

GraalVM: Aplicaciones nativas AOT para los lenguajes de la JVM

Víctor Orozco

2 de febrero de 2022

Nabenik



NABENIK

¿Que es GraalVM

- Maquina virtual políglota por Oracle Labs
- JVM Langs, Truffle, LLVM
- Escrita en Java
- Open Source y Enterprise Edition

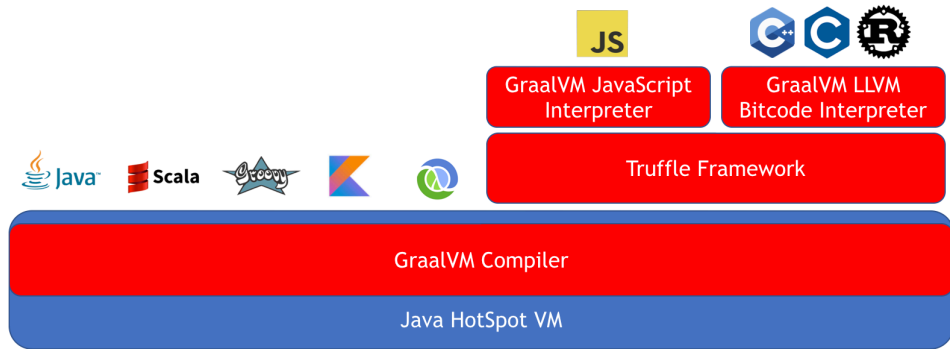


Figura 1: GraalVM Overview

Puntos a resaltar

1. TCK'd JDK
2. Compilador JIT
3. **Java Native Image**
4. Polyglot VM



GraalVM™

[Home](#) » [Blogs](#) » Oracle Includes GraalVM Support in Java SE Subscription



Oracle Includes GraalVM Support in Java SE Subscription



BY: [MIKE VIZARD](#) ON FEBRUARY 3, 2021 — [0 COMMENTS](#)

Oracle announced today it is [adding a GraalVM Enterprise virtual machine for building Java applications to Java SE Subscription](#) at no additional cost.

Donald Smith, senior director of product management from the Java Platform Group at Oracle, said GraalVM Enterprise provides developers with a high-performance application runtime, built on Oracle Java SE, that includes a compiler that accelerates [Java](#) application execution. That's critical as developers look to build microservices-based applications employing microservices that tend to be latency sensitive, Smith said.

CASE STUDY: ORACLE AND TWITTER


How Oracle GraalVM Supercharged Twitter's Microservices Platform

By Holger Mueller

Basic Research on Computing Fundamentals Delivers Once-in-a-Lifetime Performance Improvements

Twitter runs one of the most visible social networks in the world. Its move to a microservices infrastructure has addressed scalability challenges it faced in the past. Because Twitter is a free service (apart from the premium features it offers to enterprise customers), cost-effectiveness is a necessity that must be balanced with uptime.

As such, the company is constantly looking into ways to increase availability of the platform while keeping an eye on costs. Twitter saw Oracle GraalVM, a language-independent compiler engine and virtual machine, and decided to try it. Average CPU savings for compiler innovation are in the 1~2 percent range, but using Oracle GraalVM, Twitter realized between 8 and 11 percent CPU savings, depending on the microservice ported. This enables rare, once-in-a-lifetime savings for compiler innovation.



Imágenes nativas



- Thread scheduling, gestión de memoria
- JVM se desarrolló en los 90's y 2000 como un entorno con compilación JIT (C2)
- Peak performance
- Detección de hotspots

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Native Image

Native Image is a technology to ahead-of-time compile Java code to a standalone executable, called a **native image**. This executable includes the application classes, classes from its dependencies, runtime library classes, and statically linked native code from JDK. It does not run on the Java VM, but includes necessary components like memory management, thread scheduling, and so on from a different runtime system, called “Substrate VM”. Substrate VM is the name for the runtime components (like the deoptimizer, garbage collector, thread scheduling etc.). The resulting program has faster startup time and lower runtime memory overhead compared to a JVM.

Compilación AOT - Wikipedia

En informática, Compilación anticipada es el acto de compilar un lenguaje de programación de alto nivel como C o C++, o un lenguaje intermedio como Java bytecode o el Common Intermediate Language de .NET, a un **código de máquina nativo** con la intención de ejecutar el archivo binario resultante nativamente

Static Linking - Indiana University

El enlace estático es el resultado de que el enlazador copia todas las rutinas de la biblioteca utilizadas en el programa en la imagen ejecutable. Esto puede requerir más espacio en disco y memoria que la vinculación dinámica, pero es más rápido y más portátil, ya que no requiere la presencia de la biblioteca en el sistema donde se ejecuta.

