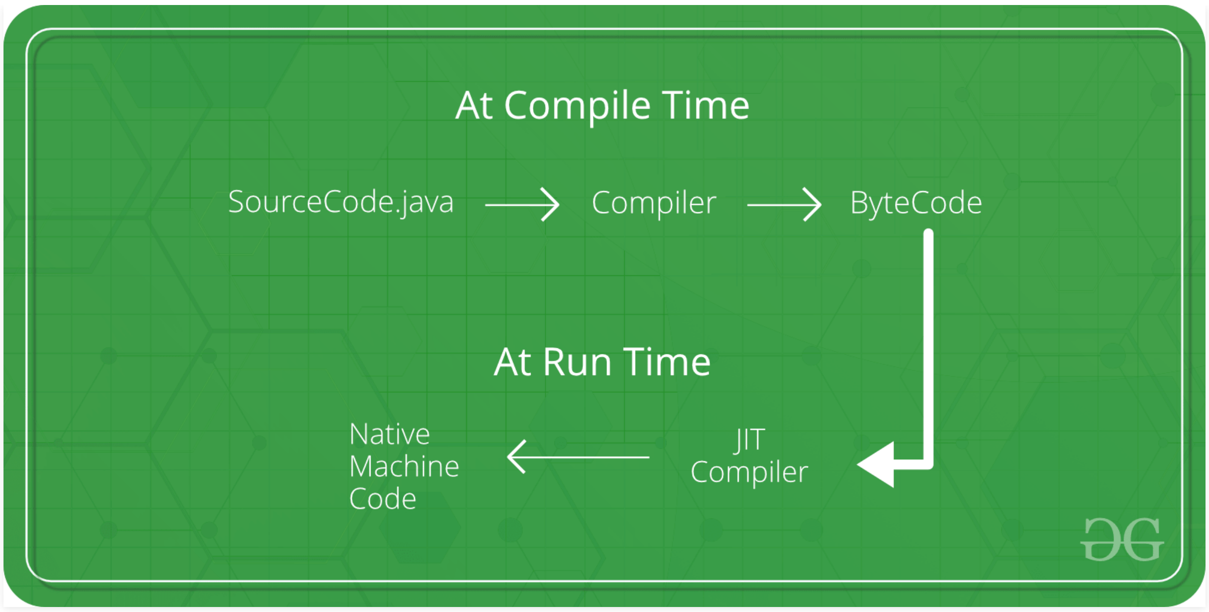
Just In Time Compiler

Difficulty Level : Easy Last Updated : 14 Oct, 2018

The Just-In-Time (JIT) compiler is a an essential part of the JRE i.e. Java Runtime Environment, that is responsible for performance optimization of java based applications at run time. Compiler is one of the key aspects in deciding performance of an application for both parties i.e. the end user and the application developer.

Java JIT Compiler : General Overview

Bytecode is one of the most important features of java that aids in cross-platform execution. Way of converting bytecode to native machine language for execution has a huge impact on the speed of it. These Bytecode have to be interpreted or compiled to proper machine instructions depending on the instruction set architecture. Moreover these can be directly executed if the instruction architecture is bytecode based. Interpreting the bytecode affects the speed of execution.
In order to improve performance, JIT compilers interact with the Java Virtual Machine (JVM) at run time and compile suitable bytecode sequences into native machine code. While using a JIT compiler, the hardware is able to execute the native code, as compared to having the JVM interpret the same sequence of bytecode repeatedly and incurring an overhead for the translation process. This subsequently leads to performance gains in the execution speed, unless the compiled methods are executed less frequently.
The JIT compiler is able to perform certain simple optimizations while compiling a series of bytecode to native machine language. Some of these optimizations performed by JIT compilers are data-analysis, reduction of memory accesses by register allocation, translation from stack operations to register operations, elimination of common sub-expressions etc. The greater is the degree of optimization done, the more time a JIT compiler spends in the execution stage. Therefore it cannot afford to do all the optimizations that a static compiler is capable of, because of the extra overhead added to the execution time and moreover it’s view of the program is also restricted.



Working of JIT Compiler

Java follows object oriented approach, as a result it consists of classes. These constitute of bytecode which are platform neutral and are executed by the JVM across diversified architectures.

- At run time, the JVM loads the class files, the semantic of each is determined and appropriate computations are performed. The additional processor and memory usage during interpretation makes a Java application perform slowly as compared to a native application.
- The JIT compiler aids in improving the performance of Java programs by compiling bytecode into native machine code at run time.
- The JIT compiler is enabled throughout, while it gets activated, when a method is invoked. For a compiled method, the JVM directly calls the compiled code, instead of interpreting it. Theoretically speaking, If compiling did not require any processor time or memory usage, the speed of a native compiler and that of a Java compiler would have been same.
- JIT compilation requires processor time and memory usage. When the java virtual machine first starts up, thousands of methods are invoked. Compiling all these methods can significantly affect startup time, even if the end result is a very good performance optimization.