

JVM笔记jdk8+

[零、参考资料 2](#_Toc24230937)

[一、常用JVM参数 2](#_Toc24230938)

[1. GC参数 2](#_Toc24230939)

[2. 堆参数 3](#_Toc24230940)

[3. 非堆参数 4](#_Toc24230941)

[4. 其他参数 4](#_Toc24230942)

[二、分析JVM 5](#_Toc24230943)

[1. JDK自带 5](#_Toc24230944)

[1.1. 基本工具 5](#_Toc24230945)

[1.2. 可视化工具 7](#_Toc24230946)

[2. MAT 7](#_Toc24230947)

[3. LINUX/MAC OS 7](#_Toc24230948)

[3.1. top 7](#_Toc24230949)

[3.2. vmstat 7](#_Toc24230950)

[3.3. iostat 7](#_Toc24230951)

[3.4. pidstat 7](#_Toc24230952)

[4. Windows 7](#_Toc24230953)

[4.1. Perfmon 7](#_Toc24230954)

[4.2. Process Explorer 7](#_Toc24230955)

[4.3. pslist 7](#_Toc24230956)

[三、JVM常用执行指令 7](#_Toc24230957)

[1、汇编基础 7](#_Toc24230958)

[2、基础指令 7](#_Toc24230959)

[五、JVM基本原理 7](#_Toc24230960)

[1. 内存结构 7](#_Toc24230961)

[2. 类结构、加载、执行 7](#_Toc24230962)

[3. 垃圾收集器内存分配和算法实现 7](#_Toc24230963)

[4. 内存模型 7](#_Toc24230964)

[5. 垃圾收集器 7](#_Toc24230965)

# 零、参考资料

Java虚拟机规范(Java SE 8版)

