JVM底层原理解析

Original 极客重生 极客重生 2021-07-04 16:50

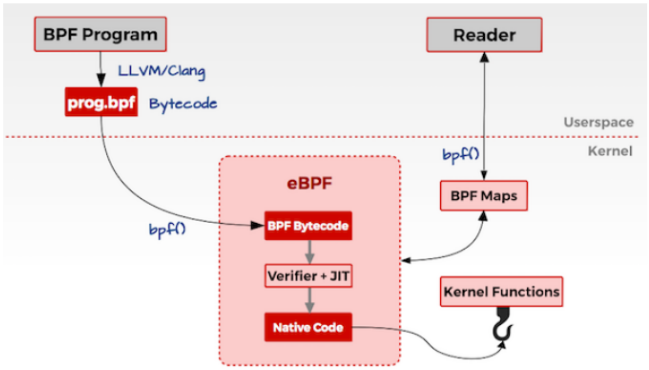
收录于合集

#深入理解编程语言

21个

hi ，大家好，我是极客重生的Alex，今天分享一篇JVM底层原理的文章，希望可以帮助大家了解一下语言虚拟机一些设计原理，不管你当前使用语言是C，C++，Golang，Python等，很多思想和原理是相通的，可以借鉴。

比如之前分析内核虚拟机eBPF架构：



详细请看：Linux网络新技术基石 |eBPF and XDP

在本文中，您将学习

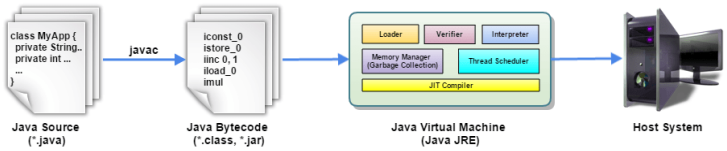
- JVM架构
- 软件代码编译执行流程
- C代码编译执行过程
- Java代码编译执行过程
- 为什么Java既是解释型语言又是编译型语言?
- 为什么 Java 很慢?

什么是JVM?

Java 虚拟机 (JVM)是提供运行时环境来驱动 Java 代码或应用程序的引擎。它将 Java 字节码转换为机器语言。JVM 是 Java 运行环境 (JRE) 的一部分。在其他编程语言中，编译器为特定系统生成机器代码。但是，Java编译器为称为Java 虚拟机的虚拟机生成代码。

JVM的工作原理

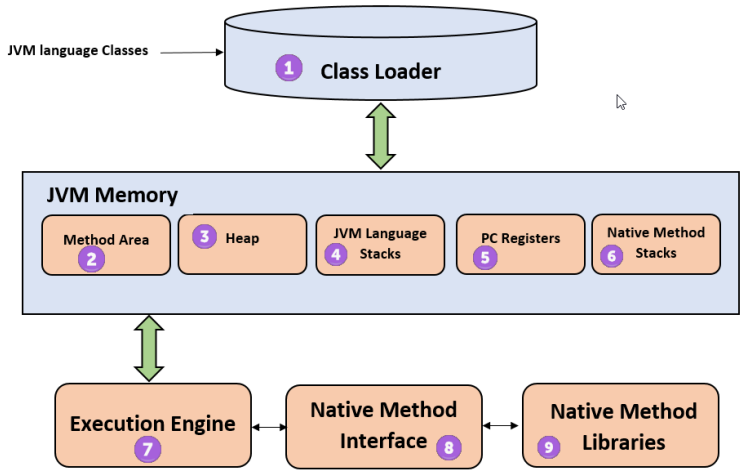
首先，Java 代码被编译成字节码，这个字节码在不同的机器上被解释，在主机系统和 Java 源代码之间，字节码是一种中介语言，Java 中的 JVM 负责分配内存空间。



