YSOcean

博客园 联系 管理 随笔 - 221 文章 - 0 评论 - 1064

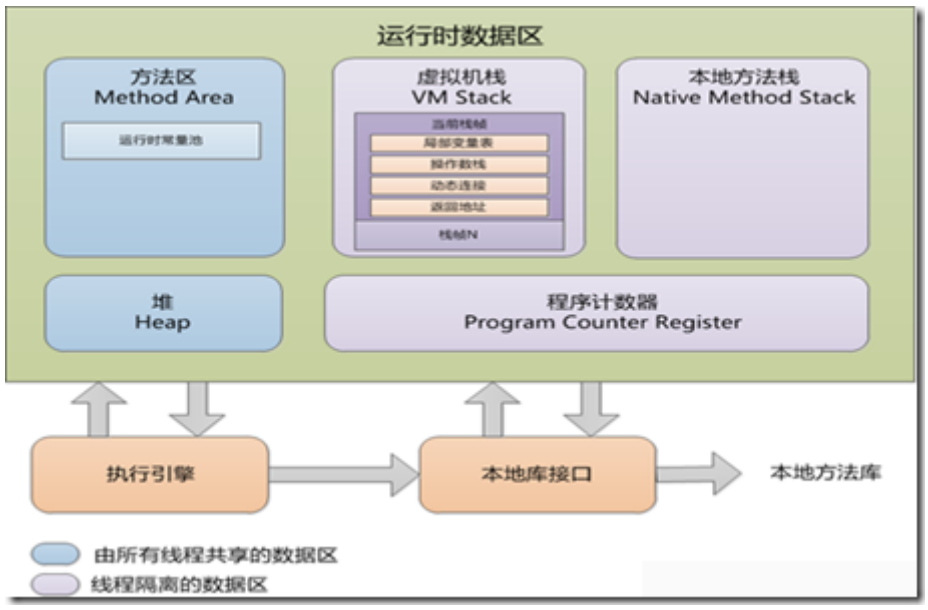
JVM 运行时的内存分配

译

首先我们必须要知道的是 Java 是跨平台的。而它之所以跨平台就是因为 JVM 不是跨平台的。JVM 建立了 Java 程序和操作系统之间的桥梁，JVM 是用 C 语言编写，而 C 语言不具备跨平台的特性。所以对于 Windows 平台，Java 有基于 Windows 平台的 JVM；对于 Linux 平台，Java 也有基于 Linux 平台的 JVM等等。不同的操作系统有不同的 JVM，所以我们编写的 Java 代码能在各个平台上运行，是因为有各个平台的 JVM。

而 Java 的内存分配也是在 JVM 中进行的。JVM 是 Java 内存分配的原理和前提。

Java 程序为了提高程序的效率，对数据进行了不同空间的分配，具体划分为如下 5 个内存空间。



1、程序计数器（Program Counter Register）

它是一块较小的内存空间，它的作用可以看做是当先线程所执行的字节码的信号指示器。每一条JVM线程都有自己的PC寄存器，各条线程之间互不影响，独立存储，这类内存区域被称为“线程私有”内存在任意时刻，一条JVM线程只会执行一个方法的代码。该方法称为该线程的当前方法（Current Method）；如果该方法是java方法，那PC寄存器保存JVM正在执行的字节码指令的地址；如果该方法是native，那PC寄存器的值是undefined。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（Java Virtual Machine Stack）

与PC寄存器一样，Java虚拟机栈也是线程私有的。每一个JVM线程都有自己的java虚拟机栈，这个栈与线程同时创建，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：每个方法被执行的时候都会同时创建一个栈帧（Stack Frame）用于存储局部变量表、操作数栈、动态链接、方法出口等信息。每一个方法被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从入栈到出栈的过程。JVM stack 可以被实现成固定大小，也可以根据计算动态扩展。如果采用固定大小的JVM stack设计，那么每一条线程的JVM Stack容量应该在线程创建时独立地选定。JVM实现应该提供调节JVM Stack初始容量的手段；如果采用动态扩展和收缩的JVM Stack方式，应该提供调节最大、最小容量的手段。如果线程请求的栈深度大于虚拟机所允许的深度将抛出StackOverflowError；如果JVM Stack可以动态扩展，但是在尝试扩展时无法申请到足够的内存时抛出OutOfMemoryError。

通常来说栈存放的是局部变量，包括：

①、保存基本数据类型的值

②、保存对象的引用

3、本地方法栈（Native Method Stack）

本地方法栈与虚拟机栈作用相似，后者为虚拟机执行Java方法服务，而前者为虚拟机用到的Native方法服务。虚拟机规范对于本地方法栈中方法使用的语言，使用方式和数据结构没有强制规定，甚至有的虚拟机（比如HotSpot）直接把二者合二为一。本地方法栈抛出的异常跟上面的虚拟机栈一样。

