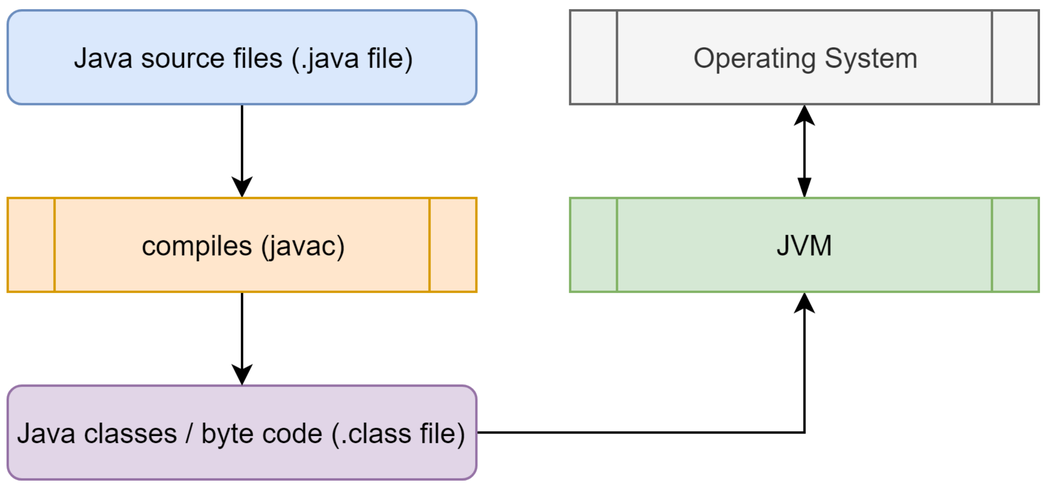
Boosting MicroJIT - A Lightweight Just-In-Time Compiler

Eric Coffin, Scott Young, Kenneth. B. Kent
University of New Brunswick, Faculty of Computer Science
Marius Pirvu, Vijay Sundaresan
IBM Canada
{eric.coffin, scott.young, ken}@unb.ca
{mprivu, vijaysun}@ca.ibm.com

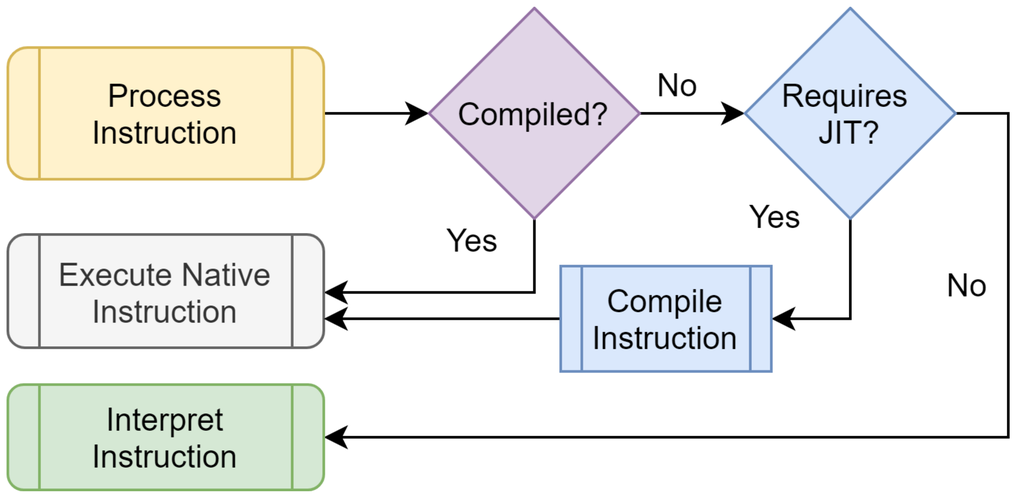
Background

- Programs commonly run on virtual machines (VMs) e.g., Java programs on the Java Virtual Machine (JVM).
- VMs provide a layer of abstraction above operating systems and processor architecture.
- Java is compiled to intermediary form called bytecode.
- Bytecode runs on any system that provides a compatible JVM. Bytecodes are interpreted at run-time by the JVM.



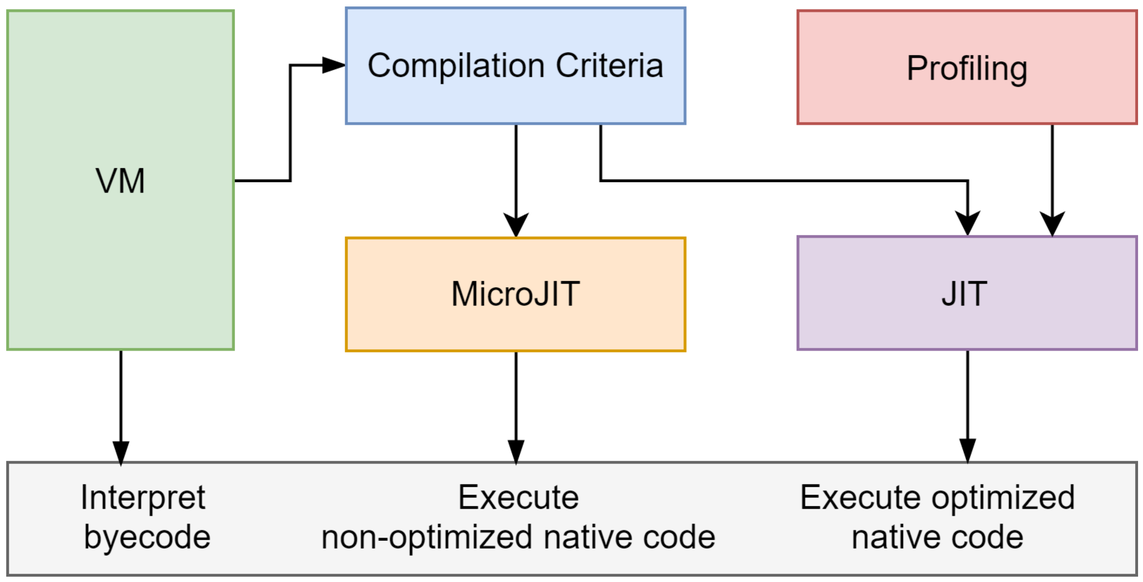
Just-In-Time (JIT) Compiling

- Interpretation slower than native instructions.
- JIT compiles frequently called methods to native instructions during run-time.
- Adds overhead but generally accepted, as throughput is increased for long running programs.



JVM + MicroJIT

- Previous work created a JIT compiler, MicroJIT, for IBM's J9 Java 6 ME JVM.
- Compiles code quickly but performs few optimizations.
- Lightweight JIT for resource constrained systems.
- In 2017 MicroJIT ported to J9 Java 8 SE providing two separately tunable JIT compilers.
- The MicroJIT reduced startup time and improved throughput for short-lived applications, or in cases when a shared-class-cache might not be available.
- Eclipse OpenJ9 JVM uses the JIT compiler in Eclipse OMR.



Motivation

Port MicroJIT to Eclipse Open J9 platform while adding additional performance enhancements:

- **Increase bytecode support** – MicroJIT does not support all bytecodes. We will increase support, further reducing the number of switches to the interpreter.
- **Port instructions to 64-bit** – MicroJIT is limited to 32-bit instructions. Adding 64-bit support will allow the MicroJIT to run on larger platforms.
- **Asynchronous compilation** – MicroJIT compiles methods synchronously on the executing thread. We will investigate an asynchronous approach further increasing performance.