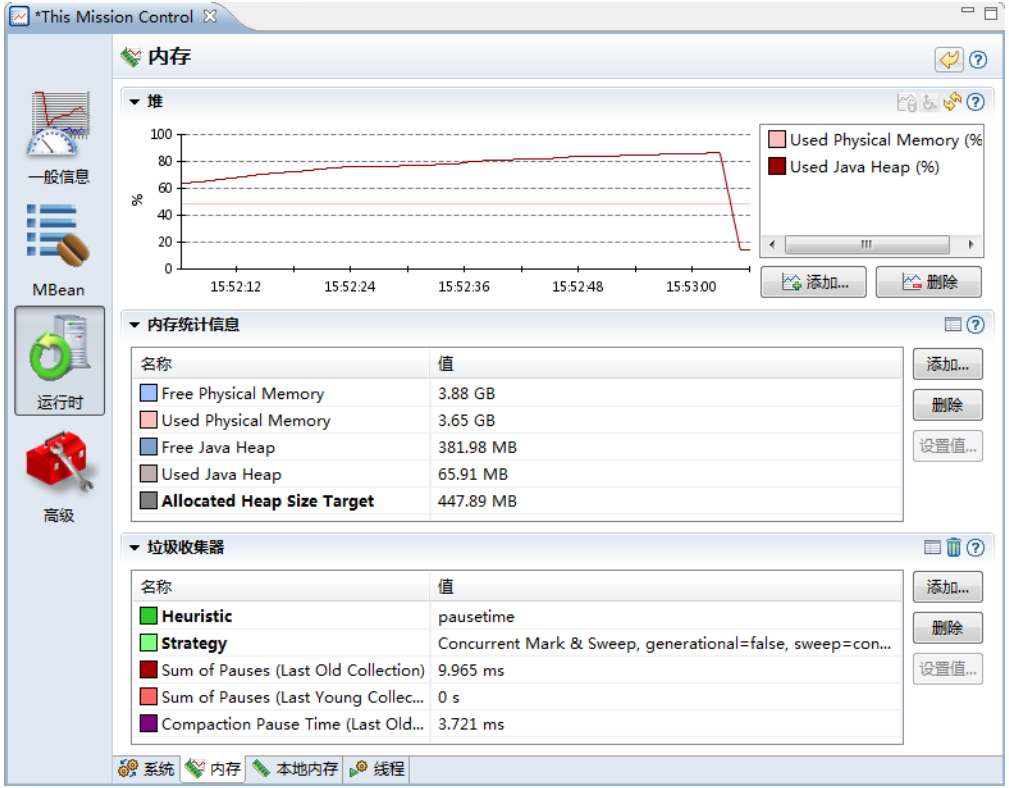




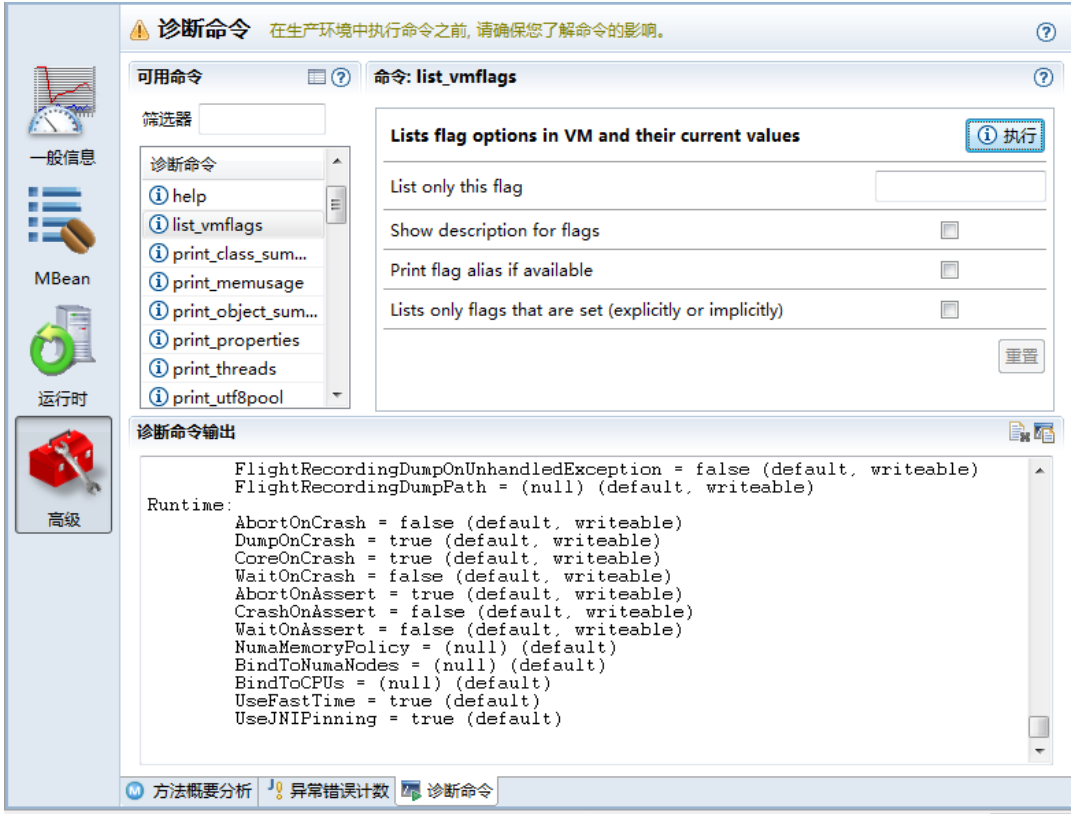
JMC-控制台(2)