Java 虚拟机 (JVM) 的工作

JVM架构

让我们了解 JVM 的架构。Java 中的 JVM 架构包含类加载器、内存区、执行引擎等。



Java 虚拟机架构

1) 类加载器

类加载器是用于加载类文件的子系统。它执行三个主要功能，即。加载、链接和初始化。

2) 方法区

JVM 方法区存储类结构，如元数据、常量运行时池和方法代码。

3) 堆

所有的Objects及其相关的实例变量和数组都存储在堆中。此内存是通用的，并在多个线程之间共享。

4) JVM 语言栈

Java 语言堆栈存储局部变量，和部分结果，每个线程都有自己的 JVM 堆栈，在创建线程时同时创建。每当调用方法时都会创建一个新的，并在方法调用过程完成时将其删除。

5) PC 寄存器

PC 寄存器存储当前正在执行的 Java 虚拟机指令的地址。在 Java 中，每个线程都有其独立的 PC 寄存器。

6) 本地方法栈

本机方法栈持有本机代码的指令取决于本机库。它是用另一种语言而不是 Java 编写的。

7) 执行引擎

它是一种用于测试硬件、软件或完整系统的软件。测试执行引擎从不携带有关被测产品的任何信息。

8) 本地方法接口

本机方法接口是一个编程框架。它允许在 JVM 中运行的 Java 代码由库和本机应用程序调用。

9) 本地方法库

本机库是执行引擎所需的本机库（C、C++）的集合。

软件代码编译执行流程

为了编写和执行软件程序，您需要以下内容

1) 编辑器 – 要输入您的程序，可以使用记事本。

2) 编译器 ––将您的高级语言程序转换为本地机器代码。

3) 链接器 ––将主程序中的不同程序文件引用组合在一起。

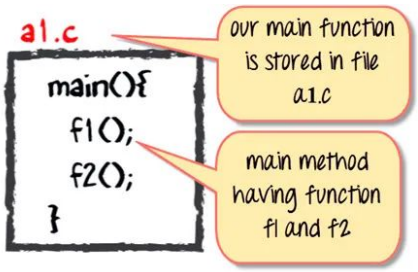
4) Loader – 将您的辅助存储设备（如硬盘、闪存驱动器、CD ）中的文件加载到 RAM 中以供执行。执行代码时会自动完成加载。

5) 执行– 由您的操作系统和处理器处理的代码的实际执行。

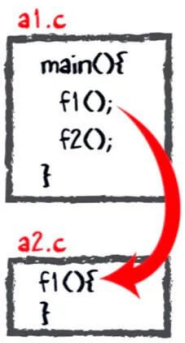
C代码编译执行过程

了解Java中的Java编译过程。首先让我们快速了解一下 C 中的编译和链接过程。

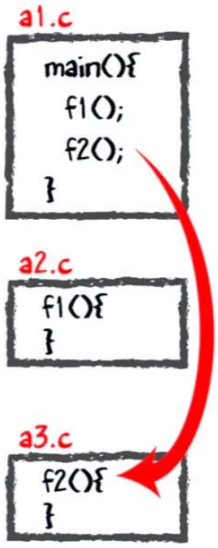
假设在 main 中，您调用了两个函数 f1 和 f2。main 函数存储在文件 a1.c 中。



