第1讲 | 谈谈你对Java平台的理解？

2018-05-05 杨晓峰

第1讲 | 谈谈你对Java平台的理解？

杨晓峰

- 00:00 / 08:03

从你接触Java开发到现在，你对Java最直观的印象是什么呢？是它宣传的 “Write once, run anywhere”，还是目前看已经有些过于形式主义的语法呢？你对于Java平台到底了解到什么程度？请你先停下来总结思考一下。

今天我要问你的问题是，谈谈你对Java平台的理解？“Java是解释执行”，这句话正确吗？

典型回答

Java本身是一种面向对象的语言，最显著的特性有两个方面，一是所谓的“书写一次，到处运行”（Write once, run anywhere），能够非常容易地获得跨平台能力；另外就是垃圾收集（GC, Garbage Collection），Java通过垃圾收集器（Garbage Collector）回收分配内存，大部分情况下，程序员不需要自己操心内存的分配和回收。

我们日常会接触到JRE（Java Runtime Environment）或者JDK（Java Development Kit）。 JRE，也就是Java运行环境，包含了JVM和Java类库，以及一些模块等。
而JDK可以看作是JRE的一个超集，提供了更多工具，比如编译器、各种诊断工具等。

对于“Java是解释执行”这句话，这个说法不太准确。我们开发的Java的源代码，首先通过Javac编译成为字节码（bytecode），然后，在运行时，通过 Java虚拟机（JVM）内嵌的解释器将字节码转换成为最终的机器码。但是常见的JVM，比如我们大多数情况使用的Oracle JDK提供的Hotspot JVM，都提供了JIT（Just-In-Time）编译器，也就是通常所说的动态编译器，JIT能够在运行时将热点代码编译成机器码，这种情况下部分热点代码就属于编译执行，而不是解释执行了。

考点分析

其实这个问题，问得有点笼统。题目本身是非常开放的，往往考察的是多个方面，比如，基础知识理解是否很清楚；是否掌握Java平台主要模块和运行原理等。很多面试者会在这种问题上吃亏，稍微紧张了一下，不知道从何说起，就给出个很简略的回答。

对于这类笼统的问题，你需要尽量表现出自己的思维深入并系统化，Java知识理解得也比较全面，一定要避免让面试官觉得你是个“知其然不知其所以然”的人。毕竟明白基本组成和机制，是日常工作中进行问题诊断或者性能调优等很多事情的基础，相信没有招聘方会不喜欢“热爱学习和思考”的面试者。

即使感觉自己的回答不是非常完善，也不用担心。我个人觉得这种笼统的问题，有时候回答得稍微片面也很正常，大多数有经验的面试官，不会因为一道题就对面试者轻易地下结论。通常会尽量引导面试者，把他的真实水平展现出来，这种问题就是做个开场热身，面试官经常会根据你的回答扩展相关问题。

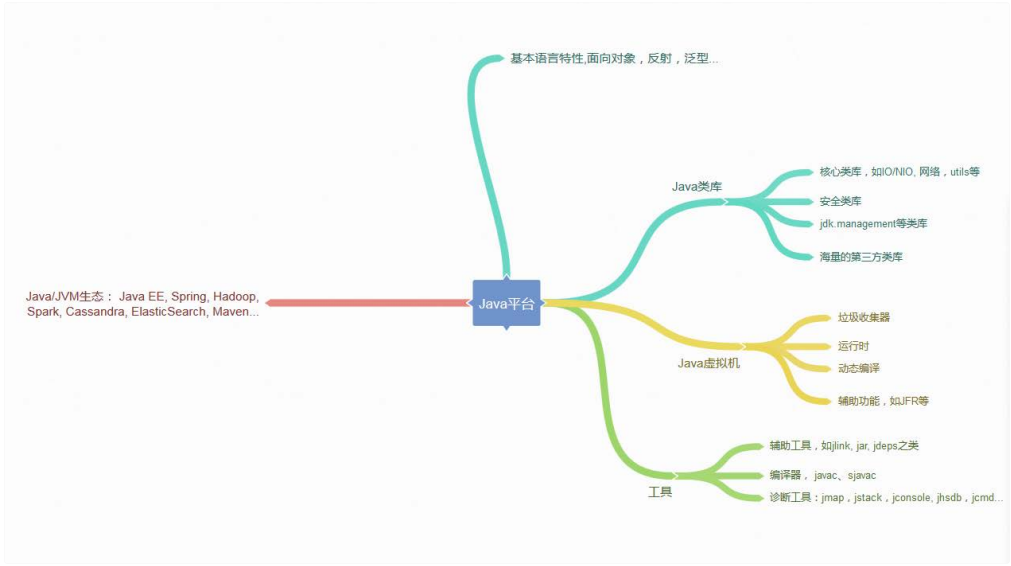
知识扩展

回归正题，对于Java平台的理解，可以从很多方面简明扼要地谈一下，例如：Java语言特性，包括泛型、Lambda等语言特性；基础类库，包括集合、IO/NIO、网络、并发、安全等基础类库。对于我们日常工作应用较多的类库，面试前可以系统化总结一下，有助于临场发挥。

或者谈谈JVM的一些基础概念和机制，比如Java的类加载机制，常用版本JDK（如JDK 8）内嵌的Class-Loader，例如Bootstrap、 Application和Extension Class-loader；类加载大致过程：加载、验证、链接、初始化（这里参考了周志明的《深入理解Java虚拟机》，非常棒的JVM上手书籍）；自定义Class-Loader等。还有垃圾收集的基本原理，最常见的垃圾收集器，如SerialGC、Parallel GC、 CMS、 G1等，对于适用于什么样的工作负载最好也心里有数。这些都是可以扩展开的领域，我会在后面的专栏对此进行更系统的介绍。

当然还有JDK包含哪些工具或者Java领域内其他工具等，如编译器、运行时环境、安全工具、诊断和监控工具等。这些基本工具是日常工作效率的保证，对于我们工作在其他语言平台上，同样有所帮助，很多都是触类旁通的。

下图是我总结的一个相对宽泛的蓝图供你参考。



不再扩展了，回到前面问到的解释执行和编译执行的问题。有些面试官喜欢在特定问题上“刨根问底儿”，因为这是进一步了解面试者对知识掌握程度的有效方法，我稍微深入探讨一下。

众所周知，我们通常把Java分为编译期和运行时。这里说的Java的编译和C/C++是有着不同的意义的，Javac的编译，编译Java源码生成“.class”文件里面实际是字节码，而不是可以直接执行的机器码。Java通过字节码和Java虚拟机（JVM）这种跨平台的抽象，屏蔽了操作系统和硬件的细节，这也是实现“一次编译，到处执行”的基础。

在运行时，JVM会通过类加载器（Class-Loader）加载字节码，解释或者编译执行。就像我前面提到的，主流Java版本中，如JDK 8实际是解释和编译混合的一种模式，即所谓的混合模式（-Xmixed）。通常运行在server模式的JVM，会进行上万次调用以收集足够的信息进行高效的编译，client模式这个门限是1500次。Oracle Hotspot JVM内置了两个不同的JIT compiler，C1对应前面说的client模式，适用于对于启动速度敏感的应用，比如普通Java桌面应用；C2对应server模式，它的优化是为长时间运行的服务器端应用设计的。默认是采用所谓的分层编译（TieredCompilation）。这里不再展开更多JIT的细节，没必要一下子就钻进去，我会在后面介绍分层编译的内容。

Java虚拟机启动时，可以指定不同的参数对运行模式进行选择。 比如，指定“-Xint”，就是告诉JVM只进行解释执行，不对代码进行编译，这种模式抛弃了JIT可能带来的性能优势。毕竟解释器（interpreter）是逐条读入，逐条解释运行的。与其相对应的，还有一个“-Xcomp”参数，这是告诉JVM关闭解释器，不要进行解释执行，或者叫作最大优化级别。那你可能会问这种模式是不是最高效啊？简单说，还真未必。“-Xcomp”会导致JVM启动变慢非常多，同时有些JIT编译器优化方式，比如分支预测，如果不进行profiling，往往并不能进行有效优化。

