Java常用命令工具

# 工具

## Gcview

通过GCView工具打开日志文件，查看整体的 GC 性能.

[gcviewer download | SourceForge.net](https://sourceforge.net/projects/gcviewer/)

# jdk命令

## java8 元空间

元空间与永久代之间最大的区别在于：元空间并不在虚拟机中，而是使用本地内存。因此，默认情况下，元空间的大小仅受本地内存限制，但可以通过以下参数来指定元空间的大小。

JDK 8 中永久代向元空间的转换理由：

1、字符串存在永久代中，容易出现性能问题和内存溢出。

2、类及方法的信息等比较难确定其大小，因此对于永久代的大小指定比较困难，太小容易出现永久代溢出，太大则容易导致老年代溢出。

3、永久代会为 GC 带来不必要的复杂度，并且回收效率偏低。

## 基础知识

吞吐量 = CPU在用户应用程序运行的时间 / （CPU在用户应用程序运行的时间 + CPU垃圾回收的时间）

并发:一个处理器同时处理多个任务。

并行:多个处理器或者是多核的处理器同时处理多个不同的任务.

## 1. javap-反编译工具

javap是jdk自带的反解析工具。它的作用就是根据class字节码文件，反解析出当前类对应的code区（汇编指令）、本地变量表、异常表和代码行偏移量映射表、常量池等等信息。

<https://www.cnblogs.com/frankcui/p/12134360.html>

常用命令选项

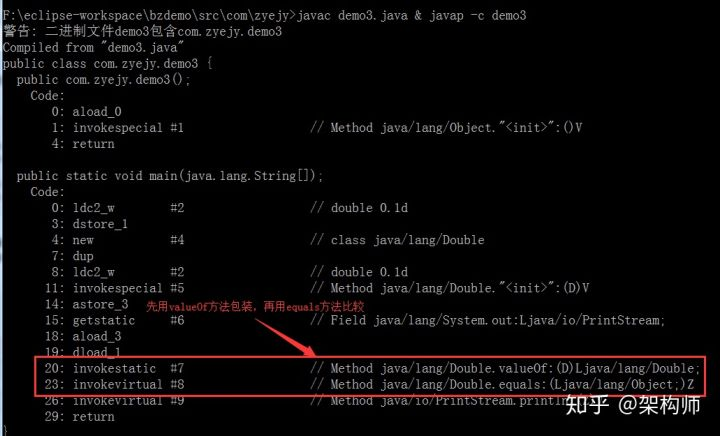
javap -v classxx，不仅会输出行号、本地变量表信息、反编译汇编代码，还会输出当前类用到的常量池等信息。

javap -l 会输出行号和本地变量表信息。

javap -c 会对当前class字节码进行反编译生成汇编代码。

案例

javac Test.java & javap -c Test

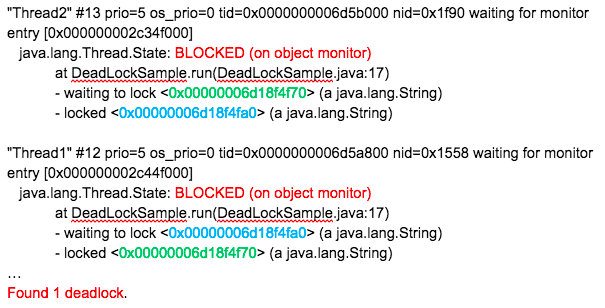


## 2. jstack-分析线程栈信息

通过jstack分析线程死锁问题。具体步骤如下：

1.使用 jps 或者系统的 ps 命令、任务管理器等工具，确定进程 ID。

2.调用 jstack 获取线程栈：${JAVA\_HOME}\bin\jstack your\_pid



## 3.jmap导出dump文件

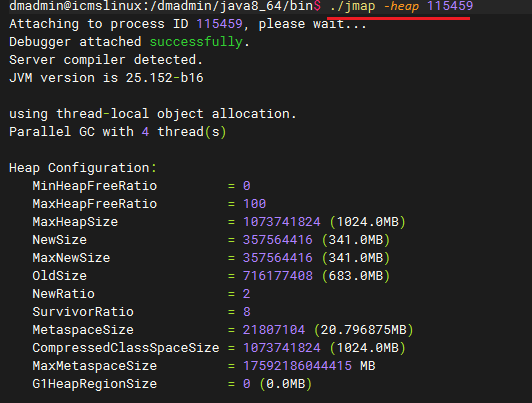
jmap -dump:file=javaDump.hprof,format=b 进程号



## 4.jmap查看各空间大小

查看jvm heap的使用情况

jmap -heap 115459



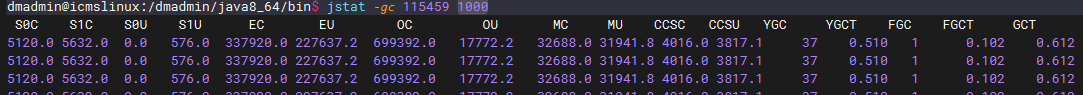
多版本java需要指定绝对路径



[sun.jvm.hotspot.runtime.VMVersionMismatchException](https://www.cnblogs.com/erlou96/p/14657174.html)

## 5.jstat查看各空间使用情况

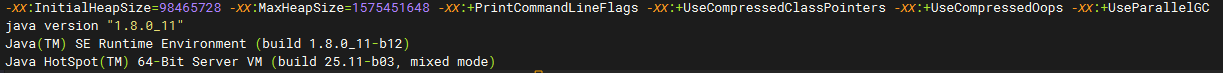
jstat -gc pid 时间



查看默认GC

java -XX:+PrintCommandLineFlags -version

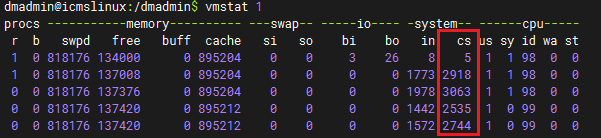




# 查看上下文切换

命令：vmstat 1

这里的 1 表示每秒刷新一次统计信息，查看cs这一列



## 查看指定进程的上下文切换情况

pidstat是sysstat工具的一个命令，用于监控全部或指定进程的cpu、内存、线程、设备IO等系统资源的占用情况



常用的参数：

-u：默认的参数，显示各个进程的cpu使用统计

-r：显示各个进程的内存使用统计

-d：显示各个进程的IO使用情况

-p：指定进程号

-w：显示每个进程的上下文切换情况

-t：显示选择任务的线程的统计信息外的额外信息

-h：在一行上显示了所有活动，这样其他程序可以容易解析。

-l：显示命令名和所有参数

**pidstat -w -l -p 7726 1 100**

