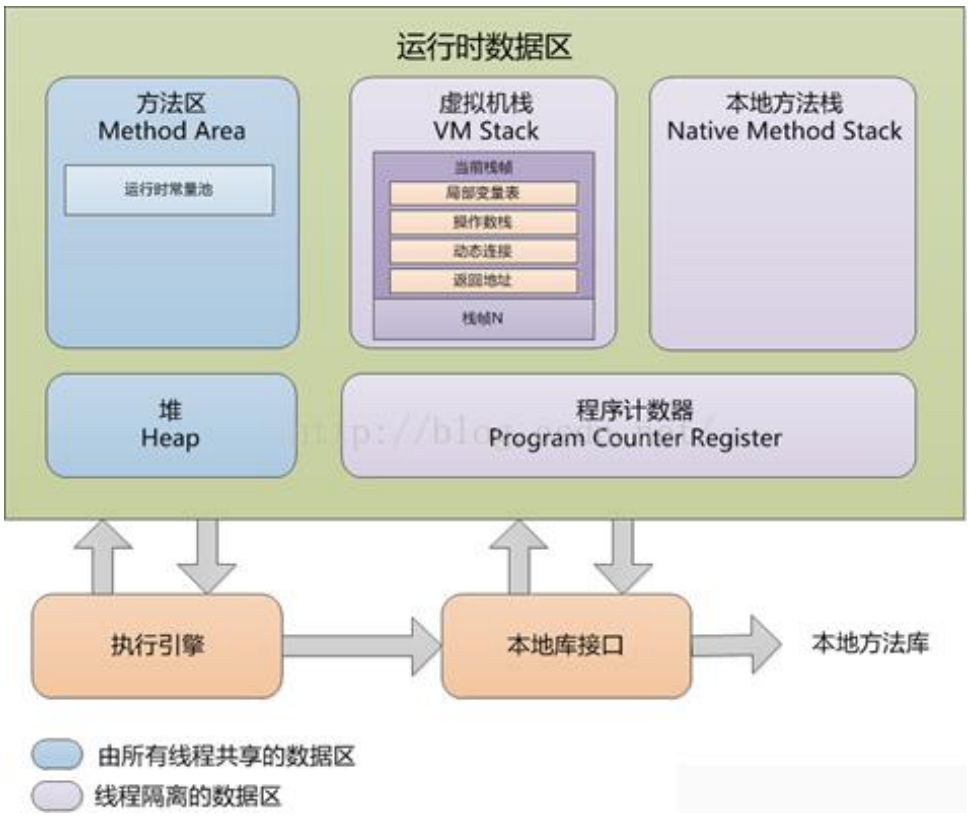
JVM 内存模型分析上篇

注：本篇只介绍 JVM 内存模型，相关概念，区分各块功能

综述：

Java 虚拟机/(Java Virtual Machine=JVM)的内存空间分为五个部分，分别是：
方法区、本地方法栈、 堆、Java 虚拟机栈、程序计数器



一、方法区

Java 虚拟机规范中定义方法区是堆的一个逻辑部分。 方法区中存放已经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。类型信息、字段信息、方法信息、其他信息。

方法区的特点

1. 线程共享： 方法区是堆的一个逻辑部分，因此和堆一样，都是线程共享的。整个虚拟机中只有一个方法区。
2. 永久代： 方法区中的信息一般需要长期存在，而且它又是堆的逻辑分区，因此用堆的划分方法，我们把方法区称为老年代。
3. 内存回收效率低: 方法区中的信息一般需要长期存在，回收一遍内存之后可能只有

少量信息无效。 对方法区的内存回收的主要目标是：对常量池的回收 和 对类型的卸载。

4. Java 虚拟机规范对方法区的要求比较宽松。 和堆一样，允许固定大小，也允许可扩展的大小，还允许不实现垃圾回收

什么是运行时常量池？

方法区中存放三种数据：类信息、常量、静态变量、即时编译器编译后的代码。其中常量存储在运行时常量池中。我们一般在一个类中通过 public static final 来声明一个常量。这个类被编译后便生成 Class 文件，这个类的所有信息都存储在这个 class 文件中。
当这个类被 Java 虚拟机加载后，class 文件中的常量就存放在方法区的运行时常量池中。而且在运行期间，可以向常量池中添加新的常量。如：String 类的 intern()方法就能在运行期间向常量池中添加字符串常量。当运行时常量池中的某些常量没有被对象引用，同时也没有被变量引用，那么就需要垃圾收集器回收。

直接内存?