除了我们日常最常见的Java使用模式，其实还有一种新的编译方式，即所谓的AOT（Ahead-of-Time Compilation），直接将字节码编译成机器代码，这样就避免了JIT预热等各方面的开销，比如Oracle JDK 9就引入了实验性的AOT特性，并且增加了新的jaotc工具。利用下面的命令把某个类或者某个模块编译成为AOT库。

```
jaotc --output libHelloWorld.so HelloWorld.class
jaotc --output libjava.base.so --module java.base
```

然后，在启动时直接指定就可以了。

```
java -XX:AOTLibrary=./libHelloWorld.so,./libjava.base.so HelloWorld
```

而且，Oracle JDK支持分层编译和AOT协作使用，这两者并不是二选一的关系。如果你有兴趣，可以参考相关文档：http://openjdk.java.net/jeps/295。AOT也不仅仅是只有这一种方式，业界早就有第三方工具（如GCJ、Excelsior JET）提供相关功能。

另外，JVM作为一个强大的平台，不仅仅只有Java语言可以运行在JVM上，本质上合规的字节码都可以运行，Java语言自身也为此提供了便利，我们可以看到类似Clojure、Scala、Groovy、JRuby、Jython等大量JVM语言，活跃在不同的场景。

今天，我简单介绍了一下Java平台相关的一些内容，目的是提纲挈领地构建一个整体的印象，包括Java语言特性、 核心类库与常用第三方类库、Java虚拟机基本原理和相关工具，希望对你有所帮助。

一课一练

关于今天我们讨论的题目你做到心中有数了吗？知道不如做到，请你也在留言区写写自己对Java平台的理解。我会选出经过认真思考的留言，送给你一份学习鼓励金，欢迎你与我一起讨论。

你的朋友是不是也在准备面试呢？你可以“请朋友读”，把今天的题目分享给好友，或许你能帮到他。

Woj

2018-05-05

"一次编译、到处运行"说的是Java语言跨平台的特性，Java的跨平台特性与Java虚拟机的存在密不可分，可在不同的环境中运行。比如说Windows平台和Linux平台都有相应的JDK，安装好JDK后也就有了Java语言的运行环境。其实Java语言本身与其他的编程语言没有特别大的差异，并不是说Java语言可以跨平台，而是在不同的平台都有可以让Java语言运行的环境而已，所以才有了Java一次编译，到处运行这样的效果。
严格的讲，跨平台的语言不止Java一种，但Java是较为成熟的一种。"一次编译，到处运行"这种效果跟编译器有关。编程语言的处理需要编译器和解释器。Java虚拟机和DOS类似，相当于一个供程序运行的平台。
程序从源代码到运行的三个阶段：编码——编译——运行——调试。Java在编译阶段则体现了跨平台的特点。编译过程大概是这样的：首先是将Java源代码转化成.CLASS文件字节码，这是第一次编译。.class文件就是可以到处运行的文件。然后Java字节码会被转化为目标机器代码，这是是由JVM来执行的，即Java的第二次编译。
"到处运行"的关键和前提就是JVM。因为在第二次编译中JVM起着关键作用。在可以运行Java虚拟机的地方都内含着一个JVM操作系统。从而使JAVA提供了各种不同平台上的虚拟机制，因此实现了"到处运行"的效果。需要强调的一点是，java并不是编译机制，而是解释机制。Java字节码的设计充分考虑了JIT这一即时编译方式，可以将字节码直接转化成高性能的本地机器码，这同样是虚拟机的一个构成部分。

作者回复

2018-05-05

高手

magict4

2018-05-05

我对『Compile once, run anywhere』这个宣传语提出的历史背景非常感兴趣。这个宣传语似乎在暗示 C 语言有一个缺点：对于每一个不同的平台，源代码都要被编译一次。我不解的地方是，为什么这会是一个问题？不同的平台，可执行的机器码必然是不一样的。源代码自然需要依据不同的平台分别被编译。 我觉得真正问题不在编译这一块，而是在 C 语言源文件这一块。我没有 C 语言的编程经验，但是似乎 C 语言程序经常需要调用操作系统层面的 API。不同的操作系统，API 一般不同。为了支持多平台，C 语言程序的源文件需要根据不同平台修改多次。这应该是一个非常大的痛点。我回头查了一下当时的宣传语，原文是『Write once, run anywhere』，焦点似乎并不在编译上，而是在对源文件的修改上。

以上是自己一点不成熟的想法，还请大家指正！

作者回复

2018-05-05

汗颜，是我记错了，非常感谢指正

三军

2018-05-05

Java特性：
面向对象（封装，继承，多态）
平台无关性（JVM运行.class文件）
语言（泛型，Lambda）
类库（集合，并发，网络，IO/NIO）
JRE（Java运行环境，JVM，类库）
JDK（Java开发工具，包括JRE，javac，诊断工具）

Java是解析运行吗？
不正确！
1，Java源代码经过Javac编译成.class文件
2，.class文件经JVM解析或编译运行。
（1）解析：.class文件经过JVM内嵌的解析器解析执行。
（2）编译：存在JIT编译器（Just In Time Compile 即时编译器）把经常运行的代码作为"热点代码"编译与本地平台相关的机器码，并进行各种层次的优化。
（3）AOT编译器：Java 9提供的直接将所有代码编译成机器码执行。

作者回复

2018-05-05

精辟

thinkers

2018-05-05

jre为java提供了必要的运行时环境，jdk为java提供了必要的开发环境！

作者回复

2018-05-05

剧透一下，未来jre将退出历史舞台！

Jerry银银

2018-05-06

关注了好久，终于期盼到了第一讲。

在看到这个题目时，我并没有立马点进来看原文，而是给了自己一些时间进行思考。

首先，个人觉得这个题目非常的抽象和笼统，这个问题没有标准答案，但是有『好』答案，而答案的好坏，完全取决于面试者自身的技术素养和对Java系统性的了解。我的理解如下：

宏观角度：
跟c/c++最大的不同点在于，c/c++编程是面向操作系统的，需要开发者极大地关心不同操作系统之间的差异性；而Java平台通过虚拟机屏蔽了操作系统的底层细节，使得开发者无需过多地关心不同操作系统之间的差异性。
通过增加一个间接的中间层来进行"解耦"是计算机领域非常常用的一种"艺术手法"，虚拟机是这样，操作系统是这样，HTTP也是这样。

Java平台已经形成了一个生态系统，在这个生态系统中，有着诸多的研究领域和应用领域：
1. 虚拟机、编译技术的研究(例如：GC优化、JIT、AOT等)：对效率的追求是人类的另一个天性之一
2. Java语言本身的优化
3. 大数据处理
4. Java并发编程
5. 客户端开发（例如：Android平台）
6. ……

微观角度：
Java平台中有两大核心：
1. Java语言本身、JDK中所提供的核心类库和相关工具
2. Java虚拟机以及其他包含的GC

1. Java语言本身、JDK中所提供的核心类库和相关工具
从事Java平台的开发，掌握Java语言、核心类库以及相关工具是必须的，我觉得这是基础中的基础。
>> 对语言本身的了解，需要开发者非常熟悉语言的语法结构；而Java又是一种面对对象的语言，这又需要开发者深入了解面对对象的设计理念；
>> Java核心类库包含集合类、线程相关类、IO、NIO、J.U.C并发包等；
>> JDK提供的工具包含：基本的编译工具、虚拟机性能检测相关工具等。

2. Java虚拟机
Java语言具有跨平台的特性，也正是因为虚拟机的存在。Java源文件被编译成字节码，被虚拟机加载后执行。这里隐含的意思有两层：
1）大部分情况下，编程者只需要关心Java语言本身，而无需特意关心底层细节。包括对内存的分配和回收，也全权交给了GC。
2）对于虚拟机而言，只要是符合规范的字节码，它们都能被加载执行，当然，能正常运行的程序光满足这点是不行的，程序本身需要保证在运行时不出现异常。所以，Scala、Kotlin、Jython等语言也可以跑在虚拟机上。

围绕虚拟机的效率问题展开，将涉及到一些优化技术，例如：JIT、AOT。因为如果虚拟机加载字节码后，完全进行解释执行，这势必会影响执行效率。所以，对于这个运行环节，虚拟机会进行一些优化处理，例如JIT技术，会将某些运行特别频繁的代码编译成机器码。而AOT技术，是在运行前，通过工具直接将字节码转换为机器码。

