Java代码是怎么运行的?

版本: 2018/9/22-1(00:02)

1、Java代码的运行需要什么?

> 必须要有JRE(java 运行时环境)

JRE

2、JRE(Java运行时环境)里面有什么?

> 1. Java虚拟机
> 2. Java核心类库

JDK

3、JDK是什么?

> 1. Java开发工具包
> 2. JDK里面也包含了JRE
> 3. 此外还有一系列开发和诊断工具

C++

4、C++代码是如何运行的?

> 1. C++代码不需要额外的运行时
> 2. C++代码会直接编译成CPU所能理解的机器码，直接执行

5、C++版本的“Hello World”编译出来的机器码和反编译的汇编代码

```
; 最左列是偏移；中间列是给机器读的机器码；最右列是给人读的汇编代码
0x00:  55                    push   rbp
0x01:  48 89 e5              mov    rbp,rsp
0x04:  48 83 ec 10           sub    rsp,0x10
0x08:  48 8d 3d 3b 00 00 00  lea    rdi,[rip+0x3b]
                                      ; 加载 "Hello, World!\n"
0x0f:  c7 45 fc 00 00 00 00  mov    DWORD PTR [rbp-0x4],0x0
0x16:  b0 00                 mov    al,0x0
0x18:  e8 0d 00 00 00        call   0x12
                                      ; 调用 printf 方法
0x1d:  31 c9                 xor    ecx,ecx
0x1f:  89 45 f8              mov    DWORD PTR [rbp-0x8],eax
0x22:  89 c8                 mov    eax,ecx
0x24:  48 83 c4 10           add    rsp,0x10
0x28:  5d                    pop    rbp
0x29:  c3                    ret
```

Java虚拟机

6、Java为什么要在虚拟机中运行

> 1. Java作为高级程序语言，愈发复杂，抽象程度高，因此无法在硬件上运行。
> 2. 这就需要设计一个面向Java语言特性的虚拟机，并通过编译器将Java程序转换为该虚拟机所能识别的指令序列，也就是 Java字节码

7、虚拟机的优点

> 1. 跨平台：一次编译到处执行。
> 2. 提供托管环境：JVM能代替程序员去做一些复杂且容易出错的工作。
> 3. 峰值性能更佳：JVM进行预热后，针对热点方法由于大量优化，且能够在运行中动态获取方法执行的情况等信息，从而能进一步优化，提高峰值性能。

Java字节码

8、为什么要叫Java字节码

> 1. 因为指令的操作码 opcode 被固定为一个字节。

2. 如下“Hello World”，都是由一个个字节组成。

```
# 最左列是偏移；中间列是给虚拟机读的机器码；最右列是给人读的代码
0x00:  b2 00 02         getstatic java.lang.System.out
0x03:  12 03            ldc "Hello, World!"
0x05:  b6 00 04         invokevirtual java.io.PrintStream.println
0x08:  b1               return
```

实现方式

9、Java虚拟机的实现方式?

1. 可以由硬件实现： Java Proccessor(Java虚拟机在硬件上的实现)，当前有十多种具体实现的JVM。
2. 最常见的是在各个平台上提供软件实现。

10、Java虚拟机采用软件实现的好处?

1. 能够实现“一次编写，到处运行”，因为程序一旦转为Java字节码，就可以在不同平台的虚拟机实现里执行。

托管环境

11、采用JVM作为托管环境的好处是什么?

1. 这个托管环境能够代 替我们处理一些代码中冗长而且容易出错的部分。
2. 自动内存管理
3. 垃圾回收
4. 数组越界、动态类型、安全权限等动态检测。

Java虚拟机如何运行Java字节码?

12、标准JDK中的HotSpot是如何运行Java字节码的呢?

需要从两个角度考虑:

1. 虚拟机角度
2. 底层硬件角度

虚拟机角度

13、从虚拟机角度是如何运行Java字节码的?