直接内存是除 Java 虚拟机之外的内存，但也有可能被 Java 使用。在 NIO 中引入了一种基于通道和缓冲的 IO 方式。它可以通过调用本地方法直接分配 Java 虚拟机之外的内存，然后通过一个存储在 Java 堆中的 DirectByteBuffer 对象直接操作该内存，而无需先将外面内存中的数据复制到堆中再操作，从而提升了数据操作的效率。直接内存的大小不受 Java 虚拟机控制，但既然是内存，当内存不足时就会抛出 OOM 异常。

二、Java 虚拟机栈(JVM Stack)

Java 虚拟机栈是描述 Java 方法运行过程的内存模型。
Java 虚拟机栈会为每一个即将运行的 Java 方法创建一块叫做“栈帧”的区域，这块区域用于存储该方法在运行过程中所需要的一些信息，这些信息包括：局部变量表 、存放基本数据类型变量、引用类型的变量、returnAddress 类型的变量、操作数栈、动态链接、方法出口信息等

当一个方法即将被运行时，Java 虚拟机栈首先会在 Java 虚拟机栈中为该方法创建一块“栈帧”，栈帧中包含局部变量表、操作数栈、动态链接、方法出口信息等。
当方法在运行过程中需要创建局部变量时，就将局部变量的值存入栈帧的局部变量表中。
当这个方法执行完毕后，这个方法所对应的栈帧将会出栈，并释放内存空间。

注意：人们常说，Java 的内存空间分为“栈”和“堆”，栈中存放局部变量，堆中存放对象。
这句话不完全正确！这里的“堆”可以这么理解，但这里的“栈”只代表了 Java 虚拟机栈中的局部变量表部分。真正的 Java 虚拟机栈是由一个个栈帧组成，而每个栈帧中都拥有：局部变量表、操作数栈、动态链接、方法出口信息。

Java 虚拟机栈的特点

1. 局部变量表的创建是在方法被执行的时候，随着栈帧的创建而创建。而且，局部变量表的大小在编译时期就确定下来了，在创建的时候只需分配事先规定好的大小即可。此外，在方法运行的过程中局部变量表的大小是不会发生改变的。
2. Java 虚拟机栈会出现两种异常：StackOverFlowError 和 OutOfMemoryError。
   a) StackOverFlowError：
   若 Java 虚拟机栈的内存大小不允许动态扩展，那么当线程请求栈的深度超过当前 Java 虚拟机栈的最大深度的时候，就抛出 StackOverFlowError 异常。
   b) OutOfMemoryError：
   若 Java 虚拟机栈的内存大小允许动态扩展，且当线程请求栈时内存用完了，无法再动态扩展了，此时抛出 OutOfMemoryError 异常。
3. Java 虚拟机栈也是线程私有的，每个线程都有各自的 Java 虚拟机栈，而且随着线程的创建而创建，随着线程的死亡而死亡。

注：StackOverFlowError 和 OutOfMemoryError 的异同？
StackOverFlowError 表示当前线程申请的栈超过了事先定好的栈的最大深度，但内存空间可能还有很多。
而 OutOfMemoryError 是指当线程申请栈时发现栈已经满了，而且内存也全都用光了。

三、本地方法栈

本地方法栈和 Java 虚拟机栈实现的功能类似，只不过本地方法区是本地方法运行的内存模型。本地方法被执行的时候，在本地方法栈也会创建一个栈帧，用于存放该本地方法的局部变量表、操作数栈、动态链接、出口信息。
方法执行完毕后相应的栈帧也会出栈并释放内存空间。
也会抛出 StackOverFlowError 和 OutOfMemoryError 异常。

四、 堆

堆是用来存放对象的内存空间。 几乎所有的对象都存储在堆中。

堆的特点

1. 线程共享
   整个 Java 虚拟机只有一个堆，所有的线程都访问同一个堆。而程序计数器、Java 虚拟机栈、本地方法栈都是一个线程对应一个的。
2. 在虚拟机启动时创建
3. 垃圾回收的主要场所。
4. 可以进一步细分为：新生代、老年代。
   新生代又可被分为：Eden、From Survior、To Survior。