作者回复

2018-05-06

**

zaiweiwoaini

2018-05-05

看评论也能学习知识。

作者回复

2018-05-05

搬个板凳，哈哈

欧阳田

2018-05-05

1，JVM的内存模型，堆、栈、方法区；字节码的跨平台性；对象在JVM中的强引用，弱引用，软引用，虚引用，是否可用finalise方法救救它？；双亲委派进行类加载，什么是双亲呢？双亲就是多亲，一份文档由我加载，然后你也加载，这份文档在JVM中是一样的吗？；多态思想是Java需要最核心的概念，也是面向对象的行为的一个最好诠释；理解方法重载与重写在内存中的执行流程，怎么定位到这个具体方法的。2，发展流程，JDK5(重写bug)，JDK6(商用最稳定版)，JDK7(switch的字符串支持)，JDK8(函数式编程)，一直在发展进化。3，理解祖先类Object，它的行为是怎样与现实生活连接起来的。4，理解23种设计模式，因为它是道与术的结合体。

作者回复

2018-05-05

高手

刻苦滴涛涛

2018-05-05

我理解的java程序执行步骤：
首先javac编译器将源代码编译成字节码。
然后jvm类加载器加载字节码文件，然后通过解释器逐行解释执行，这种方式的执行速度相对会比较慢。有些方法和代码块是高频率调用的，也就是所谓的热点代码，所以引进jit技术，提前将这类字节码直接编译成本地机器码。这样类似于缓存技术，运行时再遇到这类代码直接可以执行，而不是先解释后执行。

作者回复

2018-05-05

不错，JIT是运行时编译

姜亮

2018-05-07

写个程序直接执行字节码就是解释执行。写个程序运行时把字节码动态翻译成机器码就是jit。写个程序把java源代码直接翻译为机器码就是aot。造个CPU直接执行字节码，字节码就是机器码。

作者回复

2018-05-07

好主意，当年确实有类似项目

曹铮

2018-05-05

这种基于运行分析，进行热点代码编译的设计，是因为绝大多数的程序都表现为"小部分的热点耗费了大多数的资源"吧。只有这样才能做到，在某些场景下，一个需要跑在运行时上的语言，可以比直接编译成机器码的语言更"快"

作者回复

2018-05-05

对，看到本质了

一叶追寻

2018-05-05

对Java平台的理解，首先想到的是Java的一些特性，比如平台无关性、面向对象、GC机制等，然后会在这几个方面去回答。平台无关性依赖于JVM，将.class文件解释为适用于操作系统的机器码。面向对象则会从封装、继承、多态这些特性去解释，具体内容就不在评论里赘述了。另外Java的内存回收机制，则涉及到Java的内存结构，堆、栈、方法区等，然后围绕什么样的对象可以回收以及回收的执行。以上是我对本道题的理解，不足之处还请杨老师指出，希望通过这次学习能把Java系统的总结一下~

作者回复

2018-05-05

非常棒，不同语言对平台无关的支持是不同的，Java是最高等级，未来也许会在效率角度出发，进行某种折衷，比如AOT

scott

2018-05-07

解释执行和编译执行有何区别

作者回复

2018-05-07

类比一下，一个是同声传译，一个是放录音

石头狮子

2018-05-05

1. 一次编译，到处运行。jvm 层面封装了系统API，提供不同系统一致的调用行为。减少了为适配不同操作系统，不同架构的带来的工作量。
2. 垃圾回收，降低了开发过程中需要注意内存回收的难度。降低内存泄露出现的概率。虽然也带来了一些额外开销，但是足以弥补带来的好处。合理的分代策略，提高了内存使用率。
3. jit 与其他编译语言相比，降低了编译时间，因为大部分代码是运行时编译，避免了冷代码在编译时也参与编译的问题。
提高了代码的执行效率，之前项目中使用过 lua 进行相关开发。由于 lua 是解释性语言，并配合使用了 lua-jit。开发过程中遇到，如果编写的 lua 代码是 jit 所不支持的会导致代码性能与可编

译的相比十分低下。

作者回复

2018-05-05

高手

公号-Java大后端

2018-05-05

今日文章心得:个人理解的Java平台技术体系包括了以下几个重要组成部分：
Java程序设计语言
各种硬件平台上的Java虚拟机
Class文件格式
Java API类库及相关工具
来自商业机构和开源社区第三方Java类库

可以把Java程序设计语言、Java虚拟机、Java API类库及相关工具，这三部分统称为JDK，JDK是用于支持Java程序开发的最小环境；可以把Java API类库中的Java SE API子集和Java虚拟机这两部分统称为JRE，JRE是支持Java程序运行的标准环境。

提起Java，必然会想起TA跨平台的特性，但是跨平台重要吗？重要！因为可以write once，run anywhere，这是程序员的终极梦想之一。但是跨平台重要吗？不重要！作为程序语言，会更加关注TA的生态、兼容性、安全性、稳定性，以及语言自身的与时俱进。要要理解Java平台，JVM是必须要迈过去的坎，将会看到另外的风景。

为什么我们就不能把JVM作为透明的存在呢？

勿在浮沙筑高台，以JVM的GC为例。既然Java等诸多高级程序语言都已经实现了自动化内存管理，那我们为什么还要去理解内存管理了？因为当我们需要排查各种内存溢出、泄漏等底层问题时，当垃圾收集成为我们开发的软件系统达到更高并发量、更高性能的瓶颈时，我们就需要对这些“自动化”技术实施必要的监控与调节优化。

作者回复

2018-05-06

对，深入有利于解决更多有难度的工作

非常非常非常非常的普通中下

2018-05-08

没有一个问题是加一个中间层解决不了的，如果解决不了就加两个

樱の空

2018-05-14

以下是我在本节课所得到的收获，结合TIJ的内容整理了一下我个人的理解，若有错误，还望老师指出。
Java采用的是解释和编译混合的模式。它首先通过javac将源码编译成字节码文件class.然后在运行的时候通过解释器或者JIT将字节码转换成最终的机器码。
只是用解释器的缺点：抛弃了JIT可能带来的性能优势。如果代码没有被JIT编译的话，再次运行时需要重复解析。
只用JIT的缺点：
需要将全部的代码编译成本地机器码。要花更多的时间，JVM启动会变慢非常多；
增加可执行代码的长度（字节码比JIT编译后的机器码小很多），这将导致页面调度，从而降低程序的速度。
有些JIT编译器的优化方式，比如分支预测，如果不进行profiling，往往并不能进行有效优化。
因此，HotSpot采用了惰性评估(Lazy Evaluation)的做法，根据二八定律，消耗大部分系统资源的只有那一小部分的代码（热点代码），而这也就是JIT所需要编译的部分。JVM会根据代码每次被执行的情况收集信息并相应地做出一些优化，因此执行的次数越多，它的速度就越快。
JDK 9引入了一种新的编译模式AOT(Ahead of Time Compilation)，它是直接将字节码编译成机器码，这样就避免了JIT预热等各方面的开销。JDK支持分层编译和AOT协作使用。
注：JIT为方法级，它会缓存编译过的字节码在CodeCache中，而不需要被重复解释

作者回复

2018-05-14

不错，严格说我说的是oracle jdk和hotspot jvm的行为

吴有为

2018-05-06

老师，您好，看了文章和大家的评论有点疑问，程序执行的时候，类加载器先把class文件加载到内存中，一般情况下是解释执行，解释器把class里的内容一行行解释为机器语言然后运行。疑问1.每次执行class文件都需要解释整个class文件吗？疑问2.当new了一个对象的时候是怎么解释这个类的，是解释整个这个类对应的class？疑问3.JIT编译的热点代码是指class文件还是class文件的部分内容？

作者回复

2018-05-06

我的理解不是以class为单位；JIT是方法级

Alex

2018-05-11

评论区都是精华啊

凌

2018-05-08

国富论中讲到，社会的分工细化起到了提高生产力的关键作用。我觉得一次编写到处运行也是社会分工的一种模式，他使大部分业务程序员注重领域模型的逻辑设计，不必关心底层的实现，使软件工程达到了专业的人做专业的事这一个高度。虽然现在掌握一门技术远远不够，但是对于大部分业务程序员来说，只有把精力花在最重要的地方比如领域模型的设计，才会让业务更加流畅完善。所以我觉得JVM机制蕴含了一定的经济学原理。

