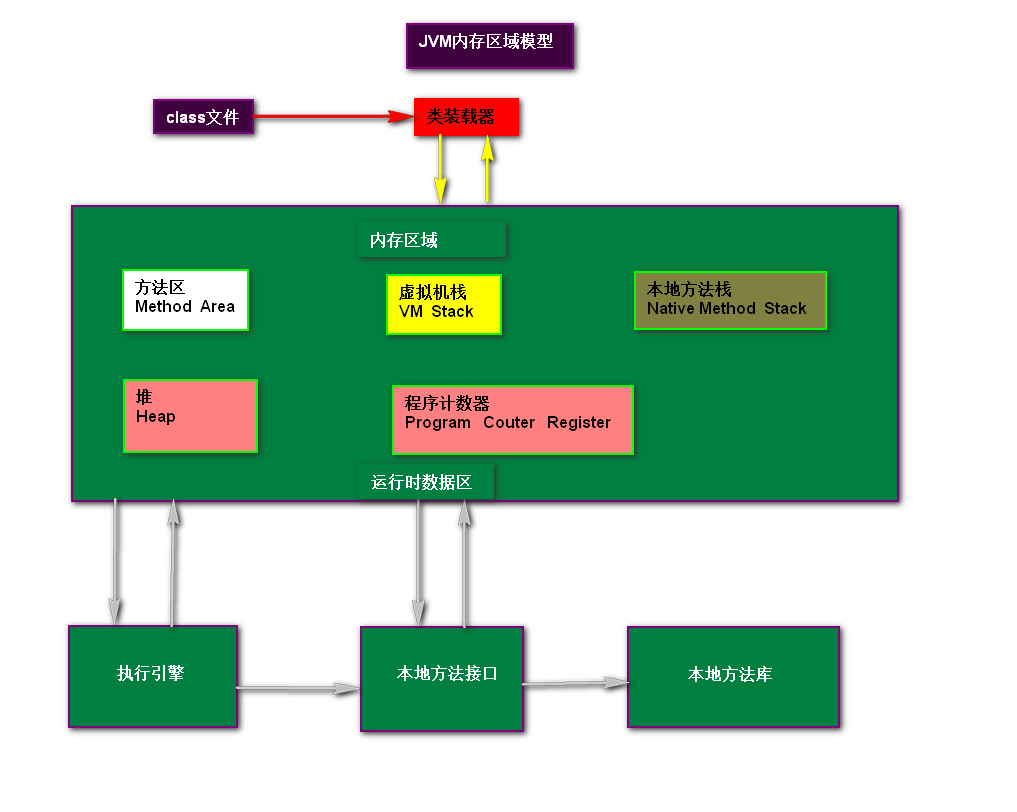
# JVM虚拟机调优

## JVM引言

1. JVM在整个jdk中处于最底层,负责于操作系统的交互,用来屏蔽操作系统环境,提供一个完整的Java运行环境,因此也叫虚拟计算机。操作系统装入JVM是通过jdk中Java.exe来完成的

## JVM内存结构



1. **方法区：**

方法区 (Mthod Area): 也称“永久代(permanent generation)”，“非堆”，用于储存虚拟机加载的类信息，常量，静态变量，是各个线程共享的内存区域,运行时常量池：方法区的一部分，Class文件中除了有类的版本，字段，方法，接口等描述信息外，还有一项信息就是常量池，用于存放编译器生成的各种符号引用，这部分内容将在类加载后放到方法区的运行时常量池中.

1. **栈：**

描述的是java方法执行的内存模型，每个方法被执行的时候，都会创建一个“栈帧”用于存储局部变量（包括参数），操作栈，方法出口等信息。

每个方法被调用到执行完成的过程，就对应着一个栈帧在虚拟机栈中从入栈到出栈的过程。声明周期与线程相同，是线程私有的.局部变量表：存放八种基本类型，对象引用，其中64位长度的long和double类型的数据会占用两个局部变量的空间，其余数据类型只占一个。

局部变量表是在编译时完成分配的，当进入一个方法时，这个方法需要在栈帧中分配多大的局部变量是完全确定的，在运行期间不再改变.

1. **堆(Heap)：**

也叫java堆，CG堆。是JVM中所管理的内存中最大的一块内存区域，是线程共享的，在JVM启动时创建。存放了对象的实例及数组（所有new的对象）

JVM的优化也可以称为堆的优化.

1. **程序计数器：**

是最小的一块内存，它的作用是当前线程所执行的字节码的行号指示器，在虚拟机的模型里，

字节码解释器工作时就是通过改变这个计数器的值来选取下一条需要执行的字节码指令，分支，循环，异常处理，

线程恢复等基础功能都需要依赖计数器完成.