触发器
一般信息
MBean
运行时
高级
触发规则
添加触发规则并激活/停用这些规则。在受监视 JVM 中不可用的触发器将处于灰显状态。
Java SE
CPU 占用率 - JVM 进程 (过高)
CPU 占用率 - JVM 进程 (过高)
CPU 占用率 - 计算机 (过高)
CPU 占用率 - 计算机 (过高)
活动集 (过大)
死锁线程数
所监视的死锁线程
线程计数 (过高)
WebLogic Server 10.3 - 示例服务
打开的会话 (过多)
等待 Bean 的线程 (过多)
等待数据库连接的请求 (过高)
服务器健康状况 (不正常)
服务器状态 (未运行)
暂挂 JMS 消息 (过高)
暂挂入队请求 (过多)
主对象 (过多)
添加...
编辑...
重命名
删除
导入...
导出...
重置
预警...
规则详细资料
条件
操作
约束条件
说明
运行时 MBean 中的属性 VMGeneratedCPULoad 报告 JVM 进程的所有处理器的平均负载。
CPU 占用率较高可能表明存在某一性能问题。
MBean 路径
oracle.jrockit.management:type=Runtime
属性名称:
VMGeneratedCPULoad
浏览...
当前值:
0.0
最大触发值:
0.9
持续时间 [秒]:
1
限制时间段 [秒]:
60
满足条件时触发。
根据条件恢复时触发
MBean 浏览器
触发器

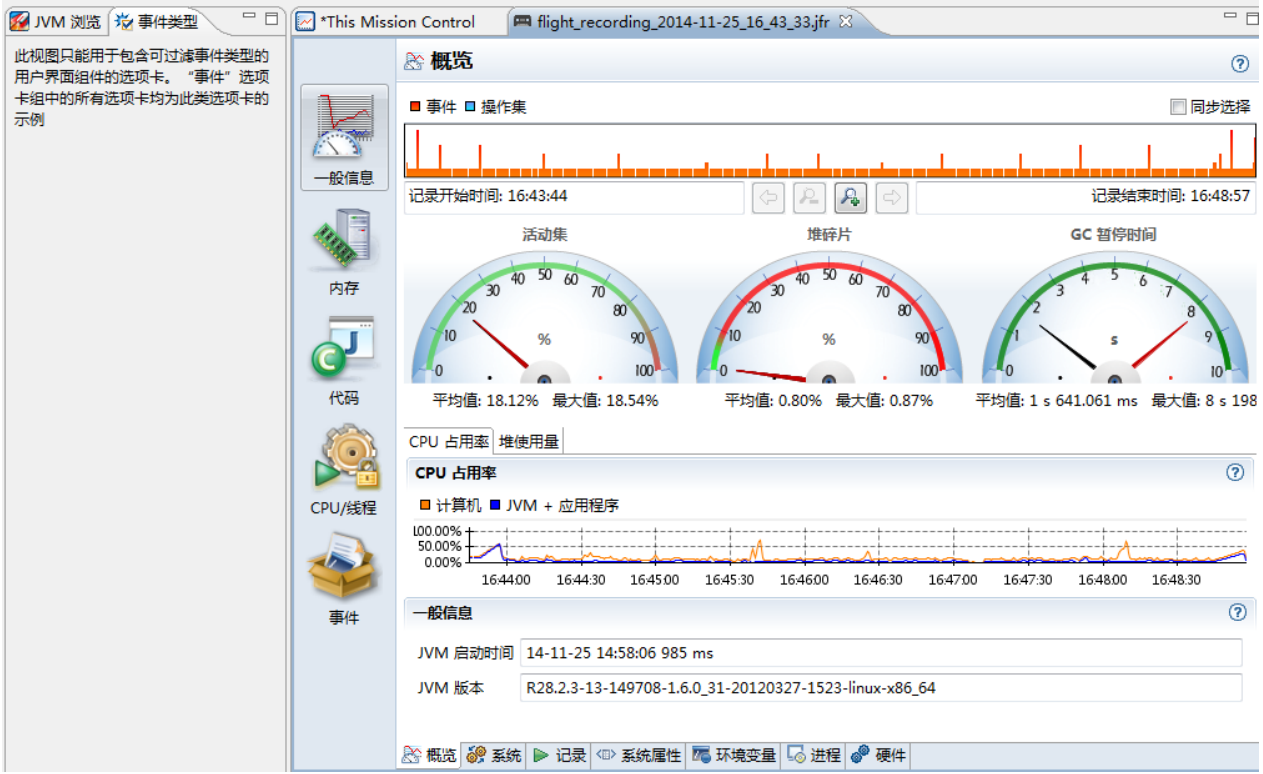
JMC-控制台(3)