张立春

2018-05-07

任何软件问题都可以通过加一层来解决：有了系统API就屏蔽了不同硬件的区别，有了编译器就屏蔽了不同机器语言的区别，有了JVM就屏蔽了不同操作系统的区别，有了TCP/IP就屏蔽了不同系统之间通讯的差异，有了语音识别和翻译就屏蔽了不同语言的差异。也许有一天人工智能可以直接把自然语言翻译成机器码直接生产可用的软件。

jack

2018-05-06

Java平台包括java语言，class文件结构，jvm，api类库，第三方库，各种编译、监控和诊断工具等。
Java语言是一种面向对象的高级语言，通过平台中立的class文件格式和屏蔽底层硬件差异的jvm实现‘一次编写，到处运行’；
通过‘垃圾收集器’管理内存的分配和回收。
jvm通过使用class文件这种中间表示和具体语言解耦，使得任何在源码早期编译过程中以class文件为中间表示或者
能够转换成class文件的具体语言，都能运行jvm之上，也就可以使用jvm的各种特性。
api类库主要包含集合、IO/NIO、网络、并发等。
第三方库包括各种商业机构和开源社区的java库，如spring、mybatis等。
各种工具如javac、jconsole、jmap、jstack等。

作者回复

2018-05-06

到位

佳人如玉巧弄心弦

2018-05-05

关于JIT与AOT，我想KVM更有发言权，哈哈，好的东西终被学习与借鉴

作者回复

2018-05-05

行家

墨川

2018-05-06

老师讲的很精彩受教了，评论区好多高手。赶紧拿个小本本记下来

祥子.Ken

2018-05-06

看课程和评论涨知识了，不在此献丑。
分享个人的听课方式，课程已标明了后续的课程名称(问题)，我已先自我测试回答后续的问题，再听后续课程，会印象更深刻一点。

迎xiang李

2018-05-05

Java新手表示学习了，java的运行机制算是看明白了，但是发现还是有很多词汇不太了解。只有看高手们的文章才能发现自己的短板和不熟悉的领悟。期待后面更精彩，全面深入的讲解。感谢作者和各位大神的精彩评论！

作者回复

2018-05-06

交流有利于提高

半日闲

2018-05-11

编译型语言：C/C++、 Pascal（Delphi）
编译就是把源代码（高级语言，人类容易读，容易理解）转换成机器码（CPU能理解，能高效的执行）

解释型语言：JavaScript、Perl、Python、Ruby
解释就简单多了，解析源代码，并且直接执行，没有编译过程

编译程序是整体编译完了，再一次性执行。 而解释程序是一边解释，一边执行

JAVA语言是一种编译型-解释型语言，同时具备编译特性和解释特性
其所谓的编译过程只是将.java文件编程成平台无关的字节码.class文件，
并不是向C一样编译成可执行的机器语言，在此请读者注意Java中所谓的“编译”和传统的“编译”的区别。

作为编译型语言，JAVA程序要被统一编译成字节码文件——文件后缀是class。此种文件在java中又称为类文件。
java类文件不能再计算机上直接执行，它需要被java虚拟机翻译成本地的机器码后才能执行，而java虚拟机的翻译过程则是解释性的。
java字节码文件首先被加载到计算机内存中，然后读出一条指令，翻译一条指令，执行一条指令，该过程被称为java语言的解释执行，是由java虚拟机完成的。
以上说的是Java的解释执行，但是比如我们大多数情况使用的Hotspot JVM，都提供了动态编译器编译器JIT，能够追踪热点代码， 然后变成机器指令，这种情况下部分热点代码就属于编译执行，而不是解释执行了

其实甭管它什么解释还是编译， 了解了底层的原理就行了

老猫

2018-05-05

还在学习Java ，给不出啥评论，但是看文章和评论，学到不少东西

---

playpi

2018-06-25

内容看了十分钟，评论看了半小时。

大熊

2018-05-26

首先javac把源码编译成字节码(.class文件)，然后在程序运行时JVM把需要的.class文件加载内存，解释器逐行把该文件解释成机器码，在同样程序运行过程中部分热点代码可以通过JIT编译成机器码(不是逐行)并存在缓存里(运行时编译保证了可移植性)，下次运行可以直接从缓存里取机器码，效率更高。
AOT跟JIT的区别还是不太理解，原文的话是 ~ AOT直接将字节码编译成机器代码，这样就避免了 JIT 预热等各方面的开销?
其他有不对的地方，麻烦指正

作者回复

2018-05-28

jit是运行时才做的，需要预热才知道哪些是热点；
aot是编译期，静态的，直接编成类似类库的东西

有铭

2018-05-16

我一直没想明白一个问题，既然jdk是jre的超集，为啥甲骨文提供的jdk安装包要同时安装他们两个。只装jdk不就行了吗，而且这一现象在所有平台均存在

作者回复

2018-05-18

未来，可以看看9以后的版本，已经去掉jdk里面的jre，长远看单独发布一个jre的需要变小了，当然我们程序总还要用，使用jdk，或者用jlink订制runtime，也许是更好选择

Jeffrey

2018-05-07

更新的好慢啊，大神快点

作者回复

2018-05-07

这个节奏不是我定，抱歉

张驰

2018-05-06

所有的类运行时解析之后，再次运行时还会重复解析吗?

作者回复

2018-05-06

简单说，如果没有被jit编译，是的

---

mongo

2018-05-05

首先什么是解释执行？什么是编译执行？我采纳了这位知乎大神的说法 https://www.zhihu.com/question/21486706/answer/18642540 。对于编译执行？是不是动态代理的实现原理字节码重组后的目标代理类的执行就是属于编译执行？包括反射的实现也是属于编译执行？这个猜想怎么验证？����

作者回复

2018-05-05

简单，jvm启动时加上 -XX:PrintCompilation,就能把相关信息打出来

Kevin Wang

2018-05-05

Java是一个不完全的面向对象编程语言，它基于JVM，经常被人吐槽的有两点：一是慢（其实也不算慢了）；二是代码冗长。同样基于JVM的还有Scala语言，它综合了面向对象与函数式编程，代码非常精简，而且可以无缝使用Java现有的库。

作者回复

2018-05-05

启动时间现在有了AOT和Appcds，再加上jigsaw提供的模块化能力，提高非常大，也许我后面补充一个这方面的文章

过往云烟

2018-05-05

java是一次编译，各个平台都会运行的。java执行的时候并不只是解释执行，有的时候会把部分代码编译成机器码。所以java是混合编译的。 到了jaca9就会出来aot，直接把某个类库编译成二进制文件，这样加快运行。

---

ls

2018-05-12

之前看到过一个理解：编译执行就好像做了一桌子的菜，坐到餐桌就可以吃了，而解释执行就像吃火锅，需要一边等，一边吃，所以效率会慢

对Java理解不好，有个疑问：为什么不能一开始把java源码翻译成各个平台的机器码？这样到JVM层操作时，再选择JVM所在的平台运行其机器码，这样效率是否会高点。

星火

2018-05-11

评论区有点强势…

---

big

2018-05-11

我连评论也不敢落下

李军

2018-05-11

狭义上讲，JVM不是跨平台的，相当于对不同的操作系统做了适配?