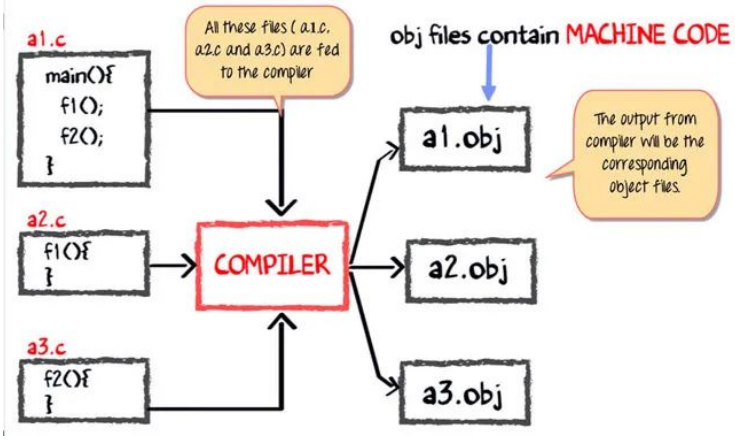
函数 f1 存储在文件 a2.c 中



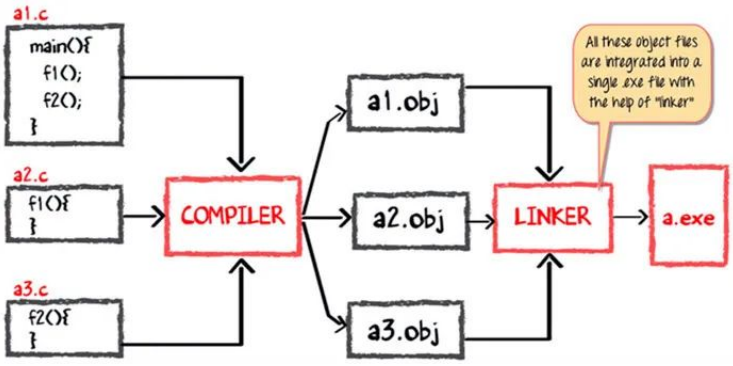
函数 f2 存储在文件 a3.c 中



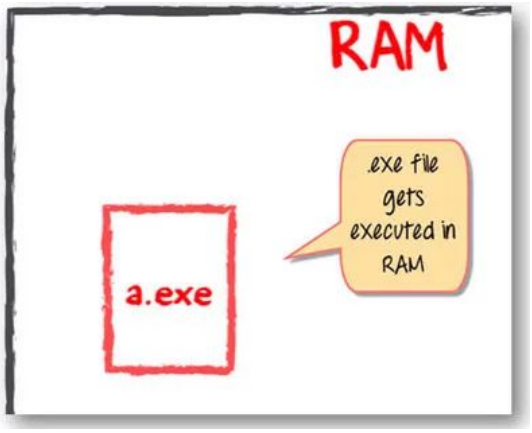
所有这些文件，即 a1.c、a2.c 和 a3.c，都被提供给编译器。其输出是相应的目标文件，即机器代码。