*This Mission Control
内存
一般信息
MBean
运行时
高级
堆
100
80
60
40
20
0
%
15:52:12
15:52:24
15:52:36
15:52:48
15:53:00
Used Physical Memory (%
Used Java Heap (%)
添加...
删除
内存统计信息
名称
值
Free Physical Memory
3.88 GB
Used Physical Memory
3.65 GB
Free Java Heap
381.98 MB
Used Java Heap
65.91 MB
Allocated Heap Size Target
447.89 MB
添加...
删除
设置值...
垃圾收集器
名称
值
Heuristic
pausetime
Strategy
Concurrent Mark & Sweep, generational=false, sweep=con...
Sum of Pauses (Last Old Collection)
9.965 ms
Sum of Pauses (Last Young Collec...
0 s
Compaction Pause Time (Last Old...
3.721 ms
添加...
删除
设置值...
系统
内存
本地内存
线程

JMC-控制台(4)


诊断命令
在生产环境中执行命令之前，请确保您了解命令的影响。
一般信息
MBean
运行时
高级
可用命令
筛选器
诊断命令
help
list_vmflags
print_class_sum...
print_memusage
print_object_sum...
print_properties
print_threads
print_utf8pool
命令: list_vmflags
Lists flag options in VM and their current values
执行
List only this flag
Show description for flags
Print flag alias if available
Lists only flags that are set (explicitly or implicitly)
重置
诊断命令输出
FlightRecordingDumpOnUnhandledException = false (default, writeable)
FlightRecordingDumpPath = (null) (default, writeable)
Runtime:
AbortOnCrash = false (default, writeable)
DumpOnCrash = true (default, writeable)
CoreOnCrash = true (default, writeable)
WaitOnCrash = false (default, writeable)
AbortOnAssert = true (default, writeable)
CrashOnAssert = false (default, writeable)
WaitOnAssert = false (default, writeable)
NumaMemoryPolicy = (null) (default)
BindToNumaNodes = (null) (default)
BindToCPUs = (null) (default)
UseFastTime = true (default)
UseJNIPinning = true (default)
方法概要分析
异常错误计数
诊断命令