昆少

2018-05-10

天啊，评论里满满都是干货。我太想进步了！

成功

2018-05-10

看了几个留言，发现点问题。首先C/C十十目前也是支持动态编译，只不过是编译成DLL库。如果不在集成环境下编译Java程序，你会发现Javac编译出的。class文件类似。O文件。

倒影

2018-05-09

Java是一种面向对象的语言，它最大的特点是跨平台性，一次运行，多处执行，javac编译器把Java代码编译成.class字节码文件，在运作的时候通过jvm加载字节码文件进行解析针对不同的操作系统编译执行，Java通过javm屏蔽不同操作系统之间的差异，让开开发者不需要考虑操作系统的不同，只关注于代码，将问题统一解决，这是开发过程中常见的一种统一解决问题的方式，比如java中的GC（垃圾回收机制），自动回收，在开发的过程中不需要考虑内存回收的问题，Ja再比如我们写一个方法，处理某几种类型的问题，只需要根据传入的参数不同来做不同的处理，从而达到一个方法多处可用。
Java是一种面向对象的语言，具有三大特性
1、封装
2、继承
3、多态

Mr．Bean

2018-05-09

看文章 很充实 知识很全面 但有一点点晦涩 结果看了评论发现好多大神 心里有一点疑惑慢慢就解决了 继续努力学习

肖智坤

2018-05-07

1.开发
按照java平台的语法规范，结合平台提供的三方类库和框架开发程序，
2.编译
开发完成后使用平台提供的javac工具将程序编译成.class文件，
3.加载
平台通过classloder加载编译成功的.class文件到JVM中
4.运行
在运行时，JVM使用JIT将.class文件中的二进制字节码解释或编译（根据使用频率来判断使用解释还是编译）成计算机底层能够读懂的机器代码去执行
5.回收
运行期间，通过JVM的垃圾回收，根据参数使用不同的回收方法，将不再使用的内存地址中的数据清空
6、监控，调优
在JVM运行时我们还可以通过平台提供的如：jmap，jconsole等进行监控诊断工具，程序的性能进行监控诊断，进行程序的诊断和调优工作

通过这次课的学习，不知道这样理解java平台是否正确

珅珅君

2018-05-07

评论干货满满！满足！

萧萧

2018-05-05

"JIT能在运行时刻将字节码编译成机器码， 这样热点代码就是编译运行了"， 这个说法有点让人困惑， JVM解释字节码的过程不就是逐条转换为对应的机器码予以执行吗？运行时刻的编译不就是节解释运行吗？这里是说热点代码会被整块编译好了运行， 无需逐条解释， 提高效率吗

作者回复

2018-05-06

jit会缓存编译过的在codecache里

小六丶

2018-05-05

菜鸟一个，看评论就可以学到好多知识！写了挺多年业务代码，看样子是白干了！

作者回复

2018-05-06

怎么会，术业有专攻，换个角度总结一下，对业务开发也有帮助

George

2018-05-05

Java 平台不会强制您时刻关注内存分配（或使用第三方库来完成此工作），它提供了开箱即用的内存管理功能。当您的 Java 应用程序在运行时创建一个对象实例时，JVM 会自动从堆 中为该对象分配内存空间— 堆是一个专门留给您的程序使用的内存池。Java 垃圾收集器 在后台运行，跟踪记录应用程序不再需要哪些对象并从它们回收内存。这种内存处理方法称为隐式内存管理，因为它不需要您编写任何内存处理代码。

作者回复

2018-05-05

所以Java程序员适合普罗大众写出质量可靠的应用

晴天

2018-05-05

问题是java平台，所以我可能会更多的回答是jvm部分，java是解释执行的话我觉得是的，目前并没有用到jdk8。唉，虽然了解一点，但是没用过、就知道点新概念和特性

---

lijun

2018-05-05

我对java平台的理解还是很肤浅，惭愧啊。

作者回复

2018-05-05

术业有专攻，加油��

dingwood

2018-07-18

老师，请教个有点弱智的问题。jre和jdk区别在哪？原来的理解是jdk为开发环境。环境变量的设置路径全在jdk的相关jar包。jre为运行环境，生产上运行需要jre。但实际开发当中，生产环境一般都重新安装jdk，而不是拷贝jre。 jre基本对我这样的开发者绝缘。 另外，既然两个环境都能跑JAVA程序，为什么安装环境的时候要弄两套？开发环境用jdk，生产环境也用jdk不就行了吗？ 据您说，jre要退出了？缘何？

作者回复

2018-07-18

jre缺少一些开发、诊断相关的工具、类库；
单独提供jre部分原因是因为applet webstart之类功能的需要，现在逐步剔除了

Andy

2018-07-17

看了这么多评论，还是没有明白解释执行和编译执行是什么意思，能不能把大道理说的简单点，举个具体的例子啥的，虽然我才学习Java 2个月，但是如果能让学习2个月人都能明白，说明做到了深入浅出不是吗？

pengshao

2018-07-17

因为一直都是开发小项目，都是开发完，打包放在服务器上运行程序，这算是编译后执行吧？热更算是JIT吗？基本上项目需要更新都是停服，重新打包后运行，这样的话，是不是就没有利用到JIT这个技术？

作者回复

2018-07-18

不是一回事儿，热部署之类不阻挠JIT起作用

张小伟

2018-07-13

评论区都是大神。

superpig

2018-07-02

C/C++程序员转型中。我理解JVM产生的用意，其实是早年间C的跨平台软件，只能靠预编译宏来实现WORA，那真是地狱一样的代码。
此外，C的编译器更不存在跨平台的概念。Java加了一层字节码，Javac这个编译器只把代码转换成字节码，这样在代码和编译器这一层，其实就是做了一次归一。
Java的跨平台能力，主要来自于JVM，如果一个芯片平台没有JVM支持，那么javac产生的字节码肯定就无法在这个平台上运行了。
JVM的思想是屏蔽掉底层硬件的差异，为开发者提供一个纯软件的平台，这样不仅简化程序运行，更简化了编程开发的过程。基于这个思想，微软也推出了自己的跨平台虚拟机.net framework，只不过近年开始才真正的跨到Linux上。
既然JVM执行的是字节码，那么只要能编译出字节码，就无所谓开发程序所用的高级语言，scala也在JVM上跑就是这个道理。对应的，微软的sharp系列语言也是这个道理。
这样看来，对上，JVM可以兼容多种高级语言，只要有编译器将这种高级语言编译成字节码就可以；对下，JVM可以兼容多个平台，只要JVM对这个平台进行过适配就可以让程序跑起来。
这个思想其实就是现代软件工程里的模块化、层次化的思想，也让我想起了TCP/IP协议族里的层次模型，Everything over IP, IP over everything. 在Java的世界里，JVM就是IP了。
但是JVM也因此成为了瓶颈，执行效率、CPU内存IO管理、进程线程等等，还有C中完全不存在的垃圾回收机制，这些事都需要JVM去做，这也是C程序员吐槽Java效率低的原因之一。不过这样也带来一个好处，就是无论我们需要定位问题、效率优化或是做什么，基本上盯着JVM就可以了。
因此要写出高效的Java代码，JVM肯定是绕不过的话题。

Winston

2018-07-01

本人将java平台认知分为基础和扩展两个部分（类似于se和ee）。

在基础部分，主要基于java语言本身，java是一门面向对象的语言，具备良好的可移植性(Write once, run anywhere.)，其提供了一系列基础的类库(io/nio/net/util/lang/math等)，能满足多线程并发（java只提供多线程）、文件读写、网络通信等多种任务需求。同时，java运行于jvm之上，jvm提供了自动的垃圾收集、类加载等功能，一方面提高了java语言的跨平台特性，另一方面，减少使用者申请、释放内存的操作，更易操作。

在扩展方面，基于java语言，开发出许多框架与组件，应用广泛。其中Spring的IOC/AOP特性，ORM框架Mybatis，能够解决网站开发，数据库交互等多方面业务需求，而一些中间件，例如：消息中间件RocketMQ，可以应用于系统模块通信，实现模块解耦。

从中能看出java语言的广泛应用与强大的潜力。

紫豪

2018-06-30

看文章是一种学习，看评论更是一种学习。
每个人分析看待问题的立场不同，得出的感悟也都有独特的语言表达方式。
很庆幸能够与大家一起学习，接触的越多，才发现自己这个工作"多年"的老开发基础是如此的薄弱，希望所有人都能通过杨老师的课程得到一些启发，感谢各位的分享。

feifei

2018-06-29

我的理解即正确，又不正确，这个说法不完全错
Java的运行过程，java源文件--》class字节码--》由jvm虚拟机来执行字节码
执行分为2种解释与编译，在早起的jvm中是解释执行的，后来演变成编译与解释一起运行
java之所以跨平台并不是因为java语言，是因为jvm虚拟机，在不同的平台上，比如windows与linux环境，都有配套的jvm虚拟机，在不同的平台上安装虚拟机，由虚拟机来执行中间代码即字节码，就实现了跨平台

