

1. 程序计数器：一块儿较小的内存空间，用于存储当前线程执行的字节码和行号指示器，控制指令执行，单个线程私有，不存在线程安全问题
2. Java虚拟机栈：每个方法被执行都会创建一个帧栈用于存储局部变量、方法出口等信息，生命周期与线程相同，线程私有，不存在线程安全问题
3. 本地方法栈：本地方法栈为虚拟机使用到的Native方法(操作系统本地接口)服务
4. Java堆：JVM最大区域，线程共享，用于存放对象实例，GC的主要区域。又可细分为：新生代、老年代，新生代又可分为：Eden区、Survivor from区、Survivor to区，堆空间物理不连续，逻辑连续。Eden：Survivor=8:1
5. 方法区(永久代)：线程共享区域，用于存储已被虚拟机加载的类信息、常量、静态变量、即时编译器编译后的代码等数据。
6. 运行时常量池：方法区的一部分，用于存储常量数据

Vm args

-Xms 堆最小空间

-Xmx 堆最大空间（将 -Xms和-Xmx设置相同，避免堆自动扩展）

-XX:+HeapDumpOnOutOfMemoryError 虚拟机出现内存溢出异常时Dump出当前内存堆转储快照以便进行分析

-XX:+PrintGCDetails 打印gc详情

-XX:PermSize、-XX:MaxPermSize 用来控制方法区的大小，通常设置为相同的值

-Xss 栈容量

-XX:MaxDirectMemorySize=10M本地直接内存，默认和-Xmx大小一样

内存溢出异常处理

1. 堆内存溢出
   1. 根据内存映像分析工具确认内存中的对象是否必要
   2. 通过GC Roots查看泄露对象信息，定位代码位置
   3. 根据系统内存调整-Xmx -Xms大小
2. 栈内存溢出StackOverflowError
3. 运行时常量池溢出

垃圾收集器

1. 引用计数算法：给对象加一个计数器，有引用加1，否则减1，计数器为0的不可能再被使用，无法解决对象互相引用的问题
2. 根搜索算法：以GC Roots作为起点向下搜索，当一个对象不可达，则认为不可用

JDK四种引用方式

1. 强引用，强引用存在GC不会回收对象
2. 软引用，SoftReference类实现软引用
3. 弱引用，WeakReference类实现弱引用
4. 虚引用，PhantomReference类实现虚引用

垃圾收集器算法

1. 标记-清除法，标记需要清除的对象，统一回收，效率低，空间碎片多
2. 复制算法，将可用内存划分相同的块，然后把存活的复制到另一块上面，然后一次性清理已使用过的空间，缺点内存浪费，商用虚拟机用这种方法回收新生代
3. 标记-整理算法，标记清除算法的优化，标记后让存活的对象都向一端移动，直接清理端边界以外的内存
4. 分代收集算法，根据各个年代的特点采用最适当的收集算法，比如新生代每次收集时都会有大批量对象死去，使用复制算法。

虚拟机性能监控与故障处理工具

数据依据：运行日志、异常堆栈、GC日志、线程快照、堆存储快照等

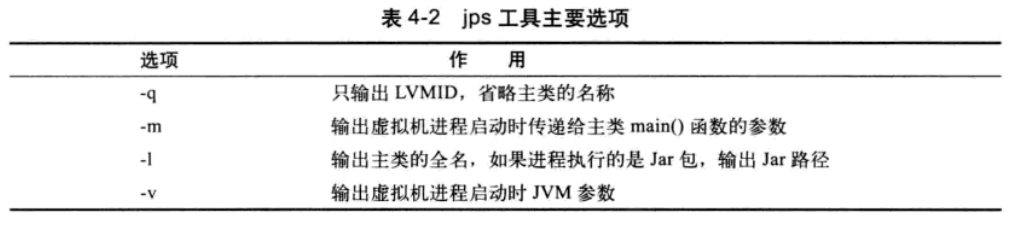
借助tools.jar类库里的接口，我们可以直接在应用程序中实现功能强大的监控分析功能

可以使用JVM 监控工具 Java VisualVM工具远程执行虚拟机监控命令

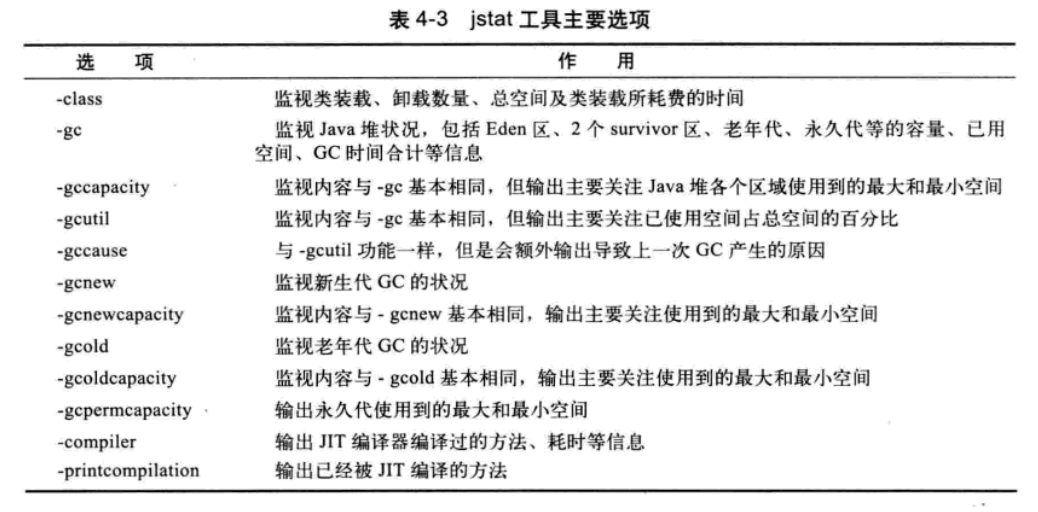
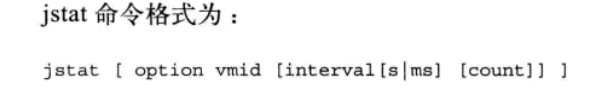
Jdk常用工具命令