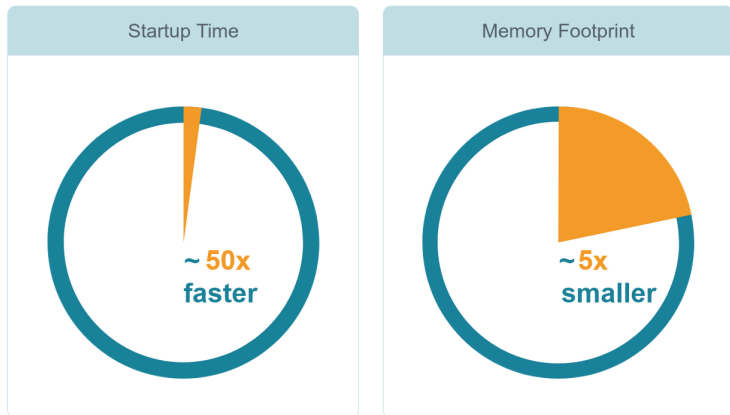
GraalVM Native

GraalVM Native es una tecnología de **compilación AOT de bytecode Java**. Permite crear un **ejecutable con static linking** que incluye clases, bibliotecas y los módulos necesarios del JDK junto a SubstrateVM

Algunas otras implementaciones Bytecode AOT

- ExcelsiorJET
- GNU Compiler for Java
- ART (Android)
- **IBM OpenJ9**

GraalVM for Microservices



Microservices frameworks integrated with GraalVM



- Proyecto Java 11
- Maven
- Oracle Helidon
- MicroProfile

Consideraciones finales



Consideraciones finales

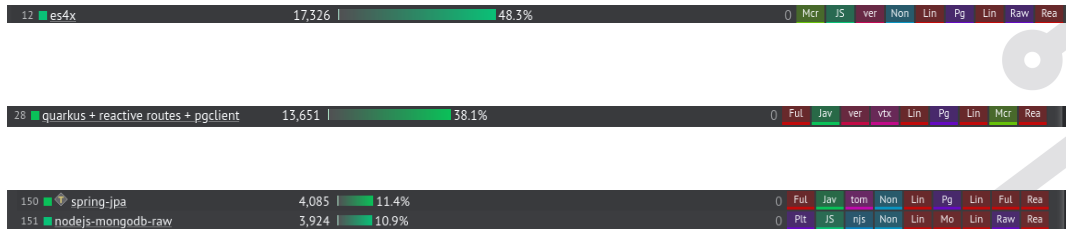
Ventajas

- Compilación AOT
- Menor consumo de memoria
- Menor tiempo de arranque
- Casos útiles: CLI, Aplicaciones de escritorio, Serverless, K8S

Desventajas

- Menor desempeño a largo plazo
- Reflection, dynamic proxies, invoke, bytecode generation
- Muchos frameworks y bibliotecas nunca serán compatibles
- Un buen servidor CI/CD
- A veces threads > procesos (Vert.x)

Threads vs procesos



<https://www.techempower.com/benchmarks/#section=data-r20>

Introduction to Reflectionless: Discover the New Trend in the Java World

Discover this new movement in Java frameworks that aim to circumvent the disuse of reflection to decrease application startup and decrease memory consumption.



by Otavio Santana MVB CORE · Mar. 27, 21 · Java Zone · Tutorial



Like (22)



Comment (4)



Save



Tweet



20.91K Views

Over the last twenty-five years, many things have changed alongside new versions of Java, such as architectural decisions and their requirements. Currently, there is the factor of cloud computing that, in general, requires the application to have a better startup in addition to a low heap of initial memory. It is necessary to redesign the way the frameworks are made, getting rid of the bottleneck with reflection. The purpose of this article is to present some of the solutions that help reflectionless, the trade-offs of that choice, in addition to presenting the Java Annotation Processor.

<https://dzone.com/articles/introduction-to-reflectionless-know-what-the-new-t>



- vorozco@nabenik.com
- @tuxtor
- <http://vorozeo.com>
- <http://tuxtor.shekalug.org>



This work is licensed under
Creative Commons Attribution-
NonCommercial-ShareAlike 3.0
Guatemala (CC BY-NC-SA 3.0 GT).