未来大神

2018-06-29

请问JIT为什么不把所有的字节码解释成机器码缓存起来，而要profiling，缓存少部分呢？JVM内存占用不是问题吧

作者回复

2018-06-30

我理解这是问题，codecache大小是有限的，编译本身也会有开销

winner_0715

2018-06-26

判断是不是热点代码是由JVM自己判断的吗？实际项目中怎么用JIT呢

wenxueliu

2018-06-22

开篇为 "解释hello world执行流程" 就可以将几乎所有知识点串起来了。
希望能解释下jvm和go的垃圾回收的区别，看到go已经在垃圾回收方面超过jvm了。

作者回复

2018-06-26

谢谢建议；
结论可能与实际不符，go gc据我了解可以看做cms的改进版，未必如宣称那么好，要看疗效

李太阳

2018-06-21

工作一年多了，听老师一席话，感觉自己的知识真的是太片面了

huffmanT

2018-06-17

JAVA诞生，恰逢internet野蛮生长之时，JAVA的对网络编程支持和易用易理解促使了互联网公司的兴盛，随着时间推移更多更优的语言的涌现了出来，如Python,Go等，Java的强大是因为它的生态比较完善各种开箱即用的框架使的Java成为了商业业务应用的一哥，随着Oracle的收编，对商业公司闭源和知识产权的忌惮(参考Oracle和谷歌关于Android的官司），会进一步促进了OpenJdk和其它开源语言的发展，对于Coder来说建议多学几门语言，想提升对Java的理解，跟着杨老师就对了。

作者回复

2018-06-19

谢谢，如果持续关注了openjdk，你会发现是在逐步更加开放的；关于你说的官司，建议去看看有点深度、更多事实的文章

Geek_59a17f

2018-06-16

Javac 的编译，编译 Java 源码生成".class"文件里面实际是字节码，而不是可以直接执行的机器码。Java 通过字节码和 Java 虚拟机（JVM）这种跨平台的抽象，屏蔽了操作系统和硬件的细节，这也是实现"一次编译，到处执行"的基础。

在运行时，JVM 会通过类加载器（Class-Loader）加载字节码，解释或者编译执行

待时而发

2018-06-16

看评论也能学习知识，大牛太多了

robbin��

2018-06-12

aot的这种模式我们一般在什么场景来使用，有什么标准规范这类可以参考吗？或者老师可以帮忙讲解下吗？

作者回复

2018-06-12

当你的应用对启动时间非常敏感时；可以看Java9以后的jdk文档

风起

2018-06-10

我的理解java平台就是帮助开发者做了很多事情， 从而减少开发者的重复工作。
1.集合，io，nio，线程池等这些让我们使用，但是像我这种新手就需要多多学习
2.java虚拟机这块， 帮助我们做了垃圾收集，因此就要学习里面的垃圾收集策略， 内存模型什么的
3.有开源框架已还有轮子的支持，比如常用的spring mybatis guava这些

改名不换人

2018-06-07

server模式下，JVM会进行上万次调用收集信息已便进行高效的编译，将其中的部门热点代码进行运行时编译，请问这上万次调用是JVM主动发生的，还是应用程序在使用过程中，JVM被动收集的呢?

作者回复

2018-06-07

我理解是被动

吴传卜

2018-06-06

果然评论区和原文一样精彩

Geek_e848d4

2018-06-05

麻烦写英文的同时把中文也写上，不然看不懂啊��

乖乖

2018-06-03

评论区 从来就不缺 神

刘辉

2018-06-02

请问jit运行将 热点代码 编译成机器码。什么是热点代码呢？还是jit会有预热开销。jit预热那些东西呢?

作者回复

2018-06-03

简单说热点就是调用比较频繁的代码，预热就是收集信息找出热点

红茶君

2018-05-30

刚刚开始学JAVA，只看懂了个大概意思，有好多名词待查.....评论区好厉害，涨姿势了。

萨拉赫

2018-05-29

这篇是关于Oracle计划放弃反序列化的文章https://www.infoworld.com/article/3275924/java/oracle-plans-to-dump-risky-java-serialization.html?from=timeline&isappinstalled=0

作者回复

2018-05-30

嗯，长久看是，Mark R.也说了没有时间表drop这个功能；后续估计会有其他改进，我不便透露或承诺什么

萨拉赫

2018-05-29

能不能讲讲java序列化/反序列化的问题。以及java很多不安全的问题都与反序列化有关，近期又看到一篇文章说java正打算放弃这个功能。希望您能权威的介绍下，谢谢。

作者回复

2018-05-29

序列化贡献了非常大比例的安全漏洞;
我得回头查查，不见得是放弃，有取舍，这几个版本一直在为他添加filter

密码123456

2018-05-25

1.jre与jdk的的关系。jre是java运行环境，有jvm和类库。而jdk是jre.扩展版本，不仅仅有jre的功能还提供一些工具，比如编译器。
2.jit是动态编译部署的工具

大月

2018-05-24

感觉自己理解的太片面了，还需要不断学习，为找工作打好基础。对于大神所说的都不太理解，我去面壁了，打卡1次

Jerwei

2018-05-24

大神，我有个小小的疑问。Java代码编译为字节码文件后，jvm得解析器对字节码文件解析执行，是解析成了机器码吗？如果是的话，这个和jit直接编译成机器码有啥区别，因为都从字节码变成了机器码。jit在运行时会对热点代码进行编译，怎么指定某些代码是热点代码呢，编译热点代码的过程是提前做的吗？

---

c@ini@o

2018-05-20

老师讲的很精彩，让人豁然开朗！能否指点一下是否有好的代码例子，来快速理解提升那？

日光倾城

2018-05-18

谈谈你对java平台的理解？java的底层，包括jvm架构，有类加载子系统，运行时数据区：java堆 栈 计数器 方法区 本地方法栈，以及执行引擎：JITCompiler GC 还有本地方法库部分。java核心类库，从.java文件到最后运行出结果的整个过程的理解。上层的话java生态圈，包括各种框架，第三方类库等。

---

aka. peng ��

2018-05-18

Android虚拟机由最早的Dalvik VM变成了ART，关键变化就是从JIT变成了AOT。 jvm一直不变成AOT编译的理由是什么呢？ 运行时编译的好处是不是就是编译速度快点。 对于client端来说，绝对倾向AOT呀。

作者回复

2018-05-18

说的有道理，但主流是用oracle jdk做server应用

---

njzy_sbl51

2018-05-17

老师，针对“从你接触Java开发到现在，你对Java最直观的印象是什么”该问题，我的答案是：易理解，因其是一门面向对象的编程语言，较清晰地表述了客观世界中的对象；简洁方便，因其是一门较规范的编程语言。老师，您好，我想请问一下，若您是面试者，我是求职者，这样的回复，老师您会做何反应？同时，还望老师就我的回复，给一些相关建议，谢谢老师。

作者回复

2018-05-18

挺好，不过我的角度和应用开发有区别，可能问“简洁方便”之类，具体体现在什么地方，“规范”是什么意思？类型安全？那和“简洁”可能有冲突吗

Yang

2018-05-17

高手在民间，看评论也有很多收获

---

zero

2018-05-15

有个疑问，什么时候用的是解释执行，什么时候用的是编译执行。类加载的时候算是解释执行，方法调用时是编译执行吗？

杨振效

2018-05-14

评论区里，依稀见到众多大神的背影

樱の空

2018-05-14

谢谢老师指点，是我疏忽了。因为平时都是使用的oracle jdk 和 hotspot 就没有加上限制。佩服老师严谨的态度哈，我还需要多学习学习

邱新海

2018-05-14

感觉语速再慢一些就更好了。

卡斯瓦德

2018-05-14

老师你好：
1.对于aot还没接触，因为不是java9，以及最新的java10都不是长期支持的对于jdk的版本选择你有什么建议么？
2.关于刚才看到说.class文件每次new一个对象时都需要重新解释下？哪怕应用没有重启也要重新解释下是么？那么反问下如果是jit的那么还会每次new对象的时候去编译么？

作者回复

2018-05-14

1，11会是长期支持版本；
2，最好有真正做jvm的人回答，虽然我认为是，但没人保证每个细节都清楚

十年后的思念

2018-05-13

杨老师，扩展的知识多来点

作者回复

2018-05-14

��

今天也在为演唱会门票努力着

2018-05-12

老师 那AOT编译是直接从源码编译为了机器码吗？ 字节码不生成了吗 这是否会影响java通过解释+编译字节码文件达到在不同运行环境“一次编译 处处运行”的特点？

作者回复

2018-05-12

是的，有代价

飞鸿雪浪

2018-05-12

jre最终取消，是不是和golang一样，将运行时和代码一同编译成目标机器码，来实现跨平台？

作者回复

2018-05-12

我可能没说清楚，有了jigsaw jlink，自己可以定制最合适的runtime，另外jre最初存在部分原因是类似web start，applet的需要，现在也基本进入终场了