Java语言规范基于Java SE 8

深入理解Java虚拟机——JVM高级特性与最佳实践

实战Java虚拟机——JVM故障诊断与性能优化

垃圾回收的算法与实现

深入理解计算机系统

Oracle官方DOC：

https://docs.oracle.com/javase/8/docs/technotes/tools/unix/toc.html

https://docs.oracle.com/en/java/javase/13/docs/specs/man/index.html

# 一、常用JVM参数

## GC参数

* 1. 打印GC基本日志

JDK8: -XX:PrintGC JDK9+: -Xlog:gc

* 1. 打印GC详细日志

JDK8: -XX:PrintGCDetails JDK9+: -Xlog:gc\*

-XX:PrintGCDetails 还会使JVM在退出前打印堆详细信息

* 1. 打印GC时堆的全面信息

JDK8: -XX:PrintHeapAtGC JDK9+:移除

* 1. 打印GC发生时间

JDK8: -XX:PrintGCTimeStamps JDK9+: -Xlog:gc\*已包含

* 1. 打印程序执行时间

-XX:+PrintGCApplicationConcurrentTime

* 1. 打印程序因STW而停顿的时间

-XX:+PrintGCApplicationStoppedTime

* 1. 打印软引用、弱引用、虚引用和Finallize队列信息

JDK8: +PrintReferenceGC JDK9+移除

* 1. 输出GC信息至文本

JDK8:-Xloggc:log/${filePath} JDK9+:-Xlog:gc:log/${filePath}

* 1. 打印类加载日志

JDK8: -XX:+TraceClassLoading JDK9+: -Xlog:class+load=info

* 1. 打印类卸载日志

JDK8: -XX:+TraceClassUnLoading JDK9+: -Xlog:class+unload=info

* 1. 显示当前的类信息柱状图

-XX:+PrintClassHistogram

* 1. 查看JVM运行显示参数

-XX:+PrintVMOptions

* 1. 打印JVM显示和隐式(默认)参数

-XX:+PrintCommandLineFlags

* 1. 打印所有系统参数

-XX:+PrintFlagsFinal

* 1. 在新生代和老年代使用串行回收器

-XX:UseSerialGC

* 1. 在新生代使用并行回收器

-XX:+UseParNewGC

* 1. 在老年代使用并行回收器

-XX:+UseParallelOldGC

* 1. 设置用于垃圾回收的线程数

-XX:ParallelGCThreads

* 1. 设置最大GC停顿时间

-XX:MaxGCPauseMillis

* 1. 设置吞吐量大小

-XX:GCTimeRatio

* 1. 打开开自适应GC策略

-XX:+UseAdaptiveSizePolicy

* 1. 使用G1

-XX:+UseG1GC

* 1. 设置停顿间隔

-XX:GCPauseIntervalMillis

* 1. 使用CMS

-XX:+UseConcMarkSweepGC

* 1. 设置CMS线程数

-XX:ParallelCMSThreads

* 1. 设置CMS触发时，老年代比例

-XX:CMSInitiatingOccupancyFraction

* 1. 设置CMS在FullGC后是否进行碎片整理

-XX:+UseCMSCompactAtFullCollection

* 1. 设置CMS进行内存压缩前，FullGC次数

-XX:CMSFullGCsBeforeCompaction

* 1. 允许对类元数据回收

-XX:+CMSClassUnloadingEnabled

* 1. 设置CMS触发时，永久区比例,仅在激活CMSClassUnloadingEnabled时有效

-XX:CMSInitiatingPermOccupancyFraction

* 1. 只有达到阈值才执行CMS GC

-XX:UseCMSInitiatingOccupancyOnly

## 堆参数

* 1. 初始堆空间

-Xms

* 1. 最大堆空间

-Xmx

* 1. 设置新生代大小

-Xmn

* 1. 设置新生代中eden区和from/to区的比例

-XX:SurvivorRatio

SurvivorRatio=eden/from=eden/to

* 1. 设置新生代与老年代的比例

-XX:NewRatio

NewRatio=老年代/新生代

* 1. 设置在内存溢出时到处整个堆信息

-XX:+HeapDumpOnOutOfMemoryError

* 1. 设置导出堆文件存放路径

-XX:HeapDumpPath

与-XX:+HeapDumpOnOutOfMemoryError配合

* 1. 设置大对象直接进入老年代的阈值

-XX:MaxTenuringThreshold

## 非堆参数

* 1. 最大直接内存

-XX:MaxDirectMemorySize

如果不设置，则默认与-Xmx相同

* 1. 每个线程的栈大小

-Xss

* 1. 打开TLAB

-XX:+UseTLAB

* 1. 打印TLAB相关信息

-XX:+PrintTLAB

JDK9+不支持

* 1. TLAB大小设置

-XX:TLABSize

* 1. 设置TLAB大小自动调整

-XX:+ResizeTLAB

## 其他参数

* 1. 设置工作模式为客户端

-client

* 1. 设置工作模式为服务端

-server

* 1. 禁用System.gc()（显示GC）

-XX:+DisableExplicitGC

* 1. 设置并发处理显示GC

-XX:+ExplicitGCInvokesConcurrent

* 1. 后台编译

-XX:+BackgroundCompilation

* 1. 开启逃逸分析

-XX:+DoEscapeAnalysis

仅在server模式下有效

* 1. 打开标量替换

-XX:+EliminateAllocation

# 二、分析JVM

## JDK自带

### 基本工具

* + 1. jps

类似Linux的ps命令，只列出当前用户的有权限查看的java程序的进程

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

|  |  |
| --- | --- |
| -q | 只输出进程ID |
| -m | 列出传递给主函数的参数 |
| -l | 主函数完整路径 |
| -v | 列出传递给jvm的参数 |
| -V | 列出通过flags文件传递给jvm的参数 |
| -Joption | 将参数传递给java启动器，EX: jps -J-Xms48M -J-XX:+PrintGCDetails |