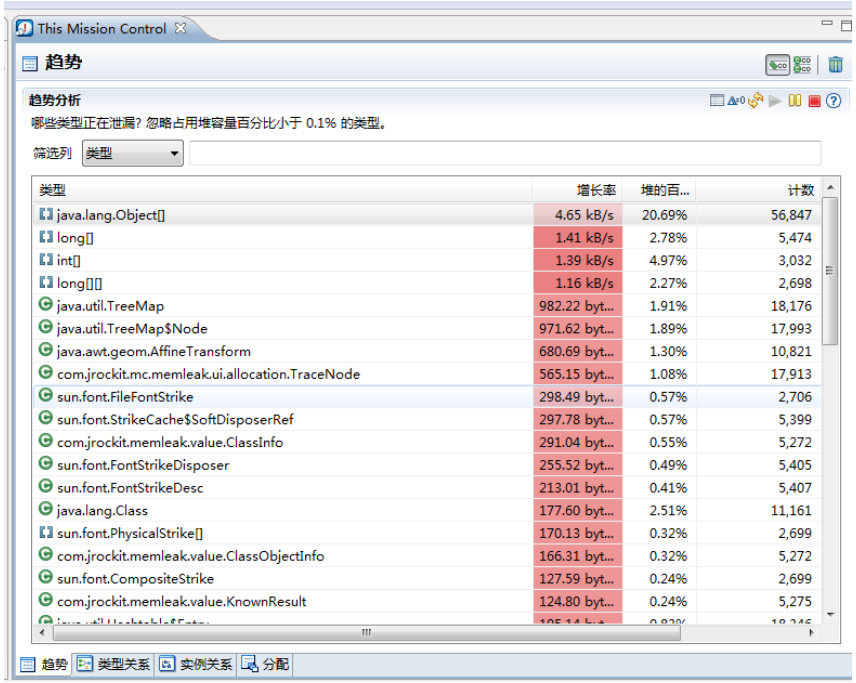
JMC-飞行记录器


JVM 浏览
事件类型
此视图只能用于包含可过滤事件类型的用户界面组件的选项卡。“事件”选项卡组中的所有选项卡均为此类选项卡的示例
*This Mission Control
flight_recording_2014-11-25_16_43_33.jfr
概览
一般信息
内存
代码
CPU/线程
事件
事件
操作集
同步选择
记录开始时间: 16:43:44
记录结束时间: 16:48:57
活动集
堆碎片
GC 暂停时间
平均值: 18.12% 最大值: 18.54%
平均值: 0.80% 最大值: 0.87%
平均值: 1 s 641.061 ms 最大值: 8 s 198
CPU 占用率
堆使用量
CPU 占用率
计算机
JVM + 应用程序
100.00%
50.00%
0.00%
16:44:00
16:44:30
16:45:00
16:45:30
16:46:00
16:46:30
16:47:00
16:47:30
16:48:00
16:48:30
一般信息
JVM 启动时间
14-11-25 14:58:06 985 ms
JVM 版本
R28.2.3-13-149708-1.6.0_31-20120327-1523-linux-x86_64
概览
系统
记录
系统属性
环境变量
进程
硬件