4、Java堆（Java Heap）

虚拟机管理的内存中最大的一块，同时也是被所有线程所共享的，它在虚拟机启动时创建，这货存在的意义就是存放对象实例，几乎所有的对象实例以及数组都要在这里分配内存。这里面的对象被自动管理，也就是俗称的GC（Garbage Collector）所管理。用就是了，有GC扛着呢，不用操心销毁回收的事儿。Java堆的容量可以是固定大小，也可以随着需求动态扩展（-Xms和-Xmx），并在不需要过多空间时自动收缩。Java堆所使用的内存不需要保证是物理连续的，只要逻辑上是连续的即可。JVM实现应当提供给程序员调节Java 堆初始容量的手段，对于可动态扩展和收缩的堆来说，则应当提供调节其最大和最小容量的手段。如果堆中没有内存完成实例分配并且堆也无法扩展，就会抛OutOfMemoryError。

堆存放的是所有 new 出来的东西，注意：这里创建出来的对象只包括属于各自的 成员变量，不包括成员方法。因为同一个类的对象拥有各自的成员变量，存储在各自的堆中，但是他们共享该类的方法，并不是每创建一个对象就把成员变量复制一遍。

5、方法区（Method Area）

跟堆一样是被各个线程共享的内存区域，用于存储以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。虽然这个区域被虚拟机规范把方法区描述为堆的一个逻辑部分，但是它的别名叫非堆，用来与堆做一下区别。方法区在虚拟机启动的时候创建。方法区的容量可以是固定大小的，也可以随着程序执行的需求动态扩展，并在不需要过多空间时自动收缩。方法区在实际内存空间中可以是不连续的。Java虚拟机实现应当提供给程序员或者最终用户调节方法区初始容量的手段，对于可以动态扩展和收缩方法区来说，则应当提供调节其最大、最小容量的手段。当方法区无法满足内存分配需求时就会抛OutOfMemoryError。

5.1 运行时常量池（Runtime Constant Pool）

它是方法区的一部分。Class文件中除了有类的版本、字段、方法、接口等描述等信息外，还有一项信息是常量池（Constant Pool Table），用于存放编译期生成的各种字面量和符号引用，这部分内容将在类加载后存放到方法区的运行时常量池中。Java虚拟机对Class文件的每一部分（自然也包括常量池）的格式都有严格的规定，每一个字节用于存储哪种数据都必须符合规范上的要求，这样才会被虚拟机认可、装载和执行。但对于运行时常量池，Java虚拟机规范没有做任何细节的要求，不同的提供商实现的虚拟机可以按照自己的需要来实现这个内存区域。不过，一般来说，除了保存Class文件中描述的符号引用外，还会把翻译出来的直接引用也存储在运行时常量池中。运行时常量池相对于Class文件常量池的另外一个重要特征是具备动态性，Java语言并不要求常量一定只能在编译期产生，也就是并非预置入Class文件中常量池的内容才能进入方法区运行时常量池，运行期间也可能将新的常量放入池中，这种特性被开发人员利用得比较多的便是String类的intern()方法。既然运行时常量池是方法区的一部分，自然会受到方法区内存的限制，当常量池无法再申请到内存时会抛出OutOfMemoryError异常。

分类: Java SE

标签: JVM内存分配

« 上一篇： Sublime Text 使用教程
» 下一篇： JS 中 cookie 的使用

posted @ 2017-04-24 23:21 YSOcean 阅读(2299) 评论(2) 编辑 收藏

评论列表

#1楼 2018-07-06 14:49 祎丫丫

随笔分类

Java SE(22)
JavaWeb(34)
Java关键字(6)
Java数据结构和算法(15)
Java虚拟机(11)

最新评论

1. Re:Java设计模式之（一）------单例模式

想问一下单例模式之静态内部类这部分，利用静态内部类，创建类实例。是否违反了静态不能访问非静态呢?

--明月知我心

2. Re:Java设计模式之（一）------单例模式

楼主写的很好，能多出几种设计模式的文章吗，坐等

--zhf123

3. Re:Spring详解（八）-----事务管理

@我要成为大牛vip 扫描当前你配置的类，就是这个类所有的方法。如果打在方法上，那就是这个方法生效...

--originator

4. Re:Spring详解（二）-----IOC控制反转

楼主， bean的生命周期 内容是不是 太少了点。

--originator

5. Re:Spring详解（二）-----IOC控制反转

@卓tr1ste 应该是表明，静态工厂方法方式中， HelloStaticFactory 构造器， 不会被调用。 个人意见哈...

--originator

阅读排行榜