GraalVM

JVM over JVMs?

26/09/2019

github.com/jedrzejserwa/pozitive-tech-graalvm

Agenda

- What is GraalVM
  - High-level overview
  - JVM, JIT, JVMCI
- GraalVM key features
  - Graal JIT
  - Native images
  - Truffle framework
- Summary
  - Few thoughts

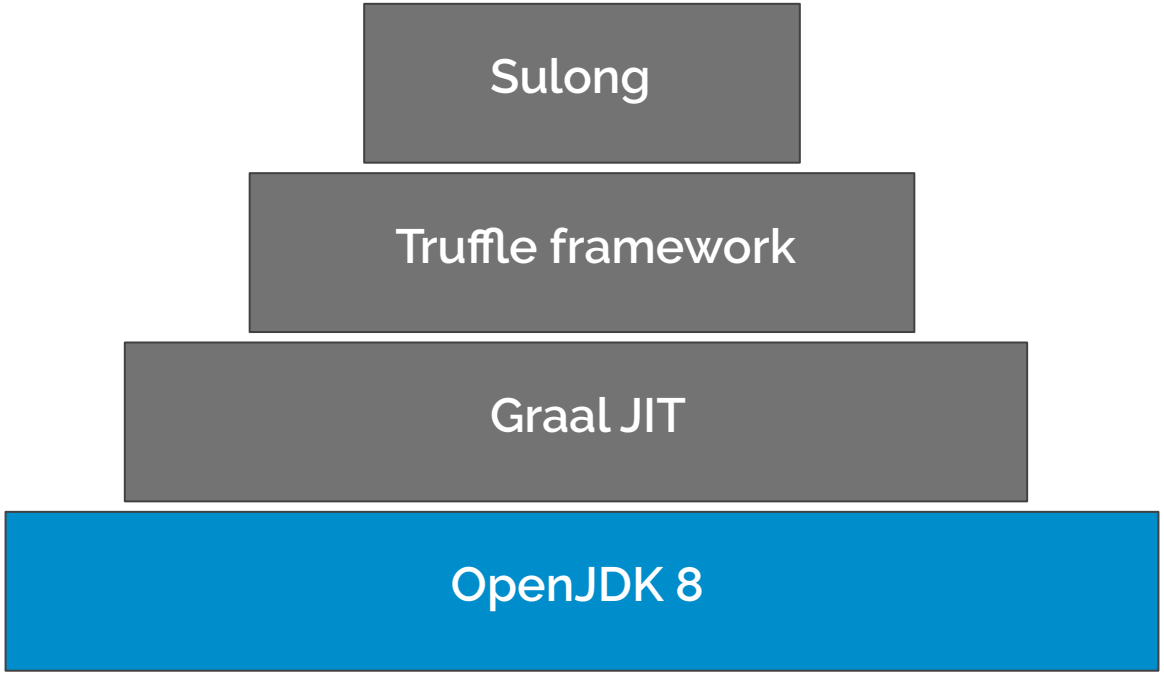
What is GraalVM

High-level overview

- Universal polyglot virtual machine
- Designed and developed in cooperation with Oracle Labs and Johannes Kepler University Linz, Austria
- Available as Community Edition and Enterprise Edition
- Based on OpenJDK8 implementation
- Current version 19.2.x