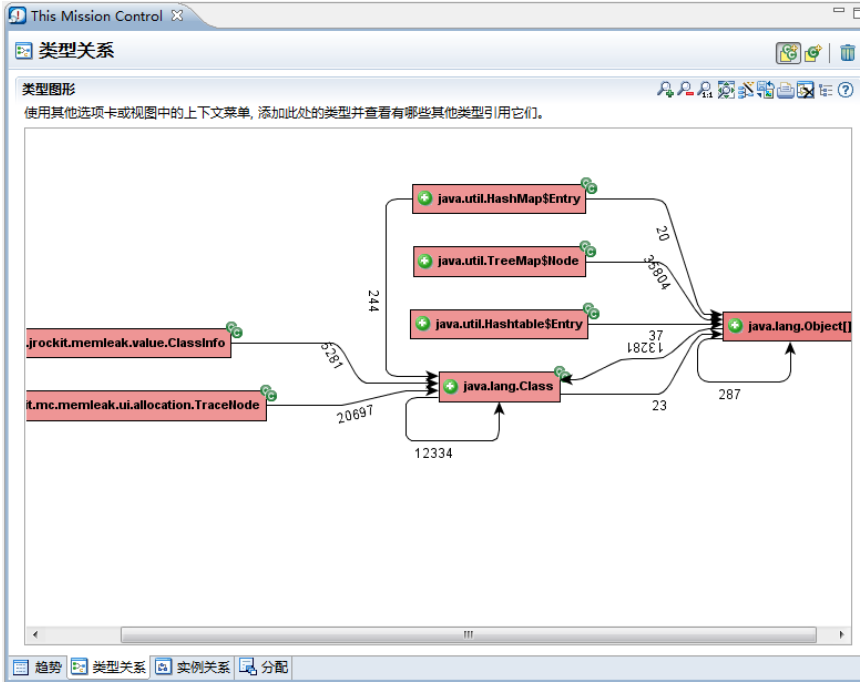
JMC-飞行记录器


*This Mission Control
flight_recording_2014-11-25_16_43_33.jfr
概览
事件 操作集
同步选择
记录开始时间: 16:43:44
记录结束时间: 16:48:57
一般信息
内存
代码
CPU/线程
事件
内存使用量
计算机物理内存总量 已用计算机物理内存 进程提交的 Java 堆 堆使用量 活动集
128.00 GB
112.00 GB
96.00 GB
80.00 GB
64.00 GB
48.00 GB
32.00 GB
16.00 GB
0 bytes
16:44:00
16:44:30
16:45:00
16:45:30
16:46:00
16:46:30
16:47:00
16:47:30
16:48:00
16:48:30
GC 配置
初始堆大小 24.41 GB
最大堆大小 27.34 GB
预定的暂停时间 0
堆地址 (位) 36
硬件地址 (位) 64
GC 统计信息
垃圾收集 16
最短主暂停时间 114.057 ms
平均主暂停时间 1 s 641.061 ms
最长主暂停时间 8 s 198.058 ms
分配
在 TLA 中分配 36.76 GB
分配总计 37.94 GB
TLA 中的分配比率 120.21 MB/s
总分配比率 124.07 MB/s
概览
GC 一般信息
GC 图形
分配
堆内容
对象统计信息

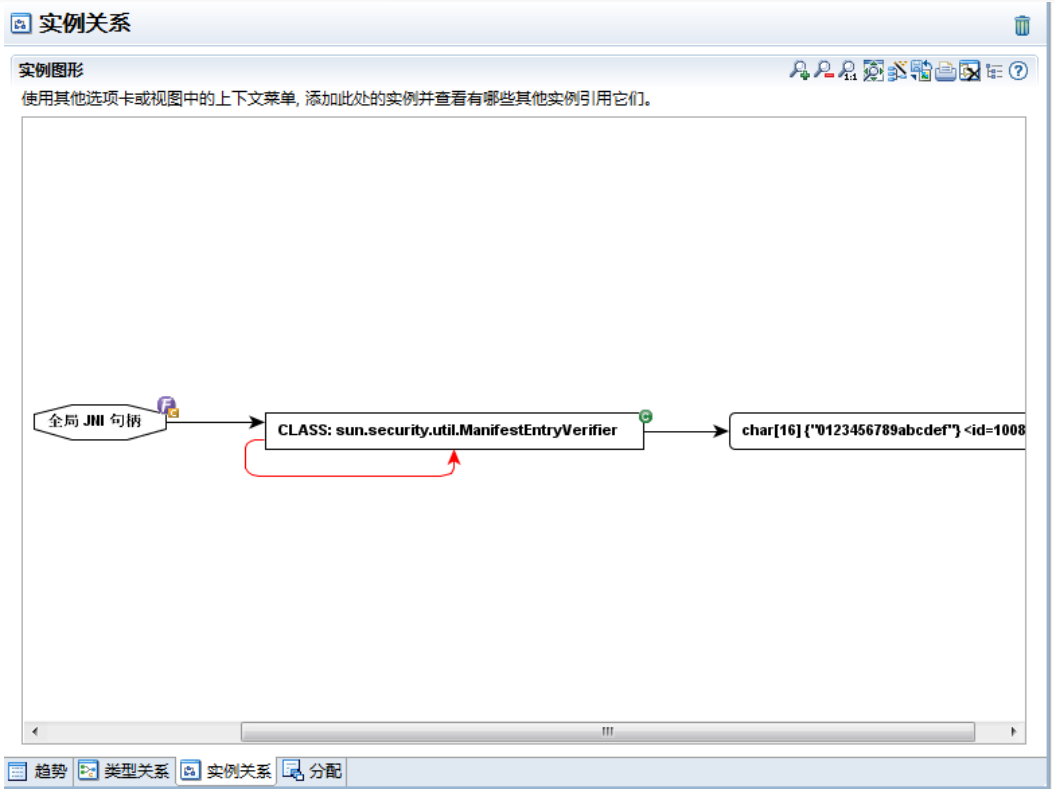
JMC-MemLeak(1)