1. 首先会将Java代码编译成的class文件加载到JVM
2. 加载后的Java类会存放到方法区中。

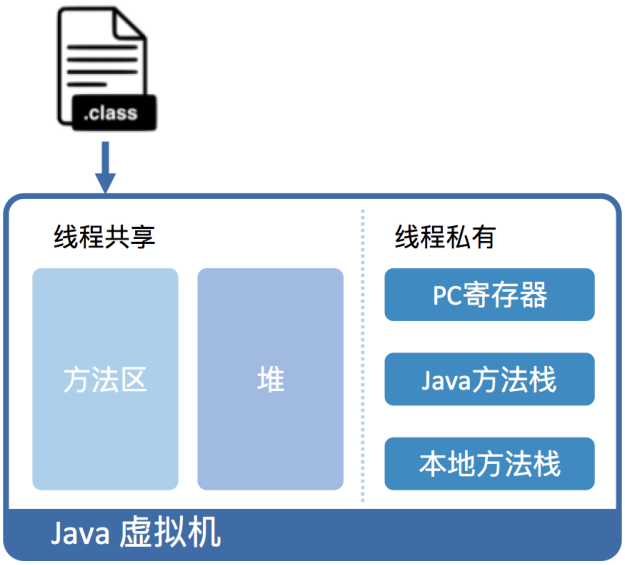
3. 实际运行时，JVM会执行方法区内的代码。
4. 运行时，每当调用进入一个Java方法，JVM会在当前线程的Java方法栈中生成一个栈帧。
5. 退出执行的Java方法时，无论是不是正常返回，JVM都会弹出并舍弃掉当前线程的当前栈帧。

14、栈帧是什么?

1. 用于存放局部变量、字节码的操作数
2. 栈帧的大小是提前计算好的。
3. JVM不要求栈帧在内存空间内连续分布。

15、JVM中内存组成部分有哪些?线程共享的是哪些部分?线程私有的是哪些部分?

线程共享:
1. 方法区：存放加载后的Java类
1. 堆：创建的对象实例
线程私有:
1. Java方法栈: 面向Java方法
1. 本地方法栈: 面向C++写的native方法
1. PC寄存器: 存放各个线程执行位置。



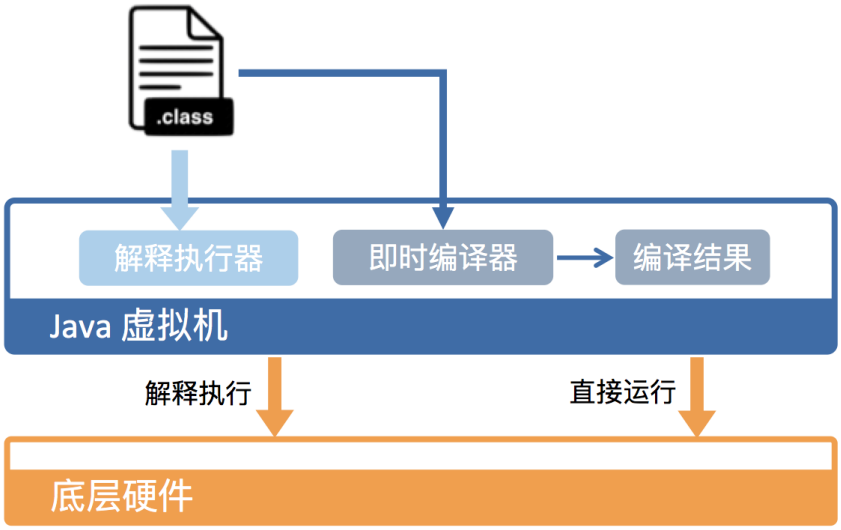
硬件角度

16、从硬件角度角度是如何运行Java字节码的?

1. Java字节码无法直接执行
2. JVM需要将字节码翻译成机器码

17、JVM(如HotSpot)如何将字节码翻译成机器码?

> 两种方法:
> 1. 解释执行-逐条将字节码翻译成机器码执行
> 1. 即时编译(Just-In-Time compilation, JIT)-将一个方法中的字节码都编译成机器码后再执行。



18、HotSpot是如何采用这两种方法去执行Java代码呢?

> 1. 默认采用采用混合模式
> 2. 综合了两者的优点
> 3. 会先解释执行字节码
> 4. 之后将其中反复执行的热点代码，以方法为单位进行即时编译

19、即时编译是以什么为单位进行优化的?