What is GraalVM

High-level overview



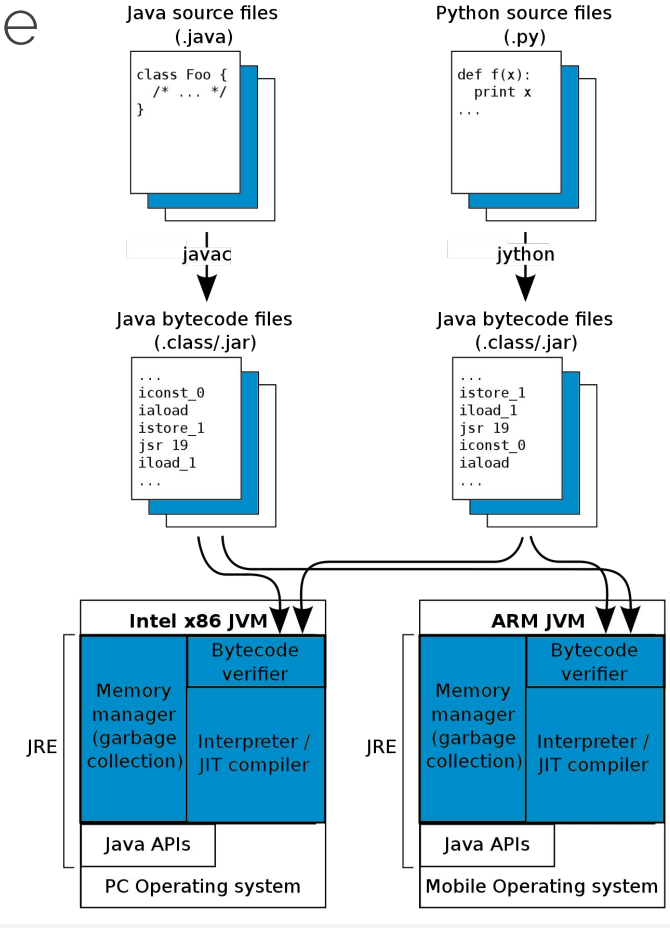
Java Virtual Machine

Brief overview

- A program to run other programs
- Described by Java Virtual Machine Specification[1]
- Implemented/Forked by many companies: Oracle, Azul, Amazon, Alibaba
- Supporting Java, Scala, Groovy, Kotlin, Closure and others

Java Virtual Machine

High-level architecture



JIT

Just In Time Compiler

Types:

- C1 (client)
- C2 (server)
- Tiered

Operations & Optimizations:

- Escape analysis
- Inlining
- Dead code elimination
- Loop unrolling
- Machine code compilation

JVMCI

Java-Level JVM Compiler Interface

- JEP 243: Java-Level JVM Compiler Interface[2]
- Allows JVM to use Java based JIT implementation
- Introduced in Java 9 builds
- Backported to GraalVM
- -XX:+UnlockExperimentalVMOptions -XX:+EnableJVMCI -XX:+UseJVMCICompiler -Djvmci.Compiler=<name of compiler>

GraalVM key features

Graal JIT

- JEP 317: Experimental Java-Based JIT Compiler[3]
- Written in Java
- Available as experimental compiler since Java 9*
- Two modes - jargraal and libgraal
- Supposed to better optimize code with strong abstraction than C2
- Missing key feature - loop vectorization[4]
- The most widely known example of use in production - Twitter

GraalVM key features

Native images

- Allows to create standalone executable
- Fast application startup and low memory footprint
- SubstrateVM as key component
- Restrictions on usage
- Still in experimental stage
- Usage - CLI tools, FaaS

GraalVM key features

Truffle framework

- Language AST interpreter
- Allows to integrate your own programming language with GraalVM
- Built on top of Graal Compiler
- Ships with implementations of
  - JavaScript (ECMAScript 2017 compliant)
  - Python 3
  - R
  - Ruby
- Sulong built on top of Truffle framework

Summary

With GraalVM you *can**

- Boost your application runtime
- Implement your own programming language that runs on GraalVM
- Mix different languages on the same VM
- Boost your application startup

** depending on your context*

Summary

Few thoughts

- Know your business use case
- Know the limits
- Know the metrics
- Do your own research

Sources

Sources:

[1] https://docs.oracle.com/javase/specs/index.html
[2] https://openjdk.java.net/jeps/243
[3] https://openjdk.java.net/jeps/317
[4] https://github.com/oracle/graal/issues/864
[5] https://openjdk.java.net/jeps/295

https://www.graalvm.org/docs
https://github.com/oracle/graal
https://bugs.openjdk.java.net/browse/JDK-8220623
https://www.dynatrace.com/news/blog/new-ways-introducing-compiled-code-java-9
https://commons.wikimedia.org/wiki/File:Java_virtual_machine_architecture.svg#/media/Plik:Java_virtual_machine_architecture.svg

Materials

GraalVM research paper
GraalVM CE Source code
GraalVM Docs

Understanding How Graal Works
Libgraal explained
Graal JIT vs C2
Native image Java Limitations
Native image tool flags

GraalVM in Goldman Sachs
GraalVM CUDA integration
Graal in Twitter
React Server Side Rendering for Closure

GraalVM Demos
Renaissance benchmarks

Q & A

Thank you