类型	增长率	堆的百...	计数
java.lang.Object[]	4.65 kB/s	20.69%	56,847
long[]	1.41 kB/s	2.78%	5,474
int[]	1.39 kB/s	4.97%	3,032
long[][]	1.16 kB/s	2.27%	2,698
java.util.TreeMap	982.22 byt...	1.91%	18,176
java.util.TreeMap$Node	971.62 byt...	1.89%	17,993
java.awt.geom.AffineTransform	680.69 byt...	1.30%	10,821
com.jrockit.mc.memleak.ui.allocation.TraceNode	565.15 byt...	1.08%	17,913
sun.font.FileFontStrike	298.49 byt...	0.57%	2,706
sun.font.StrikeCache$SoftDisposerRef	297.78 byt...	0.57%	5,399
com.jrockit.memleak.value.ClassInfo	291.04 byt...	0.55%	5,272
sun.font.FontStrikeDisposer	255.52 byt...	0.49%	5,405
sun.font.FontStrikeDesc	213.01 byt...	0.41%	5,407
java.lang.Class	177.60 byt...	2.51%	11,161
sun.font.PhysicalStrike[]	170.13 byt...	0.32%	2,699
com.jrockit.memleak.value.ClassObjectInfo	166.31 byt...	0.32%	5,272
sun.font.CompositeStrike	127.59 byt...	0.24%	2,699
com.jrockit.memleak.value.KnownResult	124.80 byt...	0.24%	5,275



JMC-MemLeak(2)



实战案例(1)

```
cd E:\Workspace\netbeans\JavaTest\build\classes
E:
set JAVA_HOME=D:\Java\jdk1.6.0_27
%JAVA_HOME%\bin\java -server -Xmx512m > screen.log 2>&1
```


Heap_OOM.java


TestThread.java

实战案例(2)

```
Exception in thread "Session.26" java.lang.OutOfMemoryError: Java heap space
        at TestThread.useHeapMem(TestThread.java:29)
        at TestThread.run(TestThread.java:14)
        at java.lang.Thread.run(Thread.java:662)
Session.27
Exception in thread "Session.27" java.lang.OutOfMemoryError: Java heap space
        at TestThread.useHeapMem(TestThread.java:29)
        at TestThread.run(TestThread.java:14)
        at java.lang.Thread.run(Thread.java:662)
Session.28
Exception in thread "Session.28" java.lang.OutOfMemoryError: Java heap space
        at TestThread.useHeapMem(TestThread.java:29)
        at TestThread.run(TestThread.java:14)
        at java.lang.Thread.run(Thread.java:662)
Session.29
Exception in thread "Session.29" java.lang.OutOfMemoryError: Java heap space
        at TestThread.useHeapMem(TestThread.java:29)
        at TestThread.run(TestThread.java:14)
        at java.lang.Thread.run(Thread.java:662)
```

screen.log

实战案例(3)

```
cd E:\Workspace\netbeans\JavaTest\build\classes
E:
set JAVA_HOME=D:\Java\jdk1.6.0_27
%JAVA_HOME%\bin\java -server -Xmx512m -XX:+HeapDumpOnOutOfMemoryError
-verbose:gc -Xloggc:gc.log -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -
XX:+PrintHeapAtGC Heap_OOM > screen.log 2>&1
```

实战案例(4)

```
java.lang.OutOfMemoryError: Java heap spaceDumping heap to java_pid9536.hprof ...Heap dump file created [418588668 bytes in 5.526
secs]Exception in thread "Session.24" java.lang.OutOfMemoryError: Java heap space
                    at TestThread.useHeapMem(TestThread.java:29)
                    at TestThread.run(TestThread.java:14)
                    at java.lang.Thread.run(Thread.java:662)
Session.25
Exception in thread "Session.25" java.lang.OutOfMemoryError: Java heap space
                    at TestThread.useHeapMem(TestThread.java:29)
                    at TestThread.run(TestThread.java:14)
                    at java.lang.Thread.run(Thread.java:662)
Session.26
Exception in thread "Session.26" java.lang.OutOfMemoryError: Java heap space
                    at TestThread.useHeapMem(TestThread.java:29)
                    at TestThread.run(TestThread.java:14)
                    at java.lang.Thread.run(Thread.java:662)
Session.27
Exception in thread "Session.27" java.lang.OutOfMemoryError: Java heap space
                    at TestThread.useHeapMem(TestThread.java:29)
                    at TestThread.run(TestThread.java:14)
                    at java.lang.Thread.run(Thread.java:662)
```

加了-XX:+HeapDumpOnOutOfMemoryError参数后，产生java_pid9536.hprof文件，这个文件是java虚拟机heap的dump文件，可以使用MAT工具来分析。

screen.log

实战案例(5)