## JVM代的划分

永久代(Permanent Generation)

年轻代(Young Generation)

老年代(Old Generation)

Heap(堆) = 年轻代 (Eden + survivor1+survivor2) + 老年代

## 触发GC和Full GC

GC: 一般情况下，当新对象生成，并且在Eden申请空间失败时，就会触发GC

Full GC:

老年代（Old ）被写满

永久代（Perm）被写满

System.gc()被显式调用

## 优化参数配置

1. **堆设置**

-Xms:初始堆大小

-Xmx:最大堆大小

-XX:NewSize=n:设置年轻代大小

-XX:NewRatio=n:设置年轻代和年老代的比值。如:为3，表示年轻代与年老代比值为1：3，年轻代占整个年轻代年老代和的1/4

-XX:SurvivorRatio=n:年轻代中Eden区与两个Survivor区的比值。注意Survivor区有两个。如：3，表示Eden：Survivor=3：2，一个Survivor区占整个年轻代的1/5

-XX:MaxPermSize=n:设置永久代大小

1. **收集器设置**

-XX:+UseSerialGC:设置串行收集器

-XX:+UseParallelGC:设置并行收集器

-XX:+UseParalledlOldGC:设置并行年老代收集器

-XX:+UseConcMarkSweepGC:设置并发收集器

-XX:+UseG1GC 设置G1收集器

1. **垃圾回收统计信息**

-verbose:gc

-XX:+PrintGC

-XX:+PrintGCDetails

-XX:+PrintGCTimeStamps

-Xloggc:filename

1. **并行收集器设置**

-XX:ParallelGCThreads=n:设置并行收集器收集时使用的CPU数。并行收集线程数。

-XX:MaxGCPauseMillis=n:设置并行收集最大暂停时间

-XX:GCTimeRatio=n:设置垃圾回收时间占程序运行时间的百分比。公式为1/(1+n)

1. **并发收集器设置**

-XX:+CMSIncrementalMode:设置为增量模式。适用于单CPU情况。

-XX:ParallelGCThreads=n:设置并发收集器年轻代收集方式为并行收集时，使用的CPU数。并行收集线程数。

## 优化工具的启动虚拟机

1. **打开工具的.ini加入如下配置**

-verbose:gc

-XX:+PrintGC

-XX:+PrintGCDetails

-XX:+PrintGCDateStamps

-Xloggc:D:\jvm.log

1. **查看jvm.log文件**

查看触发了Gc Full Gc

1. **更改配置**

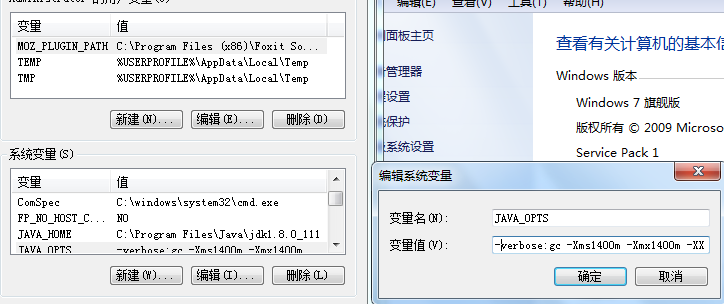
-Xms3096m 设置堆最小空间

-Xmx3096m 设置堆最大空间

-XX:MaxPermSize=256m 设置永久代空间

## Tomcat JVM优化

1. **Windows系统中修改环境变量**

****

-verbose:gc -Xms1400m -Xmx1400m -XX:PermSize=400m

-XX:MaxPermSize=400m -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:D:\jvm.log

1. **Linux中在/etc/profile中加入**

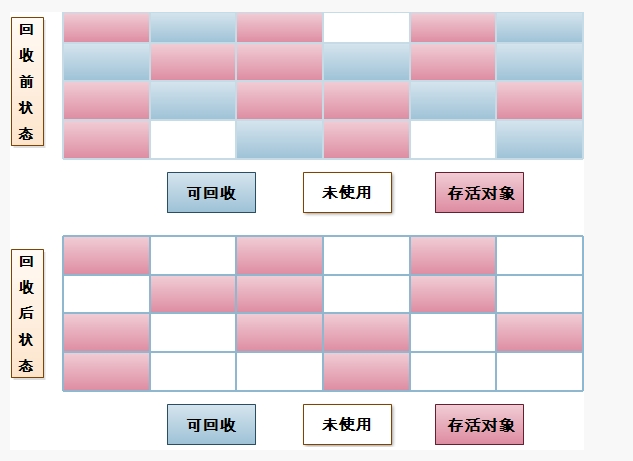
Export -verbose:gc -Xms1400m -Xmx1400m -XX:PermSize=400m

-XX:MaxPermSize=400m -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:D:\jvm.log