2768zxh

2018-05-12

看了诸位大神的评论，我一个应届毕业生瞬间感觉自己的java知识面是那么的窄，希望能通过交流得到提升。

我们俩

2018-05-10

从本文中得到的信息可以分为两点，java可以一次编译，到处执行，也就是跨平台能力。第二点就是java有垃圾回收机制，不需要人工操作回收垃圾，java的jvm可以自动执行回收

Andy

2018-05-10

还有作业批改，太贴心了，十分感谢！

呆呆很爱你

2018-05-10

可否有JavaGC的详解？这个问题一直了解的很粗

Mason

2018-05-10

小白，读专栏有些吃力，能否推进一本Java入门书籍啊？

作者回复

2018-05-11

以前入门是Thinking in Java， core Java

林长健@Damon

2018-05-09

我想另一个角度来回答一下问题

Java平台的宗旨是一次编写跨平台运行，其实这也是所有程序员的梦想了，如果有一门编程语言能实现计算机上的任何功能，谁不乐意呢？

然而Java 在发展的过程中，初期因为硬件性能瓶颈，Java 的表现性能上比不上c语言类的性能而被诟病，很多场景用Java 是无法忍受的，但是随着摩尔定律的存在，而且Java 这类存面相对象的语言又是软件合作非常需要的特性，Java 这类语言的可读性是很高的，很大程度具备了高协作特性，记得软件产业有一次危机就是因为可读性的问题，这说明代码可读性的重要性。

同时Java 也在不停地进步，Android 开发也基于Java ，看到别的同学留言说到即时编译，运行时编译肯定是有一些性能影响的，不然也不会有art 虚拟机了。

可以看的出来Java 虚拟机类别很多，但它们都遵循统一Java 规范来，让开发者始终能用同一语言来进行开发。

Java 平台终极思想也是一种容器思维，容器即平台，分层的思维在软件行业真心巴不得弄成千层饼，挺有意思的，也正因为层次多，分工也就越来越多，分工多了才会有今天真的庞大的互联网协作平台，才有了今天的互联网。

作者回复

2018-05-10

不错的总结，不过你提到的那个可以去查查，它不遵守规范的，嘿嘿

吕啡品

2018-05-09

山外有山，天外有天，能找到一扇窗是最好不过的事，期待之后的深入讲解，让我们这些泡在业务中的人，能跳出来看看，快哉快哉！

不瘦十斤不换名字

2018-05-09

上面一个朋友说Java内存模型是 堆 栈 方法区，说法有误区吧，堆 栈 方法区应该是内存结构，JMM是 并发里面的 主存 cpu缓存那块的东西

作者回复

2018-05-09

是的，可能说的时候上下文不同

QuincySx

2018-05-09

jre 为什么要退出了，我有注意到 JDK 10 jre 要单独下载了

Andy

2018-05-08

1.JDK与JRE区别：JRE包括jvm虚拟机，工具包，支持模块。JDK是JRE的超集，多了编译工具。
2.java一次编译到处运行。通常情况下，java编译成为字节码通过虚拟机来进行解释执行。但java也可以通过命令把若干个包或部分程序编译执行。如oracle hotspot中的jin特性可以把java中的包编译执行。也可以通过命令直接吧java进行编译。来加快运行速度。
3.JVM有两种启动模式。c1客户端，c2服务端。可以通过java命令参数指定。服装方式启动的慢，他会收集上千次服务相关的运行情况。来计算出最佳运行方式。
4.gc有4种，serilized gc,parallize gc ,m1,g1.

作者回复

2018-05-09

细节有点不准确，1不只编译；2，AOT；3，c1c2是jit compiler名字，如果深入看，不是简单的对应两种启动模式；4 m1->cms? 这是hotspot目前提供的，很多其他选择

Leo

2018-05-08

老师，请问解析执行是解析成机器码吗？这和编译成机器码有什么区别？

作者回复

2018-05-08

动态编译会缓存起来，适合重复使用的场景

YANGFEI

2018-05-08

所谓的跨平台、一次编写到处运行，我觉得这个说法并不严谨，它也需要在目标运行机器上安装好jre

小陈

2018-05-07

老师，AOT的存在，不是又有可能丧失跨平台特性吗？

Aaron亚伦

2018-05-07

评论区的高手真多

华烬

2018-05-07

关于jvm，之前看过这样一种说法，老师帮忙看下理解。jvm其实是有3个概念的：规范，实现和实例。个人理解规范就是jvm的各中机制，比如类加载机制，垃圾回收，JMM等等，只要符合规范，不同人可以实现不同的虚拟机，也是因为规范，可以让jvm可以运行java以外的语言，只要满足字节码规范即可。然后是实现，就如刚才所说，只要符合规范就可以，所以现在是有很多种虚拟机的，除最常用的Hotspot VM,还有JRockit,以及J9 VM等等，内部实现是有所不同的，比如关于老年代，Hotspot在jdk8之前是用永久代来实现的，其他虚拟机是没有这个概念的，这就是不同的虚拟机，实现可以是不同的例子。最后是实例，这个是让java语言跨平台的正常原因，提供了一个运行的软件环境，一个中间层，来屏蔽差异，当然，不同平台上的虚拟机也是不一样的，就算都是Hotspot。类比的话，规范就是接口，实现就是一个实现了接口的类，而实例才是一个最后new出来的对象。

作者回复

2018-05-07

不错，文章基本是以最广泛的oracle jdk为基础

Hesher

2018-05-07

平时工作编写的主要是业务代码，对于Java基础类库、底层特性不需要太过关注也很少用得上，但对于线上服务，性能分析和调优需要懂得一些JVM知识，常用的内存和线程堆栈分析工具要掌握用来救火。

Java发展到现在，越来越好用，生态越来越强大，已经成为了企业中最重要的程序语言了。基础类库越发完善，开源框架百家争鸣，即使是JVM也有了一众顶级的上层语言。无论从市场、就业还是技术本身来说，Java平台都值得全面深入地学习。

作者回复

2018-05-07

顶你，最大的用户群体，最广泛的社区和厂商，实际开发中语言酷不见得是足够的，很多硬性的能力才是决定因素

Hidden

2018-05-07

最近，每天都在纠结要不要离职，在这家公司一年了，感觉什么提升都没有，除了获得 报酬，别的好像并没有什么收获

作者回复

2018-05-07

问下自己 跳槽为了什么，是不是跳槽就/才能解决，如果是也不用犹豫，否则也可以发现其他选择

miren6

2018-05-07

看到作者一个回复说未来jre将退出舞台，此话怎讲？

作者回复

2018-05-07

有了jigsaw，还有其他的一些变化，jre单独存在的必要已经没有了

miren6

2018-05-07

看到作者一个回复说未来jre将退出舞台，此话怎讲？

作者回复

2018-05-07

有了jigsaw，利用jlink可以定制最合适的runtime，plugin之类也逐渐退役了，jre的存在需求在降低

陳新Fāns

2018-05-07

看了老师讲的觉得自己要学的东西还有很多

фщэьшух

2018-05-06

老师看到您给同学的评论说是jit在运行时，aot在编译时，我更懵了，老师给我说明一下，我理解的是两者都是在编译时。

作者回复

2018-05-07

jit是运行时的动态编译，通常说的编译阶段是说javac那种，看语境

фщэьшух

2018-05-06

JAVA的jit是运行前将源码编译为字节码文件。但是jit和aot都可以将字节码编译为机器码，只是jit将一些高频代码编译，aot将所有代码编译。

老师这样说对嘛？

那个谁

2018-05-06

语言设计是一种权衡的结果。关键是要理解解释型语言和编译型语言的适应场景和优缺点，如何扬长避短。jit是一种对热点代码效率的优化，但有时是会反编译，这个情况不知道作者怎么看