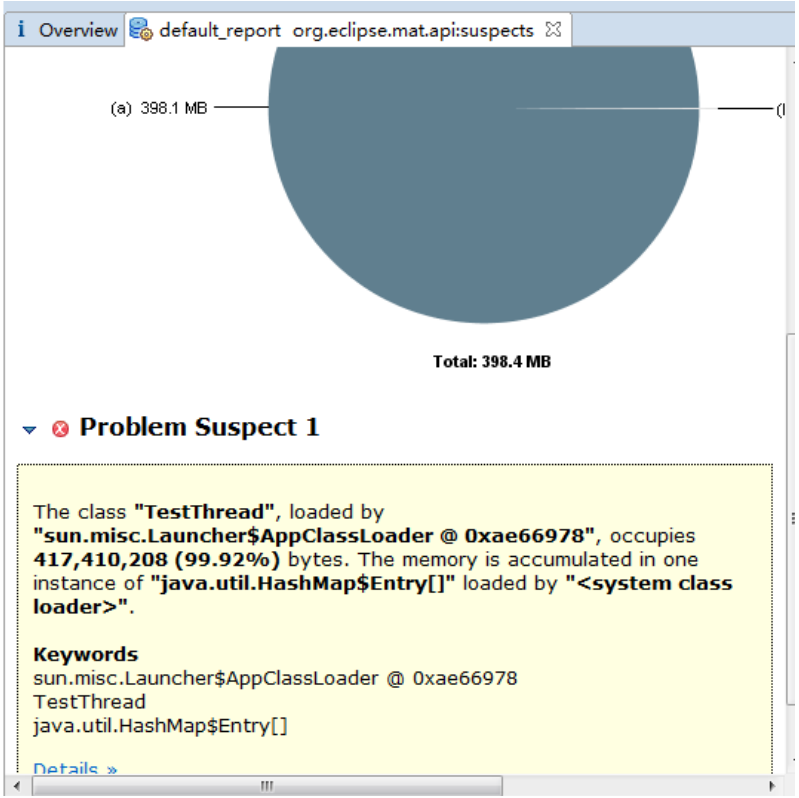
```
{Heap before GC invocations=1 (full 0):
 PSYoungGen      total 18560K, used 15936K [0x1e7e0000, 0x1fc90000, 0x29280000)
  eden space 15936K, 100% used [0x1e7e0000,0x1f770000,0x1f770000)
  from space 2624K, 0% used [0x1fa00000,0x1fa00000,0x1fc90000)
  to   space 2624K, 0% used [0x1f770000,0x1f770000,0x1fa00000)
 PSOldGen        total 42496K, used 0K [0x09280000, 0x0bc00000, 0x1e7e0000)
  object space 42496K, 0% used [0x09280000,0x09280000,0x0bc00000)
 PSPermGen       total 16384K, used 2139K [0x05280000, 0x06280000, 0x09280000)
  object space 16384K, 13% used [0x05280000,0x05496c50,0x06280000)
0.072: [GC [PSYoungGen: 15936K->2621K(18560K)] 15936K->15803K(61056K), 0.0050695 secs] [Times: user=0.00 sys=0.00, real=0.01 secs]
Heap after GC invocations=1 (full 0):
 PSYoungGen      total 18560K, used 2621K [0x1e7e0000, 0x20c20000, 0x29280000)
  eden space 15936K, 0% used [0x1e7e0000,0x1e7e0000,0x1f770000)
```

```
Find result - 40 hits
Search "Full GC" (40 hits in 1 file)
  E:\Workspace\netbeans\JavaTest\build\classes\gc.log (40 hits)
    Line 50: 0.093: [Full GC [PSYoungGen: 2620K->0K(34496K)] [PSOldGen: 28623K->31209K(66112K)] 31243K->31
    Line 90: 1.120: [Full GC [PSYoungGen: 2611K->0K(34496K)] [PSOldGen: 59819K->62373K(113728K)] 62430K-
    Line 130: 2.099: [Full GC [PSYoungGen: 2620K->0K(58752K)] [PSOldGen: 91143K->93690K(152256K)] 937
    Line 170: 6.110: [Full GC [PSYoungGen: 2617K->0K(58752K)] [PSOldGen: 146022K->148334K(227520K)
    Line 230: 13.123: [Full GC [PSYoungGen: 58237K->33198K(116480K)] [PSOldGen: 203432K->227504K(316800K)] 261670K->26
    Line 290: 18.142: [Full GC [PSYoungGen: 58239K->26395K(116480K)] [PSOldGen: 285465K->316781K(349568K)] 343704K->34
    Line 310: 23.093: [Full GC [PSYoungGen: 58233K->24420K(116480K)] [PSOldGen: 316781K->349565K(349568K)] 375015K->37
    Line 330: 24.092: [Full GC [PSYoungGen: 58232K->57624K(116480K)] [PSOldGen: 349565K->349552K(349568K)] 407797K->40
    Line 350: 24.158: [Full GC [PSYoungGen: 58234K->58234K(116480K)] [PSOldGen: 349552K->349552K(349568K)] 407786K->40
    Line 370: 24.173: [Full GC [PSYoungGen: 58234K->58202K(116480K)] [PSOldGen: 349552K->349562K(349568K)] 407786K->40
    Line 390: 24.249: [Full GC [PSYoungGen: 58240K->58240K(116480K)] [PSOldGen: 349562K->349562K(349568K)] 407802K->40
    Line 410: 24.263: [Full GC [PSYoungGen: 58240K->58232K(116480K)] [PSOldGen: 349562K->349562K(349568K)] 407802K->40
```