## 垃圾回收算法

1. **Mark-Sweep（标记-清除）**

它是最基础的垃圾回收算法，其他算法都是基于这种思想。标记-清除算法分为“标记”，“清除”两个阶段：首先标记出需要回收的对象，标记完成后统一清除对象



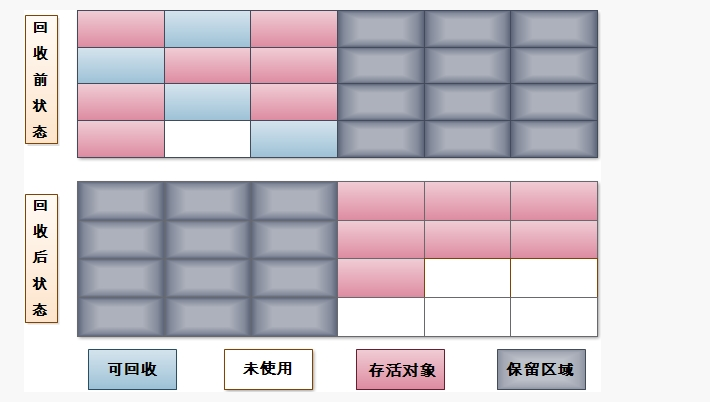
缺点：

1：标记和清除的效率不高

2：标记之后会产生大量不连续的内存碎片

1. **Copying（复制）算法**

它将可用内存分为两块，每次只用其中的一块，当这块内存用完以后，将还存活的对象复制到另一块上面，然后再把已经使用的内存空间一次清理掉



优点：

1：不会产生内存碎片

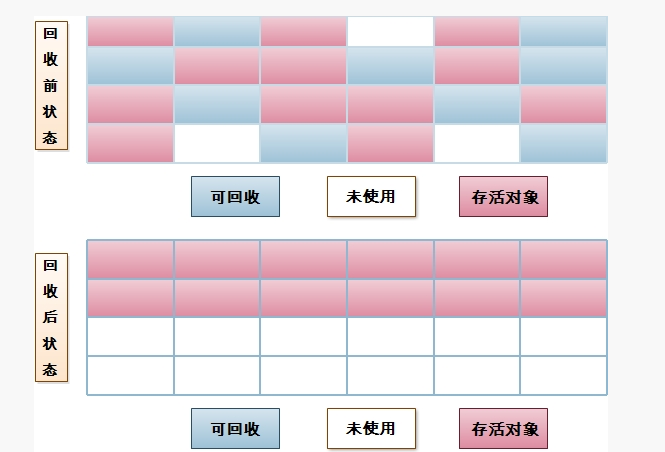
2：只要移动堆顶的指针，按顺序分配内存即可，实现简单，运行高效

缺点：

1：内存缩小为原来的一半

1. **Mark-Compact（标记-整理）算法**

标记操作和”标记-清除“算法一样，后续操作变成不直接清理对象，而是在清理无用对象的时候完成让所有存活的对象都像一端移动，并更新对象的指针



优点：不会产生内存碎片

缺点：在“标记-清除”基础上还要进行对象的移动，成本相对较高

1. **Generational Collection（分代收集）算法（重点）**

是目前大部分JVM的垃圾收集器采用的算法。它的核心思想是根据对象存活的生命周期将内存划分为若干个不同的区域。一般情况下将堆区划分为老年代（Tenured Generation）和新生代（Young Generation），老年代的特点是每次垃圾收集时只有少量对象需要被回收，而新生代的特点是每次垃圾回收时都有大量的对象需要被回收，那么就可以根据不同代的特点采取最适合的收集算法

目前大部分垃圾收集器对于新生代都采取Copying算法，因为新生代中每次垃圾回收都要回收大部分对象，也就是说需要复制的操作次数较少，但是实际中并不是按照1：1的比例来划分新生代的空间的，一般来说是将新生代划分为一块较大的Eden空间和两块较小的Survivor空间，每次使用Eden空间和其中的一块Survivor空间，当进行回收时，将Eden和Survivor中还存活的对象复制到另一块Survivor空间中，然后清理掉Eden和刚才使用过的Survivor空间而由于老年代的特点是每次回收都只回收少量对象，一般使用的是Mark-Compact算法

注意，在堆区之外还有一个代就是永久代（Permanet Generation），它用来存储class类、常量、方法描述等。对永久代的回收主要回收两部分内容：废弃常量和无用的类

## 垃圾收集器

1. **Parallel Scavenge(并行收集器)**

Parallel Scavenge收集器是一个新生代的多线程收集器（并行收集器），它在回收期间不需要暂停其他用户线程，其采用的是Copying算法，它主要是为了达到一个可控的吞吐量

1. **Parallel Old(年老代并行收集器)**

Parallel Old是Parallel Scavnge收集器的老年代版本（并行收集器），使用多线程和Mark-Compact算法

1. **CMS(并发收集器)**

CMS（Current Mark Sweep）收集器是一种以获取最短回收停顿时间为目标的收集器，它是一种并发收集器，采用的是Mark-Sweep算法

1. **G1**

G1收集器是当今收集器技术发展最前沿的成果，它是一款面向服务端应用的收集器，它能充分利用多CPU、多核环境。因此它是一款并行与并发收集器，并且它能建立可预测的停顿时间模型