

1. 为什么学java内存管理

Java程序是跑在jvm上，了解并掌握java的内存管理可以优化程序的性能

运行时数据区域：栈内存、堆内存和方法区。

**程序计数器**：当前线程所执行的字节码的行号指示器。

**Java虚拟机栈**：是线程私有的，它的生命周期与线程相同。虚拟机栈描述的是Java方法执行的内存模型：每个方法在执行的同时都会创建一个栈帧用于存储局部变量表、操作数栈、动态链表、方法出口信息等。每一个方法从调用直至执行完成的过程，就对应着一个栈帧在虚拟机栈中入栈到出栈的过程。

**局部变量表**：存放了编译器可知的各种基本数据类型(boolean、byte、char、short、int、float、long、double)、对象引用和returnAddress类型(指向了一条字节码指令的地址)。 如果扩展时无法申请到足够的内存，就会抛出OutOfMemoryError异常。

**本地方法栈**：本地方法栈与虚拟机的作用相似，不同之处在于虚拟机栈为虚拟机执行的Java方法服务，而本地方法栈则为虚拟机使用到的Native方法服务。有的虚拟机直接把本地方法栈和虚拟机栈合二为一。

**Java堆**

堆内存用来存放由new创建的对象实例和数组。Java堆是所有线程共享的一块内存区域，在虚拟机启动时创建，此内存区域的唯一目的就是存放对象实例 。Java堆是垃圾收集器管理的主要区域。由于现在收集器基本采用分代回收算法，所以Java堆还可细分为：新生代和老年代。从内存分配的角度来看，线程共享的Java堆中可能划分出多个线程私有的分配缓冲区(TLAB)。Java堆可以处于物理上不连续的内存空间，只要逻辑上连续的即可。在实现上，既可以实现固定大小的，也可以是扩展的。

**Java栈**

在栈内存中保存的是堆内存空间的访问地址，或者说栈中的变量指向堆内存中的变量（Java中的指针）（重点）。

　　Java栈是Java方法执行的内存模型每个方法在执行的同时都会创建一个栈帧的用于存储局部变量表、操作数栈、动态链接、方法出口等信息。每个方法从调用直至执行完成的过程就对应着一个栈帧在虚拟机中入栈和出栈的过程。

**堆和栈的关系**

当在堆中产生了一个数组或者对象时，可以在栈中定义一个特殊的变量，让栈中的这个变量的取值等于数组或对象在堆内存中的首地址，栈中的这个变量就成了数组或对象的引用变量，以后就可以在程序中使用栈中的引用变量来访问堆中的数组或者对象，引用变量就相当于是为数组或者对象起的一个名称。引用变量是普通的变量，定义时在栈中分配，引用变量在程序运行到其作用域之外后被释放。而数组和对象本身在堆中分配，即使程序运行到使用new产生数组或者对象的语句所在的代码块之外，数组和对象本身占据的内存不会被释放，数组和对象在没有引用变量指向它的时候，才变为垃圾，不能在被使用，但仍然占据内存空间不放，在随后的一个不确定的时间被垃圾回收器收走（释放掉）。例如：

Jvm常见参数：

-Xms:设置堆的最小空间大小

-Xmx:设置堆的最大空间大小

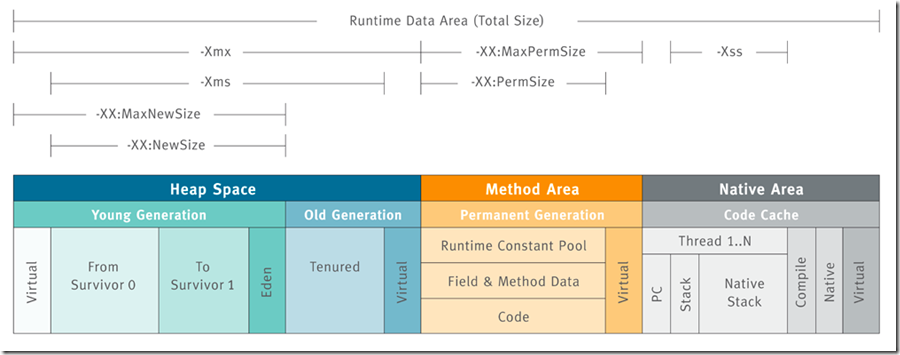
-XX:NewSize设置新生代最小空间大小

-XX:MaxNewSize设置新生代最大空间大小

-XX:PermSize设置永久代最小空间大小

-XX:MaxPermSize设置永久代最大空间大小

-Xss设置每个线程的堆栈大小



类加载机制

<https://www.cnblogs.com/ityouknow/p/5603287.html>

垃圾回收

<https://www.cnblogs.com/ityouknow/p/5614961.html>

IO中的设计模式：主要有装饰器模式和适配器模式

<https://www.cnblogs.com/wxgblogs/p/5649933.html>