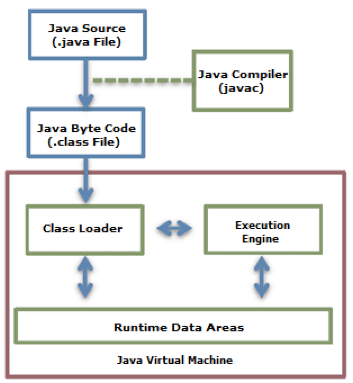
Most Java performance improvement is accomplished by improving the execution engine. In the JIT compiler, various optimization techniques are being introduced so the JVM performance can be continuously improved. The biggest difference between the initial JVM and the latest JVM is the execution engine.

<http://www.cubrid.org/blog/dev-platform/understanding-jvm-internals/>

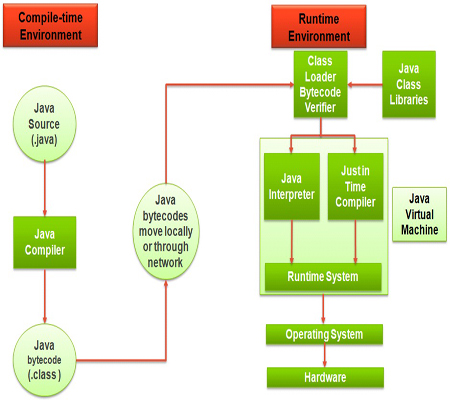


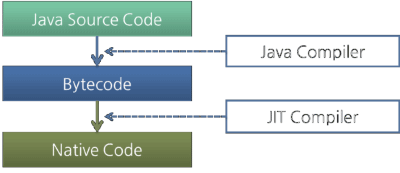
The bytecode (.class) loaded by the class loader is executed by the execution engine.

The Java Bytecode is written in a language that a human can understand, rather than in the language that the machine directly executes. Therefore, the execution engine must change the bytecode to the language that can be executed by the machine in the JVM. The bytecode can be changed to the suitable language in one of two ways.

1. Interpreter
2. Just In Time(JIT) Compiler

Interpreter is slow and JIT is fast, so most JDK includes JIT for bytecode to native code conversion. Under certain situations Interpreters are fast.





What should an Execution Engine contain is not defined in the JVM specifications. Therefore, JVM vendors improve their execution engines using various techniques, and introduce various types of JIT compilers and interpreters.

Oracle Hotspot VM uses a JIT compiler called Hotspot Compiler. It is called Hotspot because Hotspot Compiler searches the 'Hotspot' that requires compiling with the highest priority through profiling, and then it compiles the hotspot to native code. If the method that has the bytecode compiled is no longer frequently invoked, in other words, if the method is not the hotspot any more, the Hotspot VM removes the native code from the cache and runs in interpreter mode. The Hotspot VM is divided into the Server VM and the Client VM, and the two VMs use different JIT compilers.