* + 1. jstat

查看java运行时信息

https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jstat.html#BEHHGFAE

jstat [Options] ${pid} [ ${interval}[s|ms] [ ${count} ] ]

|  |  |
| --- | --- |
| -help | 帮助信息 |
| -class | 显示classLoader信息 |
| -compiler | 显示jit相关信息 |
| -gc | 显示gc信息 |
| -gccapacity | 显示各个代信息 |
| -gccause | 显示垃圾回收信息 |
| -gcnew | 新生代信息 |
| -gcnewcapacity | 新生代大小 |
| -gcold | 老年代信息和元数据区信息 |
| -gcoldcapacity | 老年代大小 |
| -gcmetacapacity | 显示元数据区大小 |
| -gcutil | 显示垃圾回收信息，同gccause |
| -printcompilation | 显示jit编译的方法信息 |
| -t | 显示程序运行时间 |
| -h | 指定周期性输出数据， |
| pid | 进程ID |
| interval | 周期时间 |
| count | 循环次数 |

* + 1. jinfo

查看java的参数，支持在运行时修改部分参数，这个命令在jdk8中处于实验性质

https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jinfo.html#BCGEBFDD

jinfo [option] pid

|  |  |
| --- | --- |
| no-option | 输出全部的参数和系统属性 |
| -flag<name> | 打印指定参数 |
| -flag [+|-]<name> | 开启或关闭指定参数, Boolean |
| -flag <name>=<value> | 设置指定jvm参数值 |
| - sysprops | 输出系统属性 |

* + 1. jmap

可以生产Java程序的堆dump文件，也可以查看堆统计信息、classLoader信息、finalizer队列

https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jmap.html#CEGCECJB

jmap [options] pid

|  |  |
| --- | --- |
| no option |  |
| -histo[:live] | 打印每个class的实例数目,内存占用,类全名信息. VM的内部类名字开头会加上前缀”\*”. 如果live子参数加上后,只统计活的对象数量 |
| - dump:[live,] format=b, file=filename | 使用hprof二进制形式,输出jvm的heap内容到文件，live参数可选，如果指定，则只输出活对象 |
| -heap | 打印堆摘要、配置、gc算法，打印字符串的数量和大小。 |
| -clstats | 打印classLoader信息 |
| -F | 在没有pid时，强制使用-histo和-dump，同时不支持live |
| -h | 帮助信息 |
| -help | 帮助信息 |
| -Jflag | 传递参数给jmap启动的jvm |

* + 1. jhat在jdk9+被移除，官方进阿姨用visualVm代替
    2. jstack

导出线程栈信息，进行死锁检查

https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jstack.html#BABGJDIF

|  |  |
| --- | --- |
| -F | 强制打印栈信息 |
| -l | 长列表，打印锁的附加信息 |
| -m | 打印Java 和 native C/C++ frames的所有栈信息 |
| -h ｜-help | 打印帮助信息 |

* + 1. jstatd

启用远程监控，需要配置java的安全策略，并保存于jstatd.all.policy文件中

jstatd J-Djava.security.policy=jstatd.all.policy [ options ]

|  |  |
| --- | --- |
| -nr | 找不到RMI注册表时，不尝试创建 |
| -p | 指定端口 |
| -n | RMI名称，默认JstatRemoteHost，如果本地有多个jstatd服务，需要保证唯一 |

* + 1. jcmd

将诊断命令请求发送到正在运行的JVM，用来导出堆、查看Java进程、导出线程信息、执行GC、还可以进行采样分析

https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jcmd.html#CIHEEDIB

jcmd <pid | main-class> <command ... | PerfCounter.print | option>

|  |  |
| --- | --- |
| -l | 列出所有JVM |
| -h ｜-help | 列出JVM支持的命令 |
| -f filename | 从文件中读取命令 |
| PerfCounter.print | 打印目标Java进程上可用的性能计数器 |

### 可视化工具

* + 1. OQL
    2. Visual VM
    3. JConsole
    4. Mission Control

## MAT

## LINUX/MAC OS

### top

### vmstat

### iostat

### pidstat

## Windows

### Perfmon

### Process Explorer

### pslist

# 三、JVM常用执行指令

## 1、汇编基础

## 2、基础指令

# 五、JVM基本原理

## 内存结构

## 类结构、加载、执行

## 垃圾收集器内存分配和算法实现

## 内存模型

## 垃圾收集器

## 编译原理

堆分配参数示意图



Thread Local Allocation Buffer

逃逸分析(Escape Analysis)