DEC
12

Just In Time & The Adaptive Optimizer

Any program is ultimately run on a processor. Processors understand machine code.

Programs written in any language need to be converted to machine code before they can be executed. Some programming languages when compiled, directly becomes the machine code; but in Java, machine code is not formed on compilation, rather Byte code is generated.

JVM interprets the byte code during run-time. JVM needs to covert the byte code into machine code. Now, this conversion is one extra step, isn't ? If machine code is directly available, the conversion wouldn't be needed. That is why, generally interpreted code is slower than executable code. But this Byte Code strategy helps in establishing Java's main advantages- Portability and Security.

So, is execution of Java programs slower than programs that give machine code directly ? The answer is partly Yes. But then, the difference is very small. Java achieves the speed via two techniques incorporated into it - **JIT** {*Just-In-Time*}compiler and **Adaptive Optimizer**.

JIT Compiler translates Byte Code into native code at run-time. When a particular method is called during run-time, the method's byte code will be converted by the JIT into native code.Conversion of these byte codes by JIT will need processor time and memory usage, which would normally result in slower execution. But then, JIT doesn't compile every method every time a method is called. Neither does it compile the very first time and use it later. It holds a counter against each method and makes note of how many times the method has been called. When the count of method calls crosses a ***threshold***, JIT converts it into machine code, thus balancing performance and time.

Whether Java performs JIT (i.e., executing native codes) or interprets the Byte codes, the outcome will be the same. The only difference would be that the compiled code would run faster. But then, all of the Java program can not be compiled and given, as that would take a lot of processor time and memory which would slow down the execution ! Also, Java has run-time checks to make, which makes compilation of all java program not possible. And from Security standpoint, although JIT converts byte code into native code, it is not a risk to security as it is the JVM that is in control of the JIT and the execution environment.

For those methods that have not reached the JIT threshold yet, JVM continues interpreting them. What ***Adaptive Optimizer*** does is, it looks during run-time which chunk of code is being executed largely and calls on the JIT to have it compiled, so that, it can be run faster. The Adaptive Optimizer makes a trade-off between using JIT and Interpreting the Byte code.