不同的区域存放具有不同生命周期的对象。这样可以根据不同的区域使用不同的垃圾回收算法，从而更具有针对性，从而更高效。

5. 堆的大小既可以固定也可以扩展，但主流的虚拟机堆的大小是可扩展的，因此当线程请求分配内存，但堆已满，且内存已满无法再扩展时，就抛出 OutOfMemoryError。

五、程序计数器

程序计数器是一块较小的内存空间，可以把它看作当前线程正在执行的字节码的行号指示器。也就是说，程序计数器里面记录的是当前线程正在执行的那一条字节码指令的地址。
注：但是，如果当前线程正在执行的是一个本地方法，那么此时程序计数器为空。

这里记录了线程执行的字节码的行号，在分支、循环、跳转、异常、线程恢复等都依赖这个计数器。程序计数器有两个作用：

1. 字节码解释器通过改变程序计数器来依次读取指令，从而实现代码的流程控制，如：顺序执行、选择、循环、异常处理。
2. 在多线程的情况下，程序计数器用于记录当前线程执行的位置，从而当线程被切换回来的时候能够知道该线程上次运行到哪儿了。

程序计数器的特点

1. 是一块较小的存储空间
2. 线程私有。每条线程都有一个程序计数器。
3. 是唯一一个不会出现 OutOfMemoryError 的内存区域。
4. 生命周期随着线程的创建而创建，随着线程的结束而死亡。

补充：强引用，软引用和弱引用的区别

强引用：
只有这个引用被释放之后，对象才会被释放掉，只要引用存在，垃圾回收器永远不会回收，这是最常见的 New 出来的对象。

软引用：
内存溢出之前通过代码回收的引用。软引用主要用户实现类似缓存的功能，在内存足够的情况下直接通过软引用取值，无需从繁忙的真实来源查询数据，提升速度；当内存不足时，自动删除这部分缓存数据，从真正的来源查询这些数据。

弱引用：
第二次垃圾回收时回收的引用，短时间内通过弱引用取对应的数据，可以取到，当执行过第二次垃圾回收时，将返回 null。弱引用主要用于监控对象是否已经被垃圾回收器标记为即将回收的垃圾，可以通过弱引用的 isEnQueued 方法返回对象是否被垃圾回收器标记。

综上所述

1. Java 虚拟机的内存模型中一共有两个“栈”，分别是：Java 虚拟机栈和本地方法栈。两个“栈”的功能类似，都是方法运行过程的内存模型。并且两个“栈”内部构造相同，都是线程私有。
   只不过 Java 虚拟机栈描述的是 Java 方法运行过程的内存模型，而本地方法栈是描述 Java 本地方法运行过程的内存模型。
2. Java 虚拟机的内存模型中一共有两个“堆”，一个是原本的堆，一个是方法区。方法区本质上是属于堆的一个逻辑部分。堆中存放对象，方法区中存放类信息、常量、静态变量、即时编译器编译的代码。
3. 堆是 Java 虚拟机中最大的一块内存区域，也是垃圾收集器主要的工作区域。
4. 程序计数器、Java 虚拟机栈、本地方法栈是线程私有的，即每个线程都拥有各自的程序计数器、Java 虚拟机栈、本地方法区。并且他们的生命周期和所属的线程一样。而堆、方法区是线程共享的，在 Java 虚拟机中只有一个堆、一个方法栈。并在 JVM 启动的时候就创建，JVM 停止才销毁。

名称	特征	作用	配置	异常
栈区	线程私有，使用一段连续的内存空间	存放局部变量表、操作栈、动态链接、方法出口	-XSs	StackOverflowError OutOfMemoryError
堆	线程共享，生命周期与虚拟机相同	保存对象实例	-Xms -Xmx -Xmn	OutOfMemoryError
程序计数器	线程私有、占用内存小	字节码行号	无	无
方法区	线程共享	存储类加载信息、常量、静态变量等	-XX:PermSize -XX:MaxPermSize	OutOfMemoryError

欢迎进大牛群交流学习：387063215