链滴

JVM 内存模型

作者：someone33881
原文链接：https://ld246.com/article/1548040714225
来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20180414.jpg?imageView2/1/w/960/h/520/interlace/1/q/100">
关键词：程序计数器、java 虚拟机栈、本地方法栈、静态方法区、堆区</
>
<h3 id="问题">问题</h3>
1、栈帧存储的是哪些东西？
<h3 id="一-概述">一、概述</h3>
    程序计数器是较小一块内存空间，用于当
线程所执行的字节码的行号指示器；
    栈分为 java 虚拟机栈和本地方法栈，主要用于
法的执行；
    方法区存储类信息、常量、静态变量</strong
等数据，也称静态方法区，为与 java 堆进行区分也称之为非堆（别名 Non-He
p）
    堆内存由年轻代（默
占堆内存 1/3，又分为 Eden 空间 - 伊甸区（也即新生代）、From Survivor
间 - From 幸存者区、To Survivor 空间 - To 幸存者区，
认 8:1:1 比例分配）和老年代组成，是 JVM 中最大的一块内存
    内存分布：
        Java 内存 ≈Heap(堆内存)+PermGen(方法区)+
hread(栈)
        Heap（堆内存）=Young（年轻
）+Old（老年代）
        Young（年轻代）=EdenSpace
新生代）+From Survivor+To Survivor
    参数设置：
        每个线程栈的大小：-Xss
**   堆内存最小空间大小：-Xms**
**   堆内存最大空间大小：-Xmx**
        新生代最小空间大小：-XX:newSize
        新生代最大空间大小：-XX:maxNewSize
  - （或者**-Xmn**1g，或者 -XX:NewRatio=1）
        老年代空间大小 = 堆内存空间大小 - 新生代空
大小 - 新生代 * （2 * 1/(8 + 1 + 1)）
    注意：
        与 java 线程内存模型进行区分，即线程、主内
、工作内存：
<h3 id="二-Java虚拟机运行时数据区">二、Java 虚拟机运行时数据区</h3>
   1、 程序计数器：当前线程（-> 线程私有</s
rong>）所执行的字节码的行号指示器。
    当前线程正在执行的是一个 java 方法，则
数器记录的是正在执行的虚拟机字节码指令的地址；
    当前线程正在执行的是一个 native 方法，则计数器值
空；
    此内存区域是唯一一个在 Java 虚拟机规范中没有规定任何 OutOfMemoryError
情况的区域。
    2、 Java 虚拟机栈：是描述 Java 方法执行（-&
t; 线程私有）的内存模型（每个方法在执行的同时都会创建一个栈帧</strong
，用于存储局部变量表、操作数栈、动态链接、方法出口等信息），每个方法
调用到执行完成的过程，都是对应着一个栈帧在虚拟机栈中入栈到出栈的过程。
/p>
    局部变量表：存放编译期已知的各种</stro
g>基本数据类型（如 boolean 等）、对象引用（reference 类型，
指向对象起始地址的引用指针，或指向一个代表对象的句柄或其他与此对象相关的位置）、

eturnAddress 类型（为指向了一条字节码指令的地址）；；且 64 位长度的 long 和 double 会占用 2 个局部变量空间（Slot），其余只占用 1 个
    StackOverflowError 异常：当线程请求的栈深度大于虚拟机所允许的深度时抛异常
    OutOfMemoryError 异常：如果虚拟机栈可以动态扩展，且扩展时无法申请到够的内存时抛该异常
  3、本地方法栈：相较于 Java 虚拟机栈（Java 方法）而言，只是虚拟机使用到的 Native 方法服务的（-> 线程私有）；同样，抛 StackOverflowError 和 OutOfMemoryError 异常
  4、方法区：[jdk8 -> Metaspace 内存空间]
    用于存储已被虚拟机加载的类信息、常量、静态变量、即编译器编译后的代码等数据，线程共享的；
    JVM 虚拟机规范中将方法区描述为堆的一个逻辑部分，因此方法区也叫非堆，与堆进行区分；
    Hotspot - 方法区也被 javaer 称为永久代（PermGen），实际上二者并不等，只是因为在 Hotspot 中 GC 分代收集被扩展至了方法区，或者说使用永久代来实现方法区而已

    方法区的内存回收主要是针对常量池的回收和类型的卸载
    当无法满足内存分配需求时，那么将抛出 OutOfMemoryError 常
  5、堆区：
     虚拟机启动时创建且是线程共享的一块内存区域唯一目的就是存储对象实例，几乎所有的对象实例均在此分配；
    是垃圾收集器管理的主要区域，也因此称为 GC 堆（由于 GC 算法都是采用分收集算法，因此也可以细分为年轻代（又分为 Eden 空间、From Survivor 空间、To Survivor 空间默认 8:1:1 比例分配）和老年代）
    当在堆中没有内存完成实例分配，且也无法扩展堆内存，那么将抛出 OutOfMoryError 异常、
<h3 id="三-其他">三、其他</h3>
    一个 2G 堆大小的 JVM，可能总共占据的内存大小如下：
        堆内存 + 线程数 * 线程栈 + 久代 + 二进制代码 + 堆外内存
        2G + 1000 * 1M + 256M + 48/40M + (~2G) = 5.5G (3.5G)
        - 堆内存： 存储 Java 对象，默认为物内存的 1/64
        - 线程栈： 存储局部变量（原子类型引用）及其他，默认为 1M
        - 永久代-方法区： 存储类定义及常量，注意 JDK7/8 的区别
        - 二进制代码：JDK7 与 8，打开多层译时的默认值不一样，从 48 到 240M
        - 堆外内存： 被 Netty，堆外缓存等用，默认最大值约为堆内存大小
<h3 id="四-参考">四、参考</h3>
1、https://blog.csdnnet/liuxinghao/article/details/73963399