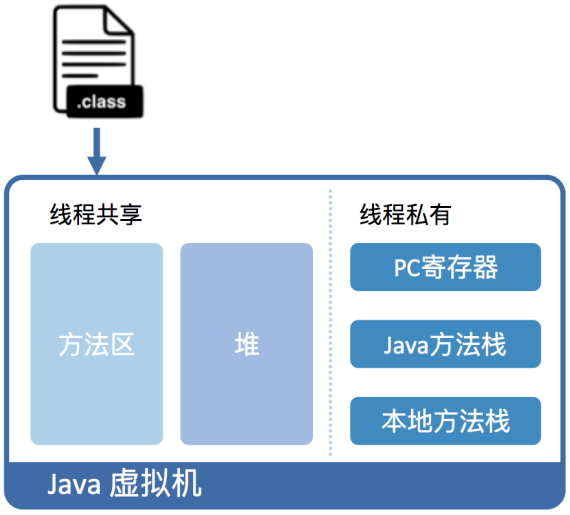
Java字节码,是因为Java字节码指令的操作码被固定为一个字节.

1.1 Java虚拟机具体是怎样运行Java字节码

从虚拟机角度看,执行Java代码首先需要将它编译而成的class文件加载到虚拟机中.加载后的Java类会被存放于方法区(Method Area)中,实际运行时,虚拟机会执行方法区内的代码.

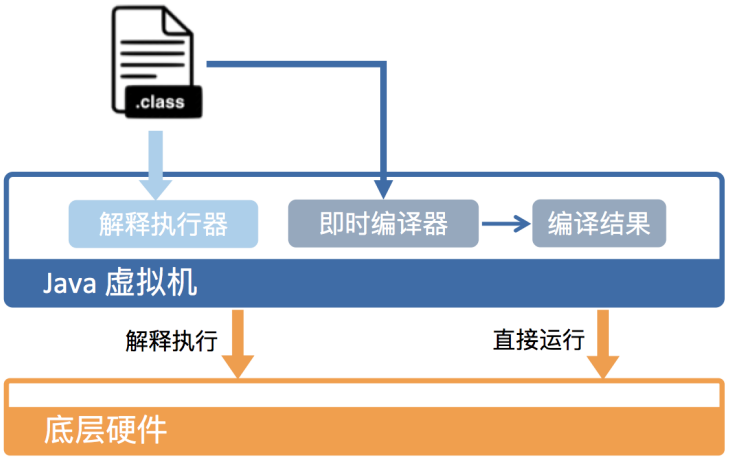
Java虚拟机会将栈细分为面向Java方法的Java方法栈,面向本地方法的本地方法栈,以及存放各个线程执行位置的pc寄存器:



在运行过程中,每当调用一个Java方法,Java虚拟机会在当前线程的Java方法栈中生成一个栈帧,用以存放局部变量以及字节码的操作数,这个栈帧是提前计算好的,而且Java虚拟机不要求栈帧在内存空间里连续分布.当退出当前执行的方法时,不管是正常返回还是异常返回,Java虚拟机均会弹出当前线程的当前栈帧.,并将之舍弃.

从硬件角度看,Java字节码无法直接执行,因此,Java虚拟机需要将字节码翻译成机器码.

在HotSpot里,上述翻译过程有两种形式:1.解释执行,即逐条将字节码翻译成机器码并执行,第二种是即使编译,即将一个方法中包含的所有字节码编译成机器码后再执行.



hotspot内置了多个即时编译器:C1,C2和Graal.Graal是Java 10正式引入的实验性即时编译器.

C1又叫做Client编译器,面向的是对启动性能有要求的客户端GUI程序,采用的优化手段相对简单,因此编译时间较短.

C2又叫做Server编译器,面向的是对峰值性能有要求的服务器端程序,采用的优化手段相对复杂,编译事件较长,生成代码的执行效率较高.

Java7开始,hotspot默认采用分层编译的方式:热点方法首先会被C1编译,而后热点方法中的热点会进一步被C2编译.

为了不干扰应用的正常进行,hotspot的即使编译是放在额外的编译线程中进行的.hotspot会根据CPU的数量设置编译线程的数目,并且按1:2的比例配置给C1和C2编译器.

总结

Java程序编译而成的class文件,需要先加载至方法区中,方能在Java虚拟机中运行,为了提高运行效率,标准jdk中的hotspot虚拟机采用的是一种混合执行的策略