链滴

[译] 一文了解 JVM

作者：valarchie
原文链接：https://ld246.com/article/1577004839254
来源网站：链滴

原文地址：《The JVM Architecture Explained 》
每个 Java 开发者都知道字节码是被 JRE(Java 运行时环境)所执行的。但是许多人并不知道实际上 JRE 是 Java 虚拟机（JVM）的实现。它分析字节码，解释代码并执行它。作为一个开发者，了解 JVM 的体系结构是非常重要的，因为它使我们能更有效的编写代码。在这篇文章中，我们将更深入的了解 Java 中的 JVM 体系结构和 JVM 的不同组件。
<h3 id="什么是JVM-">什么是 JVM?</h3>
虚拟机是物理机器的软件实现。Java 语言当时是基于编写一次到处运行的理念被开发出来的，中到处运行指的是运行在虚拟机上。编译器将 java 文件编译成.class 文件，然后.class 文件被输入到 VM 中，JVM 会加载并运行这些.class 文件。下图展示了 JVM 的体系结构。
<blockquote>
JVM 体系结构


</blockquote>
<h3 id="JVM是如何工作的-">JVM 是如何工作的？</h3>
正如上图展示的，JVM 被分为以下三个主要的子系统：

类加载器
运行时数据区域
执行引擎

<h4 id="1-类加载器">1.类加载器</h4>
Java 的动态类加载功能是由类加载子系统处理。它加载，链接，并初始化类（当在运行时第一引用该类时）。
<h5 id="1-1加载">1.1 加载</h5>
类都通过该组件进行加载。BootStrap ClassLoader、 Extension ClassLoader、Application ClassLoader 这三个类加载器协同完成这个目标。

BootStrap ClassLoader：负责从引导类路径加载类，除了 rt.jar。将给予此加载程序最高优先级
Extension ClassLoader：负责加载 ext 文件夹中的类。
Application ClassLoader：负责从应用程序类路径加载类。

以上这些类加载器将遵循双亲委托机制装载类文件。（即当一个类需要加载时，先从 Bootstrap 加载，没有的话，再从 Extension 加载，还是没有的话，才从 Application 加载。）
<h5 id="1-2链接">1.2 链接</h5>

验证：字节码验证器将会验证生成的字节码正确与否。如果验证失败，我们将得到验证错误信息
准备：为所有静态变量开辟内存空间并赋默认值。
解析：将所有符号引用替换为方法区中的原始引用。

<h5 id="1-3初始化">1.3 初始化</h5>
这是类加载的最后阶段。所有静态变量将会被赋予初始值并且静态构造块将会被执行。
<h4 id="2-运行时数据区">2.运行时数据区</h4>
运行时数据区被分为五个主要的区域。

方法区：所有类级别的数据将会被存于此处。包括静态变量。一个虚拟机仅有一个方法区。它属共享资源。
堆区：所有对象及对象中的实例变量和数组都将存于此处。一个虚拟机也仅有一个堆区。因为方区和堆区在多线程情况下是共享内存的，所以存于其中的数据并非线程安全的。

虚拟机栈：对于每一个线程，都将创建一个单独的运行时方法栈。对于每一个方法调用，栈中都生成一个栈帧。所有本地变量都将在栈中创建。栈区域是线程安全的，因为它并非共享资源。栈帧又分为三个主要组成：

局部变量数组：与方法相关，涉及多少局部变量以及相应的值将存在这里。
操作数栈：如果需要执行任何中间操作，操作数栈充当运行时工作区来执行操作。
帧数据：与该方法对应的所有符号都存储在这里。在任何异常的情况下，catch 块信息将会保存帧数据中。

程序计数器：每个线程都拥有独立的程序计数器。它保存当前执行指令的地址，一旦指令执行，序计数器将更新为下一条指令。
本地方法栈：本地方法栈保存了本地方法调用信息。对于每一个线程，都会创建独立的本地方法。

<h4 id="3-执行引擎">3.执行引擎</h4>
分配到运行时数据区的字节码将会被执行引擎执行。执行引擎将会读取字节码并逐个执行。

解释器：解释器解释字节码快，但执行慢。并且在多次调用同一个方法时，都需要重新解释。</
>

JIT 编译器：JIT 编译器弥补了解释器的缺点。执行引擎会借助解释器去转换字节码。但是当它发有重复的代码时，会使用 JIT 去编译整份字节码并将它改为本地代码。本地代码将会被直接使用在重的方法调用，从而提高系统的性能。

中间代码生成器：生成中间代码
代码优化器：负责优化中间代码
目标代码生成器：负责生成机器码或本地代码
分析器：特殊组件，负责发现热点代码，即方法是否被多次调用。

垃圾回收器：回收和删除未被引用的对象。垃圾回收可被 System.gc()方法触发，但是不保证立执行。