下一步是在链接器的帮助下将所有这些目标文件集成到一个 .exe 文件中。链接器会将所有这些文件组合在一起并生成 .exe 文件。



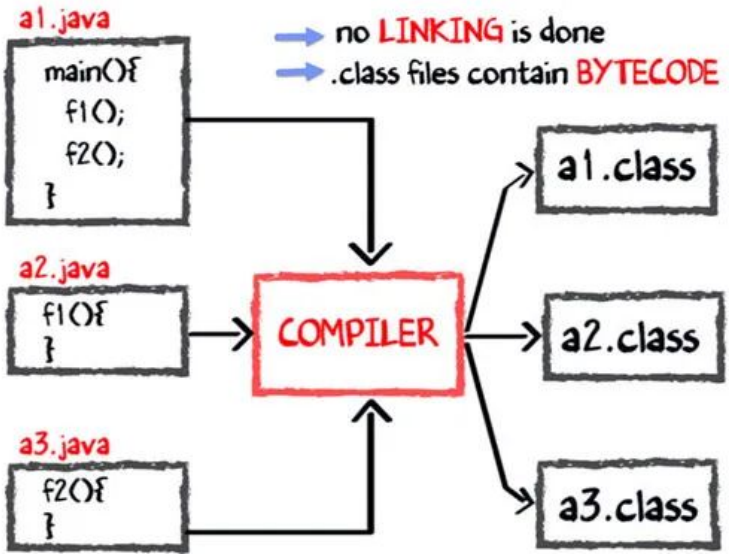
在程序运行过程中，加载程序会将 a.exe 加载到 RAM 中执行。



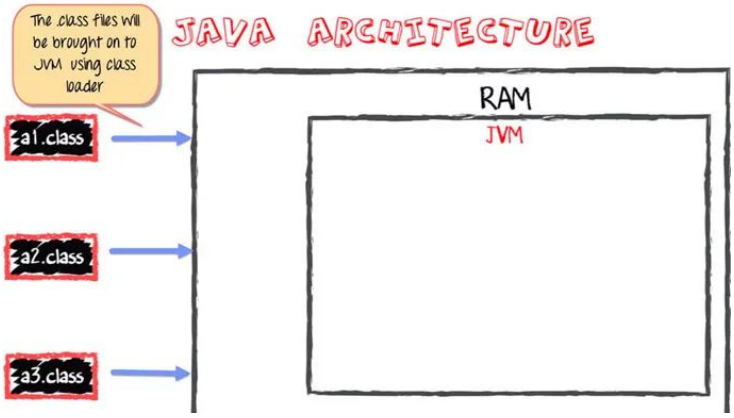
Java VM 中的 Java 代码编译和执行

让我们看看 JAVA 的编译执行过程。在您的主文件中，您有两个方法 f1 和 f2。

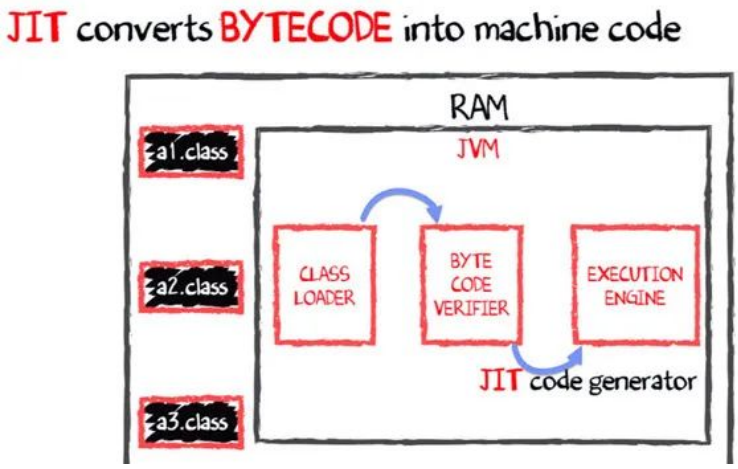
- main方法存放在文件a1.java中
- f1 作为 a2.java 存储在文件中
- f2 作为 a3.java 存储在文件中



编译器将编译这三个文件，并生成 3 个对应的 .class 文件，其中包含字节码。与 C 不同，没有进行链接。Java VM 或 Java 虚拟机驻留在 RAM 上。在执行期间，使用类加载器将类文件带到 RAM 中。字节码已验证是否存在任何安全漏洞。



接下来，执行引擎会将字节码转换为本地机器码。这只是及时编译，这是Java相对较慢的主要原因之一。



注意：JIT或即时编译器是 Java 虚拟机 (JVM) 的一部分。它同时转化具有相似功能的部分字节码。

为什么Java既是解释型语言又是编译型语言?

编程语言被分类为

- 高级语言，例如C++、Java
- 中级语言，例如C
- 低级语言，例如汇编
- 最后是最低级别的机器语言。

编译器是一个程序，它把程序从一个级别转化到另外一个级别（一般是从高到低），比如把C++程序转化到机器码。

java编译器将高级java代码转换成字节码（也是一种机器码）。

解释器是一个程序，它把程序转化为相同等级的其他语言，比如把Java程序转换成C++。

在 Java 中，Jit生成器将字节码转换为处于相同级别的本机机器代码，因此，Java 既是编译型语言，又是解释型语言。

为什么 Java 很慢?

Java 运行缓慢的两个主要原因是

1. **动态链接**：与 C 不同，链接是在运行时完成的，每次程序在 Java 中运行时。
2. **运行时解释器**：字节码到本地机器码的转换是在 Java 运行时完成的，这进一步减慢了速度。

但是，最新版本的 Java 在很大程度上解决了性能瓶颈。

总结：

- JVM 的完整形式是Java Virtual Machine.（Java虚拟机），Java 中的 JVM 是驱动 Java 代码的引擎，它将 Java 字节码转换为机器语言。
- Java 中的 JVM 架构包含类**加载器**、**内存区**、**执行引擎**等。
- 在 JVM 中，Java 代码被编译为字节码。这个字节码在不同的机器上被解释成不同的机器码。
- JIT 代表即时编译器。JIT 是 Java 虚拟机 (JVM) 的一部分。它用于加快执行时间。
- 与其他编译器机器相比，Java 中的 JVM 执行速度可能较慢。

- END -