江东去

2018-05-06

可不可以这样理解，JIT编译器就是把运行时JVM解释执行产生的codeCache对应的热点代码，动态编译成机器码去执行。

龙猫9527

2018-05-06

笔记：前端编译器有javac，作用：将.java源文件编译成.class文件；后端编译器有jit编译器，可以将字节码生成本地机器码；热点代码是指频繁运行的某个方法或者是方法块，为了优化热点代码的执行效率，使用JIT编译器将这些代码编译成机器码。

汉彬

2018-05-06

只用到jdk8，现在jdk10都要出来了。对于jdk版本的使用有什么要特别注意的地方？

作者回复

2018-05-06

看需求，有没有什么特性是至关重要的；也要考虑团队能力；对于大公司，安全往往至关重要，那就要考虑，有没有安全漏洞，有没有可靠支持

天天向上

2018-05-05

编译执行是指直接将源文件编译成机器码，然后直接执行；解析执行是指解析一条指令，执行一条指令。
JIT，其用了2个计数器:方法计数器，回边计数器，当两者之和到达一定组阈值时，会直接将字节码编译成机器指令。由于JVM内存有限，所以仅仅将热点程序编译成机器码
杨老师，有2个问题想请教下：
1，我了解过C的编译过程：源文件->汇编指令->二进制(机器码)，那么JAVA的编译执行过程也是会有转成"汇编指令"这一步吗？
2，aot直接将源文件编译成机器码，那么对于不同的平台，那不是需要对源文件各编译一次，不就做不到编译一次。到处执行了？

作者回复

2018-05-06

1，我的理解jit是通过IR（中间指令）
2，是的AOT有代价

YI

2018-05-05

曾经研究过动态程序分析（dynamic program analysis），读了很多论文，存在很多疑问，今天杨老师的开篇第一讲却让我想明白了很多问题。

作者回复

2018-05-06

互相交流，三人行必有我师

『OMG』Moshow

2018-05-05

對了，還要記得提及一個，一次編譯多次運行，也要是對應兼容的jdk版本。例如1.5編譯ok的，1.10不一定ok，jdk10可以的，jdk6不一定可以。

作者回复

2018-05-06

向后兼容，Java算是最高等级支持，虽然不能100%

会飞的毛毛虫

2018-05-05

Java平台由于jvm的存在，使得编写、编译的平台无关性，也实现了语言的无关性，使编程语言中出现了一种新语言，叫jvm语言。Java是官方的语言，随着JDK的升级，也在吸收其他语言的

长处。lambda，jshell.....高手很多，凳子已经准备好啦��

作者回复

2018-05-06

的确，日新月异

---

zhhp

2018-05-05

1、我对Java跨平台的理解是Java不像C/C++之类的语言那样是面向操作系统编程的(线程、内存管理、网络、IO等功能由操作系统来提供)，而是由Java虚拟机提供一个中立的平台，屏蔽掉操作系统之间的差异，Java语言面向这个中立的平台编程，所以能有很好的跨平台能力。

2、请问老师Java的AOT和Android的ART有关系吗？以后Java的运行时有没有可能变得很小巧，然后Java应用可以和运行时一起打包为一个单独的可执行文件？

作者回复

2018-05-06

我说的是OracleJdk的特性，其实虚拟机有各种实现，ART也是使用了一种AOT技术

代码狂徒

2018-05-05

您好，博主，我这边有看到一篇16年的关于JIT 的文章，那时候还没有java9吧？上面有提到说AOT是事前编译，属于静态编译，本人理解不是很深入，所以还请博主解释一下，跟您说的这个AOT是一回事吗？

作者回复

2018-05-06

是的，只是oracle jdk9才加入，有的实现早得多

不会游泳的鱼

2018-05-05

jit一般在什么情况下使用？jit是jvm的扩展么？

作者回复

2018-05-06

可以看作一个模块，算是高性能jvm的标配，比如hotspot、j9

啃yi嘴泥

2018-05-05

JVM虚拟机，从另一个角度来看，可不可以理解为也是一个微型的操作系统，对开发人员来说，这个操作系统提供了系统API(jdk类库)以及一些管理接口用来反应操作系统当前运行时的各项指标，包括内存，线程信息等。不知道可不可以这样去理解？

作者回复

2018-05-05

有道理，不过需要注意的是Java平台的管理相关API大部分的抽象是平台中立的

---

mao

2018-05-05

看评论真的涨知识，这个课程很赞

作者回复

2018-05-05

藏龙卧虎，专栏只是个大家交流的引子

十三

2018-05-05

瞬间觉得自己面对的是一片大海，作为一个小白，有一点点的兴奋和好奇

学无止境

2018-05-05

希望老师后面多讲讲高并发相关知识。谢谢

作者回复

2018-05-05

必须的

1024

2018-05-05

Java语言是跨平台，但底下的运行环境却是平台相关的，正是Java虚拟机与字节码造就了"compile once，run anywhere"。也正是这一特性，给人错觉"Java运行慢，不如C/C++"，可以Java真的慢？答案是否定的。快慢，是相对的！Hadoop、Spark等框架恰恰是这一否定的有力支持。

作者回复

2018-05-05

Java自身在不断提高，我回头补充一个专题系统介绍提高启动速度的几大法宝；hadoop稍微不同，本质是打群架.....

Libra

2018-05-05

个人观点，java的跨平台实质是利用不同版本的jvm对底层屏蔽，这样的跨平台只是相对于用户来说的。

作者回复

2018-05-05

对，跑在不同机器的应用不需要重新编译，这也是我为什么说Java跨平台是最高标准，但也不是没有代价；
类似Go等跨平台是需要重新编译的，但也带来了很多好处，比如更轻量级。
各有适用场合，Java有了aot，能力相对更加多样化

---

gesanri

2018-05-05

文章中提到aot是直接将字节码转换成机器码，如果是这样，那不是跟直接编译执行一样吗？我看周志明的书说的是aot是将Java文件直接编译成机器码，不知道哪种是对的

作者回复

2018-05-05

注意aot是编译期发生的，jit是运行时，不是一码事

孙晓明

2018-05-05

对"java平台"应该怎么理解，不应该是语言吗？感觉eclipse之类的是平台。

作者回复

2018-05-05

不同角度，Java不仅是语言，相关jvm、工具、类库组成的平台，才是我们日常使用的东西

---

sky

2018-05-05

蓝图里不应该少了concurrent包，这个面试必问。

作者回复

2018-05-05

必须的，后面有单独章节

kbrx93

2018-05-05

1. 用解释型或 XX 型语言本身就不准确，设计者眼中哪有什么型，只不过设计出来发现用某种方式运行更好用，所以 XX 型就是呈现给开发者的印象。
2. 比如你设计一个变量，你在设计的时候不会因为这个变量要设计成 static 的，所以它不能干什么，能干什么。相反，你会想，设计成 static 与 非 static 那个更好用，当然是怎么好用怎么来。
3. 理解了这一点，你就会发现，Java 的 JIT 与 AOT 之类的，都是为了好用。

作者回复

2018-05-05

实用主义

朱林朋

2018-05-05

我们还在用java7,不过前面我看新闻，java10已经出来了，9既然提出了aot这种新特性，为啥还能这么快被淘汰，它有啥缺点呢，请老师指点!

作者回复

2018-05-05

9的AOT是试验特性，能力有限，比如只支持Linux x64，java base模块等，快速发展是业界的要求，云计算等推动Java加速，后面可以考虑使用LTS版本

A小星

2018-05-05

java垃圾回收机制！是回收不再使用的堆内存嘛？那内存泄漏，gc不会主动回收这部分内存嘛

作者回复

2018-05-05

基本正确，但不只是堆，还有很多东西在堆外，后面会有详细分析

杨文奇

2018-05-05

解释器是把代码一行一行解释为二进制指令，编译器是把代码一次性编译为二进制指令，对JAVA语言来说，在执行阶段区分编译和解释，但对C语言来说，在执行阶段是直接执行二进制指令，不存在编译和解释，不知道理解对吗？

作者回复

2018-05-05

嗯，c语言直接编译为机器码，并不存在虚拟机，不存在Java的部分概念

锐

2018-05-05

Jdk8已经停止维护，是不是可以直接上10了？

作者回复

2018-05-05

不是停止维护，已经免费支持了5年了啊，时间太快了，商业支持还有，是可以考虑新的版本，很多能力有大幅提升