> 1. 以方法为单位

20、Java虚拟机的效率如何?

> 1. HotSpot采用多种技术来提升启动性能和峰值性能。
> 2. 即时编译就是重要技术之一。
> 3. 即时编译建立在二八定律之上，也就是20%的代码占据了80%的计算资源。

21、为什么Java不和C++一样直接编译成机器码呢?

> 1. 理论上Java的性能可以比C++更高。
> 2. 与静态变异相比，即时编译拥有程序的运行时信息，并能以此进行相应的优化。
> 3. 借助这些信息，可以规避虚方法调用的开销，从而比静态变异的C++程序的性能更高。

22、HotSpot内置了哪几种即时编译器？分别有什么作用和特点?

> 1. C1：Client编译器，优化手段简单，编译时间短，适合对启动性能有要求的客户端程序。
> 2. C2：Server编译器，适合对峰值性能有要求的服务端程序，优化手段复杂，编译时间长，生成的代码执行效率高。
> 3. Graal：实验性即使编译器，Java10中引入。

23、HotSpot如何采用这些即时编译器进行编译工作?

> 1. HotSpot默认采用分层编译(从Java7开始)
> 2. 热点方法会先用C1编译
> 3. 然后热点方法中的热点，进一步用C2编译。

24、 HotSpot中编译线程是什么?

> 1. 为了不干扰应用的执行，即时编译会在额外的编译线程中进行。
> 2. HotSpot会根据CPU数量设置编译线程的数目
> 3. 并且1:2的比例分配给C1和C2编译器

25、“资源充足时，即时编译会替换解释执行”这种说法是否正确?

> 正确!
>
> 1. 在计算资源充足的情况下，字节码的解释执行和即时编译可同时进行。
> 2. 即时编译完成后的机器码会在 下次调用该方法时启用，以替换原本的解释执行。

思考题

26、Java语言和JVM看待boolean类型的方式是否相同?

```
public static void main(String[] args) {
  boolean flag = true;
  if (flag) System.out.println("Hello, Java!");
  if (flag == true) System.out.println("Hello, JVM!");
}
```

> 1. JVM将boolean当做int值来处理, 因此flag = true = iconst_1
> 2. 如果通过工具将flag改为 iconst_2, 会发现if(flag)依旧成立，因为iconst_2也为真
> 3. 但是if(flag == true)就不成立了，iconst_1 不等于 inconst_2

问题汇总

1. Java代码的运行必须要有什么?

2. JRE(Java运行时环境)里面有什么?
3. JDK是什么?
4. C++代码是如何运行的?
5. Java为什么要在虚拟机中运行
6. 虚拟机的优点
7. Java字节码为什么要叫“Java字节码”?
8. Java虚拟机的实现方式有哪两类?
9. Java虚拟机采用软件实现的好处是什么?
10. 采用JVM作为托管环境的好处是什么?
11. 标准JDK中的HotSpot是如何运行Java字节码的呢?(虚拟机角度和底层硬件角度)
12. 从虚拟机角度是如何运行Java字节码的?
13. 栈帧是什么?
14. JVM中内存组成部分有哪些?线程共享的是哪些部分?线程私有的是哪些部分?
15. 从底层硬件角度是如何运行Java字代码的?
16. JVM(如HotSpot)如何将字节码翻译成机器码?有哪几种方法?
17. 解释执行是指什么?
18. 即时编译是指什么?
19. HotSpot是如何采用这两种方法去高效执行Java代码呢?
20. 即时编译是以什么为单位进行优化的?
21. Java虚拟机的效率如何
22. 为什么Java不和C++一样直接编译成机器码呢?
23. HotSpot内置了哪几种即时编译器?分别有什么作用和特点?
24. HotSpot如何采用这些即时编译器进行编译工作?
25. HotSpot中编译线程是什么?
26. “资源充足时,即时编译会替换解释执行”这种说法是否正确?
27. Java语言和JVM看待boolean类型的方式是否相同?为什么呢?

参考资料

1. 深入拆解Java虚拟机