gc.log

可以看出Full GC次数很多，达到了40多次，然而内存却没有有效释放。

实战案例(6)-使用MAT进行分析



Shortest Paths To the Accumulation Point

Class Name	Shallow Heap	Retained Heap
java.util.HashMap$Entry[32768] @ 0x1ee95f88	131,088	417,410,160
table java.util.HashMap @ 0xae664d8	40	417,410,200
map class TestThread @ 0x5494c10	8	417,410,208
<class> TestThread @ 0x21748920	8	8
<Java Local>, target java.lang.Thread @ 0x217488b0 Session.24 Thread	112	18,712
<class> TestThread @ 0x1e492ea8 »	8	8
<class> TestThread @ 0x1e492d70 »	8	8
[1] java.lang.Object[10] @ 0xae68030 »	56	56
Total: 4 entries		

实战案例(7)-使用MAT进行分析

ATAGURU 炼数成金

Accumulated Objects in Dominator Tree

Class Name	Shallow Heap	Retained Heap	Percentage
class TestThread @ 0x5494c10	8	417,410,208	99.92%
java.util.HashMap @ 0xae664d8	40	417,410,200	99.92%
java.util.HashMap$Entry[32768] @ 0x1ee95f88	131,088	417,410,160	99.92%
java.util.HashMap$Entry @ 0x20d5ce48	24	172,336	0.04%
java.util.HashMap$Entry @ 0x104b29d8	24	151,888	0.04%
java.util.HashMap$Entry @ 0x20d5c728	24	144,320	0.03%
java.util.HashMap$Entry @ 0x165e50f0	24	138,688	0.03%
java.util.HashMap$Entry @ 0x21341290	24	137,344	0.03%
java.util.HashMap$Entry @ 0x15338100	24	137,312	0.03%

Overview | default_report org.eclipse.mat.api:suspects | dominator_tree | thread_overview

Object / Stack Frame	Name	Shallow Heap	Retained Heap	Context
<Regex>	<Regex>	<Numeric>	<Numeric>	<Regex>
java.lang.Thread @ 0x217488b0	Session.24	112	18,712	sun.misc.
at java.lang.OutOfMemoryError.<init>()V (OutOfMemory				
at TestThread.useHeapMem(I)V (TestThread.java:29)				
at TestThread.run()V (TestThread.java:14)				
at java.lang.Thread.run()V (Thread.java:662)				
Total: 4 entries				
java.lang.Thread @ 0xae668f0	main	112	496	sun.misc.
java.lang.ref.Finalizer$FinalizerThread @ 0xae66748	Finalizer	112	312	
java.lang.Thread @ 0x1e492e38	Session.23	112	256	sun.misc.
java.lang.Thread @ 0x1e492d00	Session.22	112	256	sun.misc.
java.lang.ref.Reference$ReferenceHandler @ 0xae667d8	Reference Handler	112	192	
java.lang.Thread @ 0xae666d8	Signal Dispatcher	112	192	sun.misc.
java.lang.Thread @ 0xae66668	Attach Listener	112	192	sun.misc.
Total: 8 entries		**896**	**20,608**	

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

Thanks

FAQ时间