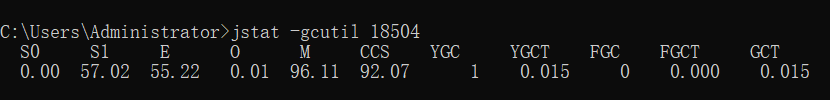
jps:虚拟机进程状况工具，显示正在运行的虚拟机进程，显示主类名称，进程id(VMID)





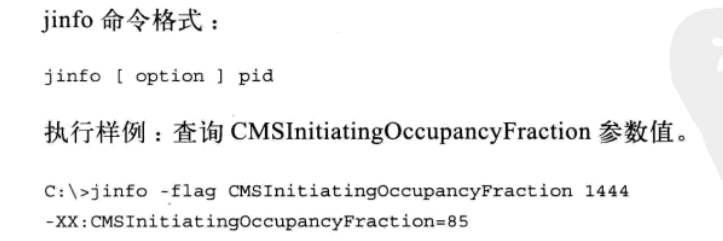
jstat：用于监控虚拟机各种运行状态信息，定位性能问题首选工具



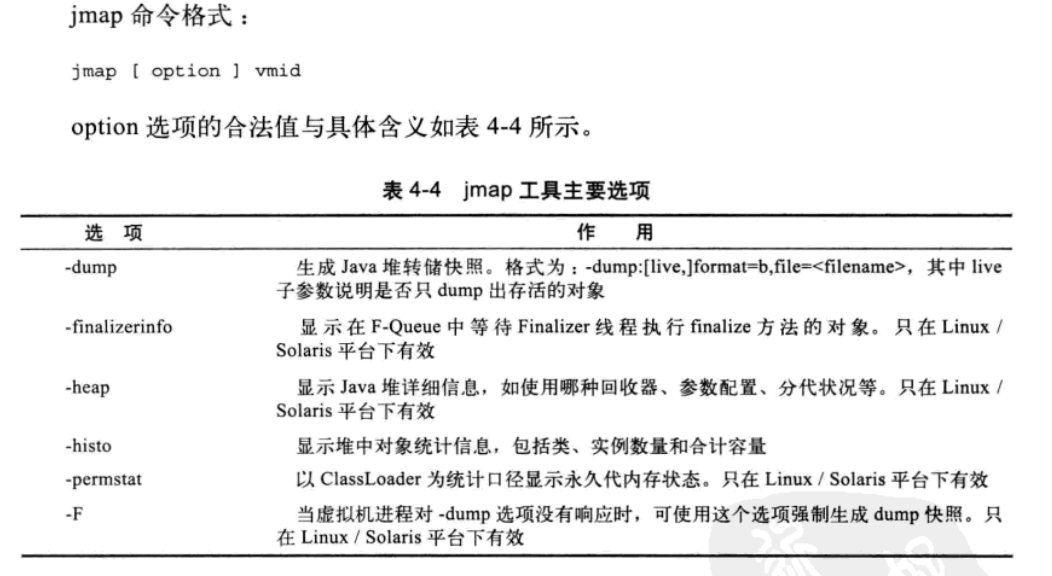


S0、S1表示Survivor0、Survivor1，E表示Eden，O表示老年代

jinfo:实时地查看和调整虚拟机的各项参数



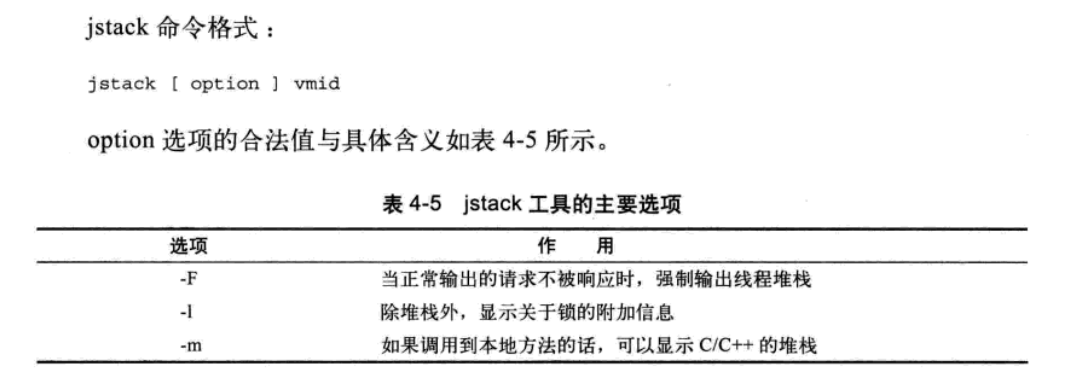
jmap:生成堆转储快照(一般称为heapdump或dump文件)



jhat:虚拟机堆存储快照分析工具



jstack:堆栈跟踪工具，用于生成虚拟机当前时刻的线程快照，主要目的是定位线程出现长时间停顿的原因，如线程死锁、死循环、等待时间长等原因。



jdk可视化工具：jconsole

